AWS IoT Analytics

AWS IoT Analytics User Guide

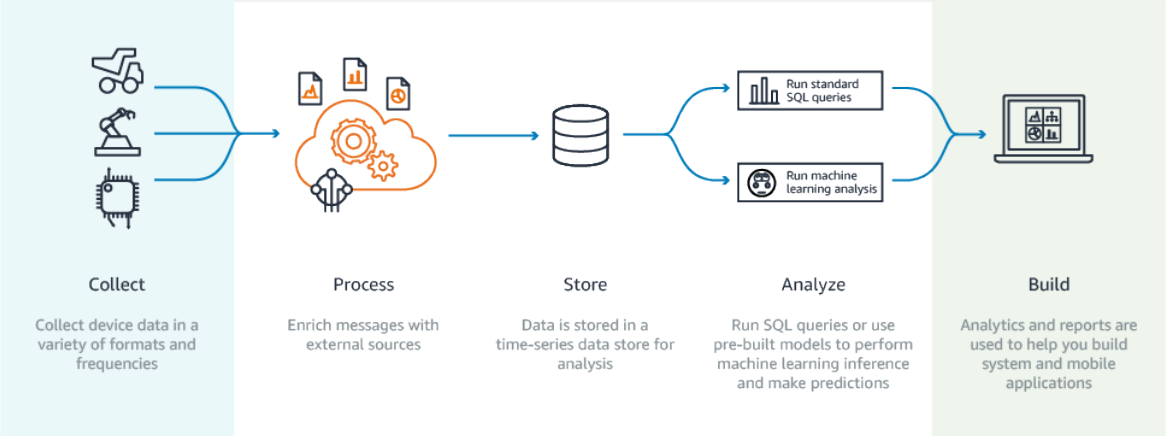
Table of Contents

What Is AWS IoT Analytics?

AWS IoT Analytics automates the steps required to analyze data from IoT devices. It filters, transforms, and enriches IoT data before storing it in a time-series data store for analysis. You can set up the service to collect only the data you need from your devices, apply mathematical transforms to process the data, and enrich the data with device-specific metadata such as device type and location before storing it. Then, you can analyze your data by running queries using the built-in SQL query engine, or perform more complex analytics and machine learning inference. AWS IoT Analytics enables advanced data exploration through integration with Jupyter Notebooks. It also enables data visualization through integration with Amazon QuickSight. (Amazon QuickSight is available in these regions.)

Traditional analytics and business intelligence tools are designed to process structured data. IoT data often comes from devices that record noisy processes (such as temperature, motion, or sound). As a result the data from these devices can have significant gaps, corrupted messages, and false readings that must be cleaned up before analysis can occur. Also, IoT data is often only meaningful in the context of other data from external sources. AWS IoT Analytics enables you to address these issues and collect large amounts of device data, process messages, and store them. You can then query the data and analyze it. AWS IoT Analytics includes pre-built models for common IoT use cases so you can answer questions like which devices are about to fail or which customers are at risk of abandoning their wearable devices.

How to Use AWS IoT Analytics



Key Features

Collect

- **Integrated with AWS IoT Core** – AWS IoT Analytics is fully integrated with AWS IoT Core so it can receive messages from connected devices as they stream in.
- **Use a batch API to add data from any source** – AWS IoT Analytics can receive data from any source through HTTP. That means that any device or service that is connected to the internet can send data to AWS IoT Analytics. (BatchPutMessage (p. 135))
- **Collect only the data you want to store and analyze** – You can use the AWS IoT Analytics console to configure AWS IoT Analytics to receive messages from devices through MQTT topic filters in various formats and frequencies. AWS IoT Analytics validates that the data is within specific

parameters you define and creates channels. Then, the service routes the channels to appropriate pipelines for message processing, transformation, and enrichment.

Process

- **Cleanse and filter** – AWS IoT Analytics lets you define AWS Lambda functions that are triggered when AWS IoT Analytics detects missing data, so you can run code to estimate and fill gaps. You can also define max/min filters and percentile thresholds to remove outliers in your data.
- **Transform** – AWS IoT Analytics can transform messages using mathematical or conditional logic you define, so you can perform common calculations like Celsius into Fahrenheit conversion.
- **Enrich** – AWS IoT Analytics can enrich data with external data sources such as a weather forecast, and then route the data to the AWS IoT Analytics data store.

Store

- **Time-Series Data Store** - AWS IoT Analytics stores the device data in an optimized time-series data store for faster retrieval and analysis. You can also manage access permissions, implement data retention policies and export your data to external access points.
- **Store Processed and Raw Data** - AWS IoT Analytics stores the processed data and also automatically stores the raw ingested data so you can process it at a later time.

Analyze

- **Run Ad-Hoc SQL Queries** - AWS IoT Analytics provides a SQL query engine so you can run ad-hoc queries and get results quickly. For example, you might want to run a quick query to find the number of active users for each device in your fleet.
- **Time-Series Analysis** - AWS IoT Analytics supports time-series analysis so you can analyze the performance of devices over time and understand how and where they are being used, continuously monitor device data to predict maintenance issues, and monitor sensors to predict and react to environmental conditions.
- **Hosted Notebooks for Sophisticated Analytics and Machine Learning** - AWS IoT Analytics includes support for hosted notebooks in Jupyter Notebooks for statistical analysis and machine learning. The service includes a set of notebook templates that contain AWS-authored machine learning models and visualizations to help you get started with IoT use cases related to device failure profiling, forecasting events such as low usage that might signal the customer will abandon the product, or segmenting devices by customer usage levels (for example heavy users, weekend users) or device health. After you author a notebook, you can containerize it and execute it on a schedule you specify (Automating Your Workflow (p. 67)).
- **Prediction** - You can do statistical classification through a method called logistic regression. You can also use Long-Short-Term Memory (LSTM), which is a powerful neural network technique for predicting the output or state of a process that varies over time. The pre-built notebook templates also support the K-means clustering algorithm for device segmentation, which clusters your devices into cohorts of like devices. These templates are typically used to profile device health and device state such as HVAC units in a chocolate factory or wear and tear of blades on a wind turbine. Again, these notebook templates can be containerized and executed on a schedule.

Build/Visualize

- **QuickSight Integration** - AWS IoT Analytics provides a connector to Amazon QuickSight so you can visualize your data sets in a QuickSight dashboard.
- **Console Integration** - You can also visualize the results or your ad-hoc analysis in the embedded Jupyter Notebooks in the AWS IoT Analytics' console.

AWS IoT Analytics Components and Concepts

Channel - collect the data

A channel collects data from an MQTT topic and archives the raw, unprocessed messages before publishing the data to a pipeline. You can also send messages to a channel directly with the

BatchPutMessage (p. 135) command. The unprocessed messages are stored in an Amazon S3 bucket managed by IoT Analytics or in one managed by you.

Pipeline - process the data

A pipeline consumes messages from a channel and enables you to process the messages before storing them in a data store. The processing steps, called **activities** (Pipeline Activities (p. 44)), perform transformations on your messages such as removing, renaming or adding message attributes, filtering messages based on attribute values, invoking your Lambda functions on messages for advanced processing or performing mathematical transformations to normalize device data.

Data store - store the data

Pipelines store their processed messages in a data store. A data store is not a database, but it is a scalable and queryable repository of your messages. You can have multiple data stores for messages coming from different devices or locations, or filtered by message attributes depending on your pipeline configuration and requirements. As with unprocessed channel messages, a data store's processed messages are stored in an Amazon S3 bucket managed by IoT Analytics or in one managed by you.

Data set - retrieve the data for analysis

You retrieve data from a data store by creating a data set. AWS IoT Analytics enables you to create a SQL data set or a container data set.

After you have a data set, you can explore and gain insights into your data through integration with Amazon QuickSight. Or you can perform more advanced analytical functions through integration with Jupyter Notebooks. Jupyter Notebooks provide powerful data science tools that can perform machine learning and a range of statistical analyses. For more information, see Notebook Templates (p. 30).

You can send data set contents to an Amazon Simple Storage Service (S3) bucket, enabling integration with your existing data lakes or access from in-house applications and visualization tools. You can also send data set contents as an input to AWS IoT Events, a service which enables you to monitor devices or processes for failures or changes in operation, and to trigger additional actions when such events occur.

SQL data set - retrieve the data using SQL actions

A SQL data set is similar to a materialized view from a SQL database. In fact, you create a SQL data set by applying a SQL action. SQL data sets can be generated automatically on a recurring schedule by specifying a trigger.

Container data set - combine data retrieval and analysis tools

A container data set enables you to automatically run your analysis tools and generate results (Automating Your Workflow (p. 67)). It brings together a SQL data set as input, a Docker container with your analysis tools and needed library files, input and output variables, and an optional schedule trigger. The input and output variables tell the executable image where to get the data and store the results. The trigger can run your analysis when a SQL data set finishes creating its content or according to a time schedule expression. A container data set automatically runs, generates and then saves the results of the analysis tools.

Trigger - automate data retrieval

You can automatically create a data set by specifying a trigger. The trigger can be a time interval (for example, create this data set every two hours) or when another data set's content has been created (for example, create this data set when "myOtherDataset" finishes creating its content). Or, you can generate data set content manually by calling `CreateDatasetContent`.

Docker container - everything needed to run

According to www.docker.com, "a container is a lightweight, stand-alone, executable package of a piece of software that includes everything needed to run it: code, runtime, system tools, system

libraries, settings." You can create your own Docker container to package your analysis tools or use options provided by Amazon SageMaker. You can store a container in an Amazon ECR registry that you specify so it is available to install on your desired platform. Containerizing A Notebook (p. 85) describes how to containerize a notebook. Docker containers are capable of running your custom analytical code prepared with Matlab, Octave, Wise.io, SPSS, R, Fortran, Python, Scala, Java, C++ and so on.

Delta windows - eliminate data overlaps

Delta windows are a series of user-defined, non-overlapping and contiguous time intervals. Delta windows enable you to create the data set content with, and perform analysis on, new data that has arrived in the data store since the last analysis. You create a delta window by setting the `deltaTime` in the `filters` portion of a `queryAction` of a data set (CreateDataset (p. 141)). Usually, you'll want to create the data set content automatically by also setting up a time interval trigger (`triggers:schedule:expression`). Basically, this enables you to filter messages that have arrived during a specific time window, so the data contained in messages from previous time windows doesn't get counted twice. See Example 6 – Creating a SQL dataset with a Delta Window (CLI): (p. 84).

How to Access AWS IoT Analytics

As part of AWS IoT, AWS IoT Analytics provides the following interfaces to enable your devices to generate data and your applications to interact with the data they generate:

AWS Command Line Interface (AWS CLI)

Run commands for AWS IoT Analytics on Windows, OS X, and Linux. These commands enable you to create and manage things, certificates, rules, and policies. To get started, see the AWS Command Line Interface User Guide. For more information about the commands for AWS IoT, see iot in the *AWS Command Line Interface Reference*.

> **Important**
> Use the **aws iotanalytics** *command* to interact with AWS IoT Analytics using the CLI. Use the **aws iot** *command* to interact with other parts of the IoT system using the CLI.

AWS IoT API

Build your IoT applications using HTTP or HTTPS requests. These API actions enable you to create and manage things, certificates, rules, and policies. For more information about the API actions for AWS IoT, see Actions in the *AWS IoT API Reference*.

AWS SDKs

Build your AWS IoT Analytics applications using language-specific APIs. These SDKs wrap the HTTP/HTTPS API and enable you to program in any of the supported languages. For more information, see AWS SDKs and Tools.

AWS IoT Device SDKs

Build applications that run on your devices that send messages to AWS IoT Analytics. For more information see AWS IoT SDKs.

AWS IoT Analytics Console

From building the components to visualizing the results, it's all available on the AWS IoT Analytics console.

Why Use AWS IoT Analytics?

Benefits

Run Queries on IoT Data

With AWS IoT Analytics, you can run simple, ad-hoc queries using the built-in AWS IoT Analytics SQL query engine. The service enables you to use standard SQL queries to extract data from the data store to answer questions like the average distance traveled for a fleet of connected vehicles or how many doors in a smart building are locked after 7pm. These queries can be re-used even if connected devices, fleet size, and analytic requirements change.

Run Time-Series Analytics

AWS IoT Analytics also supports time-series analyses so you can analyze the performance of devices over time and understand how and where they are being used. It enables you to continuously monitor device data to predict maintenance issues, and monitor sensors to predict and react to environmental conditions.

Data Storage Optimized for IoT

AWS IoT Analytics stores the processed device data in a time-series data store that is optimized to deliver the fast response times typically needed by IoT queries. The raw data is also automatically stored for later processing or to reprocess it for another use case.

Prepares Your IoT Data for Analysis

AWS IoT Analytics includes data preparation techniques that enable you to prepare and process your data for analysis. AWS IoT Analytics is integrated with AWS IoT Core so it is easy to ingest device data directly from connected devices. It can clean false readings, fill gaps in the data, and perform mathematical transformations of message data. As the data is ingested, AWS IoT Analytics can process it using conditional statements, filter data to collect just the data you want to analyze, and enrich it with information from the AWS IoT Registry. You can also use AWS Lambda functions to enrich your device data from external sources like the Weather Service, HERE Maps, Salesforce, or Amazon DynamoDB.

Tools for Machine Learning

AWS IoT Analytics enables you to apply machine learning to your IoT data with hosted Jupyter Notebooks. You can directly connect your IoT data to the notebook and build, train, and execute models right from the AWS IoT Analytics console without having to manage any of the underlying infrastructure. Using AWS IoT Analytics, you can apply machine learning algorithms to your device data to produce a health score for each device in your fleet. After you author a notebook, you can containerize it and execute it on a schedule you specify (Automating Your Workflow (p. 67)).

Use Cases

Predictive Maintenance

AWS IoT Analytics provides templates to build predictive maintenance models and apply them to your devices. For example, you can use AWS IoT Analytics to predict when heating and cooling systems are likely to fail on connected cargo vehicles so the vehicles can be rerouted to prevent shipment damage. Or, an auto manufacturer can detect which of its customers have worn brake pads and alert them to seek maintenance for their vehicles.

Proactive Replenishing of Supplies

AWS IoT Analytics lets you build IoT applications that can monitor inventories in real time. For example, a food and drink company can analyze data from food vending machines and proactively reorder merchandise whenever the supply is running low.

Process Efficiency Scoring

With AWS IoT Analytics, you can build applications that constantly monitor the efficiency of different processes and take action to improve the process. For example, a mining company can increase the efficiency of its ore trucks by maximizing the load for each trip. With AWS IoT Analytics, the company can identify the most efficient load for a location or truck over time, and then compare any deviations from the target load in real time, and better plan loading guidelines to improve efficiency.

Smart Agriculture

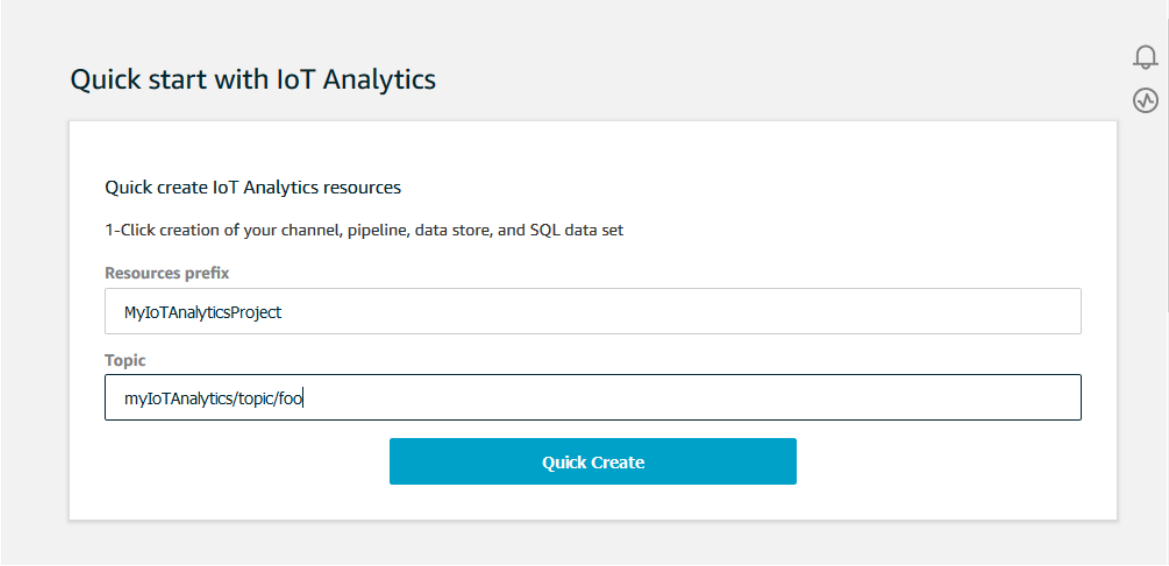
AWS IoT Analytics can enrich IoT device data with contextual metadata using AWS IoT Registry data or public data sources so that your analytis factors in time, location, temperature, altitude, and other environmental conditions. With that analysis, you can write models that output recommended actions for your devices to take in the field. For example, to determine when to water, irrigation systems might enrich humidity sensor data with data on rainfall, enabling more efficient water usage.

AWS IoT Analytics Console Quickstart Guide

This section shows you how to use the AWS IoT Analytics console to collect, store, process, and query your device data. Follow the instructions below to see details of how to create a channel, data store, pipeline and data set, and how to use the IoT Core console to send messages that will be ingested into AWS IoT Analytics.

> **Note**
> Be aware as you enter the names of AWS IoT Analytics entities (channel, data set, data store, and pipeline) in the steps that follow, that any upper-case letters you use are automatically changed to lower-case by the system. The names of entities must start with a lower-case letter and contain only lower-case letters, underscores and digits.

The AWS IoT Analytics console also has a **Quick start** feature that enables you to create a channel, data store, pipeline and data set with one click. Look for this page when you enter the AWS IoT Analytics console:



Sign in to the AWS IoT Analytics Console

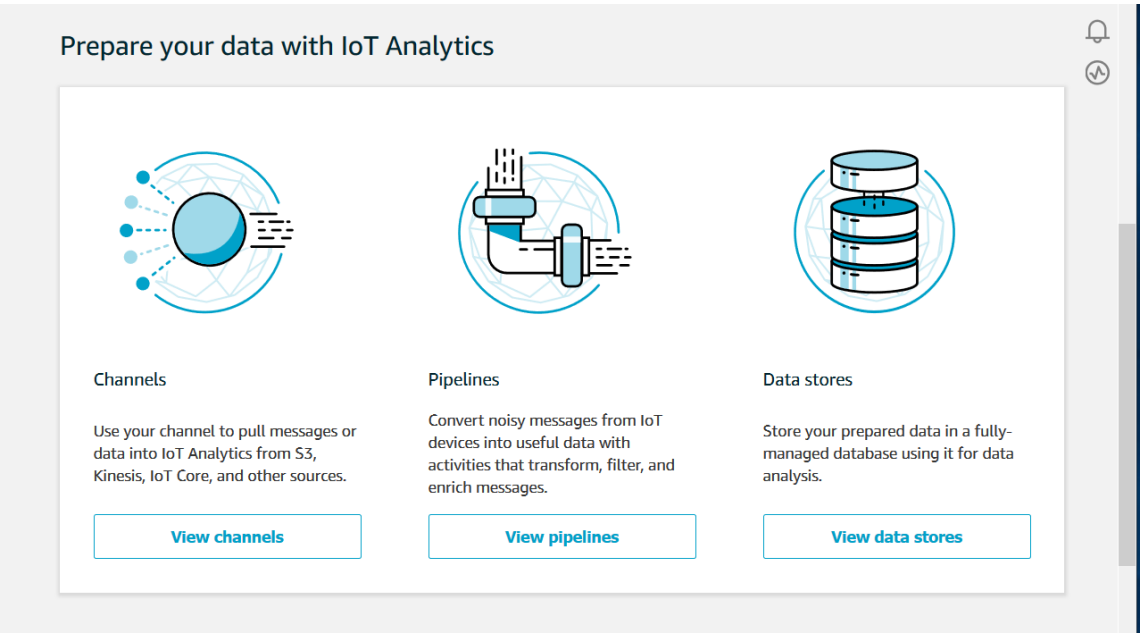
If you do not have an AWS account, create one.

1. To create an AWS account, open the AWS home page and choose **Create an AWS Account**.
2. Follow the online instructions. Part of the sign-up procedure involves receiving a phone call and entering a PIN using your phone's keypad.
3. Sign in to the AWS Management Console and open the AWS IoT Analytics console.

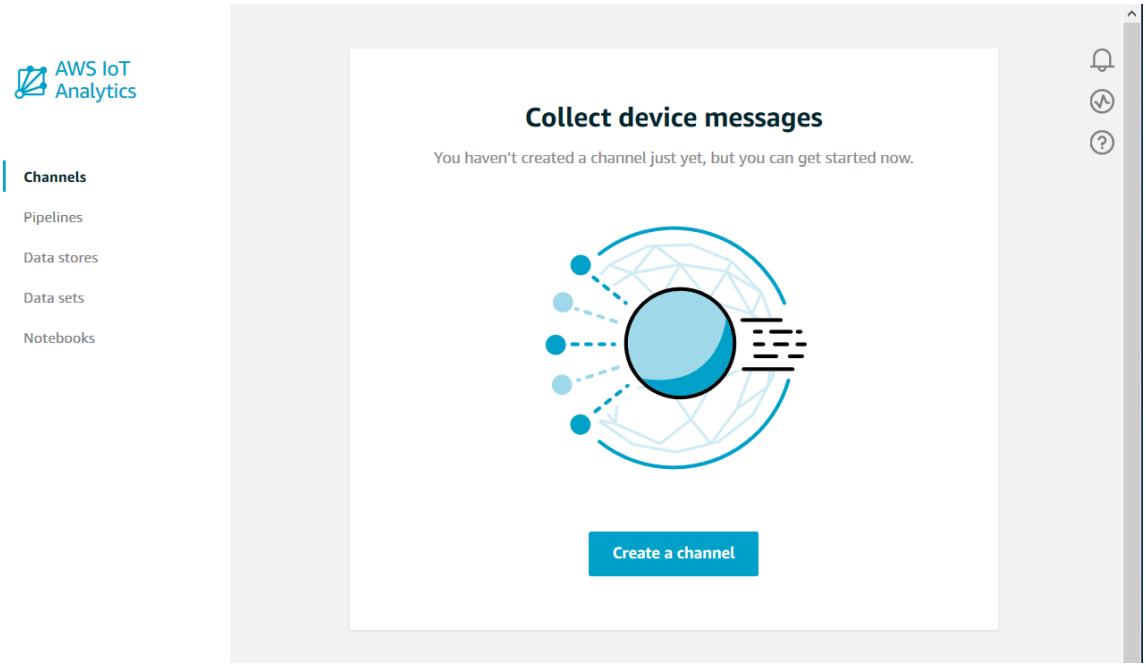
Create a Channel

Incoming messages are sent to a channel.

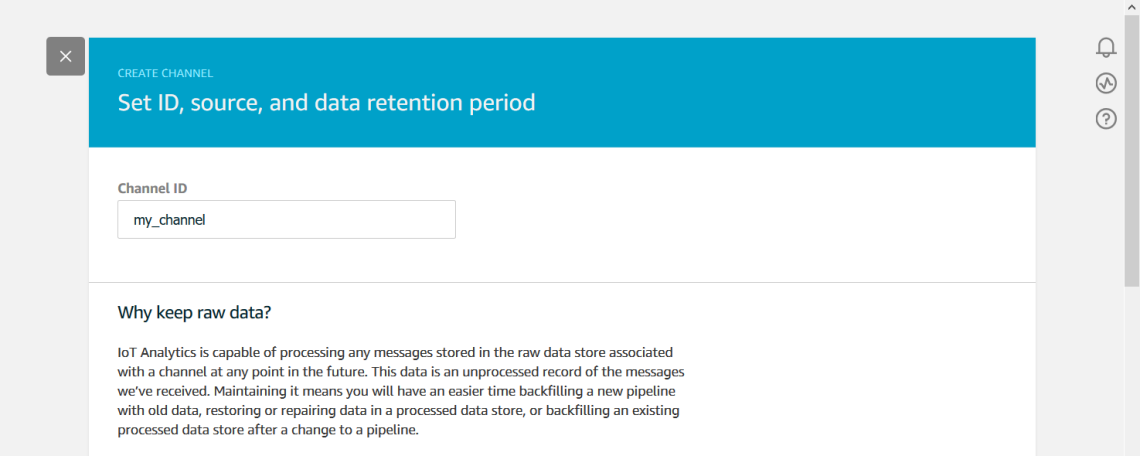
1. On the AWS IoT Analytics console landing page, in the **Prepare your data with IoT Analytics** section, under **Channels** choose **View channels**.



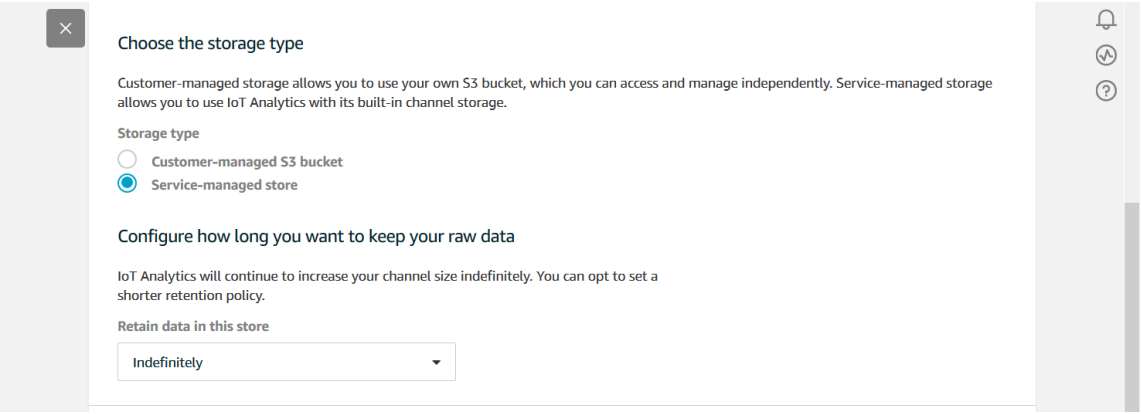
2. On the **Collect device messages** page, choose **Create a channel**.



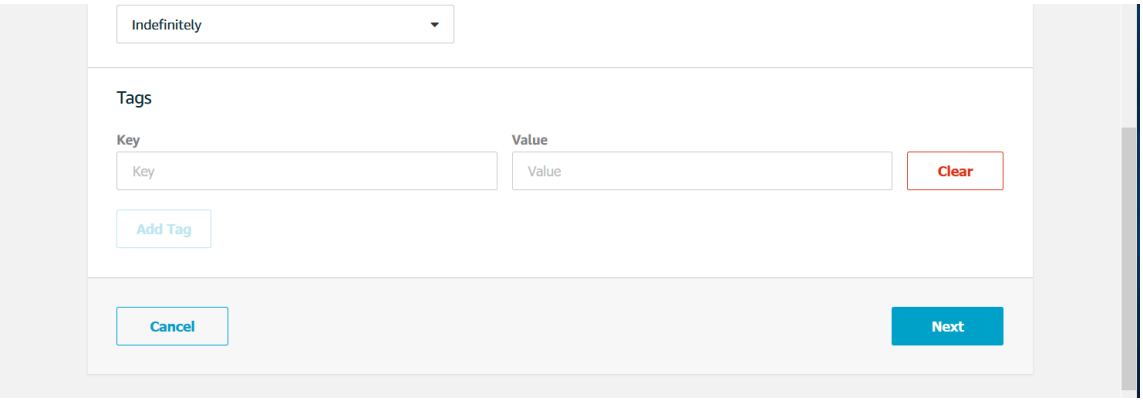
3. On the **Set ID, source, and data retention period** page, enter a channel ID, then scroll down.



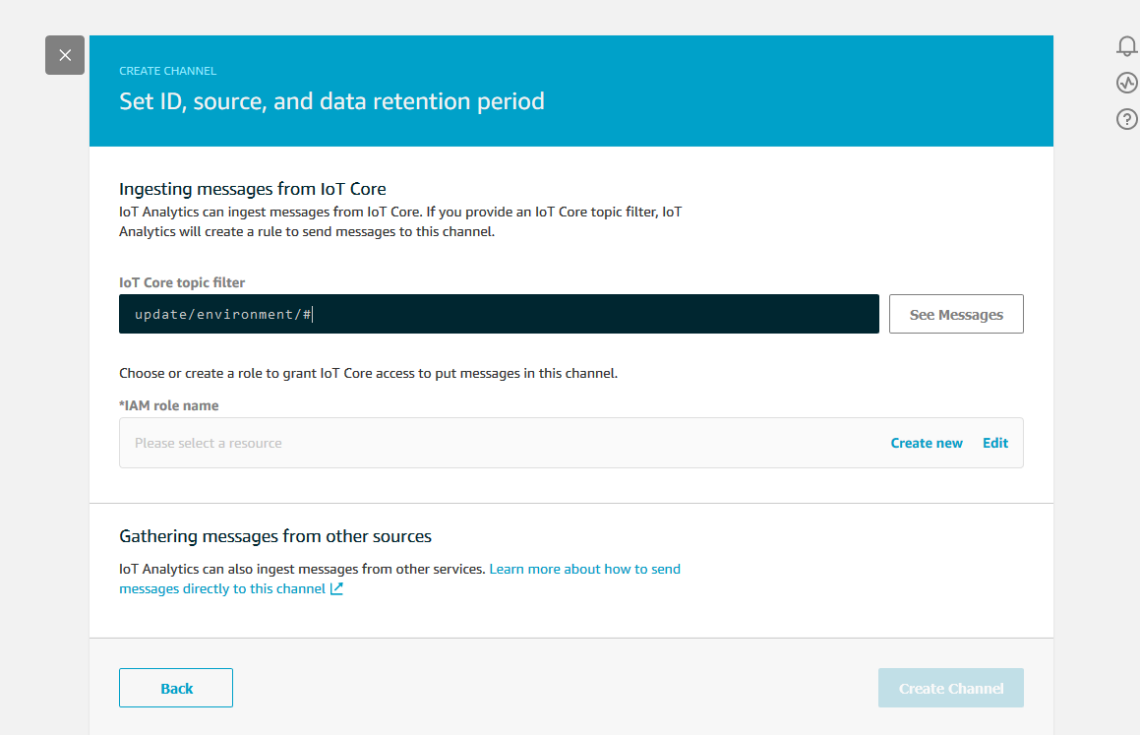
4. Under **Choose the storage type**, choose **Service-managed store**, then scroll down.



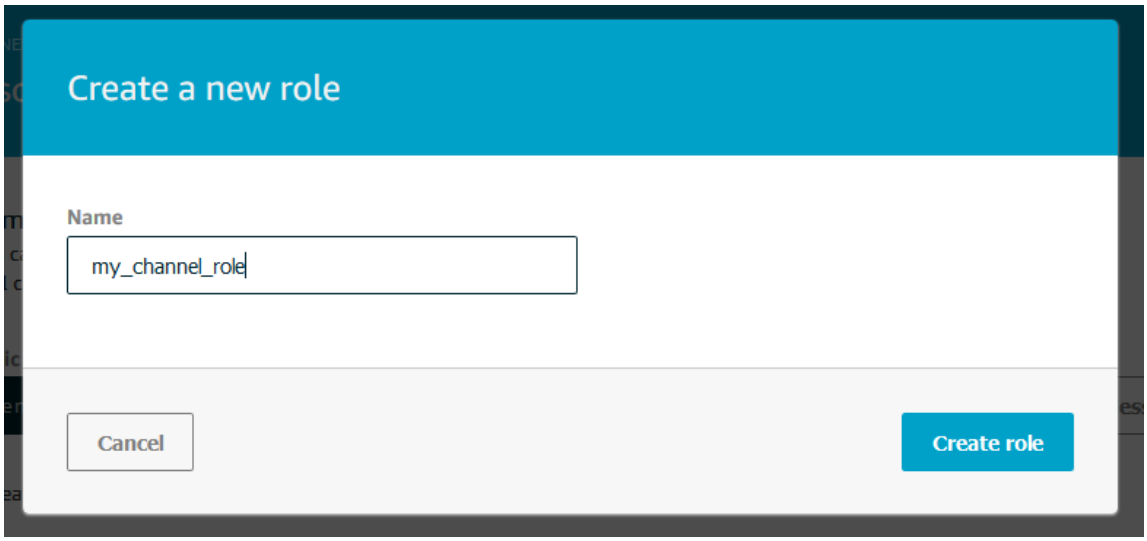
5. Choose **Next**.



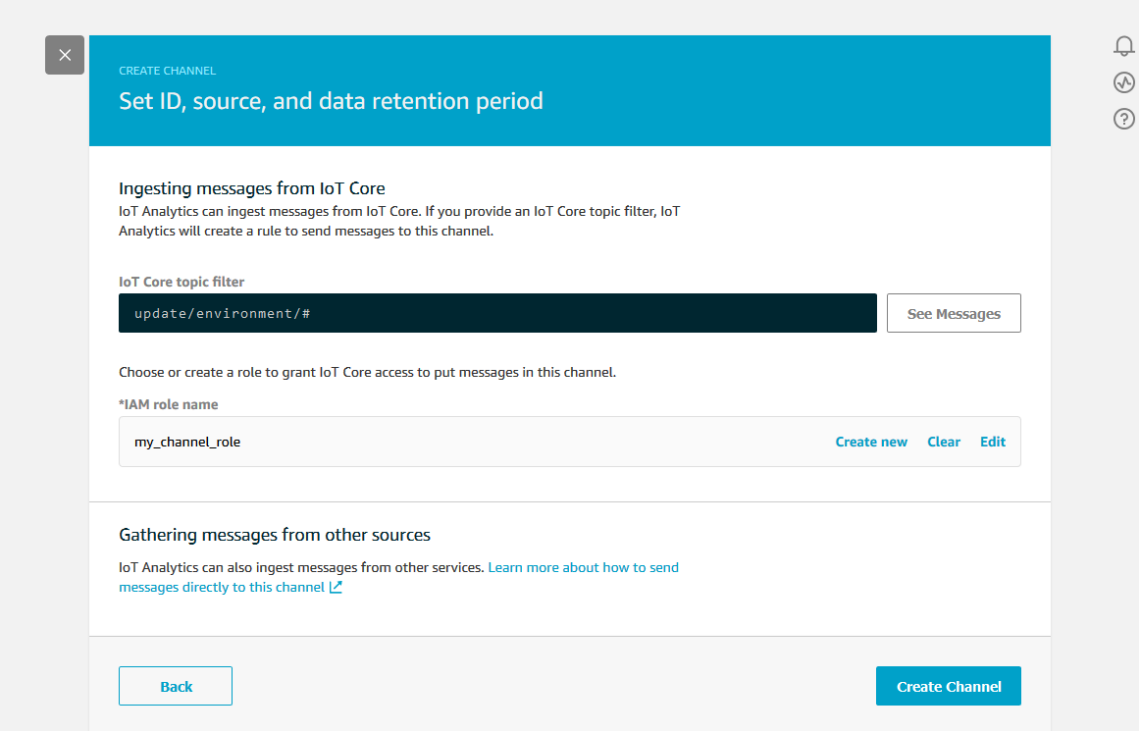
6. Enter an IoT Core (MQTT) topic filter. Make a note of the topic filter you entered here, because you need it in a later step in order to create a message that gets picked up by your channel. (For this example, we are using a topic filter with a wildcard: "`update/environment/#`".)



7. In the **IAM role name** area choose **Create new**. In the **Create a new role** window enter a **Name** for the role, then select **Create role**. This automatically creates a new role with an appropriate policy attached to it.



8. Choose **Create channel**.

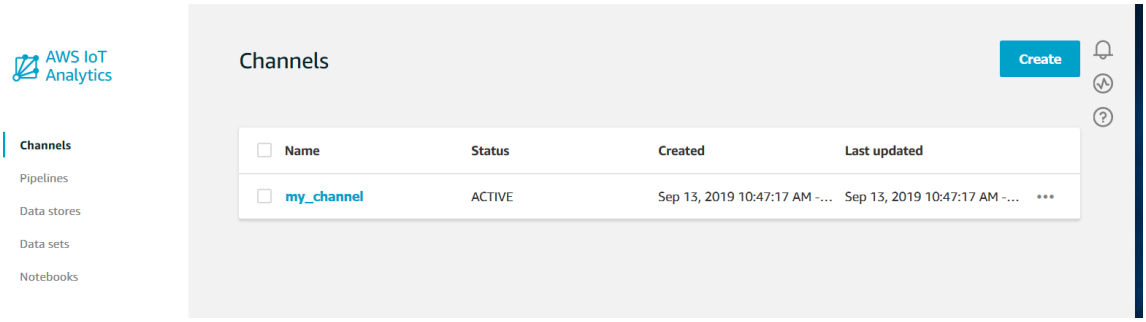


You have successfully created a channel.

