aws

用户指南

AWS IoT Analytics

AWS IoT Analytics: 用户指南

Table of Contents

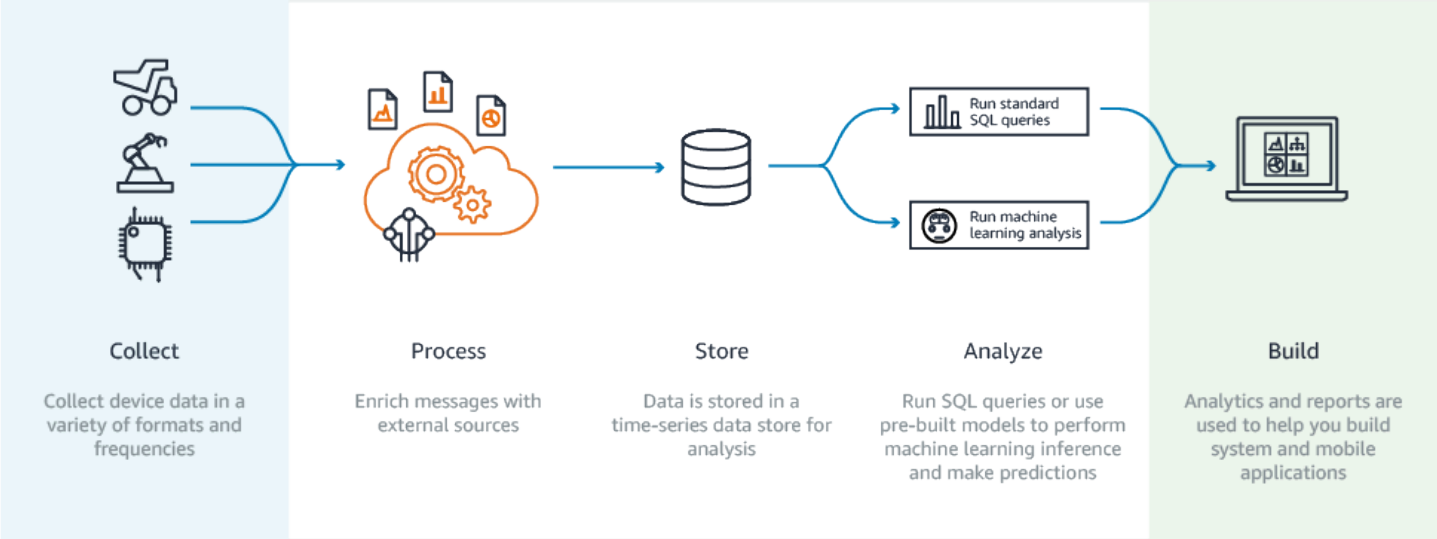
什么是 AWS IoT Analytics？

AWS IoT Analytics 自动执行分析来自 IoT 设备的数据所需的步骤。AWS IoT Analytics 对 IoT 数据进行筛选、转换和丰富，然后再将其存储在时间序列数据存储中以供分析。您可以将服务设置为只从您的设备中收集所需数据，应用数学转换来处理数据，并使用设备特定的元数据 (例如设备类型和位置) 来丰富数据，然后再存储这些数据。接下来，您可使用内置的 SQL 查询引擎通过运行查询来分析数据，也可执行更复杂的分析和机器学习。AWS IoT Analytics 通过与 Jupyter Notebook 集成，实现高级数据探索。AWS IoT Analytics 还通过与 Amazon QuickSight 集成，实现数据可视化。可以在以下区域中使用 Amazon QuickSight。

传统分析和商业智能工具设计用于处理结构化数据。原始 IoT 数据通常来自记录结构化程度较低的数据（如温度、动作或声音）的设备。因而来自这些设备的数据可能具有大量空白、损坏的消息和错误的读数，必须先清除这些，才能进行分析。另外，IoT 数据通常仅在来自外部来源的其他数据中才有意义。通过 AWS IoT Analytics，您可以解决这些问题，收集大量设备数据，以及处理和存储消息。然后，您可以对数据进行查询和分析。AWS IoT Analytics 包括适用于常见 IoT 使用案例的预构建模型，因此您可以回答一些问题，如哪些设备即将出现故障或哪些客户有放弃其可穿戴设备的风险。

如何使用 AWS IoT Analytics

下图概述了如何使用 AWS IoT Analytics。



主要特征

收集

- 与 AWS IoT Core 集成 – AWS IoT Analytics 与 AWS IoT Core 完全集成，因此可接收连接设备的消息流。
- 使用批处理 API 从任何源添加数据 – AWS IoT Analytics 可以通过 HTTP 从任意源接收数据。这意味着连接到 Internet 的任意设备或服务可以将数据发送到 AWS IoT Analytics。有关更多信息，请参阅《AWS IoT Analytics API 参考》中的 BatchPutMessage。
- 仅收集要存储和分析的数据–您可以使用 AWS IoT Analytics 控制台将 AWS IoT Analytics 配置为通过 MQTT 主题筛选条件，以各种格式和频率从设备接收消息。AWS IoT Analytics 验证数据是否在您定义的特定参数范围内，并创建通道。然后，服务将通道路由到适当的管道来对消息进行处理、转换和丰富。

过程

- 清理和筛选 – 使用 AWS IoT Analytics，您定义可在 AWS IoT Analytics 检测到缺少的数据时触发的 AWS Lambda 函数，因此可以运行代码来估算并填补空白。您还可以定义最大值和最小值筛选条件和百分位数阈值来删除数据中的异常值。
- 转换 – AWS IoT Analytics 可以使用您定义的数学或条件逻辑来转换消息，以便您可以执行常见计算，例如摄氏温度到华氏温度的换算。
- 丰富 – AWS IoT Analytics 可以使用外部数据源 (如天气预报) 来丰富数据，然后将数据路由到 AWS IoT Analytics 数据存储。

存储

- 时间序列数据存储 - AWS IoT Analytics 将设备数据存储在优化的时间序列数据存储中，以更快地进行检索和分析。您还可以管理访问权限，实施数据保留策略以及将数据导出到外部访问点。
- 存储处理的数据和原始数据 - AWS IoT Analytics 存储处理的数据，并且还会自动存储提取的原始数据，以便您以后对其进行处理。

分析

- 运行临时 SQL 查询 - AWS IoT Analytics 提供了 SQL 查询引擎，因此您可以运行临时查询并快速获得结果。通过使用该服务，您可以使用标准 SQL 查询从数据存储中提取数据以回答一些问题，如互联车队的平均行驶距离，或者在晚上 7 点以后将智能建筑中的多少个门锁上。这些查询可重复使用，即使互连设备、队列大小和分析要求发生了更改也是如此。
- 时间序列分析 - AWS IoT Analytics 支持时间序列分析，因此您可以分析设备随时间推移的性能并了解设备的使用方式和使用地点，持续监控设备数据来预测维护问题，以及监控传感器来预测环境条件并相应地作出反应。

- 用于复杂分析和机器学习的托管笔记本 - AWS IoT Analytics 支持托管在 Jupyter Notebook 中的笔记本，用于统计分析和机器学习。该服务包括一组笔记本模板，其中包含 AWS 编写的机器学习模型和可视化效果。您可以通过模板开始使用与以下内容相关的 IoT 使用案例：设备故障分析、预测事件 (如可能表示客户将放弃产品的低使用率) 或者按客户使用水平 (如大量用户、周末用户) 或设备运行状况细分设备。在创作笔记本之后，可将其容器化并按指定计划执行它。有关更多信息，请参阅自动化工作流。
- 预测 - 您可以通过称为逻辑回归的方法进行统计分类。您还可以使用长短期记忆 (LSTM)，这是一种强大的神经网络技术，用于预测随时间变化的过程的输出或状态。预构建的笔记本模板还支持用于设备细分的 K-means 集群算法，它将您的设备划分为类似设备组。这些模板通常用于剖析设备运行状况和设备状态，如巧克力工厂中的 HVAC 装置的运行状况或风力涡轮机叶片的磨损状态。同样，这些笔记本模板也可以容器化并按计划执行。

构建和可视化

- QuickSight 集成 - AWS IoT Analytics 提供了与 Amazon QuickSight 的连接器，因此可以在 QuickSight 控制面板中可视化您的数据集。
- 控制台集成 - 您还可以在 AWS IoT Analytics 控制台上的嵌入式 Jupyter Notebook 中可视化结果或临时分析。

AWS IoT Analytics 组件和概念

渠道

通道收集来自 MQTT 主题的数据，并在将数据发布到管道之前将原始未处理消息归档。您也可以使用 BatchPutMessage API 直接将消息发送到通道。未处理的消息存储在您或 AWS IoT Analytics 管理的 Amazon Simple Storage Service (Amazon S3)存储桶中。

管道运输

管道使用来自通道的消息，并允许您在将消息存储在数据存储之前处理消息。名为活动（管道活动）的处理步骤对您的消息执行转换，例如，删除、重命名或添加消息属性，根据属性值筛选消息，为消息调用 Lambda 函数以进行高级处理，或执行数学转换以规范化设备数据。

数据存储

管道将它们处理过的消息存储在数据存储中。数据存储不是数据库，但它是一个可扩展且可查询的消息存储库。对于来自不同设备或位置的消息，您可以有多个数据存储，或者根据您的管道配置和要求，通过消息属性进行筛选。与未处理的通道消息一样，数据存储的处理消息存储在您或 AWS IoT Analytics 管理的 Amazon S3 存储桶。

数据集

您可通过创建数据集从数据存储中检索数据。通过 AWS IoT Analytics，您可以创建 SQL 数据集或容器数据集。

在拥有数据集之后，您可以通过使用 Amazon QuickSight 进行集成来探索数据并获得见解。您也可以通过与 Jupyter Notebook 的集成来执行更高级的分析函数。Jupyter Notebook 提供强大的数据科学工具，可以执行机器学习和一系列统计分析。有关更多信息，请参阅笔记本模板。

您可以将数据集内容发送到 Amazon S3 存储桶，从而允许与现有数据湖集成在一起，或者从内部应用程序和可视化工具中进行访问。您还可以将数据集内容作为输入发送到 AWS IoT Events，该服务允许您监控设备或进程故障或操作更改，并在发生此类事件时触发其他操作。

SQL 数据集

SQL 数据集类似于 SQL 数据库的具体化视图。您可以通过应用 SQL 操作来创建 SQL 数据集。SQL 数据集可以通过指定触发器，按定期计划自动生成。

容器数据集

通过容器数据集，您可以自动运行分析工具和生成结果。有关更多信息，请参阅自动化工作流。它将作为输入的 SQL 数据集、Docker 容器及分析工具和所需库文件、输入和输出变量以及可选的调度触发器组合到一起。输入和输出变量告知可执行映像获取数据和存储结果的位置。触发器可以在 SQL 数据集完成创建其内容时或者按照时间调度表达式来运行分析。容器数据集自动运行，生成并随后保存分析工具的结果。

触发器

您可以通过指定触发器来自动创建数据集。触发器可以是时间间隔（例如，每隔两小时创建该数据集）或在创建另一个数据集的内容时（例如，在 `myOtherDataset` 完成创建其内容时创建该数据集）。或者，您可以使用 CreateDatasetContent API 手动生成数据集内容。

Docker 容器

您可以创建自己的 Docker 容器以打包您的分析工具，或使用 SageMaker 提供的选项。有关更多信息，请参阅Docker 容器。您可以创建自己的 Docker 容器以打包您的分析工具，或使用 SageMaker 提供的选项。您可以将容器存储在您指定的 Amazon ECR 注册表中，以便可以将其安装在所需的平台上。Docker 容器能够运行您通过 Matlab、Octave、Wise.io、SPSS、R、Fortran、Python、Scala、Java、C++ 等准备的自定义分析代码。有关更多信息，请参阅容器化笔记本。

增量时段

增量时段是一系列用户定义的不重叠且连续的时间间隔。通过使用增量时段，您可以使用在上次分析后到达数据存储的新数据创建数据集内容，并对新数据执行分析。您可在数据集 queryAction 的 filters 部分中，通过设置 deltaTime 来创建增量时段。有关更多信息，请参阅 CreateDataset API。通常，您还希望设置时间间隔触发器以自动创建数据集内容 (triggers:schedule:expression)。基本上，这使您可以筛选在特定时间窗口中到达的消息，这样来自以前时间窗口的消息中包含的数据不会重复计数。有关更多信息，请参阅示例 6 - 创建具有增量时段的 SQL 数据集 (CLI)。

访问 AWS IoT Analytics

作为 AWS IoT 的一部分，AWS IoT Analytics 提供以下界面以允许您的设备生成数据，并允许您的应用程序与它们生成的数据进行交互：

AWS Command Line Interface (AWS CLI)

在 Windows、OS X 和 Linux 上运行适用于 AWS IoT Analytics 的命令。您可以使用这些命令创建并管理事物、证书、规则和策略。要开始使用，请参阅 AWS Command Line Interface 用户指南。有关 AWS IoT 命令的更多信息，请参阅 AWS Command Line Interface 参考中的 iot。

> **Important**
>
> 使用 aws iotanalytics 命令与 AWS IoT Analytics 交互。使用 aws iot 命令与 IoT 系统的其他部分交互。

AWS IoT API

使用 HTTP 或 HTTPS 请求构建您的 IoT 应用程序。您可以使用这些 API 操作创建和管理事物、证书、规则及策略。有关更多信息，请参阅《 AWS IoTAPI 参考》中的操作。

AWS 软件开发工具包

使用语言特定 API 构建您的AWS IoT Analytics 应用程序。这些 SDK 中封装了 HTTP 和 HTTPS API，并且您可以用任何受支持的语言进行编程。有关更多信息，请参阅 AWS 软件开发工具包和工具。

AWS IoT Device SDK

构建在您的设备上运行的应用程序，以便向 AWS IoT Analytics 发送消息。有关更多信息，请参阅 AWS IoT 软件开发工具包。

AWS IoT Analytics 控制台

您可以通过构建组件在 AWS IoT Analytics 控制台中可视化结果。

使用案例

预测性维护

AWS IoT Analytics 提供模板来构建预测性维护模型并将其应用于您的设备。例如，您可以使用 AWS IoT Analytics 预测互联货车上的加热和冷却系统何时可能发生故障，从而重新安排车辆路线以防止货物损坏。或者，汽车制造商可以检测哪些客户的刹车片已磨损并通知他们维修车辆。

主动补充供应品

使用 AWS IoT Analytics，可以构建可实时监控库存的 IoT 应用程序。例如，食品和饮料公司可以分析食品自动售货机中的数据并在供应不足时主动再订购商品。

过程效率评分

使用 AWS IoT Analytics，可以构建不断监控不同过程的效率并采取操作来改进过程的 IoT 应用程序。例如，矿业公司可以通过最大化每次行程的装载量来提高其运矿卡车的效率。使用 AWS IoT Analytics，公司可以确定位置或卡车随时间推移的最高效装载量，然后实时比较与目标装载量的任何偏差，并更好地规划装载指南以提高效率。

智能农业

AWS IoT Analytics 可以使用 AWS IoT 注册表数据或公共数据源来通过上下文元数据丰富 IoT 设备数据，以便您的分析可以包括时间、位置、温度、海拔及其他环境条件。通过该分析，您可以编写模型来输出建议设备在现场执行的操作。例如，要确定何时浇水，灌溉系统可使用降雨量数据来丰富湿度传感器数据，从而更高效地使用水资源。

入门 AWS IoT Analytics （控制台）

使用本教程创建所需的 AWS IoT Analytics 资源（也称为组件），以发现有关物联网设备数据的有用见解。

> **注意**
>
> - 如果您在以下教程中输入大写字符，则 AWS IoT Analytics 会自动将其更改为小写字符。
> - AWS IoT Analytics 控制台具有一键入门功能，用于创建频道、管道、数据存储和数据集。登录 AWS IoT Analytics 控制台后即可找到此功能。
>   - 本教程将引导您完成创建 AWS IoT Analytics 资源的每个步骤。

按照以下说明创建 AWS IoT Analytics 频道、管道、数据存储和数据集。本教程还向您展示了如何使用 AWS IoT Core 控制台发送要采集到 AWS IoT Analytics的消息。

主题

登录到控制 AWS IoT Analytics 台

要开始使用，您必须拥有一个 AWS 帐户。如果你已经有一个 AWS 账户，请导航到 https://console.aws.amazon.com/iotanalytics/。

如果您没有 AWS 帐户，请按照以下步骤创建一个。

创建 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。

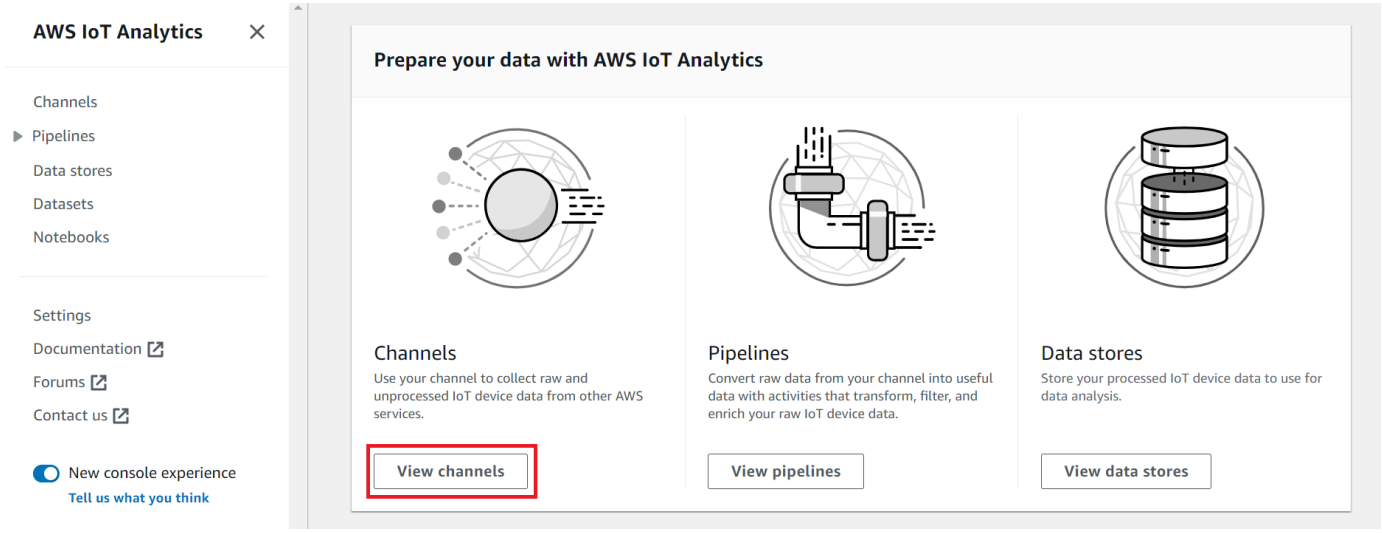
   当您注册时 AWS 账户，就会创建AWS 账户根用户一个。根用户有权访问该账户中的所有 AWS 服务 和资源。作为安全最佳实践，请为用户分配管理访问权限，并且只使用根用户来执行需要根用户访问权限的任务。
3. 登录 AWS Management Console 并导航至 https://console.aws.amazon.com/iotanalytics/。

创建通道

通道收集并存档未处理的、非结构化的原始 IoT 设备数据。按照以下步骤创建您的通道。

创建通道

1. 在 https://console.aws.amazon.com/iotanalytics/ 的使用 AWS IoT Analytics准备数据部分中，选择查看通道。



> **Tip**
>
> 也可以从导航窗格中选择 通道。

2. 在 Channels (通道) 页面上，选择 Create channel (创建通道)。
3. 在指定通道详情页面上，输入有关通道的详细信息。

   a. 输入唯一且易于识别的通道名称。

   b. （可选）在标签中，将一个或多个自定义标签（键值对）添加到通道中。标签有助于标识您为AWS IoT Analytics创建的资源。

   c. 选择下一步。
4. AWS IoT Analytics 将未经处理的原始物联网设备数据存储在亚马逊简单存储服务 (Amazon S3) 存储桶中。您可以选择自己的 Amazon S3 存储桶，您可以访问和管理该存储桶， AWS IoT Analytics 也可以为您管理 Amazon S3 存储桶。

   a. 在本教程中，对于存储类型，选择服务托管存储。

   b. 对于选择原始数据的存储时间，选择无限期。

   c. 选择下一步。
5. 在配置源页面上，输入 AWS IoT Analytics 要从中收集消息数据的信息 AWS IoT Core。

   a. 输入 AWS IoT Core 主题筛选条件，例如`update/environment/dht1`。在本教程的后面部分，您将使用此主题筛选条件向通道发送消息数据。

   b. 在 IAM 角色 区域中，选择 新建。在 创建新角色窗口中，输入角色的 名称，然后选择 创建角色。这会自动创建附加了相应策略的角色。

   c. 选择下一步。
6. 检查您的选择，然后选择创建渠道。
7. 确认您的新通道显示在通道页面上。

创建数据存储

数据存储接收并存储您的消息数据。数据存储不是数据库。相反，数据存储是 Amazon S3 存储桶中可扩展且可查询的存储库。您可以为来自不同设备或位置的消息使用多个数据存储。或者，您可以根据管道配置和要求筛选消息数据。

按照以下步骤创建数据存储。

创建数据存储

1. 在 https://console.aws.amazon.com/iotanalytics/ 的使用 AWS IoT Analytics准备数据部分，选择查看数据存储。

2. 在 数据存储 页面上，选择 创建数据存储。
3. 在指定数据存储详细信息页面上，输入数据存储相关的基本信息。

   a. 在数据存储 ID 中，输入唯一的数据存储 ID。该 ID 在创建后无法更改。

   b. （可选）对于标签，选择添加新标签，将一个或多个自定义标签（键值对）添加到数据存储中。标签有助于标识您为 AWS IoT Analytics创建的资源。

   c. 选择 下一步。
4. 在配置存储类型页面上，指定如何存储数据。

   a. 对于存储类型，选择服务托管存储。

   b. 对于配置要保留已处理数据的时间，选择无限期。

   c. 选择 下一步。
5. AWS IoT Analytics 数据存储支持 JSON 和 Parquet 文件格式。对于数据存储数据格式，请选择 JSON 或 Parquet。有关 AWS IoT Analytics 支持的文件类型的更多信息，请参阅 文件格式。

   选择下一步。
6. （可选）AWS IoT Analytics 支持数据存储中的自定义分区，因此您可以查询已修剪的数据以缩短延迟。有关支持的自定义分区的更多信息，请参阅自定义分区。

   选择 下一步。
7. 检查您的选择，然后选择 创建数据存储。
8. 确认新数据存储显示在数据存储页面上。

创建管道

您必须创建一个管道，将通道连接到数据存储。基本管道仅指定收集数据的通道，并标识消息发送到的数据存储。有关更多信息，请参阅管道活动。

在本教程中，您将创建一个仅将通道连接到数据存储的管道。以后，您可以引入管道活动来处理此数据。

执行以下步骤创建管道。

要创建管道

1. 在 https://console.aws.amazon.com/iotanalytics/ 的使用 AWS IoT Analytics准备数据部分，选择查看管道。

> **Tip**
>
> 也可以从导航窗格中选择 管道。

2. 在 管道 页面上，选择 创建管道。
3. 输入有关管道的详细信息。
   a. 在设置管道 ID 和源中，输入管道名称。
   b. 选择管道的来源，这是您的管道将从中读取消息的 AWS IoT Analytics 渠道。
   c. 指定管道的输出，即存储已处理消息数据的数据存储。
   d. （可选）在标签中，将一个或多个自定义标签（键值对）添加到管道中。
   e. 在推断消息属性页面上，输入属性名称和示例值，从列表中选择数据类型，然后选择添加属性。
   f. 对所需数量的属性重复上一步操作，然后选择下一步。
   g. 现在您不会添加任何管道活动。在丰富、转换和筛选消息页面上，选择下一步。
4. 检查您的选择，然后选择 创建管道。
5. 确认您的新管道显示在管道页面上。

> **Note**
>
> 您创建 AWS IoT Analytics 资源是为了让它们可以执行以下操作：
>
> - 使用通道收集未处理的原始 IoT 设备消息数据。
> - 将您的 IoT 设备消息数据存储在数据存储中。
> - 使用管道清理、筛选、转换和丰富数据。
>
> 接下来，您将创建一个 AWS IoT Analytics SQL 数据集，以发现有关您的物联网设备的有用见解。

创建数据集

> **Note**
>
> 数据集通常是数据的集合，这些数据可能以表格形式呈现，也可能不以表格形式呈现。相比之下，通过对数据存储中的数据应用 SQL 查询来 AWS IoT Analytics 创建数据集。

您现在具有一个通道以将原始消息数据路由到管道，该管道将数据存储在数据存储以对其进行查询。要查询数据，您需要创建一个数据集。数据集包含用于查询数据存储的 SQL 语句和表达式，以及在您指定的日期和时间重复该查询的可选计划。您可以使用类似于 Amazon CloudWatch 计划表达式的表达式来创建可选计划。

若要创建数据集

1. 在 https://console.aws.amazon.com/iotanalytics/ 的左侧导航窗格中，选择数据集。
2. 在创建数据集页面上，选择创建 SQL。
3. 在指定数据集详细信息页面上，指定数据集的详细信息。
   a. 输入数据集的名称。
   b. 对于数据存储源，选择用于标识先前所创建数据存储的唯一 ID。
   c. （可选）在标签中，将一个或多个自定义标签（键值对）添加到数据集中。
4. 使用 SQL 表达式查询数据并回答分析问题。您的查询结果存储在此数据集中。
   a. 在作者查询字段中，输入使用通配符显示最多五行数据的 SQL 查询。

      ```
      SELECT * FROM my_data_store LIMIT 5
      ```

      有关中支持的 SQL 功能的更多信息 AWS IoT Analytics，请参见AWS IoT Analytics 中的 SQL 表达式。
   b. 您可以选择测试查询来验证您的输入是否正确，并在查询后以表格显示结果。

      > **Note**
      >
      > - 在本教程中，此时您的数据存储可能为空。在空数据存储上运行 SQL 查询不会返回结果，因此您可能只会看到 `__dt`。

- 您必须小心地将 SQL 查询限制为合理的大小以免长时间运行，因为 Athena 限制了运行的最大查询数。因此，必须小心地将 SQL 查询限制为合理的大小。

  我们建议测试期间在查询中使用 `LIMIT` 子句。测试成功后，您可以删除此子句。

5. （可选）使用指定时间范围内的数据创建数据集内容时，某些数据可能无法及时送达处理。要允许延迟，可指定偏移量或增量。有关更多信息，请参阅 通过 Amazon CloudWatch Events 获取延迟数据通知。

   此时无需配置数据选择筛选条件。在配置数据选择筛选条件页面上，选择下一步。

6. （可选）您可以安排定期运行此查询以刷新数据集。您可随时创建和编辑数据集计划。

   此时，您不会计划定期运行查询，因此，在 设置查询计划 页面上选择 下一步。

7. AWS IoT Analytics 将创建此数据集内容的版本并存储指定时间段内的分析结果。我们建议 90 天，但您可以选择设置自定义保留策略。您也可限制数据集内容存储版本的数量。

   您可以将默认数据集保留期限设置为无限期，并禁用版本控制。在 配置分析结果 页面上，选择 下一步。

8. （可选）您可通过配置数据集结果传送规则，将其传送到特定目的地，例如 AWS IoT Events。

   您不会在本教程的其他地方传送结果，因此在配置数据集内容传送规则页面上，选择下一步。

9. 检查您的选择，然后选择创建数据集。

10. 确认您的新数据集显示在数据集页面上。

使用发送消息数据 AWS IoT

如果您具有一个通道以将数据路由到管道，该管道将数据存储在数据存储以在其中对其进行查询，则可以将 IoT 设备数据发送到 AWS IoT Analytics。您可以使用以下选项 AWS IoT Analytics 将数据发送到：

- 使用 AWS IoT 消息代理。
- 使用 AWS IoT Analytics BatchPutMessage API 操作。

在以下步骤中，您将从 AWS IoT Core 控制台中的 AWS IoT 消息代理发送消息数据，以便消息代理 AWS IoT Analytics 可以提取这些数据。

> **Note**
>
> 在为消息创建主题名称时，请注意以下几点：
>
> - 主题名称不区分大小写。同一负载中名为 `example` 和 `EXAMPLE` 的字段被视为重复字段。
> - 主题名称不能以 `$` 字符开头。以 `$` 开头的主题均为只能由 AWS IoT使用的预留主题。
> - 请勿在主题名称中包含个人身份信息，因为这些信息可能会出现在未加密的通信和报告中。
> - AWS IoT Core 无法在 AWS 账户或 AWS 地区之间发送消息。

使用发送消息数据 AWS IoT

1. 登录 AWS IoT 控制台。
2. 在导航窗格中选择测试，然后选择MQTT 测试客户端 。
3. 在 MQTT 测试客户端 中，选择 发布到主题。
4. 对于主题名称，请输入与通道创建时输入的主题筛选条件相匹配的名称。此示例使用 `update/environment/dht1`。
5. 在消息负载部分中，输入以下 JSON 内容。

```
{
  "thingid": "dht1",
  "temperature": 26,
  "humidity": 29,
  "datetime": "2018-01-26T07:06:01"
}
```

6. （可选）选择添加配置以获取其他消息协议选项。
7. 选择 发布。

   这会发布一条由您的通道捕获的消息。然后，您的管道会将消息路由到数据存储。

查看 AWS IoT 消息的进度

您可以按照以下步骤检查消息是否提取到您的通道中。

查看 AWS IoT 消息的进度

1. 登录 https://console.aws.amazon.com/iotanalytics/。

2. 在导航窗格中，选择通道，然后选择您此前创建的通道名称。
3. 在通道的详细信息页面上，向下滚动到监控部分，然后调整显示的时间范围 (1h 3h 12h 1d 3d 1w)。选择一个值（例如 1w）以查看上周的数据。

您可以使用类似的功能在管道的详细信息页面上监控管道活动运行时和错误。在本教程中，您尚未将活动指定为管道的一部分，因此不应看到任何运行时错误。

监控管道活动

1. 在导航窗格中，选择管道，然后选择先前所创建管道的名称。
2. 在管道的详细信息页面上，向下滚动到监控部分，然后通过选择一个时间范围指示符 (1h 3h 12h 1d 3d 1w) 来调整显示的时间范围。

访问查询结果

数据集内容是 CSV 格式的文件，其中包含查询结果。

1. 在 https://console.aws.amazon.com/iotanalytics/ 的左侧导航窗格中，选择数据集。
2. 在 数据集 页面上，选择您以前创建的数据集的名称。
3. 在数据集信息页面上，在右上角选择立即运行。
4. 要检查数据集是否已准备就绪，请在数据集下方查看一条类似于您已成功启动数据集查询的消息。数据集内容选项卡包含查询结果并显示已成功。
5. 要预览成功查询的结果，请在数据集内容选项卡上，选择查询名称。选择 下载 来查看或保存包含查询结果的 CSV 文件。

> **Note**
>
> AWS IoT Analytics 可以将 Jupyter 笔记本的 HTML 部分嵌入到数据集内容页面上。有关更多信息，请参阅 使用控制台可视化 AWS IoT Analytics 数据。

探索您的数据

您可以通过多种方法存储、分析和可视化数据。

Amazon Simple Storage Service

您可以将数据集内容发送到 Amazon S3 存储桶，从而允许与现有数据湖集成在一起，或者从内部应用程序和可视化工具中进行访问。参见CreateDataset操作`contentDeliveryRules::destination::s3DestinationConfiguration`中的字段。

AWS IoT Events

您可以将数据集内容作为输入发送到该服务 AWS IoT Events，该服务使您能够监控设备或进程的故障或操作变化，并在此类事件发生时启动其他操作。

为此，请使用CreateDataset操作创建数据集并在字段中指定 AWS IoT Events 输入`contentDeliveryRules :: destination :: iotEventsDestinationConfiguration :: inputName`。您还必须指定角色的`roleArn`，该角色授予运行 AWS IoT Analytics 权限`iotevents:BatchPutMessage`。每当创建数据集内容时，都会 AWS IoT Analytics 将每个数据集内容条目作为消息发送到指定的 AWS IoT Events 输入。例如，如果您的数据集包含以下内容。

```
"what","who","dt"
"overflow","sensor01","2019-09-16 09:04:00.000"
"overflow","sensor02","2019-09-16 09:07:00.000"
"underflow","sensor01","2019-09-16 11:09:00.000"
...
```

然后 AWS IoT Analytics 发送包含以下字段的消息。

```
{ "what": "overflow", "who": "sensor01", "dt": "2019-09-16 09:04:00.000" }
```

```
{ "what": "overflow", "who": "sensor02", "dt": "2019-09-16 09:07:00.000" }
```

您需要创建一个可识别您感兴趣的字段（`what`、`who`、`dt`中的一个或多个）的 AWS IoT Events 输入，并创建一个 AWS IoT Events 检测器模型，该模型在事件中使用这些输入字段来触发操作或设置内部变量。

Jupyter Notebook

Jupyter Notebook 是使用脚本语言运行临时数据探索和高级分析的开源解决方案。您可以深入研究并应用更复杂的分析，并对 IoT 设备数据使用机器学习方法，例如用于预测的 k 均值集群和回归模型。

AWS IoT Analytics 使用亚马逊 SageMaker 笔记本实例托管其 Jupyter 笔记本电脑。在创建笔记本实例之前，您必须在 AWS IoT Analytics 和 Amazon 之间创建关系 SageMaker：

1. 导航到SageMaker 控制台并创建笔记本实例：
   a. 填写详细信息，然后选择 Create a new role (创建新角色)。记录角色 ARN。
   b. 创建笔记本实例。
2. 前往 I AM 控制台并修改 SageMaker角色：
   a. 打开角色。它应该具有一个托管策略。
   b. 选择添加内联策略，对于服务，选择 iotAnalytics。选择 选择操作，然后在搜索框中键入 **`GetDatasetContent`**并选择它。选择 Review Policy（查看策略）。
   c. 检查策略的准确性，为其输入名称，然后选择 创建策略。

这为新创建的角色提供了从中读取数据集的权限 AWS IoT Analytics。

1. 返回 https://console.aws.amazon.com/iotanalytics/，然后在左侧导航窗格中选择笔记本。在笔记本页面上，选择创建笔记本。
2. 在选择模板页面上，选择 IoTA 空白模板。
3. 在 设置笔记本 页面上，为笔记本输入名称。在选择数据集源中，选中，然后选择您之前创建的数据集。在选择笔记本实例中，选择您在中创建的笔记本实例 SageMaker。
4. 查看您的选择后，选择创建笔记本。
5. 在笔记本页面上，您的笔记本实例将在亚马逊 SageMaker控制台中打开。

笔记本模板

AWS IoT Analytics 笔记本模板包含 AWS 创作的机器学习模型和可视化效果，可帮助您开始 AWS IoT Analytics 使用案例。您可以使用这些笔记本模板来了解更多信息，也可以重复使用它们来适应 IoT 设备数据并提供即时值。

您可以在 AWS IoT Analytics 控制台中找到以下笔记本模板：

- 检测情境异常 – 使用泊松指数加权移动平均 (PEWMA) 模型，在测得的风速中应用情境异常检测。
- 太阳能电池板输出预测 – 应用分段、季节性、线性时间序列模型来预测太阳能电池板的输出。
- 喷气发动机的预测性维护 – 应用多变量长短期记忆 (LSTM) 神经网络和逻辑回归来预测喷气发动机故障。

- 智能家居客户细分 – 应用 K 均值和主成分分析法 (PCA) 分析，从智能家居使用数据中发现不同客户群。
- 智能城市拥堵预测 – 应用 LSTM 预测城市主干道的利用率。
- 智能城市空气质量预测 – 应用 LSTM 预测城市中心区的颗粒物污染情况。

AWS IoT Analytics 入门

本节讨论了通过 AWS IoT Analytics 收集、存储、处理和查询设备数据时使用的基本命令。此处显示的示例使用 AWS Command Line Interface (AWS CLI)。有关 AWS CLI 的更多信息，请参阅 AWS Command Line Interface 用户指南。有关 AWS IoT CLI 命令的更多信息，请参阅 AWS Command Line Interface 参考中的 iot。

> **Important**
>
> 使用 AWS CLI，通过 `aws iotanalytics` 命令与 AWS IoT Analytics 进行交互。使用AWS CLI，通过 `aws iot`命令 与 IoT 系统的其他部分交互。

> **Note**
>
> 请注意，在后面的示例中输入 AWS IoT Analytics实体（通道、数据集、数据存储和管道）的名称时，系统自动将您使用的任何大写字母更改为小写字母。实体的名称必须以小写字母开头，并且只能包含小写字母、下划线和数字。

创建通道

通道收集并存档未处理的原始消息数据，然后再将该数据发布到管道。传入消息将发送到通道，因此，第一步是为数据创建通道：

```
aws iotanalytics create-channel --channel-name mychannel
```

如果要将 AWS IoT 消息提取到 AWS IoT Analytics 中，您可以创建一个AWS IoT 规则引擎规则以将消息发送到该通道。这将在后面的 将数据提取到 AWS IoT Analytics 中显示。要将数据传送到通道，另一种方法是使用 AWS IoT Analytics命令`BatchPutMessage`。

列出您已经创建的通道：

```
aws iotanalytics list-channels
```

获取有关通道的更多信息。

```
aws iotanalytics describe-channel --channel-name mychannel
```

未处理的通道消息存储在 AWS IoT Analytics 管理的 Amazon S3 存储桶中，或存储在您管理的存储桶中。可以使用 `channelStorage` 参数指定相应的存储桶。默认设置是服务管理的 Amazon S3 存储桶。如果您选择将通道消息存储在您管理的 Amazon S3 存储桶中，则必须为 AWS IoT Analytics 授予权限以代表您对 Amazon S3 存储桶执行以下操作：`s3:GetBucketLocation`（验证存储桶位置）、`s3:PutObject`（存储）、`s3:GetObject`（读取）和 `s3:ListBucket`（重新处理）。

Example

```
{
    "Version": "2012-10-17",
    "Id": "MyPolicyID",
    "Statement": [
        {
            "Sid": "MyStatementSid",
            "Effect": "Allow",
            "Principal": {
                "Service": "iotanalytics.amazonaws.com"
            },
            "Action": [
                "s3:GetObject",
                "s3:GetBucketLocation",
                "s3:ListBucket",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::my-iot-analytics-bucket",
                "arn:aws:s3:::my-iot-analytics-bucket/*"
            ]
        }
    ]
}
```