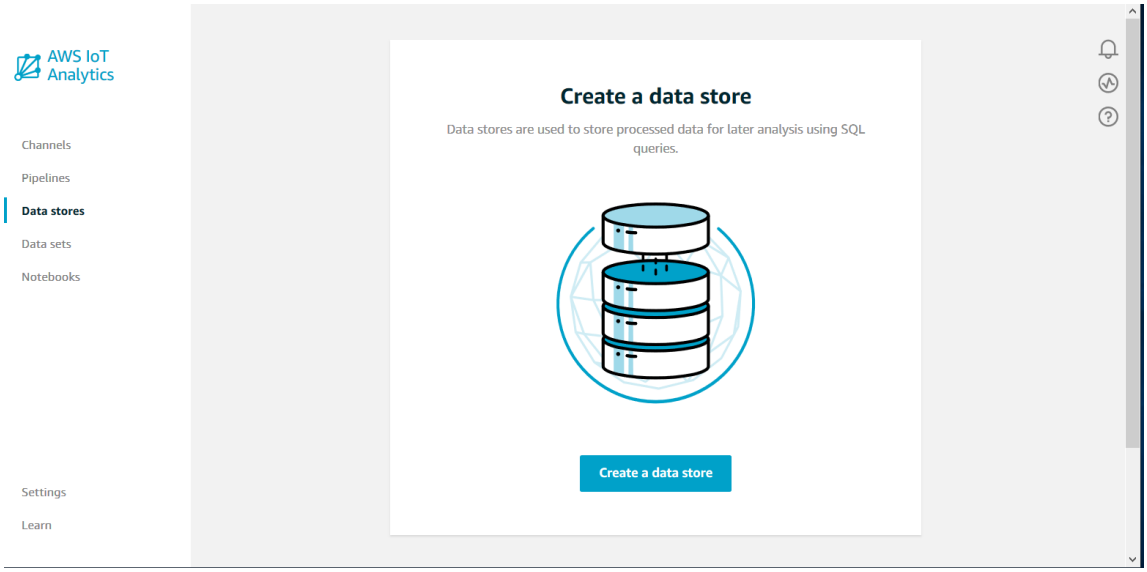
Create a Data store

A data store receives and stores your messages. You can create multiple data stores to store data according to your needs. For this example, you create a single data store to receive your AWS IoT messages:

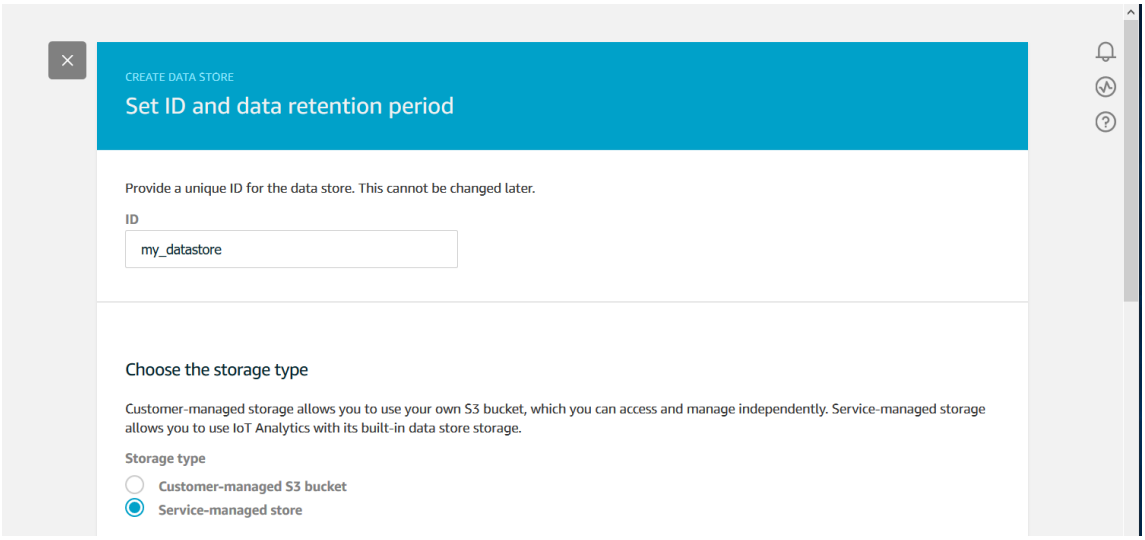
1. On the **Channels** page, in the left navigation pane, choose **Data stores**.

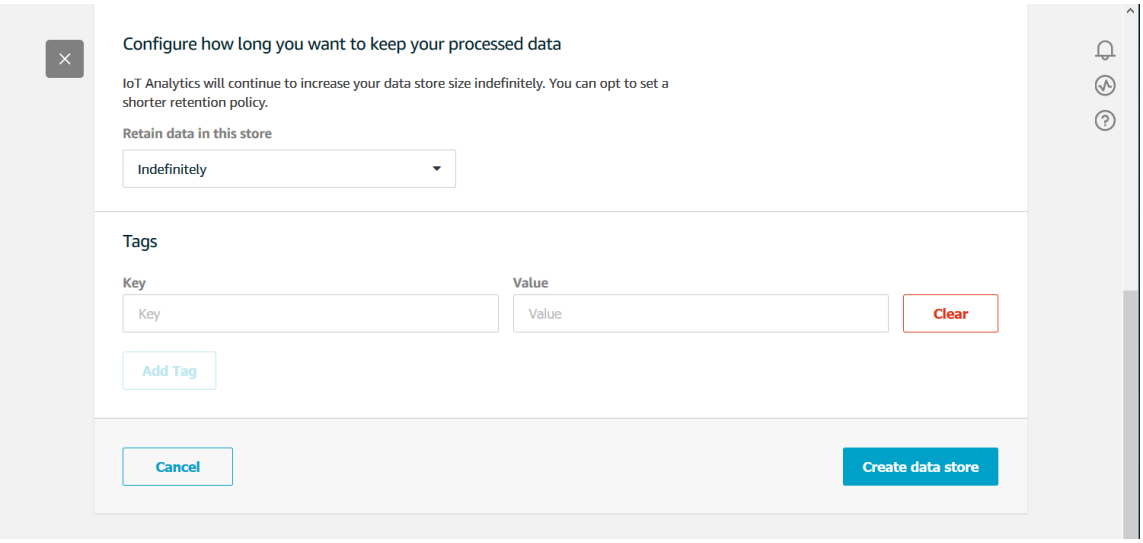


1. On the **Create a data store** page, choose **Create a data store**:



2. Enter an **ID** for your data store. Under **Choose the storage type**, choose **Service-managed store**, then scroll down and choose **Create data store**.





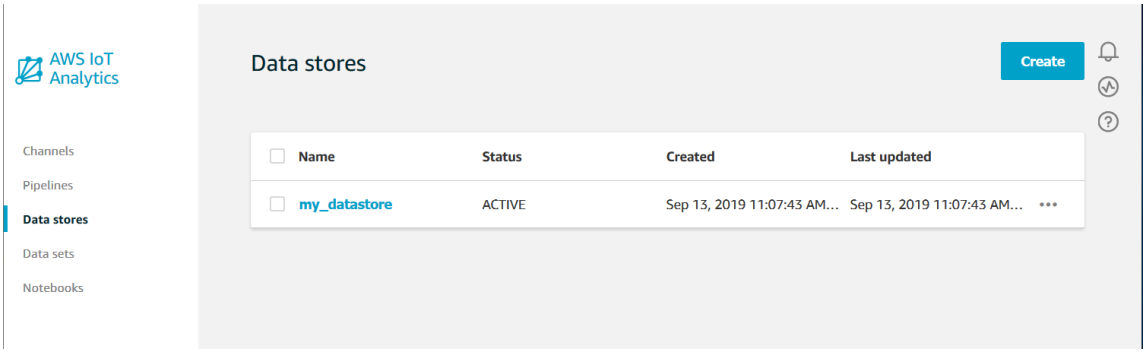
You have successfully created a data store.

Create a Pipeline

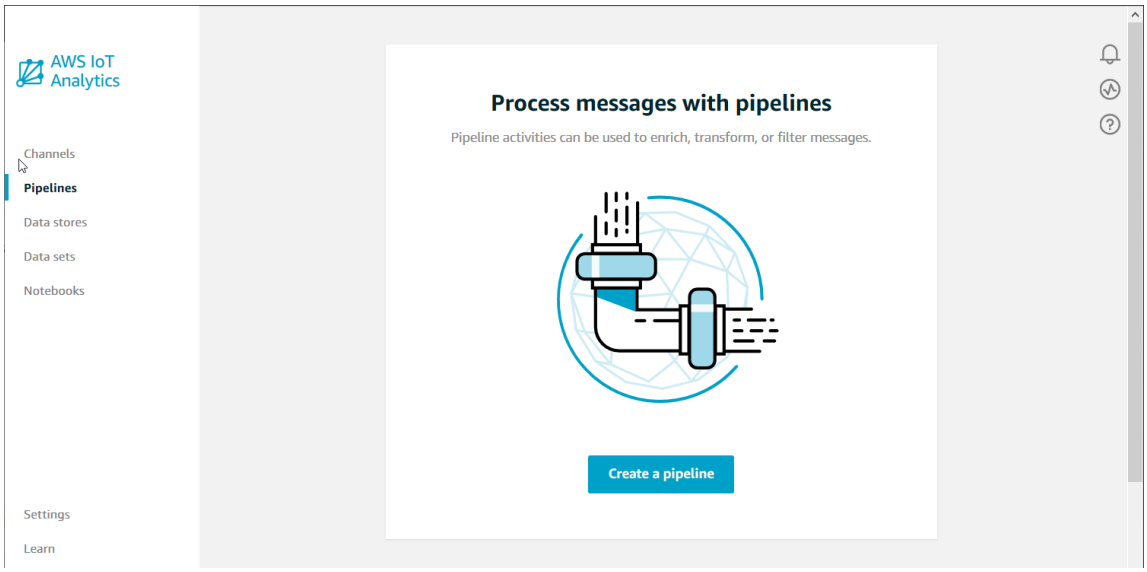
To connect a channel to a data store, you need to create a pipeline. The simplest possible pipeline contains no activities other than specifying the channel that collects the data and identifying the data store to which the messages are sent. We explore more complicated pipelines in Pipeline Activities (p. 44).

For this example, you create a pipeline that does nothing other than connect a channel to a data store. You can see how raw data flows to the data store. Later, you can introduce pipeline activities to process this data.

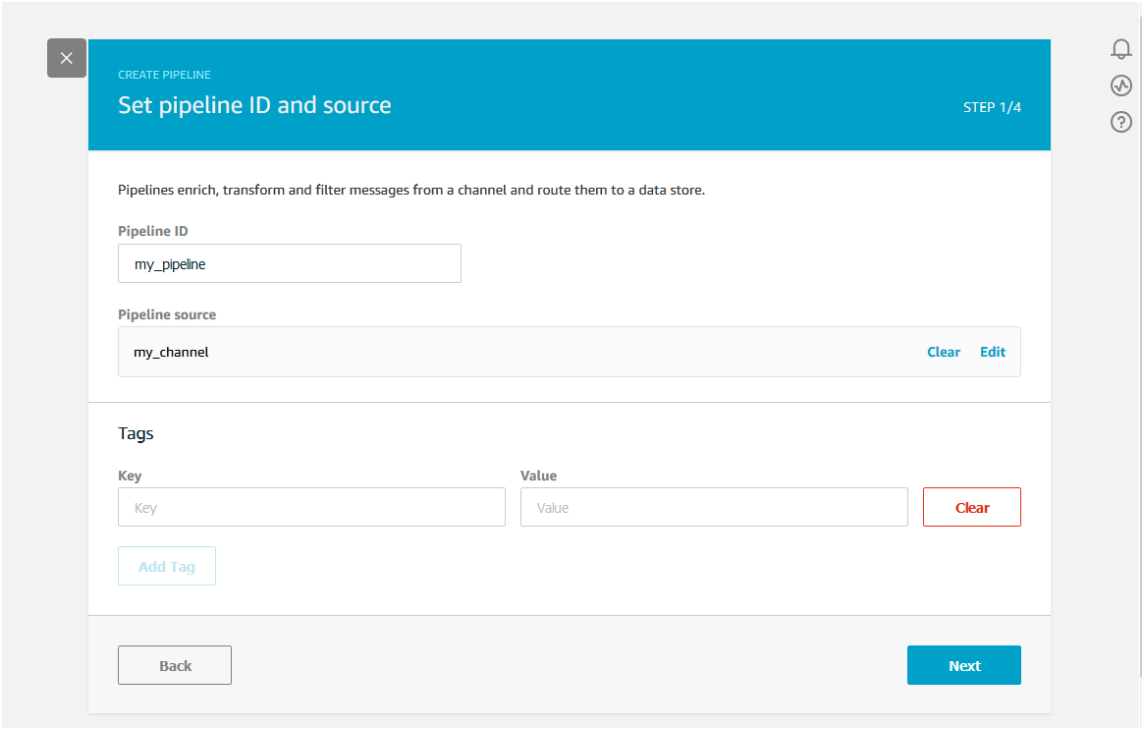
1. On the **Data stores** page, in the left navigation pane, choose **Pipelines**.



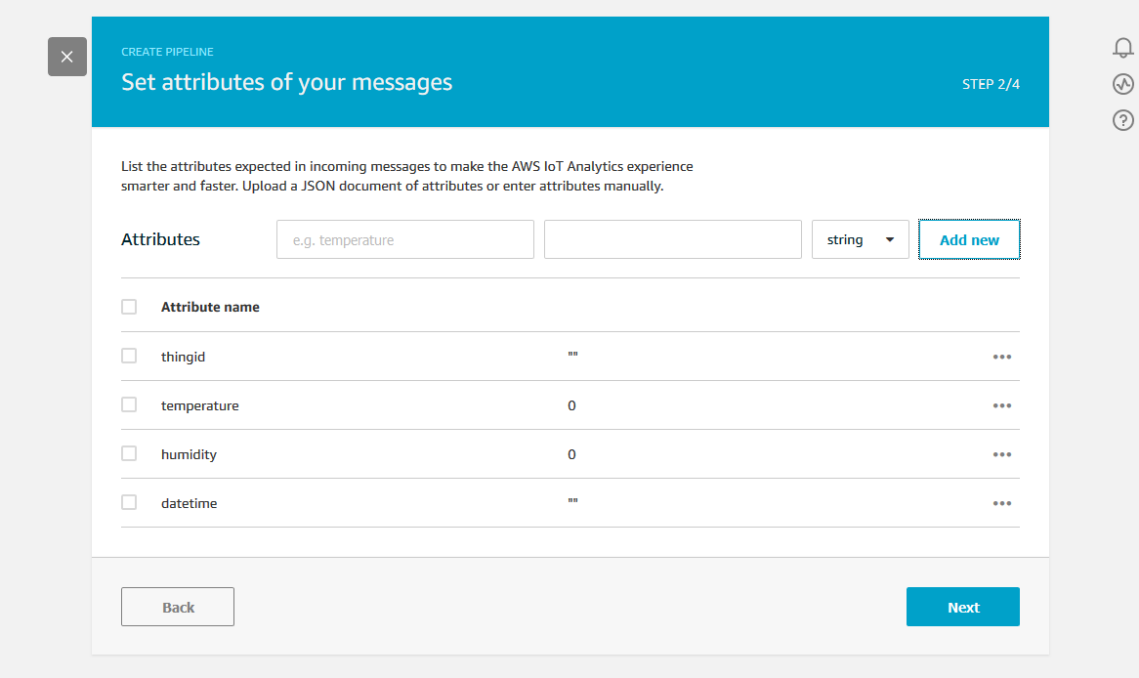
1. On the **Process messages with pipelines** page, choose **Create a pipeline**:



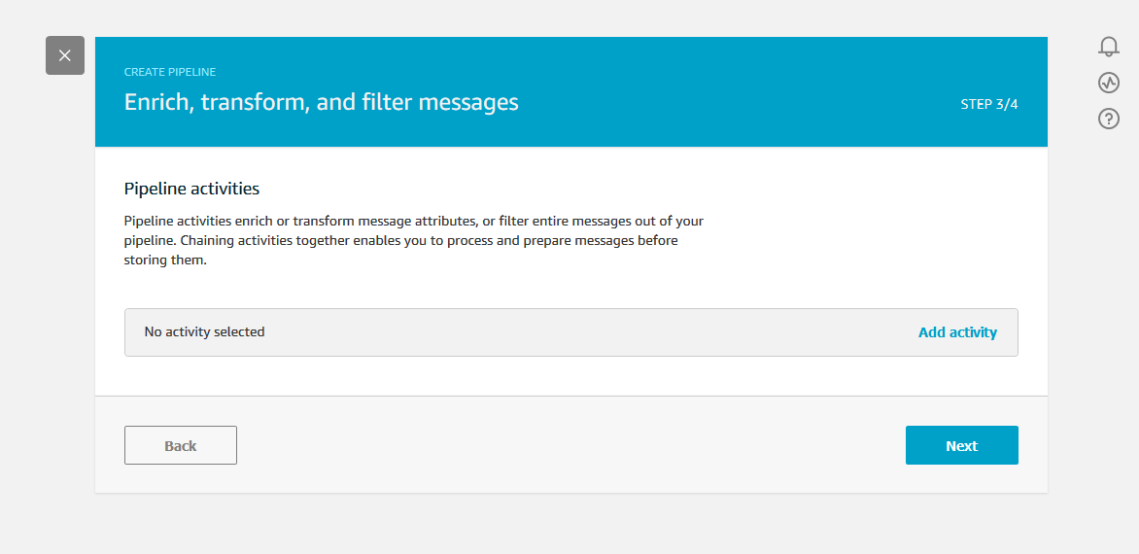
2. Enter a **Pipeline ID**. In **Pipeline source**, choose **Edit**, then select the channel you created before. Then choose **Next**:



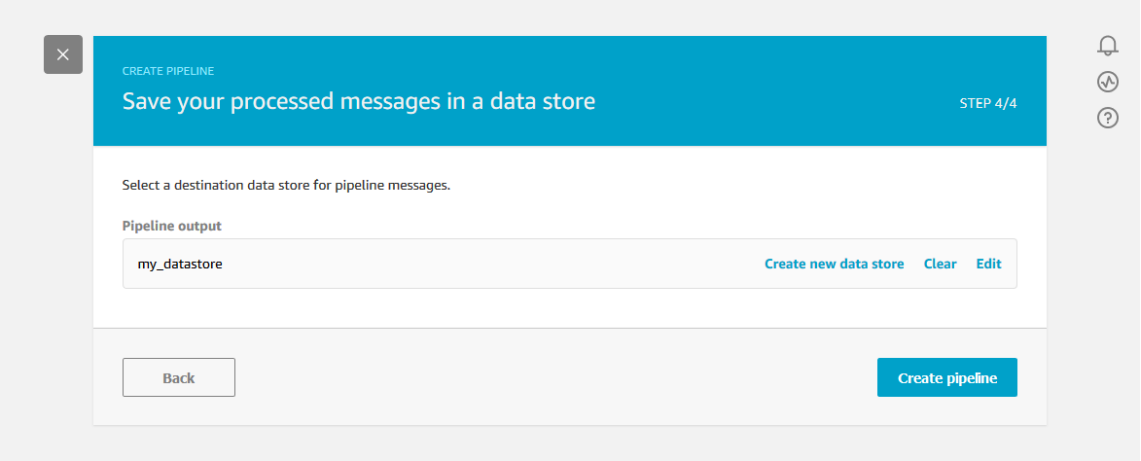
3. On the **Set attributes of your messages** page, enter an attribute name, select a type from the pull-down list, and enter an example value, then choose **Add new**. Repeat this for as many attributes as you want. When done, choose **Next**.



4. You won't be adding any pipeline activities right now, so on the **Enrich, transform, and filter messages** page, just choose **Next**.



5. On the **Save your processed messages in a data store** page, choose **Edit**, select the data store you created earlier, and then choose **Create pipeline**.

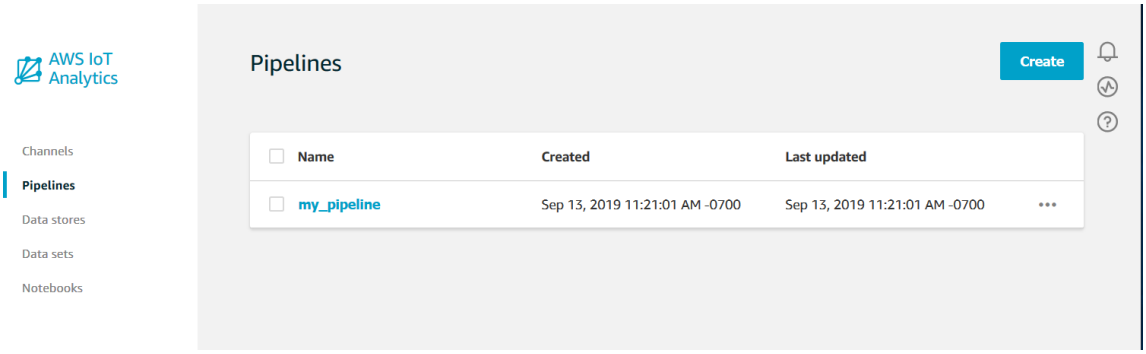


You have successfully created a pipeline.

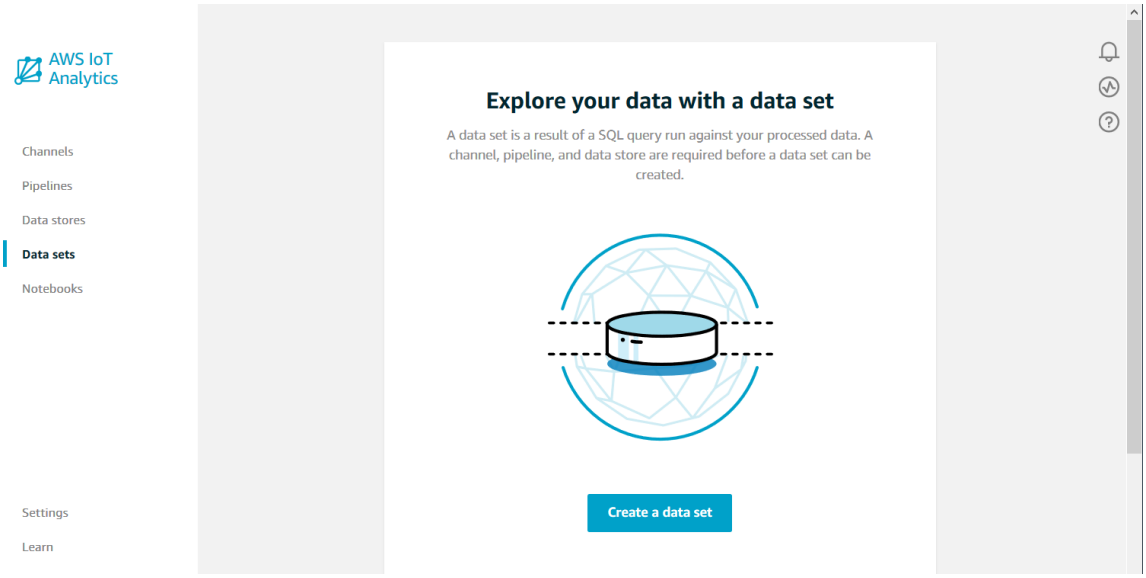
Create a Data Set

You now have a channel that routes data to a pipeline that stores data in a data store where it can be queried. To query the data, you create a data set. A data set contains SQL expressions that you use to query the data store along with an optional schedule that repeats the query at a day and time you choose. You can create the optional schedules by using expressions similar to Amazon CloudWatch schedule expressions.

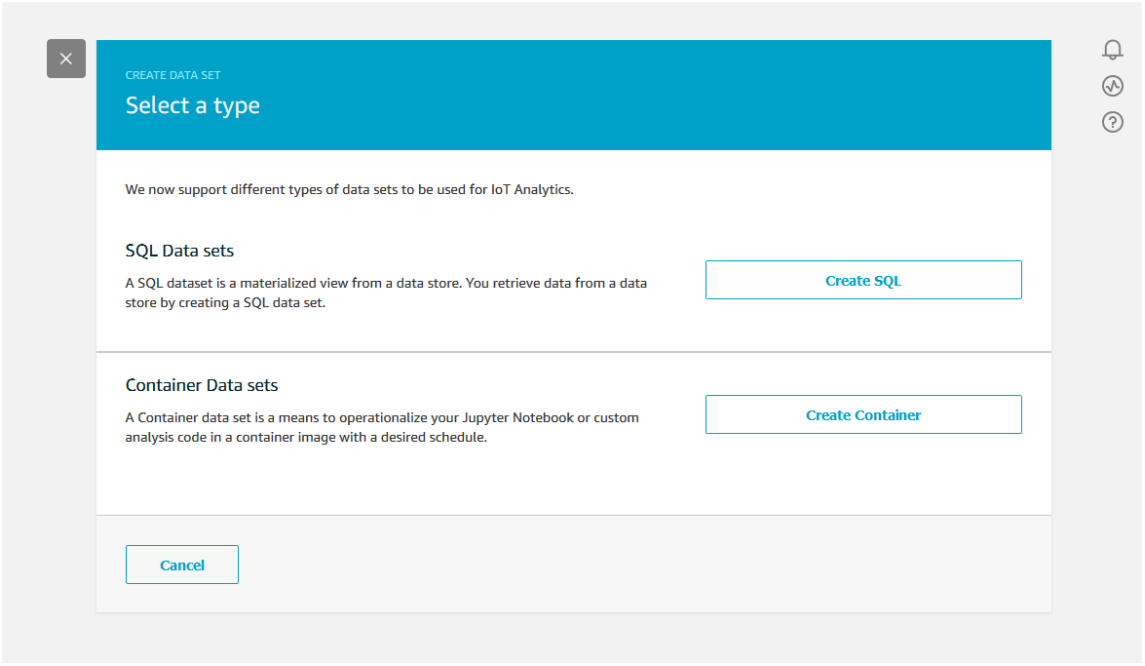
1. On the **Pipelines** page, in the left navigation pane, choose **Data sets**.



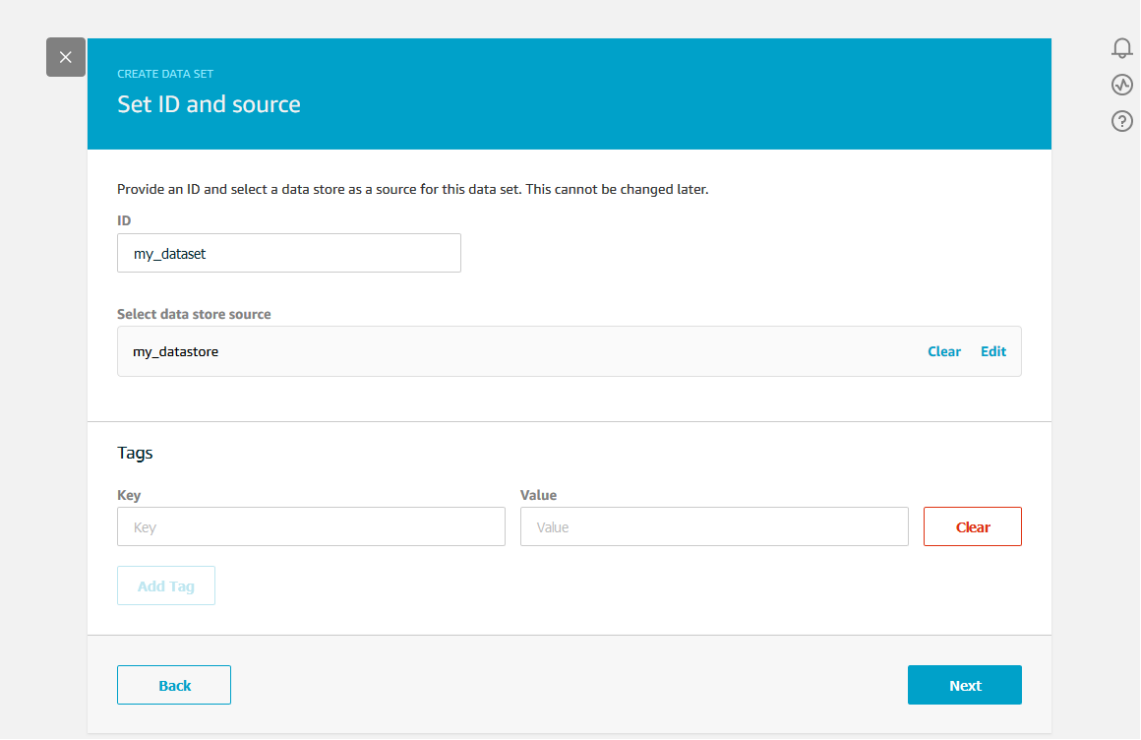
2. On the **Explore your data with a data set** page, choose **Create a data set**:



3. On the **Select a type** page, choose **Create SQL**.

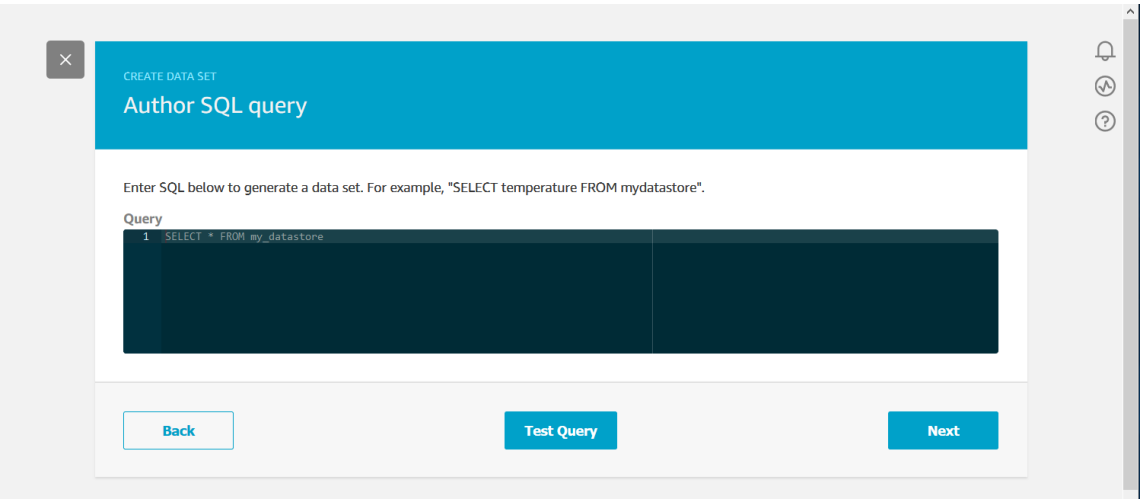


4. On the **Set ID and source** page, enter an **ID**. In **Select data store source** choose **Edit** and select the data store you created earlier. Then choose **Next**:

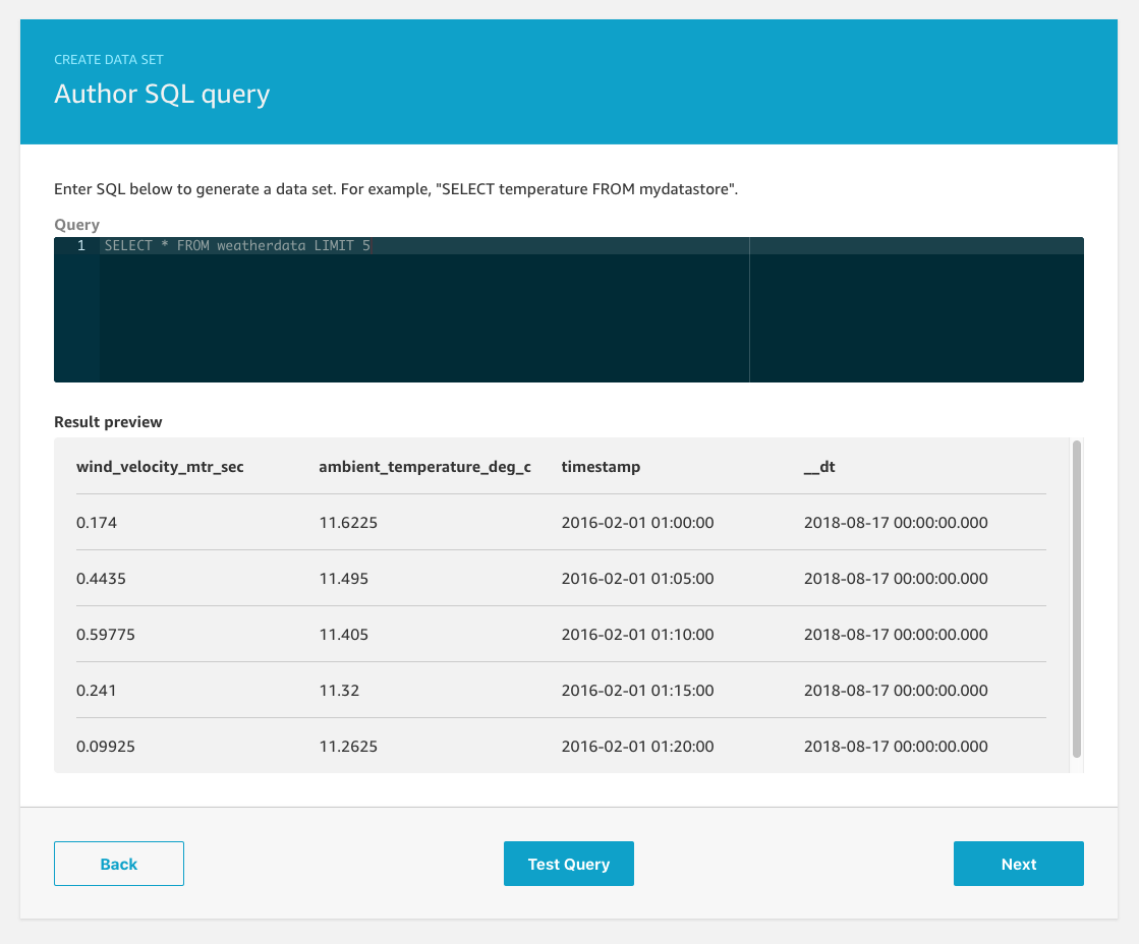


5. On the **Author SQL Query** page, in the **Query** area, enter a SQL expression that selects your attributes, or with a wildcard expression which selects all attributes, and then choose **Next**. In this example we are using a SQL expression with a wildcard:

```
SELECT * FROM my_datastore
```



The **Test query** button can be used to validate that the SQL **Query** you input is correct. It will execute the query in Amazon Athena and display the results in a window below the query. Here is an example of a successful test:

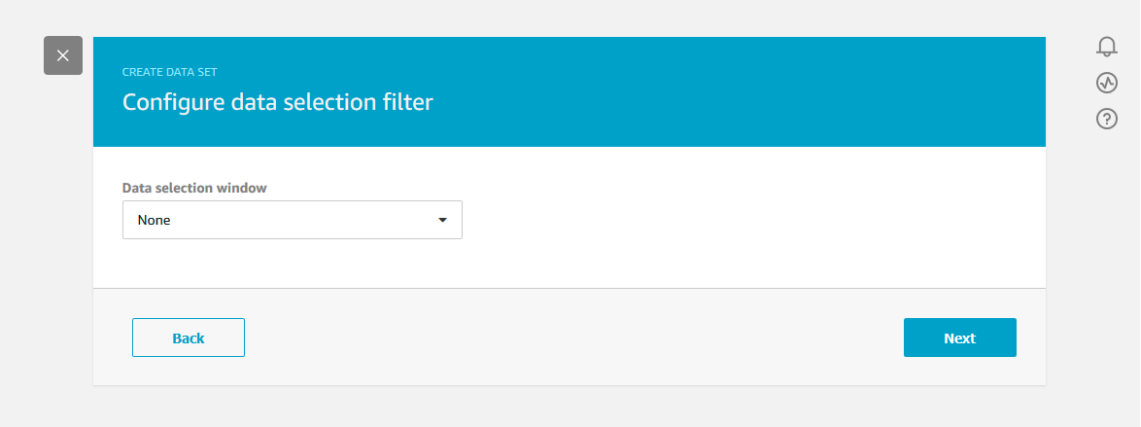


Note that executing a query at this point may return no, or very few, results depending on how much data is in your data store. (You may see only "__dt" at this point.) Amazon Athena also limits the maximum number of running queries. Because of this, you must be careful to limit the SQL query to a reasonable size so that it does not run for an extended period. We suggest using a "LIMIT" clause in the "SQL Query" during testing. For example:

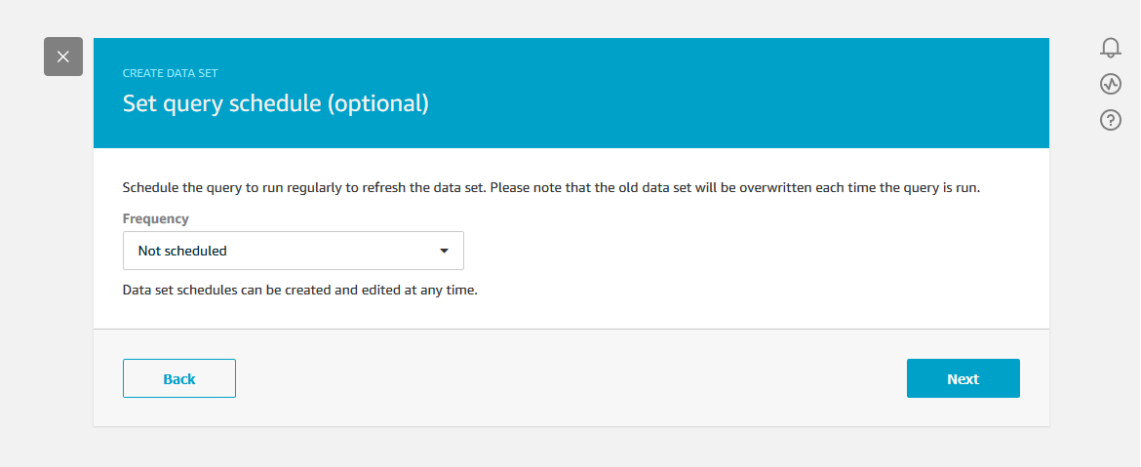
```
SELECT * FROM my_datastore LIMIT 5
```

After the test is successful, you can remove the "LIMIT 5".

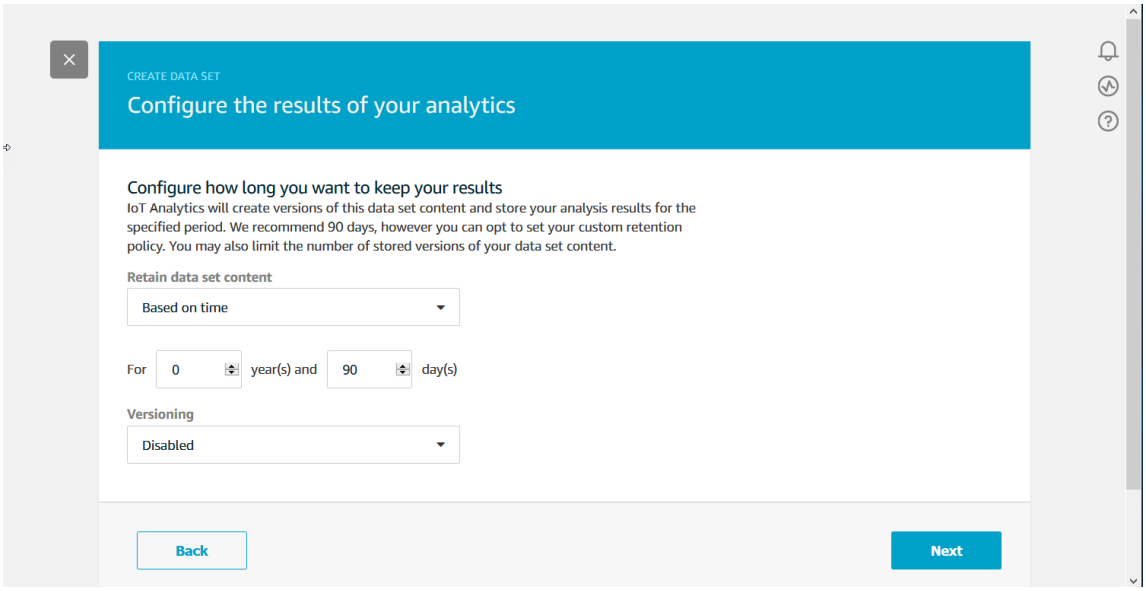
6. You won't configure a data selection filter at this point, so on the **Configure data selection filter** page just choose **Next**:



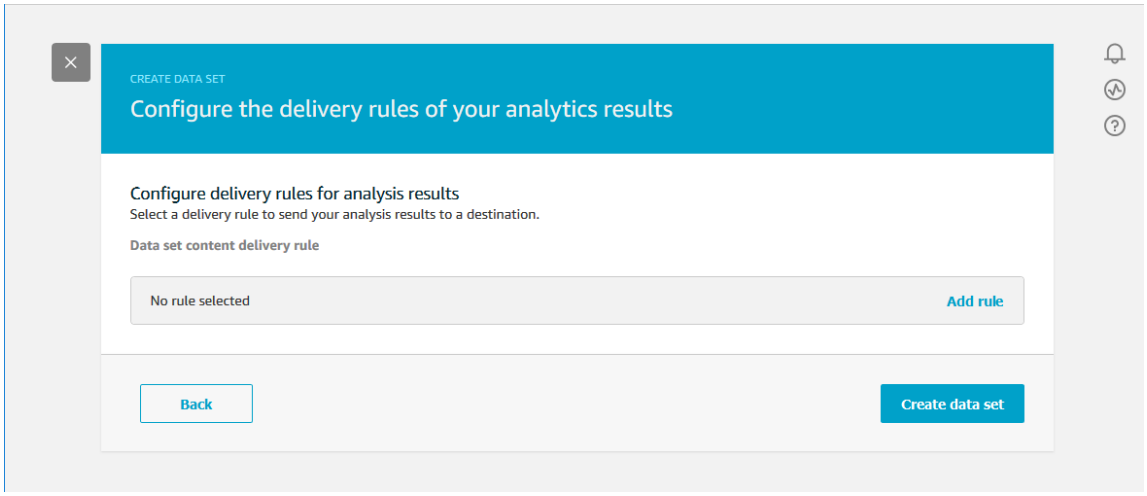
7. You won't schedule a recurring run of the query at this point, so on the **Set query schedule** page just choose **Next**:



8. You can use the default data set retention period (90 days) and leave **Versioning** "Disabled", so on the **Configure the results of your analytics** page just choose **Next**:



9. On the **Configure the delivery rules of your analytics results** page choose **Create data set**:



You have successfully created a data set.

Send an AWS IoT Message

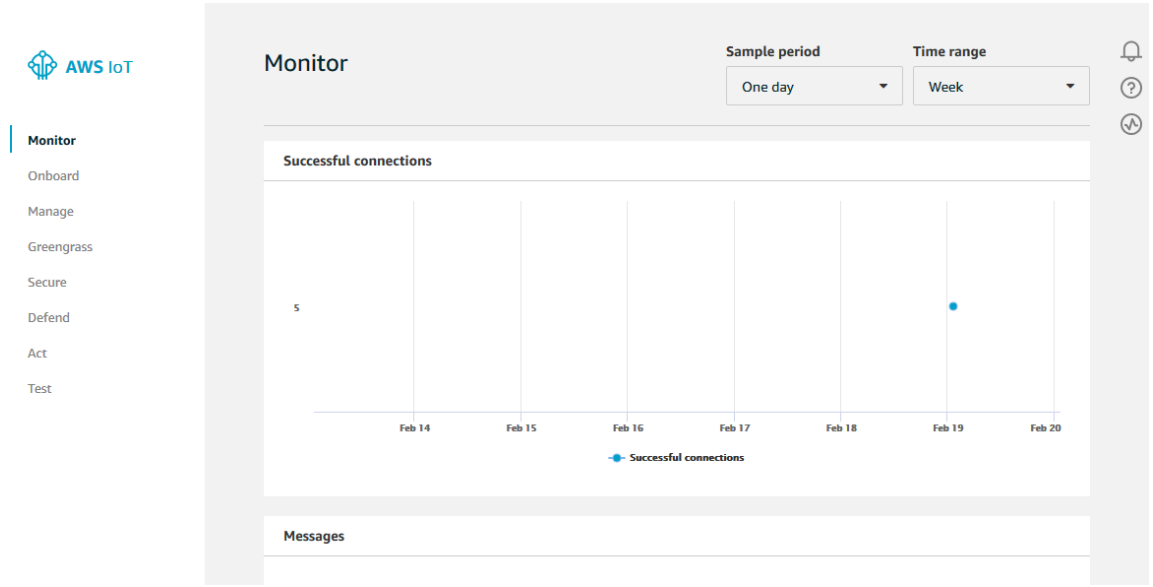
To generate some sample data, use the AWS IoT console to send an AWS IoT message.

Note
The field names of message payloads (data) that you send to AWS IoT Analytics:

- Must contain only alphanumeric characters and underscores (_); no other special characters are allowed.
- Must begin with an alphabetic character or single underscore (_).
- Cannot contain hyphens (-).
- In regular expression terms: `"^[A-Za-z_]([A-Za-z0-9]*|[A-Za-z0-9][A-Za-z0-9_]*)$"`.
- Cannot be greater than 255 characters.
- Are case-insensitive. (Fields named “foo” and “FOO” in the same payload are considered duplicates.)

For example, `{"temp_01": 29}` or `{"_temp_01": 29}` are valid, but `{"temp-01": 29}`, `{"01_temp": 29}` or `{"__temp_01": 29}` are invalid in message payloads.

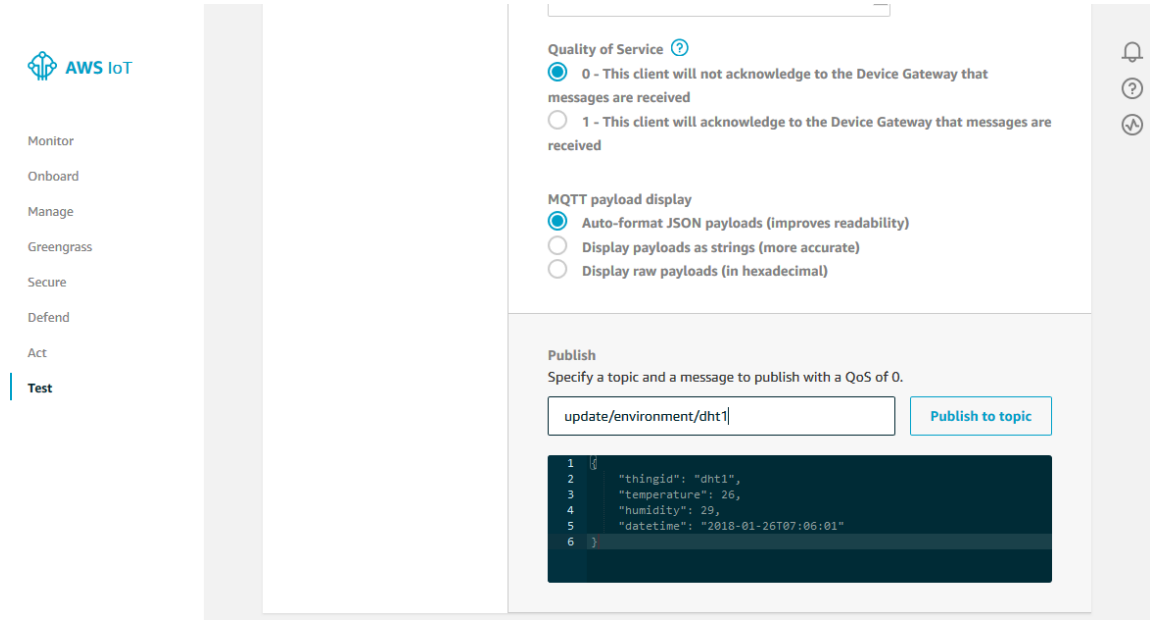
1. In the AWS IoT console, in the left navigation pane, choose **Test**.



2. On the **MQTT client** page, in the **Publish** section, in **Specify a topic**, type a topic that will match the topic filter you entered when you created a channel. (For this example, we use `update/environment/dht1`). In the message payload section, enter the following JSON contents:

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

3. Choose **Publish to topic**.

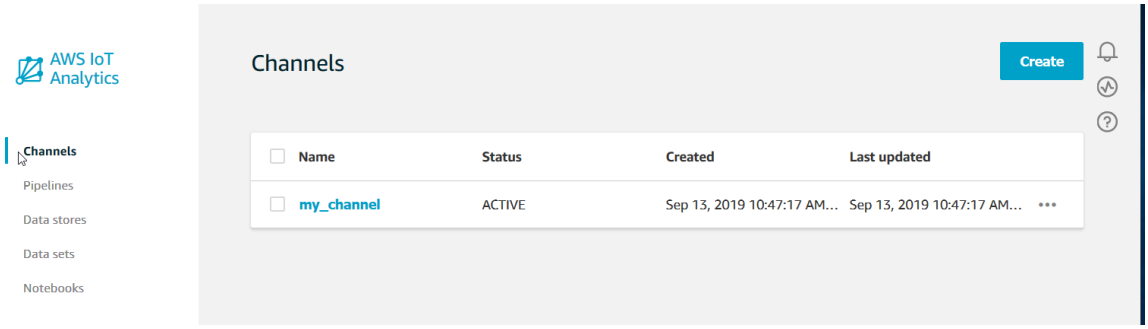


If you have followed the example to this point, then this publishes a message that is captured by your channel, and then routed by your pipeline to your data store.

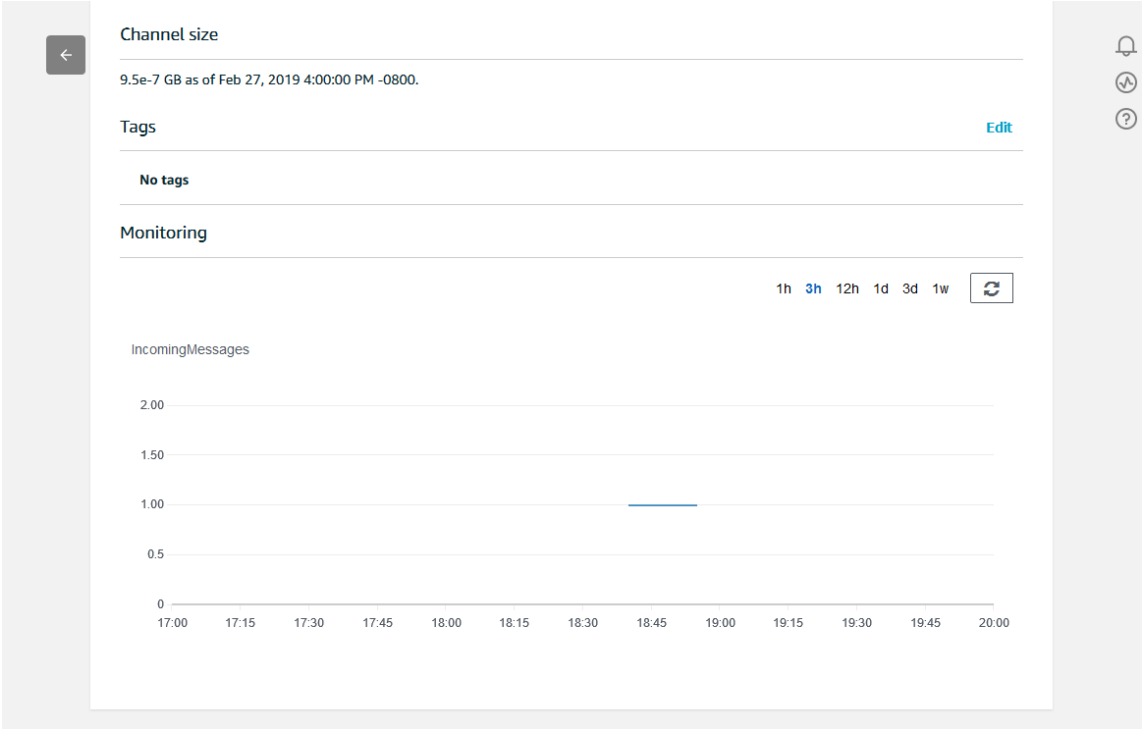
Check the Progress of IoT Messages

You can check that messages are being ingested into your channel by following these steps:

1. In the AWS IoT Analytics console, in the left navigation pane, choose **Channels**, then choose the name of the channel you created earlier:

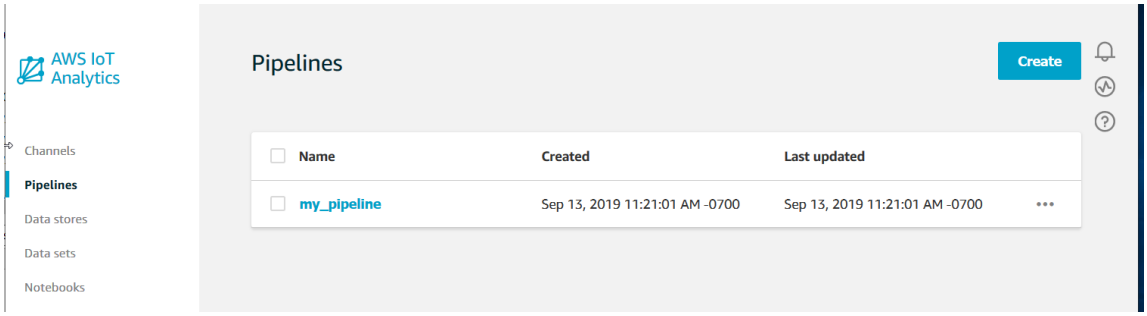


2. On the channel detail page, scroll down to the **Monitoring** section. Adjust the displayed time frame as necessary by choosing one of the time frame indicators (**1h 3h 12h 1d 3d 1w**). You should see a graph line indicating the number of messages ingested into this channel during the specified time frame:

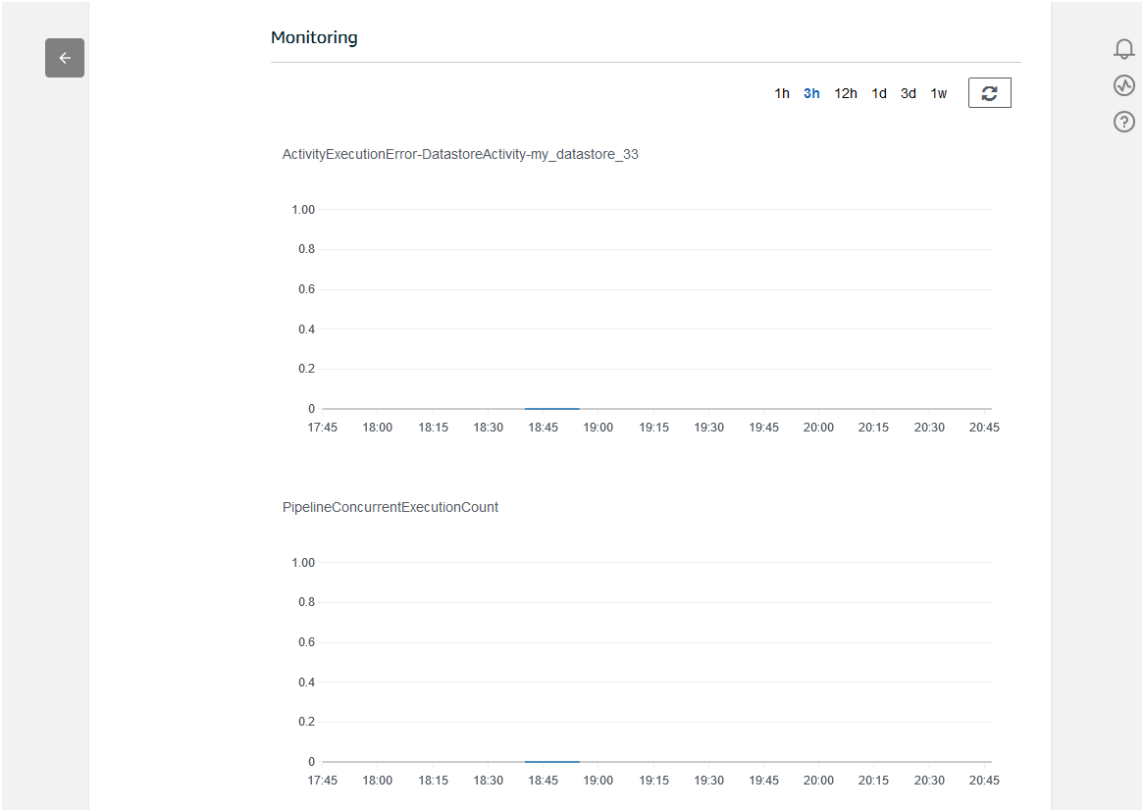


A similar monitoring capability exists for checking pipeline activity executions. You can monitor activity execution errors on the pipeline's detail page. (We haven't specified activities as part of our pipeline, so 0 execution errors should be displayed.)

1. In the AWS IoT Analytics console, in the left navigation pane, choose **Pipelines**, then choose the name of a pipeline you created earlier:



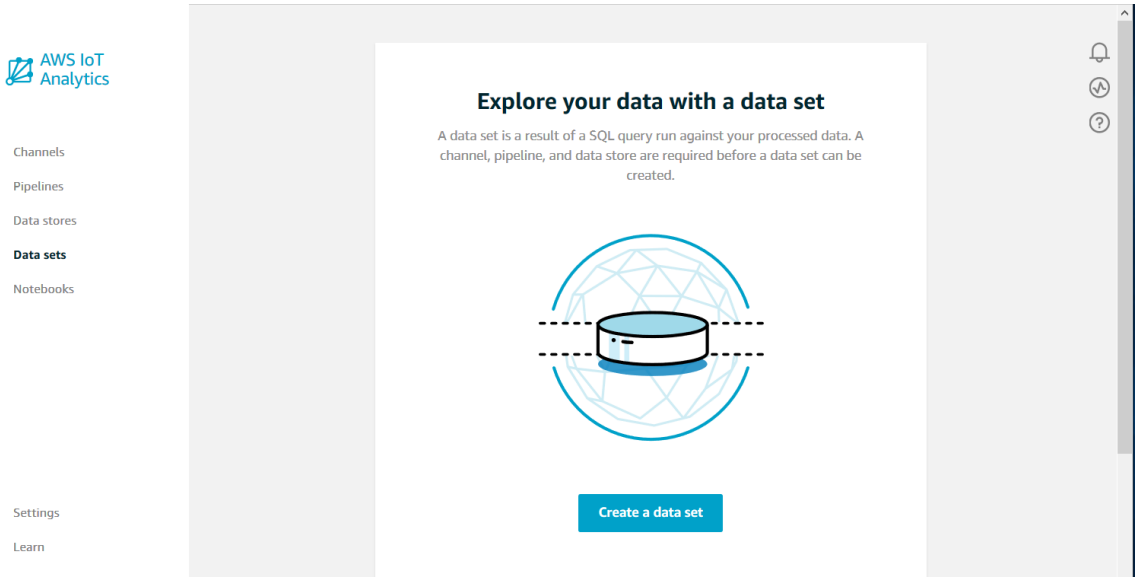
2. On the pipeline detail page, scroll down to the **Monitoring** section. Adjust the displayed time frame as necessary by choosing one of the time frame indicators (**1h 3h 12h 1d 3d 1w**). You should see a graph line indicating the number of pipeline activity execution errors during the specified time frame:



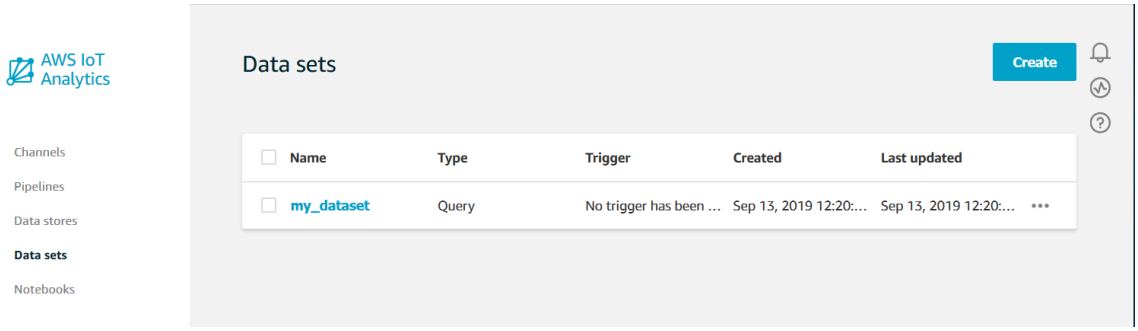
Access the Query Results

The data set content is the result of your query in a file, in CSV format.

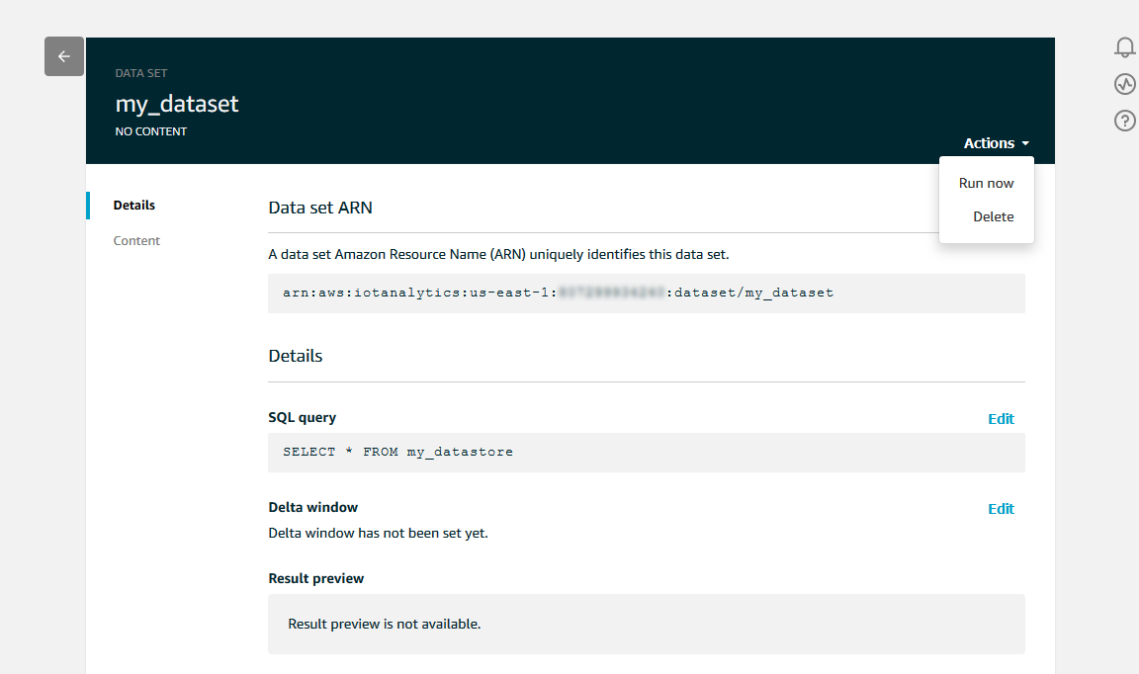
1. In the AWS IoT Analytics console, in the left navigation pane, choose **Data sets**.



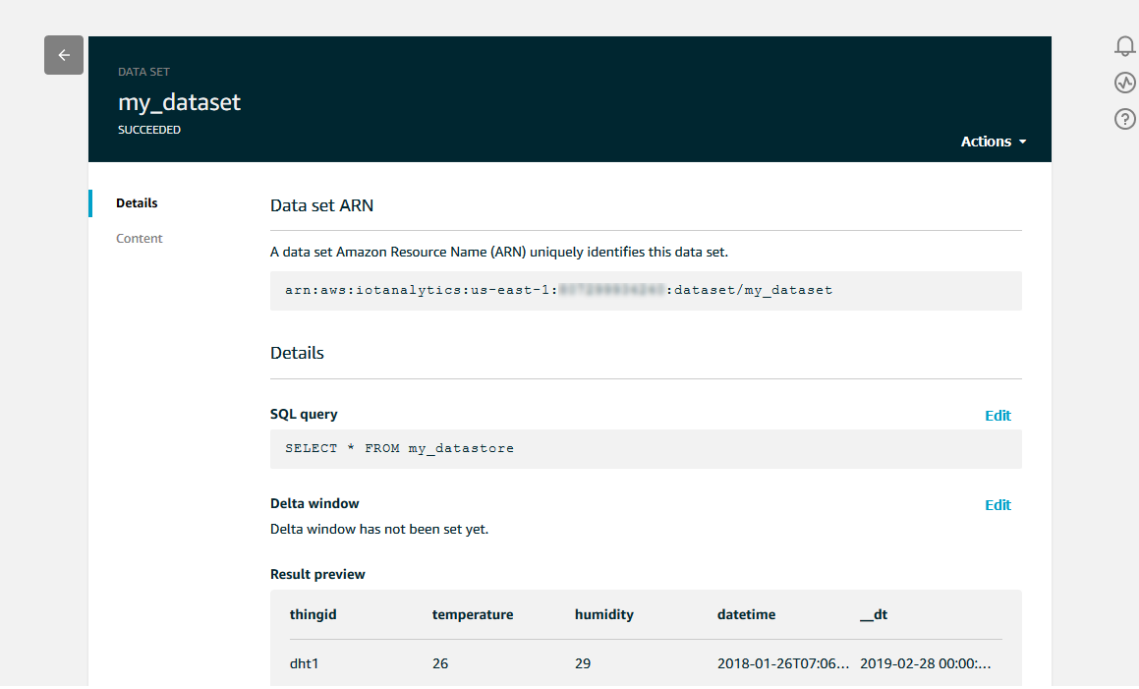
1. On the **Data sets** page, choose the name of the data set you created previously:



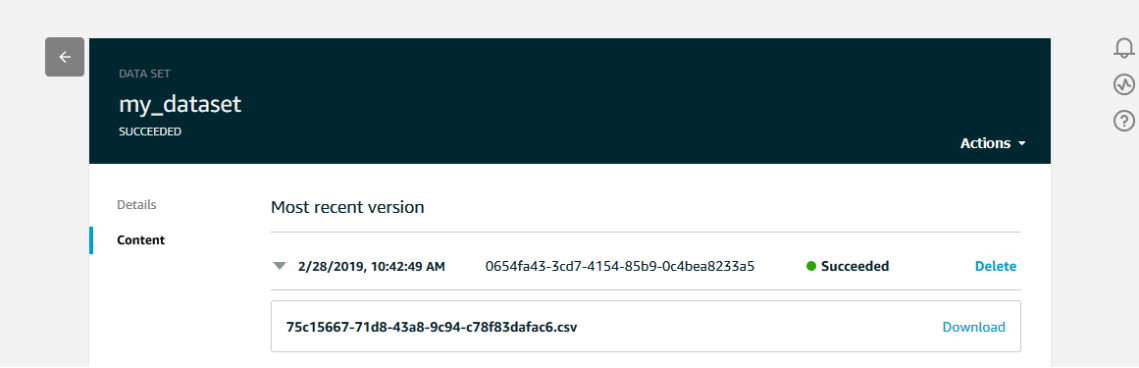
2. On the data set information page, in the upper-right corner, choose **Actions**, and then choose **Run now**:



3. To check if the data set is ready, look for **SUCCEEDED** under the name of the data set in the upper left-hand corner. The **Details** section contains the query results:



4. In the left navigation pane, click **Content:**. Choose **Download** (to the right of the name of the most recent version) to view or save the CSV file that contains the query results.



It should look similar to this:

```
"thingid","temperature","humidity","datetime","__dt"
"dht1","26","29","2018-01-26T07:06:01","2019-02-27 00:00:00.000"
```

AWS IoT Analytics can also embed the HTML portion of a Jupyter notebook on this **Data Set** content page. For more information see Visualizing AWS IoT Analytics Data with the Console (p. 103).

5. Choose the left arrow in the upper-left corner to return to the main page of the **AWS IoT Analytics console**.

Explore Your Data

You have several options for storing, analyzing and visualizing your data…

Amazon S3

You can send data set contents to an Amazon Simple Storage Service (S3) bucket, enabling integration with your existing data lakes or access from in-house applications and visualization tools. See the field `contentDeliveryRules::destination::s3DestinationConfiguration` in CreateDataset (p. 141).

AWS IoT Events

You can send data set contents as an input to AWS IoT Events, a service which enables you to monitor devices or processes for failures or changes in operation, and to trigger additional actions when such events occur.

To do this, create a data set using CreateDataset (p. 141) and specify an AWS IoT Events input in the field `contentDeliveryRules :: destination :: iotEventsDestinationConfiguration :: inputName`. You must also specify the `roleArn` of a role which grants AWS IoT Analytics permission to execute "iotevents:BatchPutMessage". Whenever the data set's contents are created, AWS IoT Analytics will send each data set content entry as a message to the specified AWS IoT Events input. For example, if your data set contains:

```
"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"
...
```

then AWS IoT Analytics will send messages containing fields like this:

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

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

and you will want to create an AWS IoT Events input that recognizes the fields you are interested in (one or more of `what`, `who`, `dt`) and to create an AWS IoT Events detector model that uses these input fields in events to trigger actions or set internal variables.

Jupyter Notebooks

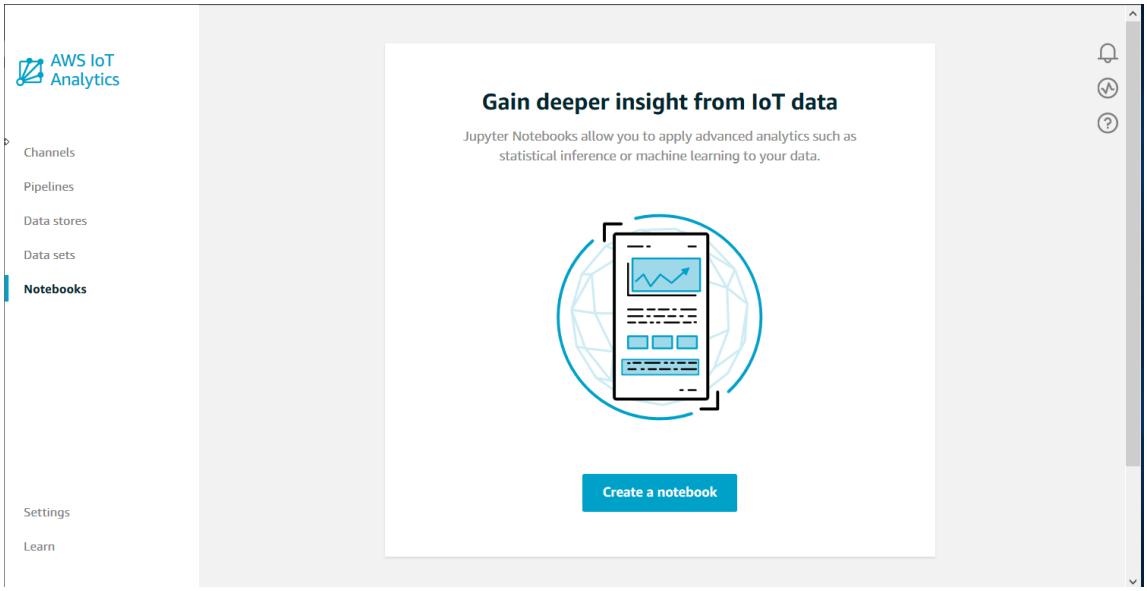
Jupyter Notebooks is an open source solution for advanced analyses and ad-hoc data exploration. Notebooks enable you to use templates and a scripting language, typically Python, to apply different transformations or normalizations to the data, aggregate metrics, and analyze and visualize data using data science libraries. You can even apply more complex analytics and machine learning, such as k-means clustering, to your data using these notebooks.

AWS IoT Analytics uses Amazon SageMaker notebook instances to host its Jupyter notebooks. Before you create a notebook instance, you must create a relationship between AWS IoT Analytics and Amazon SageMaker:

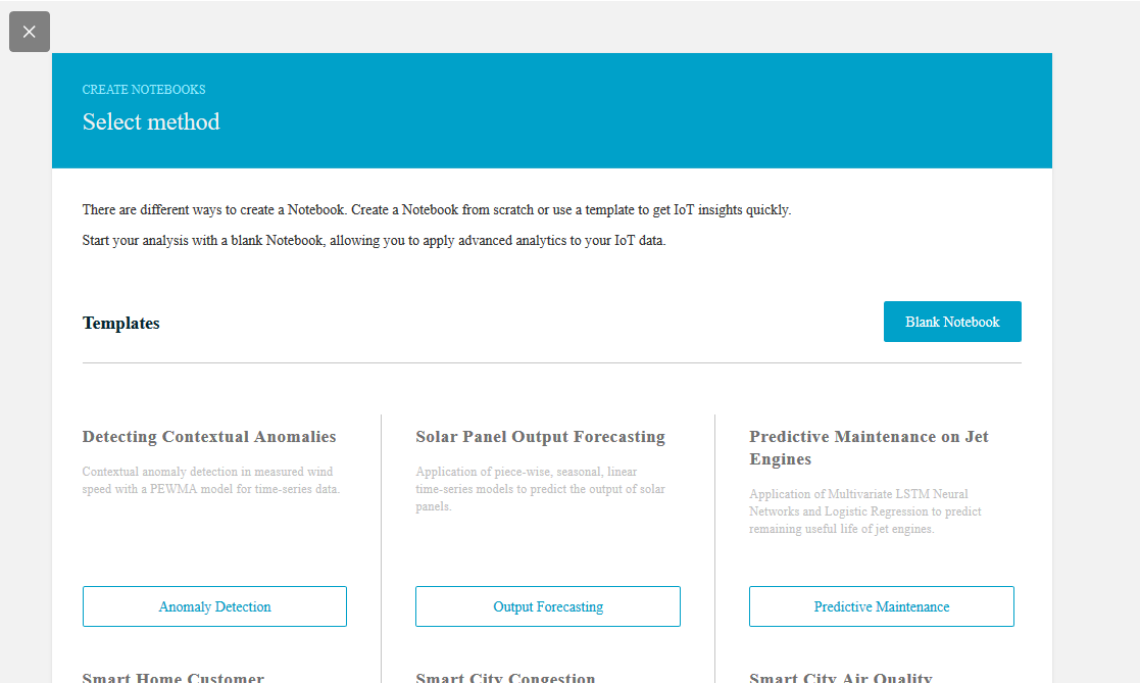
1. Go to the Amazon SageMaker console and create a notebook instance:
   a. Fill in the details, and then choose **Create a new role**. Make a note the role ARN.
   b. Create a notebook instance.

   This creates a role that Amazon SageMaker can use and a new notebook instance
2. Go to the IAM console and modify the Amazon SageMaker role:
   a. Open the role. It should have one managed policy.
   b. Choose **add inline policy**, and then for **Service**, choose IotAnalytics. Choose **Select actions**, and then type “GetDatasetContent” in the search box and select it. Choose **Review policy**.
   c. Review the policy for accuracy, give it a name, and then choose **Create policy**.