如果更改客户管理的通道存储的选项或权限，您可能需要重新处理通道数据，以确保以前提取的数据包含在数据集内容中。请参阅重新处理通道数据。

创建数据存储

数据存储接收并存储您的消息。数据存储不是数据库，但它是一个可扩展且可查询的消息存储库。您可创建多个数据存储来存储来自不同设备或位置的消息，也可使用单个数据存储来接收所有 AWS IoT 消息。

```
aws iotanalytics create-datastore --datastore-name mydatastore
```

列出您已经创建的数据存储：

```
aws iotanalytics list-datastores
```

获取有关数据存储的更多信息。

```
aws iotanalytics describe-datastore --datastore-name mydatastore
```

适用于 AWS IoT Analytics 资源的 Amazon S3 策略

处理过的数据可存储在 AWS IoT Analytics 管理的 Amazon S3 存储桶中，也可存储在您管理的 Amazon S3 存储桶中。创建数据存储时，使用 `datastoreStorage` API 参数选择所需的 Amazon S3 存储桶。默认设置是服务管理的 Amazon S3 存储桶。

如果您选择将数据存储消息存储在您管理的 Amazon S3 存储桶中，则必须为 AWS IoT Analytics 授予权限以代表您对 Amazon S3 存储桶执行以下操作：

- `s3:GetBucketLocation`
- `s3:PutObject`
- `s3:DeleteObject`

如果将数据存储作为 SQL 查询数据集源，须设置 Amazon S3 存储桶策略，以便为 AWS IoT Analytics 授予权限以对存储桶内容调用 Amazon Athena 查询。

> **Note**
>
> 我们建议在存储桶策略中指定 `aws:SourceArn`，以免出现混淆代理安全问题。这通过仅允许来自指定账户的请求来限制访问。有关混淆代理问题的更多信息，请参阅the section called “跨服务混淆代理问题防范”。

以下是授予所需权限的示例存储桶策略。

```
{
    "Version": "2012-10-17",
    "Id": "MyPolicyID",
    "Statement": [
        {
            "Sid": "MyStatementSid",
            "Effect": "Allow",
            "Principal": {
                "Service": "iotanalytics.amazonaws.com"
            },
            "Action": [
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:ListMultipartUploadParts",
                "s3:AbortMultipartUpload",
                "s3:PutObject",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET",
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
            ],
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": [
                        "arn:aws:iotanalytics:us-east-1:123456789012:dataset/DOC-
EXAMPLE-DATASET",
                        "arn:aws:iotanalytics:us-east-1:123456789012:datastore/DOC-
EXAMPLE-DATASTORE"
                    ]
                }
            }
        }
    ]
}
```

有关更多信息，请参阅《Amazon Athena 用户指南》中的“跨账户访问”。

> **Note**
>
> 此外，如果更新客户管理的数据存储的选项或权限，您可能需要重新处理通道数据，以确保以前提取的数据包含在数据集内容中。有关更多信息，请参阅重新处理通道数据。

文件格式

AWS IoT Analytics 数据存储目前支持 JSON 和 Parquet 文件格式。默认文件格式为 JSON。

- JSON (JavaScript 对象表示法) - 一种支持名称-值对和有序值列表的文本格式。
- Apache Parquet - 一种列式存储格式，用于高效存储和查询大量数据。

要配置 AWS IoT Analytics 数据存储的文件格式，可在创建数据存储时使用 `FileFormatConfiguration` 对象。

`fileFormatConfiguration`

包含文件格式的配置信息。 AWS IoT Analytics数据存储支持 JSON 和 Parquet。

默认文件格式为 JSON。只能指定一种格式。创建数据存储后，无法更改文件格式。

`jsonConfiguration`

包含 JSON 格式的配置信息。

`parquetConfiguration`

包含 Parquet 格式的配置信息。

`schemaDefinition`

定义架构所需的信息。

`columns`

指定存储数据的一个或多个列。

每个架构最多可有 100 列。每列最多可有 100 种嵌套类型。

`name`

列的名称。

长度限制：1-255 个字符。

`type`

数据的类型。有关受支持的数据类型的更多信息，请参阅《AWS Glue 开发人员指南》中的常见数据类型。

长度限制：1-131072 个字符。

AWS IoT Analytics 支持 Amazon Athena 数据类型页面上列出的所有数据类型，但 `DECIMAL(precision, scale)` - *precision* 除外。

创建数据存储（控制台）

以下步骤说明如何创建以 Parquet 格式保存数据的数据存储。

创建数据存储

1. 登录 https://console.aws.amazon.com/iotanalytics/。
2. 在导航窗格中，选择数据存储。
3. 在 数据存储 页面上，选择创建数据存储。
4. 在指定数据存储详细信息页面上，输入数据存储相关的基本信息。
   a. 在数据存储 ID 中，输入唯一的数据存储 ID。该 ID 在创建后无法更改。
   b. （可选）对于标签，选择添加新标签，将一个或多个自定义标签（键值对）添加到数据存储中。标签有助于标识为 AWS IoT Analytics 创建的资源。
   c. 选择 下一步。
5. 在配置存储类型页面上，指定如何存储数据。
   a. 对于存储类型，选择服务托管存储。
   b. 对于配置已处理数据要保留多长时间，选择无限期。
   c. 选择 下一步。
6. 在配置数据格式页面上，定义数据记录的结构和格式。
   a. 对于 分类，选择 Parquet。创建数据存储后，无法更改该格式。
   b. 对于推理源，请为您的数据存储选择 JSON 字符串。
   c. 对于字符串，请以 JSON 格式输入您的架构，如下例所示。

```
{
    "device_id": "0001",
    "temperature": 26,
    "humidity": 29,
    "datetime": "2018-01-26T07:06:01"
}
```

d. 选择推断架构。

e. 在配置 Parquet 架构下，确认格式与 JSON 示例相匹配。如果格式不匹配，请手动更新 Parquet 架构。

- 如果希望架构显示更多列，请选择添加新列，输入列名，然后选择数据类型。

> **Note**
>
> 默认情况下，您的架构可以有 100 列。有关更多信息，请参阅 AWS IoT Analytics 配额。

- 您可以更改现有列的数据类型。有关受支持的数据类型的更多信息，请参阅《AWS Glue 开发人员指南》中的常见数据类型。

> **Note**
>
> 创建数据存储后，无法更改现有列的数据类型。

- 要删除现有列，请选择移除列。

f. 选择 下一步。

7. （可选）AWS IoT Analytics 支持数据存储中的自定义分区，以便您可以查询已清理的数据以缩短延迟。有关受支持的自定义分区的更多信息，请参阅自定义分区。

   选择 下一步。

8. 在 查看和创建 页面上，查看您的选择，然后选择 创建数据存储。

> **Important**
>
> 创建数据存储后，无法更改列的数据存储 ID、文件格式或数据类型。

9. 确认新数据存储显示在数据存储页面上。

自定义分区

AWS IoT Analytics 支持数据分区，以便您可以整理数据存储中的数据。使用数据分区来组织数据时，可以查询已清理的数据。这减少了每次查询扫描的数据量并减少了延迟。

您可根据消息数据属性或通过管道活动添加的属性，对数据进行分区。

首先，启用数据存储中的数据分区。指定一个或多个数据分区维度，并将分区后的数据存储连接到 AWS IoT Analytics 管道。然后，编写利用 `WHERE` 子句优化性能的查询。

创建数据存储（控制台）

下面的过程演示如何使用自定义分区来创建数据存储。

创建数据存储

1. 登录到 AWS IoT Analytics 控制台。
2. 在导航窗格中，选择数据存储。
3. 在 数据存储 页面上，选择 创建数据存储。
4. 在指定数据存储详细信息页面上，输入数据存储相关的基本信息。
   a. 在数据存储 ID 中，输入唯一的数据存储 ID。该 ID 在创建后无法更改。
   b. （可选）对于标签，选择添加新标签，将一个或多个自定义标签（键值对）添加到数据存储中。标签有助于标识为 AWS IoT Analytics 创建的资源。
   c. 选择 下一步。
5. 在配置存储类型页面上，指定如何存储数据。
   a. 对于存储类型，选择服务托管存储。
   b. 对于配置已处理数据要保留多长时间，选择无限期。
   c. 选择 下一步。
6. 在配置数据格式页面上，定义数据记录的结构和格式。
   a. 对于数据存储数据格式分类，选择 JSON 或 Parquet。有关 AWS IoT Analytics 支持的文件类型的更多信息，请参阅文件格式。

   > **Note**
   > 创建数据存储后，无法更改该格式。

b. 选择 下一步。

7. 为此数据存储创建自定义分区。

a. 对于添加数据分区，选择启用。

b. 对于数据分区源，请指定分区源相关的基本信息。

选择示例源，然后选择为此数据存储收集消息的 AWS IoT Analytics 通道。

c. 在消息示例属性中，选择要用于对数据存储进行分区的消息属性。接下来，将您的选择添加为操作下的属性分区维度或时间戳分区维度。

> **Note**
>
> 您只能将一个时间戳分区添加到数据存储中。

d. 对于自定义数据存储分区维度，请定义有关分区维度的基本信息。您在上一步中选择的每个消息示例属性都将成为分区的维度。使用以下选项自定义每个维度：

- 分区类型 - 指定此分区维度是属性还是时间戳分区类型。
- 属性名称和维度名称 - 默认情况下，AWS IoT Analytics 将使用您选择的消息样本属性的名称作为属性分区维度的标识符。编辑属性名称以自定义分区维度的名称。您可以使用 `WHERE` 子句中的维度名称来优化查询性能。
  - 任何分区属性维度的名称均带有 `__partition_` 前缀。
  - 对于时间戳分区类型，AWS IoT Analytics 创建了以下四个维度，名称分别为 `__year`、`__month`、`__day`、`__hour`。
- 排序 - 重新排列分区维度以缩短查询的延迟。

对于时间戳格式，请通过匹配消息数据中提取的时间戳来指定时间戳分区的格式。您可从 AWS IoT Analytics 列出的格式选项中选择一个，也可指定一个与您的数据格式相匹配的选项。详细了解如何指定日期时间格式化程序。

要添加不是消息属性的新维度，请选择 添加新分区。

e. 选择 下一步。

8. 在 查看和创建 页面上，查看您的选择，然后选择 创建数据存储。

> **Important**
>
> - 创建数据存储后，无法更改数据存储 ID。
> - 要编辑现有分区，必须创建另一个数据存储并通过管道重新处理数据。

9. 确认新数据存储显示在数据存储页面上。

创建管道

管道使用来自通道的消息，并在将消息存储在数据存储之前允许您处理和筛选消息。要将通道连接到数据存储，请创建一个管道。最简单的管道除了指定收集数据的通道和标识将向其发送消息的数据存储之外，不包含任何其他活动。有关更复杂管道的信息，请参阅管道活动。

在开始时，我们建议您创建一个管道，该管道仅将通道连接到数据存储，而不执行任何其他操作。然后，在确认原始数据流到数据存储后，您可以引入额外的管道活动以处理该数据。

运行以下命令以创建管道。

```
aws iotanalytics create-pipeline --cli-input-json file://mypipeline.json
```

mypipeline.json 文件包含以下内容。

```
{
    "pipelineName": "mypipeline",
    "pipelineActivities": [
        {
            "channel": {
                "name": "mychannelactivity",
                "channelName": "mychannel",
                "next": "mystoreactivity"
            }
        },
        {
            "datastore": {
                "name": "mystoreactivity",
                "datastoreName": "mydatastore"
            }
        }
    ]
```

```
}
```

运行以下命令可列出现有管道。

```
aws iotanalytics list-pipelines
```

运行以下命令可查看单个管道的配置。

```
aws iotanalytics describe-pipeline  --pipeline-name mypipeline
```

将数据提取到 AWS IoT Analytics

如果您具有一个通道以将数据路由到管道，该管道将数据存储在数据存储以在其中对其进行查询，则可以将消息数据发送到 AWS IoT Analytics。此处，我们介绍了两种将数据传送到 AWS IoT Analytics 的方法。您可用 AWS IoT 消息代理或 AWS IoT Analytics `BatchPutMessage` API 发送消息。

主题

- 使用 AWS IoT 消息代理
- 使用 BatchPutMessage API

使用 AWS IoT 消息代理

要使用 AWS IoT消息代理，请使用 AWS IoT 规则引擎创建一个规则。该规则将具有特定主题的消息路由到 AWS IoT Analytics。但首先，该规则要求您创建一个角色以授予所需的权限。

创建 IAM 角色

要将 AWS IoT 消息路由到 AWS IoT Analytics 通道，请设置一个规则。但首先，您必须创建一个 IAM 角色，以便为该规则授予权限以将消息数据发送到 AWS IoT Analytics 通道。

运行以下命令以创建角色。

```
aws iam create-role --role-name myAnalyticsRole --assume-role-policy-document file://
arpd.json
```

`arpd.json`文件的内容应与以下内容类似。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "iot.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

然后，将策略文档附加到该角色：

```
aws iam put-role-policy --role-name myAnalyticsRole --policy-name myAnalyticsPolicy --
policy-document file://pd.json
```

pd.json文件的内容应与以下内容类似。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iotanalytics:BatchPutMessage",
            "Resource": [
               "arn:aws:iotanalytics:us-west-2:your-account-number:channel/mychannel"
            ]
        }
    ]
}
```

创建 AWS IoT 规则

创建一个 AWS IoT规则以将消息发送到您的通道。

```
aws iot create-topic-rule --rule-name analyticsTestRule --topic-rule-payload file://
rule.json
```

rule.json文件的内容应与以下内容类似。

```
{
    "sql": "SELECT * FROM 'iot/test'",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [ {
        "iotAnalytics": {
            "channelName": "mychannel",
            "roleArn": "arn:aws:iam::your-account-number:role/myAnalyticsRole"
        }
    } ]
}
```

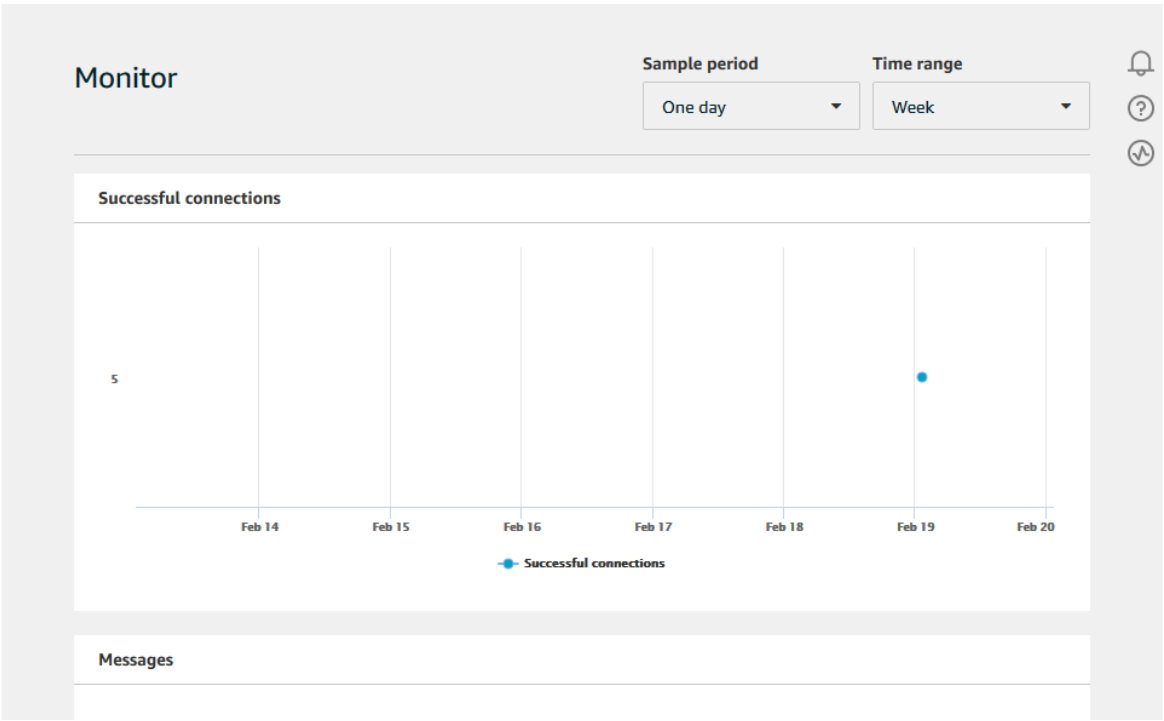
将 `iot/test` 替换为应路由的消息的 MQTT 主题。将通道名称和角色替换为您在前面章节中创建的通道和角色。

将 MQTT 消息发送到 AWS IoT Analytics

在将规则加入通道、将通道加入管道并将管道加入数据存储后，与规则匹配的任何数据现在通过 AWS IoT Analytics 流到数据存储，以准备好在其中进行查询。要测试该内容，您可以使用 AWS IoT 控制台发送消息。

> **Note**
>
> 您发送至 AWS IoT Analytics 的消息负载（数据）的字段名称。
>
> - 必须仅包含字母数字字符和下划线 (_)；不允许使用其他特殊字符。
> - 必须以字母字符或单个下划线 (_) 开头。
> - 不能包含连字符 (-)。
> - 正则表达式术语：`^[A-Za-z_]([A-Za-z0-9]*|[A-Za-z0-9][A-Za-z0-9_]*)$`。
> - 长度不能超过 255 个字符。
> - 不区分大小写。同一负载中名为“foo”和“FOO”的字段被视为重复字段。
>
> 例如，在消息负载中，`{"temp_01": 29}` 或 `{"_temp_01": 29}` 有效，但 `{"temp-01": 29}`、`{"01_temp": 29}` 或 `{"__temp_01": 29}` 无效。

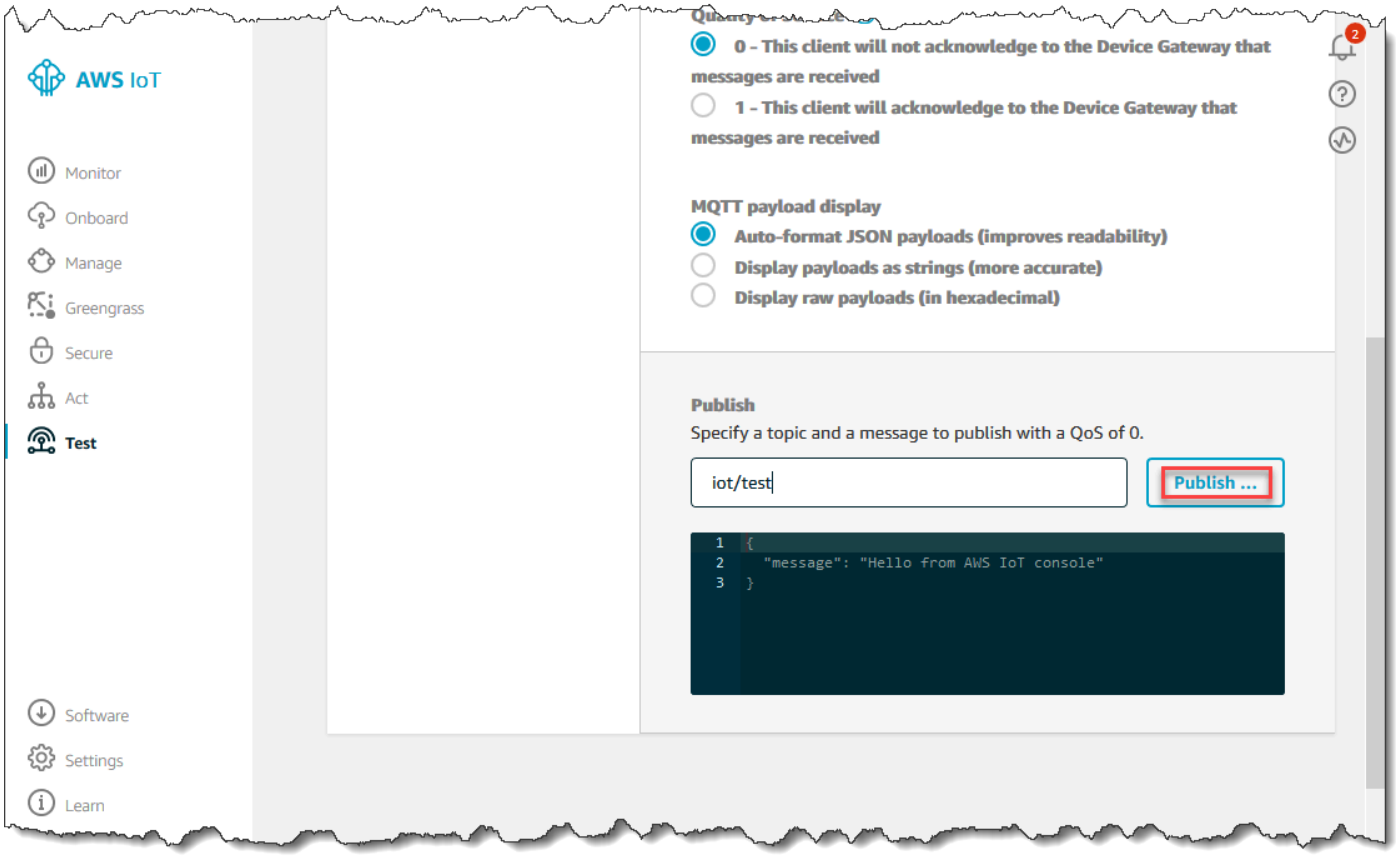
1. 在 AWS IoT 控制台的左侧导航窗格中选择 Test (测试)。



2. 在 MQTT 客户端页面上，在 Publish 部分的 Specify a topic 中，键入 **`iot/test`**。在消息负载部分，请验证以下 JSON 内容是否存在，如果不存在，则键入它们。

```
{
    "message": "Hello from the IoT console"
}
```

3. 选择 Publish to topic（发布到主题）。



这会发布一条消息，该消息路由到您之前创建的数据存储。

使用 BatchPutMessage API

将消息数据输入 AWS IoT Analytics 的另一种方法是使用 `BatchPutMessage` API 命令。此方法不要求您设置 AWS IoT 规则来将具有特定主题的消息路由到您的通道。但它要求向通道发送数据/消息的设备能够运行使用 AWS 开发工具包创建的软件或能够使用AWS CLI 调用`BatchPutMessage`。

1. 创建一个文件`messages.json`，其中包含要发送的消息（在本示例中，只发送一条消息）。

```
[
    { "messageId": "message01", "payload": "{ \"message\": \"Hello from the CLI
\" }" }
]
```

2. 运行 `batch-put-message` 命令。

```
aws iotanalytics batch-put-message  --channel-name mychannel  --messages file://
messages.json --cli-binary-format raw-in-base64-out
```

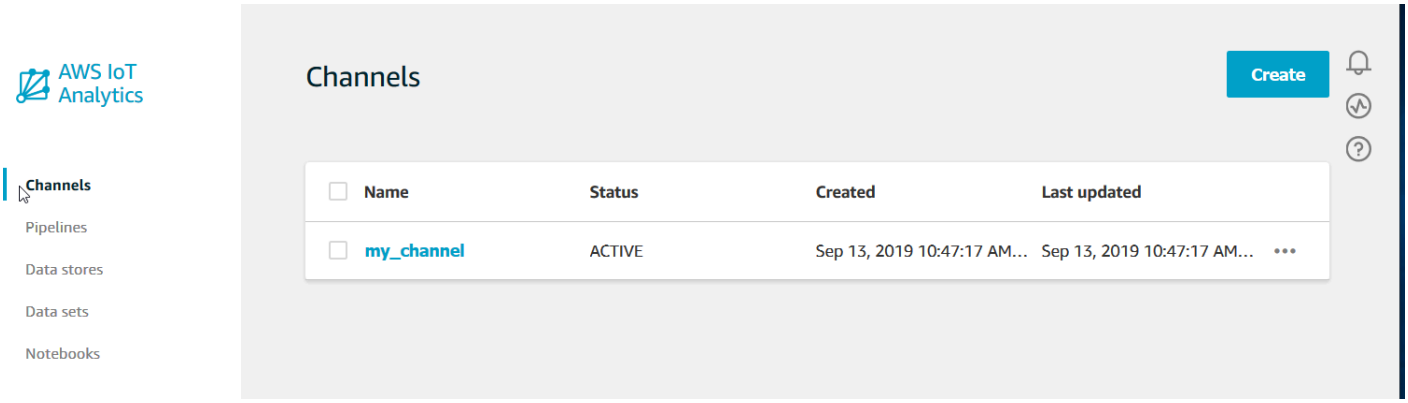
如果没有错误，将显示以下输出。

```
{
    "batchPutMessageErrorEntries": []
}
```

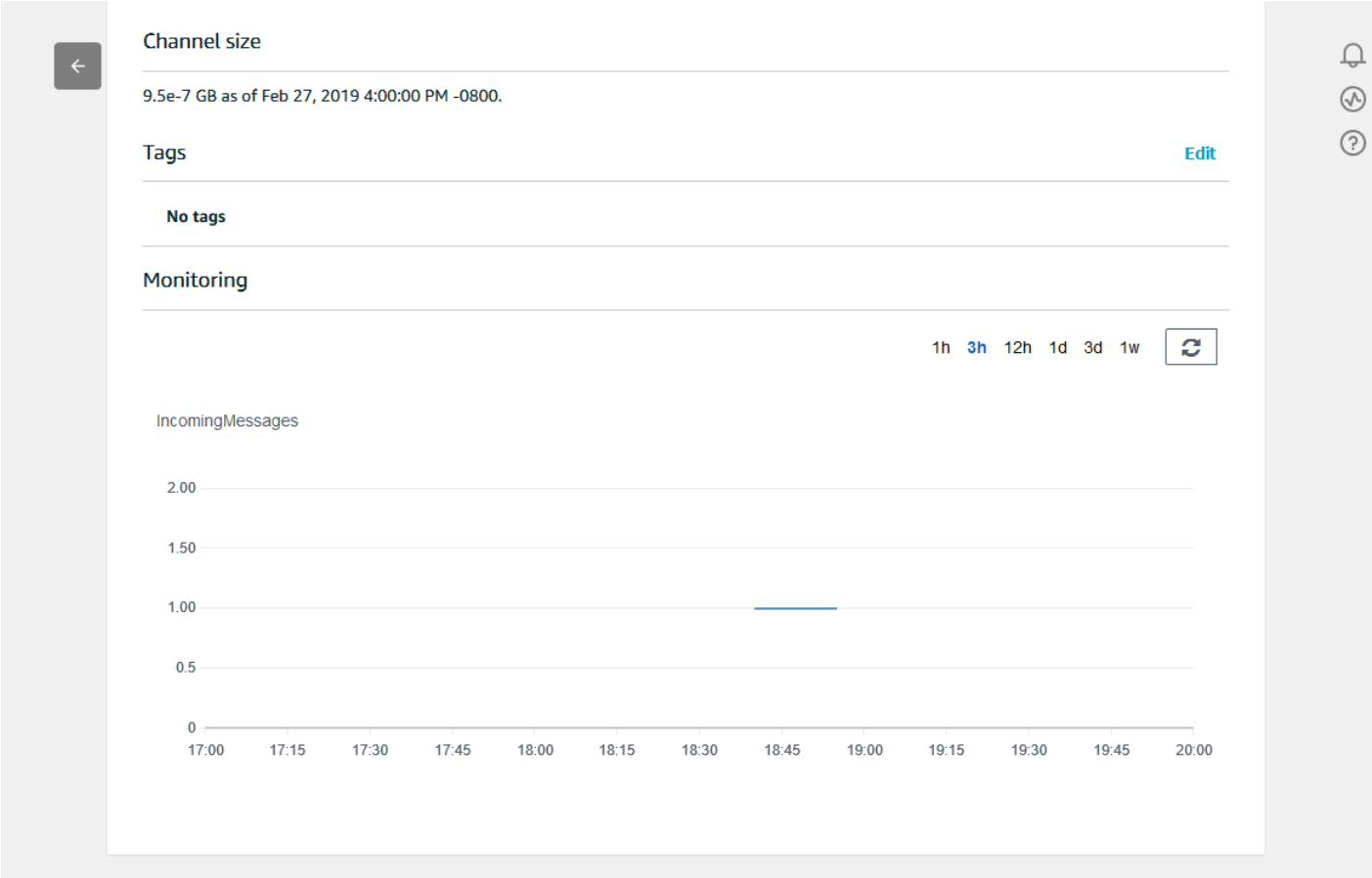
监控提取的数据

您可以使用 AWS IoT Analytics 控制台检查您发送的消息是否提取到您的通道中。

1. 在 AWS IoT Analytics 控制台中，在左侧导航窗格中选择 Prepare (准备) 并选择 Channel (通道) (如有必要) ，然后选择您之前创建的通道的名称：

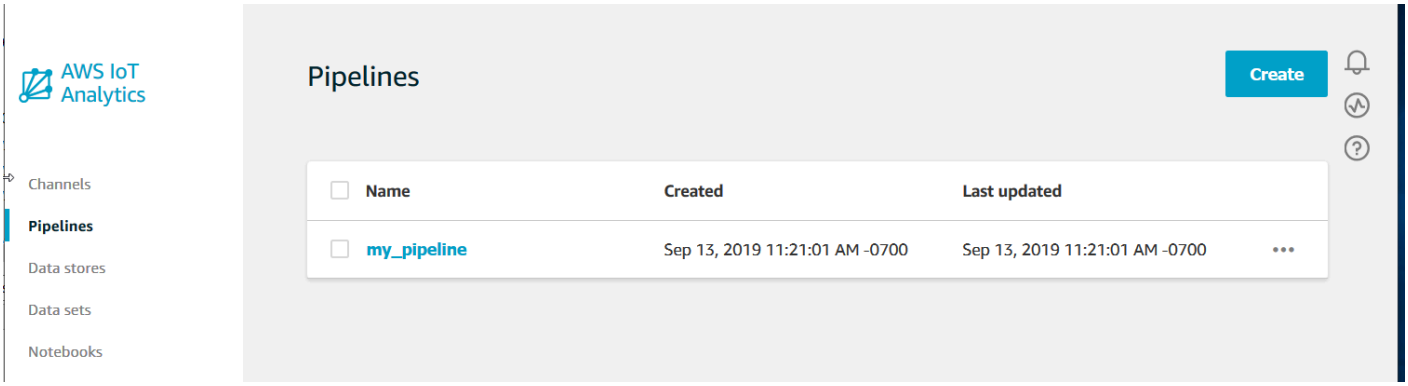


2. 在通道详细信息页面上，向下滚动到监控部分。根据需要，选择其中的一个时间范围指示符 (1 小时 3 小时 12 小时 1 天 3 天 1 周) 以调整显示的时间范围。将会看到一条图表线，以指示在指定时间范围内提取到该通道的消息数：

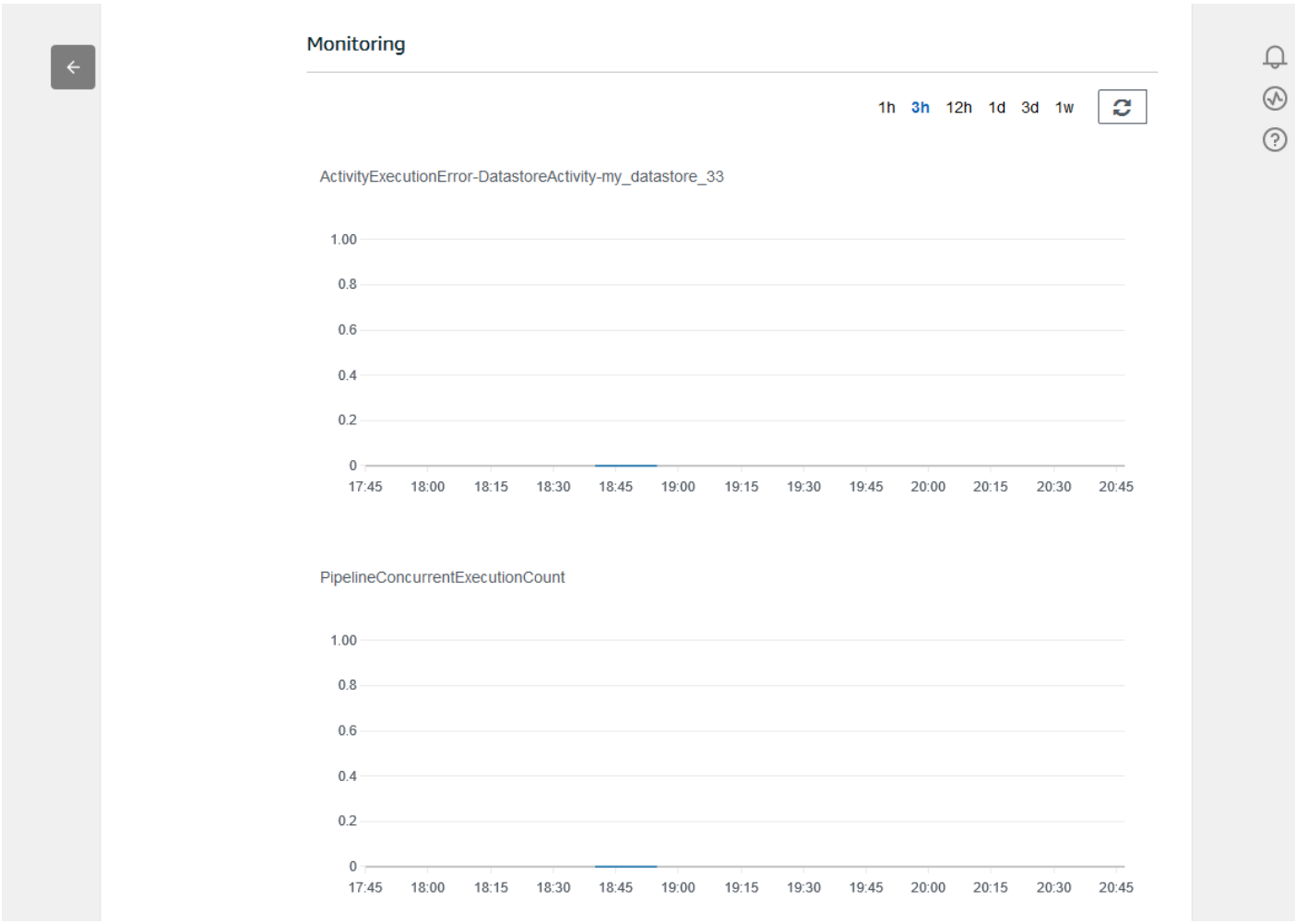


可以使用类似的监控功能以检查管道活动执行。您可以在管道的详细信息页面上监控活动执行错误。如果未将活动指定为管道的一部分，将显示 0 个执行错误。

1. 在 AWS IoT Analytics 控制台中，在左侧导航窗格中选择 Prepare (准备)，选择 Pipelines (管道)，然后选择您之前创建的管道的名称。



2. 在管道详细信息页面上，向下滚动到监控部分。根据需要，选择其中的一个时间范围指示符（1 小时 3 小时 12 小时 1 天 3 天 1 周）以调整显示的时间范围。将会看到一个图形线，以指示指定时间范围内的管道活动执行错误数。



创建数据集

您可通过创建 SQL 数据集或容器数据集从数据存储中检索数据。AWS IoT Analytics 可通过查询数据来回答分析问题。虽然数据存储不是数据库，但您可以使用 SQL 表达式来查询数据并生成存储在数据集中的结果。

主题

- 查询数据
- 访问查询的数据

查询数据

要查询数据，您需要创建一个数据集。数据集包含用于查询数据存储的 SQL 以及在您选择的日期和时间重复该查询的可选计划。您可以使用类似于 Amazon CloudWatch 计划表达式的表达式创建可选的计划。

运行以下命令以创建数据集。

```
aws iotanalytics create-dataset --cli-input-json file://mydataset.json
```

其中 mydataset.json 文件包含以下内容。

```
{
    "datasetName": "mydataset",
    "actions": [
        {
            "actionName":"myaction",
            "queryAction": {
                "sqlQuery": "select * from mydatastore"
            }
        }
    ]
}
```

运行以下命令，通过执行查询来创建数据集内容。

```
aws iotanalytics create-dataset-content --dataset-name mydataset
```

等待几分钟，在创建完数据集内容后，便可继续。

访问查询的数据

查询结果是以 CSV 文件格式存储的数据集内容。将通过 Amazon S3 向您提供该文件。以下示例说明了如何检查结果是否已准备就绪、文件是否已下载。

运行以下 get-dataset-content 命令：

```
aws iotanalytics get-dataset-content --dataset-name mydataset
```

如果数据集包含任何数据，则 get-dataset-content 输出在 status 字段中具有 "state": "SUCCEEDED"，如以下示例所示：

```
{
    "timestamp": 1508189965.746,
    "entries": [
        {
          "entryName": "someEntry",
          "dataURI": "https://aws-iot-analytics-datasets-f7253800-859a-472c-aa33-
e23998b31261.s3.amazonaws.com/results/f881f855-c873-49ce-abd9-b50e9611b71f.csv?X-Amz-"

        }
    ],
    "status": {
      "state": "SUCCEEDED",
      "reason": "A useful comment."
    }
}
```

`dataURI` 是输出结果的签名 URL。它在较短的一段时间内有效 (几个小时)。根据您的工作流，您可能需要在访问内容之前始终调用 `get-dataset-content`，因为调用此命令会生成新的签名 URL。

探索 AWS IoT Analytics 数据

您可以通过多种方法存储、分析和可视化AWS IoT Analytics数据。

本页面上的主题：

- Amazon S3
- AWS IoT Events
- Amazon QuickSight
- Jupyter Notebook

Amazon S3

您可以将数据集内容发送到 Amazon Simple Storage Service (Amazon S3) 存储桶，从而允许与现有数据湖集成在一起，或者从内部应用程序和可视化工具中进行访问。请参阅 CreateDataset 中的 `contentDeliveryRules::destination::s3DestinationConfiguration` 字段。

AWS IoT Events

您可以将数据集内容作为输入发送到AWS IoT Events，您可以使用该服务监控设备或进程故障或操作更改，并在发生此类事件时触发其他操作。

为此，请使用 CreateDataset 并在 `contentDeliveryRules :: destination :: iotEventsDestinationConfiguration :: inputName` 字段中指定 AWS IoT Events 输入。您还必须指定授权 AWS IoT Analytics 执行“iotevents:BatchPutMessage”的角色的 `roleArn`。每当创建数据集的内容时，AWS IoT Analytics 都会将每个数据集内容条目作为消息发送到指定的 AWS IoT Events 输入。例如，如果您的数据集包含：

```
"what","who","dt"
"overflow","sensor01","2019-09-16 09:04:00.000"
"overflow","sensor02","2019-09-16 09:07:00.000"
"underflow","sensor01","2019-09-16 11:09:00.000"
...
```

然后 AWS IoT Analytics 会发送包含以下字段的消息：

```
{ "what": "overflow", "who": "sensor01", "dt": "2019-09-16 09:04:00.000" }
```

```
{ "what": "overflow", "who": "sensor02", "dt": "2019-09-16 09:07:00.000" }
```

您需要创建一个 AWS IoT Events 输入来识别感兴趣的字段（`what`、`who`、`dt` 中的一个或多个）的，并创建一个 AWS IoT Events 探测器模型，以在事件中使用这些输入字段来触发操作或设置内部变量。

Amazon QuickSight

AWS IoT Analytics 提供与 Amazon QuickSight 的直接集成。Amazon QuickSight 是一项快速业务分析服务，可用于构建可视化内容，执行临时分析，并快速地从您的数据中获得业务见解。Amazon QuickSight 使组织可以扩展至数十万用户，并通过使用强大的内存中引擎 (SPICE) 提供响应及时的性能。可以在这些区域中使用 Amazon QuickSight。

Jupyter Notebook

AWS IoT Analytics 数据集也可以由 Jupyter Notebook 直接使用，以执行高级分析和数据探索。Jupyter Notebook 是一个开源解决方案。您可以从 http://jupyter.org/install.html 安装并下载。还额外提供与 SageMaker 的集成，后者是一项 Amazon 托管的笔记本解决方案。

保留多个版本的数据集

在调用 CreateDataset 和 UpdateDataset API 时，可通过为数据集 `retentionPeriod and versioningConfiguration` 字段指定值来选择要保留的数据集内容数量以及保留的时长：

```
...
"retentionPeriod": {
  "unlimited": "boolean",
  "numberOfDays": "integer"
},
"versioningConfiguration": {
  "unlimited": "boolean",
  "maxVersions": "integer"
},
...
```

这两个参数的设置一起使用，以通过以下方式确定保留的数据集内容版本数和保留天数。

	retentionPeriod [未指定]	retentionPeriod： unlimited = TRUE，numberOfDays = 未设置	retentionPeriod： unlimited = FALSE，numberOfDays = X
versioningConfiguration： [未指定]	仅将最新版本以及最新的成功版本（如果不同）保留 90 天。	仅无限期保留最新版本以及最新的成功版本（如果不同）。	仅将最新版本以及最新的成功版本（如果不同）保留 X 天。
versioningConfiguration： unlimited = TRUE，未设置 maxVersions	将保留过去 90 天的所有版本，而无论具有多少个版本。	对保留的版本数没有任何限制。	将保留过去 X 天的所有版本，而无论具有多少个版本。
versioningConfiguration： unlimited = FALSE，maxVersions = Y	将保留过去 90 天内不超过 Y 个版本。	将最多保留 Y 个版本，而无论它们存在多长时间。	将保留过去 X 天内不超过 Y 个版本。

消息负载语法

您发送至 AWS IoT Analytics 的消息负载（数据）的字段名称：

- 必须仅包含字母数字字符和下划线 (_)；不允许使用其他特殊字符
- 必须以字母字符或单个下划线 (_) 开头。
- 不能包含连字符 (-)。
- 正则表达式术语：“^[A-Za-z_]([A-Za-z0-9]*|[A-Za-z0-9][A-Za-z0-9_]*)$”。
- 长度不能超过 255 个字符。
- 不区分大小写。同一负载中名为“foo”和“FOO”的字段被视为重复字段。

例如，在消息负载中，{"temp_01": 29} 或 {"_temp_01": 29} 有效，但 {"temp-01": 29}、{"01_temp": 29} 或 {"__temp_01": 29} 无效。

使用 AWS IoT SiteWise 数据

AWS IoT SiteWise 是一种托管式服务，能够用于大规模采集、建模、分析和可视化显示来自工业设备的数据。该服务提供了一个资产建模框架，用来构建工业设备、流程和设施的表示形式。

使用 AWS IoT SiteWise 资产模型，您可以定义要使用的工业设备数据以及如何将数据处理为复杂的指标。您可以配置资产模型以在 AWS 云中收集和处理数据。有关更多信息，请参阅 AWS IoT SiteWise 用户指南。

AWS IoT Analytics 与 AWS IoT SiteWise 集成，以便您可以对 AWS IoT SiteWise 数据运行和安排 SQL 查询。要开始查询 AWS IoT SiteWise 数据，请按照《AWS IoT SiteWise 用户指南》中配置存储设置中的步骤创建数据存储。然后，按照 使用 AWS IoT SiteWise 数据创建数据集（控制台） 或 使用数据 AWS IoT SiteWise (AWS CLI)创建数据集 中的步骤创建 AWS IoT Analytics 数据集并对工业数据运行 SQL 查询。

主题

- 使用 AWS IoT SiteWise 数据创建 AWS IoT Analytics 数据集
- 访问数据集内容
- 教程：在中查询 AWS IoT SiteWise 数据 AWS IoT Analytics

使用 AWS IoT SiteWise 数据创建 AWS IoT Analytics 数据集

AWS IoT Analytics数据集包含用于查询数据存储中数据的 SQL 语句和表达式，以及在您指定的日期和时间重复该查询的可选计划。您可以使用与 Amazon CloudWatch 计划表达式类似的表达式创建可选的计划。

> **Note**
>
> 数据集通常是数据的集合，这些数据可能以表格形式呈现，也可能不以表格形式呈现。相比之下，AWS IoT Analytics 通过对数据存储中的数据应用 SQL 查询来创建数据集。

按照以下步骤操作，开始为 AWS IoT SiteWise 数据创建数据集。

主题

使用 AWS IoT SiteWise 数据创建数据集（控制台）

使用以下步骤在 AWS IoT Analytics 控制台中为 AWS IoT SiteWise 数据创建数据集。

创建数据集

1. 在 https://console.aws.amazon.com/iotanalytics/ 的左导航窗格中，选择数据集。
2. 在创建数据集页面上，选择创建 SQL。
3. 在指定数据集详细信息页面上，指定数据集的详细信息。
   a. 输入数据集的名称。
   b. 对于数据存储源，选择用于标识 AWS IoT SiteWise 数据存储的唯一 ID。
   c. （可选）在标签中，将一个或多个自定义标签（键值对）添加到数据集中。
4. 使用 SQL 表达式查询数据并回答分析问题。
   a. 在作者查询字段中，输入使用通配符显示最多五行数据的 SQL 查询。

```
SELECT * FROM my_iotsitewise_datastore.asset_metadata LIMIT 5
```

有关AWS IoT Analytics中支持的 SQL 功能的更多信息，请参阅AWS IoT Analytics 中的 SQL 表达式。或者，参阅 教程：在中查询 AWS IoT SiteWise 数据 AWS IoT Analytics 了解能够提供数据见解的统计查询示例。

b. 您可以选择测试查询 来验证输入是否正确，并在查询后的表格中显示结果。

> **Note**
>
> 由于 Amazon Athena 限制了运行查询的最大数量，因此应将 SQL 查询限制为合理的大小，以免长时间运行。

5. （可选）使用指定时间范围内的数据创建数据集内容时，某些数据可能无法及时送达处理。要允许延迟，可指定偏移量或增量。有关更多信息，请参阅通过 Amazon CloudWatch Events 获取延迟数据通知。

   在配置数据选择筛选条件页面上配置数据选择筛选条件后，选择下一步。

6. （可选）在设置查询计划页面上，可将此查询安排为定期运行以刷新数据集。您可随时创建和编辑数据集计划。

   > **Note**
   >
   > AWS IoT Analytics 每六小时从 AWS IoT SiteWise 提取一次数据。我们建议选择六小时或更长时间的提取频率。

   选择频率，然后选择 下一步。

7. AWS IoT Analytics 将创建多个版本的数据集内容，并存储指定时间段内的分析结果。我们建议 90 天，但您可以选择设置自定义保留策略。您也可限制数据集内容存储版本的数量。

   在配置数据集结果页面上选择选项后，选择下一步。

8. （可选）您可通过配置数据集结果传送规则，将其传送到特定目的地，例如 AWS IoT Events。

   在配置数据集内容传送规则页面上选择选项后，选择下一步。

9. 检查您的选择，然后选择 创建。

10. 验证您的新数据集是否显示在数据集页面上。

使用数据 AWS IoT SiteWise (AWS CLI)创建数据集

运行以下 AWS CLI 命令，开始查询 AWS IoT SiteWise 数据。

此处显示的示例使用 AWS Command Line Interface (AWS CLI)。有关 AWS CLI 的更多信息，请参阅《AWS Command Line Interface 用户指南》。有关 AWS IoT Analytics 可用的 CLI 命令的更多信息，请参阅《AWS Command Line Interface 参考》中的 iotanalytics。

若要创建数据集

1. 请运行以下 `create-dataset` 命令以创建数据集。

```
aws iotanalytics create-dataset --cli-input-json file://my_dataset.json
```

其中 `my_dataset.json` 文件包含以下内容。

```
{
    "datasetName": "my_dataset",
    "actions": [
        {
            "actionName":"my_action",
            "queryAction": {
                "sqlQuery": "SELECT * FROM my_iotsitewise_datastore.asset_metadata
 LIMIT 5"
            }
        }
    ]
}
```

有关 AWS IoT Analytics SQL 功能的更多信息，请参阅AWS IoT Analytics 中的 SQL 表达式。或者，参阅 教程：在中查询 AWS IoT SiteWise 数据 AWS IoT Analytics 了解能够提供数据见解的统计查询示例。

2. 运行以下 `create-dataset-content` 命令，通过运行查询来创建数据集内容。

```
aws iotanalytics create-dataset-content --dataset-name my_dataset
```

访问数据集内容

SQL 查询结果是以 CSV 文件格式存储的数据集内容。将通过 Amazon S3 向您提供该文件。以下步骤说明了如何检查结果是否已准备就绪、文件是否已下载。

主题

访问 AWS IoT Analytics（控制台）中的数据集内容

如果数据集包含任何数据，则可在 AWS IoT Analytics 控制台中预览和下载 SQL 查询结果。

访问 AWS IoT Analytics 数据集结果

1. 在控制台的数据集页面上，选择要访问的数据集的名称。
2. 在数据集摘要页面上，选择内容选项卡。
3. 在数据集内容表中，选择要预览结果或下载结果的 csv 文件的查询名称。

访问 AWS IoT Analytics (AWS CLI)中的数据集内容

如果数据集包含任何数据，则可预览和下载 SQL 查询结果。

此处显示的示例使用 AWS Command Line Interface (AWS CLI)。有关 AWS CLI 的更多信息，请参阅 AWS Command Line Interface 用户指南。有关 AWS IoT Analytics 可用的 CLI 命令的更多信息，请参阅《AWS Command Line Interface 参考》中的 iotanalytics。

访问 AWS IoT Analytics 数据集结果(AWS CLI)

1. 运行以下 `get-dataset-content` 命令以查看查询结果。

```
aws iotanalytics get-dataset-content --dataset-name my_iotsitewise_dataset
```

2. 如果数据集包含任何数据，则 `get-dataset-content` 输出的 `status` 字段中包含 `"state": "SUCCEEDED"`，如下例所示。

```
{
    "timestamp": 1508189965.746,
    "entries": [
```

```
        {
          "entryName": "my_entry_name",
          "dataURI": "https://aws-iot-analytics-datasets-f7253800-859a-472c-aa33-
e23998b31261.s3.amazonaws.com/results/f881f855-c873-49ce-abd9-b50e9611b71f.csv?X-
Amz-"

        }
    ],
    "status": {
      "state": "SUCCEEDED",
      "reason": "A useful comment."
    }
}
```

3. `get-dataset-content` 的输出包括 `dataURI`，这是输出结果的签名 URL。它在较短的一段时间内有效 (几个小时)。访问 `dataURI` 网址以访问 SQL 查询结果。

> **Note**
>
> 根据您的工作流，您可能需要在访问内容之前始终调用 `get-dataset-content`，因为调用此命令会生成新的签名 URL。

教程：在中查询 AWS IoT SiteWise 数据 AWS IoT Analytics

本教程演示了如何在中查询 AWS IoT SiteWise 数据 AWS IoT Analytics。本教程使用演示中的数据 AWS IoT SiteWise ，该演示提供了风力发电场的样本数据集。

> **Important**
>
> 此演示创建和使用的资源是收费的。

主题

先决条件

在此教程中，您需要以下资源：

- 您必须有一个 AWS 帐户才能开始使用 AWS IoT SiteWise 和 AWS IoT Analytics。如果您还没有账户，请按创建 AWS 账户中的步骤进行操作。
- 运行 Windows、macOS、Linux 或 Unix 的开发计算机（用于访问 AWS Management Console）。有关更多信息，请参阅 AWS Management Console入门。
- AWS IoT SiteWise 定义 AWS IoT SiteWise 模型和资产的数据，以及流式传输代表风力发电场设备数据的数据。要创建数据，请按照《AWS IoT SiteWise 用户指南》中创建 AWS IoT SiteWise 演示中的步骤进行操作。
- 您的 AWS IoT SiteWise 演示风电场设备数据存储在您管理的现有数据存储中。有关如何为数据创建数据存储的更多信息，请参阅《AWS IoT SiteWise 用户指南》中的配置存储设置。 AWS IoT SiteWise

> **Note**
>
> 您的 AWS IoT SiteWise 元 AWS IoT SiteWise 数据在创建后不久就会出现在您的数据存储中；但是，原始数据最多可能需要六个小时才能显示。同时，您可以创建 AWS IoT Analytics 数据集并对元数据进行查询。

后续步骤

加载并验证数据

加载并验证数据

您在本教程中查询的数据是一组样本 AWS IoT SiteWise 数据，用于对风力发电场中的风力发电机涡轮机进行建模。

> **Note**
>
> 在本教程中，您将查询数据存储中的三个表：
>
> - `raw` - 包含各项资产未经处理的原始数据。
> - `asset_metadata` - 包含各项资产的一般信息。
> - `asset_hierarchy_metadata` - 包含有关资产之间关系的信息。

运行本教程中的 SQL 查询

1. 按照使用 AWS IoT SiteWise 数据创建数据集（控制台）或使用数据 AWS IoT SiteWise (AWS CLI)创建数据集中的步骤为您的 AWS IoT Analytics 数据创建 AWS IoT SiteWise 数据集。
2. 要更新本教程中的数据集查询，请执行以下操作。

   a. 在 AWS IoT Analytics 控制台的 “数据集” 页面上，选择您在上一页上创建的数据集的名称。

   b. 在数据集摘要页面上，选择编辑以编辑 SQL 查询。

   c. 要在查询后的表中显示结果，请选择测试查询。

   或者，您可以使用 AWS CLI运行以下 `update-dataset` 命令来修改 SQL 查询。

   ```
   aws iotanalytics update-dataset --cli-input-json file://update-query.json
   ```

   `update-query.json` 的内容：

   ```
   {
       "datasetName": "my_dataset",
       "actions": [
           {
               "actionName": "myDatasetUpdateAction",
               "queryAction": {
                   "sqlQuery": "SELECT * FROM my_iotsitewise_datastore.asset_metadata
    LIMIT 3"
               }
           }
       ]
   }
   ```

3. 在 AWS IoT Analytics 控制台中或使用中 AWS CLI，对您的数据运行以下查询，以验证您的`asset_metadata`表是否成功加载。

   ```
   SELECT COUNT(*) FROM my_iotsitewise_datastore.asset_metadata
   ```

   同样，您可以验证 `asset_hierarchy_metadata` 和 `raw` 表是否为空。

下一个步骤

数据探究

数据探究

创建 AWS IoT SiteWise 数据并将其加载到数据存储后，您可以创建数据 AWS IoT Analytics 集并在中运行 SQL 查询， AWS IoT Analytics 以发现有关资产的见解。以下查询演示了如何在运行统计查询之前探索数据。

使用 SQL 查询探索数据

1. 查看每个表（例如原始表）中的列和值的示例。

```
SELECT * FROM my_iotsitewise_datastore.raw LIMIT 5
```

2. SELECT DISTINCT用于查询您的asset_metadata表并列出 AWS IoT SiteWise 资产的（唯一）名称。

```
SELECT DISTINCT assetname FROM my_iotsitewise_datastore.asset_metadata ORDER BY
 assetname
```

3. 要列出有关特定 AWS IoT SiteWise 资产属性的信息，请使用子WHERE句。

```
SELECT assetpropertyname,
     assetpropertyunit,
     assetpropertydatatype
FROM my_iotsitewise_datastore.asset_metadata
WHERE assetname = 'Demo Turbine Asset 2'
```

4. 使用 AWS IoT Analytics，您可以联接来自数据存储中两个或多个表的数据，如以下示例所示。

```
SELECT * FROM my_iotsitewise_datastore.raw AS raw
JOIN my_iotsitewise_datastore.asset_metadata AS asset_metadata
ON raw.seriesId = asset_metadata.timeseriesId
```

要查看资产之间的所有关系，请使用以下查询中的 JOIN 功能。

```
SELECT DISTINCT parent.assetName as "Parent name",
    child.assetName AS "Child name"
FROM (
    SELECT sourceAssetId AS parent,
           targetAssetId AS child
    FROM my_iotsitewise_datastore.asset_hierarchy_metadata
    WHERE associationType = 'CHILD'
```

```
)
AS relations
JOIN my_iotsitewise_datastore.asset_metadata AS child
    ON relations.child = child.assetId
JOIN my_iotsitewise_datastore.asset_metadata AS parent
    ON relations.parent = parent.assetId
```

后续步骤

运行统计查询

运行统计查询

既然您已经浏览了 AWS IoT SiteWise 数据，就可以运行统计查询，为您的工业设备提供宝贵的见解。以下查询演示了您可以检索的一些信息。

对 AWS IoT SiteWise 演示风电场数据运行统计查询

1. 运行以下 SQL 命令，查找特定资产（演示涡轮机资产 4）所有数值属性的最新值。

```
SELECT assetName,
    assetPropertyName,
    assetPropertyUnit,
    max_by(value, timeInSeconds) AS Latest
FROM (
    SELECT *,
        CASE assetPropertyDataType
        WHEN 'DOUBLE' THEN
        cast(doubleValue AS varchar)
        WHEN 'INTEGER' THEN
        cast(integerValue AS varchar)
        WHEN 'STRING' THEN
        stringValue
        WHEN 'BOOLEAN' THEN
        cast(booleanValue AS varchar)
        ELSE NULL
        END AS value
    FROM my_iotsitewise_datastore.asset_metadata AS asset_metadata
    JOIN my_iotsitewise_datastore.raw AS raw
        ON raw.seriesId = asset_metadata.timeSeriesId
    WHERE startYear=2021
        AND startMonth=7
```

```
        AND startDay=8
        AND assetName='Demo Turbine Asset 4'
)
GROUP BY assetName, assetPropertyName, assetPropertyUnit
```

2. 将元数据表和原始表链接起来，确定所有资产（父资源除外）的最大风速属性。

```
SELECT child_assets_data_set.parentAssetId,
        child_assets_data_set.childAssetId,
        asset_metadata.assetPropertyId,
        asset_metadata.assetPropertyName,
        asset_metadata.timeSeriesId,
        raw_data_set.max_speed
FROM (
   SELECT sourceAssetId AS parentAssetId,
        targetAssetId AS childAssetId
    FROM my_iotsitewise_datastore.asset_hierarchy_metadata
    WHERE associationType = 'CHILD'
)
AS child_assets_data_set
JOIN mls_demo.asset_metadata AS asset_metadata
    ON asset_metadata.assetId = child_assets_data_set.childAssetId
JOIN (
    SELECT seriesId, MAX(doubleValue) AS max_speed
    FROM my_iotsitewise_datastore.raw
    GROUP BY seriesId
)
AS raw_data_set
ON raw_data_set.seriesId = asset_metadata.timeseriesid
WHERE assetPropertyName = 'Wind Speed'
ORDER BY max_speed DESC
```

3. 要查找资产（演示涡轮机资产 2）的特定属性（风速）的平均值，请运行以下 SQL 命令。必须将 `my_bucket_id` 替换为存储桶 ID。

```
SELECT AVG(doubleValue) as "Average wind speed"
FROM my_iotsitewise_datastore.raw
WHERE seriesId =
    (SELECT timeseriesId
    FROM my_iotsitewise_datastore.asset_metadata as asset_metadata
    WHERE asset_metadata.assetname = 'Demo Turbine Asset 2'
            AND asset_metadata.assetpropertyname = 'Wind Speed')
```

后续步骤

清除教程资源

清除教程资源

完成本教程后，清理资源以避免产生费用。

要删除您的 AWS IoT SiteWise 演示

一周后，该 AWS IoT SiteWise 演示会自行删除。如果您已完成演示资源的使用，则可以提前删除演示。使用以下步骤手动删除演示。

1. 导航到 AWS CloudFormation 控制台。
2. 从堆栈列表中选择IoTSiteWiseDemoAssets。
3. 选择删除。当您删除堆栈时，为演示创建的所有资源都将被删除。
4. 在确认对话框中，选择 删除。

   删除堆栈约需 15 分钟时间。如果演示无法删除，请再次选择右上角的 删除。如果演示无法再次删除，请按照 AWS CloudFormation 控制台中的步骤跳过删除失败的资源，然后重试。

删除数据存储

- 要删除您的托管数据存储，请运行 CLI 命令 `delete-datastore`，如下例所示。

```
aws iotanalytics delete-datastore --datastore-name my_IotSiteWise_datastore
```

删除您的 AWS IoT Analytics 数据集

- 要删除您的数据集，请运行 CLI 命令 `delete-dataset`，如下例所示。在执行此操作之前，您不必删除数据集的内容。

```
aws iotanalytics delete-dataset --dataset-name my_dataset
```

> **Note**
>
> 此命令不生成任何输出。

管道活动

最简单的正常运行的管道可将一个通道连接到数据存储，这使其成为具有两个活动的管道：一个 `channel` 活动和一个 `datastore` 活动。您可以通过向管道添加额外的活动，实现更强大的消息处理功能。

您可以使用 RunPipelineActivity 操作，来模拟在所提供消息负载上运行管道活动的结果。在开发和调试管道活动时，您会发现这非常有用。RunPipelineActivity 示例演示了具体用法。

通道活动

管道中的第一个活动必须是 `channel` 活动，它确定要处理的消息的来源。

```
{
    "channel": {
        "name": "MyChannelActivity",
        "channelName": "mychannel",
        "next": "MyLambdaActivity"
    }
}
```

数据存储活动

`datastore` 活动是最后一个活动，指定将处理后的数据存储到何处。

```
{
    "datastore": {
        "name": "MyDatastoreActivity",
        "datastoreName": "mydatastore"
    }
}
```

AWS Lambda 活动

`lambda` 活动可用于对消息执行复杂的处理。例如，您可使用来自外部 API 操作输出的数据来丰富消息，或者根据来自 Amazon DynamoDB 的逻辑筛选消息。但是，在进入数据存储之前，您无法使用此管道活动来添加其他消息或删除现有消息。

用于**lambda** 活动的 AWS Lambda 函数必须接收并返回一组 JSON 对象。有关示例，请参阅 the section called “Lambda 函数示例 1”。

要授权 AWS IoT Analytics 调用您的 Lambda 函数，您必须添加一个策略。例如，运行以下 CLI 命令并将 *exampleFunctionName* 替换为您的 Lambda 函数的名称，将 *123456789012* 替换为您的 AWS 账户 ID，然后使用给定 Lambda 函数调用管道的 Amazon 资源名称 (ARN)。

```
aws lambda add-permission --function-name exampleFunctionName --
action lambda:InvokeFunction --statement-id iotanalytics --principal
 iotanalytics.amazonaws.com --source-account 123456789012 --source-arn
 arn:aws:iotanalytics:us-east-1:123456789012:pipeline/examplePipeline
```

该命令将返回以下输出：

```
{
    "Statement": "{\"Sid\":\"iotanalyticsa\",\"Effect\":\"Allow\",
\"Principal\":{\"Service\":\"iotanalytics.amazonaws.com\"},\"Action\":
\"lambda:InvokeFunction\",\"Resource\":\"arn:aws:lambda:aws-region:aws-
account:function:exampleFunctionName\",\"Condition\":{\"StringEquals\":
{\"AWS:SourceAccount\":\"123456789012\"},\"ArnLike\":{\"AWS:SourceArn\":
\"arn:aws:iotanalytics:us-east-1:123456789012:pipeline/examplePipeline\"}}}"
}
```

有关更多信息，请参阅AWS Lambda开发人员指南中的为AWS Lambda使用基于资源的策略。

Lambda 函数示例 1

在该示例中，Lambda 函数根据原始消息中的数据添加信息。设备发布一条包含类似于以下负载的消息。

```
{
  "thingid": "00001234abcd",
  "temperature": 26,
  "humidity": 29,
  "location": {
    "lat": 52.4332935,
    "lon": 13.231694
  },
  "ip": "192.168.178.54",
  "datetime": "2018-02-15T07:06:01"
}
```

并且该设备具有以下管道定义。

```
{
    "pipeline": {
        "activities": [
            {
                "channel": {
                    "channelName": "foobar_channel",
                    "name": "foobar_channel_activity",
                    "next": "lambda_foobar_activity"
                }
            },
            {
                "lambda": {
                    "lambdaName": "MyAnalyticsLambdaFunction",
                    "batchSize": 5,
                    "name": "lambda_foobar_activity",
                    "next": "foobar_store_activity"
                }
            },
            {
                "datastore": {
                    "datastoreName": "foobar_datastore",
                    "name": "foobar_store_activity"
                }
            }
        ],
        "name": "foobar_pipeline",
        "arn": "arn:aws:iotanalytics:eu-west-1:123456789012:pipeline/foobar_pipeline"
    }
}
```

下面的 Lambda Python 函数 (`MyAnalyticsLambdaFunction`) 向消息中添加 GMaps URL 和华氏温度。

```
import logging
import sys

# Configure logging
logger = logging.getLogger()
logger.setLevel(logging.INFO)
streamHandler = logging.StreamHandler(stream=sys.stdout)
formatter = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s - %(message)s')
```

```
streamHandler.setFormatter(formatter)
logger.addHandler(streamHandler)

def c_to_f(c):
    return 9.0/5.0 * c + 32

def lambda_handler(event, context):
    logger.info("event before processing: {}".format(event))
    maps_url = 'N/A'

    for e in event:
        #e['foo'] = 'addedByLambda'
        if 'location' in e:
            lat = e['location']['lat']
            lon = e['location']['lon']
            maps_url = "http://maps.google.com/maps?q={},{}".format(lat,lon)

        if 'temperature' in e:
            e['temperature_f'] = c_to_f(e['temperature'])

        logger.info("maps_url: {}".format(maps_url))
        e['maps_url'] = maps_url

    logger.info("event after processing: {}".format(event))

    return event
```

Lambda 函数示例 2

一种有用的方法是压缩并序列化消息负载，以降低传输和存储成本。在该第二个示例中，Lambda 函数假定消息负载表示已压缩并以字符串形式进行 Base64 编码（序列化）的 JSON 原始数据。它返回原始 JSON。

```
import base64
import gzip
import json
import logging
import sys

# Configure logging
logger = logging.getLogger()
logger.setLevel(logging.INFO)
streamHandler = logging.StreamHandler(stream=sys.stdout)
```

```
formatter = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s - %(message)s')
streamHandler.setFormatter(formatter)
logger.addHandler(streamHandler)

def decode_to_bytes(e):
    return base64.b64decode(e)

def decompress_to_string(binary_data):
    return gzip.decompress(binary_data).decode('utf-8')

def lambda_handler(event, context):
    logger.info("event before processing: {}".format(event))

    decompressed_data = []

    for e in event:
        binary_data = decode_to_bytes(e)
        decompressed_string = decompress_to_string(binary_data)

        decompressed_data.append(json.loads(decompressed_string))

    logger.info("event after processing: {}".format(decompressed_data))

    return decompressed_data
```

AddAttributes 活动

addAttributes 活动根据消息中现有的属性添加属性。这样，您就可以在存储之前更改消息的配置。例如，您可以使用 addAttributes 规范化来自不同代的设备固件的数据。

考虑以下输入消息。

```
{
    "device": {
        "id": "device-123",
        "coord": [ 47.6152543, -122.3354883 ]
    }
}
```

addAttributes 活动类似于以下内容。

```
{
```

```
    "addAttributes": {
        "name": "MyAddAttributesActivity",
        "attributes": {
            "device.id": "id",
            "device.coord[0]": "lat",
            "device.coord[1]": "lon"
        },
        "next": "MyRemoveAttributesActivity"
    }
}
```

该活动会将设备 ID 移到根级别，并提取 coord 数组中的值，将它们提升到称为 lat 和 lon 的顶级属性。作为此活动的结果，输入消息将转换为以下示例。

```
{
    "device": {
        "id": "device-123",
        "coord": [ 47.6, -122.3 ]
    },
    "id": "device-123",
    "lat": 47.6,
    "lon": -122.3
}
```

原始设备属性仍然存在。如果要删除它，您可以使用 removeAttributes 活动。

RemoveAttributes 活动

removeAttributes 活动从消息中删除属性。例如，给定以下消息，它是 addAttributes 活动的结果。

```
{
    "device": {
        "id": "device-123",
        "coord": [ 47.6, -122.3 ]
    },
    "id": "device-123",
    "lat": 47.6,
    "lon": -122.3
}
```

为了规范化该消息，使其只包含根级别所需的数据，请使用以下 removeAttributes 活动。

```
{
    "removeAttributes": {
        "name": "MyRemoveAttributesActivity",
        "attributes": [
            "device"
        ],
        "next": "MyDatastoreActivity"
    }
}
```

这会产生沿管道流动的以下消息。

```
{
    "id": "device-123",
    "lat": 47.6,
    "lon": -122.3
}
```

SelectAttributes 活动

selectAttributes 活动仅使用原始消息中的指定属性创建新消息。其他属性都将被丢弃。selectAttributes 只会在消息的根位置下创建新属性。因此，给定此消息：

```
{
    "device": {
        "id": "device-123",
        "coord": [ 47.6152543, -122.3354883 ],
        "temp": 50,
        "hum": 40
    },
    "light": 90
}
```

和此活动：

```
{
    "selectAttributes": {
        "name": "MySelectAttributesActivity",
```

```
        "attributes": [
            "device.temp",
            "device.hum",
            "light"
        ],
        "next": "MyDatastoreActivity"
    }
}
```

结果将是流经管道的以下消息。

```
{
    "temp": 50,
    "hum": 40,
    "light": 90
}
```

同样，`selectAttributes` 只能创建根级别对象。

Filter 活动

`filter` 活动根据消息属性筛选消息。此活动中使用的表达式看起来像一个 SQL `WHERE` 子句，它必须返回一个布尔值。

```
{
    "filter": {
        "name": "MyFilterActivity",
        "filter": "temp > 40 AND hum < 20",
        "next": "MyDatastoreActivity"
    }
}
```

DeviceRegistryEnrich 活动

`deviceRegistryEnrich` 活动允许您将 AWS IoT 设备注册表中的数据添加到您的消息负载中。例如，给定了以下 消息：

```
{
    "temp": 50,
```

```
    "hum": 40,
    "device" {
        "thingName": "my-thing"
    }
}
```

和类似下面这样的 deviceRegistryEnrich 活动：

```
{
    "deviceRegistryEnrich": {
        "name": "MyDeviceRegistryEnrichActivity",
        "attribute": "metadata",
        "thingName": "device.thingName",
        "roleArn": "arn:aws:iam::<your-account-number>:role:MyEnrichRole",
        "next": "MyDatastoreActivity"
    }
}
```

输出消息现在如以下示例所示：

```
{
    "temp" : 50,
    "hum" : 40,
    "device" {
        "thingName" : "my-thing"
    },
    "metadata" : {
        "defaultClientId": "my-thing",
        "thingTypeName": "my-thing",
        "thingArn": "arn:aws:iot:us-east-1:<your-account-number>:thing/my-thing",
        "version": 1,
        "thingName": "my-thing",
        "attributes": {},
        "thingId": "aaabbbccc-dddeeef-gghh-jjkk-llmmnnoopp"
    }
}
```

您必须在活动定义的 roleArn 字段中指定已附加适当权限的角色。该角色必须具有类似以下示例的权限策略：

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:DescribeThing"
            ],
            "Resource": [
                "arn:aws:iot:<region>:<account-id>:thing/<thing-name>"
            ]
        }
    ]
}
```

和类似如下的信任策略：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": "iotanalytics.amazonaws.com"
            },
            "Action": [
                "sts:AssumeRole"
            ]
        }
    ]
}
```

DeviceShadowEnrich 活动

deviceShadowEnrich 活动将来自 AWS IoT 设备影子服务的信息添加到消息中。例如，给定以下消息：

```
{
    "temp": 50,
    "hum": 40,
    "device": { "thingName": "my-thing" }
}
```

和以下 deviceShadowEnrich 活动：

```
{
    "deviceShadowEnrich": {
        "name": "MyDeviceShadowEnrichActivity",
        "attribute": "shadow",
        "thingName": "device.thingName",
        "roleArn": "arn:aws:iam::<your-account-number>:role:MyEnrichRole",
        "next": "MyDatastoreActivity"
    }
}
```

结果是类似于以下示例的消息。

```
{
    "temp": 50,
    "hum": 40,
    "device": {
        "thingName": "my-thing"
    },
    "shadow": {
        "state": {
            "desired": {
                "attributeX": valueX, ...
            },
            "reported": {
                "attributeX": valueX, ...
            },
            "delta": {
                "attributeX": valueX, ...
            }
        },
        "metadata": {
            "desired": {
                "attribute1": {
                    "timestamp": timestamp
                }, ...
            },
            "reported": ": {
                "attribute1": {
                    "timestamp": timestamp
                }, ...
            }
```

```
        },
        "timestamp": timestamp,
        "clientToken": "token",
        "version": version
    }
}
```

您必须在活动定义的 roleArn 字段中指定已附加适当权限的角色。该角色必须具有类似如下的权限策略。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:GetThingShadow"
            ],
            "Resource": [
                "arn:aws:iot:<region>:<account-id>:thing/<thing-name>"
            ]
        }
    ]
}
```

和类似如下的信任策略：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": "iotanalytics.amazonaws.com"
            },
            "Action": [
                "sts:AssumeRole"
            ]
        }
    ]
}
```

Math 活动

math 活动使用消息的属性计算算术表达式。表达式必须返回数字。例如，给定以下输入消息：

```
{
    "tempF": 50,
}
```

经过以下 math 活动处理后：

```
{
    "math": {
        "name": "MyMathActivity",
        "math": "(tempF - 32) / 2",
        "attribute": "tempC",
        "next": "MyDatastoreActivity"
    }
}
```

生成的消息类似于：