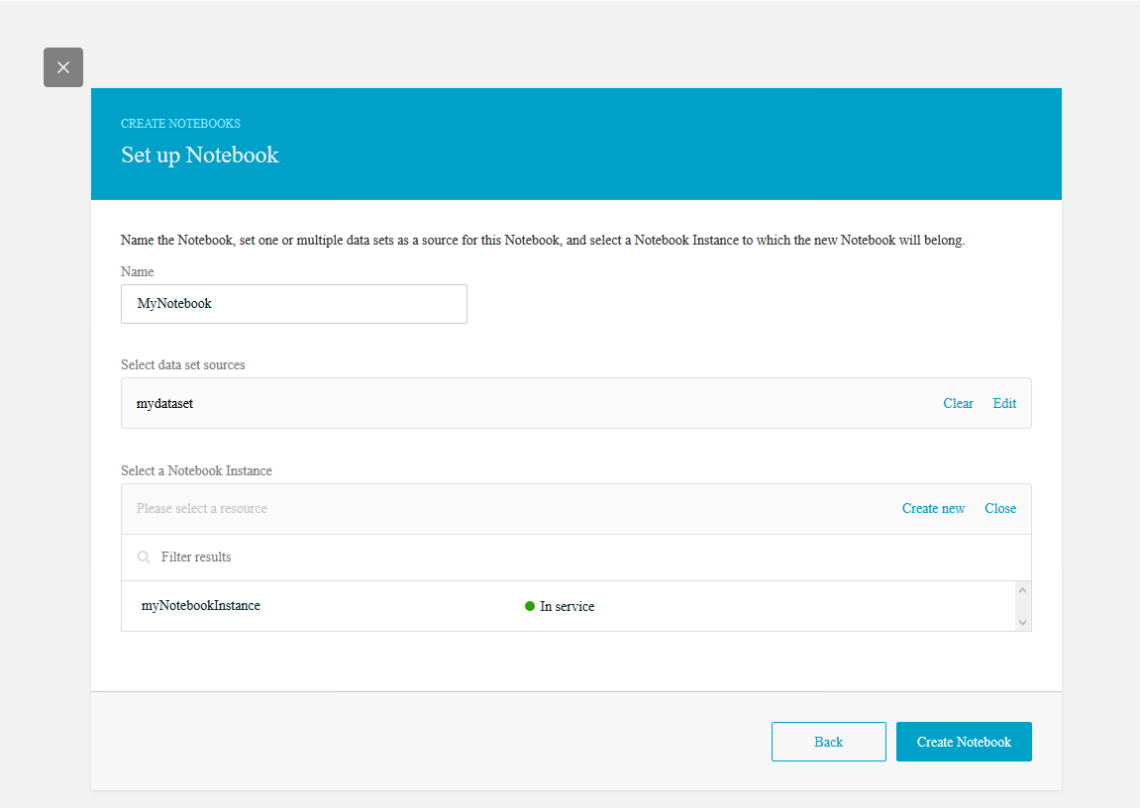
   This gives the newly created role permission to read a data set from AWS IoT Analytics.
3. Return to the AWS IoT Analytics console, and in the left navigation pane, choose **Notebooks**. In the **Gain deeper insight from IoT Data** page choose **Create a notebook**:



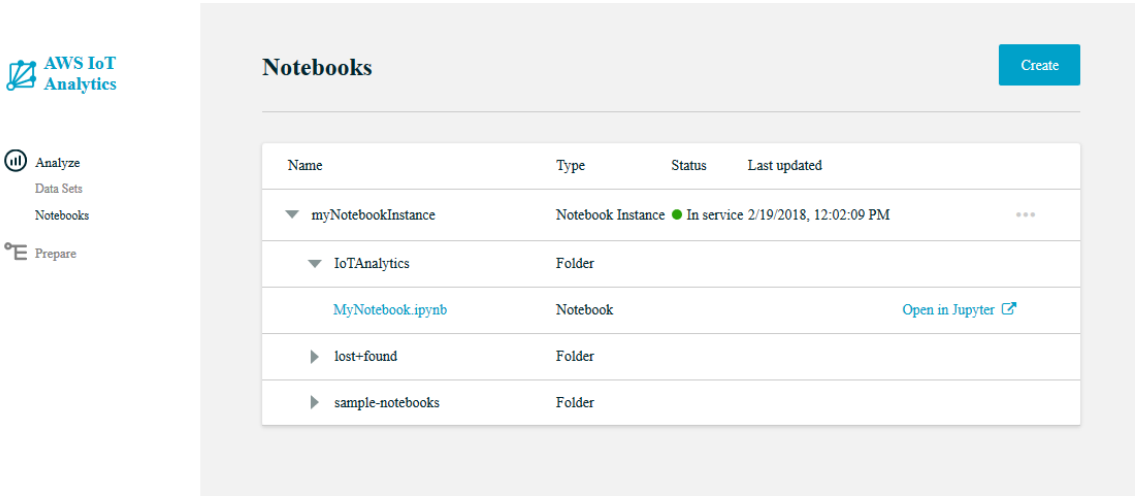
4. On the **Select method** page, choose **Blank Notebook**:



5. On the **Set up notebook** page, enter a name for the notebook. In **Select data set sources**, choose **Select**, and then select the data set you created earlier. In **Select a Notebook Instance**, choose **Select**, and then choose the notebook instance you created in Amazon SageMaker. Choose **Create Notebook**:



6. On the **Notebooks** page, use the triangles to open your notebook instance and the **IoTAnalytics** folder. Use the links to explore your data in Jupyter Notebooks:



You can download and install Jupyter Notebooks on your computer. Additional integration with an Amazon hosted notebook solution is also available.

Notebook Templates

The AWS IoT Analytics notebook templates contain AWS-authored machine learning models and visualizations to help you get started with AWS IoT Analytics use cases. These notebook templates can be explored as-is for educational purposes, or re-purposed to fit your data and deliver immediate value.

AWS IoT Analytics provides the following notebook templates:

1. Detecting Contextual Anomalies: Application of contextual anomaly detection in measured wind speed with a PEWMA model for time series data.
2. Solar Panel Output Forecasting: Application of piecewise, seasonal, linear time series models with trend to predicting the output of solar panels.
3. Predictive Maintenance on Jet Engines: Application of multivariate LSTM neural networks and logistic regression to predict remaining useful life of jet engines.
4. Smart Home Customer Segmentation: Application of k-means and PCA analysis to detect different customer segments in smart home usage data.
5. Smart City Congestion Forecasting: Application of LSTM to predict the utilization rates for city highways.
6. Smart City Air Quality Forecasting: Application of LSTM to predict particulate pollution in city centers.

You can find more information about notebook templates in the AWS IoT Analytics console under **Analyze/Notebooks**.

How to Use AWS IoT Analytics

This section discusses the basic commands you use to collect, store, process and query your device data using AWS IoT Analytics. The examples shown here use the AWS Command Line Interface (AWS CLI). For more information on the CLI, see the AWS Command Line Interface User Guide. For more information about the CLI commands available for AWS IoT, see iot in the *AWS Command Line Interface Reference*.

Important
Use the **aws iotanalytics** *command* to interact with AWS IoT Analytics using the CLI. Use the **aws iot** *command* to interact with other parts of the IoT system using the CLI.

Note
Be aware as you enter the names of AWS IoT Analytics entities (channel, data set, data store, and pipeline) in the examples that follow, that any upper-case letters you use are automatically changed to lower-case by the system. The names of entities must start with a lower-case letter and contain only lower-case letters, underscores and digits.

AWS IoT Analytics Components and Concepts

In this section we cover these basic components and concepts:

Channel - collect the data

A channel collects and archives raw, unprocessed message data before publishing this data to a pipeline.

Pipeline - process the data

A pipeline consumes messages from a channel and enables you to process and filter the messages before storing them in a data store.

Data store - store the data

A data store is not a database, but it is a scalable and queryable repository of your messages. You can have multiple data stores for messages that come from different devices or locations.

Data set - retrieve the data for analysis

You retrieve data from a data store by creating a data set. AWS IoT Analytics enables you to create a SQL data set or a container data set. Here we look at a simple SQL data set which is similar to a materialized view from a SQL database. In fact, you create a SQL data set by applying a SQL action.

Data set contents - get the results

After you create data set contents, you can view them from the console.

Channel

Incoming messages are sent to a channel, so the first step is to create a channel for your data:

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

If you want AWS IoT messages to be ingested into AWS IoT Analytics, you can create an AWS IoT Rules Engine rule to send the messages to this channel. This is shown later in Create an AWS IoT Rule to send messages to AWS IoT Analytics (p. 35). Another way to get the data in to a channel is to use the AWS IoT Analytics command "`BatchPutMessage`".

To list the channels you have already created:

```
aws iotanalytics list-channels
```

To get more information about a channel:

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

Unprocessed channel messages are stored in an Amazon S3 bucket managed by AWS IoT Analytics, or in one managed by you. Use the `channelStorage` parameter to specify which. The default is a service-managed Amazon S3 bucket. If you choose to have channel messages stored in an Amazon S3 bucket that you manage, you must grant AWS IoT Analytics permission to perform these actions on your Amazon S3 bucket on your behalf: `s3:GetBucketLocation` (verify bucket location) `s3:PutObject` (store), `s3:GetObject` (read), `s3:ListBucket` (reprocessing). For 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/*"
            ]
        }
    ]
}
```

If you make changes in the options or permissions of your customer-managed channel storage, you may need to reprocess channel data to ensure that previously ingested data is included in data set contents. See Reprocessing Channel Data (p. 66).

Data Store

A data store receives and stores your messages. You can create multiple data stores to store data according to your needs, or you can use a single data store to receive all of your AWS IoT messages:

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

To list the data stores you have already created:

```
aws iotanalytics list-datastores
```

To get more information about a data store:

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

Processed data store messages are stored in an Amazon S3 bucket managed by AWS IoT Analytics or in one managed by you. Use the `datastoreStorage` parameter to specify which. The default is a service-managed Amazon S3 bucket. If you choose to have data store messages stored in an Amazon S3 bucket that you manage, you must grant AWS IoT Analytics permission to perform these actions on your Amazon S3 bucket on your behalf: `s3:GetBucketLocation` (verify bucket location) `s3:PutObject`, `s3:DeleteObject`. If you use the data store as a source for an SQL query data set, you must set up an Amazon S3 bucket policy that grants AWS IoT Analytics permission to execute Amazon Athena queries on the contents of your bucket. The following is an example bucket policy that grants the required permissions:

```
{
    "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:::my-athena-data-bucket",
                "arn:aws:s3:::my-athena-data-bucket/*"
            ]
        }
    ]
}
```

See Cross-account Access in the **Amazon Athena User Guide** for more information. Also, if you make changes in the options or permissions of your customer-managed data store storage, you may need to reprocess channel data to ensure that previously ingested data is included in data set contents. See Reprocessing Channel Data (p. 66).

Pipeline

To connect a channel to a data store, you create a pipeline. The simplest possible pipeline contains no activities other than specifying the channel that collects the data and identifying the data store to which the messages are sent. For information about more complicated pipelines, see Pipeline Activities (p. 44).

When starting out, we recommend that you create a pipeline that does nothing other than connect a channel to a data store. Then, after you verify that raw data flows to the data store, you can introduce additional pipeline activities to process this data.

Create a pipeline:

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

where the file `mypipeline.json` contains:

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

To list your existing pipelines:

```
aws iotanalytics list-pipelines
```

To view the configuration of an individual pipeline:

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

Get Data Into AWS IoT Analytics

If you have a channel that routes data to a pipeline that stores data in a data store where it can be queried, then you're ready to send message data into AWS IoT Analytics. Here we show two methods of getting data into AWS IoT Analytics. You can send a message using the AWS IoT message broker or use the AWS IoT Analytics "BatchPutMessage" command.

To use the AWS IoT message broker, you create a rule using the AWS IoT Rules Engine. The rule routes messages with a specific topic into AWS IoT Analytics. But first, this rule requires you to create a role which grants the required permissions. We'll show that next. Later in this section we'll also show how to use the AWS IoT Analytics "BatchPutMessage" command to send messages directly to a channel.

Grant Permission with an IAM Role

To have AWS IoT messages routed into an AWS IoT Analytics channel, you set up a rule. But first, you must create an IAM role that grants that rule permission to send message data to an AWS IoT Analytics channel.

Create the role:

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

where the contents of the file `arpd.json` should look like this:

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

Then, you attach a policy document to the role:

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

where the contents of the file `pd.json` looks like this:

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

Create an AWS IoT Rule to send messages to AWS IoT Analytics

Create an AWS IoT rule that sends messages to your channel:

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

The contents of the `rule.json` file should look like this:

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

Replace `iot/test` with the MQTT topic of the messages that should be routed. Replace the channel name and the role with the ones you created in the previous sections.

Use the AWS IoT Console to Send Message Data Into AWS IoT Analytics

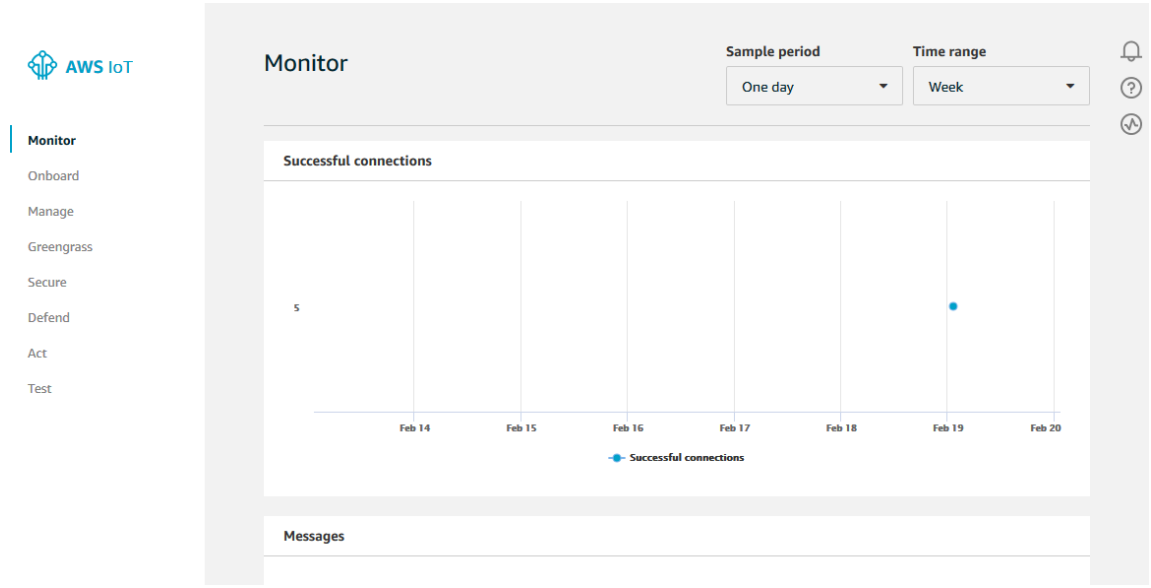
After you have joined a rule to a channel, a channel to a pipeline, and a pipeline to a data store, any data matching the rule now flows through AWS IoT Analytics to the data store ready to be queried. To test this, you can use the AWS IoT console to send a message.

Note
The field names of message payloads (data) that you send to AWS IoT Analytics:

- Must contain only alphanumeric characters and underscores (_); no other special characters are allowed.
- Must begin with an alphabetic character or single underscore (_).
- Cannot contain hyphens (-).
- In regular expression terms: `"^[A-Za-z_]([A-Za-z0-9]*|[A-Za-z0-9][A-Za-z0-9_]*)$"`.
- Cannot be greater than 255 characters.
- Are case-insensitive. (Fields named “foo” and “FOO” in the same payload are considered duplicates.)

For example, `{"temp_01": 29}` or `{"_temp_01": 29}` are valid, but `{"temp-01": 29}`, `{"01_temp": 29}` or `{"__temp_01": 29}` are invalid in message payloads.

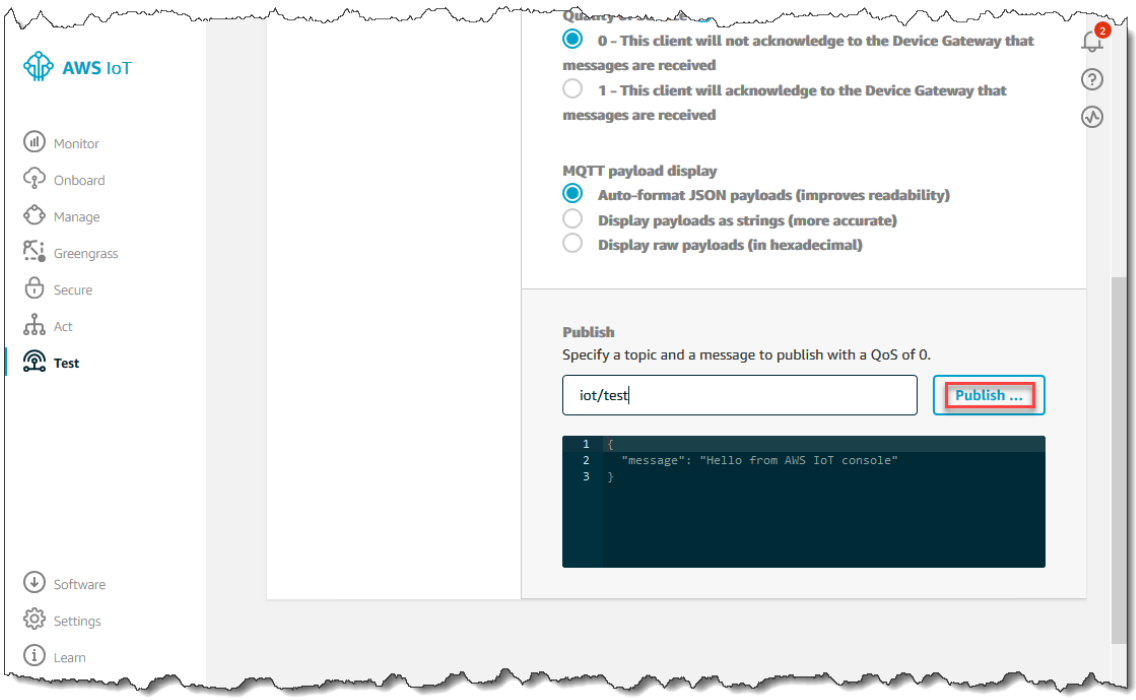
1. In the AWS IoT console, in the left navigation pane, choose **Test**.



2. On the MQTT client page, in the **Publish** section, in **Specify a topic**, type `iot/test`. In the message payload section, verify the following JSON contents are present, or type them if not:

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

3. Choose **Publish to topic**.



This publishes a message that is routed to the data store you created earlier.

Use “BatchPutMessage” to Send a Message to the Channel

Another way to get message data into AWS IoT Analytics is to use the “BatchPutMessage” API command. This method does not require that you set up an AWS IoT rule to route messages with a specific topic to your channel. But it does require that the device which sends its data/messages to the channel is capable of running software created with the AWS SDK or is capable of using the AWS CLI to call “BatchPutMessage”.

1. Create a file “messages.json” which contains the messages to be sent (in this example only one message is sent):

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

2. Call `batch-put-message`:

```
aws iotanalytics batch-put-message  --channel-name mychannel  --messages file://
messages.json
```

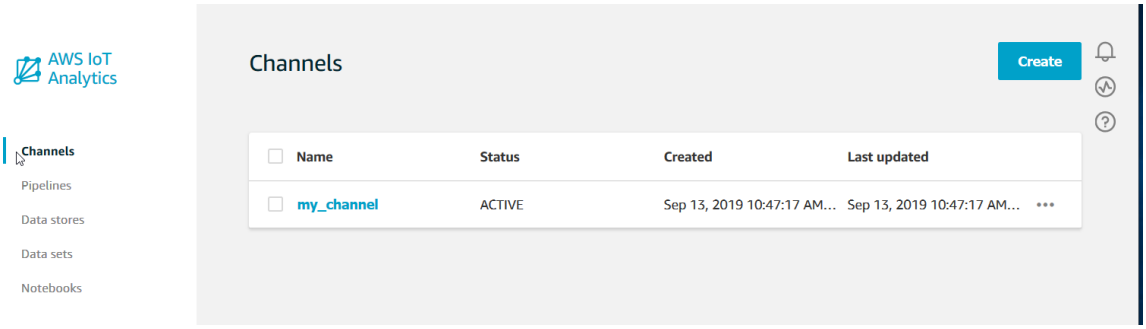
3. If there are no errors, you see the following output:

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

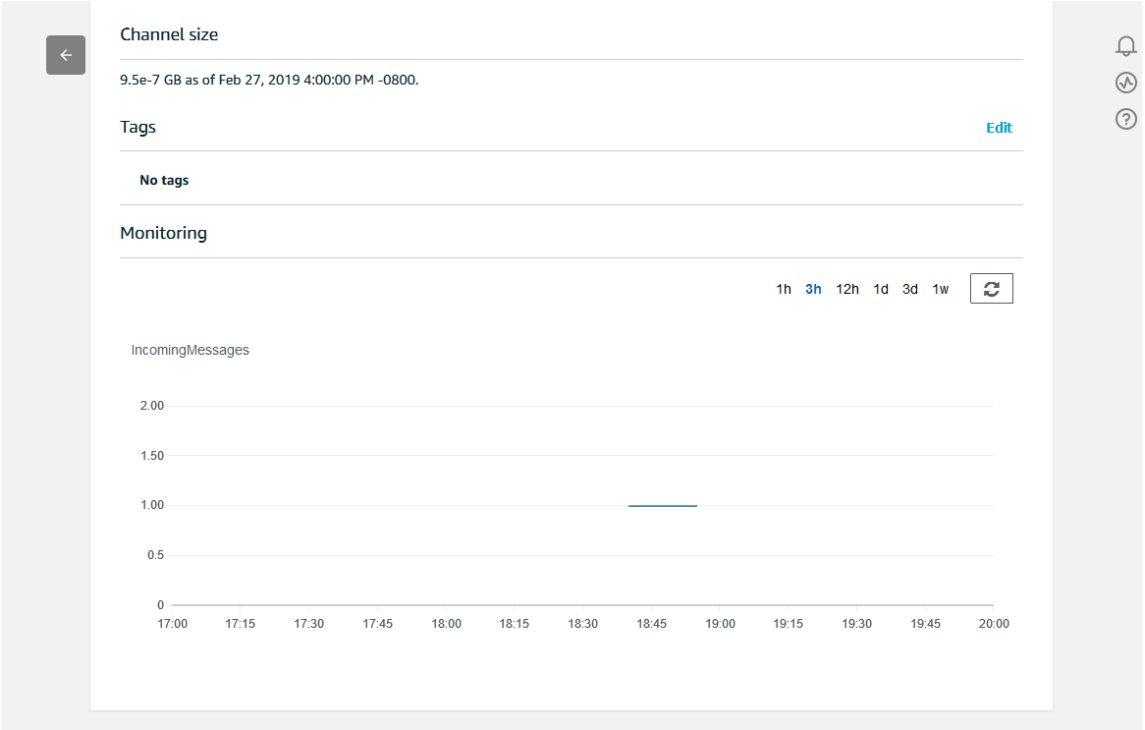
Checking the Progress of IoT Messages

You can check that the messages you sent are being ingested into your channel by using the AWS IoT Analytics console. Follow these steps:

1. In the AWS IoT Analytics console, in the left navigation pane, choose **Prepare** and (if necessary) choose **Channels**, then choose the name of the channel you created earlier:

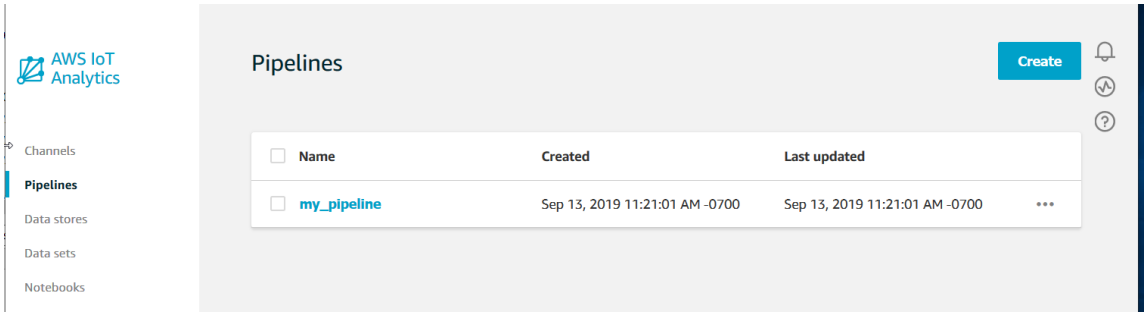


2. On the channel detail page, scroll down to the **Monitoring** section. Adjust the displayed time frame as necessary by choosing one of the time frame indicators (**1h 3h 12h 1d 3d 1w**). You should see a graph line indicating the number of messages ingested into this channel during the specified time frame:

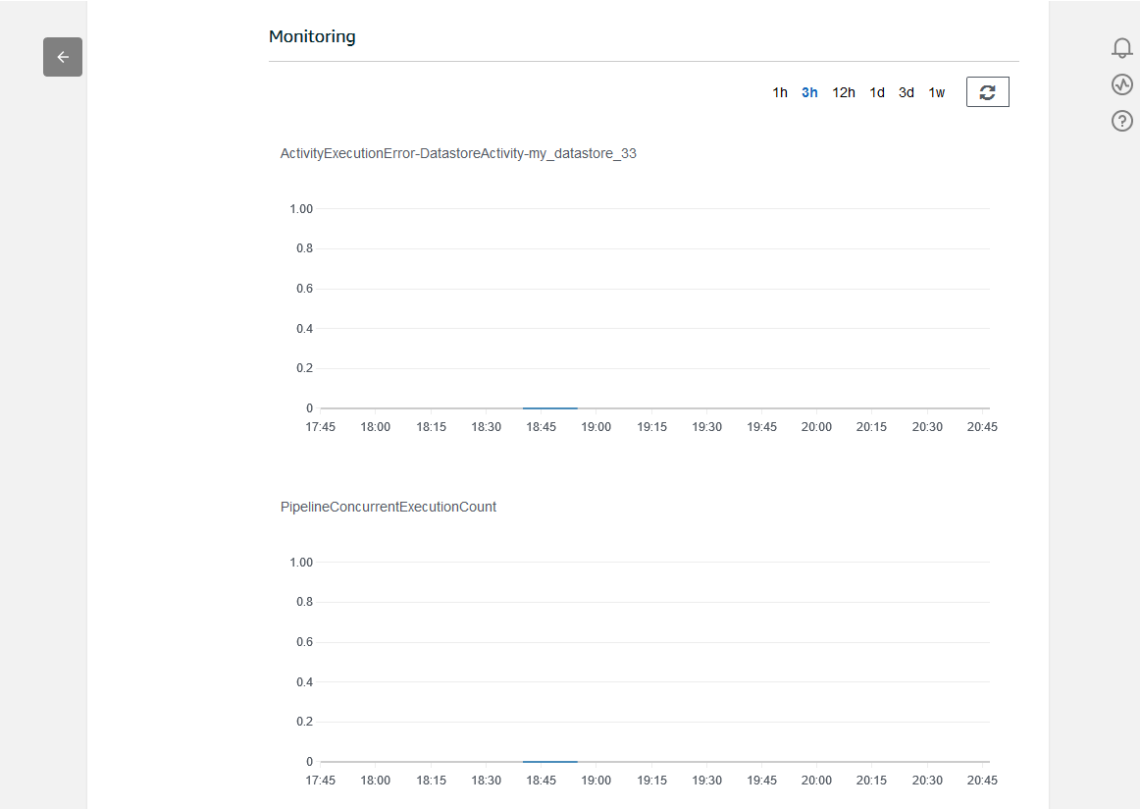


A similar monitoring capability exists for checking pipeline activity executions. You can monitor activity execution errors on the pipeline's detail page. (If you haven't specified activities as part of your pipeline, then 0 execution errors should be displayed.)

1. In the AWS IoT Analytics console, in the left navigation pane, choose **Prepare** and then choose **Pipelines**, then choose the name of a pipeline you created earlier:



2. On the pipeline detail page, scroll down to the **Monitoring** section. Adjust the displayed time frame as necessary by choosing one of the time frame indicators (**1h 3h 12h 1d 3d 1w**). You should see a graph line indicating the number of pipeline activity execution errors during the specified time frame:



Data Set - Query Your Data

When you have data in a data store, you can query it to answer analytical questions. Although a data store is not a database, you use SQL expressions to query the data and produce results that are stored in a data set.

To query the data, you create a data set. A data set contains the SQL that you use to query the data store along with an optional schedule that repeats the query at a day and time you choose. You create the optional schedules using expressions similar to Amazon CloudWatch schedule expressions.

Create a data set:

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

where the file `mydataset.json` contains:

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

Create the data set content by executing the query:

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

Wait a few minutes for the data set content to be created before you continue.

Data Set Content - Access the Query Results

The result of the query is your data set content, stored as a file, in CSV format. The file is made available to you through Amazon S3. The following example shows how you can check that your results are ready and download the file.

Call `get-dataset-content`:

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

If your data set contains any data, then the output from `get-dataset-content` has `"state": "SUCCEEDED"` in the `status` field, like this:

```
{
    "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-
          Security-Token=<TOKEN>&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-
Date=20171016T214541Z&X-Amz-
          SignedHeaders=host&X-Amz-Expires=7200&X-Amz-Credential=<CREDENTIAL>&X-Amz-
          Signature=<SIGNATURE>"
        }
    ],
    "status": {
      "state": "SUCCEEDED",
      "reason": "A useful comment."
    }
}
```

`dataURI` is a signed URL to the output results. It is valid for a short period of time (a few hours). Depending on your workflow, you might want to always call `get-dataset-content` before you access the content because calling this command generates a new signed URL.

Explore Your Data

You have several options for storing, analyzing and visualizing your data…

Amazon S3

You can send data set contents to an Amazon Simple Storage Service (S3) bucket, enabling integration with your existing data lakes or access from in-house applications and visualization tools. See the field `contentDeliveryRules::destination::s3DestinationConfiguration` in CreateDataset (p. 141).

AWS IoT Events

You can send data set contents as an input to AWS IoT Events, a service which enables you to monitor devices or processes for failures or changes in operation, and to trigger additional actions when such events occur.

To do this, create a data set using CreateDataset (p. 141) and specify an AWS IoT Events input in the field `contentDeliveryRules :: destination :: iotEventsDestinationConfiguration :: inputName`. You must also specify the `roleArn` of a role which grants AWS IoT Analytics permission to execute "iotevents:BatchPutMessage". Whenever the data set's contents are created, AWS IoT Analytics will send each data set content entry as a message to the specified AWS IoT Events input. For example, if your data set contains:

```
"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"
...
```

then AWS IoT Analytics will send messages containing fields like this:

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

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

and you will want to create an AWS IoT Events input that recognizes the fields you are interested in (one or more of `what`, `who`, `dt`) and to create an AWS IoT Events detector model that uses these input fields in events to trigger actions or set internal variables.

Amazon QuickSight

AWS IoT Analytics provides direct integration with Amazon QuickSight. Amazon QuickSight is a fast business analytics service you can use to build visualizations, perform ad-hoc analysis, and quickly get business insights from your data. Amazon QuickSight enables organizations to scale to hundreds of thousands of users, and delivers responsive performance by using a robust in-memory engine (SPICE). Amazon QuickSight is available in these regions.

Jupyter Notebooks

AWS IoT Analytics data sets can also be directly consumed by Jupyter Notebooks in order to perform advanced analytics and data exploration. Jupyter Notebooks is an open source solution. You can install and download from http://jupyter.org/install.html. Additional integration with SageMaker, an Amazon hosted notebook solution, is also available.

Keeping Multiple Versions of AWS IoT Analytics Data Sets

You can choose how many versions of your data set contents to retain, and for how long, by specifying values for the data set `retentionPeriod` and `versioningConfiguration` fields when invoking the CreateDataset (p. 141) and UpdateDataset (p. 220) APIs:

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

The settings of these two parameters work together to determine how many versions of data set contents are retained, and for how long, in the following ways:

	retentionPeriod [not specified]	**retentionPeriod:** unlimited = TRUE, numberOfDays = not set	**retentionPeriod:** unlimited = FALSE, numberOfDays = X
versioningConfiguration: [not specified]	Only the latest version plus the latest succeeded version (if different) are retained for 90 days.	Only the latest version plus the latest succeeded version (if different) are retained for an unlimited time.	Only the latest version plus the latest succeeded version (if different) are retained for X days.
versioningConfiguration: unlimited = TRUE, maxVersions not set	All versions from the last 90 days will be retained, regardless of how many.	There is no limit to the number of versions retained.	All versions from the last X days will be retained, regardless of how many.
versioningConfiguration: unlimited = FALSE, maxVersions = Y	No more than Y versions from the last 90 days will be retained.	Up to Y versions will be retained, regardless of how old they are.	No more than Y versions from the last X days will be retained.

AWS IoT Analytics Message Payload Restrictions

The field names of message payloads (data) that you send to AWS IoT Analytics:

- Must contain only alphanumeric characters and underscores (_); no other special characters are allowed.
- Must begin with an alphabetic character or single underscore (_).
- Cannot contain hyphens (-).
- In regular expression terms: `"^[A-Za-z_]([A-Za-z0-9]*|[A-Za-z0-9][A-Za-z0-9_]*)$"`.

- Cannot be greater than 255 characters.
- Are case-insensitive. (Fields named “foo” and “FOO” in the same payload are considered duplicates.)

For example, `{"temp_01": 29}` or `{"_temp_01": 29}` are valid, but `{"temp-01": 29}`, `{"01_temp": 29}` or `{"__temp_01": 29}` are invalid in message payloads.

AWS IoT Analytics Service Limits

API	Limit Description	Adjustable?
`SampleChannelData`	1 transaction per second per channel	yes
`CreateDatasetContent`	1 transaction per second per data set	yes
`RunPipelineActivity`	1 transaction per second	yes
other management APIs	20 transactions per second	yes
`BatchPutMessage`	100,000 messages or 500MB total message size per second per channel; 100 messages per batch; 128Kb per message	yes; yes; no

Resource	Limit Description	Adjustable?
channel	50 per account	yes
data store	25 per account	yes
pipeline	100 per account	yes
activities	25 per pipeline	no
data set	100 per account	yes
minimum SQL data set refresh interval	1 minute	no
minimum container data set refresh interval	15 minutes	yes
concurrent data set content generation	2 data sets simultaneously	no
container data sets that can be triggered from a single SQL data set	10	no
concurrent container data set runs	20	no

Pipeline Activities

The simplest functional pipeline connects a channel to a data store, which makes it a pipeline with two activities: a `channel` activity and a `datastore` activity. You can achieve more powerful message processing by adding additional activities to your pipeline.

AWS IoT Analytics provides the RunPipelineActivity (p. 206) command which enables you to simulate the results of running a pipeline activity on a message payload you provide. You might find this helpful when you are developing and debugging your pipeline activities. RunPipelineActivity Example (p. 64) demonstrates how it is used.

Channel Activity

The first activity in a pipeline must be the `channel` activity which determines the source of the messages to be processed.

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

Datastore Activity

The `datastore` activity, which specifies where to store the processed data, is the last activity.

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

Lambda Activity

A `lambda` activity can be used to perform more complex processing on the message. Examples include enriching the message with data from the output of external APIs or filtering the message based on logic from DynamoDB. However, you can use this activity to perform any sort of message-based processing, including filtering which messages are stored in the data store.

The AWS Lambda function used in this activity *must* receive and return an array of JSON objects. In the following example, the Lambda function modifies, and then returns, its `event` parameter.

The `batchSize` determines how many messages your Lambda function receives on each invocation. When you set it, keep in mind that an AWS Lambda function has a maximum timeout of five minutes. So the Lambda function must be able to process all messages in the batch in less than five minutes.

```
{
    "lambda": {
        "name": "MyLambdaActivity",
        "lambdaName": "mylambda",
        "batchSize": 10,
        "next": "MyDatastoreActivity"
    }
}
```

You must add a function policy to allow AWS IoT Analytics to invoke your Lambda function. Use the following CLI command:

```
aws lambda add-permission --function-name <lambda-function-name> --statement-id <your-
statement> --principal iotanalytics.amazonaws.com --action lambda:InvokeFunction
```

Lambda Function Example 1

In this example, the Lambda function adds additional information based on data in the original message. Given a device that publishes a message with a payload similar to:

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

and the following pipeline definition:

```
{
    "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"
    }
}
```

The following Lambda Python function (MyAnalyticsLambdaFunction) adds the GMaps URL and the temperature, in Fahrenheit, to the message:

```
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 Function Example 2

A useful technique is to compress and serialize message payloads to reduce transport and storage costs. In this second example, the Lambda function assumes the message payload represents a JSON original which has been compressed and then base64-encoded (serialized) as a string. It returns the original 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 Activity

An `addAttributes` activity adds attributes based on existing attributes in the message. This lets you alter the shape of the message before it is stored. For example, you can use `addAttributes` to normalize data coming from different generations of device firmware.

This is best explained by example. Consider the input message:

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

and an `addAttributes` activity that looks like this:

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

This activity moves the device ID to the root level and extracts the values in the `coord` array, promoting them to top-level attributes called `lat` and `lon`. As a result of this activity, the input message is transformed to the following:

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

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

The original device attribute is still present. If you want to remove it, you can use the `removeAttributes` activity.

RemoveAttributes Activity

A `removeAttributes` activity removes attributes from a message. For example, given the message that was the result of the `addAttributes` activity:

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

To normalize that message so that it includes only the required data at the root level, use the following `removeAttributes` activity:

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

This results in the following message flowing along the pipeline:

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

SelectAttributes Activity

The `selectAttributes` activity creates a new message using only the specified attributes from the original message. Every other attribute is dropped. `selectAttributes` creates new attributes under the root of the message only. So given this message:

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

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

and this activity:

```
{
    "selectAttributes": {
        "name": "MySelectAttributesActivity",
        "attributes": [
            "device.temp",
            "device.hum",
            "light"
        ],
        "next": "MyDatastoreActivity"
    }
}
```

The result is the following message flowing through the pipeline:

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

Again, `selectAttributes` can only create root-level objects.