```
{
    "tempF" : 50,
    "tempC": 9
}
```

数学活动运算符和函数

您可以在 math 活动中使用以下运算符：

+	加
-	减
*	乘
/	除

%	取模

您可以在 `math` 活动中使用以下函数：

- abs(Decimal)
- acos(Decimal)
- asin(Decimal)
- atan(Decimal)
- atan2(Decimal, Decimal)
- ceil(Decimal)
- cos(Decimal)
- cosh(Decimal)
- exp(Decimal)
- ln(Decimal)
- log(Decimal)
- mod(Decimal, Decimal)
- power(Decimal, Decimal)
- round(Decimal)
- sign(Decimal)
- sin(Decimal)
- sinh(Decimal)
- sqrt(Decimal)
- tan(Decimal)
- tanh(Decimal)
- trunc(Decimal, int)

abs(Decimal)

返回数字的绝对值。

示例：`abs(-5)` 返回 5。

参数类型	结果
Int	Int，参数的绝对值。
Decimal	Decimal，参数的绝对值
Boolean	Undefined 。
String	Decimal。结果是参数的绝对值。如果字符串无法转换，则结果为 Undefined 。
数组	Undefined 。
对象	Undefined 。
Null	Undefined 。
未定义	Undefined 。

acos(Decimal)

以弧度为单位返回数字的反余弦值。在代入函数之前，Decimal 参数舍入到双精度。

示例：acos(0) = 1.5707963267948966

参数类型	结果
Int	Decimal (双精度)，参数的反余弦值。虚数结果返回 Undefined 。
Decimal	Decimal (双精度)，参数的反余弦值。虚数结果返回 Undefined 。
Boolean	Undefined 。
String	Decimal (双精度)参数的反余弦值。如果字符串无法转换，则结果为 Undefined 。虚数结果返回 Undefined 。

参数类型	结果
数组	Undefined .
对象	Undefined .
Null	Undefined .
未定义	Undefined .

asin(Decimal)

以弧度为单位返回数字的反正弦值。在代入函数之前，Decimal 参数舍入到双精度。

示例：asin(0) = 0.0

参数类型	结果
Int	Decimal (双精度)，参数的反正弦值。虚数结果返回 Undefined 。
Decimal	Decimal (双精度)，参数的反正弦值。虚数结果返回 Undefined 。
Boolean	Undefined .
String	Decimal (双精度)，参数的反正弦值。如果字符串无法转换，则结果为 Undefined 。虚数结果返回 Undefined 。
数组	Undefined .
对象	Undefined .
Null	Undefined .
未定义	Undefined .

atan(Decimal)

以弧度为单位返回数字的反正切值。在代入函数之前，Decimal 参数舍入到双精度。

示例：`atan(0)` = 0.0

参数类型	结果
Int	Decimal (双精度)，参数的反正切值。虚数结果返回 Undefined 。
Decimal	Decimal (双精度)，参数的反正切值。虚数结果返回 Undefined 。
Boolean	Undefined .
String	Decimal (双精度)，参数的反正切值。如果字符串无法转换，则结果为 Undefined 。虚数结果返回 Undefined 。
数组	Undefined .
对象	Undefined .
Null	Undefined .
未定义	Undefined .

atan2(Decimal, Decimal)

以弧度的形式返回 x 轴正方向与由两个参数定义的 (x,y) 点之间的角度。逆时针的角，角度为正数（上半平面，y > 0），顺时针的角，角度为负数 在代入函数之前，Decimal 参数舍入到双精度。

示例：`atan(1, 0)` = 1.5707963267948966

参数类型	参数类型	结果
Int / Decimal	Int / Decimal	Decimal (双精度)，x 轴和指定的 (x,y) 点之间的角度

参数类型	参数类型	结果
Int / Decimal / String	Int / Decimal / String	Decimal，所描述点的反正切值。如果字符串无法转换，则结果为 Undefined 。
其他值	其他值	Undefined .

ceil(Decimal)

将给定的 Decimal 向上舍入到最近的 Int。

示例：

ceil(1.2) = 2

ceil(11.2) = -1

参数类型	结果
Int	Int，参数值。
Decimal	Int，字符串将转换为 Decimal 并向上舍入到最近的 Int。如果字符串无法转换为 Decimal，则结果为 Undefined 。
其他值	Undefined .

cos(Decimal)

以弧度为单位返回数字的余弦值。在代入函数之前，Decimal 参数舍入到双精度。

示例：cos(0) = 1

参数类型	结果
Int	Decimal (双精度)，参数的余弦值。虚数结果返回 Undefined 。

参数类型	结果
Decimal	Decimal (双精度)，参数的余弦值。虚数结果返回 Undefined 。
Boolean	Undefined .
String	Decimal (双精度)，参数的余弦值。如果字符串无法转换为 Decimal，则结果为 Undefined 。虚数结果返回 Undefined 。
数组	Undefined .
对象	Undefined .
Null	Undefined .
未定义	Undefined .

cosh(Decimal)

以弧度为单位返回数字的双曲余弦值。在代入函数之前，Decimal 参数舍入到双精度。

示例：cosh(2.3) = 5.037220649268761

参数类型	结果
Int	Decimal (双精度)，参数的双曲余弦值。虚数结果返回 Undefined 。
Decimal	Decimal (双精度)，参数的双曲余弦值。虚数结果返回 Undefined 。
Boolean	Undefined .
String	Decimal (双精度)，参数的双曲余弦值。如果字符串无法转换为 Decimal，则结果为 Undefined 。虚数结果返回 Undefined 。

参数类型	结果
数组	Undefined .
对象	Undefined .
Null	Undefined .
未定义	Undefined .

exp(Decimal)

返回 e 的参数次方。在代入函数之前，Decimal 参数舍入到双精度。

示例：exp(1) = 1

参数类型	结果
Int	Decimal (双精度)，e ^ 参数。
Decimal	Decimal (双精度)，e ^ 参数。
String	Decimal (双精度)，e ^ 参数。如果 String 无法转换为 Decimal，则结果为 Undefined 。
其他值	Undefined .

ln(Decimal)

返回参数的自然对数。在代入函数之前，Decimal 参数舍入到双精度。

示例：ln(e) = 1

参数类型	结果
Int	Decimal (双精度)，参数的自然对数。
Decimal	Decimal (双精度)，参数的自然对数。

参数类型	结果
Boolean	Undefined .
String	Decimal (双精度)，参数的自然对数。如果字符串无法转换为 Decimal，则结果为 Undefined 。
数组	Undefined .
对象	Undefined .
Null	Undefined .
未定义	Undefined .

log(Decimal)

返回参数的以 10 为底的对数。在代入函数之前，Decimal 参数舍入到双精度。

示例：log(100) = 2.0

参数类型	结果
Int	Decimal (双精度)，参数以 10 为底的对数。
Decimal	Decimal (双精度)，参数以 10 为底的对数。
Boolean	Undefined .
String	Decimal (双精度)，参数以 10 为底的对数。如果 String 无法转换为 Decimal，则结果为 Undefined 。
数组	Undefined .
对象	Undefined .
Null	Undefined .

参数类型	结果
未定义	Undefined .

mod(Decimal, Decimal)

返回第一个参数除以第二个参数的余数。您还可以使用 % 作为相同取模功能的插入运算符。

示例：mod(8, 3) = 3

左侧操作数	右侧操作数	输出
Int	Int	Int，第一个参数对第二个参数取模。
Int / Decimal	Int / Decimal	Decimal，第一个参数对第二个参数取模。
String / Int / Decimal	String / Int / Decimal	如果所有字符串转换为 Decimals，则结果为第一个参数对第二个参数取模的值。否则为 Undefined 。
其他值	其他值	Undefined .

power(Decimal, Decimal)

返回第一个参数的第二个参数次幂的值。在代入函数之前，Decimal 参数舍入到双精度。

示例：power(2, 5) = 32.0

参数类型 1	参数类型 2	输出
Int / Decimal	Int / Decimal	Decimal (双精度)，返回第一个参数的第二个参数次幂的值。

参数类型 1	参数类型 2	输出
Int / Decimal / String	Int / Decimal / String	Decimal (双精度)，返回第一个参数的第二个参数次幂的值。所有字符串均转换为 Decimals。如果任何 String 无法转换为 Decimal，则结果为 Undefined 。
其他值	其他值	Undefined .

round(Decimal)

将给定的 Decimal 舍入到最近的 Int。如果 Decimal 与上下两个 Int 值距离相同 (例如 0.5)，Decimal 将向上进位。

示例：

Round(1.2) = 1

Round(1.5) = 2

Round(1.7) = 2

Round(-1.1) = -1

Round(-1.5) = -2

参数类型	结果
Int	参数
Decimal	Decimal 会向下舍入至最近的 Int。
String	Decimal 会向下舍入至最近的 Int。如果字符串无法转换为 Decimal，则结果为 Undefined 。
其他值	Undefined .

sign(Decimal)

返回给定数字的符号。当参数的符号为正时，将返回 1。当参数的符号为负时，将返回 -1。如果参数为 0，则返回 0。

示例：

`sign(-7) = -1`

`sign(0) = 0`

`sign(13) = 1`

参数类型	结果
Int	Int，Int 值的符号。
Decimal	Int，Decimal 值的符号。
String	Int，Decimal 值的符号。字符串将转换为 Decimal 值，并返回 Decimal 值的符号。如果 String 无法转换为 Decimal，则结果为 Undefined 。
其他值	Undefined .

sin(Decimal)

以弧度为单位返回数字的正弦值。在代入函数之前，Decimal 参数舍入到双精度。

示例：`sin(0) = 0.0`

参数类型	结果
Int	Decimal (双精度)，参数的正弦值。
Decimal	Decimal (双精度)，参数的正弦值。
Boolean	Undefined .

参数类型	结果
String	Decimal，参数的正弦值。如果字符串无法转换为 Decimal，则结果为 Undefined 。
Array	Undefined .
Object	Undefined .
Null	Undefined .
Undefined	Undefined .

sinh(Decimal)

以弧度为单位返回数字的双曲正弦值。在代入函数之前，Decimal 值舍入到双精度。结果是双精度的 Decimal 值。

示例：sinh(2.3) = 4.936961805545957

参数类型	结果
Int	Decimal (双精度)，参数的双曲正弦值。
Decimal	Decimal (双精度)，参数的双曲正弦值。
Boolean	Undefined .
String	Decimal，参数的双曲正弦值。如果字符串无法转换为 Decimal，则结果为 Undefined 。
Array	Undefined .
Object	Undefined .
Null	Undefined .
Undefined	Undefined .

sqrt(Decimal)

返回数字的平方根。在代入函数之前，Decimal 参数舍入到双精度。

示例：sqrt(9) = 3.0

参数类型	结果
Int	参数的平方根。
Decimal	参数的平方根。
Boolean	Undefined .
String	参数的平方根。如果字符串无法转换为 Decimal，则结果为 Undefined 。
Array	Undefined .
Object	Undefined .
Null	Undefined .
Undefined	Undefined .

tan(Decimal)

以弧度为单位返回数字的正切值。在代入函数之前，Decimal 值舍入到双精度。

示例：tan(3) = -0.1425465430742778

参数类型	结果
Int	Decimal (双精度)，参数的正切值。
Decimal	Decimal (双精度)，参数的正切值。
Boolean	Undefined .

参数类型	结果
String	Decimal (双精度)，参数的正切值。如果字符串无法转换为 Decimal，则结果为 Undefined 。
Array	Undefined .
Object	Undefined .
Null	Undefined .
Undefined	Undefined .

tanh(Decimal)

以弧度为单位返回数字的双曲正切值。在代入函数之前，Decimal 值舍入到双精度。

示例：tanh(2.3) = 0.9800963962661914

参数类型	结果
Int	Decimal (双精度)，参数的双曲正切值。
Decimal	Decimal (双精度)，参数的双曲正切值。
Boolean	Undefined .
String	Decimal (双精度)，参数的双曲正切值。如果字符串无法转换为 Decimal，则结果为 Undefined 。
Array	Undefined .
Object	Undefined .
Null	Undefined .
Undefined	Undefined .

trunc(Decimal, int)

按照第二个参数指定的 Decimal 位数截断第一个参数。如果第二个参数小于零，则会设置为零。如果参数第二大于 34，则会设置为 34。将从结果中删除结尾的零。

示例：

`trunc(2.3, 0)` = 2

`trunc(2.3123, 2)` = 2.31

`trunc(2.888, 2)` = 2.88

`trunc(2.00, 5)` = 2

参数类型 1	参数类型 2	结果
Int	Int	源值。
Int / Decimal / String	Int / Decimal	第一个参数被截断到由第二个参数所指定的长度。第二个参数如果不是 Int，将向下舍入至最近的 Int。所有字符串均转换为 Decimal 值。如果字符串转换失败，则结果为 Undefined 。
其他值		Undefined。

RunPipelineActivity

以下示例演示如何使用 RunPipelineActivity 命令测试管道活动。在本示例中，我们将测试数学活动：

1. 创建一个 maths.json 文件，其中包含您要测试的管道活动的定义。

```
{
    "math": {
        "name": "MyMathActivity",
```

```
        "math": "((temp - 32) * 5.0) / 9.0",
        "attribute": "tempC"
    }
}
```

2. 创建一个 payloads.json 文件，其中包含用于测试管道活动的示例负载。

```
[
    "{\"humidity\": 52, \"temp\": 68 }",
    "{\"humidity\": 52, \"temp\": 32 }"
]
```

3. 从命令行调用 RunPipelineActivities 操作。

```
aws iotanalytics run-pipeline-activity  --pipeline-activity file://maths.json  --
payloads file://payloads.json --cli-binary-format raw-in-base64-out
```

这会产生以下结果。

```
{
    "logResult": "",
    "payloads": [
        "eyJodW1pZGl0eSI6NTIsInRlbXAiOjY4LCJ0ZW1wQyI6MjB9",
        "eyJodW1pZGl0eSI6NTIsInRlbXAiOjMyLCJ0ZW1wQyI6MH0="
    ]
}
```

结果中列出的负载是 Base64 编码的字符串。对这些字符串解码时，您将获得以下结果。

```
{"humidity":52,"temp":68,"tempC":20}
{"humidity":52,"temp":32,"tempC":0}
```

重新处理通道消息

AWS IoT Analytics 使您能够重新处理通道数据。这在以下情况下很有用：

- 您希望重放现有已提取的数据，而不是重新开始。
- 您希望对管道进行更新，并希望引入带有更改的现有最新数据。
- 您希望包含在更改客户托管存储选项、通道权限或数据存储之前提取的数据。

参数

使用 AWS IoT Analytics 重新处理管道中的通道消息时，必须指定以下信息：

`StartPipelineReprocessing`

启动通过管道重新处理通道消息。

`ChannelMessages`

指定要重新处理的一组或多组通道消息。

如果使用 `channelMessages` 对象，则不得为 `startTime` 和 `endTime` 指定值。

`s3Paths`

指定一个或多个密钥，用于标识保存通道消息的 Amazon Simple Storage Service (Amazon S3)对象。您必须使用该密钥的完整路径。

路径示
例：`00:00:00/1582940490000_1582940520000_123456789012_mychannel_0_2118.0.`

类型：字符串数组

数组成员限制：1-100 项。

长度限制：1-1024 个字符。

`endTime`

重新处理的通道数据的结束时间 (不含)。

如果为 `endTime` 参数指定值，则不得使用 `channelMessages` 对象。

类型：时间戳

`startTime`

重新处理的原始消息数据的开始时间 (含)。

如果为 `startTime` 参数指定值，则不得使用 `channelMessages` 对象。

类型：时间戳

`pipelineName`

要开始重新处理的管道的名称。

类型：字符串

长度限制：1-128 个字符。

重新处理通道消息（控制台）

本教程介绍了如何在 AWS IoT Analytics 控制台中重新处理存储在指定 Amazon S3 对象中的通道数据。

在开始之前，请确保将要重新处理的通道消息保存在客户托管的 Amazon S3 存储桶中。

1. 登录到 AWS IoT Analytics 控制台。
2. 在导航窗格中，选择管道。
3. 选择您的目标管道。
4. 从操作中选择重新处理消息。
5. 在管道重新处理页面上，为重新处理消息选择 S3 对象。

   AWS IoT Analytics 控制台还提供以下选项：

   - 所有可用范围 - 重新处理通道中的所有有效数据。
   - 最近 120 天 - 重新处理最近 120 天到达的数据。
   - 最近 90 天 - 重新处理最近 90 天到达的数据。
   - 最近 30 天 - 重新处理最近 30 天到达的数据。
   - 自定义范围 - 重新处理在指定时间范围内到达的数据。您可以选择任何时间范围。
6. 输入用于存储通道消息的 Amazon S3 对象的密钥。

要查找密钥，请执行以下操作：

a. 前往 Amazon S3 控制台。
b. 选择目标 Amazon S3 对象。
c. 在属性下的对象概述部分中，复制密钥。

7. 选择开始重新处理。

重新处理通道消息(API)

在使用 `StartPipelineReprocessing` API 时，请注意以下几点：

- `startTime` 和 `endTime` 参数指定提取原始数据的时间，不过这些是粗略估计的时间。您可以舍入到最接近的小时。`startTime` 含本数，但 `endTime` 不含本数。
- 该命令异步启动重新处理并立即返回。
- 不保证重新处理消息的顺序与接收时的顺序相同。这大致相同，但不完全一致。
- 每 24 小时最多可发出 1000 个 `StartPipelineReprocessing` API 请求，用于通过管道重新处理相同的通道消息。
- 重新处理原始数据将产生额外的费用。

有关更多信息，请参阅 《AWS IoT Analytics API 参考》中的 StartPipelineReprocessing API。

取消通道重新处理活动

要取消管道重新处理活动，可使用 CancelPipelineReprocessing API，也可在 AWS IoT Analytics 控制台的活动页面上选择取消重新处理。如果取消重新处理，则不会重新处理剩余的数据。您必须启动另一个重新处理请求。

使用 DescribePipeline API 检查重新处理的状态。参阅响应中的 `reprocessingSummaries` 字段。

实现工作流程自动化

AWS IoT Analytics 为 AWS IoT 提供了高级数据分析。您可以使用数据分析和机器学习工具自动收集 IoT 数据，处理数据以及存储和分析数据。您可以执行托管您自己的自定义分析代码或 Jupyter Notebook 的容器，或者使用第三方自定义代码容器，以便无需重新创建现有的分析工具。您可以使用以下功能从数据存储中提取输入数据，并将其提供给自动化工作流程：

按照定期计划创建数据集内容

在调用 `CreateDataset` 时指定触发器以计划自动创建数据集内容（`triggers:schedule:expression`）。数据存储中的数据用于创建数据集内容。您可以使用 SQL 查询 (`actions:queryAction:sqlQuery`) 选择所需的字段。

定义非重叠的连续时间间隔，以确保新数据集内容仅包含在上次传送后到达的数据。使用 `actions:queryAction:filters:deltaTime` 和 `:offsetSeconds` 字段指定增量时间间隔。然后，指定一个触发器，以便在时间间隔结束时创建数据集内容。请参阅the section called “示例 6 - 创建具有增量时段的 SQL 数据集 (CLI)”。

在另一个数据集完成时创建数据集内容。

在另一个数据集的内容创建完成时触发创建新的数据集内容 `triggers:dataset:name`。

自动运行您的分析应用程序

将您自己的自定义数据分析应用程序容器化，并在创建另一个数据集的内容时触发以运行这些应用程序。这样，您就可以为应用程序提供按照定期计划创建的数据集内容中的数据。您可以从应用程序中自动对分析结果执行操作。(`actions:containerAction`)

在另一个数据集完成时创建数据集内容

在另一个数据集的内容创建完成时触发创建新的数据集内容 `triggers:dataset:name`。

自动运行您的分析应用程序

将您自己的自定义数据分析应用程序容器化，并在创建另一个数据集的内容时触发以运行这些应用程序。这样，您就可以为应用程序提供按照定期计划创建的数据集内容中的数据。您可以从应用程序中自动对分析结果执行操作。(`actions:containerAction`)

使用案例

自动执行产品质量测量以降低运营支出

您的系统具有可测量压力、湿度和温度的智能值。系统定期整理事件，并且还会在发生某些事件时整理事件，例如阀门打开和关闭时。通过使用 AWS IoT Analytics，您可以自动执行分析以聚合这些定期时段的非重叠数据，并创建有关最终产品质量的 KPI 报告。处理每个批次后，您可以测量总产品质量，并通过最大化运行卷来降低运营支出。

自动分析设备队列

您每 15 分钟对数百台设备生成的数据运行一次分析（算法、数据科学或 KPI 机器学习）。每个分析周期都会生成并存储下一次分析运行的状态。对于每个分析，您都希望只使用在指定时间窗口内收到的数据。借助 AWS IoT Analytics，您可以协调您的分析，为每次运行创建 KPI 和报告，然后存储数据以供未来的分析使用。

自动执行异常检测

AWS IoT Analytics 让您能够自动执行异常检测工作流程，否则您必须每 15 分钟对到达数据存储的新数据手动运行该检测。您还可以自动更新控制面板，其中显示在指定时间段内的设备使用情况和顶级用户。

预测工业过程的成果

您具有工业生产线。使用发送到 AWS IoT Analytics 的数据（包括可用的过程测量值），您可以运行分析工作流程来预测过程结果。用于模型的数据可在 M x N 矩阵中排列，其中每行包含执行实验室采样的各个时间点的数据。AWS IoT Analytics 通过创建增量窗口并使用数据科学工具创建 KPI 和保存测量设备的状态，帮助您实现分析工作流程操作化。

使用 Docker 容器。

本节包含有关如何构建您自己的 Docker 容器的信息。如果重复使用第三方构建的 Docker 容器，则存在安全风险：这些容器可能使用您的用户权限执行任意代码。在使用之前，请确保您信任任何第三方容器的作者。

使用以下步骤可设置针对自上次执行分析以来到达的数据的定期数据分析：

1. 创建一个 Docker 容器，其中包含您的数据应用程序以及任何必要的库或其他依赖项。

   IotAnalytics Jupyter 扩展提供了容器化 API 来协助容器化过程。您还可以运行自己创建内容的映像，其中您创建或组装了应用程序工具集，以执行所需的数据分析或计算。通过 AWS IoT

Analytics，您借助变量来定义容器化应用程序的输入数据源和 Docker 容器输出数据的目标。（自定义 Docker 容器输入/输出变量包含有关将变量与自定义容器结合使用的更多信息。）

2. 将容器上传到 Amazon ECR 注册表。
3. 创建数据存储，以接收并存储来自设备的消息（数据）(iotanalytics: CreateDatastore)
4. 创建消息发送通道(iotanalytics: CreateChannel)。
5. 创建用于将通道连接至数据存储的管道 (iotanalytics: CreatePipeline)。
6. 创建一个 IAM 角色来授予向 AWS IoT Analytics 通道发送数据的权限(iam: CreateRole.)
7. 创建一个 IoT 规则，以使用 SQL 查询将通道连接到消息数据源（iot: CreateTopicRule 字段 topicRulePayload:actions:iotAnalytics）。在设备通过 MQTT 发送具有相应主题的消息时，它将路由到您的通道。或者，您可以使用 iotanalytics: BatchPutMessage 从能够使用 AWS 开发工具包或 AWS CLI 的设备直接向通道中发送消息。
8. 创建一个 SQL 数据集，其创建由时间计划（iotanalytics: CreateDataset, 字段 actions: queryAction:sqlQuery）触发。

   您还可以指定要应用于消息数据的预筛选器，以帮助将消息限制为自上次执行操作以来到达的消息。（字段 actions:queryAction:filters:deltaTime:timeExpression 提供了可用于确定消息时间的表达式，而字段 actions:queryAction:filters:deltaTime:offsetSeconds 则指定了消息到达的可能延迟。）

   预筛选器和触发器计划共同确定了您的增量时段。每个新的 SQL 数据集是使用在上次创建 SQL 数据集后收到的消息创建的。（首次创建 SQL 数据集是如何实现的？根据计划和预筛选器估算上次创建数据集的时间。）
9. 创建由第一个 (CreateDataset 字段 trigger:dataset) 的创建所触发的另一个数据集。对于此数据集，您需指定一个指向您在第一步中创建的 Docker 容器并提供运行该 Docker 容器所需信息的“容器操作”（字段 actions:containerAction）。在这里，您还可以指定：

   - 您的账户中存储的 Docker 容器的 ARN (image)。
   - 角色的 ARN，该角色用于授予访问所需资源所需的系统权限，以便运行容器操作 (executionRoleArn)。
   - 执行容器操作的资源的配置 (resourceConfiguration)。
   - 用于执行容器操作的计算资源的类型（computeType，可能的值为：ACU_1 [vCPU=4, memory=16GiB] or ACU_2 [vCPU=8, memory=32GiB]）。
   - 用于执行容器操作的资源实例可以使用的持久性存储的大小（以 GB 为单位）(volumeSizeInGB)。

- 在应用程序的执行上下文中使用的变量值（基本上是传递给应用程序的参数）(`variables`)。

  在执行容器时，将替换这些变量。这样，您就可以使用在创建数据集内容时提供的不同变量（参数）来运行相同的容器。在容器化过程中，IotAnalytics Jupyter 扩展通过在笔记本中自动识别变量并将其标记为可用来简化此过程。您可以选择已识别的变量或添加自己的自定义变量。在运行容器之前，系统会将每个变量的值替换为执行时的当前值。
- 其中的一个变量是数据集的名称，数据集的最新内容将作为应用程序输入（这是您在上一步中创建的数据集的名称）(`datasetContentVersionValue:datasetName`)。

通过可生成数据集的 SQL 查询和增量窗口，以及应用程序的容器，AWS IoT Analytics 会创建一个计划的生产数据集，它以您指定的时间间隔对增量窗口中的数据运行，会生成所需的输出并发送通知。

您可以随时暂停您的生产数据集应用程序，然后再恢复。当您恢复生产数据集应用程序时，AWS IoT Analytics 在默认情况下将同步自上次执行以来已到达但尚未分析的所有数据。您还可通过执行一系列连续运行，来配置恢复生产数据集作业窗口长度的方式。或者，您也可以通过只捕获适合增量时段的指定大小的新到达数据来恢复生产数据集应用程序。

在创建或定义由其他数据集的创建所触发的数据集时，请注意以下限制：

- 只有容器数据集可由 SQL 数据集触发。
- 一个 SQL 数据集最多可以触发 10 个容器数据集。

在创建由 SQL 数据集触发的容器数据集时，可能会返回以下错误：

- “Triggering dataset can only be added on a container dataset”(只能在容器数据集中添加触发数据集)
- “There can only be one triggering dataset”(只能存在一个触发数据集)

  如果您尝试定义由两个不同 SQL 数据集触发的容器数据集，则会发生此错误。
- “The triggering data set <dataset-name> cannot be triggered by a container dataset”(触发数据集 <数据集名称> 不能被容器数据集触发)

  如果您尝试定义由其他容器数据集触发的容器数据集，则会发生此错误。
- “<N> datasets are already dependent on <dataset-name> dataset”(已经有 <N> 个数据集依赖于 <数据集名称> 数据集)

  如果您尝试定义的另一个容器数据集是由已触发 10 个容器数据集的 SQL 数据集触发的，则会出现该错误。

- “Exactly one trigger type should be provided”(应确切提供一种触发器类型)

  如果您尝试定义同时由计划触发器和数据集触发器触发的数据集，则会发生此错误。

自定义 Docker 容器的输入/输出变量

本节演示您的自定义 Docker 映像运行的程序如何读取输入变量并上传其输出。

参数文件

输入变量以及要将输出上传到的目的地存储在一个 JSON 文件中，它位于执行 Docker 映像的实例上的 `/opt/ml/input/data/iotanalytics/params` 中。下面是该文件的内容示例。

```
{
   "Context": {
       "OutputUris": {
           "html": "s3://aws-iot-analytics-dataset-xxxxxxx/notebook/results/
iotanalytics-xxxxxxx/output.html",
           "ipynb": "s3://aws-iot-analytics-dataset-xxxxxxx/notebook/results/
iotanalytics-xxxxxxx/output.ipynb"
       }
   },
   "Variables": {
       "source_dataset_name": "mydataset",
       "source_dataset_version_id": "xxxx",
       "example_var": "hello world!",
       "custom_output": "s3://aws-iot-analytics/dataset-xxxxxxx/notebook/results/
iotanalytics-xxxxxxx/output.txt"
   }
}
```

除了数据集的名称和版本 ID 外，`Variables` 部分还包含在 `iotanalytics:CreateDataset` 调用中指定的变量 - 在此示例中，为变量 `example_var` 给定值 `hello world!`。在 `custom_output` 变量中还提供了自定义输出 URI。该 `OutputUris` 字段包含容器可将其输出上传到的默认位置 - 在本示例中，同时提供了 ipynb 和 html 输出的默认输出 URI。

输入变量

通过 Docker 映像启动的程序可从 `params` 文件中读取变量。下面的示例程序将打开 `params` 文件、对文件进行分析并打印 `example_var` 变量的值。

```
import json
```

```
with open("/opt/ml/input/data/iotanalytics/params") as param_file:
    params = json.loads(param_file.read())
example_var = params["Variables"]["example_var"]
print(example_var)
```

上传输出

通过 Docker 映像启动的程序还可以将其输出存储在 Amazon S3 位置。必须使用“bucket-owner-full-control”访问控制列表加载输出。该访问列表将向 AWS IoT Analytics 服务授予对已上传输出的控制。在本示例中，我们对上一示例进行扩展，将 `example_var` 的内容上传至 `params` 文件中由 `custom_output` 定义的 Amazon S3 位置。

```
import boto3
import json
from urllib.parse import urlparse

ACCESS_CONTROL_LIST = "bucket-owner-full-control"

with open("/opt/ml/input/data/iotanalytics/params") as param_file:
    params = json.loads(param_file.read())
example_var = params["Variables"]["example_var"]

outputUri = params["Variables"]["custom_output"]
# break the S3 path into a bucket and key
bucket = urlparse(outputUri).netloc
key = urlparse(outputUri).path.lstrip("/")

s3_client = boto3.client("s3")
s3_client.put_object(Bucket=bucket, Key=key, Body=example_var, ACL=ACCESS_CONTROL_LIST)
```

权限

您必须创建两个 角色。一个角色用于授予启动 SageMaker 实例以实现笔记本容器化的权限。执行容器需要使用另一个角色。

您可以自动或手动创建第一个角色。如果您使用 AWS IoT Analytics 控制台创建新的 SageMaker 实例，那么您可以选择自动创建新角色，该角色将授予执行 SageMaker 实例和容器化笔记本所需的全部权限。或者，您也可以手动创建具有这些权限的角色。要手动创建，请创建一个已附加 `AmazonSageMakerFullAccess` 策略的角色并添加以下策略。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecr:BatchDeleteImage",
        "ecr:BatchGetImage",
        "ecr:CompleteLayerUpload",
        "ecr:CreateRepository",
        "ecr:DescribeRepositories",
        "ecr:GetAuthorizationToken",
        "ecr:InitiateLayerUpload",
        "ecr:PutImage",
        "ecr:UploadLayerPart"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": "arn:aws:s3:::iotanalytics-notebook-containers/*"
    }
  ]
}
```

您必须手动创建第二个角色，该角色将授予执行容器所需的权限。即使您使用 AWS IoT Analytics 控制台来自动创建第一个角色，也必须执行此操作。创建一个附加以下策略和信任策略的角色。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:PutObject",
                "s3:GetObject",
                "s3:PutObjectAcl"
            ],
            "Resource": "arn:aws:s3:::aws-*-dataset-*/*"
```

```
        },
        {
            "Effect": "Allow",
            "Action": [
                "iotanalytics:*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ecr:GetAuthorizationToken",
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage",
                "ecr:BatchCheckLayerAvailability",
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:DescribeLogStreams",
                "logs:GetLogEvents",
                "logs:PutLogEvents"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:ListBucket",
                "s3:ListAllMyBuckets"
            ],
            "Resource": "*"
        }
    ]
}
```

以下是信任策略的示例。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
```

```
      "Principal": {
        "Service": ["sagemaker.amazonaws.com", "iotanalytics.amazonaws.com"]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

通过 Java 和 AWS CLI 使用 CreateDataset API

创建数据集。数据集存储通过应用 queryAction（SQL 查询）或 containerAction（执行容器化应用程序）从数据存储中检索的数据。此操作创建数据集的骨架。该数据集可以通过调用 CreateDatasetContent 手动填充，也可以根据您指定的 trigger 自动填充。有关更多信息，请参阅 CreateDataset 和 CreateDatasetContent。

主题

- 示例 1 - 创建 SQL 数据集 (java)
- 示例 2 - 创建具有增量时段的 SQL 数据集 (Java)
- 示例 3 - 创建自带计划触发器的容器数据集 (Java)
- 示例 4 - 创建以 SQL 数据集作为触发器的容器数据集 (Java)
- 示例 5 - 创建 SQL 数据集 (CLI)
- 示例 6 - 创建具有增量时段的 SQL 数据集 (CLI)

示例 1 - 创建 SQL 数据集 (java)

```
CreateDatasetRequest request = new CreateDatasetRequest();
request.setDatasetName(dataSetName);
DatasetAction action = new DatasetAction();

//Create Action
action.setActionName("SQLAction1");
action.setQueryAction(new SqlQueryDatasetAction().withSqlQuery("select * from
 DataStoreName"));

// Add Action to Actions List
List<DatasetAction> actions = new ArrayList<DatasetAction>();
actions.add(action);
```

```
//Create Trigger
DatasetTrigger trigger = new DatasetTrigger();
trigger.setSchedule(new Schedule().withExpression("cron(0 12 * * ? *)"));

//Add Trigger to Triggers List
List<DatasetTrigger> triggers = new ArrayList<DatasetTrigger>();
triggers.add(trigger);

// Add Triggers and Actions to CreateDatasetRequest object
request.setActions(actions);
request.setTriggers(triggers);

// Add RetentionPeriod to CreateDatasetRequest object
request.setRetentionPeriod(new RetentionPeriod().withNumberOfDays(10));
final CreateDatasetResult result = iot.createDataset(request);
```

成功时的输出：

```
{DatasetName: <datatsetName>, DatasetArn: <datatsetARN>, RetentionPeriod: {unlimited:
 true} or {numberOfDays: 10, unlimited: false}}
```

示例 2 - 创建具有增量时段的 SQL 数据集 (Java)

```
CreateDatasetRequest request = new CreateDatasetRequest();
request.setDatasetName(dataSetName);
DatasetAction action = new DatasetAction();

//Create Filter for DeltaTime
QueryFilter deltaTimeFilter = new QueryFilter();
deltaTimeFilter.withDeltaTime(
                new DeltaTime()
                .withOffsetSeconds(-1 * EstimatedDataDelayInSeconds)
                .withTimeExpression("from_unixtime(timestamp)"));

//Create Action
action.setActionName("SQLActionWithDeltaTime");
action.setQueryAction(new SqlQueryDatasetAction()
                .withSqlQuery("SELECT * from DataStoreName")
                .withFilters(deltaTimeFilter));

// Add Action to Actions List
List<DatasetAction> actions = new ArrayList<DatasetAction>();
```

```
actions.add(action);

//Create Trigger
DatasetTrigger trigger = new DatasetTrigger();
trigger.setSchedule(new Schedule().withExpression("cron(0 12 * * ? *)"));

//Add Trigger to Triggers List
List<DatasetTrigger> triggers = new ArrayList<DatasetTrigger>();
triggers.add(trigger);

// Add Triggers and Actions to CreateDatasetRequest object
request.setActions(actions);
request.setTriggers(triggers);

// Add RetentionPeriod to CreateDatasetRequest object
request.setRetentionPeriod(new RetentionPeriod().withNumberOfDays(10));
final CreateDatasetResult result = iot.createDataset(request);
```

成功时的输出：

```
{DatasetName: <datatsetName>, DatasetArn: <datatsetARN>, RetentionPeriod: {unlimited:
 true} or {numberOfDays: 10, unlimited: false}}
```

示例 3 - 创建自带计划触发器的容器数据集 (Java)

```
CreateDatasetRequest request = new CreateDatasetRequest();
request.setDatasetName(dataSetName);
DatasetAction action = new DatasetAction();

//Create Action
action.setActionName("ContainerActionDataset");
action.setContainerAction(new ContainerDatasetAction()
        .withImage(ImageURI)
        .withExecutionRoleArn(ExecutionRoleArn)
        .withResourceConfiguration(
                new ResourceConfiguration()
                .withComputeType(new ComputeType().withAcu(1))
                .withVolumeSizeInGB(1))
        .withVariables(new Variable()
        .withName("VariableName")
        .withStringValue("VariableValue"));

// Add Action to Actions List
```

```
List<DatasetAction> actions = new ArrayList<DatasetAction>();
actions.add(action);

//Create Trigger
DatasetTrigger trigger = new DatasetTrigger();
trigger.setSchedule(new Schedule().withExpression("cron(0 12 * * ? *)"));

//Add Trigger to Triggers List
List<DatasetTrigger> triggers = new ArrayList<DatasetTrigger>();
triggers.add(trigger);

// Add Triggers and Actions to CreateDatasetRequest object
request.setActions(actions);
request.setTriggers(triggers);

// Add RetentionPeriod to CreateDatasetRequest object
request.setRetentionPeriod(new RetentionPeriod().withNumberOfDays(10));
final CreateDatasetResult result = iot.createDataset(request);
```

成功时的输出：

```
{DatasetName: <datatsetName>, DatasetArn: <datatsetARN>, RetentionPeriod: {unlimited:
 true} or {numberOfDays: 10, unlimited: false}}
```

示例 4 - 创建以 SQL 数据集作为触发器的容器数据集 (Java)

```
CreateDatasetRequest request = new CreateDatasetRequest();
request.setDatasetName(dataSetName);
DatasetAction action = new DatasetAction();

//Create Action
action.setActionName("ContainerActionDataset");
action.setContainerAction(new ContainerDatasetAction()
        .withImage(ImageURI)
        .withExecutionRoleArn(ExecutionRoleArn)
        .withResourceConfiguration(
                new ResourceConfiguration()
                .withComputeType(new ComputeType().withAcu(1))
                .withVolumeSizeInGB(1))
        .withVariables(new Variable()
        .withName("VariableName")
        .withStringValue("VariableValue"));
```

```
// Add Action to Actions List
List<DatasetAction> actions = new ArrayList<DatasetAction>();
actions.add(action);

//Create Trigger
DatasetTrigger trigger = new DatasetTrigger()
        .withDataset(new TriggeringDataset()
                .withName(TriggeringSQLDataSetName));

//Add Trigger to Triggers List
List<DatasetTrigger> triggers = new ArrayList<DatasetTrigger>();
triggers.add(trigger);

// Add Triggers and Actions to CreateDatasetRequest object
request.setActions(actions);
request.setTriggers(triggers);
final CreateDatasetResult result = iot.createDataset(request);
```

成功时的输出：

```
{DatasetName: <datatsetName>, DatasetArn: <datatsetARN>}
```

示例 5 - 创建 SQL 数据集 (CLI)

```
aws iotanalytics --endpoint <EndPoint>  --region <Region> create-dataset --dataset-
name="<dataSetName>" --actions="[{\"actionName\":\"<ActionName>\", \"queryAction\":
{\"sqlQuery\":\"<SQLQuery>\"}}]" --retentionPeriod numberOfDays=10
```

成功时的输出：

```
{
    "datasetName": "<datasetName>",
    "datasetArn": "<datatsetARN>",
    "retentionPeriod": {unlimited: true} or {numberOfDays: 10, unlimited: false}
}
```

示例 6 - 创建具有增量时段的 SQL 数据集 (CLI)

增量时段是一系列用户定义的不重叠且连续的时间间隔。通过使用增量时段，您可以使用在上次分析后到达数据存储的新数据创建数据集内容，并对新数据执行分析。您可在数据集 `queryAction` 的 `filters` 部分中，通过设置 `deltaTime` 来创建增量时段 (CreateDataset)。通常，您还希望设置时间

间隔触发器以自动创建数据集内容 (triggers:schedule:expression)。基本上，这使您可以筛选在特定时间窗口中到达的消息，这样来自以前时间窗口的消息中包含的数据不会重复计数。

在该示例中，我们创建一个新数据集，它仅使用在上次传送后到达的数据每隔 15 分钟自动创建新的数据集内容。我们指定 3 分钟（180 秒）的 deltaTime 偏差，以允许到达指定数据存储的消息延迟 3 分钟。因此，如果数据集内容是在上午 10:30 创建的，则使用的数据（包含在数据集内容中）的时间戳在上午 10:12 到上午 10:27 之间（即上午 10:30 - 15 分钟 - 3 分钟到上午 10:30 - 3 分钟）。

```
aws iotanalytics --endpoint <EndPoint>  --region <Region> create-dataset --cli-input-
json file://delta-window.json
```

delta-window.json 文件包含以下内容。

```
{
  "datasetName": "delta_window_example",
  "actions": [
    {
      "actionName": "delta_window_action",
      "queryAction": {
        "sqlQuery": "SELECT temperature, humidity, timestamp FROM my_datastore",
        "filters": [
          {
            "deltaTime": {
              "offsetSeconds": -180,
              "timeExpression": "from_unixtime(timestamp)"
            }
          }
        ]
      }
    }
  ],
  "triggers": [
    {
      "schedule": {
        "expression": "cron(0/15 * * * ? *)"
      }
    }
  ]
}
```

成功时的输出：