Filter Activity

A `filter` activity filters a message based on its attributes. The expression used in this activity looks like an SQL WHERE clause which *must* return a boolean:

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

DeviceRegistryEnrich Activity

The `deviceRegistryEnrich` activity enables you to add data from the AWS IoT device registry to your message payload. For example, given the following message:

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

and a `deviceRegistryEnrich` activity that looks like this:

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

The output message now looks like this:

```
{
    "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"
    }
}
```

You must specify a role in the `roleArn` field of the activity definition that has the appropriate permissions attached. The role must have a permissions policy that looks like:

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

and a trust policy that looks like:

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

```
                "sts:AssumeRole"
            ]
        }
    ]
}
```

DeviceShadowEnrich Activity

A `deviceShadowEnrich` activity adds information from the AWS IoT Device Shadows service to a message. For example, given the message:

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

and the following `deviceShadowEnrich` activity:

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

the result is a message that looks like this:

```
{
    "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
    }
}
```

You must specify a role in the `roleArn` field of the activity definition that has the appropriate permissions attached. The role must have a permissions policy that looks like:

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

and a trust policy that looks like:

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

Math Activity

A `math` activity computes an arithmetic expression using the message's attributes. The expression *must* return a number. For example, given the following input message:

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

after processing by the following `math` activity:

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

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

the resulting message looks like:

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

Math Activity Operators and Functions

You can use the following operators in a `math` activity:

+	addition
-	subtraction
*****	multiplication
/	division
%	modulo

You can use the following functions in a `math` activity:

abs(Decimal)

Returns the absolute value of a number.

Example: `abs(-5)` returns 5.

Argument Type	Result
`Int`	`Int`, the absolute value of the argument.
`Decimal`	`Decimal`, the absolute value of the argument.
`Boolean`	`Undefined`.
`String`	`Decimal`. The result is the absolute value of the argument. If the string cannot be converted, the result is `Undefined`.
Array	`Undefined`.
Object	`Undefined`.
Null	`Undefined`.
Undefined	`Undefined`.

acos(Decimal)

Returns the inverse cosine of a number in radians. `Decimal` arguments are rounded to double precision before function application.

Example: `acos(0)` = 1.5707963267948966

Argument Type	Result
`Int`	`Decimal` (with double precision), the inverse cosine of the argument. Imaginary results are returned as `Undefined`.
`Decimal`	`Decimal` (with double precision), the inverse cosine of the argument. Imaginary results are returned as `Undefined`.
`Boolean`	`Undefined`.
`String`	`Decimal`, the inverse cosine of the argument. If the string cannot be converted, the result is `Undefined`. Imaginary results are returned as `Undefined`.
Array	`Undefined`.
Object	`Undefined`.
Null	`Undefined`.
Undefined	`Undefined`.

asin(Decimal)

Returns the inverse sine of a number in radians. `Decimal` arguments are rounded to double precision before function application.

Example: `asin(0)` = 0.0

Argument Type	Result
`Int`	`Decimal` (with double precision), the inverse sine of the argument. Imaginary results are returned as `Undefined`.
`Decimal`	`Decimal` (with double precision), the inverse sine of the argument. Imaginary results are returned as `Undefined`.
`Boolean`	`Undefined`.
`String`	`Decimal` (with double precision), the inverse sine of the argument. If the string cannot be converted, the result is `Undefined`. Imaginary results are returned as `Undefined`.
Array	`Undefined`.

Argument Type	Result
Object	`Undefined`.
Null	`Undefined`.
Undefined	`Undefined`.

atan(Decimal)

Returns the inverse tangent of a number in radians. `Decimal` arguments are rounded to double precision before function application.

Example: `atan(0)` = 0.0

Argument Type	Result
`Int`	`Decimal` (with double precision), the inverse tangent of the argument. Imaginary results are returned as `Undefined`.
`Decimal`	`Decimal` (with double precision), the inverse tangent of the argument. Imaginary results are returned as `Undefined`.
`Boolean`	`Undefined`.
`String`	`Decimal`, the inverse tangent of the argument. If the string cannot be converted, the result is `Undefined`. Imaginary results are returned as `Undefined`.
Array	`Undefined`.
Object	`Undefined`.
Null	`Undefined`.
Undefined	`Undefined`.

atan2(Decimal, Decimal)

Returns the angle, in radians, between the positive x-axis and the (x, y) point defined in the two arguments. The angle is positive for counter-clockwise angles (upper half-plane, y > 0), and negative for clockwise angles (lower half-plane, y < 0). `Decimal` arguments are rounded to double precision before function application.

Example: `atan2(1, 0)` = 1.5707963267948966

Argument Type	Argument Type	Result
`Int` / `Decimal`	`Int` / `Decimal`	`Decimal` (with double precision), the angle between the x-axis and the specified (x,y) point.

Argument Type	Argument Type	Result
`Int` / `Decimal` / `String`	`Int` / `Decimal` / `String`	`Decimal`, the inverse tangent of the point described. If a string cannot be converted, the result is `Undefined`.
Other Value	Other Value	`Undefined`.

ceil(Decimal)

Rounds the given `Decimal` up to the nearest `Int`.

Examples:

`ceil(1.2)` = 2

`ceil(11.2)` = -1

Argument Type	Result
`Int`	`Int`, the argument value.
`Decimal`	`Int`, the `Decimal` value rounded up to the nearest `Int`.
`String`	`Int`. The string is converted to `Decimal` and rounded up to the nearest `Int`. If the string cannot be converted to a `Decimal`, the result is `Undefined`.
Other Value	`Undefined`.

cos(Decimal)

Returns the cosine of a number in radians. `Decimal` arguments are rounded to double precision before function application.

Example:

`cos(0)` = 1.

Argument Type	Result
`Int`	`Decimal` (with double precision), the cosine of the argument. Imaginary results are returned as `Undefined`.
`Decimal`	`Decimal` (with double precision), the cosine of the argument. Imaginary results are returned as `Undefined`.
`Boolean`	`Undefined`.
`String`	`Decimal` (with double precision), the cosine of the argument. If the string cannot be converted

Argument Type	Result
	to a `Decimal`, the result is `Undefined`. Imaginary results are returned as `Undefined`.
Array	`Undefined`.
Object	`Undefined`.
Null	`Undefined`.
Undefined	`Undefined`.

cosh(Decimal)

Returns the hyperbolic cosine of a number in radians. `Decimal` arguments are rounded to double precision before function application.

Example: `cosh(2.3)` = 5.037220649268761.

Argument Type	Result
`Int`	`Decimal` (with double precision), the hyperbolic cosine of the argument. Imaginary results are returned as `Undefined`.
`Decimal`	`Decimal` (with double precision), the hyperbolic cosine of the argument. Imaginary results are returned as `Undefined`.
`Boolean`	`Undefined`.
`String`	`Decimal` (with double precision), the hyperbolic cosine of the argument. If the string cannot be converted to a `Decimal`, the result is `Undefined`. Imaginary results are returned as `Undefined`.
Array	`Undefined`.
Object	`Undefined`.
Null	`Undefined`.
Undefined	`Undefined`.

exp(Decimal)

Returns e raised to the `Decimal` argument. `Decimal` arguments are rounded to double precision before function application.

Example: `exp(1)` = e.

Argument Type	Result
`Int`	`Decimal` (with double precision), e ^ argument.

Argument Type	Result
`Decimal`	`Decimal` (with double precision), e ^ argument.
`String`	`Decimal` (with double precision), e ^ argument. If the `String` cannot be converted to a `Decimal`, the result is `Undefined`.
Other Value	`Undefined`.

ln(Decimal)

Returns the natural logarithm of the argument. `Decimal` arguments are rounded to double precision before function application.

Example: `ln(e)` = 1.

Argument Type	Result
`Int`	`Decimal` (with double precision), the natural log of the argument.
`Decimal`	`Decimal` (with double precision), the natural log of the argument.
`Boolean`	`Undefined`.
`String`	`Decimal` (with double precision), the natural log of the argument. If the string cannot be converted to a `Decimal` the result is `Undefined`.
Array	`Undefined`.
Object	`Undefined`.
Null	`Undefined`.
Undefined	`Undefined`.

log(Decimal)

Returns the base 10 logarithm of the argument. `Decimal` arguments are rounded to double precision before function application.

Example: `log(100)` = 2.0.

Argument Type	Result
`Int`	`Decimal` (with double precision), the base 10 log of the argument.
`Decimal`	`Decimal` (with double precision), the base 10 log of the argument.
`Boolean`	`Undefined`.

Argument Type	Result
`String`	`Decimal` (with double precision), the base 10 log of the argument. If the `String` cannot be converted to a `Decimal`, the result is `Undefined`.
Array	`Undefined`.
Object	`Undefined`.
Null	`Undefined`.
Undefined	`Undefined`.

mod(Decimal, Decimal)

Returns the remainder of the division of the first argument by the second argument. You can also use "%" as an infix operator for the same modulo functionality.

Example: `mod(8, 3)` = 2.

Left Operand	Right Operand	Output
`Int`	`Int`	`Int`, the first argument modulo the second argument.
`Int` / `Decimal`	`Int` / `Decimal`	`Decimal`, the first argument modulo the second operand.
`String` / `Int` / `Decimal`	`String` / `Int` / `Decimal`	If all strings convert to `Decimal`s, the result is the first argument modulo the second argument. Otherwise, `Undefined`.
Other Value	Other Value	`Undefined`.

power(Decimal, Decimal)

Returns the first argument raised to the second argument. `Decimal` arguments are rounded to double precision before function application.

Example: `power(2, 5)` = 32.0.

argument Type 1	argument Type 2	Output
`Int` / `Decimal`	`Int` / `Decimal`	A `Decimal` (with double precision), the first argument raised to the second argument's power.
`Int` / `Decimal` / `String`	`Int` / `Decimal` `String`	A `Decimal` (with double precision), the first argument raised to the second argument's

argument Type 1	argument Type 2	Output
		power. Any strings are converted to `Decimals`. If any `String` fails to be converted to `Decimal`, the result is `Undefined`.
Other Value	Other Value	`Undefined`.

round(Decimal)

Rounds the given `Decimal` to the nearest `Int`. If the `Decimal` is equidistant from two `Int` values (for example, 0.5), the `Decimal` is rounded up.

Example: `Round(1.2)` = 1.

`Round(1.5)` = 2.

`Round(1.7)` = 2.

`Round(-1.1)` = -1.

`Round(-1.5)` = -2.

Argument Type	Result
`Int`	The argument.
`Decimal`	`Decimal` is rounded down to the nearest `Int`.
`String`	`Decimal` is rounded down to the nearest `Int`. If the string cannot be converted to a `Decimal`, the result is `Undefined`.
Other Value	`Undefined`.

sign(Decimal)

Returns the sign of the given number. When the sign of the argument is positive, 1 is returned. When the sign of the argument is negative, -1 is returned. If the argument is 0, 0 is returned.

Examples:

`sign(-7)` = -1.

`sign(0)` = 0.

`sign(13)` = 1.

Argument Type	Result
`Int`	`Int`, the sign of the `Int` value.
`Decimal`	`Int`, the sign of the `Decimal` value.

Argument Type	Result
`String`	`Int`, the sign of the `Decimal` value. The string is converted to a `Decimal` value, and the sign of the `Decimal` value is returned. If the `String` cannot be converted to a `Decimal`, the result is `Undefined`.
Other Value	`Undefined`.

sin(Decimal)

Returns the sine of a number in radians. `Decimal` arguments are rounded to double precision before function application.

Example: `sin(0)` = 0.0

Argument Type	Result
`Int`	`Decimal` (with double precision), the sine of the argument.
`Decimal`	`Decimal` (with double precision), the sine of the argument.
`Boolean`	`Undefined`.
`String`	`Decimal` (with double precision), the sine of the argument. If the string cannot be converted to a `Decimal`, the result is `Undefined`.
Array	`Undefined`.
Object	`Undefined`.
Null	`Undefined`.
`Undefined`	`Undefined`.

sinh(Decimal)

Returns the hyperbolic sine of a number. `Decimal` values are rounded to double precision before function application. The result is a `Decimal` value of double precision.

Example: `sinh(2.3)` = 4.936961805545957

Argument Type	Result
`Int`	`Decimal` (with double precision), the hyperbolic sine of the argument.
`Decimal`	`Decimal` (with double precision), the hyperbolic sine of the argument.
`Boolean`	`Undefined`.

Argument Type	Result
`String`	`Decimal` (with double precision), the hyperbolic sine of the argument. If the string cannot be converted to a `Decimal`, the result is `Undefined`.
Array	`Undefined`.
Object	`Undefined`.
Null	`Undefined`.
Undefined	`Undefined`.

sqrt(Decimal)

Returns the square root of a number. `Decimal` arguments are rounded to double precision before function application.

Example: `sqrt(9)` = 3.0.

Argument Type	Result
`Int`	The square root of the argument.
`Decimal`	The square root of the argument.
`Boolean`	`Undefined`.
`String`	The square root of the argument. If the string cannot be converted to a `Decimal`, the result is `Undefined`.
Array	`Undefined`.
Object	`Undefined`.
Null	`Undefined`.
Undefined	`Undefined`.

tan(Decimal)

Returns the tangent of a number in radians. `Decimal` values are rounded to double precision before function application.

Example: `tan(3)` = -0.1425465430742778

Argument Type	Result
`Int`	`Decimal` (with double precision), the tangent of the argument.
`Decimal`	`Decimal` (with double precision), the tangent of the argument.

Argument Type	Result
`Boolean`	`Undefined`.
`String`	`Decimal` (with double precision), the tangent of the argument. If the string cannot be converted to a `Decimal`, the result is `Undefined`.
Array	`Undefined`.
Object	`Undefined`.
Null	`Undefined`.
Undefined	`Undefined`.

tanh(Decimal)

Returns the hyperbolic tangent of a number in radians. `Decimal` values are rounded to double precision before function application.

Example: `tanh(2.3)` = 0.9800963962661914

Argument Type	Result
`Int`	`Decimal` (with double precision), the hyperbolic tangent of the argument.
`Decimal`	`Decimal` (with double precision), the hyperbolic tangent of the argument.
`Boolean`	`Undefined`.
`String`	`Decimal` (with double precision), the hyperbolic tangent of the argument. If the string cannot be converted to a `Decimal`, the result is `Undefined`.
Array	`Undefined`.
Object	`Undefined`.
Null	`Undefined`.
Undefined	`Undefined`.

trunc(Decimal, Int)

Truncates the first argument to the number of `Decimal` places specified by the second argument. If the second argument is less than zero, it will be set to zero. If the second argument is greater than 34, it will be set to 34. Trailing zeroes are stripped from the result.

Examples:

`trunc(2.3, 0)` = 2.

`trunc(2.3123, 2` = 2.31.

`trunc(2.888, 2)` = 2.88.

`(2.00, 5)` = 2.

argument Type 1	argument Type 2	Result
`Int`	`Int`	The source value.
`Int` / `Decimal` / `String`	`Int` / `Decimal`	The first argument is truncated to the length described by the second argument. The second argument, if not an `Int`, will be rounded down to the nearest `Int`. `Strings` are converted to `Decimal` values. If the string conversion fails, the result is `Undefined`.
Other Value		`Undefined`.

RunPipelineActivity Example

Here is an example of how you would use the "RunPipelineActivity" command to test a pipeline activity. For this example we test a Math activity:

1. Create a file "maths.json" which contains the definition of the pipeline activity you want to test:

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

2. Create a file "payloads.json" which contains the example payloads that are used to test the pipeline activity:

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

3. Call "RunPipelineActivities" from the command line:

```
aws iotanalytics run-pipeline-activity  --pipeline-activity file://maths.json  --payloads
 file://payloads.json
```

4. This produces the following results:

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

5. The “payloads” listed in the results are Base64-encoded strings. When these strings are decoded, you get the following results:

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

Reprocessing Channel Data

AWS IoT Analytics enables you to reprocess channel data or, to put it another way, to replay existing raw data through a pipeline. This can be useful if:

- You want to replay existing ingested data rather than starting over.
- You make an update to a pipeline and want to bring existing data up-to-date with the changes.
- You make changes to your customer-based storage options or permissions for channels or data stores and you want to include data which was ingested before these changes in data set contents going forward.

To trigger the reprocessing of existing raw data, use the StartPipelineReprocessing (p. 214) command. Note the following:

- The “startTime” and “endTime” parameters specify when the raw data was ingested, but these are rough estimates. You can round to the nearest hour. The “startTime” is inclusive, but the “endTime” is exclusive.
- The command launches the reprocessing asynchronously and returns immediately.
- There is no guarantee that reprocessed messages are processed in the order they were originally received. It is roughly the same, but not exact.
- Reprocessing your raw data will incur additional costs.

To cancel the reprocessing of existing raw data, use the CancelPipelineReprocessing (p. 137) command.

Use the DescribePipeline (p. 181) command to check the status of the reprocessing. See the “reprocessingSummaries” field in the response.

Automating Your Workflow

AWS IoT Analytics provides advanced data analysis for AWS IoT. You can automatically collect IoT data, process it, store it and analyze it using data analysis and machine-learning tools. You can execute containers that host your own custom analytical code or Jupyter Notebooks or use third party custom code containers so you don't have to recreate existing analytical tools. You can use the following capabilities to take input data from a data store and feed it into an automated workflow:

Create data set content on a recurring schedule.

Schedule the automatic creation of data set content by specifying a trigger when you call `CreateDataset` (see `triggers:schedule:expression`). Data that has arrived in a data store is used to create the data set content. You select the fields you want by using a SQL query (`actions:queryAction:sqlQuery`).

Define a non-overlapping, contiguous time interval to ensure the new data set content contains only that data which has arrived since the last time. Use the `actions:queryAction:filters:deltaTime` and `:offsetSeconds` fields to specify the delta time interval. Then specify a trigger to create the data set content when the time interval has elapsed. See Example 6 – Creating a SQL dataset with a Delta Window (CLI): (p. 84).

Create data set content upon completion of another data set.

Trigger creation of new data set content when another data set's content creation is complete (`triggers:dataset:name`).

Automatically run your analysis applications.

Containerize your own, custom data analysis applications and trigger them to run when another data set's content is created. This way, you can feed your application with data from a data set's content that is created on a recurring schedule. You can automatically take action on the results of your analysis from within your application. (`actions:containerAction`)

Use Cases

Automate product quality measurement to lower OpEx

You have a system with a smart valve that measures pressure, humidity and temperature. The system collates events periodically and also when certain events occur, such as when a valve opens and closes. With AWS IoT Analytics, you can automate an analysis that aggregates non-overlapping data from these periodic windows and creates KPI reports on end-product quality. After processing each product batch, you can measure the overall product quality and lower your operational expenditure through maximized run volume.

Automate the analysis of a fleet of devices

You run analytics (algorithm, data science or ML for KPI) every 15 minutes on data generated by 100s of devices, with each analytics cycle generating and storing state for next analytics run. For each of your analyses, you want to use only that data received within a specified time window. With AWS IoT Analytics you can orchestrate your analyses and create the KPI and report for each run then store the data for future analytics.

Automate anomaly detection

AWS IoT Analytics enables you to automate your anomaly detection workflow that you manually have to run every 15 minutes on new data which has arrived in a data store. You can also automate a dashboard that shows device usage and top users within a specified period of time.

Predict industrial process outcomes

You have industrial production lines. Using the data sent to AWS IoT Analytics, including available process measurements, you can operationalize the analytical workflows to predict process outcomes. Data for the model can be arranged in an M x N matrix where each row contains data from various time points where laboratory samples are taken. AWS IoT Analytics helps you operationalize your analytical workflow by creating delta windows and using your data science tools to create KPIs and save the state of the measurement devices.

How to Proceed

Note
This section includes informaton about how to build your own Docker container. There is a security risk if you re-use Docker containers built by third parties: these containers can execute arbitrary code with your user permissions. Make sure you trust the author of any third-party container before using it.

Here are the steps you would take to set up periodic data analysis on data which has arrived since the last analysis was performed:

1. Create a Docker container that contains your data application plus any required libraries or other dependencies.

   The IotAnalytics Jupyter extension provides a containerization API to assist in the containerization process. You can also run images of your own creation in which you create or assemble your application toolset to perform the desired data analysis or computation. AWS IoT Analytics enables you to define the source of the input data to the containerized application and the destination for the output data of the Docker container by means of variables. (Custom Docker Container Input/Output Variables (p. 70) contains more information about using variables with a custom container.)
2. Upload the container to an Amazon ECR registry.
3. Create a data store to receive and store messages (data) from devices (iotanalytics:CreateDatastore (p. 150))
4. Create a channel where the messages are sent (iotanalytics:CreateChannel (p. 138)).
5. Create a pipeline to connect the channel to the data store (iotanalytics:CreatePipeline (p. 154)).
6. Create an IAM role that grants permission to send message data to an AWS IoT Analytics channel (iam:CreateRole).
7. Create an IoT rule that uses a SQL query to connect a channel to the source of the message data (iot:CreateTopicRule field `topicRulePayload:actions:iotAnalytics`). When a device sends a message with the appropriate topic via MQTT, it is routed to your channel. Or, you can use iotanalytics:BatchPutMessage (p. 135) to send messages directly into a channel from a device capable of using the AWS SDK or CLI.
8. Create a SQL data set whose creation is triggered by a time schedule (iotanalytics:CreateDataset (p. 141), field `trigger:schedule`) denoting the desired interval of N minutes/hours for data collection.

   You specify a query action with the appropriate SQL query to select the data fields you want (field `"actions:queryAction:sqlQuery`).

   You also specify a pre-filter to be applied to the message data to help limit the messages to those which have arrived since the last execution of the action.

(Field `actions:queryAction:filters:deltaTime:timeExpression` gives an expression by which the time of a message may be determined, while field `actions:queryAction:filters:deltaTime:offsetSeconds` specifies possible latency in the arrival of a message.)

The pre-filter, along with the trigger schedule, determines your "delta window". Each new SQL data set is created using messages received since the last time the SQL data set was created. (What about the first time the SQL data set is created? An estimate of when the "last time" the data set would have been created is made based on the schedule and the pre-filter.)

9. Create another data set that is triggered by the creation of the first (CreateDataset (p. 141) field `trigger:dataset`). For this data set, you specify a "container action" (field `actions:containerAction`) that points to, and gives information needed to run, the Docker container you created in the first step. Here you also specify:
   - The ARN of the docker container stored in your account (`image`).
   - The ARN of the role which gives permission to the system to access needed resources in order to run the container action (`executionRoleArn`).
   - The configuration of the resource that executes the container action (`resourceConfiguration`).
   - The type of the compute resource used to execute the container action (`computeType` with possible values: ACU_1 [vCPU=4, memory=16GiB] or ACU_2 [vCPU=8, memory=32GiB]).
   - The size (in GB) of the persistent storage available to the resource instance used to execute the container action (`volumeSizeInGB`).
   - The values of variables used within the context of the execution of the application (basically, parameters passed to the application) (`variables`) .

     These variables are replaced at the time a container is executed. This enables you to run the same container with different variables (parameters) which are supplied at the time the data set content is created.

     The IotAnalytics Jupyter extension simplifies this process by automatically recognizing the variables in a notebook and making them available as part of the containerization process. You can choose the recognized variables or add custom variables of your own. Before it runs a container, the system replaces each of these variables with the value current at the time of execution.
   - One of the variables is the name of the data set whose latest content is used as input to the application (this is the name of the data set you created in the previous step) (`datasetContentVersionValue:datasetName`).
   - Another variable specifies an output file URI where the output data set's content is stored (usually the URI of a file in an S3 bucket (`outputFileUriValue:filename`).

With the SQL query and delta window to generate the data set, and the container with your application, AWS IoT Analytics creates a scheduled production data set that runs at the interval you specify on data from the delta window, producing your desired output and sending notifications.

You can pause your production data set application and resume it whenever you choose to do so. When you resume your production data set application, AWS IoT Analytics, by default, catches up all the data that has arrived since last execution, but hasn't been analyzed yet. You can also configure how you want to resume your production data set job besides the default catch up option. You can catch up in multiple fixed-length windows (as defined by your delta window length) by performing a series of consecutive runs. Alternatively, you can resume your Production data set application by capturing only the newly arrived data that fits within the specified size of your delta window. This option skips any data beyond the length of your delta window.

Please note the following limitations when creating/defining a data set which is triggered by the creation of another data set:

- Only container data sets can be triggered by SQL data sets.

- A SQL data set can trigger at most 10 container data sets.

The following errors may be returned when creating a container data set which is triggered by a SQL data set:

- "Triggering dataset can only be added on a container dataset"
- "There can only be one triggering dataset"

  This error occurs if you attempt to define a container data set which is triggered by two different SQL data sets.
- "The triggering data set <dataset-name> cannot be triggered by a container dataset"

  This error occurs if you attempt to define a container data set which is triggered by another container data set.
- "<N> datasets are already dependent on <dataset-name> dataset"

  This error occurs if you attempt to define another container data set which is triggered by a SQL data set which already triggers 10 container data sets.
- "Exactly one trigger type should be provided"

  This error occurs is you attempt to define a data set which is triggered by both a schedule trigger and a data set trigger.

Custom Docker Container Input/Output Variables

This section demonstrates how the program which is run by your custom Docker image may read input variables and upload its output.

Params File

The input variables and the destinations to which you want to upload output are stored in a JSON file located at `/opt/ml/input/data/iotanalytics/params` on the instance that executes your docker image. Here is an example of the contents of that file:

```
{
   "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"
   }
}
```

In addition to the name and version ID of your dataset, the `Variables` section contains the variables specified in the `iotanalytics:CreateDataset` invocation– in this example, a variable `example_var` was given the value `hello world!`. A custom output URI was also provided in the `custom_output` variable. The `OutputUris` field contains default locations to which the container can upload its output– in this example, default output URIs were provided for both ipynb and html output.

Input Variables

The program launched by your Docker image can read variables from the `params` file. Here is an example program which opens the `params` file, parses it, and prints the value of the `example_var` variable:

```
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)
```

Uploading Output

The program launched by your Docker image may also store its output in an AWS S3 location. The output must be loaded with a "bucket-owner-full-control" access control list. The access list grants the AWS IoT Analytics service control over the uploaded output. In this example we extend the previous one to upload the contents of `example_var` to the S3 location defined by `custom_output` in the `params` file:

```
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)
```

Permissions

You must create two roles. One role grants permission to launch a SageMaker instance in order to containerize a notebook. Another role is needed to execute a container.

You can create the first role automatically or manually. If you create your new Amazon SageMaker instance with the AWS IoT Analytics console, you are given the option to automatically create a new role which grants all privileges necessary to execute SageMaker instances and containerize notebooks. Or, you may create a role with these privileges manually. To do this, create a role with the "AmazonSageMakerFullAccess" policy attached and add the following policy:

```
{
  "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/*"
    }
  ]
}
```

You must manually create the second role which grants permission to execute a container. (You must do this even if you used the AWS IoT Analytics console to create the first role automatically.) Create a role with the following policy and trust policy attached:

```
{
    "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": "*"
        }
    ]
}
```

Trust policy:

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

The CreateDataset API

Creates a data set. A data set stores data retrieved from a data store by applying a “queryAction” (a SQL query) or a “containerAction” (executing a containerized application). This operation creates the skeleton of a data set. The data set can be populated manually by calling “CreateDatasetContent” or automatically according to a “trigger” you specify.

CLI Synopsis:

```
aws iotanalytics  create-dataset
 --dataset-name <value>
 --actions <value>
[--triggers <value>]
[--content-delivery-rules <value>]
[--retention-period <value>]
[--versioning-configuration <value>]
[--tags <value>]
[--cli-input-json <value>]
[--generate-cli-skeleton]
```

cli-input-json format:

```
{
  "datasetName": "string",
  "actions": [
    {
      "actionName": "string",
      "queryAction": {
        "sqlQuery": "string",
        "filters": [
          {
            "deltaTime": {
              "offsetSeconds": "integer",
              "timeExpression": "string"
            }
          }
        ]
      },
```

```
    "containerAction": {
      "image": "string",
      "executionRoleArn": "string",
      "resourceConfiguration": {
        "computeType": "string",
        "volumeSizeInGB": "integer"
      },
      "variables": [
        {
          "name": "string",
          "stringValue": "string",
          "doubleValue": "double",
          "datasetContentVersionValue": {
            "datasetName": "string"
          },
          "outputFileUriValue": {
            "fileName": "string"
          }
        }
      ]
    }
  }
],
"triggers": [
  {
    "schedule": {
      "expression": "string"
    },
    "dataset": {
      "name": "string"
    }
  }
],
"contentDeliveryRules": [
  {
    "entryName": "string",
    "destination": {
      "iotEventsDestinationConfiguration": {
        "inputName": "string",
        "roleArn": "string"
      },
      "s3DestinationConfiguration": {
        "bucket": "string",
        "key": "string",
        "glueConfiguration": {
          "tableName": "string",
          "databaseName": "string"
        },
        "roleArn": "string"
      }
    }
  }
],
"retentionPeriod": {
  "unlimited": "boolean",
  "numberOfDays": "integer"
},
"versioningConfiguration": {
  "unlimited": "boolean",
  "maxVersions": "integer"
},
"tags": [
  {
    "key": "string",
    "value": "string"
  }
```

```
  ]
}
```

fields:

- datasetName

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the data set.
- actions

  type: list member: DatasetAction

  A list of actions that create the data set contents.
- actionName

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the data set action by which data set contents are automatically created.
- queryAction

  type: SqlQueryDatasetAction

  An “SqlQueryDatasetAction” object that uses an SQL query to automatically create data set contents.
- sqlQuery

  type: string

  A SQL query string.
- filters

  type: list member: QueryFilter

  Pre-filters applied to message data.
- deltaTime

  type: DeltaTime

  Used to limit data to that which has arrived since the last execution of the action.
- offsetSeconds

  type: integer java class: java.lang.Integer

  The number of seconds of estimated “in flight” lag time of message data. When you create data set contents using message data from a specified time frame, some message data may still be “in flight” when processing begins, and so will not arrive in time to be processed. Use this field to make allowances for the “in flight” time of your message data, so that data not processed from a previous time frame will be included with the next time frame. Without this, missed message data would be excluded from processing during the next time frame as well, because its timestamp places it within the previous time frame.
- timeExpression

  type: string

  An expression by which the time of the message data may be determined. This may be the name of a timestamp field, or a SQL expression which is used to derive the time the message data was generated.
- containerAction

type: ContainerDatasetAction

Information which enables the system to run a containerized application in order to create the data set contents. The application must be in a Docker container along with any needed support libraries.

- image

  type: string; (length- max:255)

  The ARN of the Docker container stored in your account. The Docker container contains an application and needed support libraries and is used to generate data set contents.
- executionRoleArn

  type: string; (length- max:2048 min:20)

  The ARN of the role which gives permission to the system to access needed resources in order to run the "containerAction". This includes, at minimum, permission to retrieve the data set contents which are the input to the containerized application.
- resourceConfiguration

  type: ResourceConfiguration

  Configuration of the resource which executes the "containerAction".
- computeType

  type: string

  The type of the compute resource used to execute the "containerAction". Possible values are: ACU_1 (vCPU=4, memory=16GiB) or ACU_2 (vCPU=8, memory=32GiB). enum: ACU_1 | ACU_2
- volumeSizeInGB

  type: integer range- max:50 min:1

  The size (in GB) of the persistent storage available to the resource instance used to execute the "containerAction" (min: 1, max: 50).
- variables

  type: list member: Variable

  The values of variables used within the context of the execution of the containerized application (basically, parameters passed to the application). Each variable must have a name and a value given by one of "stringValue", "datasetContentVersionValue", or "outputFileUriValue".
- name

  type: string; (length- max:256 min:1)

  The name of the variable.
- stringValue

  type: string; (length- max:1024 min:0)

  The value of the variable as a string.
- datasetContentVersionValue

  type: DatasetContentVersionValue

  The value of the variable as a structure that specifies a data set content version.
- datasetName

type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

The name of the data set whose latest contents are used as input to the notebook or application.

- outputFileUriValue

  type: OutputFileUriValue

  The value of the variable as a structure that specifies an output file URI.
- fileName

  type: string; (pattern: [w.-]{1,255})

  The URI of the location where data set contents are stored, usually the URI of a file in an S3 bucket.
- triggers

  type: list member: DatasetTrigger

  A list of triggers. A trigger causes data set contents to be populated at a specified time interval or when another data set's contents are created. The list of triggers can be empty or contain up to five DataSetTrigger objects.
- schedule

  type: Schedule

  The "Schedule" when the trigger is initiated.
- expression

  type: string

  The expression that defines when to trigger an update. For more information, see Schedule Expressions for Rules in the Amazon CloudWatch Events User Guide.
- dataset

  type: TriggeringDataset

  The data set whose content creation triggers the creation of this data set's contents.
- name

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the data set whose content generation triggers the new data set content generation.
- contentDeliveryRules

  type: list member: DatasetContentDeliveryRule

  When data set contents are created they are delivered to destinations specified here.
- entryName

  type: string

  The name of the data set content delivery rules entry.
- destination

  type: DatasetContentDeliveryDestination

  The destination to which data set contents are delivered.
- iotEventsDestinationConfiguration

type: IotEventsDestinationConfiguration

Configuration information for delivery of data set contents to AWS IoT Events.

- inputName

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z][a-zA-Z0-9_]*$)

  The name of the AWS IoT Events input to which data set contents are delivered.
- roleArn

  type: string; (length- max:2048 min:20)

  The ARN of the role which grants AWS IoT Analytics permission to deliver data set contents to an AWS IoT Events input.
- s3DestinationConfiguration

  type: S3DestinationConfiguration

  Configuration information for delivery of data set contents to Amazon S3.
- bucket

  type: string; (length- max:255 min:3); (pattern: ^[a-zA-Z0-9.-_]*$)

  The name of the Amazon S3 bucket to which data set contents are delivered.
- key

  type: string; (length- max:255 min:1); (pattern: ^[a-zA-Z0-9!_.*'()/{}:-]*$)

  The key of the data set contents object. Each object in an Amazon S3 bucket has a key that is its unique identifier within the bucket (each object in a bucket has exactly one key).
- glueConfiguration

  type: GlueConfiguration

  Configuration information for coordination with the AWS Glue ETL (extract, transform and load) service.
- tableName

  type: string; (length- max:150 min:1); (pattern: [u0020-uD7FFuE000-uFFFDuD800uDC00-uDBFFuDFFFt]*)

  The name of the table in your AWS Glue Data Catalog which is used to perform the ETL (extract, transform and load) operations. (An AWS Glue Data Catalog table contains partitioned data and descriptions of data sources and targets.)
- databaseName

  type: string; (length- max:150 min:1); (pattern: [u0020-uD7FFuE000-uFFFDuD800uDC00-uDBFFuDFFFt]*)

  The name of the database in your AWS Glue Data Catalog in which the table is located. (An AWS Glue Data Catalog database contains Glue Data tables.)
- roleArn

  type: string; (length- max:2048 min:20)

  The ARN of the role which grants AWS IoT Analytics permission to interact with your Amazon S3 and AWS Glue resources.

- retentionPeriod

  type: RetentionPeriod

  [Optional] How long, in days, versions of data set contents are kept for the data set. If not specified or set to null, versions of data set contents are retained for at most 90 days. The number of versions of data set contents retained is determined by the versioningConfiguration parameter. (For more information, see https://docs.aws.amazon.com/iotanalytics/latest/userguide/getting-started.html#aws-iot-analytics-dataset-versions)
- unlimited

  type: boolean

  If true, message data is kept indefinitely.
- numberOfDays

  type: integer java class: java.lang.Integer range- min:1

  The number of days that message data is kept. The "unlimited" parameter must be false.
- versioningConfiguration

  type: VersioningConfiguration

  [Optional] How many versions of data set contents are kept. If not specified or set to null, only the latest version plus the latest succeeded version (if they are different) are kept for the time period specified by the "retentionPeriod" parameter. (For more information, see https://docs.aws.amazon.com/iotanalytics/latest/userguide/getting-started.html#aws-iot-analytics-dataset-versions)
- unlimited

  type: boolean

  If true, unlimited versions of data set contents will be kept.
- maxVersions

  type: integer java class: java.lang.Integer range- max:1000 min:1

  How many versions of data set contents will be kept. The "unlimited" parameter must be false.
- tags

  type: list member: Tag

  Metadata which can be used to manage the data set.
- key

  type: string; (length- max:128 min:1)

  The tag's key.
- value

  type: string; (length- max:128 min:1)

  The tag's value.

Output:

```
{
```

```
  "datasetName": "string",
  "datasetArn": "string",
  "retentionPeriod": {
    "unlimited": "boolean",
    "numberOfDays": "integer"
  }
}
```

fields:

- datasetName

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the data set.
- datasetArn

  type: string

  The ARN of the data set.
- retentionPeriod

  type: RetentionPeriod

  How long, in days, data set contents are kept for the data set.
- unlimited

  type: boolean

  If true, message data is kept indefinitely.
- numberOfDays

  type: integer java class: java.lang.Integer range- min:1

  The number of days that message data is kept. The “unlimited” parameter must be false.

Errors:

- InvalidRequestException

  The request was not valid.

  HTTP response code: 400
- ResourceAlreadyExistsException

  A resource with the same name already exists.

  HTTP response code: 409
- InternalFailureException

  There was an internal failure.

  HTTP response code: 500
- ServiceUnavailableException

  The service is temporarily unavailable.

  HTTP response code: 503

- ThrottlingException

  The request was denied due to request throttling.

  HTTP response code: 429
- LimitExceededException

  The command caused an internal limit to be exceeded.