```
{
    "datasetName": "<datasetName>",
    "datasetArn": "<datatsetARN>",
}
```

容器化笔记本

本节包含有关如何使用 Jupyter notebook 构建 Docker 容器的信息。如果重复使用第三方构建的笔记本，则存在安全风险：包含的容器可能使用您的用户权限执行任意代码。此外，笔记本生成的 HTML 可能显示在 AWS IoT Analytics 控制台中，从而在显示 HTML 的计算机上提供潜在的攻击媒介。在使用之前，请确保您信任任何第三方笔记本的作者。

执行高级分析函数的一个选项是使用 Jupyter 笔记本。Jupyter Notebook 提供强大的数据科学工具，可以执行机器学习和一系列统计分析。有关更多信息，请参阅笔记本模板。（请注意，我们目前在 JupyterLab 中不支持容器化。）您可以将 Jupyter Notebook 和库打包到一个容器中，当 AWS IoT Analytics 在您定义的增量时间窗口内收到一批新数据时，该库将针对新数据定期运行。您可以计划使用容器和在指定时间窗口中捕获的新分段数据的分析作业，然后存储该作业的输出以用于以后的计划分析。

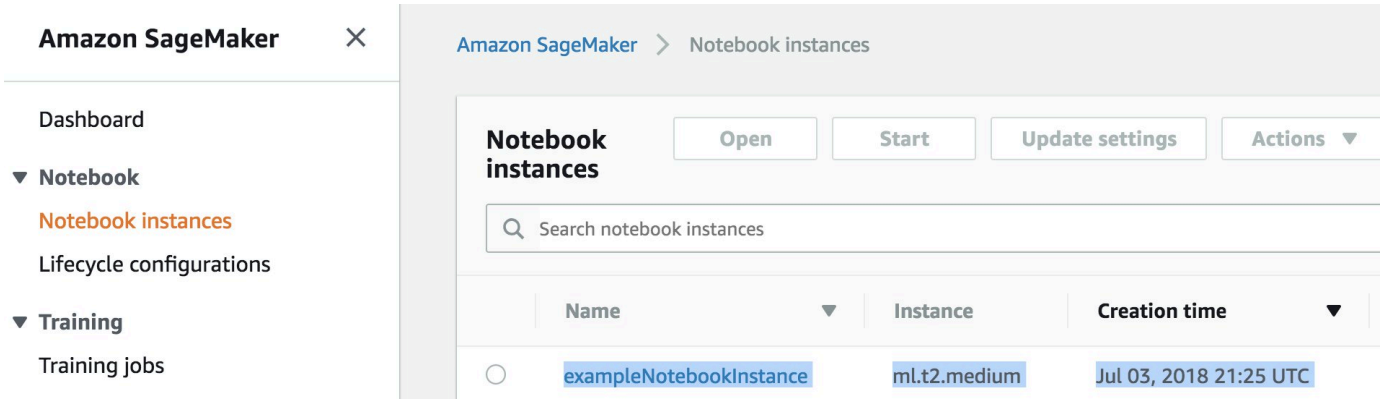
如果您在 2018 年 8 月 23 日以后使用AWS IoT Analytics 控制台创建了 SageMaker 实例，则会自动为您安装容器化扩展，并且您可以开始创建容器化映像。否则，请按照本节中列出的步骤来对 SageMaker 实例启用笔记本容器化。接下来，您修改 SageMaker 执行角色，以允许将容器映像上传到 Amazon EC2 并安装容器化扩展。

对不是通过 AWS IoT Analytics 控制台创建的笔记本实例启用容器化

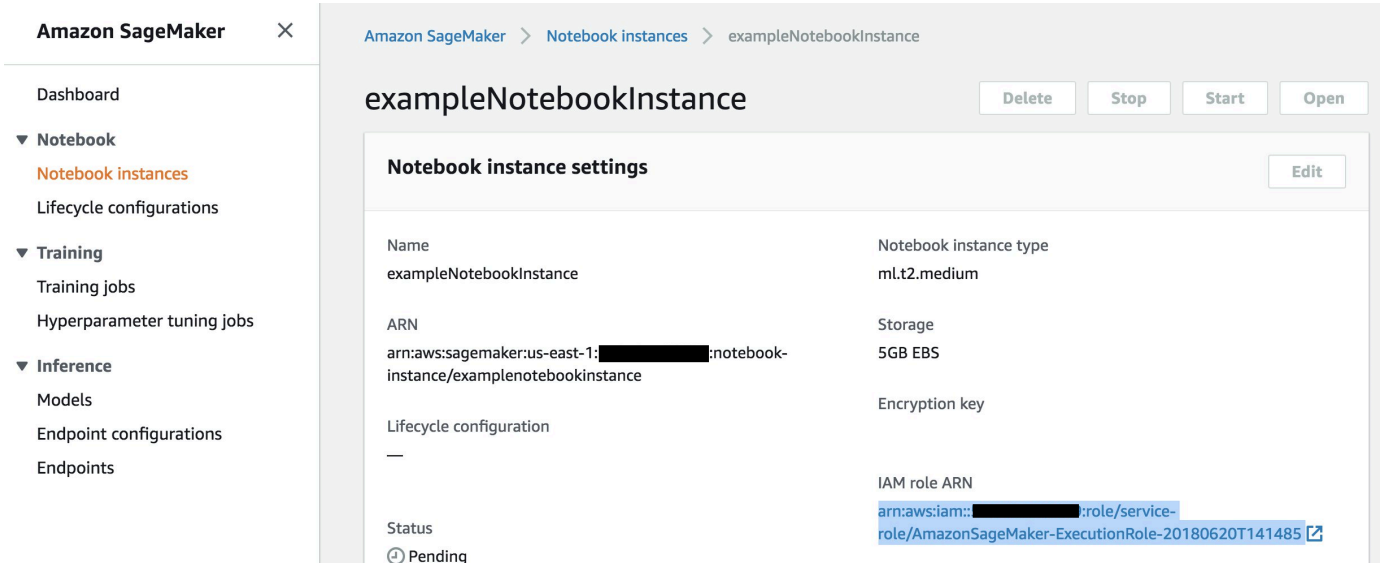
建议您通过 AWS IoT Analytics 控制台而非以下步骤创建新的 SageMaker 实例。新实例将自动支持容器化。

如果在启用容器化后重新启动 SageMaker 实例（如此处所示），则不必重新添加 IAM 角色和策略，但必须重新安装扩展，如最后一步中所示。

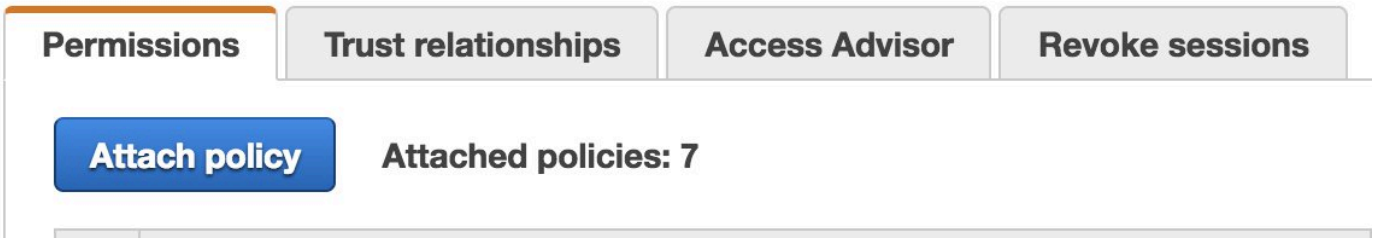
1. 要向笔记本实例授予对 Amazon ECS 的访问权限，请在 SageMaker 页面上选择您的 SageMaker 实例：



2. 在 IAM 角色 ARN 下，选择 SageMaker 执行角色。

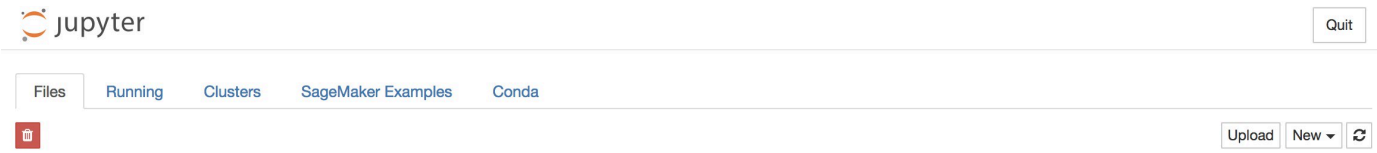


3. 选择 Attach Policy (附加策略)，然后定义并附加权限中显示的策略。如果尚未附加 `AmazonSageMakerFullAccess` 策略，则附加策略。



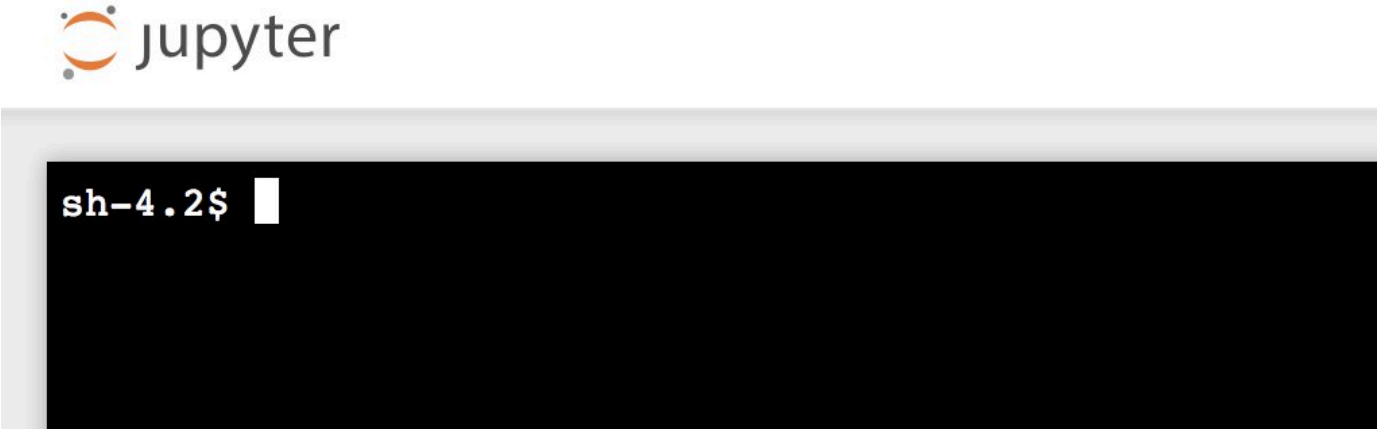
您还必须从 Amazon S3 下载容器化代码并将其安装在您的笔记本实例中，第一步是访问 SageMaker 实例的终端。

1. 在 Jupyter 中，选择 新建。



2. 在显示的菜单中，选择 终端。

3. 在终端内，输入以下命令来下载、解压缩和安装代码。请注意，这些命令终止您的笔记本在该 SageMaker 实例上运行的任何进程。




```
cd /tmp

aws s3 cp s3://iotanalytics-notebook-containers/iota_notebook_containers.zip /tmp

unzip iota_notebook_containers.zip

cd iota_notebook_containers

chmod u+x install.sh

./install.sh
```

等待一两分钟，以便对扩展进行验证和安装。

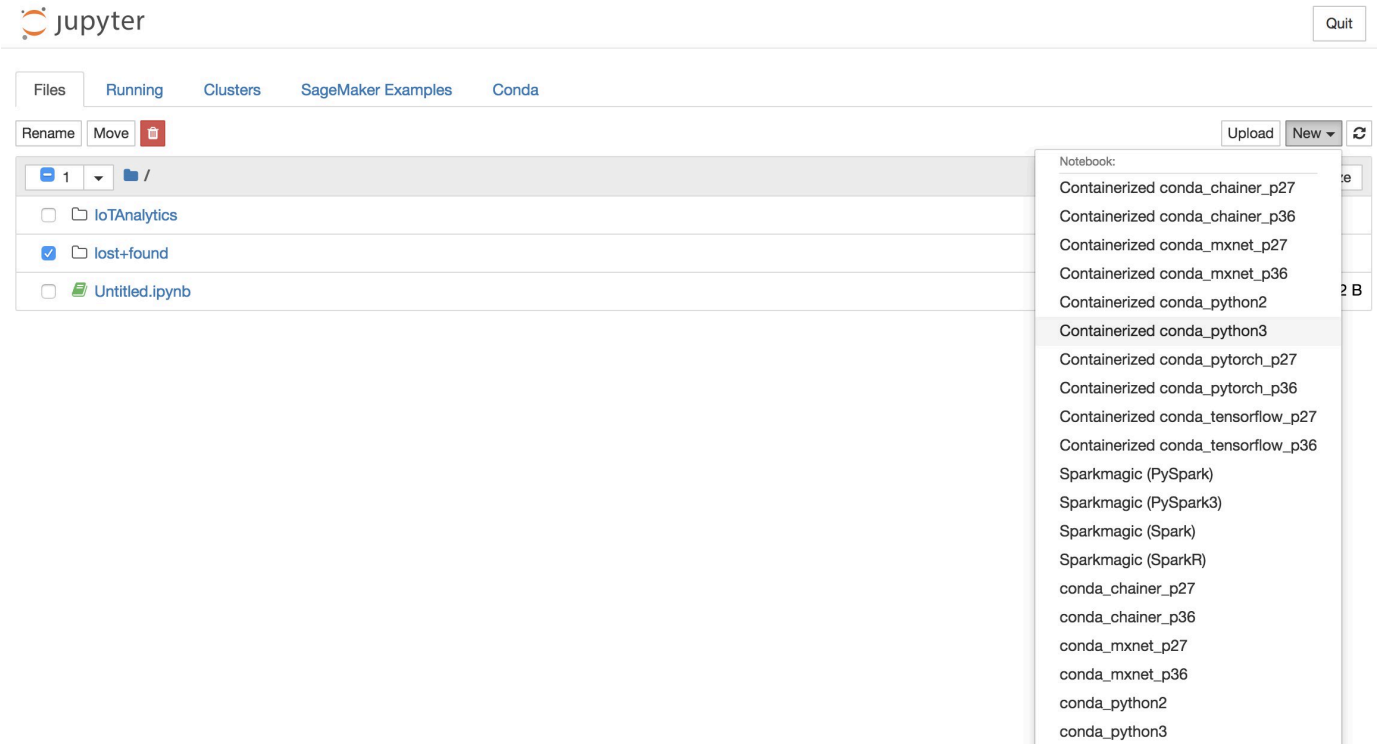
更新笔记本容器化扩展

如果您在 2018 年 8 月 23 日之后使用 AWS IoT Analytics 控制台创建了 SageMaker 实例，则已自动安装容器化扩展。您可以通过从 SageMaker 控制台重新启动实例来更新扩展。如果您是手动安装的扩展，则可通过重新运行“对不是通过 AWS IoT Analytics 控制台创建的笔记本实例启用容器化”中列出的命令来更新该扩展。

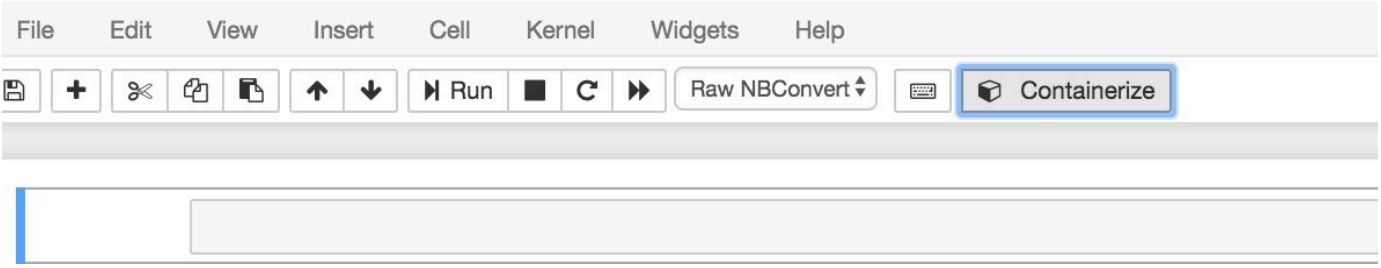
创建容器化映像

在本节中，我们将演示实现笔记本容器化所需的步骤。首先，请转至您的 Jupyter 笔记本，创建具有容器化内核的笔记本。

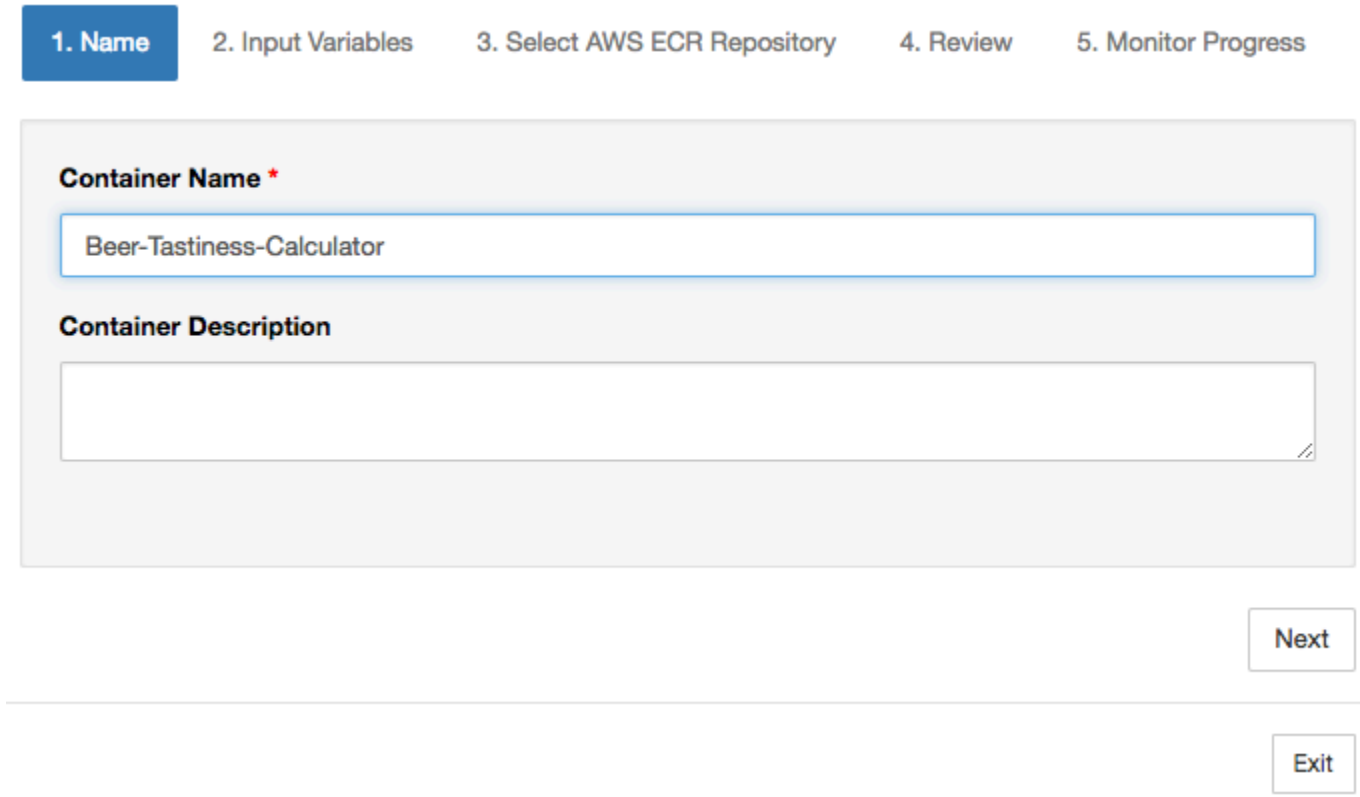
1. 在您的 Jupyter 笔记本中，选择 New (新建)，然后从下拉列表中选择所需内核类型。（内核类型应以 “Containerized” 开头，并以原本要选择的任何内核结尾。例如，如果只要一个像“conda_python3”这样的纯 Python 3.0 环境，请选择“Containerized conda_python3”）。



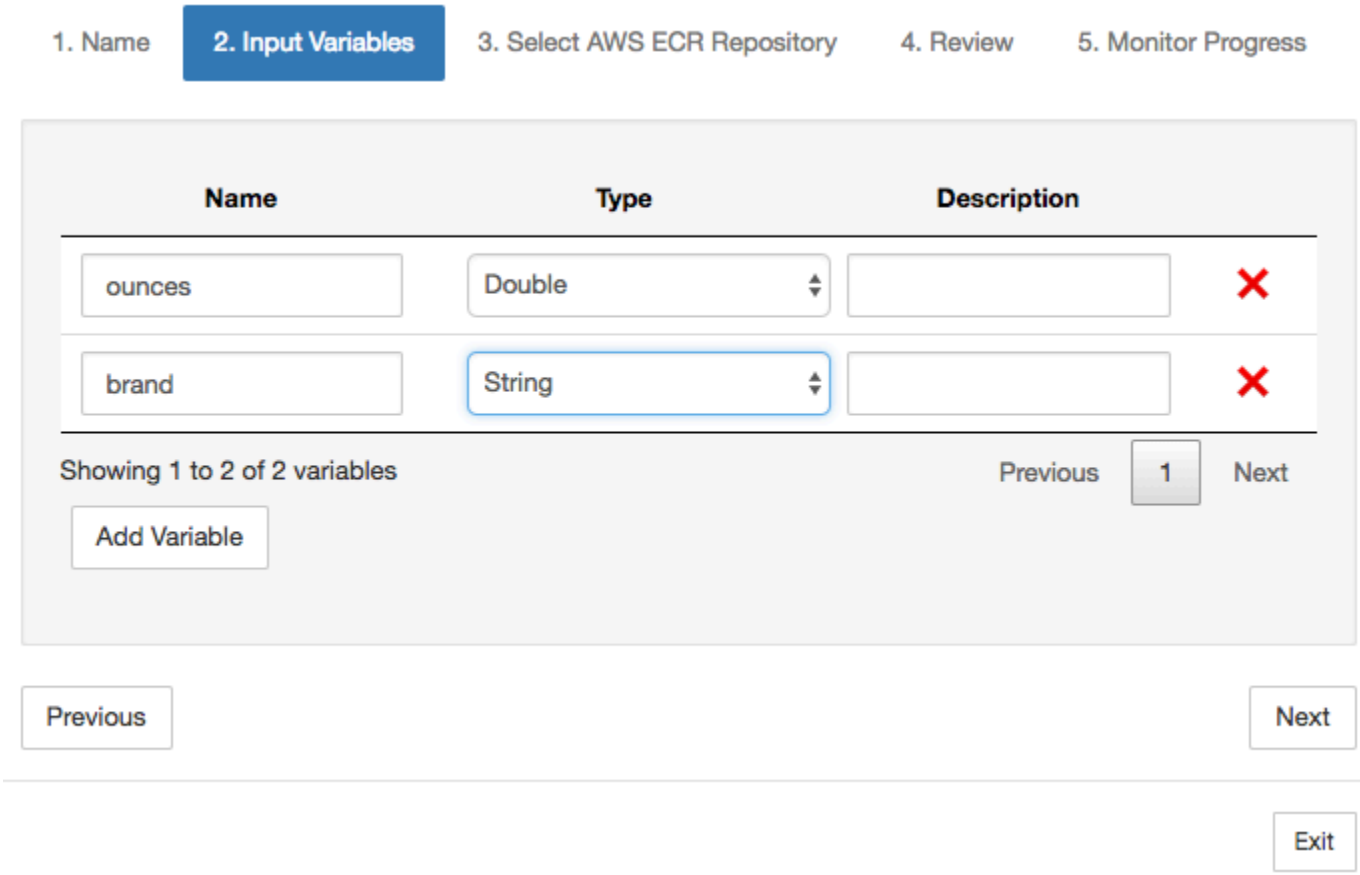
2. 在完成笔记本中的工作并且想要将其容器化时，请选择 容器化 按钮：



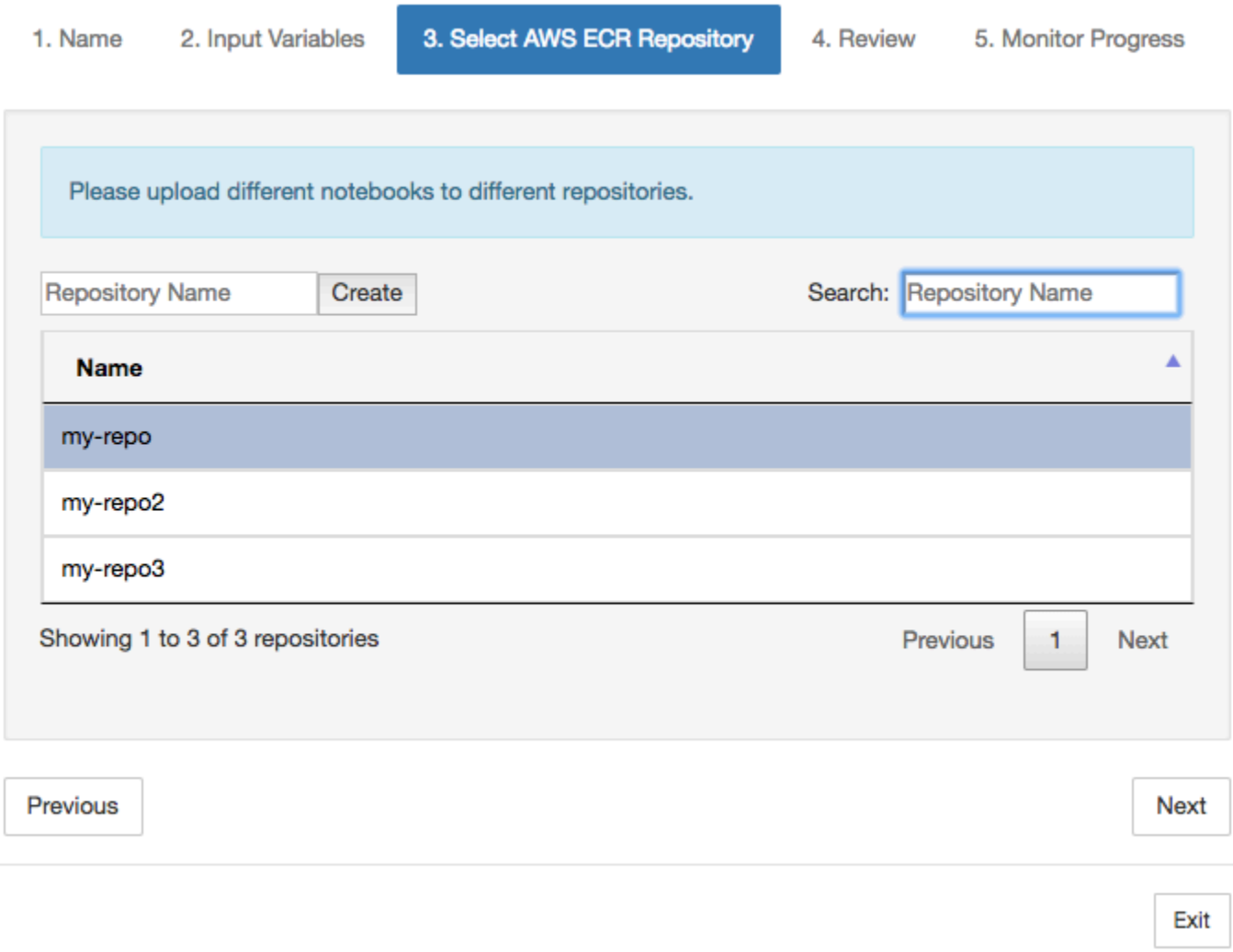
3. 输入容器化笔记本的名称。您也可以输入可选描述。



4. 指定应用于调用笔记本的 Input Variables (输入变量) (参数) 。您可以选择从笔记本中自动检测到的输入变量，或定义自定义变量。（请注意，仅当之前已执行过笔记本时，才能检测到输入变量。）对于每个输入变量，选择一种类型。您还可以输入输入变量的可选描述。

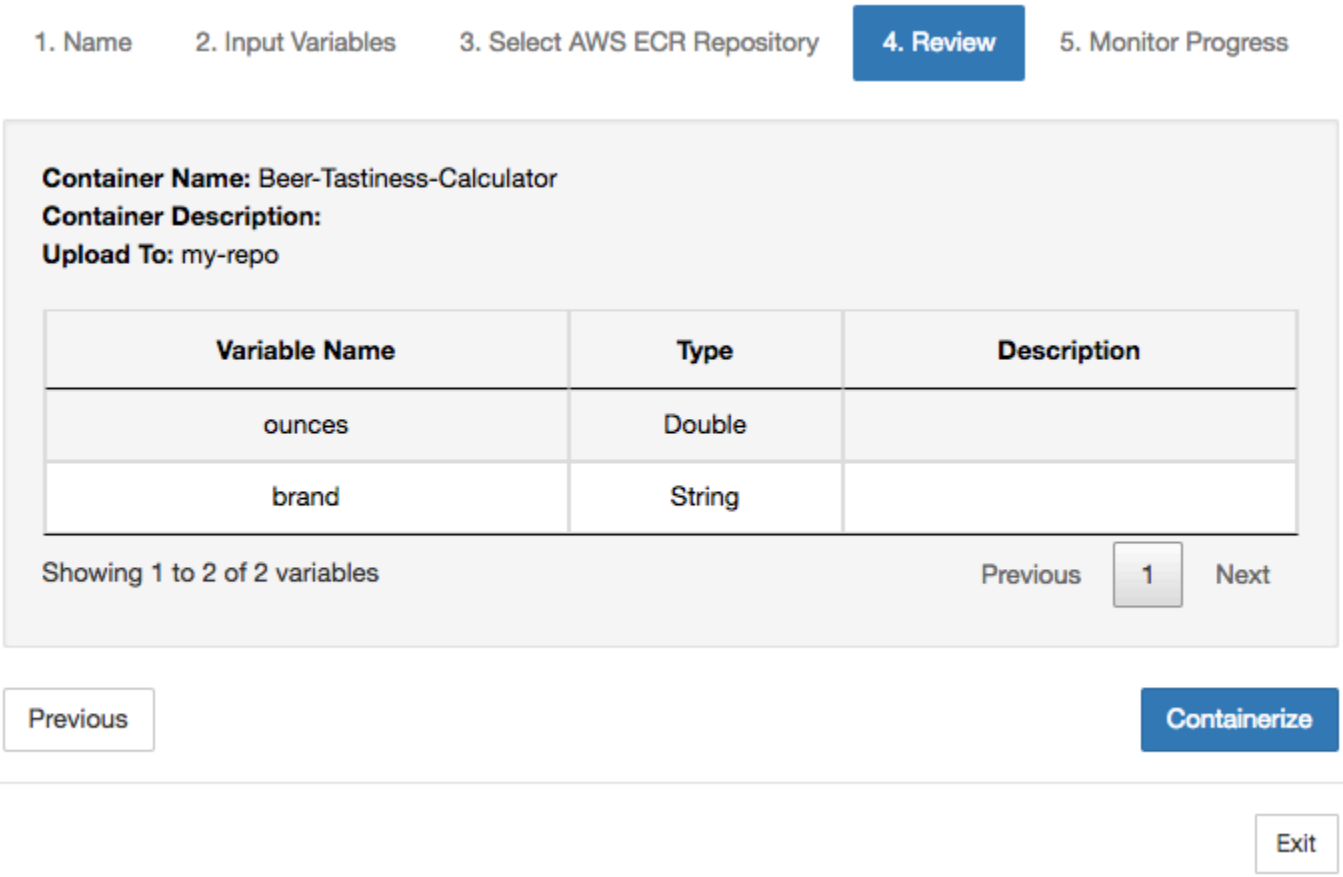


5. 选择应将从笔记本中创建的映像上传到的 Amazon ECR 存储库。

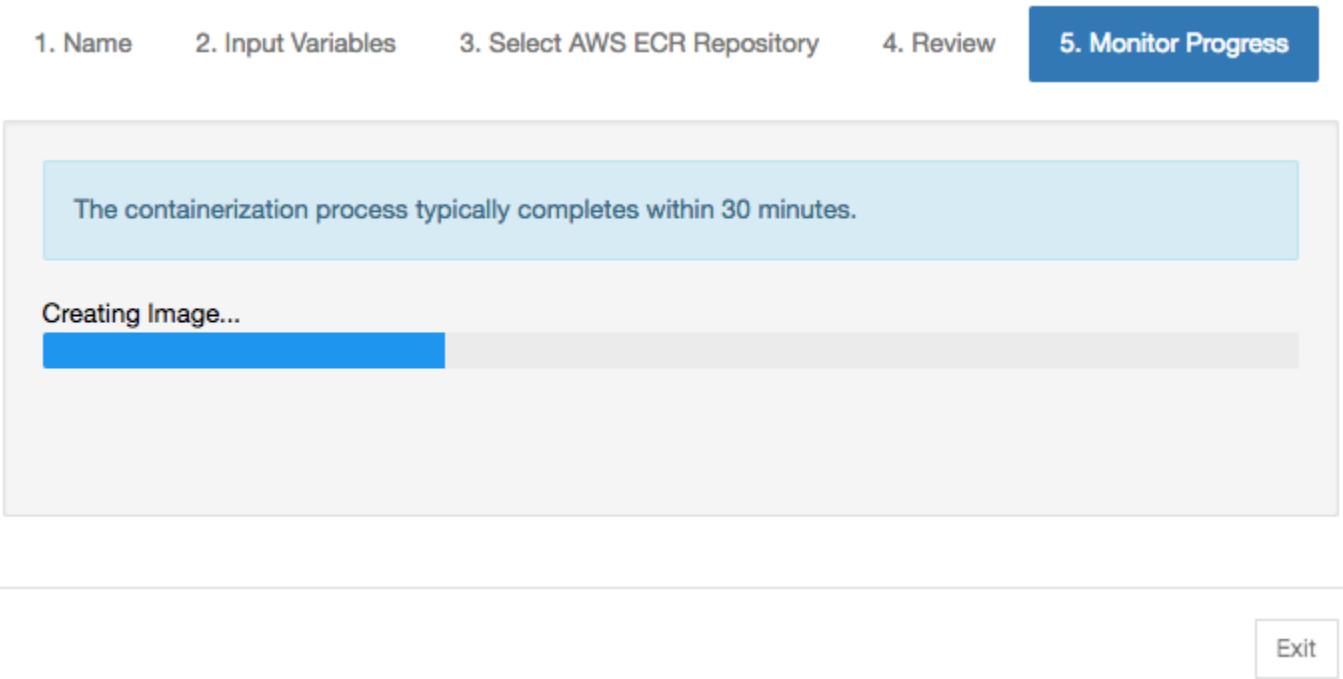


6. 选择 容器化 开始容器化过程。

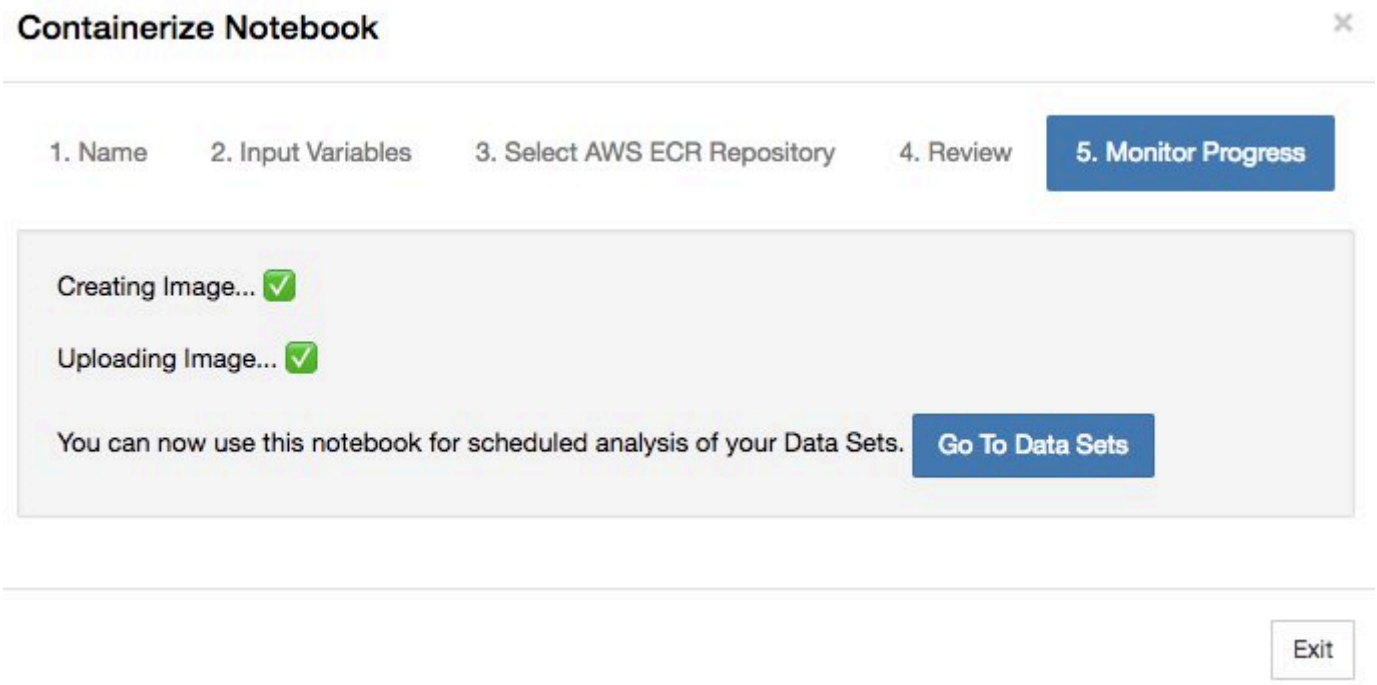
   您将看到汇总输入情况的概述。请注意，容器化过程启动后便无法取消。该过程可能长达一小时。


1. Name
2. Input Variables
3. Select AWS ECR Repository
4. Review
5. Monitor Progress
Container Name: Beer-Tastiness-Calculator
Container Description:
Upload To: my-repo
Variable Name
Type
Description
ounces
Double
brand
String
Showing 1 to 2 of 2 variables
Previous
1
Next
Previous
Containerize
Exit

7. 下一页将显示进度。


1. Name
2. Input Variables
3. Select AWS ECR Repository
4. Review
5. Monitor Progress
The containerization process typically completes within 30 minutes.
Creating Image...
Exit

8. 如果您意外关闭了浏览器，则可从 AWS IoT Analytics 控制台的笔记本部分监视容器化过程的状态。
9. 该过程完成后，容器化映像将存储在 Amazon ECR 中，随时可供使用。



使用自定义容器进行分析

本节包含有关如何使用 Jupyter notebook 构建 Docker 容器的信息。如果重复使用第三方构建的笔记本，则存在安全风险：包含的容器可能使用您的用户权限执行任意代码。此外，笔记本生成的 HTML 可能显示在 AWS IoT Analytics 控制台中，从而在显示 HTML 的计算机上提供潜在的攻击媒介。在使用之前，请确保您信任任何第三方笔记本的作者。

您可以创建自己的自定义容器，然后使用 AWS IoT Analytics 服务运行该容器。为此，您需要安装 Docker 映像并将其上传到 Amazon ECR，然后设置数据集来运行容器操作。本节提供一个使用 Octave 的过程示例。

本教程假定：

- 本地计算机上已安装 Octave
- 本地计算机上已设置 Docker 账户
- 具有 Amazon ECR 或 AWS IoT Analytics 访问权限的 AWS 账户

步骤 1：设置 Docker 映像

本教程需要三个主要文件。其名称和内容如下：

- `Dockerfile` - 用于 Docker 容器化过程的初始设置。

```
FROM ubuntu:16.04

# Get required set of software
RUN apt-get update
RUN apt-get install -y software-properties-common
RUN apt-get install -y octave
RUN apt-get install -y python3-pip

# Get boto3 for S3 and other libraries
RUN pip3 install --upgrade pip
RUN pip3 install boto3
RUN pip3 install urllib3

# Move scripts over
ADD moment moment
ADD run-octave.py run-octave.py

# Start python script
ENTRYPOINT ["python3", "run-octave.py"]
```

- `run-octave.py` – 对来自 AWS IoT Analytics 的 JSON 进行分析，运行 Octave 脚本，并将项目上传到 Amazon S3。

```
import boto3
import json
import os
import sys
from urllib.parse import urlparse

# Parse the JSON from IoT Analytics
with open('/opt/ml/input/data/iotanalytics/params') as params_file:
    params = json.load(params_file)

variables = params['Variables']

order = variables['order']
input_s3_bucket = variables['inputDataS3BucketName']
```

```
input_s3_key = variables['inputDataS3Key']
output_s3_uri = variables['octaveResultS3URI']

local_input_filename = "input.txt"
local_output_filename = "output.mat"

# Pull input data from S3...
s3 = boto3.resource('s3')
s3.Bucket(input_s3_bucket).download_file(input_s3_key, local_input_filename)

# Run Octave Script
os.system("octave moment {} {} {}".format(local_input_filename,
 local_output_filename, order))

# # Upload the artifacts to S3
output_s3_url = urlparse(output_s3_uri)
output_s3_bucket = output_s3_url.netloc
output_s3_key = output_s3_url.path[1:]

s3.Object(output_s3_bucket, output_s3_key).put(Body=open(local_output_filename,
 'rb'), ACL='bucket-owner-full-control')
```

- moment - 一个简单的 Octave 脚本，可根据输入或输出文件和指定的顺序计算时刻。

```
#!/usr/bin/octave -qf

arg_list = argv ();
input_filename = arg_list{1};
output_filename = arg_list{2};
order = str2num(arg_list{3});

[D,delimiterOut]=importdata(input_filename)
M = moment(D, order)

save(output_filename,'M')
```

1. 下载每个文件的内容。创建一个新目录，并将所有文件都放入该目录，然后cd到该目录。
2. 运行以下命令。

```
docker build -t octave-moment .
```

3. 您应该会在 Docker 存储库中看到一个新映像。通过运行以下命令验证它。

```
docker image ls | grep octave-moment
```

步骤 2：将 Docker 映像上传到 Amazon ECR 存储库

1. 创建 Amazon ECR 存储库。

```
aws ecr create-repository --repository-name octave-moment
```

2. 获取 Docker 环境的登录信息。

```
aws ecr get-login
```

3. 复制输出并运行它。输出应与以下内容类似。

```
docker login -u AWS -p password -e none https://your-aws-account-
id.dkr.ecr..amazonaws.com
```

4. 用 Amazon ECR 存储库标签标记您创建的映像。

```
docker tag your-image-id  your-aws-account-id.dkr.ecr.region.amazonaws.com/octave-
moment
```

5. 将映像推送到 Amazon ECR

```
docker push your-aws-account-id.dkr.ecr.region.amazonaws.com/octave-moment
```

步骤 3：将示例数据上传到 Amazon S3 存储桶

1. 将以下内容下载至文件`input.txt`。

```
0.857549  -0.987565  -0.467288  -0.252233  -2.298007
 0.030077  -1.243324  -0.692745   0.563276   0.772901
-0.508862  -0.404303  -1.363477  -1.812281  -0.296744
-0.203897   0.746533   0.048276   0.075284   0.125395
 0.829358   1.246402  -1.310275  -2.737117   0.024629
 1.206120   0.895101   1.075549   1.897416   1.383577
```

2. 创建名称为 `octave-sample-data-your-aws-account-id` 的 Amazon S3 存储桶。

3. 将文件 `input.txt` 上传到您刚刚创建的 Amazon S3 存储桶。现在，您应该有了一个名为 `octave-sample-data-*your-aws-account-id*` 的存储桶，其中包含文件 `input.txt`。

步骤 4：创建容器执行角色

1. 将以下内容复制到名为 `role1.json` 的文件中。将 *`your-aws-account-id`* 替换为您的账户 AWS ID，将 *`aws-region`* 替换为您的 AWS 资源 AWS 区域。

> **Note**
>
> 此示例包括一个全局条件上下文密钥，用于防止混淆代理安全问题。有关更多信息，请参阅the section called “跨服务混淆代理问题防范”。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "sagemaker.amazonaws.com",
                    "iotanalytics.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole",
            "Condition": {
              "StringEquals": {
                "aws:SourceAccount": "your-aws-account-id"
              },
              "ArnLike": {
                "aws:SourceArn": "arn:aws:iotanalytics:aws-region:your-aws-account-
id:dataset/DOC-EXAMPLE-DATASET"
              }
            }
    ]
}
```

2. 使用您下载的 `role1.json` 文件，创建一个向 SageMaker 和 AWS IoT Analytics 授予访问权限的角色。

```
aws iam create-role --role-name container-execution-role --assume-role-policy-
document file://role1.json
```

3. 将以下内容下载到名为 policy1.json 的文件，并将 *your-account-id* 替换为您的账户 ID（请参阅 Statement:Resource 下的第二个 ARN）。

```
{
 "Version": "2012-10-17",
 "Statement": [
   {
     "Effect": "Allow",
     "Action": [
       "s3:GetBucketLocation",
       "s3:PutObject",
       "s3:GetObject",
       "s3:PutObjectAcl"
     ],
     "Resource": [
       "arn:aws:s3:::*-dataset-*/*",
       "arn:aws:s3:::octave-sample-data-your-account-id/*"
   },
   {
     "Effect": "Allow",
     "Action": [
       "iotanalytics:*"
     ],
     "Resource": "*"
   },
   {
     "Effect": "Allow",
     "Action": [
       "ecr:GetAuthorizationToken",
       "ecr:GetDownloadUrlForLayer",
       "ecr:BatchGetImage",
       "ecr:BatchCheckLayerAvailability",
       "logs:CreateLogGroup",
       "logs:CreateLogStream",
       "logs:DescribeLogStreams",
       "logs:GetLogEvents",
       "logs:PutLogEvents"
     ],
     "Resource": "*"
```

```
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketLocation",
        "s3:ListBucket",
        "s3:ListAllMyBuckets"
      ],
      "Resource" : "*"
    }
  ]
}
```

4. 使用您刚刚下载的文件 `policy.json` 创建一个 IAM policy。

```
aws iam create-policy --policy-name ContainerExecutionPolicy --policy-document
 file://policy1.json
```

5. 将 策略附加到该角色。

```
aws iam attach-role-policy --role-name container-execution-role --policy-arn
 arn:aws:iam::your-account-id:policy/ContainerExecutionPolicy
```

步骤 5：使用容器操作创建一个数据集

1. 将以下内容下载到名为 `cli-input.json` 的文件中，并用相应的值替换 *`your-account-id`* 和 *`region`* 的所有实例。

```
{
    "datasetName": "octave_dataset",
    "actions": [
        {
            "actionName": "octave",
            "containerAction": {
                "image": "your-account-id.dkr.ecr.region.amazonaws.com/octave-
moment",
                "executionRoleArn": "arn:aws:iam::your-account-id:role/container-
execution-role",
                "resourceConfiguration": {
                    "computeType": "ACU_1",
                    "volumeSizeInGB": 1
                },
```

```
                "variables": [
                    {
                        "name": "octaveResultS3URI",
                        "outputFileUriValue": {
                            "fileName": "output.mat"
                        }
                    },
                    {
                        "name": "inputDataS3BucketName",
                        "stringValue": "octave-sample-data-your-account-id"
                    },
                    {
                        "name": "inputDataS3Key",
                        "stringValue": "input.txt"
                    },
                    {
                        "name": "order",
                        "stringValue": "3"
                    }
                ]
            }
        }
    ]
}
```

2. 使用您刚刚下载并编辑的文件 `cli-input.json` 创建一个数据集。

```
aws iotanalytics create-dataset –cli-input-json file://cli-input.json
```

步骤 6：调用数据集内容生成

1. 运行以下命令。

```
aws iotanalytics create-dataset-content --dataset-name octave-dataset
```

步骤 7：获取数据集内容

1. 运行以下命令。

```
aws iotanalytics get-dataset-content --dataset-name octave-dataset --version-id \
$LATEST
```

2. 您可能需要等待几分钟时间，直到 DatasetContentState 为 SUCCEEDED。

步骤 8：在 Octave 中打印输出

1. 使用 Octave shell 通过运行以下命令从容器中打印输出。

```
bash> octave
octave> load output.mat
octave> disp(M)
-0.016393 -0.098061 0.380311 -0.564377 -1.318744
```

可视化 AWS IoT Analytics 数据

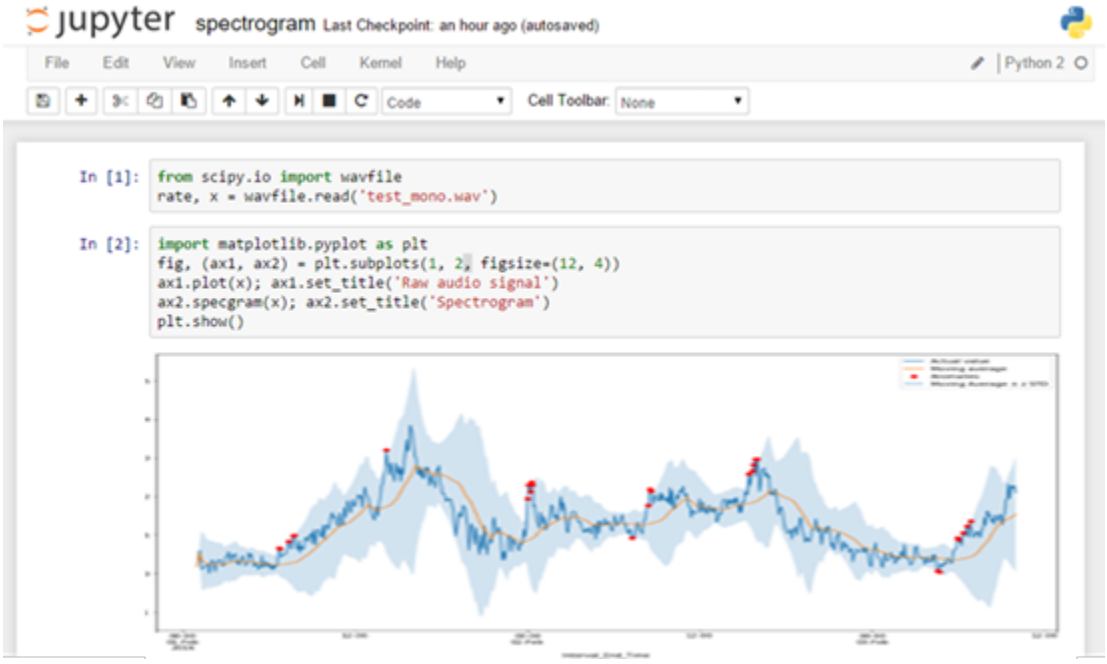
要将 AWS IoT Analytics 数据可视化，可使用 AWS IoT Analytics 控制台或 Amazon QuickSight。

主题

- 使用控制台可视化 AWS IoT Analytics 数据
- 使用 Amazon QuickSight 实现 AWS IoT Analytics 数据可视化

使用控制台可视化 AWS IoT Analytics 数据

AWS IoT Analytics 可在 AWS IoT Analytics控制台的容器数据集内容页面上嵌入容器数据集的 HTML 输出（可在文件 `output.html` 中找到）。例如，如果您定义一个运行 Jupyter notebook 的容器数据集，并在 Jupyter notebook 中创建可视化内容，您的数据集可能类似以下内容。



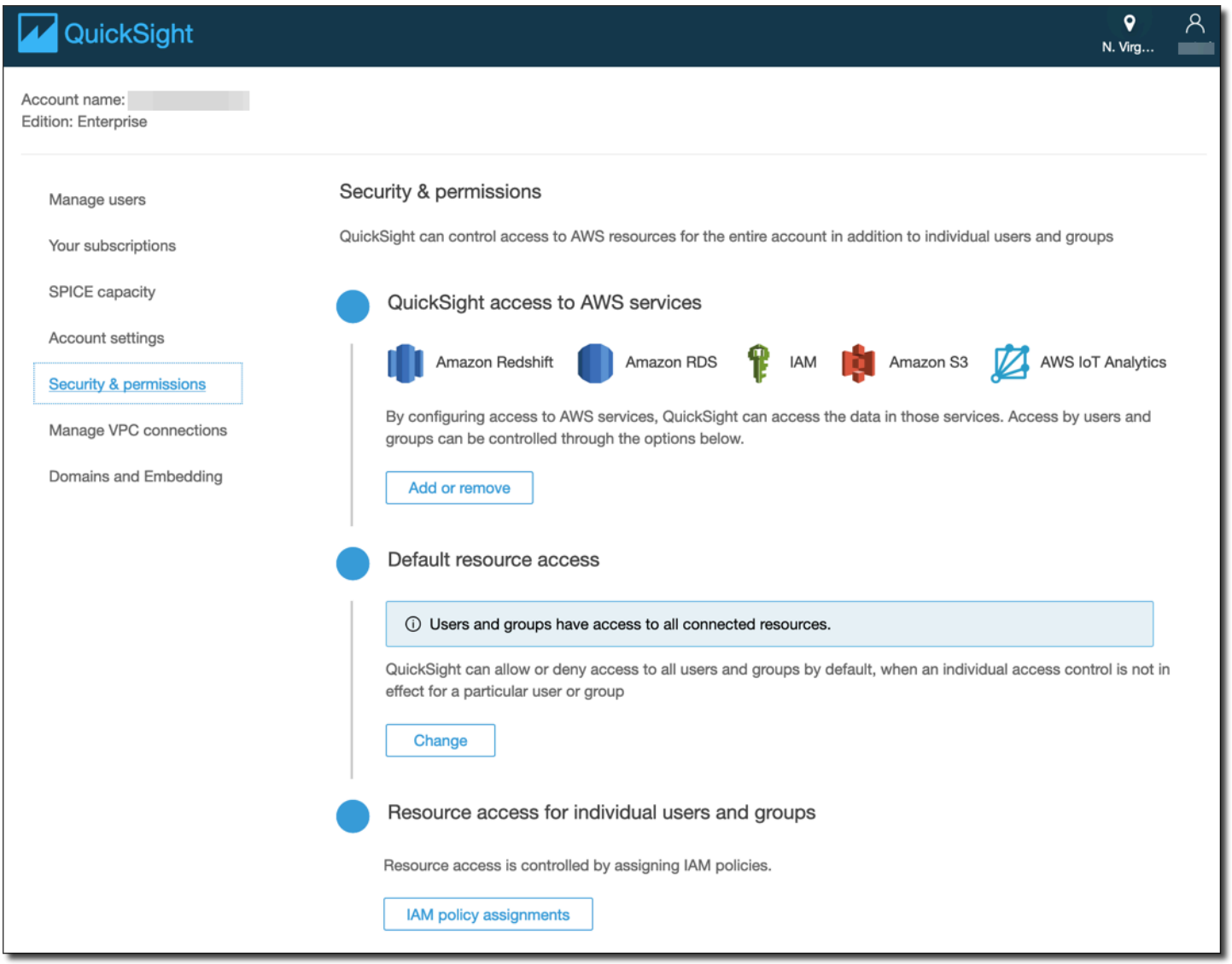
然后，在创建容器数据集内容后，您可以在控制台的数据集内容页面上查看该可视化内容。

有关创建运行 Jupyter notebook 的容器数据集的信息，请参阅自动执行工作流程。

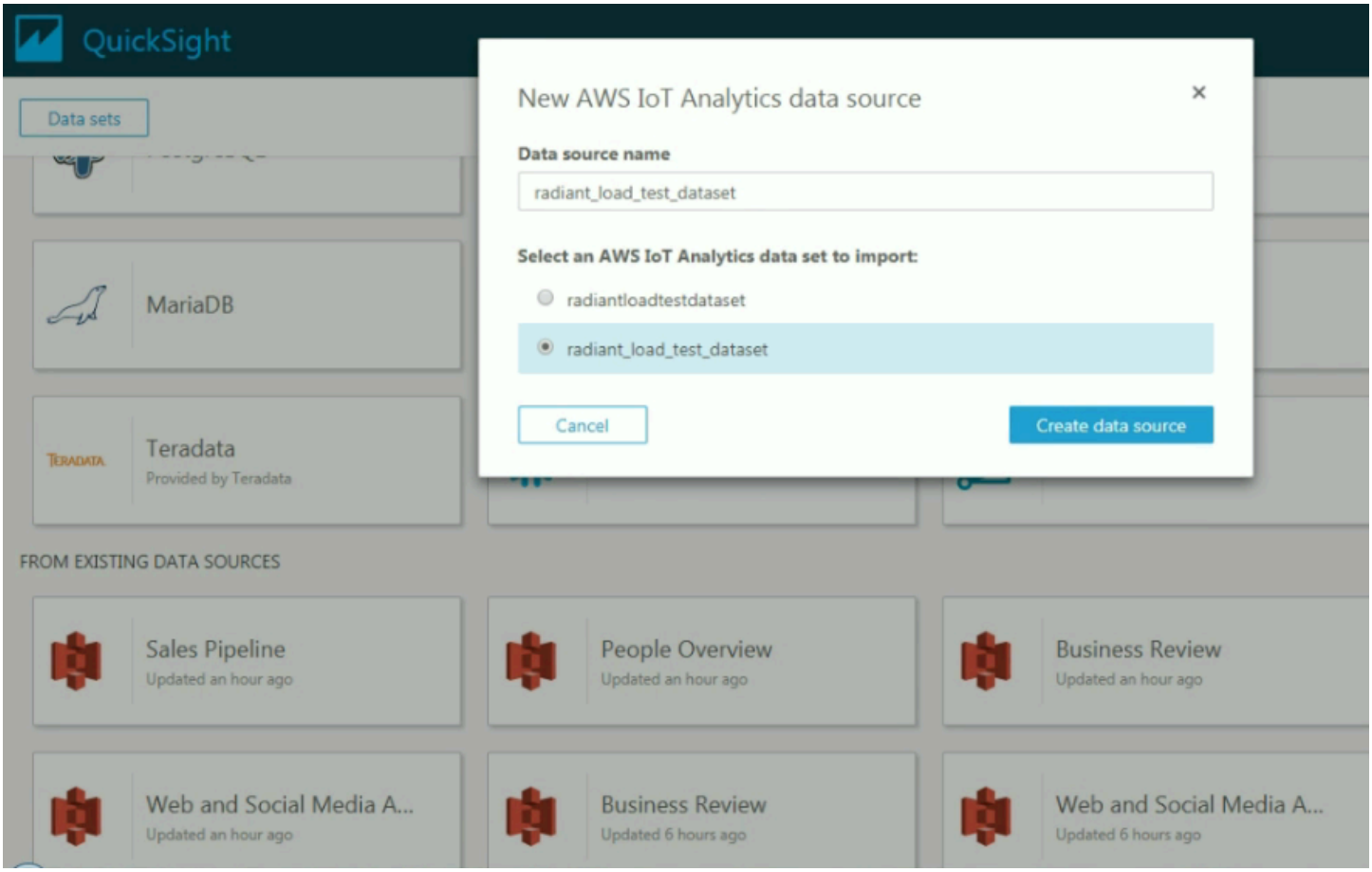
使用 Amazon QuickSight 实现 AWS IoT Analytics 数据可视化

AWS IoT Analytics 提供与 Amazon QuickSight 的直接集成。Amazon QuickSight 是一项快速业务分析服务，可用于构建可视化内容，执行临时分析，并快速地从您的数据中获得业务见解。Amazon QuickSight 使组织可以扩展至数十万用户，并通过使用强大的内存中引擎 (SPICE) 提供响应及时的性能。您可以在 Amazon QuickSight 控制台中选择您的 AWS IoT Analytics 数据集并开始创建控制面板和可视化内容。可以在这些区域中使用 Amazon QuickSight。

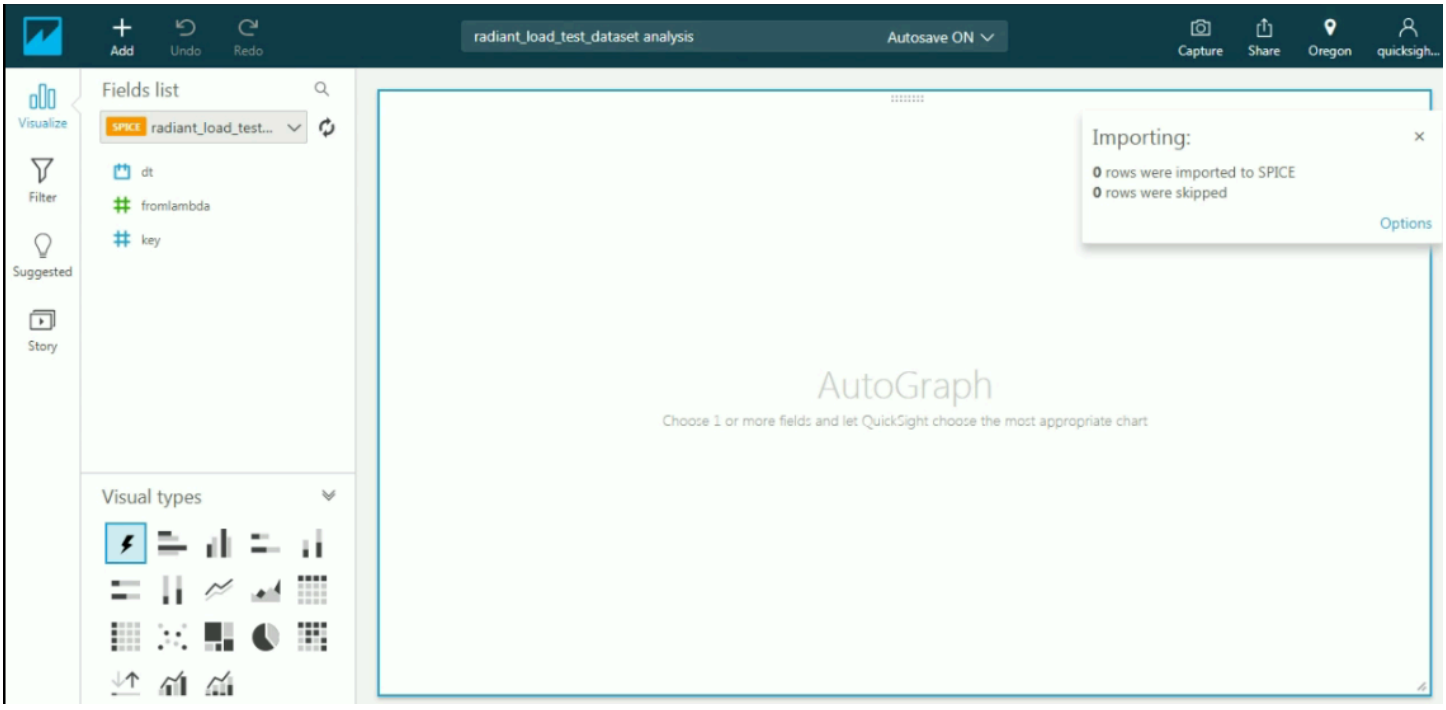
要开始使用 Amazon QuickSight 可视化内容，您必须创建 Amazon QuickSight 账户。在设置账户时，请确保为 Amazon QuickSight 提供了访问您的 AWS IoT Analytics 数据的权限。如果您已有账户，请通过选择管理员、管理 QuickSight、安全和权限，授权 Amazon QuickSight 访问您的 AWS IoT Analytics 数据。在QuickSight 访问 AWS 服务下，选择添加或删除，然后选中 AWS IoT Analytics 旁边的复选框并选择更新。



设置账户后，从管理员 Amazon QuickSight 控制台页面中选择新分析和新数据集，然后选择 AWS IoT Analytics 作为源。键入数据源的名称，选择要导入的数据集，然后选择 创建数据源。



创建数据源后，您便可以在 Amazon QuickSight 中创建可视化内容了。



有关 Amazon QuickSight 控制面板和数据集的信息，请参阅 Amazon QuickSight 文档。

给您的 AWS IoT Analytics 资源加标签

为了方便管理您的通道、数据集、数据存储和管道，您可以选择通过标签的形式为每个资源分配您自己的元数据。本章介绍标签并说明如何创建标签。

主题

有关标签的基本知识

标签可让您按各种标准（例如用途、所有者或环境）对 AWS IoT Analytics 资源进行分类。这在您拥有许多同类型资源时很有用 - 您可以根据分配给资源的标签快速识别特定资源。每个标签都包含您定义的一个键和一个可选值。例如，您可以为通道定义一组标签，以跟踪负责每个通道的消息源的设备类型。我们建议您针对每类资源设计一组标签，以满足您的需要。使用一组连续的标签键，管理资源时会更加轻松。您可以根据添加的标签搜索和筛选资源。

您还可以使用标签对成本进行分类和跟踪。当您将标签应用于通道、数据集、数据存储或管道时，AWS会生成一份逗号分隔值 (CSV) 文件形式的成本分配报告，该报告按标签汇总您的使用率和成本。您可以设置代表业务类别 (例如成本中心、应用程序名称或所有者) 的标签，以便整理多种服务的成本。有关对成本分配使用标签的更多信息，请参阅 AWS Billing 用户指南中的使用成本分配标签。

为便于使用，您可以使用AWS Billing and Cost Management控制台中的标签编辑器，此编辑器提供了一种用于创建和管理标签的集中而统一的方法。有关更多信息，请参阅 AWS Management Console 入门中的使用标签编辑器。

您也可以使用 AWS CLI 和 AWS IoT Analytics API 处理标签。可在创建标签时将其与通道、数据集、数据存储和管道关联；可在以下命令中使用 Tags (标签) 字段：

- CreateChannel
- CreateDataset
- CreateDatastore
- CreatePipeline

您可以为支持标记的现有资源添加、修改或删除标签。使用以下命令：

- TagResource
- ListTagsForResource
- UntagResource

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。如果您添加的标签的值与该实例上现有标签的值相同，新的值就会覆盖旧值。如果删除资源，则所有与资源相关的标签都将被删除。

在 IAM 策略中使用标签

您可以在 IAM 策略中将 Condition 元素（也称为 Condition 块）与以下条件上下文键/值一起使用，以根据资源的标签控制用户访问权限（权限）：

- 使用 `iotanalytics:ResourceTag/<tag-key>: <tag-value>` 可允许或拒绝带特定标签的资源上的用户操作。
- 使用 `aws:RequestTag/<tag-key>: <tag-value>` 可要求在发出创建或修改允许标签的资源的 API 请求时使用（或不使用）特定标签。
- 使用 `aws:TagKeys: [<tag-key>, ...]` 可要求在发出创建或修改允许标签的资源的 API 请求时使用（或不使用）一组特定标签键。

> **Note**
>
> IAM policy 中的条件上下文键/值仅适用于能够标记的资源的标识符是必需参数的那些 AWS IoT Analytics 操作。例如，不会根据条件上下文键/值允许/拒绝使用 DescribeLoggingOptions，因为在该请求中没有引用任何可标记的资源（通道、数据集、数据存储或管道）。

有关更多信息，请参阅《IAM 用户指南》中的使用标签控制访问。该指南的 IAM JSON 策略参考一节包含 IAM 中的 JSON 策略的元素、变量和评估逻辑的详细语法、描述和示例。

以下示例策略应用两个基于标签的限制。受此策略限制的用户：

1. 无法为资源提供标签“env=prod”（请参阅示例中的 `"aws:RequestTag/env" : "prod"` 行）。
2. 无法修改或访问具有现有标签“env=prod”的资源（在示例中，请参阅第 `"iotanalytics:ResourceTag/env" : "prod"` 行）。

```
{
  "Version" : "2012-10-17",
  "Statement" :
  [
    {
      "Effect" : "Deny",
      "Action" : "iotanalytics:*",
      "Resource" : "*",
      "Condition" : {
        "StringEquals" : {
          "aws:RequestTag/env" : "prod"
        }
      }
    },
    {
      "Effect" : "Deny",
      "Action" : "iotanalytics:*",
      "Resource" : "*",
      "Condition" : {
        "StringEquals" : {
          "iotanalytics:ResourceTag/env" : "prod"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "iotanalytics:*"
      ],
      "Resource": "*"
    }
  ]
}
```

您还可以为给定标签键指定多个标签值，方法是将它们括在列表中，如以下示例所示：

```
"StringEquals" : {
  "iotanalytics:ResourceTag/env" : ["dev", "test"]
}
```

> **Note**
>
> 如果您根据标签允许/拒绝用户访问资源，请务必考虑显式拒绝用户在相同资源中添加或删除这些标签的功能。否则，用户可能会修改资源标签以绕过您的限制，并获得该资源的访问权限。

标签限制

下面是适用于标签的基本限制：

- 每个资源的最大标签数 - 50
- 最大密钥长度 - 127 个 Unicode 字符（采用 UTF-8 格式）
- 最大值长度 - 255 个 Unicode 字符（采用 UTF-8 格式）
- 标签键和值区分大小写。
- 请勿在标签名称或值中使用 `aws:` `prefix`，因为它专为 AWS 使用预留。您无法编辑或删除带此前缀的标签名称或值。具有此前缀的标签不计入每个资源的标签数限制。
- 如果您的标记模式针对多个服务和资源使用，请记得其它服务可能对允许使用的字符有限制。通常允许使用的字符包括：可用 UTF-8 格式表示的字母、空格和数字以及以下特殊字符 + - = . _ : / @。

AWS IoT Analytics 中的 SQL 表达式

数据集是使用 SQL 表达式通过数据存储中的数据生成的。AWS IoT Analytics 使用与 Amazon Athena 相同的 SQL 查询、函数和运算符。

AWS IoT Analytics支持 ANSI 标准 SQL 语法的子集。

```
SELECT [ ALL | DISTINCT ] select_expression [, ...]
[ FROM from_item [, ...] ]
[[ INNER | OUTER ] LEFT | RIGHT | FULL | CROSS JOIN join_item [ ON join_condition ]]
[ WHERE condition ]
[ GROUP BY [ ALL | DISTINCT ] grouping_element [, ...] ]
[ HAVING condition ]
[ UNION [ ALL | DISTINCT ] union_query ]
[ ORDER BY expression [ ASC | DESC ] [ NULLS FIRST | NULLS LAST] [, ...] ]
[ LIMIT [ count | ALL ] ]
```

有关参数的描述，请参阅 Amazon Athena 文档中的参数部分。

AWS IoT Analytics 和 Amazon Athena 不支持以下内容：

- WITH子句。
- CREATE TABLE AS SELECT 语句
- INSERT INTO 语句
- 预编译语句，您无法用 USING 运行 EXECUTE。
- CREATE TABLE LIKE
- DESCRIBE INPUT 和 DESCRIBE OUTPUT
- EXPLAIN 语句
- 用户定义的函数 (UDF 或 UDAF)
- 存储过程
- 联合连接器

主题

AWS IoT Analytics 中支持的 SQL 功能

数据集是使用 SQL 表达式从数据存储中的数据生成的。您在 AWS IoT Analytics 中运行的查询基于 Presto 0.217。

支持的数据类型

AWS IoT Analytics 和 Amazon Athena 支持这些数据类型。

- primitive_type
  - TINYINT
  - SMALLINT
  - INT
  - BIGINT
  - BOOLEAN
  - DOUBLE
  - FLOAT
  - STRING
  - TIMESTAMP
  - DECIMAL(precision, scale)
  - DATE
  - CHAR (具有指定长度的长度固定的字符数据)
  - VARCHAR (具有指定长度的长度可变的字符数据)
- array_type
  - ARRAY<data_type>
- map_type
  - MAP<primitive_type, data_type>
- struct_type
  - STRUCT<col_name:data_type[COMMENT col_comment][,...]>

> **Note**
>
> AWS IoT Analytics 和 Amazon Athena 不支持某些数据类型。

支持的函数

Amazon Athena 和 AWS IoT Analytics SQL 功能基于 Presto 0.217。有关相关函数、运算符和表达式的信息，请参阅 函数和运算符以及 Presto 文档中的以下具体章节。

- 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

> **Note**
>
> AWS IoT Analytics 和 Amazon Athena 不支持用户定义的函数（UDF 或 UDAF）或存储的过程。

解决 AWS IoT Analytics 中 SQL 查询的常见问题

使用以下信息可帮助您排查 AWS IoT Analytics 中的 SQL 查询问题。

- 要转义单引号，请在其前面加上另一个单引号。不要将这种情况与双引号混淆。

  Example 示例

  ```
  SELECT 'O''Reilly'
  ```

- 要转义下划线，可以使用反引号将以下划线开头的数据存储列名称括起来。

  Example 示例

  ```
  SELECT `_myMessageAttribute` FROM myDataStore
  ```

- 要对含有数字的名称进行转义，请用双引号将包含数字的数据存储名称括起来。

  Example 示例

  ```
  SELECT * FROM "myDataStore123"
  ```

- 要对保留关键字进行转义，请用双引号将保留关键字括起来。如需更多信息，请参阅 SQL SELECT 语句中保留关键字的列表。

安全性 AWS IoT Analytics

云安全 AWS 是重中之重。作为 AWS 客户，您可以受益于专为满足大多数安全敏感型组织的要求而构建的数据中心和网络架构。

安全是双方共同承担 AWS 的责任。责任共担模式将其描述为云的安全性和云中的安全性：

- 云安全- AWS 负责保护在 AWS 云中运行 AWS 服务的基础架构。 AWS 还为您提供可以安全使用的服务。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。要了解适用的合规计划 AWS IoT Analytics，请参阅按合规计划划分的范围内的AWS 服务。
- 云端安全-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。

本文档将帮助您了解在使用时如何应用分担责任模型 AWS IoT Analytics。以下主题向您介绍如何进行配置 AWS IoT Analytics 以满足您的安全和合规性目标。您还将学习如何使用其他 AWS 服务来帮助您监控和保护您的 AWS IoT Analytics 资源。

AWS Identity and Access Management 在 AWS IoT Analytics

AWS Identity and Access Management (IAM) 是一项 AWS 服务，可帮助管理员安全地控制对 AWS 资源的访问。IAM 管理员控制谁可以进行身份验证（登录）和授权（拥有权限）使用 AWS IoT Analytics 资源。IAM 是一项无需额外付费即可使用的 AWS 服务。

受众

您的使用方式 AWS Identity and Access Management (IAM) 会有所不同，具体取决于您所做的工作 AWS IoT Analytics。

服务用户-如果您使用 AWS IoT Analytics 服务完成工作，则管理员会为您提供所需的凭证和权限。当你使用更多 AWS IoT Analytics 功能来完成工作时，你可能需要额外的权限。了解如何管理访问权限有助于您向管理员请求适合的权限。如果您无法访问 AWS IoT Analytics中的特征，请参阅 对 AWS IoT Analytics 身份和访问进行故障排除。

服务管理员-如果您负责公司的 AWS IoT Analytics 资源，则可能拥有完全访问权限 AWS IoT Analytics。您的工作是确定您的服务用户应访问哪些 AWS IoT Analytics 功能和资源。然后，您必须向 IAM 管理员提交请求以更改服务用户的权限。请查看该页面上的信息以了解 IAM 的基本概念。要详细

了解您的公司如何将 IAM 与配合使用 AWS IoT Analytics，请参阅如何 AWS IoT Analytics 与 IAM 配合使用。

IAM 管理员：如果您是 IAM 管理员，您可能希望了解如何编写策略以管理对 AWS IoT Analytics的访问权限的详细信息。要查看您可以在 IAM 中使用的 AWS IoT Analytics 基于身份的策略示例，请参阅。AWS IoT Analytics 基于身份的策略示例

使用身份进行身份验证

身份验证是您 AWS 使用身份凭证登录的方式。您必须以 IAM 用户身份或通过担 AWS 账户根用户任 IAM 角色进行身份验证（登录 AWS）。

您可以使用通过身份源提供的凭据以 AWS 联合身份登录。 AWS IAM Identity Center （IAM Identity Center）用户、贵公司的单点登录身份验证以及您的 Google 或 Facebook 凭据就是联合身份的示例。当您以联合身份登录时，您的管理员以前使用 IAM 角色设置了身份联合验证。当你使用联合访问 AWS 时，你就是在间接扮演一个角色。

根据您的用户类型，您可以登录 AWS Management Console 或 AWS 访问门户。有关登录的更多信息 AWS，请参阅《AWS 登录 用户指南》中的如何登录到您 AWS 账户的。

如果您 AWS 以编程方式访问，则会 AWS 提供软件开发套件 (SDK) 和命令行接口 (CLI)，以便使用您的凭据对请求进行加密签名。如果您不使用 AWS 工具，则必须自己签署请求。有关使用推荐的方法自行签署请求的更多信息，请参阅 IAM 用户指南中的签署 AWS API 请求。

无论使用何种身份验证方法，您可能需要提供其他安全信息。例如， AWS 建议您使用多重身份验证 (MFA) 来提高账户的安全性。要了解更多信息，请参阅《AWS IAM Identity Center 用户指南》中的多重身份验证和《IAM 用户指南》中的在 AWS中使用多重身份验证（MFA）。

AWS 账户 root 用户

创建时 AWS 账户，首先要有一个登录身份，该身份可以完全访问账户中的所有资源 AWS 服务 和资源。此身份被称为 AWS 账户 root 用户，使用您创建账户时使用的电子邮件地址和密码登录即可访问该身份。强烈建议您不要使用根用户执行日常任务。保护好根用户凭证，并使用这些凭证来执行仅根用户可以执行的任务。有关需要您以根用户身份登录的任务的完整列表，请参阅 IAM 用户指南 中的 需要根用户凭证的任务。

IAM 用户和群组

I AM 用户是您 AWS 账户 内部对个人或应用程序具有特定权限的身份。在可能的情况下，我们建议使用临时凭证，而不是创建具有长期凭证（如密码和访问密钥）的 IAM 用户。但是，如果您有一些特定