  HTTP response code: 410

Examples Using CreateDataset

Example 1 – Creating a SQL data set (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);
```

Output on success:

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

Example 2 – Creating a SQL Dataset with a delta window (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);
```

Output on success:

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

Example 3 – Creating a container data set with its own schedule trigger (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);
```

Output on success:

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

Example 4 – Creating a container data set with a SQL data set as a trigger (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);
```

Output on success:

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

Example 5 – Creating a SQL dataset (CLI):

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

Output on success:

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

Example 6 – Creating a SQL dataset with a Delta Window (CLI):

Delta windows are a series of user-defined, non-overlapping and contiguous time intervals. Delta windows enable you to create data set content with, and perform analysis on, new data that has arrived in the data store since the last analysis. You create a delta window by setting the `deltaTime` in the `filters` portion of a `queryAction` of a data set (CreateDataset (p. 141)). Usually, you'll want to create the data set content automatically by also setting up a time interval trigger (`triggers:schedule:expression`). Basically, this enables you to filter messages that have arrived during a specific time window, so the data contained in messages from previous time windows doesn't get counted twice.

In this example, we create a new data set that automatically creates new data set content every 15 minutes using only that data which has arrived since the last time. We specify a 3 minute (180 second) `deltaTime` offset that allows for a delay of 3 minutes for messages to arrive in the specified data store. So, if data set content is created at 10:30AM, the data used (included in the data set content) would be that with timestamps between 10:12AM and 10:27AM (that is 10:30AM - 15 minutes - 3 minutes to 10:30AM - 3 minutes).

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

where the file `delta-window.json` contains:

```
{
  "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 * * * ? *)"
      }
    }
  ]
}
```

Output on success:

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

Containerizing A Notebook

Note
This section includes informaton about how to build a Docker container using a Jupyter notebook. There is a security risk if you re-use notebooks built by third parties: included containers can execute arbitrary code with your user permissions. In addition, the HTML generated by the notebook can be displayed in the AWS IoT Analytics console, providing a potential attack vector on the computer displaying the HTML. Make sure you trust the author of any third-party notebook before using it.

One option to perform advanced analytical functions is to use a Jupyter Notebook. Jupyter Notebooks provide powerful data science tools that can perform machine learning and a range of statistical analyses. For more information, see Notebook Templates (p. 30). (Note that we do not currently support containerization inside JupyterLab.) You can package your Jupyter Notebooks and libraries into a container that periodically runs on a new batch of data as it is received by AWS IoT Analytics during a delta time window you define. You can schedule an analysis job that uses the container and the new, segmented data captured within the specified time window, then stores the job's output for future scheduled analytics.

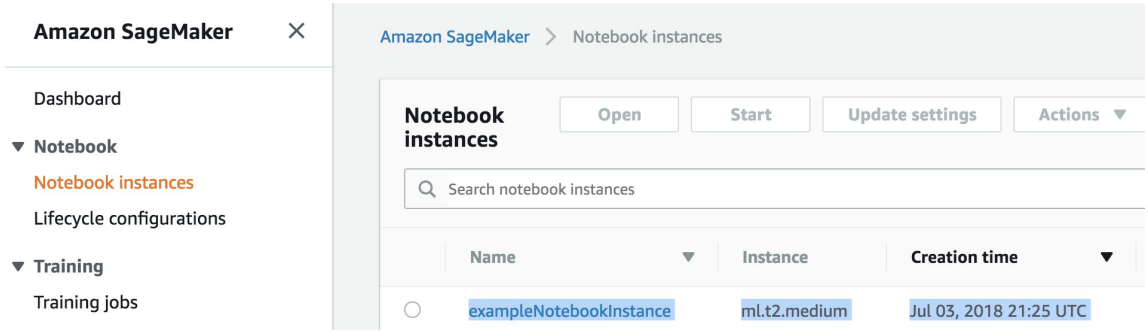
If you have created a SageMaker Instance using the AWS Iot Analytics console after August 23, 2018, then the installation of the containerization extension has been done for you automatically and you can begin creating a containerized image (p. 87). Otherwise, follow the steps listed in this section to enable notebook containerization on your SageMaker instance. In what follows, you modify your SageMaker Execution Role to allow you to upload the container image to AWS ECR and you install the containerization extension.

Enable Containerization Of Notebook Instances Not Created Via AWS Iot Analytics Console

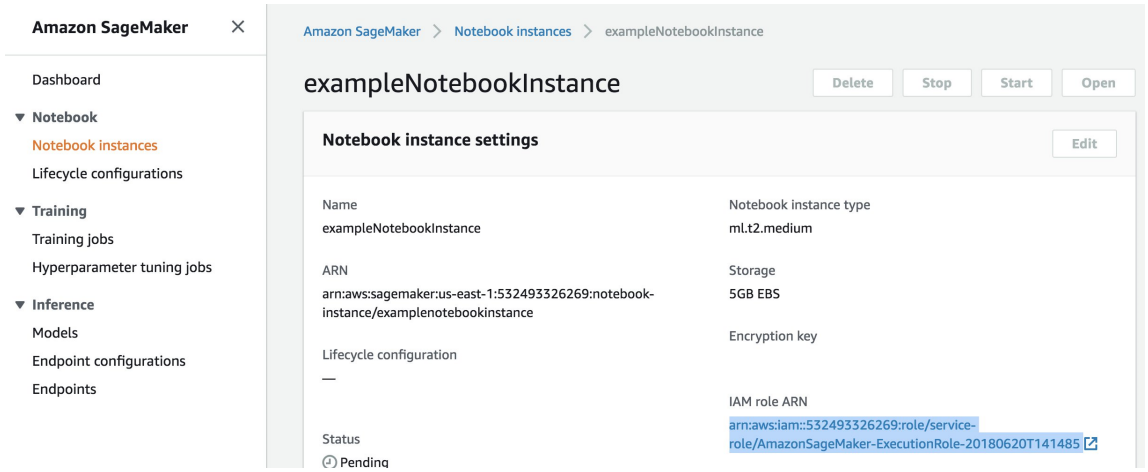
We recommend that you create a new SageMaker instance via the AWS Iot Analytics console instead of following these steps. New instances automatically support containerization.

If you restart your SageMaker instance after enabling containerization as shown here, you won't have to re-add the IAM roles and policies, but you must re-install the extension, as shown in the final step.

1. To grant your notebook instance access to AWS ECS, select your SageMaker instance on the Amazon SageMaker page:

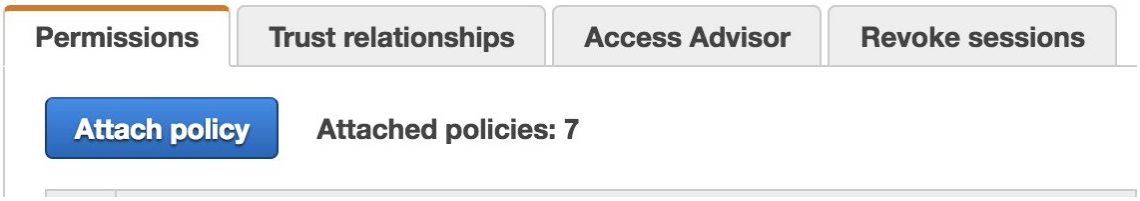


2. Under **IAM role ARN** choose the SageMaker Execution Role:



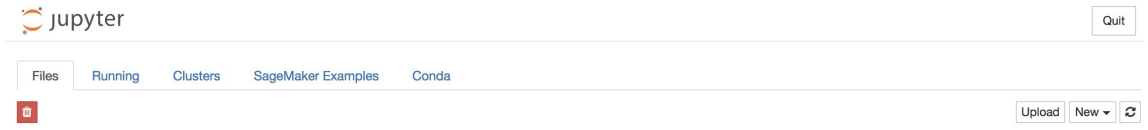
3. Choose **Attach Policy**, then define and attach the policy shown in Permissions (p. 71).

   If the "AmazonSageMakerFullAccess" policy is not already attached, attach it as well:



You also must download the containerization code from S3 and install it on your notebook instance. The first step is to access the SageMaker instance's terminal.

1. Inside Jupyter, choose **New**:



2. In the dropdown menu that appears, select **Terminal**:

3. Inside the terminal, enter the following commands to download the code, unzip it, and install it. Note that these commands kill any processes being run by your notebooks on this SageMaker instance.


```
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
```

Wait for a minute or two for the extension to be validated and installed.

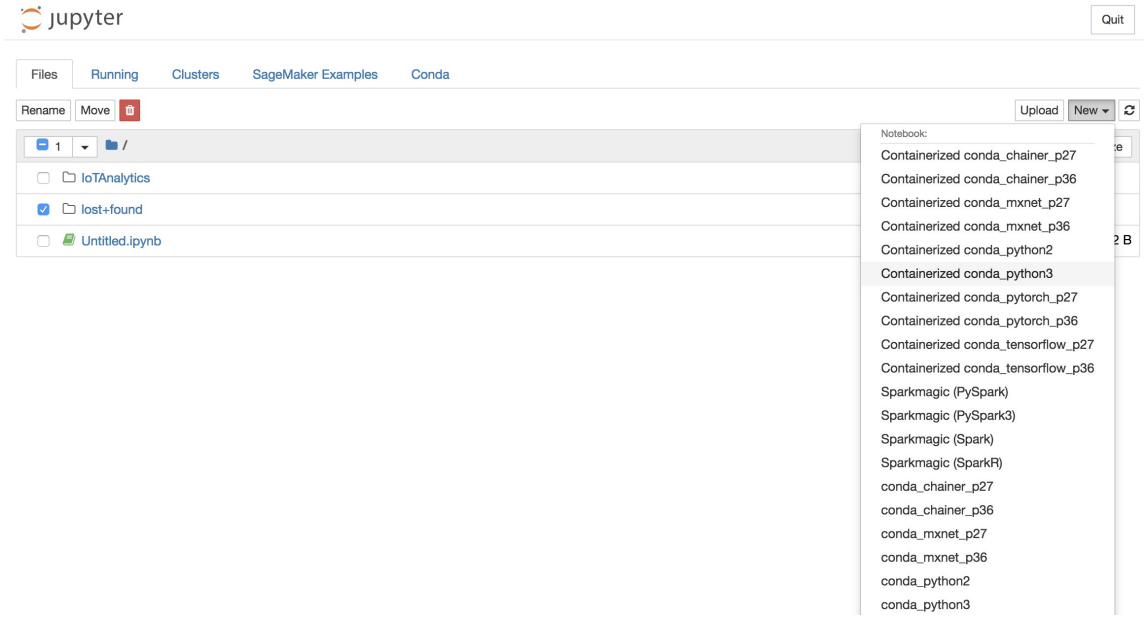
Update Your Notebook Containerization Extension

If you created your SageMaker Instance via the AWS Iot Analytics console after August 23, 2018, then the containerization extension was installed automatically. You can update the extension by restarting your instance from the SageMaker Console. If you installed the extension manually, then you may update it by re-running the terminal commands listed in Enable Containerization Of Notebook Instances Not Created Via AWS Iot Analytics Console.

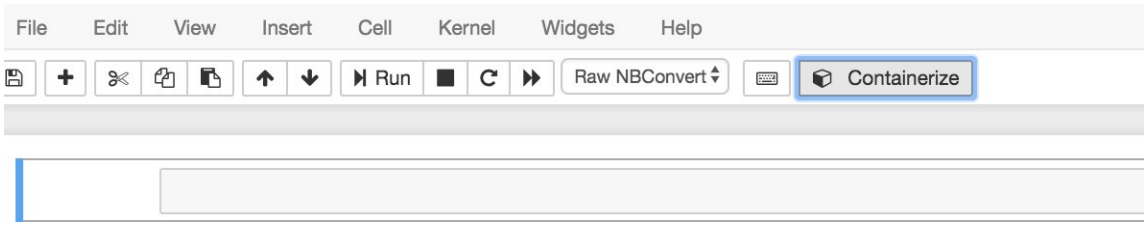
Create A Containerized Image

In this section we show the steps necessary to containerize a notebook. To begin, go to your Jupyter Notebook to create a notebook with a containerized kernel.

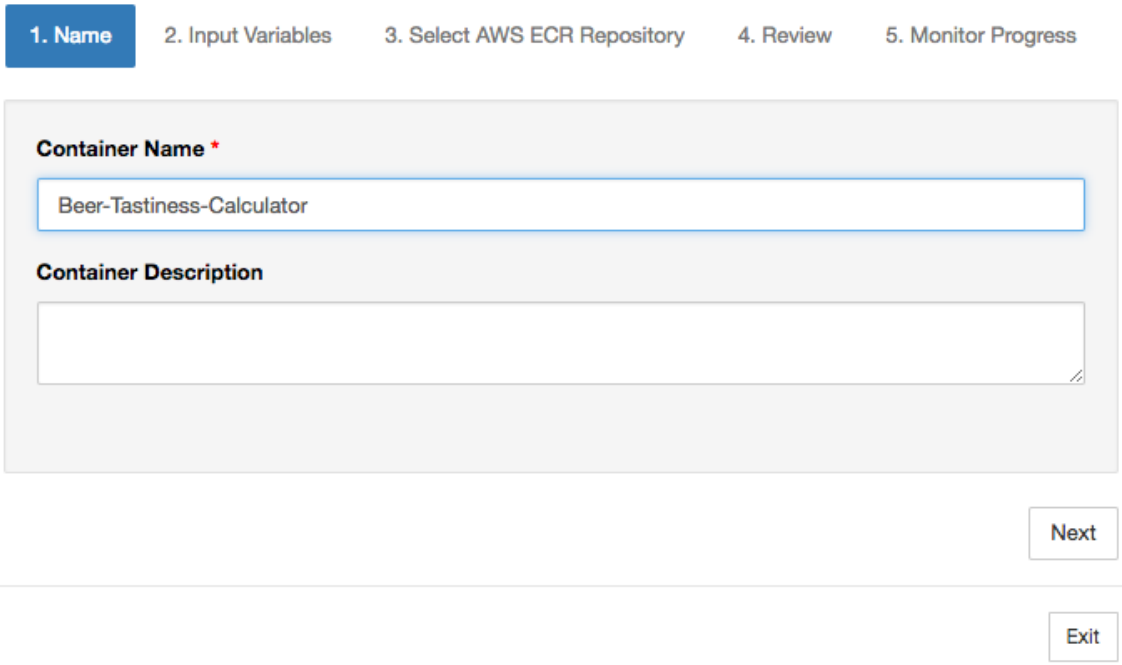
1. In your Jupyter Notebook, choose **New**, then choose the kernel type you want from the dropdown list. (The kernel type should start with “Containerized” and end with whatever kernel you would have otherwise selected. For example, if you just want a plain python3 environment like “conda_python3”, choose “Containerized conda_python3”):



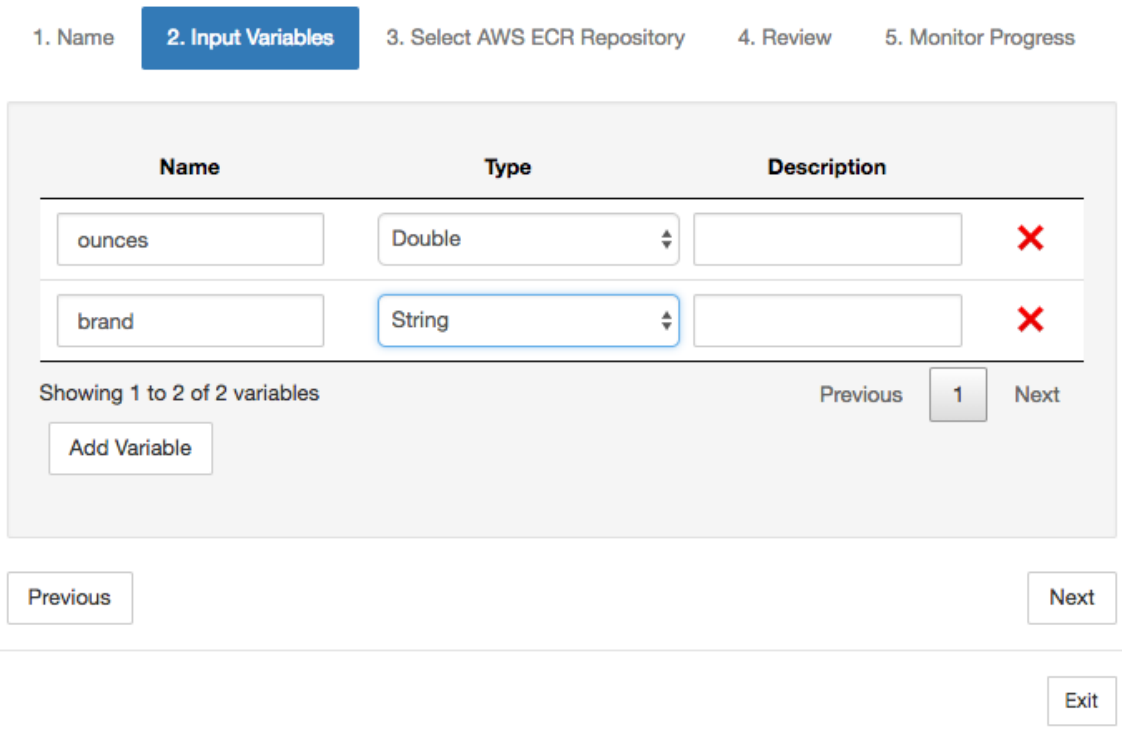
2. After you have completed work on your notebook and you want to containerize it, choose the **containerize** button:



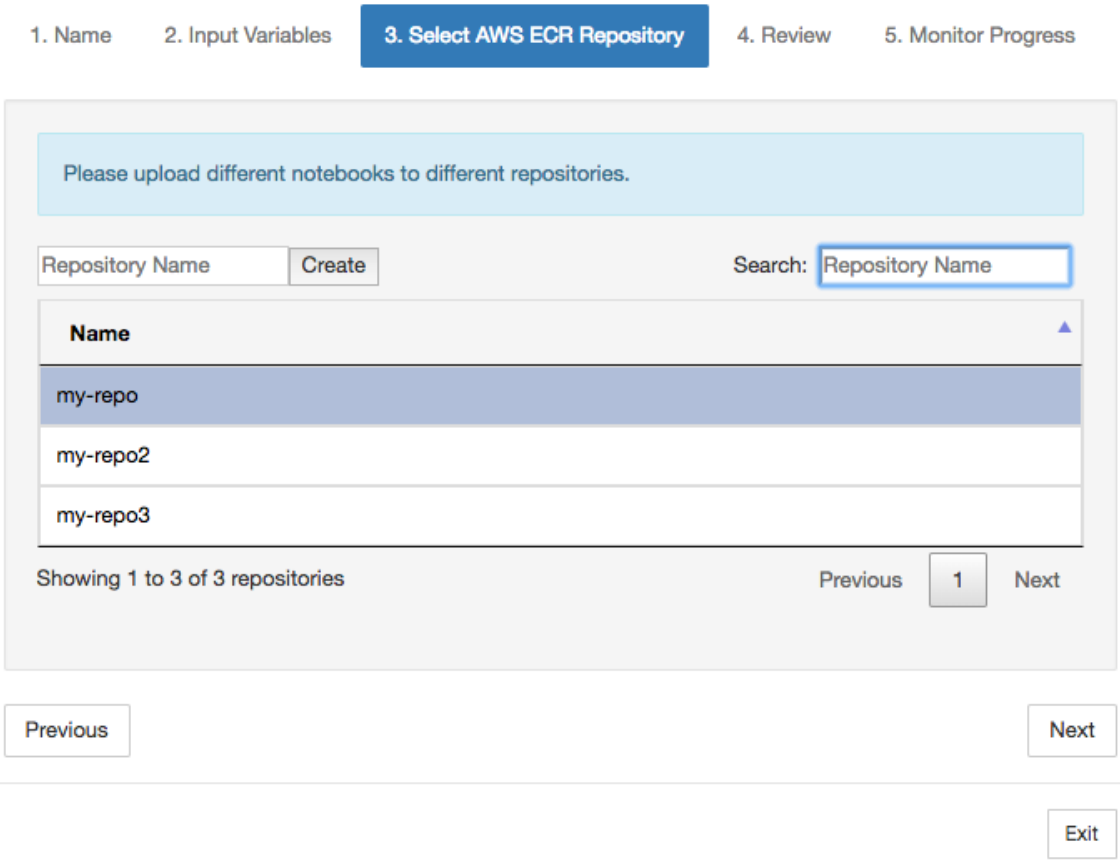
3. Enter a name for the containerized notebook. You may also enter an optional description:



4. Specify the **Input Variables** (parameters) that your notebook should be invoked with. You can select the input variables that are automatically detected from your notebook or define custom variables. (Note that input variables are only detected if you have previously executed your notebook.) For each input variable choose a type. You can also enter an optional description of the input variable:

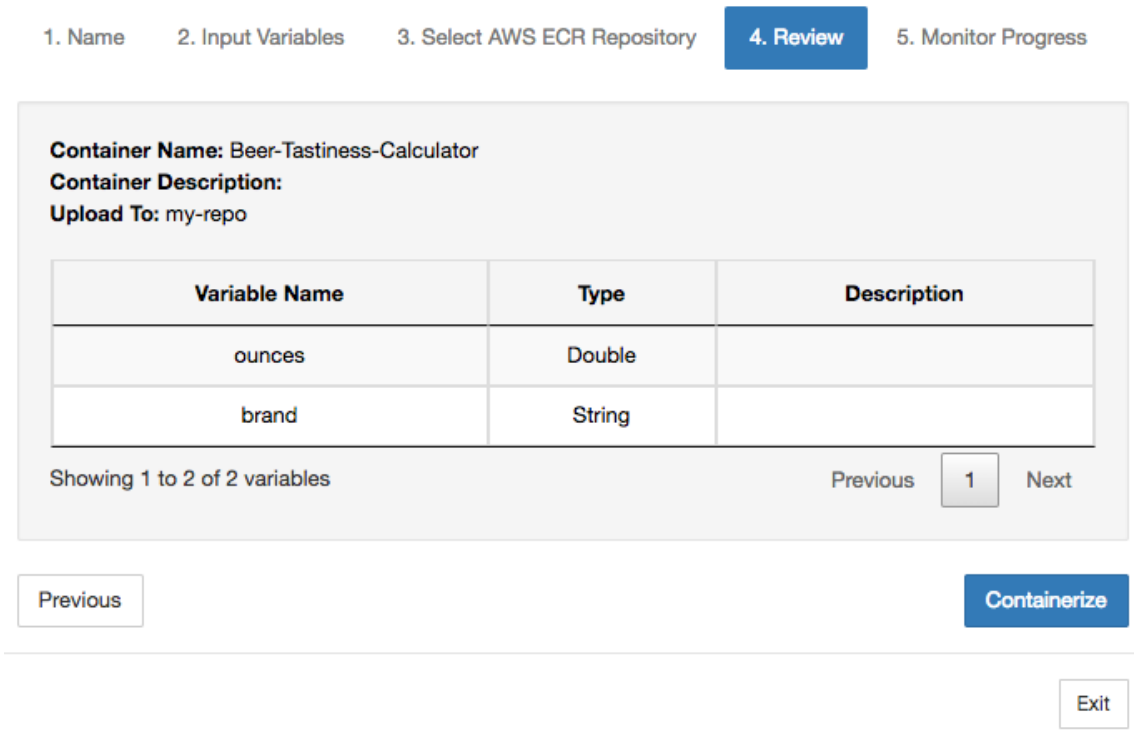


5. Choose the AWS ECR repository where the image created from the notebook should be uploaded:

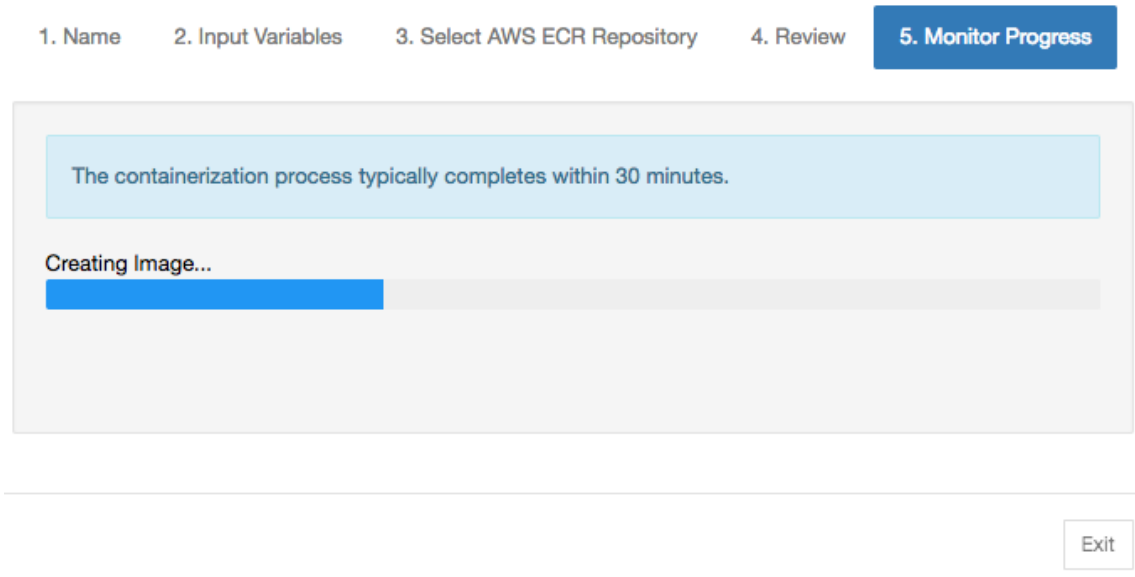


6. You are presented with with an overview summarizing your input. Note that after you have started the containerization process you cannot cancel it. The process may last for over an hour.

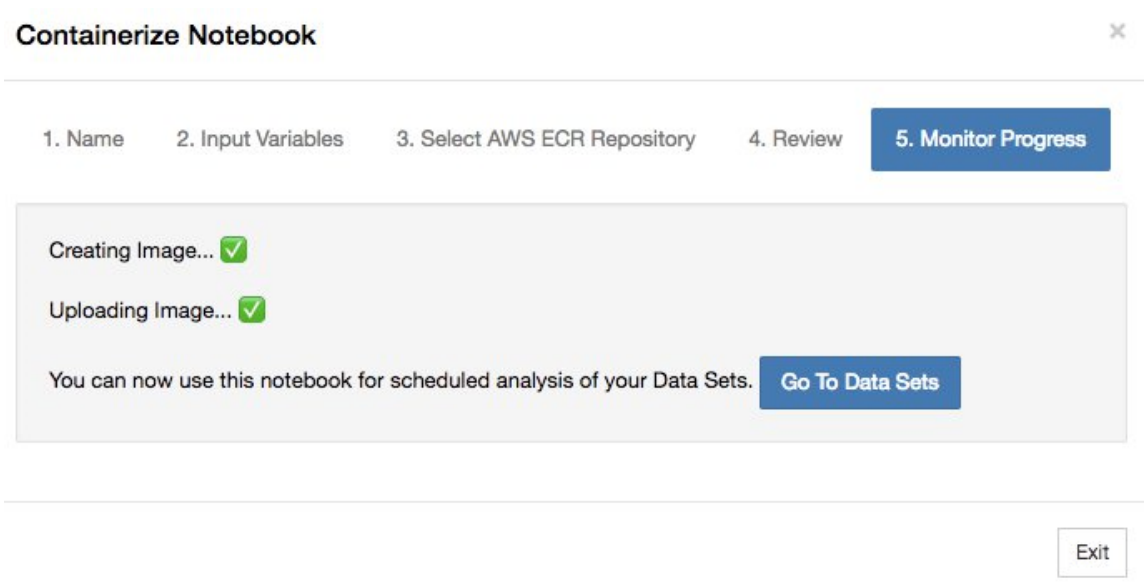
   Choose **containerize** to begin the containerization process.



7. The next page shows the progress:



8. If you accidentally close your browser, you can monitor the status of the containerization process from the **Notebooks** section of the AWS IoT Analytics console.
9. After the process is complete, the containerized image is stored on AWS ECR ready for use:



Using Your Own Custom Container for Analysis

Note
This section includes informaton about how to build a Docker container using a Jupyter notebook. There is a security risk if you re-use notebooks built by third parties: included containers can execute arbitrary code with your user permissions. In addition, the HTML generated by the notebook can be displayed in the AWS IoT Analytics console, providing a potential attack vector on the computer displaying the HTML. Make sure you trust the author of any third-party notebook before using it.

You can create your own custom container and run it with the AWS IoT Analytics service. To do so, you setup a Docker image and upload it to Amazon ECR, then set up a data set to run a container action. This section gives an example of the process using Octave.

This tutorial assumes that you have:

- Octave installed on your local computer
- a Docker account set up on your local computer
- an AWS account with ECR/IoT Analytics access

Step 1: Set up a Docker image

There are three main files you need for this tutorial. Their names and contents are here:

- `Dockerfile` - The initial setup for Docker's containerization process.

```
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` - Parses JSON from AWS IoT Analytics, runs the Octave script, and uploads artifacts to S3.

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

# Parse the JSON from AWS 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` - A simple Octave script which calculates the moment based on an input/output file and a specified order.

```
#!/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. Download the contents of each file. Create a new directory and place all the files in it. Then `cd` to that directory.
2. Run:

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

3. You should see a new image in your Docker repo. Verify it by running:

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

Step 2: Upload the Docker image to an ECR repository

1. Create a new repository in ECR:

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

2. Get the login to your Docker environment:

```
aws ecr get-login
```

3. Copy the output and run it. The output should look something like:

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

4. Tag the image you created with the ECR Repository Tag:

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

5. Push the image to ECR

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

Step 3: Upload your sample data to an S3 bucket

1. Download the following to file “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. Create a new S3 Bucket called “octave-sample-data-<your-account-id>”.
3. Upload the file “input.txt” to the S3 Bucket you just created. You should now have a bucket named `octave-sample-data-<your-aws-account-id>` containing the file `input.txt`.

Step 4: Create a container execution role

1. Download the following to a file named “role1.json”:

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

2. Create a new role that gives access permissions to Amazon SageMaker and AWS IoT Analytics, using the file “role1.json” you just downloaded:

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

3. Download the following to a file named “policy1.json” and replace “<your-account-id>” with your account id (see the second ARN under `Statement:Resource`):

```
{
  "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. Create an IAM policy, using the file “policy1.json” you just downloaded:

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

5. Attach the policy to the role:

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

Step 5: Create a data set with a container action

1. Download the following to a file named “cli-input.json” and replace all instances of “<your-account-id>” and “<region>” with the appropriate values:

```
  {
    "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. Create a data set using the file “cli-input.json” you just downloaded and edited:

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

Step 6: Invoke data set content generation

1. Run:

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

Step 7: Get data set content

1. Run:

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

2. You may need to wait several minutes until the `DatasetContentState` is `SUCCEEDED`.

Step 8: Print the output on Octave

1. Use the Octave shell to print the output from the container by running:

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

SQL Expressions in AWS IoT Analytics

Data sets are generated using SQL expressions on data in a data store. AWS IoT Analytics uses the same SQL queries, functions and operators as Amazon Athena.

AWS IoT Analytics supports a subset of ANSI standard SQL syntax as follows:

```
SELECT [ ALL | DISTINCT ] select_expression [, ...]
[ FROM from_item [, ...] ]
[ 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 ] ]
```

However, AWS IoT Analytics does not support `JOIN` or `WITH` clauses.

Amazon Athena and AWS IoT Analytics SQL functionality are based on Presto 0.172, so they support the following functions:

- Logical Operators
- Comparison Functions and Operators
- Conditional Expressions
- Conversion Functions
- Mathematical Functions and Operators
- Bitwise Functions
- Decimal Functions and Operators
- String Functions and Operators
- Binary Functions
- Date and Time Functions and Operators
- Regular Expression Functions
- JSON Functions and Operators
- URL Functions
- Aggregate Functions
- Window Functions
- Color Functions
- Array Functions and Operators
- Map Functions and Operators
- Lambda Expressions and Functions
- Teradata Functions

However, AWS IoT Analytics and Amazon Athena do not support:

- User-defined functions (UDFs or UDAFs).

- Stored procedures.
- Some data types.
- CREATE TABLE AS SELECT statements.
- INSERT INTO statements.
- Prepared statements. You cannot run EXECUTE with USING.
- CREATE TABLE LIKE.
- DESCRIBE INPUT and DESCRIBE OUTPUT.
- EXPLAIN statements.
- Federated connectors.

These are the supported data types:

- primitive_type
  - TINYINT
  - SMALLINT
  - INT
  - BIGINT
  - BOOLEAN
  - DOUBLE
  - FLOAT
  - STRING
  - TIMESTAMP
  - DECIMAL[(precision, scale)]
  - DATE
  - CHAR (fixed-length character data with a specified length)
  - VARCHAR (variable-length character data with a specified length)
- array_type
  - ARRAY<data_type>
- map_type
  - MAP<primitive_type, data_type>
- struct_type
  - STRUCT<col_name:data_type[COMMENT col_comment][,…]>

AWS IoT Analytics entity names (channel, data set, data store, and pipeline names) must begin with a lower-case letter and can contain only lower-case letters, digits and underscores.

AWS IoT Analytics and Athena table, database and column names must contain only lower-case letters, digits and underscore. Use backticks to enclose table or column names that begin with an underscore. For example:

```
CREATE_TABLE `_myunderscoretable`(
  `_id` string,
  `_index` string,
  ...
```

Enclose table names that include numbers in quotation marks. For example:

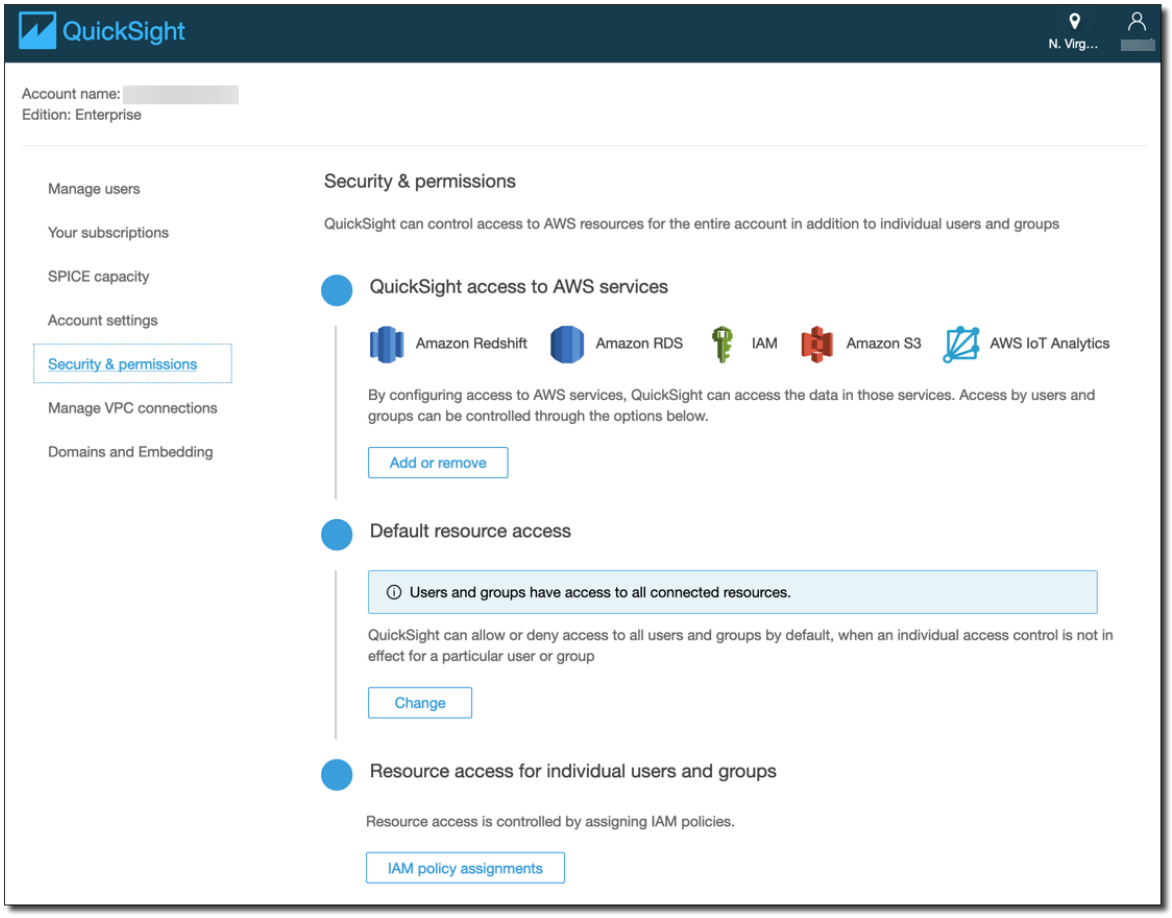
```
CREATE_TABLE "table123"(
  `_id` string,
```

```
`_index` string,
...
```

Visualizing AWS IoT Analytics Data with QuickSight

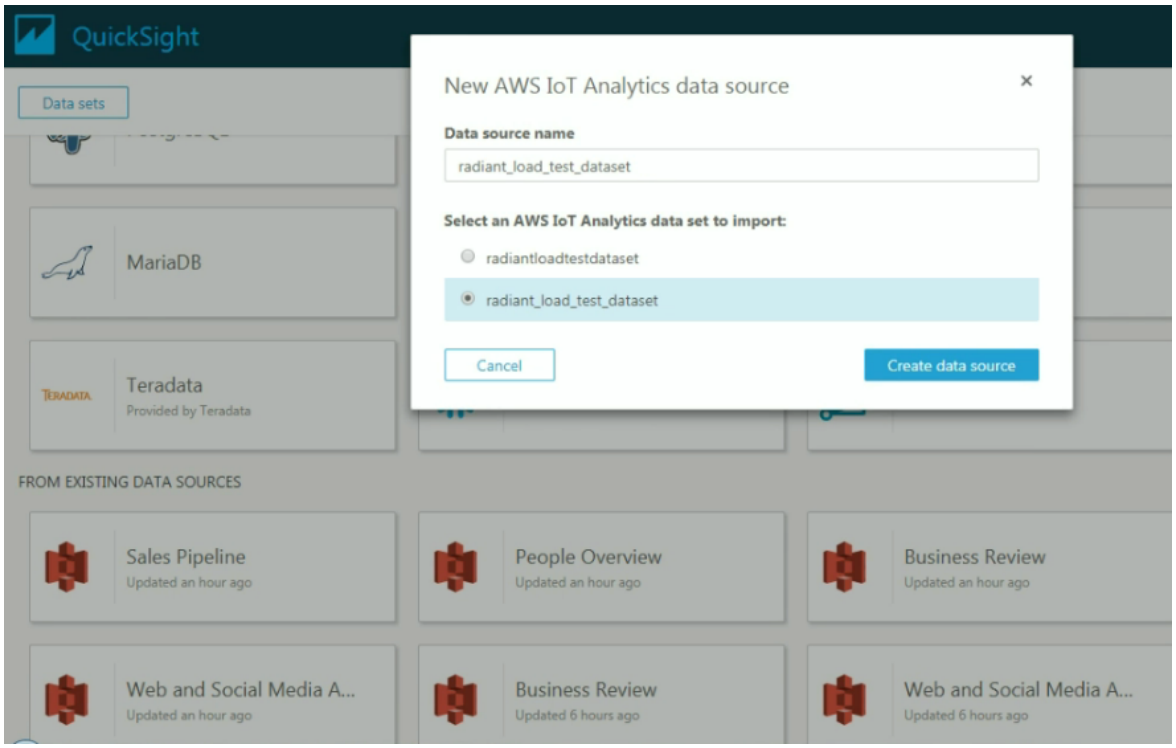
AWS IoT Analytics provides direct integration with Amazon QuickSight. Amazon QuickSight is a fast business analytics service you can use to build visualizations, perform ad-hoc analysis, and quickly get business insights from your data. Amazon QuickSight enables organizations to scale to hundreds of thousands of users, and delivers responsive performance by using a robust in-memory engine (SPICE). You can select your AWS IoT Analytics data sets in the QuickSight console and start creating dashboards and visualizations. Amazon QuickSight is available in these regions.