的使用场景需要长期凭证以及 IAM 用户，建议您轮换访问密钥。有关更多信息，请参阅《IAM 用户指南》中的对于需要长期凭证的使用场景定期轮换访问密钥。

IAM 组是一个指定一组 IAM 用户的身份。您不能使用组的身份登录。您可以使用组来一次性为多个用户指定权限。如果有大量用户，使用组可以更轻松地管理用户权限。例如，您可能具有一个名为 IAMAdmins 的组，并为该组授予权限以管理 IAM 资源。

用户与角色不同。用户唯一地与某个人员或应用程序关联，而角色旨在让需要它的任何人代入。用户具有永久的长期凭证，而角色提供临时凭证。要了解更多信息，请参阅《IAM 用户指南》中的何时创建 IAM 用户（而不是角色）。

IAM 角色

I AM 角色是您内部具有特定权限 AWS 账户 的身份。它类似于 IAM 用户，但与特定人员不关联。您可以 AWS Management Console 通过切换角色在中临时担任 IAM 角色。您可以通过调用 AWS CLI 或 AWS API 操作或使用自定义 URL 来代入角色。有关使用角色的方法的更多信息，请参阅《IAM 用户指南》中的 使用 IAM 角色。

具有临时凭证的 IAM 角色在以下情况下很有用：

- 联合用户访问 – 要向联合身份分配权限，请创建角色并为角色定义权限。当联合身份进行身份验证时，该身份将与角色相关联并被授予由此角色定义的权限。有关联合身份验证的角色的信息，请参阅《IAM 用户指南》中的 为第三方身份提供商创建角色。如果您使用 IAM Identity Center，则需要配置权限集。为控制您的身份在进行身份验证后可以访问的内容，IAM Identity Center 将权限集与 IAM 中的角色相关联。有关权限集的信息，请参阅《AWS IAM Identity Center 用户指南》中的权限集。
- 临时 IAM 用户权限 – IAM 用户可代入 IAM 用户或角色，以暂时获得针对特定任务的不同权限。
- 跨账户存取 – 您可以使用 IAM 角色以允许不同账户中的某个人（可信主体）访问您的账户中的资源。角色是授予跨账户访问权限的主要方式。但是，对于某些资源 AWS 服务，您可以将策略直接附加到资源（而不是使用角色作为代理）。要了解用于跨账户访问的角色和基于资源的策略之间的差别，请参阅《IAM 用户指南》中的 IAM 角色与基于资源的策略有何不同。
- 跨服务访问 — 有些 AWS 服务 使用其他 AWS 服务服务中的功能。例如，当您在某个服务中进行调用时，该服务通常会在 Amazon EC2 中运行应用程序或在 Amazon S3 中存储对象。服务可能会使用发出调用的主体的权限、使用服务角色或使用服务相关角色来执行此操作。
  - 转发访问会话 (FAS) — 当您使用 IAM 用户或角色在中执行操作时 AWS，您被视为委托人。使用某些服务时，您可能会执行一个操作，然后此操作在其他服务中启动另一个操作。FAS 使用调用委托人的权限以及 AWS 服务 向下游服务发出请求的请求。 AWS 服务只有当服务收到需要与其他 AWS 服务 或资源交互才能完成的请求时，才会发出 FAS 请求。在这种情况下，您必须具有执行这两个操作的权限。有关发出 FAS 请求时的策略详情，请参阅转发访问会话。

- 服务角色 - 服务角色是服务代表您在您的账户中执行操作而分派的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。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。
- 服务相关角色-服务相关角色是一种链接到的服务角色。 AWS 服务服务可以代入代表您执行操作的角色。服务相关角色出现在您的中 AWS 账户 ，并且归服务所有。IAM 管理员可以查看但不能编辑服务相关角色的权限。

- 在 A@@ mazon EC2 上运行的应用程序 — 您可以使用 IAM 角色管理在 EC2 实例上运行并发出 AWS CLI 或 AWS API 请求的应用程序的临时证书。这优先于在 EC2 实例中存储访问密钥。要向 EC2 实例分配 AWS 角色并使其可供其所有应用程序使用，您需要创建附加到该实例的实例配置文件。实例配置文件包含角色，并使 EC2 实例上运行的程序能够获得临时凭证。有关更多信息，请参阅《IAM 用户指南》中的 使用 IAM 角色为 Amazon EC2 实例上运行的应用程序授予权限。

要了解是使用 IAM 角色还是 IAM 用户，请参阅《IAM 用户指南》中的何时创建 IAM 角色（而不是用户）。

使用策略管理访问

您可以 AWS 通过创建策略并将其附加到 AWS 身份或资源来控制中的访问权限。策略是其中的一个对象 AWS ，当与身份或资源关联时，它会定义其权限。 AWS 在委托人（用户、root 用户或角色会话）发出请求时评估这些策略。策略中的权限确定是允许还是拒绝请求。大多数策略都以 JSON 文档的 AWS 形式存储在中。有关 JSON 策略文档的结构和内容的更多信息，请参阅《IAM 用户指南》中的 JSON 策略概览。

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体 可以对什么资源执行操作，以及在什么条件下执行。

默认情况下，用户和角色没有权限。要授予用户对所需资源执行操作的权限，IAM 管理员可以创建 IAM 策略。管理员随后可以向角色添加 IAM 策略，用户可以代入角色。

IAM 策略定义操作的权限，无关乎您使用哪种方法执行操作。例如，假设您有一个允许 `iam:GetRole` 操作的策略。拥有该策略的用户可以从 AWS Management Console AWS CLI、或 AWS API 获取角色信息。

基于身份的策略

基于身份的策略是可附加到身份（如 IAM 用户、用户组或角色）的 JSON 权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的创建 IAM 策略。

基于身份的策略可以进一步归类为内联策略或托管式策略。内联策略直接嵌入单个用户、组或角色中。托管策略是独立的策略，您可以将其附加到中的多个用户、群组和角色 AWS 账户。托管策略包括 AWS 托管策略和客户托管策略。要了解如何在托管式策略和内联策略之间进行选择，请参阅《IAM 用户指南》中的在托管式策略与内联策略之间进行选择。

其它策略类型

AWS 支持其他不太常见的策略类型。这些策略类型可以设置更常用的策略类型向您授予的最大权限。

- 权限边界 - 权限边界是一个高级功能，用于设置基于身份的策略可以为 IAM 实体（IAM 用户或角色）授予的最大权限。您可为实体设置权限边界。这些结果权限是实体基于身份的策略及其权限边界的交集。在 `Principal` 中指定用户或角色的基于资源的策略不受权限边界限制。任一项策略中的显式拒绝将覆盖允许。有关权限边界的更多信息，请参阅《IAM 用户指南》中的 IAM 实体的权限边界。
- 服务控制策略 (SCP)-SCP 是 JSON 策略，用于指定组织或组织单位 (OU) 的最大权限。 AWS Organizations AWS Organizations 是一项用于对您的企业拥有的多 AWS 账户 项进行分组和集中管理的服务。如果在组织内启用了所有功能，则可对任意或全部账户应用服务控制策略 (SCP)。SCP 限制成员账户中的实体（包括每个 AWS 账户根用户实体）的权限。有关 Organizations 和 SCP 的更多信息，请参阅《AWS Organizations 用户指南》中的 SCP 的工作原理。
- 会话策略 – 会话策略是当您以编程方式为角色或联合用户创建临时会话时作为参数传递的高级策略。结果会话的权限是用户或角色的基于身份的策略和会话策略的交集。权限也可以来自基于资源的策略。任一项策略中的显式拒绝将覆盖允许。有关更多信息，请参阅《IAM 用户指南》中的会话策略。

多个策略类型

当多个类型的策略应用于一个请求时，生成的权限更加复杂和难以理解。要了解在涉及多种策略类型时如何 AWS 确定是否允许请求，请参阅 IAM 用户指南中的策略评估逻辑。

如何 AWS IoT Analytics 与 IAM 配合使用

在使用 IAM 管理访问权限之前 AWS IoT Analytics，您应该了解哪些可用的 IAM 功能 AWS IoT Analytics。要全面了解如何 AWS IoT Analytics 和其他 AWS 服务与 IAM 配合使用，请参阅 IAM 用户指南中的与 IAM 配合使用的AWS 服务。

本页面上的主题：

- AWS IoT Analytics 基于资源的政策
- 基于 AWS IoT Analytics 标签的授权
- AWS IoT Analytics IAM 角色

AWS IoT Analytics 基于身份的策略

借助 IAM 基于身份的策略，您可以指定允许或拒绝的操作和资源，以及允许或拒绝操作的条件。 AWS IoT Analytics 支持特定的操作、资源和条件键。要了解在 JSON 策略中使用的所有元素，请参阅《IAM 用户指南》中的 IAM JSON 策略元素参考。

操作

基于 IAM 身份的策略的 `Action` 元素描述该策略将允许或拒绝的特定操作。策略操作通常与关联的 AWS API 操作同名。此操作用于策略中以授予执行关联操作的权限。

策略操作在操作前 AWS IoT Analytics 使用以下前缀：`iotanalytics:`例如，要授予某人使用 AWS IoT Analytics `CreateChannel` API 操作创建 AWS IoT Analytics 频道的权限，您需要将该`iotanalytics:BatchPuMessage`操作包含在他们的策略中。策略声明必须包含`Action`或`NotAction`元素。 AWS IoT Analytics 定义了它自己的一组操作，这些操作描述了您可以使用此服务执行的任务。

要在单个语句中指定多项 操作，请使用逗号将它们隔开，如下所示。

```
"Action": [
    "iotanalytics:action1",
    "iotanalytics:action2"
    ]
```

您也可以使用通配符 (*) 指定多个操作。例如，要指定以单词 `Describe` 开头的所有操作，请包括以下操作。

```
"Action": "iotanalytics:Describe*"
```

要查看 AWS IoT Analytics 操作列表，请参阅 IAM 用户指南 AWS IoT Analytics中定义的操作。

资源

`Resource` 元素指定要向其应用操作的对象。语句必须包含 `Resource` 或 `NotResource` 元素。您可使用 ARN 来指定资源，或使用通配符 (*) 以指明该语句适用于所有资源。

AWS IoT Analytics 数据集资源具有以下 ARN。

```
arn:${Partition}:iotanalytics:${Region}:${Account}:dataset/${DatasetName}
```

有关 ARN 格式的更多信息，请参阅 Amazon Resource Name (ARN) 和 AWS 服务命名空间。

例如，要在语句中指定 Foobar 数据集，请使用以下 ARN。

```
"Resource": "arn:aws:iotanalytics:us-east-1:123456789012:dataset/Foobar"
```

要指定属于特定账户的所有实例，请使用通配符 (*)。

```
"Resource": "arn:aws:iotanalytics:us-east-1:123456789012:dataset/*"
```

某些 AWS IoT Analytics 操作（例如创建资源的操作）无法对特定资源执行。在这些情况下，您必须使用通配符 （*)。

```
"Resource": "*"
```

某些 AWS IoT Analytics API 操作涉及多个资源。例如，CreatePipeline 引用通道和数据集，因此，用户必须具有使用通道和数据集的权限。要在单个语句中指定多个资源，请使用逗号分隔 ARN。

```
"Resource": [
    "resource1",
    "resource2"
     ]
```

要查看 AWS IoT Analytics 资源类型及其 ARN 的列表，请参阅 IAM 用户指南 AWS IoT Analytics中定义的资源。要了解可以在哪些操作中指定每个资源的 ARN，请参阅 AWS IoT Analytics定义的操作。

条件键

在 Condition 元素（或 Condition 块）中，可以指定语句生效的条件。Condition 元素是可选的。您可以构建使用条件运算符（例如，等于或小于）的条件表达式，以使策略中的条件与请求中的值相匹配。

如果您在一个语句中指定多个 Condition 元素，或在单个 Condition 元素中指定多个键，则 AWS 使用逻辑 AND 运算评估它们。如果您为单个条件键指定多个值，则 AWS 使用逻辑 OR 运算来评估条件。在授予语句的权限之前必须满足所有的条件。

在指定条件时，您也可以使用占位符变量。例如，仅当用户使用其用户名进行标记时，您才可为其授予访问资源的权限。有关更多信息，请参阅 IAM 用户指南 中的 IAM policy 元素：变量和标签。

AWS IoT Analytics 不提供任何特定于服务的条件键，但它确实支持使用某些全局条件键。要查看所有 AWS 全局条件键，请参阅 IAM 用户指南中的AWS 全局条件上下文密钥。

示例

要查看 AWS IoT Analytics 基于身份的策略的示例，请参阅。AWS IoT Analytics 基于身份的策略示例

AWS IoT Analytics 基于资源的政策

AWS IoT Analytics 不支持基于资源的策略。要查看详细的基于资源的策略页面的示例，请参阅AWS Lambda 开发人员指南中为 AWS Lambda使用基于资源的策略。

基于 AWS IoT Analytics 标签的授权

您可以为 AWS IoT Analytics 资源附加标签或在请求中传递标签 AWS IoT Analytics。要根据标签控制访问，您需要在策略的条件元素中使用 `iotanalytics:ResourceTag/{key-name}`, `aws:RequestTag/{key-name}` 或 `aws:TagKeys` 条件键提供标签信息。有关为 AWS IoT Analytics 资源添加标签的更多信息，请参阅为资源添加 AWS IoT Analytics 标签。

要查看基于身份的策略示例，该策略用于根据资源上的标签限制对该资源的访问权限，请参阅根据标签查看 AWS IoT Analytics 频道。

AWS IoT Analytics IAM 角色

IAM 角色是 AWS 账户 中具有特定权限的实体。

使用临时凭证和 AWS IoT Analytics

可以使用临时凭证进行联合身份验证登录，分派 IAM 角色或分派跨账户角色。您可以通过调用 AWS Security Token Service (AWS STS) API 操作（例如AssumeRole或 GetFederationToken）来获取临时安全证书。

AWS IoT Analytics 不支持使用临时证书。

服务相关角色

服务线角色允许 AWS 服务访问其他服务中的资源以代表您完成操作。服务相关角色显示在 IAM 账户中，并归该服务所有。IAM 管理员可以查看但不能编辑服务相关角色的权限。

AWS IoT Analytics 不支持服务相关角色。

服务角色

此功能允许服务代表您担任服务角色。此角色允许服务访问其他服务中的资源以代表您完成操作。服务角色显示在 IAM 账户中，并归该账户所有。这意味着，IAM 管理员可以更改该角色的权限。但是，这样做可能会中断服务的功能。

AWS IoT Analytics 支持服务角色。

跨服务混淆代理问题防范

混淆代理问题是一个安全性问题，即不具有操作执行权限的实体可能会迫使具有更高权限的实体执行该操作。在 AWS 中，跨服务模拟可能会导致混淆代理问题。一个服务（呼叫服务）调用另一项服务（所谓的服务)时，可能会发生跨服务模拟。可以操纵调用服务以使用其权限对另一个客户的资源进行操作，否则该服务不应有访问权限。为了防止这种情况，AWS 提供可帮助您保护所有服务的服务委托人数据的工具，这些服务委托人有权限访问账户中的资源。

我们建议在资源策略中使用 `aws:SourceArn` 和 `aws:SourceAccount` 全局条件上下文密钥。这样可以限制 AWS IoT Analytics 为该资源提供其他服务的权限。如果使用两个全局条件上下文键，在同一策略语句中使用时，`aws:SourceAccount` 值和 `aws:SourceArn` 值中的账户必须使用相同的账户 ID。

防止混淆代理问题最有效的方法是使用具有资源完整 Amazon Resource Name (ARN) 的 `aws:SourceArn` 全局条件上下文键。如果您不知道资源的完整 ARN，或正在指定多个资源，请针对 ARN 未知部分使用带有通配符 (*) 的 `aws:SourceArn` 全局上下文条件键。例如，`arn:aws:iotanalytics::123456789012:*`。

主题

- Amazon S3 存储桶的防护
- 使用 Amazon CloudWatch Logs 防范
- 对客户管理的 AWS IoT Analytics 资源进行混淆代理预防

Amazon S3 存储桶的防护

如果使用客户托管的 Amazon S3 存储进行 AWS IoT Analytics 数据存储，存储您数据的 Amazon S3 存储桶可能会遇到混淆代理问题。

例如，Nikki Wolf 使用了客户拥有的名为 *DOC-EXAMPLE-BUCKET* 的 Amazon S3 存储桶。该存储桶存储在 *us-east-1* 区域中创建的 AWS IoT Analytics 数据存储的信息。她指定了一项策略，以便 AWS IoT Analytics 服务主体代表其查询 *DOC-EXAMPLE-BUCKET*。Nikki 的同事 Li Juan 通过自己的账户查

询 *DOC-EXAMPLE-BUCKET*，并使用查询结果创建数据集。结果，尽管该查询是通过她自己的账户进行的，但 AWS IoT Analytics 服务主体还是代表其查询了 Nikki 的 Amazon S3 存储桶。

为避免这种情况，Nikki 可在策略中对 *DOC-EXAMPLE-BUCKET* 指定 `aws:SourceAccount` 条件或 `aws:SourceArn` 条件。

指定 **`aws:SourceAccount`** 条件 - 以下存储桶策略示例指定了仅 Nikki 账户(*123456789012*)中的 AWS IoT Analytics 资源可访问 *DOC-EXAMPLE-BUCKET*。

```
{
    "Version": "2012-10-17",
    "Id": "MyPolicyID",
    "Statement": [
        {
            "Sid": "ConfusedDeputyPreventionExamplePolicy",
            "Effect": "Allow",
            "Principal": {
                "Service": "iotanalytics.amazonaws.com"
            },
            "Action": [
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:ListMultipartUploadParts",
                "s3:AbortMultipartUpload",
                "s3:PutObject",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET",
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "123456789012"
                }
            }
        }
    ]
}
```

指定 **`aws:SourceArn`** 条件 - 或者，Nikki 也可使用 `aws:SourceArn` 条件。

```
{
    "Version": "2012-10-17",
    "Id": "MyPolicyID",
    "Statement": [
        {
            "Sid": "ConfusedDeputyPreventionExamplePolicy",
            "Effect": "Allow",
            "Principal": {
                "Service": "iotanalytics.amazonaws.com"
            },
            "Action": [
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:ListMultipartUploadParts",
                "s3:AbortMultipartUpload",
                "s3:PutObject",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET",
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
            ],
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": [
                        "arn:aws:iotanalytics:us-east-1:123456789012:dataset/DOC-
EXAMPLE-DATASET",
                        "arn:aws:iotanalytics:us-east-1:123456789012:datastore/DOC-
EXAMPLE-DATASTORE"
                    ]
                }
            }
        }
    ]
}
```

使用 Amazon CloudWatch Logs 防范

在使用 Amazon CloudWatch Logs 进行监控时，可防止出现混淆代理问题。以下资源策略显示了如何防止混淆代理问题：

- 全局条件上下文密钥，`aws:SourceArn`
- 具有您 AWS 账户 ID 的 `aws:SourceAccount`
- 与 AWS IoT Analytics 中 `sts:AssumeRole` 请求关联的客户资源

如下例所示，将 *123456789012* 替换为 AWS 账户 ID，将 *us-east-1* 替换为 AWS IoT Analytics 账户所在区域。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "iotanalytics.amazonaws.com"
            },
            "Action": "logs:PutLogEvents",
            "Resource": "*",
            "Condition":{
                "ArnLike":{
                    "aws:SourceArn":"arn:aws:iotanalytics:us-east-1:123456789012:*/*"
                },
                "StringEquals":{
                    "aws:SourceAccount":"123456789012"
                }
            }
    ]
}
```

有关启用和配置 Amazon CloudWatch Logs 的更多信息，请参阅 the section called “日志记录和监控”。

对客户管理的 AWS IoT Analytics 资源进行混淆代理预防

如果授权 AWS IoT Analytics 对 AWS IoT Analytics 资源执行操作，这些资源可能会遇到混淆代理问题。为防止出现混淆代理问题，可使用以下示例资源策略来限制授予 AWS IoT Analytics 的权限。

主题

- 防范 AWS IoT Analytics 通道和数据存储
- AWS IoT Analytics 数据集内容传送规则的跨服务混淆代理预防

防范 AWS IoT Analytics 通道和数据存储

您可使用 IAM 角色来控制 AWS IoT Analytics 代表您访问的 AWS 资源。为防止您的角色面临混淆代理问题，可在 `aws:SourceAccount` 元素中指定 AWS 账户，并在附加到角色的信任策略的 `aws:SourceArn` 元素中指定 AWS IoT Analytics 资源 ARN。

在以下示例中，将 *123456789012* 替换为 AWS 账户 ID，将 *arn:aws:iotanalytics:aws-region:123456789012:channel/DOC-EXAMPLE-CHANNEL* 替换为 AWS IoT Analytics 通道或数据存储的 ARN。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ConfusedDeputyPreventionExamplePolicy",
      "Effect": "Allow",
      "Principal": {
        "Service": "iotanalytics.amazonaws.com"
       },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnLike": {
          "aws:SourceArn": "arn:aws:iotanalytics:aws-region:123456789012:channel/DOC-
EXAMPLE-CHANNEL"
        }
      }
    }
  ]
}
```

要了解有关通道和数据存储的客户托管 S3 存储选项的更多信息，请参阅《AWS IoT Analytics API 参考》中的 CustomerManagedChannelS3Storage 和 CustomerManagedDatastoreS3Storage。

AWS IoT Analytics 数据集内容传送规则的跨服务混淆代理预防

在将数据集查询结果传送到 Amazon S3 或 AWS IoT Events 时 AWS IoT Analytics 所承担的 IAM 角色可能会遇到混淆代理问题。为防止出现混淆代理问题，请在 `aws:SourceAccount` 元素中指定 AWS 账户，并在附加到角色的信任策略 `aws:SourceArn` 元素中指定 AWS IoT Analytics 资源的 ARN。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ConfusedDeputyPreventionExampleTrustPolicyDocument",
      "Effect": "Allow",
      "Principal": {
        "Service": "iotanalytics.amazonaws.com"
       },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnLike": {
          "aws:SourceArn": "arn:aws:iotanalytics:aws-region:123456789012:dataset/DOC-
EXAMPLE-DATASET"
        }
      }
    }
  ]
}
```

有关配置数据集内容传送规则的更多详细信息，请参阅《AWS IoT Analytics API 参考》中的 contentDeliveryRules。

AWS IoT Analytics 基于身份的策略示例

默认情况下，用户和角色没有创建或修改 AWS IoT Analytics 资源的权限。他们也无法使用 AWS Management Console AWS CLI、或 AWS API 执行任务。IAM 管理员必须创建 IAM 策略，以便为用户和角色授予权限以对所需的指定资源执行特定的 API 操作。然后，管理员必须将这些策略附加到需要这些权限的用户或组。

要了解如何使用这些示例 JSON 策略文档创建 IAM 基于身份的策略，请参阅 IAM 用户指南中的在 JSON 选项卡上创建策略

本页面上的主题：

- 策略最佳实践
- 使用控制 AWS IoT Analytics 台
- 允许用户查看他们自己的权限

- 访问一个 AWS IoT Analytics 输入
- 根据标签查看 AWS IoT Analytics 频道

策略最佳实践

基于身份的策略非常强大。它们决定是否有人可以在您的账户中创建、访问或删除 AWS IoT Analytics 资源。这些操作可能会使 AWS 账户产生成本。创建或编辑基于身份的策略时，请遵循以下准则和建议：

- 开始使用 AWS 托管策略-要 AWS IoT Analytics 快速开始使用，请使用 AWS 托管策略为员工提供所需的权限。这些策略已在您的账户中提供，并由 AWS维护和更新。有关更多信息，请参阅 IAM 用户指南中的使用 AWS 托管策略的权限入门。
- 授予最低权限 – 创建自定义策略时，仅授予执行任务所需的许可。最开始只授予最低权限，然后根据需要授予其它权限。这样做比起一开始就授予过于宽松的权限而后再尝试收紧权限来说更为安全。有关更多信息，请参阅 IAM 用户指南 中的授予最低权限。
- 为敏感操作启用 MFA – 为增强安全性，要求用户使用多重身份验证 (MFA) 来访问敏感资源或 API 操作。有关更多信息，请参阅《 IAM 用户指南》中的在 AWS中使用多重身份验证（MFA）。
- 使用策略条件来增强安全性 – 在切实可行的范围内，定义基于身份的策略在哪些情况下允许访问资源。例如，您可编写条件来指定请求必须来自允许的 IP 地址范围。您也可以编写条件，以便仅允许指定日期或时间范围内的请求，或者要求使用 SSL 或 MFA。有关更多信息，请参阅 IAM 用户指南中的 IAM JSON 策略元素：条件。

使用控制 AWS IoT Analytics 台

要访问 AWS IoT Analytics 控制台，您必须拥有一组最低权限。这些权限必须允许您列出和查看有关您的 AWS IoT Analytics 资源的详细信息 AWS 账户。如果创建比必需的最低权限更为严格的基于身份的策略，对于附加了该策略的实体（用户或角色），控制台将无法按预期正常运行。

为确保这些实体仍然可以使用 AWS IoT Analytics 控制台，还要将以下 AWS 托管策略附加到这些实体。有关更多信息，请参阅《IAM 用户指南》中的为用户添加权限。

```
{
     "Version": "2012-10-17",
     "Statement": [
         {
             "Effect": "Allow",
             "Action": [
                 "iotanalytics:BatchPutMessage",
```

```
                "iotanalytics:CancelPipelineReprocessing",
                "iotanalytics:CreateChannel",
                "iotanalytics:CreateDataset",
                "iotanalytics:CreateDatasetContent",
                "iotanalytics:CreateDatastore",
                "iotanalytics:CreatePipeline",
                "iotanalytics:DeleteChannel",
                "iotanalytics:DeleteDataset",
                "iotanalytics:DeleteDatasetContent",
                "iotanalytics:DeleteDatastore",
                "iotanalytics:DeletePipeline",
                "iotanalytics:DescribeChannel",
                "iotanalytics:DescribeDataset",
                "iotanalytics:DescribeDatastore",
                "iotanalytics:DescribeLoggingOptions",
                "iotanalytics:DescribePipeline",
                "iotanalytics:GetDatasetContent",
                "iotanalytics:ListChannels",
                "iotanalytics:ListDatasetContents",
                "iotanalytics:ListDatasets",
                "iotanalytics:ListDatastores",
                "iotanalytics:ListPipelines",
                "iotanalytics:ListTagsForResource",
                "iotanalytics:PutLoggingOptions",
                "iotanalytics:RunPipelineActivity",
                "iotanalytics:SampleChannelData",
                "iotanalytics:StartPipelineReprocessing",
                "iotanalytics:TagResource",
                "iotanalytics:UntagResource",
                "iotanalytics:UpdateChannel",
                "iotanalytics:UpdateDataset",
                "iotanalytics:UpdateDatastore",
                "iotanalytics:UpdatePipeline"
            ],
            "Resource": "arn:${Partition}:iotanalytics:${Region}:${Account}:channel/
${channelName}",
            "Resource": "arn:${Partition}:iotanalytics:${Region}:${Account}:dataset/
${datasetName}",
            "Resource": "arn:${Partition}:iotanalytics:${Region}:${Account}:datastore/
${datastoreName}",
            "Resource": "arn:${Partition}:iotanalytics:${Region}:${Account}:pipeline/
${pipelineName}"
        }
    ]
```

```
}
```

对于仅调用 AWS CLI 或 AWS API 的用户，您无需为其设置最低控制台权限。相反，只允许访问与您尝试执行的 API 操作相匹配的操作。

允许用户查看他们自己的权限

此示例显示您可以如何创建策略，以便允许 用户查看附加到其用户身份的内联和托管策略。此策略包括在控制台上或使用 AWS CLI 或 AWS API 以编程方式完成此操作的权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": [
                "arn:aws:iam::*:user/${aws:username}"
            ]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
```

```
}
```

访问一个 AWS IoT Analytics 输入

在此示例中，您想授予用户 AWS 账户 访问您的其中一个 AWS IoT Analytics 频道的权限exampleChannel。您还想要允许该用户添加、更新和删除通道。

此策略向用户授予 iotanalytics:ListChannels, iotanalytics:DescribeChannel, iotanalytics:CreateChannel, iotanalytics:DeleteChannel, and iotanalytics:UpdateChannel 权限。有关为用户授予权限并使用控制台测试这些权限的 Amazon S3 服务的演练示例，请参阅演练示例：使用用户策略控制对存储桶的访问。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"ListChannelsInConsole",
         "Effect":"Allow",
         "Action":[
            "iotanalytics:ListChannels"
         ],
         "Resource":"arn:aws:iotanalytics:::*"
      },
      {
         "Sid":"ViewSpecificChannelInfo",
         "Effect":"Allow",
         "Action":[
            "iotanalytics:DescribeChannel"
         ],
         "Resource":"arn:aws:iotanalytics:::exampleChannel"
      },
      {
         "Sid":"ManageChannels",
         "Effect":"Allow",
         "Action":[
            "iotanalytics:CreateChannel",
            "iotanalytics:DeleteChannel",
            "iotanalytics:DescribeChannel",
            "iotanalytics:ListChannels",
            "iotanalytics:UpdateChannel"
         ],
         "Resource":"arn:aws:iotanalytics:::exampleChannel/*"
```

```
      }
   ]
}
```

根据标签查看 AWS IoT Analytics 频道

您可以使用基于身份的策略中的条件根据标签控制对 AWS IoT Analytics 资源的访问权限。此示例显示如何创建策略以允许查看 channel。但是，仅当 channel 标签 Owner 具有该用户的用户名的值时，才授予此权限。该策略还会授予在控制台上完成该操作所需的权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ListChannelsInConsole",
            "Effect": "Allow",
            "Action": "iotanalytics:ListChannels",
            "Resource": "*"
        },
        {
            "Sid": "ViewChannelsIfOwner",
            "Effect": "Allow",
            "Action": "iotanalytics:ListChannels",
            "Resource": "arn:aws:iotanalytics:*:*:channel/*",
            "Condition": {
                "StringEquals": {"iotanalytics:ResourceTag/Owner": "${aws:username}"}
            }
        }
    ]
}
```

您可以将此策略附加到您账户中的用户。如果名为的用户`richard-roe`尝试查看 AWS IoT Analytics `channel`，则`channel`必须对其进行标记`Owner=richard-roe or owner=richard-roe`。否则，他将被拒绝访问。条件标签键 Owner 匹配 Owner 和 owner，因为条件键名称不区分大小写。有关更多信息，请参阅《IAM 用户指南》中的 IAM JSON 策略元素：条件。

对 AWS IoT Analytics 身份和访问进行故障排除

使用以下信息来帮助您诊断和修复在处理时可能遇到的常见问题 AWS IoT Analytics。

主题

- 我无权在以下位置执行操作 AWS IoT Analytics
- 我无权执行 iam:PassRole
- 我想允许我以外的人 AWS 账户 访问我的 AWS IoT Analytics 资源

我无权在以下位置执行操作 AWS IoT Analytics

如果 AWS Management Console 告诉您您无权执行某项操作，则必须联系管理员寻求帮助。您的管理员是指为您提供用户名和密码的那个人。

当 `mateojackson` 用户尝试使用控制台查看有关 `channel` 的详细信息，但不具有 `iotanalytics:ListChannels` 权限时，会发生以下示例错误。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 iotanalytics:``ListChannels`` on resource: ``my-example-channel``
```

在这种情况下，Mateo 请求他的管理员更新其策略，以允许他使用 `iotanalytics:ListChannel` 操作访问 `my-example-channel` 资源。

我无权执行 **`iam:PassRole`**

如果您收到一个错误，表明您无权执行 `iam:PassRole` 操作，则必须更新策略以允许您将角色传递给 AWS IoT Analytics。

有些 AWS 服务 允许您将现有角色传递给该服务，而不是创建新的服务角色或服务相关角色。为此，您必须具有将角色传递到服务的权限。

当名为 `marymajor` 的 IAM 用户尝试使用控制台在 AWS IoT Analytics中执行操作时，会发生以下示例错误。但是，服务必须具有服务角色所授予的权限才可执行此操作。Mary 不具有将角色传递到服务的权限。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

在这种情况下，必须更新 Mary 的策略以允许她执行 `iam:PassRole` 操作。

如果您需要帮助，请联系您的 AWS 管理员。您的管理员是提供登录凭证的人。

我想允许我以外的人 AWS 账户 访问我的 AWS IoT Analytics 资源

您可以创建一个角色，以便其他账户中的用户或您组织外的人员可以使用该角色来访问您的资源。您可以指定谁值得信赖，可以担任角色。对于支持基于资源的策略或访问控制列表 (ACL) 的服务，您可以使用这些策略向人员授予对您的资源的访问权。

要了解更多信息，请参阅以下内容：

- 要了解是否 AWS IoT Analytics 支持这些功能，请参阅如何 AWS IoT Analytics 使用 IAM。
- 要了解如何提供对您拥有的资源的访问权限 AWS 账户 ，请参阅 IAM 用户指南中的向您拥有 AWS 账户 的另一个 IAM 用户提供访问权限。
- 要了解如何向第三方提供对您的资源的访问权限 AWS 账户，请参阅 IAM 用户指南中的向第三方提供访问权限。 AWS 账户
- 要了解如何通过身份联合验证提供访问权限，请参阅《IAM 用户指南》中的为经过外部身份验证的用户（身份联合验证）提供访问权限。
- 要了解使用角色和基于资源的策略进行跨账户存取之间的差别，请参阅《IAM 用户指南》中的 IAM 角色与基于资源的策略有何不同。

AWS IoT Analytics 中的日志记录和监控

AWS 为您提供了各种可用于监控 AWS IoT Analytics 的工具。您可以配置其中的一些工具以便进行监控。一些工具需要手动干预。建议您尽可能实现监控任务自动化。

自动监控工具

您可以使用以下自动化监控工具来监控 AWS IoT 并在出现错误时报告：

- Amazon CloudWatch Logs – 监控、存储和访问来自 AWS CloudTrail 或其它来源的日志文件。有关更多信息，请参阅 Amazon CloudWatch 用户指南中的什么是AWS CloudTrail监控日志文件。
- AWS CloudTrail 日志监控 – 在账户间共享日志文件，通过将 CloudTrail 日志文件发送到 CloudWatch Logs 来进行实时监控，用 Java 编写日志处理应用程序，验证 CloudTrail 提供的日志文件是否未发生更改。有关更多信息，请参阅《AWS CloudTrail 用户指南》中的使用 CloudTrail 日志文件。

手动监控工具

监控 AWS IoT 的另一个重要环节是手动监控 CloudWatch 告警未涵盖的那些项目。AWS IoT、CloudWatch 和其它 AWS 服务控制台控制面板提供您的 AWS 环境状态的概览视图。建议您还要查看 AWS IoT Analytics 上的日志文件。

- 该 AWS IoT Analytics 控制台显示：
  - 通道
  - 管道
  - 数据存储
  - 数据集
  - 笔记本
  - 设置
  - 学习
- CloudWatch 主页显示：
  - 当前警报和状态
  - 告警和资源图表
  - 服务运行状况

  此外，还可以使用 CloudWatch 执行以下操作：
  - 创建自定义控制面板以监控您关心的服务
  - 绘制指标数据图，以排除问题并弄清楚趋势
  - 搜索并浏览您所有的 AWS 资源指标
  - 创建和编辑警报以接收有关问题的通知

使用 Amazon CloudWatch Logs 监控

AWS IoT Analytics使用 Amazon CloudWatch 支持进行日志记录 您可以使用 `PutLoggingOptions`API 操作为 AWS IoT Analytics 启用和配置 Amazon CloudWatch 日志记录。本节介绍如何使用 `PutLoggingOptions` 与 AWS Identity and Access Management (IAM)，为 AWS IoT Analytics 配置和启用 Amazon CloudWatch 日志记录。

有关 CloudWatch Logs 的更多信息，请参阅 Amazon CloudWatch Logs 用户指南。有关 AWS IAM 的更多信息，请参阅AWS Identity and Access Management 用户指南。

> **Note**
>
> 在启用 AWS IoT Analytics 日志记录之前，请务必了解 CloudWatch Logs 访问权限。拥有 CloudWatch Logs 访问权限的用户能够查看您的调试信息。有关更多信息，请参阅 Amazon CloudWatch Logs 的身份验证和访问控制。

创建 IAM 角色以启用日志记录

创建 IAM 角色以启用 Amazon CloudWatch 日志记录

1. 使用 AWSIAM 控制台或以下 AWS IAM CLI 命令 createRole 创建具有信任关系策略（信任策略）的新 IAM 角色。该信任策略向 Amazon CloudWatch 等实体授予角色担任权限。

```
aws iam create-role --role-name exampleRoleName --assume-role-policy-document
       exampleTrustPolicy.json
```

`exampleTrustPolicy.json` 文件包含以下内容。

> **Note**
>
> 此示例包括一个全局条件上下文密钥，用于防止混淆代理安全问题。将 *123456789012* 替换为您的 AWS 账户 ID，将 *aws-region* 替换为您的 AWS 资源 AWS 区域。有关更多信息，请参阅the section called “跨服务混淆代理问题防范”。

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Principal": {
            "Service": "iotanalytics.amazonaws.com"
         },
         "Action": "sts:AssumeRole",
         "Condition": {
           "StringEquals": {
             "aws:SourceAccount": "123456789012"
           },
```

```
            "ArnLike": {
              "aws:SourceArn": "arn:aws:iotanalytics:aws-region:123456789012:*"
            }
          }

      }
   ]
}
```

以后，在调用 AWS IoT Analytics PutLoggingOptions 命令时，您可以使用该角色的 ARN。

2. 使用 AWS IAM PutRolePolicy 将权限策略(a role policy)附加到步骤 1 中创建的角色。

   ```
   aws iam put-role-policy --role-name exampleRoleName --policy-name
   examplePolicyName --policy-document exampleRolePolicy.json
   ```

   示例 RolePolicy.json 文件包含以下内容。

   ```
   {
      "Version": "2012-10-17",
      "Statement": [
      {
      "Effect": "Allow",
      "Action": [
       "logs:CreateLogGroup",
       "logs:CreateLogStream"
      ],
      "Resource": [
       "arn:aws:logs:*:*:*"
      ]
      }
      ]
   }
   ```

3. 要授权 AWS IoT Analytics 向 Amazon CloudWatch 发布日志事件，请使用 Amazon CloudWatch 命令 PutResourcePolicy。

   > **Note**
   >
   > 为防范混淆代理安全问题，我们建议在资源策略中指定 aws:SourceArn。这限制了访问权限，仅接受来自指定账户的请求。有关混淆代理问题的更多信息，请参阅the section called “跨服务混淆代理问题防范”。

```
aws logs put-resource-policy --policy-in-json
exampleResourcePolicy.json
```

exampleResourcePolicy.json 文件包含以下资源策略：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "iotanalytics.amazonaws.com"
            },
            "Action": "logs:PutLogEvents",
            "Resource": "*",
            "Condition":{
                "ArnLike":{
                    "aws:SourceArn":"arn:aws:iotanalytics:us-east-1:123456789012:*/
*"
                },
                "StringEquals":{
                    "aws:SourceAccount":"123456789012"
                }
            }
    ]
}
```

配置并启用日志记录

使用 PutLoggingOptions 命令为 AWS IoT Analytics 配置并启用 Amazon CloudWatch 日志记录。loggingOptions 字段中的 roleArn 应为您在上一节中创建的角色的 ARN。还可以使用 DecribeLoggingOptions 命令来检查您的日志记录选项设置。

PutLoggingOptions

设置或更新 AWS IoT Analytics 日志选项。如果您更新任何 loggingOptions 字段的值，则最多需要一分钟，更改才能生效。此外，如果您更改附加到您在 roleArn 字段中所指定角色的策略（例如，更正无效策略），则最多需要五分钟，更改才能生效。有关更多信息，请参阅PutLoggingOptions。

DescribeLoggingOptions

检索 AWS IoT Analytics 日志记录选项的当前设置。有关更多信息，请参阅DescribeLoggingOptions。

命名空间、指标和维度

AWS IoT Analytics 会将以下指标放入 Amazon CloudWatch 存储库：

命名空间
AWS/IoTAnalytics

指标	描述
ActionExecution	已执行的操作的数量。
ActionExecutionThrottled	受限制的操作数。
ActivityExecutionError	执行管道活动时生成的错误数。
IncomingMessages	进入通道的消息数量。
PipelineConcurrentExecutionCount	并发执行的管道活动数量。

维度	描述
ActionType	正在监视的操作类型。
ChannelName	正在监视的通道的名称。
DatasetName	正在监视的数据集的名称。
DatastoreName	正在监视的数据存储的名称。
PipelineActivityName	正在监视的管道活动的名称。
PipelineActivityType	正在监视的管道活动的类型。

维度	描述
PipelineName	正在监视的管道的名称。

使用 Amazon CloudWatch Events 监控

当 AWS Lambda 活动期间发生运行时系统错误时，AWS IoT Analytics 会自动将事件发布到 Amazon CloudWatch Events。此事件包含详细的错误消息以及存储未处理通道消息的 Amazon Simple Storage Service (Amazon S3)对象的密钥。您可使用 Amazon S3 密钥重新处理未处理的通道消息。有关更多信息，请参阅 重新处理通道消息、《AWS IoT Analytics API 参考》中的 StartPipelineReprocessing 和《Amazon CloudWatch Events 用户指南》中的“什么是 Amazon CloudWatch Events”。

您还可以配置目标，使 Amazon CloudWatch Events 能够发送通知或执行后续操作。例如，您可以发送通知到 Amazon Simple Queue Service (Amazon SQS) 队列中，然后调用 `StartReprocessingMessage` API 来处理保存在 Amazon S3 对象中的通道消息。Amazon CloudWatch Events 支持多种类型的目标，例如：

- Amazon Kinesis Streams
- AWS Lambda 函数
- Amazon Simple Notification Service (Amazon SNS) 主题
- Amazon Simple Queue Service (Amazon SQS) 队列

有关支持的目标列表，请参阅 Amazon EventBridge 用户指南中的 Amazon EventBridge 目标。

CloudWatch Events 资源及关联目标必须位于所创建 AWS IoT Analytics 资源的 AWS 区域。有关更多信息，请参阅 AWS 一般参考 中的服务端点和配额。

针对 AWS Lambda 活动中运行时系统错误发送给 Amazon CloudWatch Events 的通知使用以下格式。

```
{
    "version": "version-id",
    "id": "event-id",
    "detail-type": "IoT Analytics Pipeline Failure Notification",
    "source": "aws.iotanalytics",
    "account": "aws-account",
    "time": "timestamp",
    "region": "aws-region",
    "resources": [
```

```
        "pipeline-arn"
    ],
    "detail": {
        "event-detail-version": "1.0",
        "pipeline-name": "pipeline-name",
        "error-code": "LAMBDA_FAILURE",
        "message": "error-message",
        "channel-messages": {
            "s3paths": [
                "s3-keys"
            ]
        },
        "activity-name": "lambda-activity-name",
        "lambda-function-arn": "lambda-function-arn"
    }
}
```

示例通知：

```
{
    "version": "0",
    "id": "204e672e-ef12-09af-4cfd-de3b53673ec6",
    "detail-type": "IoT Analytics Pipeline Failure Notification",
    "source": "aws.iotanalytics",
    "account": "123456789012",
    "time": "2020-10-15T23:47:02Z",
    "region": "ap-southeast-2",
    "resources": [
        "arn:aws:iotanalytics:ap-southeast-2:123456789012:pipeline/
test_pipeline_failure"
    ],
    "detail": {
        "event-detail-version": "1.0",
        "pipeline-name": "test_pipeline_failure",
        "error-code": "LAMBDA_FAILURE",
        "message": "Temp unavaliable",
        "channel-messages": {
        "s3paths": [
            "test_pipeline_failure/channel/cmr_channel/__dt=2020-10-15
 00:00:00/1602805530000_1602805560000_123456789012_cmr_channel_0_257.0.json.gz"
        ]
    },
    "activity-name": "LambdaActivity_33",
```

```
    "lambda-function-arn": "arn:aws:lambda:ap-
southeast-2:123456789012:function:lambda_activity"
    }
}
```

通过 Amazon CloudWatch Events 获取延迟数据通知

当您使用来自指定时间段的数据创建数据集时，有些消息数据可能无法及时到达以进行处理。为了允许延迟，您可通过应用 queryAction (SQL 查询)，在创建数据集时指定 QueryFilter 的 deltaTime 偏移量。AWS IoT Analytics 仍会处理在增量时间内到达的数据，并且您的数据集内容存在时间延迟。延迟数据通知功能允许 AWS IoT Analytics 在数据晚于增量时间到达时，通过 Amazon CloudWatch Events 发送通知。

您可以使用 AWS IoT Analytics 控制台、API、AWS Command Line Interface(AWS CLI) 或 AWSSDK 为数据集指定延迟数据规则。

在 AWS IoT Analytics API 中，LateDataRuleConfiguration 对象表示数据集的延迟数据规则设置。此对象是与 CreateDataset 和 UpdateDataset API 操作关联的 Dataset 对象的一部分。

参数

当您使用 AWS IoT Analytics 为数据集创建延迟数据规则时，必须指定以下信息：

ruleConfiguration (**LateDataRuleConfiguration**)

包含延迟数据规则配置信息的结构。

deltaTimeSessionWindowConfiguration

包含增量时间会话窗口的配置信息的结构。

DeltaTime 指定时间间隔。您可以通过 DeltaTime 使用上次执行后到达数据存储的数据创建数据集内容。有关 DeltaTime 的示例，请参阅使用增量窗口（CLI）创建 SQL 数据集。

timeoutInMinutes

一个时间间隔。您可以使用 timeoutInMinutes 让 AWS IoT Analytics 可以批量处理上次执行后生成的延迟数据通知。AWS IoT Analytics 一次向 CloudWatch Events 发送一批通知。

类型：整数

有效范围：1-60

`ruleName`

延迟数据规则的名称。

类型：字符串

> **Important**
> 要指定 `lateDataRules`，数据集必须使用 `DeltaTime` 筛选器。

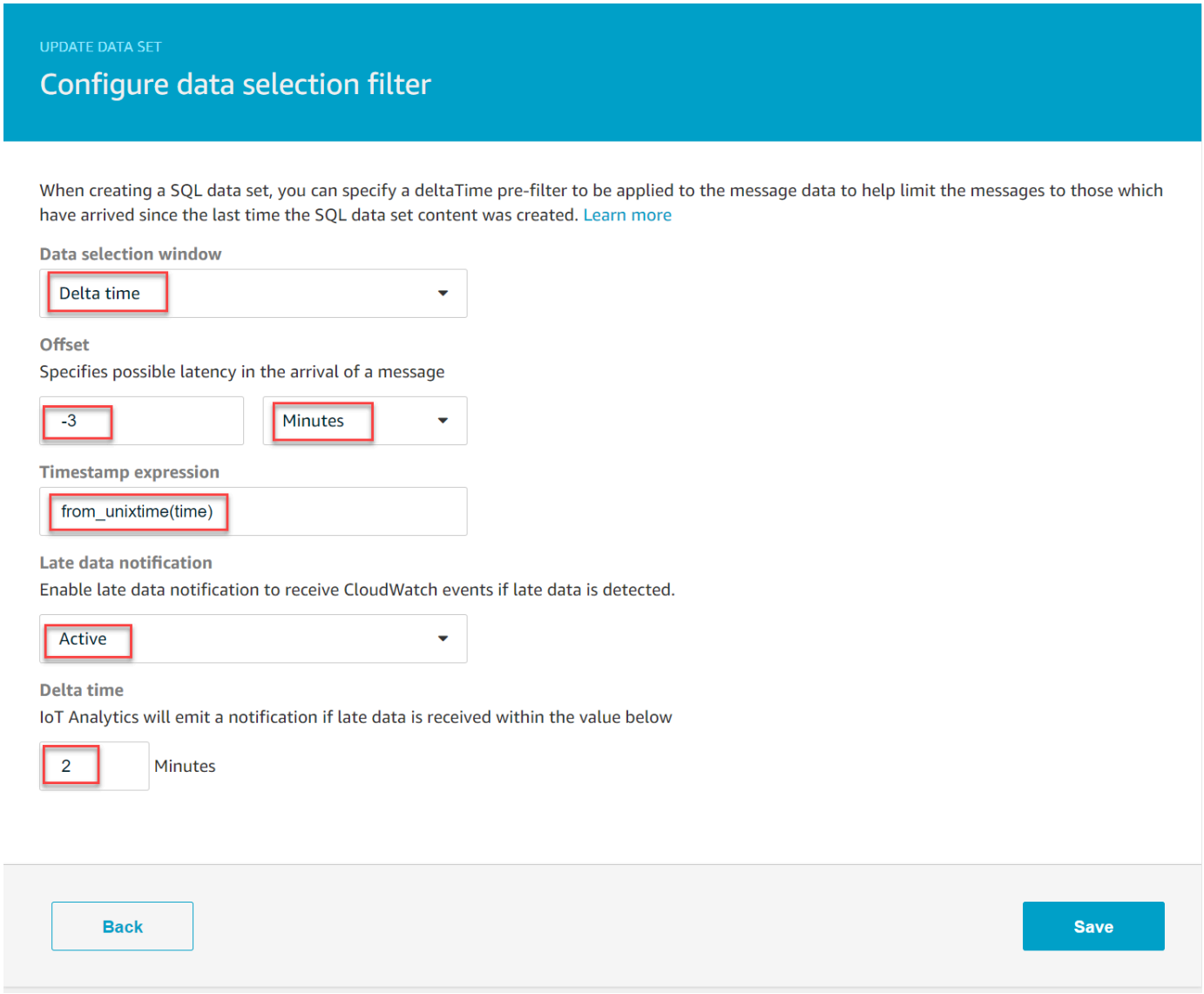
配置延迟数据规则(控制台)

以下步骤介绍了如何使用 AWS IoT Analytics 控制台配置数据集的延迟数据规则。

要配置延迟数据规则

1. 登录到 AWS IoT Analytics 控制台。
2. 在导航窗格中，选择 数据集。
3. 在数据集下，选择目标数据集。
4. 在导航窗格中，选择 详细信息。
5. 在 增量窗口部分中，选择编辑。
6. 在配置数据选择筛选条件下，执行以下操作：
   a. 对于数据选择窗口，选择增量时间。
   b. 对于偏移，输入一个时间段，然后选择一个单位。
   c. 对于时间戳表达式，输入一个表达式。这可以是时间戳字段的名称，也可以是生成时间的 SQL 表达式，例如 *`from_unixtime(time)`*。

      有关如何编写时间戳表达式的更多信息，请参阅《Presto 0.172 文档》中的日期与时间函数和运算符。
   d. 对于延迟数据通知，请选择激活。
   e. 对于增量时间，请输入一个整数。有效范围为 1-60。
   f. 选择 Save（保存）。



配置延迟数据规则(CLI)

在 AWS IoT Analytics API 中，`LateDataRuleConfiguration` 对象表示数据集的延迟数据规则设置。此对象是与 `CreateDataset` 和 `UpdateDataset` 关联的 `Dataset` 对象的一部分。您可以使用 API、AWS CLI、或 AWSSDK 为数据集指定延迟数据规则。以下示例使用 AWS CLI。

要使用指定的延迟数据规则创建数据集，请运行以下命令。以下命令假定该 `dataset.json` 文件位于当前目录中。

> **Note**
>
> 您可以使用 UpdateDataset API 来更新现有数据集。

```
aws iotanalytics create-dataset --cli-input-json file://dataset.json
```

该dataset.json文件应包含以下内容：

- 将 *demo_dataset* 替换为目标数据集名称。
- 将 *demo_datastore* 替换为目标数据存储名称。
- 将 *from_unixtime(time)* 替换为时间戳字段名称或可生成时间的 SQL 表达式。

  有关如何编写时间戳表达式的更多信息，请参阅《Presto 0.172 文档》中的日期与时间函数和运算符。
- 将*timeout* 替换为 1–60 之间的整数。
- 将 *demo_rule* 替换为任意名称。

```
{
    "datasetName": "demo_dataset",
    "actions": [
        {
            "actionName": "myDatasetAction",
            "queryAction": {
                "filters": [
                    {
                        "deltaTime": {
                            "offsetSeconds": -180,
                            "timeExpression": "from_unixtime(time)"
                        }
                    }
                ],
                "sqlQuery": "SELECT * FROM demo_datastore"
            }
        }
    ],
    "retentionPeriod": {
        "unlimited": false,
        "numberOfDays": 90
```

```
    },
    "lateDataRules": [
        {
            "ruleConfiguration": {
                "deltaTimeSessionWindowConfiguration": {
                    "timeoutInMinutes": timeout
                }
            },
            "ruleName": "demo_rule"
        }
    ]
}
```

订阅接收延迟数据通知

您可在 CloudWatch Events 中创建规则，定义如何处理从 AWS IoT Analytics 发送的延迟数据通知。当 CloudWatch Events 收到通知时，它会调用您规则中定义的指定目标操作。

创建 CloudWatch Events 规则的先决条件

在为 AWS IoT Analytics 创建 CloudWatch Events 规则之前，您应执行以下操作：

- 熟悉 CloudWatch Events 中的事件、规则和目标。
- 创建和配置由 CloudWatch Events 规则调用的目标。规则可调用多类目标，例如：
  - Amazon Kinesis Streams
  - AWS Lambda 函数
  - Amazon Simple Notification Service (Amazon SNS) 主题
  - Amazon Simple Queue Service (Amazon SQS) 队列

  CloudWatch Events 规则及关联目标必须位于 AWS IoT Analytics 资源创建所在 AWS 区域。有关更多信息，请参阅 AWS 一般参考 中的服务端点和配额。

有关更多信息，请参阅 Amazon CloudWatch Events 用户指南中的什么是 CloudWatch Events？和 Amazon CloudWatch Events 入门。

延迟数据通知事件

延迟数据通知的事件使用以下格式。

```
{
```

```
 "version": "0",
 "id": "7f51dfa7-ffef-97a5-c625-abddbac5eadd",
 "detail-type": "IoT Analytics Dataset Lifecycle Notification",
 "source": "aws.iotanalytics",
 "account": "123456789012",
 "time": "2020-05-14T02:38:46Z",
 "region": "us-east-2",
 "resources": ["arn:aws:iotanalytics:us-east-2:123456789012:dataset/demo_dataset"],
 "detail": {
  "event-detail-version": "1.0",
  "dataset-name": "demo_dataset",
  "late-data-rule-name": "demo_rule",
  "version-ids": ["78244852-8737-4650-aa4d-3071a01338fa"],
  "message": null
 }
}
```

创建 CloudWatch Events 规则以接收延迟数据通知

以下步骤介绍了如何创建规则来向 Amazon SQS 队列发送 AWS IoT Analytics 延迟数据通知。

要创建 CloudWatch Events 规则

1. 登录 Amazon CloudWatch 控制台
2. 在导航窗格中的事件下，选择规则。
3. 在 规则 页面，选择 创建规则。
4. 在事件源下，选择事件模式。
5. 在生成事件模式以按服务匹配事件部分，执行以下操作：
   a. 对于服务名称，请选择 IoT Analytics
   b. 对于事件类型，请选择 IoT Analytics 数据集生命周期通知。
   c. 选择特定数据集名称，然后输入目标数据集的名称。
6. 在目标下，选择添加目标*。
7. 选择 SQS 队列，然后执行以下操作：
   - 对于队列*，选择目标队列。
8. 选择 Configure details (配置详细信息)。
9. 在 步骤 2：配置规则详细信息 页面上，输入名称和描述。
10. 选择 Create rule（创建规则）。

使用 AWS IoT Analytics 记录 AWS CloudTrail API 调用

AWS IoT Analytics 与 AWS CloudTrail 集成，后者是在 AWS 中记录用户、角色或 AWS IoT Analytics 服务所执行操作的服务。CloudTrail 将对 AWS IoT Analytics 的 API 调用子集作为事件捕获，包括来自 AWS IoT Analytics 控制台的调用和对 AWS IoT Analytics API 的代码调用。如果您创建跟踪记录，则可以使 CloudTrail 事件持续传送到 Amazon S3 存储桶（包括 AWS IoT Analytics 的事件）。如果您不配置跟踪记录，则仍可在 CloudTrail 控制台中的 Event history（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 AWS IoT Analytics 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》。

AWS CloudTrail 中的 AWS IoT Analytics 信息

在您创建 AWS 账户时，将在该账户上启用 CloudTrail。当 AWS IoT Analytics 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 AWS 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 AWS 账户中的事件（包括 AWS IoT Analytics 的事件），请创建跟踪。通过跟踪记录，CloudTrail 可将日志文件传送至 Amazon S3 存储桶。默认情况下，在控制台中创建跟踪时，此跟踪应用于所有区域。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）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：

- 创建跟踪概览
- CloudTrail 支持的服务和集成
- 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

AWS IoT Analytics 支持将以下操作记录为 CloudTrail 日志文件中的事件：

- CancelPipelineReprocessing
- CreateChannel
- CreateDataset
- CreateDatasetContent
- CreateDatastore

- CreatePipeline
- DeleteChannel
- DeleteDataset
- DeleteDatasetContent
- DeleteDatastore
- DeletePipeline
- DescribeChannel
- DescribeDataset
- DescribeDatastore
- DescribeLoggingOptions
- DescribePipeline
- GetDatasetContent
- ListChannels
- ListDatasets
- ListDatastores
- ListPipelines
- PutLoggingOptions
- RunPipelineActivity
- SampleChannelData
- StartPipelineReprocessing
- UpdateChannel
- UpdateDataset
- UpdateDatastore
- UpdatePipeline

每个事件或日志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 用户凭证发出的。
- 请求是使用角色还是联合身份用户的临时安全凭证发出的。
- 请求是否由其它 AWS 服务发出。

有关更多信息，请参阅 CloudTrail userIdentity 元素。

了解 AWS IoT Analytics 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 S3 存储桶。CloudTrail 日志文件包含一个或多个日志条目。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。

下面的示例显示了一个 CloudTrail 日志条目，该条目说明了 `CreateChannel` 操作。

```
{
"eventVersion": "1.05",
"userIdentity": {
"type": "AssumedRole",
"principalId": "ABCDE12345FGHIJ67890B:AnalyticsChannelTestFunction",
"arn": "arn:aws:sts::123456789012:assumed-role/AnalyticsRole/
AnalyticsChannelTestFunction",
"accountId": "123456789012",
"accessKeyId": "ABCDE12345FGHIJ67890B",
"sessionContext": {
"attributes": {
 "mfaAuthenticated": "false",
 "creationDate": "2018-02-14T23:43:12Z"
},
"sessionIssuer": {
 "type": "Role",
 "principalId": "ABCDE12345FGHIJ67890B",
 "arn": "arn:aws:iam::123456789012:role/AnalyticsRole",
 "accountId": "123456789012",
 "userName": "AnalyticsRole"
}
}
},
"eventTime": "2018-02-14T23:55:14Z",
"eventSource": "iotanalytics.amazonaws.com",
"eventName": "CreateChannel",
"awsRegion": "us-east-1",
"sourceIPAddress": "198.162.1.0",
"userAgent": "aws-internal/3 exec-env/AWS_Lambda_java8",
"requestParameters": {
"channelName": "channel_channeltest"
},
```

```
"responseElements": {
"retentionPeriod": {
"unlimited": true
},
"channelName": "channel_channeltest",
"channelArn": "arn:aws:iotanalytics:us-east-1:123456789012:channel/channel_channeltest"
},
"requestID": "7f871429-11e2-11e8-9eee-0781b5c0ac59",
"eventID": "17885899-6977-41be-a6a0-74bb95a78294",
"eventType": "AwsApiCall",
"recipientAccountId": "123456789012"
}
```

下面的示例显示了一个 CloudTrail 日志条目，该条目说明了 `CreateDataset` 操作。

```
{
"eventVersion": "1.05",
"userIdentity": {
"type": "AssumedRole",
"principalId": "ABCDE12345FGHIJ67890B:AnalyticsDatasetTestFunction",
"arn": "arn:aws:sts::123456789012:assumed-role/AnalyticsRole/
AnalyticsDatasetTestFunction",
"accountId": "123456789012",
"accessKeyId": "ABCDE12345FGHIJ67890B",
"sessionContext": {
"attributes": {
 "mfaAuthenticated": "false",
 "creationDate": "2018-02-14T23:41:36Z"
},
"sessionIssuer": {
 "type": "Role",
 "principalId": "ABCDE12345FGHIJ67890B",
 "arn": "arn:aws:iam::123456789012:role/AnalyticsRole",
 "accountId": "123456789012",
 "userName": "AnalyticsRole"
}
}
},
"eventTime": "2018-02-14T23:53:39Z",
"eventSource": "iotanalytics.amazonaws.com",
"eventName": "CreateDataset",
"awsRegion": "us-east-1",
"sourceIPAddress": "198.162.1.0",
```

```
"userAgent": "aws-internal/3 exec-env/AWS_Lambda_java8",
"requestParameters": {
"datasetName": "dataset_datasettest"
},
"responseElements": {
"datasetArn": "arn:aws:iotanalytics:us-east-1:123456789012:dataset/
dataset_datasettest",
"datasetName": "dataset_datasettest"
},
"requestID": "46ee8dd9-11e2-11e8-979a-6198b668c3f0",
"eventID": "5abe21f6-ee1a-48ef-afc5-c77211235303",
"eventType": "AwsApiCall",
"recipientAccountId": "123456789012"
}
```

合规性验证 AWS IoT Analytics

要了解是否属于特定合规计划的范围，请参阅AWS 服务 “按合规计划划分的范围” ”，然后选择您感兴趣的合规计划。 AWS 服务 有关一般信息，请参阅AWS 合规计划AWS。

您可以使用下载第三方审计报告 AWS Artifact。有关更多信息，请参阅中的 “下载报告” 中的 “ AWS Artifact。

您在使用 AWS 服务 时的合规责任取决于您的数据的敏感性、贵公司的合规目标以及适用的法律和法规。 AWS 提供了以下资源来帮助实现合规性：

- 安全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，并提供了部署以安全性和合规性为重点 AWS 的基准环境的步骤。
- 在 A@@ mazon Web Services 上构建 HIPAA 安全与合规性 — 本白皮书描述了各公司如何使用 AWS 来创建符合 HIPAA 资格的应用程序。

  > **Note**
  >
  > 并非所有 AWS 服务 人都符合 HIPAA 资格。有关更多信息，请参阅符合 HIPAA 要求的服务参考。

- AWS 合AWS 规资源 — 此工作簿和指南集可能适用于您的行业和所在地区。
- AWS 客户合规指南 — 从合规角度了解责任共担模式。这些指南总结了保护的最佳实践， AWS 服务 并将指南映射到跨多个框架（包括美国国家标准与技术研究院 (NIST)、支付卡行业安全标准委员会 (PCI) 和国际标准化组织 (ISO)）的安全控制。

- 使用AWS Config 开发人员指南中的规则评估资源 — 该 AWS Config 服务评估您的资源配置在多大程度上符合内部实践、行业准则和法规。
- AWS Security Hub— 这 AWS 服务 可以全面了解您的安全状态 AWS。Security Hub 通过安全控件评估您的 AWS 资源并检查其是否符合安全行业标准和最佳实践。有关受支持服务及控件的列表，请参阅 Security Hub 控件参考。
- Amazon GuardDuty — 它通过监控您的 AWS 账户环境中是否存在可疑和恶意活动，来 AWS 服务检测您的工作负载、容器和数据面临的潜在威胁。 GuardDuty 通过满足某些合规性框架规定的入侵检测要求，可以帮助您满足各种合规性要求，例如 PCI DSS。
- AWS Audit Manager— 这 AWS 服务 可以帮助您持续审计 AWS 使用情况，从而简化风险管理以及对法规和行业标准的合规性。

韧性在 AWS IoT Analytics

AWS 全球基础设施是围绕 AWS 区域和可用区构建的。 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。

中的基础设施安全 AWS IoT Analytics

作为一项托管服务 AWS IoT Analytics ，受 AWS 全球网络安全的保护。有关 AWS 安全服务以及如何 AWS 保护基础设施的信息，请参阅AWS 云安全。要使用基础设施安全的最佳实践来设计您的 AWS 环境，请参阅 S AWS ecurity Pillar Well-Architected Fram ework 中的基础设施保护。

您可以使用 AWS 已发布的 API 调用通过网络进行访问。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。

AWS IoT Analytics 配额

《AWS 一般参考 指南》为 AWS 账户的 AWS IoT Analytics 提供默认配额。除非另行指定，否则每个配额将基于AWS区域。有关更多信息，请参阅《AWS 一般参考 指南》中的AWS IoT Analytics 服务终端节点和配额及AWS服务配额。

要请求提高服务配额，请在支持中心控制台中提交支持案例。有关更多信息，请参阅 Service Quotas 用户指南 中的请求增加配额。

AWS IoT Analytics 命令

阅读本主题以了解 AWS IoT Analytics 的 API 操作，包括支持的 Web 服务协议的示例请求、响应和错误。

AWS IoT Analytics 操作

您可以使用 AWS IoT Analytics API 命令收集、处理、存储和分析 IoT 数据。有关更多信息，请参阅《AWS IoT Analytics API 参考》中 AWS IoT Analytics 支持的“操作”。

《AWS CLI 命令参考》中的 AWS IoT Analytics 部分包括可用于管理和操作 AWS IoT Analytics 的 AWS CLI 命令。

AWS IoT Analytics 数据

您可以使用 AWS IoT Analytics Data API 命令通过 AWS IoT Analytics、`channel`、`pipeline`、`datastore` 和 `dataset` 来执行高级活动。有关更多信息，请参阅《AWS IoT Analytics API 参考》中 AWS IoT Analytics 数据支持的数据类型。

排除 AWS IoT Analytics 的故障

请参阅下一节，对错误进行故障排查，并找到可能的解决方案来解决 AWS IoT Analytics 的问题。

主题

如何知道我的消息是否进入 AWS IoT Analytics？

检查是否正确配置了通过规则引擎将数据注入通道的规则。

```
aws iot get-topic-rule --rule-name your-rule-name
```

该响应应当与以下内容相似。

```
{
    "ruleArn": "arn:aws:iot:us-west-2:your-account-id:rule/your-rule-name",
```

```
    "rule": {
        "awsIotSqlVersion": "2016-03-23",
        "sql": "SELECT * FROM 'iot/your-rule-name'",
        "ruleDisabled": false,
        "actions": [
            {
                "iotAnalytics": {
                    "channelArn":
 "arn:aws:iotanalytics:region:your_account_id:channel/your-channel-name"
                }
            }
        ],
        "ruleName": "your-rule-name"
    }
}
```

确保规则中使用的区域和通道名称正确。要确保数据进入了规则引擎并且正确执行了规则，您可能需要添加新目标，以在 Amazon S3 存储桶中临时存储传入消息。

为什么管道会丢失消息？ 如何修复此问题？

- 活动收到了无效的 JSON 输入：

  除了 Lambda 活动之外的所有活动，专门需要有效 JSON 字符串作为输入。如果活动收到的 JSON 无效，则将丢弃消息，并且消息不会传入数据存储。确保您将有效的 JSON 消息提取到服务中。在使用二进制输入时，确保管道中的第一个活动是将二进制数据转换为有效 JSON 的 Lambda 活动，然后再将其传递到下一个活动或者存储到数据存储中。有关更多信息，请参阅 Lambda 函数示例 2。

- Lambda 活动调用的 Lambda 函数权限不足：

  确保对 Lambda 活动中的每个 Lambda 函数具有权限，可以从 AWS IoT Analytics 服务调用该函数。您可以使用以下 AWS CLI 命令授予权限。

  ```
  aws lambda add-permission --function-name <name> --region <region> --statement-id
   <id> --principal iotanalytics.amazonaws.com --action lambda:InvokeFunction
  ```

- 不正确地定义了某个筛选条件或 removeAttribute 活动：

  确保任意 `filter` 或 `removeAttribute` 活动的定义正确。如果您筛选出了某条消息或者从某条消息中删除了所有属性，该消息不会添加到数据存储。

为什么我的数据存储中没有任何数据？

- 数据提取与数据可用之间存在延迟：

  将数据提取到通道中之后，可能需要几分钟时间，然后数据才会在数据存储中可用。该时间因管道活动数以及管道中的任何自定义 Lambda 活动的定义而异。
- 管道正将消息筛选掉：

  确保您未在管道中丢弃消息。(请参阅前面的问题和答复。)
- 数据集查询不正确：

  确保从数据存储生成数据集的查询正确无误。从查询中删除任何不必要的筛选条件，确保数据可以进入数据存储。

为什么我的数据集会只显示 `__dt`？

- 此列由服务自动添加，并包含数据的大致提取时间。它可用于优化查询。如果数据集中除此之外不含任何其他内容，请参阅前面的问题和答复。

如何编写由数据集完成操作驱动的事件代码？

- 您必须根据 `describe-dataset` 命令设置轮询，以检查具有特定时间戳的数据集的状态是否为已成功。

如何正确配置笔记本实例来使用 AWS IoT Analytics？

按照以下步骤确保您用于创建笔记本实例的 IAM 角色具有所需权限：

1. 转到 SageMaker 控制台并创建笔记本实例。
2. 填写详细信息，然后选择 create a new role (创建新角色)。记下角色的 ARN。
3. 创建笔记本实例。这还会创建 SageMaker 可使用的角色。
4. 转到 IAM 控制台并修改新创建的 Sagemaker 角色。当您打开该角色时，它应具有一个托管策略。
5. 单击 添加内联策略，选择 作为服务，然后在读取权限下，选择 GetDatasetContent。
6. 检查策略，添加策略名称，然后对它选择 create (创建)。新创建的角色现在具有策略权限，可以从 AWS IoT Analytics 读取数据集。

7. 转到 AWS IoT Analytics 控制台并在笔记本实例中创建笔记本。
8. 等待笔记本实例进入“In Service”(正在服务) 状态。
9. 选择 create notebooks (创建笔记本)，然后选择您创建的笔记本实例。这将使用可访问您的数据集的选定模板创建 Jupyter notebook。

为什么我无法在实例中创建笔记本？

- 确保您使用正确的 IAM 策略创建了笔记本实例。(按照上一个问题中的步骤操作。)
- 请确保笔记本实例处于“In Service”(正在服务) 状态。在您创建实例时，它开始处于“Pending”(待处理) 状态。通常大约需要 5 分钟进入“In Service”(正在服务) 状态。如果笔记本实例在大约 5 分钟之后进入“Failed”(失败) 状态，请重新检查权限。

为什么我在 Amazon QuickSight 中看不到我的数据集？

Amazon QuickSight 可能需要权限才能读取您的 AWS IoT Analytics 数据集内容。要授予权限，请按照以下步骤操作。

1. 在 Amazon QuickSight 的右上角选择您的账户名称，然后选择管理 QuickSight。
2. 在左侧导航窗格中，选择 安全和权限。在 QuickSight 访问 AWS 服务下，验证是否已授予 AWS IoT Analytics 访问权限。
   a. 如果 AWS IoT Analytics 没有访问权限，请选择添加或删除。
   b. 选中 AWS IoT Analytics 旁边的复选框，然后选择 更新。这会向 Amazon QuickSight 提供对您数据集的读取权限。
3. 请重试以可视化您的数据。

请务必为 AWS IoT Analytics 和 Amazon QuickSight 选择相同的 AWS 区域。否则，您可能会在访问 AWS 资源时遇到问题。有关支持的区域列表，请参阅 Amazon Web Services 一般参考 中的AWS IoT Analytics 端点和配额以及 Amazon QuickSight 端点和配额。

为什么我在现有的 Jupyter Notebook 上看不到容器化按钮？

- 这是因为缺少 AWS IoT Analytics 容器化插件。如果您的 SageMaker 笔记本实例是在 2018 年 8 月 23 日之前创建的，则需按照笔记本容器化中的说明手动安装插件。

- 从 AWS IoT Analytics 控制台中创建 SageMaker 笔记本实例或手动安装该实例后，如果看不到容器化按钮，请与 AWS IoT Analytics 技术支持联系。

为什么我的容器化插件安装失败？

- 通常，插件安装失败是因为 SageMaker 笔记本实例缺少权限。有关笔记本实例需要的权限，请参阅权限以及向笔记本实例角色添加所需权限。如果问题仍然存在，请从 AWS IoT Analytics 控制台创建一个新笔记本实例。
- 如果安装插件期间出现以下消息，您可以安全地忽略日志中的这一消息：“To initialize this extension in the browser every time the notebook (or other app) loads”(每当笔记本(或其他应用)加载时，在浏览器中初始化此扩展程序)。

为什么我的容器化插件引发错误？

- 容器化失败并生成错误的原因有很多。在进行笔记本容器化之前，确保您使用的是正确的内核。容器化内核以“Containerized”前缀开头。
- 由于插件会在 ECR 存储库中创建并保存一个 Docker 映像，请确保您的笔记本实例角色有足够权限来读取、列出和创建 ECR 存储库。有关笔记本实例需要的权限，请参阅权限以及向笔记本实例角色添加所需权限。
- 此外，还要确保存储库的名称符合 ECR 要求。ECR 存储库名称必须以字母开头，并且只能包含小写字母、数字、连字符、下划线和正斜杠。
- 如果容器化过程失败并显示错误“This instance has insufficient free space to run containerization”(此实例用于运行容器化的可用空间不足)，请尝试使用更大的实例来解决问题。
- 如果显示连接错误或映像创建错误，请重试。如果问题仍然存在，请重新启动实例并安装最新的插件版本。

为什么我在容器化期间看不到我的变量？

- AWS IoT Analytics 容器化插件在运行具有“容器化”内核的笔记本后，可自动识别笔记本中的所有变量。使用其中一个容器化内核运行笔记本，然后执行容器化。

我可以将哪些变量作为输入添加到我的容器中？

- 您可以将要在运行时期间修改其值的任何变量作为输入添加到容器中。这样，您就可以使用需要在创建数据集时提供的不同参数来运行相同的容器。AWS IoT Analytics 容器化 Jupyter 插件通过在笔记本中自动识别变量并使其在容器化过程中可用来简化此过程。

我如何将我的容器输出设置为后续分析的输入？

- 每次运行容器数据集时，都会创建一个可存储已执行构件的特定 S3 位置。要访问此输出位置，请在您的容器数据集中创建一个类型为 `outputFileUriValue` 的变量。该变量的值应该是用于存储其他输出文件的 S3 路径。要在后续运行时访问这些已保存的构件，您可以使用代码 `getDatasetContent` API 并选取后续运行所需的相应输出文件。

我的容器数据集为什么会失败？

- 确保将正确的 `executionRole` 传递给容器数据集。`executionRole` 的信任策略必须同时包含 `iotanalytics.amazonaws.com` 和 `sagemaker.amazonaws.com`。
- 如果 `AlgorithmError` 显示为失败原因，请尝试手动调试容器代码。如果容器代码中有错误或执行角色无权执行容器，则会出现这种情况。如果您通过使用 AWS IoT Analytics Jupyter 插件来执行容器化，请创建一个与 containerDataset 的 executionRole 具有相同角色的新 SageMaker 笔记本实例，并尝试手动运行该笔记本。如果容器是在 Jupyter 插件外创建的，请尝试手动运行代码并限制 executionRole 的权限。

文档历史记录

下表介绍了2020 年 11 月 3 日之后对 AWS IoT Analytics 用户指南 作出的重要更改。如需对此文档更新的通知，您可以订阅 RSS 源。

变更	说明	日期
区域发布	AWS IoT Analytics 现已在亚太地区 (孟买)区域 提供。	2021 年 8 月 18 日
使用JOIN查询	此更新允许您使用 JOIN 查询 AWS IoT Analytics 数据集。	2021 年 7 月 27 日
与 AWS IoT SiteWise 集成	您现可使用 AWS IoT Analytics 来查询 AWS IoT SiteWise 数据。	2021 年 7 月 27 日
自定义分区	AWS IoT Analytics 现在通常支持根据消息属性或通过管道活动添加的属性对数据进行分区。	2021 年 6 月 14 日
重新处理通道消息	此更新使您能够重新处理指定 Amazon S3 对象中的通道数据。	2020 年 12 月 15 日
Parquet 架构	AWS IoT Analytics 数据存储现支持 Parquet 文件格式。	2020 年 12 月 15 日
使用 CloudWatch Events 进行监控	当 AWS Lambda 活动期间发生运行时系统错误时，AWS IoT Analytics 会自动将事件发布到 Amazon CloudWatch Events。	2020 年 12 月 15 日
延迟数据通知	您可以使用此功能，在延迟数据到达时，通过 Amazon CloudWatch Events 接收通知。	2020 年 11 月 9 日

区域发布	AWS IoT Analytics 在中国（北京）推出。	2020 年 11 月 4 日

早期更新

下表列出了 2020 年 11 月 4 日之后对 AWS IoT Analytics 用户指南 的一些重要更改。

更改	说明	日期
区域发布	AWS IoT Analytics 在亚太地区（悉尼）区域推出。	2020 年 7 月 16 日
更新	重新组织了文档。	2020 年 5 月 7 日

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。