To get started with your QuickSight visualizations, you must create a QuickSight account. Make sure you give QuickSight access to your AWS IoT Analytics data when you set up your account. If you already have an account, give QuickSight access to your AWS IoT Analytics data by choosing **Admin**, **Manage QuickSight**, **Security & permissions**. Under **QuickSight access to AWS services** choose **Add or remove**,

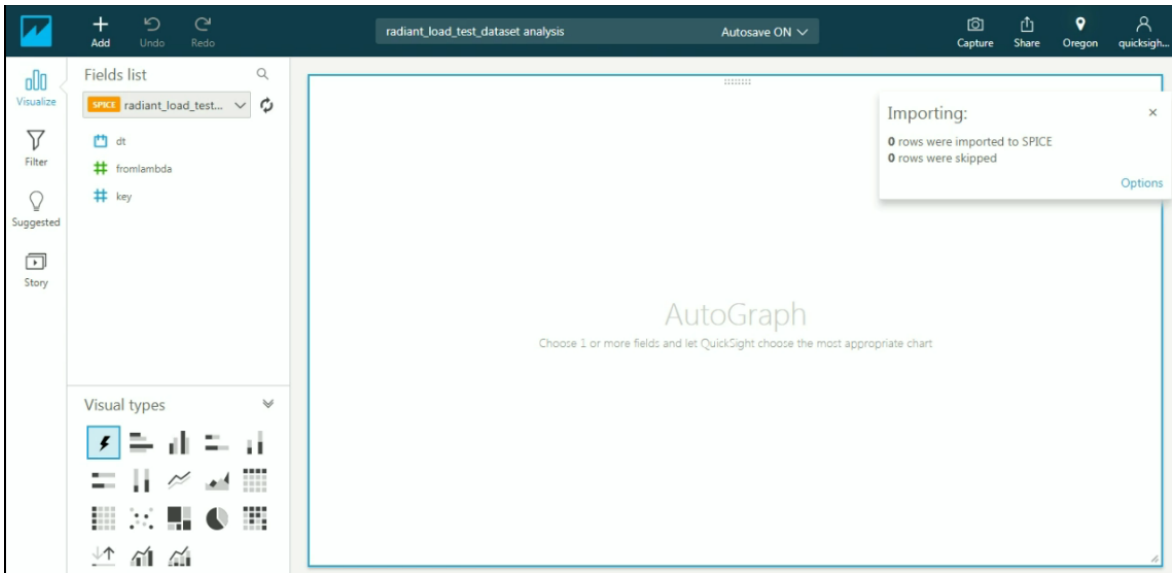


then check the box next to **AWS IoT Analytics** and select **Update**.

After your account is set up, from the main QuickSight console page choose **New Analysis** and **New data set**, and then select AWS IoT Analytics as the source. Type a name for your data source, select a data set to import, and then choose **Create data source**.



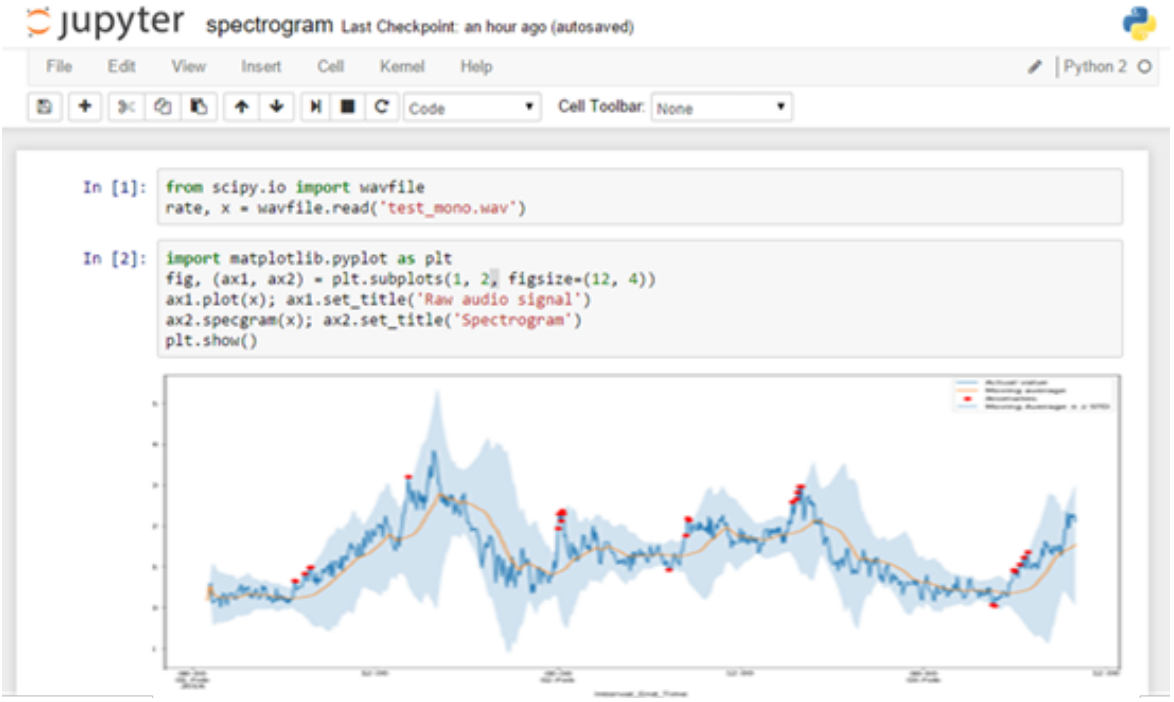
After your data source is created, you can create visualizations in QuickSight:



For information about QuickSight dashboards and data sets, see the QuickSight documentation.

Visualizing AWS IoT Analytics Data with the Console

AWS IoT Analytics can embed the HTML output of your container data set (found in the file "output.html") on the container data set content page of the AWS IoT Analytics console . For example, if you define a container data set that runs a Jupyter notebook, and you create a visualization in your Jupyter notebook:



then, after the container data set content is created, you are able view this visualization on the console's **Data Set** content page:

For information about creating a container data set that runs a Jupyter notebook, see Automating Your Workflow (p. 67).

AWS IoT Analytics Troubleshooting Guide

Q: How do I know if my messages are getting into AWS IoT Analytics?

- Check if the rule to inject data into the channel through the rules-engine is configured correctly:

```
aws iot get-topic-rule --rule-name <your-rule-name>
```

The response should look like:

```
{
    "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_number>:channel/<your-channel-name>"
                }
            }
        ],
        "ruleName": "<your-rule-name>"
    }
}
```

Make sure the region and channel name used in the rule are correct. To ensure your data is reaching the rules engine and the rule is being executed correctly, you might want to add a new target to store incoming messages in the S3 bucket temporarily.

Q: Why is my pipeline losing messages? How do I fix it?

- An activity has received an invalid JSON input:

  All activities, except Lambda activities, specifically require a valid JSON string as input. If the JSON received by an activity is invalid, then the message is dropped and does not make its way into the data store. Make sure you are ingesting valid JSON messages into the service. In case of binary input, make sure the first activity in your pipeline is a Lambda activity that converts the binary data to valid JSON before passing it to the next activity or storing it in the data store. See Lambda Function Example 2 (p. 46) for more information.
- A Lambda function invoked by a Lambda activity has insufficient permissions:

  Make sure that each Lambda function in a Lambda activity has permission to be invoked from the AWS IoT Analytics service. You can use the following CLI command to grant permission:

```
aws lambda add-permission --function-name <name> --region <region> --statement-id <id> --
principal iotanalytics.amazonaws.com --action lambda:InvokeFunction
```

- A filter or removeAttribute activity is incorrectly defined:

Make sure the definitions of any `filter` or `removeAttribute` activities are correct. If you filter out a message or remove all attributes from a message, that message is not added to the data store.

Q: Why is there no data in my data store?

- There is a delay between data ingestion and data availability:

  It might take several minutes after data is ingested into a channel before that data is available in the data store. The time varies based on the number of pipeline activities and the definition of any custom Lambda activities in your pipeline.
- Messages are being filtered out in your pipeline:

  Make sure you are not dropping messages in the pipeline. (See the previous question and response.)
- Your data set query is incorrect:

  Make sure the query that generates the data set from the data store is correct. Remove any unnecessary filters from the query to ensure your data reaches your data store.

Q: Why does my data set just show :code:`__dt`?

- This column is added by the service automatically and contains the approximate ingestion time of the data. It may be used to optimize your queries. If your data set contains nothing but this, see the previous question and response.

Q: How do I code an event driven by the data set completion?

- You must set up polling based on the `describe-dataset` command to check if the status of the data set with a particular timestamp is `SUCCEEDED`.

Q: How do I correctly configure my notebook instance to use the IotAnalytics Service?

- Follow these steps to make sure the IAM role you are using to create the notebook instance has the required permissions:
  1. Go to the Amazon SageMaker console and create a notebook instance.
  2. Fill in the details and choose **create a new role**. Make a note of the role ARN.
  3. Create the notebook instance. This also creates a role that Amazon SageMaker can use.
  4. Go to the IAM console and modify the newly created Amazon SageMaker role. When you open that role, it should have a managed policy.
  5. Click **add inline policy**, choose `IotAnalytics` as the service, and under read permission, select **GetDatasetContent**.
  6. Review the policy, add a policy name, and then **create** it. The newly created role now has policy permission to read a data set from AWS IoT Analytics.
  7. Go to the AWS IoT Analytics console and create notebooks in the notebook instance.
  8. Wait for the notebook instance to be in the “In Service” state.
  9. Choose **create notebooks**, and select the notebook instance you created. This creates a Jupyter notebook with the selected template that can access your data sets.

Q: Why can’t I create notebooks in an instance?

- Make sure you create a notebook instance with the correct IAM policy. (Follow the steps in the previous question.)

- Make sure the notebook instance is in the “In Service” state. (When you create an instance, it starts in a “Pending” state. It usually takes about five minutes for it to go into the “In Service” state. If the notebook instance goes into the “Failed” state after about five minutes, check the permissions again.)

Q: Why is my notebook unable to pull data from my data sets?

- Make sure the notebook instance has all the required permissions as described earlier. The notebook instance must have read permission to the data set’s content.

Q: Why am I not seeing my data sets in QuickSight?

- Follow these steps to make sure you have given QuickSight read permission for data set content:
  1. Click the icon in the upper-right corner (mentioning the account name) and choose **Manage QuickSight**
  2. Choose **Account settings**, and then under **Connected products & services** choose **Add or remove**
  3. Check the box next to **AWS IoT Analytics**, then select **Update**. This gives QuickSight read permissions to your data sets.
  4. Try again to visualize your data.

Q: Why am I not seeing the containerize button on my existing Jupyter Notebook?

- This is caused by a missing AWS IoT Analytics Containerization Plugin. If you created your SageMaker notebook instance before August 23, 2018, you need to manually install the plugin by following the instructions in Containerizing A Notebook (p. 85). Or, you can create a SageMaker notebook instance from the AWS IoT Analytics console where the plugin is automatically installed for you. If the SageMaker notebook instance was created in the SageMaker console, install the plugin manually.
- If you restarted the notebook instance after manually installing the plugin, you may lose the plugin and have to re-install it by following the instructions in Containerizing A Notebook (p. 85).
- If you don’t see the containerize button after creating the SageMaker notebook instance from the AWS IoT Analytics console or manually installing it, please reach out to AWS IoT Analytics Technical Support.

Q: Why is my containerization plugin installation failing?

- Usually, the plugin installation fails because of missing permissions in the SageMaker notebook instance. For the required permissions for the notebook instance, see Permissions (p. 71) and add the required permissions to the notebook instance role. If the problem persists, create a new notebook instance from the AWS Iot Analytics console.
- You can safely ignore the following message in the log if it appears during installation of the plugin: “To initialize this nbextension in the browser every time the notebook (or other app) loads.”

Q: Why is my containerization plugin throwing an error?

- Containerization can fail and generate errors for multiple reasons. Make sure that you’re using the correct kernel before containerizing your notebook. Containerized kernels begin with the “Containerized” prefix.
- Since the plugin creates and saves a docker image in an ECR repository, make sure that your notebook instance role has sufficient permissions to read, list and create ECR repositories. For the required permissions for the notebook instance, see Permissions (p. 71) and add the required permissions to the notebook instance role.

- Also make sure that the name of the repository complies with ECR requirements. ECR repository names must start with a letter and can contain only lower-case letters, numbers, hyphens, underscores, and forward slashes.
- If the containerization process fails with the error: “This instance has insufficient free space to run containerization” try using a larger instance to resolve the issue.
- If you see connection errors or an image creation error, please retry. If the problem persists, restart the instance and install the latest plugin version.

Q: Why don’t I see my variables during the containerization.

- The AWS Iot Analytics containerization plugin automatically recognizes all variables in your notebook after it runs the notebook with the “Containerized” kernel. Use one of the containerized kernels to run the notebook, and then perform containerization.

Q: What variables can I add to my container as an input?

- You can add any variable whose value you want to modify during the runtime as an input to your container. This enables you to run the same container with different parameters that need to be supplied at the time of dataset creation. The AWS IoT Analytics containerization Jupyter plugin simplifies this process by automatically recognizing the variables in the notebook and making them available as part of the containerization process.

Q: How do I set my container output as an input for subsequent analysis?

- A specific S3 location where the executed artifacts can be stored is created for each run of your container dataset. To access this output location, create a variable with type `outputFileUriValue` in your container dataset. The value of this variable should be an S3 path that is used for storing your additional output files. To access these saved artifacts in subsequent runs, you can use the code:*getDatasetContent* API and pick the appropriate output file required for the subsequent run.

Q: Why is my container dataset failing?

- Make sure that you’re passing the correct executionRole to the container dataset. The trust policy of the executionRole must include both “iotanalytics.amazonaws.com” and “sagemaker.amazonaws.com”.
- If you see “AlgorithmError” as the reason for the failure, try to debug your container code manually. This happens if there is a bug in the container code or the execution role doesn’t have permission to execute the container. If you containerized by using the AWS Iot Analytics Jupyter plugin, create a new SageMaker notebook instance with the same role as the executionRole of the containerDataset and try running the notebook manually. If the container was created outside of the Jupyter plugin, try manually running the code and limiting the permission to the executionRole.

Logging AWS IoT Analytics API Calls with CloudTrail

AWS IoT Analytics is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in AWS IoT Analytics. CloudTrail captures a subset of API calls for AWS IoT Analytics as events, including calls from the AWS IoT Analytics console and from code calls to the AWS IoT Analytics APIs. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for AWS IoT Analytics. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**. Using the information collected by CloudTrail, you can determine the request that was made to AWS IoT Analytics, the IP address from which the request was made, who made the request, when it was made, and additional details.

To learn more about CloudTrail, see the AWS CloudTrail User Guide.

AWS IoT Analytics Information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in AWS IoT Analytics, that activity is recorded in a CloudTrail event along with other AWS service events in **Event history**. You can view, search, and download recent events in your AWS account. For more information, see Viewing Events with CloudTrail Event History.

For an ongoing record of events in your AWS account, including events for AWS IoT Analytics, create a trail. A trail enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the console, the trail applies to all regions. The trail logs events from all regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see:

- Overview for Creating a Trail
- CloudTrail Supported Services and Integrations
- Configuring Amazon SNS Notifications for CloudTrail
- Receiving CloudTrail Log Files from Multiple Regions and Receiving CloudTrail Log Files from Multiple Accounts

AWS IoT Analytics supports logging the following actions as events in CloudTrail log files:

- CancelPipelineReprocessing (p. 137)
- CreateChannel (p. 138)
- CreateDataset (p. 141)
- CreateDatasetContent (p. 149)
- CreateDatastore (p. 150)
- CreatePipeline (p. 154)
- DeleteChannel (p. 161)
- DeleteDataset (p. 162)
- DeleteDatasetContent (p. 163)
- DeleteDatastore (p. 164)

- DeletePipeline (p. 165)
- DescribeChannel (p. 166)
- DescribeDataset (p. 169)
- DescribeDatastore (p. 177)
- DescribeLoggingOptions (p. 180)
- DescribePipeline (p. 181)
- GetDatasetContent (p. 189)
- ListChannels (p. 191)
- ListDatasets (p. 196)
- ListDatastores (p. 198)
- ListPipelines (p. 201)
- PutLoggingOptions (p. 205)
- RunPipelineActivity (p. 206)
- SampleChannelData (p. 212)
- StartPipelineReprocessing (p. 214)
- UpdateChannel (p. 218)
- UpdateDataset (p. 220)
- UpdateDatastore (p. 227)
- UpdatePipeline (p. 229)

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or Amazon Identity & Access Management (IAM) user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

For more information, see the CloudTrail userIdentity Element.

Understanding AWS IoT Analytics Log File Entries

A trail is a configuration that enables delivery of events as log files to an S3 bucket that you specify. CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested action, the date and time of the action, request parameters, and so on. CloudTrail log files are not an ordered stack trace of the public API calls, so they do not appear in any specific order.

The following example shows a CloudTrail log entry that demonstrates the `CreateChannel` action.

```
{
  "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"
}
```

The following example shows a CloudTrail log entry that demonstrates the `CreateDataset` action.

```
{
  "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"
}
```

Setting Up CloudWatch Logs

AWS IoT Analytics supports logging with Amazon CloudWatch. You can enable and configure Amazon CloudWatch logging for AWS IoT Analytics using the `PutLoggingOptions` command. This section describes that command and the permissions you must set using AWS Identity and Access Management (IAM) and Amazon CloudWatch.

For more information about CloudWatch Logs see the Amazon CloudWatch Logs User Guide. For more information about AWS IAM, see the AWS Identity and Access Management User Guide.

> **Note**
> Before you enable AWS IoT Analytics logging, make sure you understand the CloudWatch Logs access permissions. Users with access to CloudWatch Logs can see your debugging information. See Authentication and Access Control for Amazon CloudWatch Logs.

Create a Logging Role

First, you must create an AWS IAM role with permissions to perform the logging. Use the AWS IAM console or the AWS IAM commands:

- CreateRole

  ```
  aws iam create-role ...
  ```
- PutRolePolicy

  ```
  aws iam put-role-policy ...
  ```

You use the ARN of this role later when you call the AWS IoT Analytics `PutLoggingOptions` command. Here are the trust policy (used in `CreateRole`) and the role policy (used in `PutRolePolicy`) you should attach to this AWS IAM role.

`trust policy`:

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

`role policy`:

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

```
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream"
            ],
            "Resource": [
                "arn:aws:logs:*:*:*"
            ]
        }
    ]
}
```

In addition, you must give AWS IoT Analytics permission to put log events to Amazon CloudWatch using the Amazon CloudWatch command:

- PutResourcePolicy

```
aws logs put-resource-policy ...
```

Use the following resource policy:

`resource policy`:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "iotanalytics.amazonaws.com"
            },
            "Action": "logs:PutLogEvents",
            "Resource": "*"
        }
    ]
}
```

Configure and Enable Logging

Use the `PutLoggingOptions` command to configure and enable Amazon CloudWatch logging for AWS IoT Analytics. The `roleArn` in the `loggingOptions` field should be the ARN of the role you created in the previous section. You can also use the `DescribeLoggingOptions` command to check your logging options settings.

PutLoggingOptions

Sets or updates the AWS IoT Analytics logging options.

Note that if you update the value of any `loggingOptions` field, it takes up to one minute for the change to take effect. Also, if you change the policy attached to the role you specified in the `roleArn` field (for example, to correct an invalid policy) it takes up to 5 minutes for that change to take effect.

CLI Synopsis:

```
aws iotanalytics  put-logging-options
 --logging-options <value>
```

```
[--cli-input-json <value>]
[--generate-cli-skeleton]
```

cli-input-json format:

```
{
  "loggingOptions": {
    "roleArn": "string",
    "level": "string",
    "enabled": "boolean"
  }
}
```

fields:

- loggingOptions

  type: LoggingOptions

  The new values of the AWS IoT Analytics logging options.
- roleArn

  type: string; (length- max:2048 min:20)

  The ARN of the role that grants permission to AWS IoT Analytics to perform logging.
- level

  type: string

  The logging level. Currently, only “ERROR” is supported. enum: ERROR
- enabled

  type: boolean

  If true, logging is enabled for AWS IoT Analytics.

Output:

None

Errors:

- InvalidRequestException

  The request was not valid.

  HTTP response code: 400
- InternalFailureException

  There was an internal failure.

  HTTP response code: 500
- ServiceUnavailableException

  The service is temporarily unavailable.

  HTTP response code: 503
- ThrottlingException

The request was denied due to request throttling.

HTTP response code: 429

DescribeLoggingOptions

Retrieves the current settings of the AWS IoT Analytics logging options.

CLI Synopsis:

```
aws iotanalytics  describe-logging-options
[--cli-input-json <value>]
[--generate-cli-skeleton]
```

cli-input-json format:

```
{
}
```

fields:

Output:

```
{
  "loggingOptions": {
    "roleArn": "string",
    "level": "string",
    "enabled": "boolean"
  }
}
```

fields:

- loggingOptions

  type: LoggingOptions

  The current settings of the AWS IoT Analytics logging options.
- roleArn

  type: string; (length- max:2048 min:20)

  The ARN of the role that grants permission to AWS IoT Analytics to perform logging.
- level

  type: string

  The logging level. Currently, only “ERROR” is supported. enum: ERROR
- enabled

  type: boolean

  If true, logging is enabled for AWS IoT Analytics.

Errors:

- InvalidRequestException

  The request was not valid.

  HTTP response code: 400
- ResourceNotFoundException

  A resource with the specified name could not be found.

  HTTP response code: 404
- InternalFailureException

  There was an internal failure.

  HTTP response code: 500
- ServiceUnavailableException

  The service is temporarily unavailable.

  HTTP response code: 503
- ThrottlingException

  The request was denied due to request throttling.

  HTTP response code: 429

Namespace, Metrics and Dimensions

AWS IoT Analytics puts the following metrics into the Amazon CloudWatch repository:

Namespace
AWS/IoTAnalytics

Metric	Description
ActionExecution	The number of actions executed.
ActionExecutionThrottled	The number of actions that are throttled.
ActivityExecutionError	The number of errors generated while executing the pipeline activity.
IncomingMessages	The number of messages coming into the channel.
PipelineConcurrentExecutionCount	The number of pipeline activities which have executed concurrently.

Dimension	Description
ActionType	The type of action that is being monitored.
ChannelName	The name of the channel that is being monitored.

Dimension	Description
DatasetName	The name of the data set that is being monitored.
DatastoreName	The name of the data store that is being monitored.
PipelineActivityName	The name of the pipeline activity that is being monitored.
PipelineActivityType	The type of the pipeline activity that is being monitored.
PipelineName	The name of the pipeline that is being monitored.

Security in AWS IoT Analytics

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from a data center and network architecture that is built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The shared responsibility model describes this as security *of* the cloud and security *in* the cloud:

- **Security of the cloud** - AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. The effectiveness of our security is regularly tested and verified by third-party auditors as part of the AWS compliance programs. To learn about the compliance programs that apply to AWS IoT Analytics, see AWS Services in Scope by Compliance Program.
- **Security in the cloud** - Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your organization's requirements, and applicable laws and regulations.

This documentation will help you understand how to apply the shared responsibility model when using AWS IoT Analytics. The following topics show you how to configure AWS IoT Analytics to meet your security and compliance objectives. You'll also learn how to use other AWS services that can help you to monitor and secure your AWS IoT Analytics resources.

Identity and Access Management for AWS IoT Analytics

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use AWS IoT Analytics resources. IAM is an AWS service that you can use with no additional charge.

Audience

How you use AWS Identity and Access Management (IAM) differs, depending on the work you do in AWS IoT Analytics.

Service user - If you use the AWS IoT Analytics service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more AWS IoT Analytics features to do your work, you might need additional permissions. Understanding how access is managed can help you request the right permissions from your administrator. If you cannot access a feature in AWS IoT Analytics, see Troubleshooting AWS IoT Analytics Identity and Access (p. 129).

Service administrator - If you're in charge of AWS IoT Analytics resources at your company, you probably have full access to AWS IoT Analytics. It's your job to determine which AWS IoT Analytics features and resources your employees should access. You must then submit requests to your IAM administrator to change the permissions of your service users. Review the information on this page to understand the basic concepts of IAM. To learn more about how your company can use IAM with AWS IoT Analytics, see How AWS IoT Analytics Works with IAM (p. 123).

IAM administrator - If you're an IAM administrator, you might want to learn details about how you can write policies to manage access to AWS IoT Analytics. To view example AWS IoT Analytics identity-based policies that you can use in IAM, see AWS IoT Analytics Identity-Based Policy Examples (p. 125).

Authenticating with Identities

Authentication is how you sign in to AWS using your identity credentials. For more information about signing in using the AWS Management Console, see The IAM Console and Sign-in Page in the *IAM User Guide*.

You must be *authenticated* (signed in to AWS) as the AWS account root user, an IAM user, or by assuming an IAM role. You can also use your company's single sign-on authentication, or even sign in using Google or Facebook. In these cases, your administrator previously set up identity federation using IAM roles. When you access AWS using credentials from another company, you are assuming a role indirectly.

To sign in directly to the AWS Management Console, use your password with your root user email or your IAM user name. You can access AWS programmatically using your root user or IAM user access keys. AWS provides SDK and command line tools to cryptographically sign your request using your credentials. If you don't use AWS tools, you must sign the request yourself. Do this using *Signature Version 4*, a protocol for authenticating inbound API requests. For more information about authenticating requests, see Signature Version 4 Signing Process in the *AWS General Reference*.

Regardless of the authentication method that you use, you might also be required to provide additional security information. For example, AWS recommends that you use multi-factor authentication (MFA) to increase the security of your account. To learn more, see Using Multi-Factor Authentication (MFA) in AWS in the *IAM User Guide*.

AWS Account Root User

When you first create an AWS account, you begin with a single sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user* and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you do not use the root user for your everyday tasks, even the administrative ones. Instead, adhere to the best practice of using the root user only to create you first IAM user. Then securely lock away the root user credentials and use them to perform only a few account and service management tasks.

IAM Users and Groups

An IAM user is an identity within your AWS account that has specific permissions for a single person or application. An IAM user can have long-term credentials such as a user name and password or a set of access keys. To learn how to generate access keys, see Managing Access Keys for IAM Users in the *IAM User Guide*. When you generate access keys for an IAM user, make sure you view and securely save the key pair. You cannot recover a lost secret access key. Instead, you must generate a new access key pair.

An IAM group is an identity that specifies a collection of IAM users. You can't sign in as a group. You can use groups to specify permissions for multiple users at a time. Groups make permissions easier to manage for large sets of users. For example, you could have a group named *IAMAdmins* and give that group permissions to administer IAM resources.

Users are different from roles. A user is uniquely associated with one person or application, but a role is intended to be assumable by anyone who needs it. Users have permanent long-term credentials, but roles provide temporary credentials. To learn more, see When to Create an IAM User (Instead of a Role) in the *IAM User Guide*.

IAM Roles

An IAM role is an identity within your AWS account that has specific permissions. It is similar to an IAM user, but is not associated with a specific person. You can temporarily assume an IAM role in the AWS Management Console by switching roles. You can assume a role by calling an AWS CLI or AWS API operation or by using a custom URL. For more information about methods for using roles, see Using IAM Roles in the *IAM User Guide*.

IAM roles with temporary credentials are useful in the following situations:

- **Temporary IAM user permissions** - An IAM user can assume an IAM role to temporarily take on different permissions for a specific task.
- **Federated user access** - Instead of creating an IAM user, you can use existing identities from AWS Directory Service, your enterprise user directory, or a web identity provider. These are known as *federated users*. AWS assigns a role to a federated user when access is requested through an identity provider. For more information about federated users, see Federated Users and Roles in the *IAM User Guide*.
- **Cross-account access** - You can use an IAM role to allow someone (a trusted principal) in a different account to access resources in your account. Roles are the primary way to grant cross-account access. However, with some AWS services, you can attach a policy directly to a resource (instead of using a role as a proxy). To learn the difference between roles and resource-based policies for cross-account access, see How IAM Roles Differ from Resource-based Policies in the *IAM User Guide*.
- **AWS service access** - A service role is an IAM role that a service assumes to perform actions in your account on your behalf. When you set up some AWS service environments, you must define a role for the service to assume. This service role must include all the permissions that are required for the service to access the AWS resources that it needs. Service roles vary from service to service, but many allow you to choose your permissions as long as you meet the documented requirements for that service. Service roles provide access only within your account and cannot be used to grant access to services in other accounts. You can create, modify, and delete a service role from within IAM. For example, you can create a role that allows Amazon Redshift to access an Amazon S3 bucket on your behalf and then load data from that bucket into an Amazon Redshift cluster. For more information, see Creating a Role to Delegate Permissions to an AWS Service in the *IAM User Guide*.
- **Applications running on Amazon EC2** - You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS CLI or AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances in the *IAM User Guide*.

To learn whether to use IAM roles, see When to Create an IAM Role (Instead of a User) in the *IAM User Guide*.

Managing Access Using Policies

You control access in AWS by creating policies and attaching them to IAM identities or AWS resources. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. AWS evaluates these policies when an entity (root user, IAM user, or IAM role) makes a request. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents. For more information about the structure and contents of JSON policy documents, see Overview of JSON Policies in the *IAM User Guide*.

An IAM administrator can use policies to specify who has access to AWS resources, and what actions they can perform on those resources. Every IAM entity (user or role) starts with no permissions. In other words, by default, users can do nothing, not even change their own password. To give a user permission to do something, an administrator must attach a permissions policy to a user. Or the administrator can add the user to a group that has the intended permissions. When an administrator gives permissions to a group, all users in that group are granted those permissions.

IAM policies define permissions for an action regardless of the method that you use to perform the operation. For example, suppose that you have a policy that allows the `iam:GetRole` action. A user with that policy can get role information from the AWS Management Console, the AWS CLI, or the AWS API.

Identity-Based Policies

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, role, or group. These policies control what actions that identity can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Creating IAM Policies in the *IAM User Guide*.

Identity-based policies can be further categorized as *inline policies* or *managed policies*. Inline policies are embedded directly into a single user, group, or role. Managed policies are standalone policies that you can attach to multiple users, groups, and roles in your AWS account. Managed policies include AWS managed policies and customer managed policies. To learn how to choose between a managed policy or an inline policy, see Choosing Between Managed Policies and Inline Policies in the *IAM User Guide*.

Other Policy Types

AWS supports additional, less common policy types. These policy types can set the maximum permissions granted to you by the more common policy types.

- **Permissions boundaries** - A permissions boundary is an advanced feature in which you set the maximum permissions that an identity-based policy can grant to an IAM entity (IAM user or role). You can set a permissions boundary for an entity. The resulting permissions are the intersection of entity's identity-based policies and its permissions boundaries. Resource-based policies that specify the user or role in the `Principal` field are not limited by the permissions boundary. An explicit deny in any of these policies overrides the allow. For more information about permissions boundaries, see Permissions Boundaries for IAM Entities in the *IAM User Guide*.
- **Service control policies (SCPs)** - SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU) in AWS Organizations. AWS Organizations is a service for grouping and centrally managing multiple AWS accounts that your business owns. If you enable all features in an organization, then you can apply SCPs to any or all of your accounts. The SCP limits permissions for entities in member accounts, including each AWS Account root user. For more information about AWS Organizations and SCPs, see How SCPs Work in the *AWS Organizations User Guide*.
- **Session policies** - Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary session for a role or federated user. The resulting session's permissions are the intersection of the user or role's identity-based policies and the session policies. Permissions can also come from a resource-based policy. An explicit deny in any of these policies overrides the allow. For more information, see Session Policies in the *IAM User Guide*.

Multiple Policy Types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see Policy Evaluation Logic in the *IAM User Guide*.

Learn More

For more information about identity and access management for AWS IoT Analytics, continue to the following sections:

- How AWS IoT Analytics Works with IAM (p. 123)
- Troubleshooting AWS IoT Analytics Identity and Access (p. 129)

How AWS IoT Analytics Works with IAM

Before you use IAM to manage access to AWS IoT Analytics, you should understand what IAM features are available to use with AWS IoT Analytics. To get a high-level view of how AWS IoT Analytics and other AWS services work with IAM, see AWS Services That Work with IAM in the *IAM User Guide*.

AWS IoT Analytics Identity-Based Policies

With IAM identity-based policies, you can specify allowed or denied actions and resources and the conditions under which actions are allowed or denied. AWS IoT Analytics supports specific actions, resources, and condition keys. To learn about all of the elements that you use in a JSON policy, see IAM JSON Policy Elements Reference in the *IAM User Guide*.

Actions

The `Action` element of an IAM identity-based policy describes the specific action or actions that will be allowed or denied by the policy. Policy actions usually have the same name as the associated AWS API operation. The action is used in a policy to grant permissions to perform the associated operation.

Policy actions in AWS IoT Analytics use the following prefix before the action: `iotanalytics:` For example, to grant someone permission to create an AWS IoT Analytics channel with the AWS IoT Analytics `CreateChannel` API operation, you include the `iotanalytics:CreateChannel` action in their policy. To grant someone permission to send an message data to a channel with the AWS IoT Analytics `BatchPutMessage` API operation, you include the `iotanalytics:BatchPutMessage` action in their policy. Policy statements must include either an `Action` or `NotAction` element. AWS IoT Analytics defines its own set of actions that describe tasks that you can perform with this service.

To specify multiple actions in a single statement, separate them with commas as follows.

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

You can specify multiple actions using wildcards (*). For example, to specify all actions that begin with the word `Describe`, include the following action.

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

To see a list of AWS IoT Analytics actions, see Actions Defined by AWS IoT Analytics in the *IAM User Guide*.

Resources

The `Resource` element specifies the object or objects to which the action applies. Statements must include either a `Resource` or a `NotResource` element. You specify a resource using an ARN or using the wildcard (*) to indicate that the statement applies to all resources.

The AWS IoT Analytics dataset resource has the following ARN.

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

For more information about the format of ARNs, see Amazon Resource Names (ARNs) and AWS Service Namespaces.

For example, to specify the `Foobar` dataset in your statement, use the following ARN.

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

To specify all instances that belong to a specific account, use the wildcard (*).

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

Some AWS IoT Analytics actions, such as those for creating resources, cannot be performed on a specific resource. In those cases, you must use the wildcard (*).

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

Some AWS IoT Analytics API actions involve multiple resources. For example, `CreatePipeline` references a channel and a dataset, so an IAM user must have permissions to use the channel and the dataset. To specify multiple resources in a single statement, separate the ARNs with commas.

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

To see a list of AWS IoT Analytics resource types and their ARNs, see Resources Defined by AWS IoT Analytics in the *IAM User Guide*. To learn with which actions you can specify the ARN of each resource, see Actions Defined by AWS IoT Analytics.

Condition Keys

The `Condition` element (or `Condition` *block* lets you specify conditions in which a statement is in effect. The `Condition` element is optional. You can build conditional expressions that use condition operators, such as equals or less than, to match the condition in the policy with values in the request.

If you specify multiple `Condition` elements in a statement, or multiple keys in a single `Condition` element, AWS evaluates them using a logical `AND` operation. If you specify multiple values for a single condition key, AWS evaluates the condition using a logical `OR` operation. All of the conditions must be met before the statement's permissions are granted.

You can also use placeholder variables when you specify conditions. For example, you can grant an IAM user permission to access a resource only if it is tagged with their IAM user name. For more information, see IAM Policy Elements: Variables and Tags in the *IAM User Guide*.

AWS IoT Analytics does not provide any service-specific condition keys, but it does support using some global condition keys. To see all AWS global condition keys, see AWS Global Condition Context Keys in the *IAM User Guide*.

Examples

To view examples of AWS IoT Analytics identity-based policies, see AWS IoT Analytics Identity-Based Policy Examples (p. 125).

AWS IoT Analytics Resource-Based Policies

AWS IoT Analytics does not support resource-based policies. To view an example of a detailed resource-based policy page, see https://docs.aws.amazon.com/lambda/latest/dg/access-control-resource-based.html.

Authorization Based on AWS IoT Analytics Tags

You can attach tags to AWS IoT Analytics resources or pass tags in a request to AWS IoT Analytics. To control access based on tags, you provide tag information in the condition element of a policy using

the `iotanalytics:ResourceTag/{key-name}`, `aws:RequestTag/{key-name}` or `aws:TagKeys` condition keys. For more information about tagging AWS IoT Analytics resources, see Tagging Your AWS IoT Analytics Resources (p. 132).

To view an example identity-based policy for limiting access to a resource based on the tags on that resource, see Viewing AWS IoT Analytics Channels Based on Tags (p. 128).

AWS IoT Analytics IAM Roles

An IAM role is an entity within your AWS account that has specific permissions.

Using Temporary Credentials with AWS IoT Analytics

You can use temporary credentials to sign in with federation, assume an IAM role, or to assume a cross-account role. You obtain temporary security credentials by calling AWS Security Token Service (AWS STS) API operations such as AssumeRole or GetFederationToken.

AWS IoT Analytics does not support using temporary credentials.

Service-Linked Roles

Service-linked roles allow AWS services to access resources in other services to complete an action on your behalf. Service-linked roles appear in your IAM account and are owned by the service. An IAM administrator can view but not edit the permissions for service-linked roles.

AWS IoT Analytics does not support service-linked roles.

Service Roles

This feature allows a service to assume a service role on your behalf. This role allows the service to access resources in other services to complete an action on your behalf. Service roles appear in your IAM account and are owned by the account. This means that an IAM administrator can change the permissions for this role. However, doing so might break the functionality of the service.

AWS IoT Analytics supports service roles.

AWS IoT Analytics Identity-Based Policy Examples

By default, IAM users and roles don't have permission to create or modify AWS IoT Analytics resources. They also can't perform tasks using the AWS Management Console, AWS CLI, or AWS API. An IAM administrator must create IAM policies that grant users and roles permission to perform specific API operations on the specified resources they need. The administrator must then attach those policies to the IAM users or groups that require those permissions.

To learn how to create an IAM identity-based policy using these example JSON policy documents, see Creating Policies on the JSON Tab in the *IAM User Guide*.

Policy Best Practices

Identity-based policies are very powerful. They determine whether someone can create, access, or delete AWS IoT Analytics resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get Started Using AWS Managed Policies** - To start using AWS IoT Analytics quickly, use AWS managed policies to give your employees the permissions they need. These policies are already

available in your account and are maintained and updated by AWS. For more information, see Get Started Using Permissions With AWS Managed Policies in the *IAM User Guide*.

- **Grant Least Privilege** - When you create custom policies, grant only the permissions required to perform a task. Start with a minimum set of permissions and grant additional permissions as necessary. Doing so is more secure than starting with permissions that are too lenient and then trying to tighten them later. For more information, see Grant Least Privilege in the *IAM User Guide*.
- **Enable MFA for Sensitive Operations** - For extra security, require IAM users to use multi-factor authentication (MFA) to access sensitive resources or API operations. For more information, see Using Multi-Factor Authentication (MFA) in AWS in the *IAM User Guide*.
- **Use Policy Conditions for Extra Security** - To the extent that it's practical, define the conditions under which your identity-based policies allow access to a resource. For example, you can write conditions to specify a range of allowable IP addresses that a request must come from. You can also write conditions to allow requests only within a specified date or time range, or to require the use of SSL or MFA. For more information, see IAM JSON Policy Elements: Condition in the *IAM User Guide*.

Using the AWS IoT Analytics Console

To access the AWS IoT Analytics console, you must have a minimum set of permissions. These permissions must allow you to list and view details about the AWS IoT Analytics resources in your AWS account. If you create an identity-based policy that is more restrictive than the minimum required permissions, the console won't function as intended for entities (IAM users or roles) with that policy.

To ensure that those entities can still use the AWS IoT Analytics console, also attach the following AWS managed policy to the entities. For more information, see Adding Permissions to a User in the *IAM User Guide*.

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

You don't need to allow minimum console permissions for users that are making calls only to the AWS CLI or the AWS API. Instead, allow access to only the actions that match the API operation that you're trying to perform.

Allow Users to View Their Own Permissions

This example shows how you might create a policy that allows IAM users to view the inline and managed policies that are attached to their user identity. This policy includes permissions to complete this action on the console or programmatically using the AWS CLI or AWS API.

```
{
    "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": "*"
        }
    ]
```

```
}
```

Accessing One AWS IoT Analytics Input

In this example, you want to grant an IAM user in your AWS account access to one of your AWS IoT Analytics channels, `exampleChannel` You also want to allow the user to add, update, and delete channels.

The policy grants the `iotanalytics:ListChannels`, `iotanalytics:DescribeChannel`, `iotanalytics:CreateChannel`, `iotanalytics:DeleteChannel`, and `iotanalytics:UpdateChannel` permissions to the user. For an example walkthrough for the Amazon S3 service that grants permissions to users and tests them using the console, see An Example Walkthrough: Using user policies to control access to your bucket.

```
{
    "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/*"
       }
    ]
 }
```

Viewing AWS IoT Analytics Channels Based on Tags

You can use conditions in your identity-based policy to control access to AWS IoT Analytics resources based on tags. This example shows how you might create a policy that allows viewing a `channel`. However, permission is granted only if the `channel` tag `Owner` has the value of that user's user name. This policy also grants the permissions needed to complete this action on the console.

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

You can attach this policy to the IAM users in your account. If a user named `richard-roe` attempts to view an AWS IoT Analytics `channel`, the `channel` must be tagged `Owner=richard-roe` or `owner=richard-roe`. Otherwise, he is denied access. The condition tag key `Owner` matches both `Owner` and `owner` because condition key names are not case sensitive. For more information, see IAM JSON Policy Elements: Condition in the *IAM User Guide*.

Troubleshooting AWS IoT Analytics Identity and Access

Use the following information to help you diagnose and fix common issues that you might encounter when working with AWS IoT Analytics and IAM.

I Am Not Authorized to Perform an Action in AWS IoT Analytics

If the AWS Management console tells you that you're not authorized to perform an action, then you must contact your administrator for assistance. Your administrator is the person that provided you with your user name and password.

The following example error occurs when the `mateojackson` IAM user tries to use the console to view details about a `channel` but does not have `iotanalytics:ListChannels` permissions.

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

In this case, Mateo asks his administrator to update his policies to allow him to access the `my-example-channel` resource using the `iotanalytics:ListChannel` action.

I Am Not Authorized to Perform iam:PassRole

If you receive an error that you're not authorized to perform the `iam:PassRole` action, then you must contact your administrator for assistance. Your administrator is the person that provided you with your user name and password. Ask that person to update your policies to allow you to pass a role to AWS IoT Analytics.

Some AWS services allow you to pass an existing role to that service, instead of creating a new service role or service-linked role. To do this, you must have permissions to pass the role to the service.

The following example error occurs when an IAM user named `marymajor` tries to use the console to perform an action in AWS IoT Analytics. However, the action requires the service to have permissions granted by a service role. Mary does not have permissions to pass the role to the service.

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

In this case, Mary asks her administrator to update her policies to allow her to perform the `iam:PassRole` action.

I Want to View My Access Keys

After you create your IAM user access keys, you can view your access key ID at any time. However, you can't view your secret access key again. If you lose your secret key, you must create a new access key pair.

Access keys consist of two parts: an access key ID (for example, `AKIAIOSFODNN7EXAMPLE`) and a secret access key (for example, `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`). Like a user name and password, you must use both the access key ID and secret access key together to authenticate your requests. Manage your access keys as securely as you do your user name and password.

> **Important**
> Do not provide your access keys to a third party, even to help find your canonical user ID. By doing this, you might give someone permanent access to your account.

When you create an access key pair, you are prompted to save the access key ID and secret access key in a secure location. The secret access key is available only at the time you create it. If you lose your secret access key, you must add new access keys to your IAM user. You can have a maximum of two access keys. If you already have two, you must delete one key pair before creating a new one. To view instructions, see Managing Access Keys in the *IAM User Guide*.

I'm an Administrator and Want to Allow Others to Access AWS IoT Analytics

To allow others to access AWS IoT Analytics, you must create an IAM entity (user or role) for the person or application that needs access. They will use the credentials for that entity to access AWS. You must then attach a policy to the entity that grants them the correct permissions in AWS IoT Analytics.

To get started right away, see Creating Your First IAM Delegated User and Group in the *IAM User Guide*.

I Want to Allow People Outside of My AWS Account to Access My AWS IoT Analytics Resources

You can create a role that users in other accounts or people outside of your organization can use to access your resources. You can specify who is trusted to assume the role. For services that support resource-based policies or access control lists (ACLs), you can use those policies to grant people access to your resources.

To learn more, consult the following:

- To learn whether AWS IoT Analytics supports these features, see How AWS IoT Analytics Works with IAM (p. 123).
- To learn how to provide access to your resources across AWS accounts that you own, see Providing Access to an IAM User in Another AWS Account That You Own in the *IAM User Guide*.
- To learn how to provide access to your resources to third-party AWS accounts, see Providing Access to AWS Accounts Owned by Third Parties in the *IAM User Guide*.
- To learn how to provide access through identity federation, see Providing Access to Externally Authenticated Users (Identity Federation) in the *IAM User Guide*.
- To learn the difference between using roles and resource-based policies for cross-account access, see How IAM Roles Differ from Resource-based Policies in the *IAM User Guide*.

Compliance Validation for IoT Analytics

Third-party auditors assess the security and compliance of IoT Analytics as part of the AWS ISO Certification program.

For a list of AWS services in scope of specific compliance programs, see AWS Services in Scope by Compliance Program. For general information, see AWS Compliance Programs.

You can download third-party audit reports using &ART;. For more information, see <problematic>`Downloading Reports in AWS Artifact<artifact/latest/ug/downloading-documents.html>`_</problematic>
.

Your compliance responsibility when using IoT Analytics is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. AWS provides the following resources to help with compliance:

- Security and Compliance Quick Start Guides – These deployment guides discuss architectural considerations and provide steps for deploying security- and compliance-focused baseline environments on AWS.
- AWS Compliance Resources – This collection of workbooks and guides might apply to your industry and location.
- AWS Config – This AWS service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- AWS Security Hub – This AWS service provides a comprehensive view of your security state within AWS that helps you check your compliance with security industry standards and best practices.

Resilience in AWS IoT Analytics

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between Availability Zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see AWS Global Infrastructure.

Infrastructure Security in AWS IoT Analytics

As a managed service, AWS IoT Analytics is protected by the AWS global network security procedures that are described in the Amazon Web Services: Overview of Security Processes whitepaper.

You use AWS published API calls to access AWS IoT Analytics through the network. Clients must support Transport Layer Security (TLS) 1.0 or later. We recommend TLS 1.2 or later. Clients must also support cipher suites with perfect forward secrecy (PFS) such as Ephemeral Diffie-Hellman (DHE) or Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Most modern systems such as Java 7 and later support these modes.

Additionally, requests must be signed by using an access key ID and a secret access key that is associated with an IAM principal. Or you can use the AWS Security Token Service (AWS STS) to generate temporary security credentials to sign requests.

Tagging Your AWS IoT Analytics Resources

To help you manage your channels, data sets, data stores and pipelines, you can optionally assign your own metadata to each of these resources in the form of tags. This chapter describes tags and shows you how to create them.

Tag Basics

Tags enable you to categorize your AWS IoT Analytics resources in different ways, for example, by purpose, owner, or environment. This is useful when you have many resources of the same type — you can quickly identify a specific resource based on the tags you've assigned to it. Each tag consists of a key and optional value, both of which you define. For example, you could define a set of tags for your channels that helps you track the type of device responsible for each channel's message source. We recommend that you devise a set of tag keys that meets your needs for each resource type. Using a consistent set of tag keys makes it easier for you to manage your resources. You can search and filter the resources based on the tags you add.

You can also use tags to categorize and track your costs. When you apply tags to channels, data sets, data stores, or pipelines, AWS generates a cost allocation report as a comma-separated value (CSV) file with your usage and costs aggregated by your tags. You can apply tags that represent business categories (such as cost centers, application names, or owners) to organize your costs across multiple services. For more information about using tags for cost allocation, see Use Cost Allocation Tags in the AWS Billing and Cost Management User Guide.

For ease of use, use the Tag Editor in the AWS Management Console, which provides a central, unified way to create and manage your tags. For more information, see Working with Tag Editor in Getting Started with the AWS Management Console.

You can also work with tags using the AWS CLI and the AWS IoT Analytics API. You can associate tags with channels, data sets, data stores and pipelines when you create them; use the *Tags* field in the following commands:

- CreateChannel (p. 138)
- CreateDataset (p. 141)
- CreateDatastore (p. 150)
- CreatePipeline (p. 154)

You can add, modify, or delete tags for existing resources that support tagging; use the following commands:

- TagResource (p. 215)
- ListTagsForResource (p. 203)
- UntagResource (p. 217)

You can edit tag keys and values, and you can remove tags from a resource at any time. You can set the value of a tag to an empty string, but you can't set the value of a tag to null. If you add a tag that has

the same key as an existing tag on that resource, the new value overwrites the old value. If you delete a resource, any tags associated with the resource are also deleted.

Using Tags with IAM Policies

You can use the `Condition` element (also called the `Condition` block) with the following condition context keys/values in an IAM policy to control user access (permissions) based on a resource's tags:

- Use `iotanalytics:ResourceTag/<tag-key>: <tag-value>` to allow or deny user actions on resources with specific tags.
- Use `aws:RequestTag/<tag-key>: <tag-value>` to require that a specific tag be used (or not used) when making an API request to create or modify a resource that allows tags.
- Use `aws:TagKeys: [<tag-key>, ...]` to require that a specific set of tag keys be used (or not used) when making an API request to create or modify a resource that allows tags.

Note
The condition context keys/values in an IAM policy only apply to those AWS IoT Analytics actions where an identifier for a resource capable of being tagged is a required parameter. For example, the use of DescribeLoggingOptions (p. 180) is not allowed/denied on the basis of condition context keys/values because no taggable resource (channel, data set, data store or pipeline) is referenced in this request.

Controlling Access Using Tags in the AWS Identity and Access Management User Guide has additional information on using tags. The IAM JSON Policy Reference section of that guide has detailed syntax, descriptions and examples of the elements, variables, and evaluation logic of JSON policies in IAM.

The following example policy applies two tag-based restrictions. An IAM user restricted by this policy:

- cannot give a resource the tag "env=prod" (see the line `"aws:RequestTag/env" : "prod"` in the example).
- cannot modify or access a resource that has an existing tag "env=prod" (see the line `"iotanalytics:ResourceTag/env" : "prod"` in the example).

```
{
  "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": "*"
    }
  ]
}
```

You can also specify multiple tag values for a given tag key by enclosing them in a list, like this:

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

Note
If you allow/deny users access to resources based on tags, it is important to consider explicitly denying users the ability to add those tags to or remove them from the same resources. Otherwise, it is possible for a user to circumvent your restrictions and gain access to a resource by modifying its tags.

Tag Restrictions

The following basic restrictions apply to tags:

- Maximum number of tags per resource — 50
- Maximum key length — 127 Unicode characters in UTF-8
- Maximum value length — 255 Unicode characters in UTF-8
- Tag keys and values are case-sensitive.
- Do not use the *aws:* prefix in your tag names or values because it is reserved for AWS use. You can't edit or delete tag names or values with this prefix. Tags with this prefix do not count against your tags per resource limit.
- If your tagging schema is used across multiple services and resources, remember that other services may have restrictions on allowed characters. Generally, allowed characters are: letters, spaces, and numbers representable in UTF-8, plus the following special characters: + - = . _ : / @.

AWS IoT Analytics Commands

BatchPutMessage

Sends messages to a channel.

CLI Synopsis:

```
aws iotanalytics  batch-put-message
 --channel-name <value>
 --messages <value>
[--cli-input-json <value>]
[--generate-cli-skeleton]
```

cli-input-json format:

```
{
  "channelName": "string",
  "messages": [
    {
      "messageId": "string",
      "payload": "blob"
    }
  ]
}
```

fields:

- channelName

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the channel where the messages are sent.

- messages

  type: list member: Message

  The list of messages to be sent. Each message has format: '{ "messageId": "string", "payload": "string"}'. Note that the field names of message payloads (data) that you send to AWS IoT Analytics: Must contain only alphanumeric characters and undescores (_); no other special characters are allowed. Must begin with an alphabetic character or single underscore (_). Cannot contain hyphens (-). In regular expression terms: "^[
  <problematic>A-Za-z_</problematic>
  ]([A-Za-z0-9]*|[A-Za-z0-9][
  <problematic>A-Za-z0-9_</problematic>
  ]*)$". Cannot be greater than 255 characters. Are case-insensitive. (Fields named "foo" and "FOO" in the same payload are considered duplicates.) For example, {"temp_01": 29} or {"_temp_01": 29} are valid, but {"temp-01": 29}, {"01_temp": 29} or {"__temp_01": 29} are invalid in message payloads.

- messageId

  type: string; (length- max:128 min:1)

The ID you wish to assign to the message. Each “messageId” must be unique within each batch sent.

- payload

  type: blob

  The payload of the message. This may be a JSON string or a Base64-encoded string representing binary data (in which case you must decode it by means of a pipeline activity).

Output:

```
{
  "batchPutMessageErrorEntries": [
    {
      "messageId": "string",
      "errorCode": "string",
      "errorMessage": "string"
    }
  ]
}
```

fields:

- batchPutMessageErrorEntries

  type: list member: BatchPutMessageErrorEntry

  A list of any errors encountered when sending the messages to the channel.
- messageId

  type: string; (length- max:128 min:1)

  The ID of the message that caused the error. (See the value corresponding to the “messageId” key in the message object.)
- errorCode

  type: string

  The code associated with the error.
- errorMessage

  type: string

  The message associated with the error.

Errors:

- ResourceNotFoundException

  A resource with the specified name could not be found.

  HTTP response code: 404
- InvalidRequestException

  The request was not valid.

  HTTP response code: 400
- InternalFailureException

There was an internal failure.

HTTP response code: 500

- ServiceUnavailableException

  The service is temporarily unavailable.

  HTTP response code: 503
- ThrottlingException

  The request was denied due to request throttling.

  HTTP response code: 429

CancelPipelineReprocessing

Cancels the reprocessing of data through the pipeline.

CLI Synopsis:

```
aws iotanalytics  cancel-pipeline-reprocessing
 --pipeline-name <value>
 --reprocessing-id <value>
[--cli-input-json <value>]
[--generate-cli-skeleton]
```

cli-input-json format:

```
{
  "pipelineName": "string",
  "reprocessingId": "string"
}
```

fields:

- pipelineName

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of pipeline for which data reprocessing is canceled.
- reprocessingId

  type: string

  The ID of the reprocessing task (returned by “StartPipelineReprocessing”).

Output:

None

Errors:

- ResourceNotFoundException

  A resource with the specified name could not be found.

HTTP response code: 404

- InvalidRequestException

  The request was not valid.

  HTTP response code: 400
- InternalFailureException

  There was an internal failure.

  HTTP response code: 500
- ServiceUnavailableException

  The service is temporarily unavailable.

  HTTP response code: 503
- ThrottlingException

  The request was denied due to request throttling.

  HTTP response code: 429

CreateChannel

Creates a channel. A channel collects data from an MQTT topic and archives the raw, unprocessed messages before publishing the data to a pipeline.

CLI Synopsis:

```
aws iotanalytics  create-channel
 --channel-name <value>
[--channel-storage <value>]
[--retention-period <value>]
[--tags <value>]
[--cli-input-json <value>]
[--generate-cli-skeleton]
```

cli-input-json format:

```
{
  "channelName": "string",
  "channelStorage": {
    "serviceManagedS3": {
    },
    "customerManagedS3": {
      "bucket": "string",
      "keyPrefix": "string",
      "roleArn": "string"
    }
  },
  "retentionPeriod": {
    "unlimited": "boolean",
    "numberOfDays": "integer"
  },
  "tags": [
    {
      "key": "string",
```

```
      "value": "string"
    }
  ]
}
```

fields:

- channelName

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the channel.
- channelStorage

  type: ChannelStorage

  Where channel data is stored. You may choose one of “serviceManagedS3” or “customerManagedS3” storage. If not specified, the default is “serviceManagedS3”. This cannot be changed after creation of the channel.
- serviceManagedS3

  type: ServiceManagedChannelS3Storage

  Use this to store channel data in an S3 bucket managed by the AWS IoT Analytics service. The choice of service-managed or customer-managed S3 storage cannot be changed after creation of the channel.
- customerManagedS3

  type: CustomerManagedChannelS3Storage

  Use this to store channel data in an S3 bucket that you manage. If customer managed storage is selected, the “retentionPeriod” parameter is ignored. The choice of service-managed or customer-managed S3 storage cannot be changed after creation of the channel.
- bucket

  type: string; (length- max:255 min:3); (pattern: ^[a-zA-Z0-9.-_]*$)

  The name of the Amazon S3 bucket in which channel data is stored.
- keyPrefix

  type: string; (length- max:255 min:1); (pattern: ^[a-zA-Z0-9!_.*'()/{}:-]*/$)

  [Optional] The prefix used to create the keys of the channel data objects. Each object in an Amazon S3 bucket has a key that is its unique identifier within the bucket (each object in a bucket has exactly one key). The prefix must end with a ‘/’.
- roleArn

  type: string; (length- max:2048 min:20)

  The ARN of the role which grants AWS IoT Analytics permission to interact with your Amazon S3 resources.
- retentionPeriod

  type: RetentionPeriod

  How long, in days, message data is kept for the channel. When “customerManagedS3” storage is selected, this parameter is ignored.
- unlimited

type: boolean

If true, message data is kept indefinitely.

- numberOfDays

  type: integer java class: java.lang.Integer range- min:1

  The number of days that message data is kept. The “unlimited” parameter must be false.

- tags

  type: list member: Tag

  Metadata which can be used to manage the channel.

- key

  type: string; (length- max:128 min:1)

  The tag’s key.

- value

  type: string; (length- max:128 min:1)

  The tag’s value.

Output:

```
{
  "channelName": "string",
  "channelArn": "string",
  "retentionPeriod": {
    "unlimited": "boolean",
    "numberOfDays": "integer"
  }
}
```

fields:

- channelName

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the channel.

- channelArn

  type: string

  The ARN of the channel.

- retentionPeriod

  type: RetentionPeriod

  How long, in days, message data is kept for the channel.

- unlimited

  type: boolean

  If true, message data is kept indefinitely.

- numberOfDays

  type: integer java class: java.lang.Integer range- min:1

  The number of days that message data is kept. The “unlimited” parameter must be false.

Errors:

- InvalidRequestException

  The request was not valid.

  HTTP response code: 400
- ResourceAlreadyExistsException

  A resource with the same name already exists.

  HTTP response code: 409
- InternalFailureException

  There was an internal failure.

  HTTP response code: 500
- ServiceUnavailableException

  The service is temporarily unavailable.

  HTTP response code: 503
- ThrottlingException

  The request was denied due to request throttling.

  HTTP response code: 429
- LimitExceededException

  The command caused an internal limit to be exceeded.

  HTTP response code: 410

CreateDataset

Creates a data set. A data set stores data retrieved from a data store by applying a “queryAction” (a SQL query) or a “containerAction” (executing a containerized application). This operation creates the skeleton of a data set. The data set can be populated manually by calling “CreateDatasetContent” or automatically according to a “trigger” you specify.

CLI Synopsis:

```
aws iotanalytics  create-dataset
 --dataset-name <value>
 --actions <value>
[--triggers <value>]
[--content-delivery-rules <value>]
[--retention-period <value>]
[--versioning-configuration <value>]
[--tags <value>]
```

```
[--cli-input-json <value>]
[--generate-cli-skeleton]
```

cli-input-json format:

```
{
  "datasetName": "string",
  "actions": [
    {
      "actionName": "string",
      "queryAction": {
        "sqlQuery": "string",
        "filters": [
          {
            "deltaTime": {
              "offsetSeconds": "integer",
              "timeExpression": "string"
            }
          }
        ]
      },
      "containerAction": {
        "image": "string",
        "executionRoleArn": "string",
        "resourceConfiguration": {
          "computeType": "string",
          "volumeSizeInGB": "integer"
        },
        "variables": [
          {
            "name": "string",
            "stringValue": "string",
            "doubleValue": "double",
            "datasetContentVersionValue": {
              "datasetName": "string"
            },
            "outputFileUriValue": {
              "fileName": "string"
            }
          }
        ]
      }
    }
  ],
  "triggers": [
    {
      "schedule": {
        "expression": "string"
      },
      "dataset": {
        "name": "string"
      }
    }
  ],
  "contentDeliveryRules": [
    {
      "entryName": "string",
      "destination": {
        "iotEventsDestinationConfiguration": {
          "inputName": "string",
          "roleArn": "string"
        },
        "s3DestinationConfiguration": {
          "bucket": "string",
          "key": "string",
```

```
          "glueConfiguration": {
            "tableName": "string",
            "databaseName": "string"
          },
          "roleArn": "string"
        }
      }
    }
  ],
  "retentionPeriod": {
    "unlimited": "boolean",
    "numberOfDays": "integer"
  },
  "versioningConfiguration": {
    "unlimited": "boolean",
    "maxVersions": "integer"
  },
  "tags": [
    {
      "key": "string",
      "value": "string"
    }
  ]
}
```

fields:

- datasetName

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the data set.
- actions

  type: list member: DatasetAction

  A list of actions that create the data set contents.
- actionName

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the data set action by which data set contents are automatically created.
- queryAction

  type: SqlQueryDatasetAction

  An “SqlQueryDatasetAction” object that uses an SQL query to automatically create data set contents.
- sqlQuery

  type: string

  A SQL query string.
- filters

  type: list member: QueryFilter

  Pre-filters applied to message data.
- deltaTime

  type: DeltaTime

Used to limit data to that which has arrived since the last execution of the action.

- offsetSeconds

  type: integer java class: java.lang.Integer

  The number of seconds of estimated “in flight” lag time of message data. When you create data set contents using message data from a specified time frame, some message data may still be “in flight” when processing begins, and so will not arrive in time to be processed. Use this field to make allowances for the “in flight” time of your message data, so that data not processed from a previous time frame will be included with the next time frame. Without this, missed message data would be excluded from processing during the next time frame as well, because its timestamp places it within the previous time frame.

- timeExpression

  type: string

  An expression by which the time of the message data may be determined. This may be the name of a timestamp field, or a SQL expression which is used to derive the time the message data was generated.

- containerAction

  type: ContainerDatasetAction

  Information which allows the system to run a containerized application in order to create the data set contents. The application must be in a Docker container along with any needed support libraries.

- image

  type: string; (length- max:255)

  The ARN of the Docker container stored in your account. The Docker container contains an application and needed support libraries and is used to generate data set contents.

- executionRoleArn

  type: string; (length- max:2048 min:20)

  The ARN of the role which gives permission to the system to access needed resources in order to run the “containerAction”. This includes, at minimum, permission to retrieve the data set contents which are the input to the containerized application.

- resourceConfiguration

  type: ResourceConfiguration

  Configuration of the resource which executes the “containerAction”.

- computeType

  type: string

  The type of the compute resource used to execute the “containerAction”. Possible values are: ACU_1 (vCPU=4, memory=16GiB) or ACU_2 (vCPU=8, memory=32GiB). enum: ACU_1 | ACU_2

- volumeSizeInGB

  type: integer range- max:50 min:1

  The size (in GB) of the persistent storage available to the resource instance used to execute the “containerAction” (min: 1, max: 50).

- variables

  type: list member: Variable

The values of variables used within the context of the execution of the containerized application (basically, parameters passed to the application). Each variable must have a name and a value given by one of “stringValue”, “datasetContentVersionValue”, or “outputFileUriValue”.

- name

  type: string; (length- max:256 min:1)

  The name of the variable.
- stringValue

  type: string; (length- max:1024 min:0)

  The value of the variable as a string.
- datasetContentVersionValue

  type: DatasetContentVersionValue

  The value of the variable as a structure that specifies a data set content version.
- datasetName

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the data set whose latest contents are used as input to the notebook or application.
- outputFileUriValue

  type: OutputFileUriValue

  The value of the variable as a structure that specifies an output file URI.
- fileName

  type: string; (pattern: [w.-]{1,255})

  The URI of the location where data set contents are stored, usually the URI of a file in an S3 bucket.
- triggers

  type: list member: DatasetTrigger

  A list of triggers. A trigger causes data set contents to be populated at a specified time interval or when another data set’s contents are created. The list of triggers can be empty or contain up to five DataSetTrigger objects.
- schedule

  type: Schedule

  The “Schedule” when the trigger is initiated.
- expression

  type: string

  The expression that defines when to trigger an update. For more information, see Schedule Expressions for Rules in the Amazon CloudWatch Events User Guide.
- dataset

  type: TriggeringDataset

  The data set whose content creation triggers the creation of this data set’s contents.
- name

type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

The name of the data set whose content generation triggers the new data set content generation.

- contentDeliveryRules

  type: list member: DatasetContentDeliveryRule

  When data set contents are created they are delivered to destinations specified here.

- entryName

  type: string

  The name of the data set content delivery rules entry.

- destination

  type: DatasetContentDeliveryDestination

  The destination to which data set contents are delivered.

- iotEventsDestinationConfiguration

  type: IotEventsDestinationConfiguration

  Configuration information for delivery of data set contents to AWS IoT Events.

- inputName

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z][a-zA-Z0-9_]*$)

  The name of the AWS IoT Events input to which data set contents are delivered.

- roleArn

  type: string; (length- max:2048 min:20)

  The ARN of the role which grants AWS IoT Analytics permission to deliver data set contents to an AWS IoT Events input.

- s3DestinationConfiguration

  type: S3DestinationConfiguration

  Configuration information for delivery of data set contents to Amazon S3.

- bucket

  type: string; (length- max:255 min:3); (pattern: ^[a-zA-Z0-9.-_]*$)

  The name of the Amazon S3 bucket to which data set contents are delivered.

- key

  type: string; (length- max:255 min:1); (pattern: ^[a-zA-Z0-9!_.*'()/{}:-]*$)

  The key of the data set contents object. Each object in an Amazon S3 bucket has a key that is its unique identifier within the bucket (each object in a bucket has exactly one key). To produce a unique key, you can use "!{iotanalytics:scheduleTime}" to insert the time of the scheduled SQL query run, or "!{iotanalytics:versioned} to insert a unique hash identifying the data set, for example: "/DataSet/!{iotanalytics:scheduleTime}/!{iotanalytics:versioned}.csv".

- glueConfiguration

  type: GlueConfiguration

Configuration information for coordination with the AWS Glue ETL (extract, transform and load) service.

- tableName

  type: string; (length- max:150 min:1); (pattern: [u0020-uD7FFuE000-uFFFDuD800uDC00-uDBFFuDFFFt]*)

  The name of the table in your AWS Glue Data Catalog which is used to perform the ETL (extract, transform and load) operations. (An AWS Glue Data Catalog table contains partitioned data and descriptions of data sources and targets.)

- databaseName

  type: string; (length- max:150 min:1); (pattern: [u0020-uD7FFuE000-uFFFDuD800uDC00-uDBFFuDFFFt]*)

  The name of the database in your AWS Glue Data Catalog in which the table is located. (An AWS Glue Data Catalog database contains Glue Data tables.)

- roleArn

  type: string; (length- max:2048 min:20)

  The ARN of the role which grants AWS IoT Analytics permission to interact with your Amazon S3 and AWS Glue resources.

- retentionPeriod

  type: RetentionPeriod

  [Optional] How long, in days, versions of data set contents are kept for the data set. If not specified or set to null, versions of data set contents are retained for at most 90 days. The number of versions of data set contents retained is determined by the versioningConfiguration parameter. (For more information, see https://docs.aws.amazon.com/iotanalytics/latest/userguide/getting-started.html#aws-iot-analytics-dataset-versions)

- unlimited

  type: boolean

  If true, message data is kept indefinitely.

- numberOfDays

  type: integer java class: java.lang.Integer range- min:1

  The number of days that message data is kept. The “unlimited” parameter must be false.

- versioningConfiguration

  type: VersioningConfiguration

  [Optional] How many versions of data set contents are kept. If not specified or set to null, only the latest version plus the latest succeeded version (if they are different) are kept for the time period specified by the “retentionPeriod” parameter. (For more information, see https://docs.aws.amazon.com/iotanalytics/latest/userguide/getting-started.html#aws-iot-analytics-dataset-versions)

- unlimited

  type: boolean

  If true, unlimited versions of data set contents will be kept.

- maxVersions

  type: integer java class: java.lang.Integer range- max:1000 min:1

  How many versions of data set contents will be kept. The “unlimited” parameter must be false.
- tags

  type: list member: Tag

  Metadata which can be used to manage the data set.
- key

  type: string; (length- max:128 min:1)

  The tag’s key.
- value

  type: string; (length- max:128 min:1)

  The tag’s value.

Output:

```
{
  "datasetName": "string",
  "datasetArn": "string",
  "retentionPeriod": {
    "unlimited": "boolean",
    "numberOfDays": "integer"
  }
}
```

fields:

- datasetName

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the data set.
- datasetArn

  type: string

  The ARN of the data set.
- retentionPeriod

  type: RetentionPeriod

  How long, in days, data set contents are kept for the data set.
- unlimited

  type: boolean

  If true, message data is kept indefinitely.
- numberOfDays

  type: integer java class: java.lang.Integer range- min:1

The number of days that message data is kept. The “unlimited” parameter must be false.

Errors:

- InvalidRequestException

  The request was not valid.

  HTTP response code: 400
- ResourceAlreadyExistsException

  A resource with the same name already exists.

  HTTP response code: 409
- InternalFailureException

  There was an internal failure.

  HTTP response code: 500
- ServiceUnavailableException

  The service is temporarily unavailable.

  HTTP response code: 503
- ThrottlingException

  The request was denied due to request throttling.

  HTTP response code: 429
- LimitExceededException

  The command caused an internal limit to be exceeded.

  HTTP response code: 410

CreateDatasetContent

Creates the content of a data set by applying a “queryAction” (a SQL query) or a “containerAction” (executing a containerized application).

CLI Synopsis:

```
aws iotanalytics  create-dataset-content
 --dataset-name <value>
[--cli-input-json <value>]
[--generate-cli-skeleton]
```

cli-input-json format:

```
{
  "datasetName": "string"
}
```

fields:

- datasetName

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the data set.

Output:

```
{
  "versionId": "string"
}
```

fields:

- versionId

  type: string; (length- max:36 min:7)

  The version ID of the data set contents which are being created.

Errors:

- InvalidRequestException

  The request was not valid.

  HTTP response code: 400
- ResourceNotFoundException

  A resource with the specified name could not be found.

  HTTP response code: 404
- InternalFailureException

  There was an internal failure.

  HTTP response code: 500
- ServiceUnavailableException

  The service is temporarily unavailable.

  HTTP response code: 503
- ThrottlingException

  The request was denied due to request throttling.

  HTTP response code: 429

CreateDatastore

Creates a data store, which is a repository for messages.

CLI Synopsis:

```
aws iotanalytics  create-datastore
```

```
 --datastore-name <value>
[--datastore-storage <value>]
[--retention-period <value>]
[--tags <value>]
[--cli-input-json <value>]
[--generate-cli-skeleton]
```

cli-input-json format:

```
{
  "datastoreName": "string",
  "datastoreStorage": {
    "serviceManagedS3": {
    },
    "customerManagedS3": {
      "bucket": "string",
      "keyPrefix": "string",
      "roleArn": "string"
    }
  },
  "retentionPeriod": {
    "unlimited": "boolean",
    "numberOfDays": "integer"
  },
  "tags": [
    {
      "key": "string",
      "value": "string"
    }
  ]
}
```

fields:

- datastoreName

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the data store.
- datastoreStorage

  type: DatastoreStorage

  Where data store data is stored. You may choose one of “serviceManagedS3” or “customerManagedS3” storage. If not specified, the default is “serviceManagedS3”. This cannot be changed after the data store is created.
- serviceManagedS3

  type: ServiceManagedDatastoreS3Storage

  Use this to store data store data in an S3 bucket managed by the AWS IoT Analytics service. The choice of service-managed or customer-managed S3 storage cannot be changed after creation of the data store.
- customerManagedS3

  type: CustomerManagedDatastoreS3Storage

  Use this to store data store data in an S3 bucket that you manage. When customer managed storage is selected, the “retentionPeriod” parameter is ignored. The choice of service-managed or customer-managed S3 storage cannot be changed after creation of the data store.

- bucket

  type: string; (length- max:255 min:3); (pattern: ^[a-zA-Z0-9.-_]*$)

  The name of the Amazon S3 bucket in which data store data is stored.
- keyPrefix

  type: string; (length- max:255 min:1); (pattern: ^[a-zA-Z0-9!_.*'()/{}:-]*/$)

  [Optional] The prefix used to create the keys of the data store data objects. Each object in an Amazon S3 bucket has a key that is its unique identifier within the bucket (each object in a bucket has exactly one key). The prefix must end with a '/'.
- roleArn

  type: string; (length- max:2048 min:20)

  The ARN of the role which grants AWS IoT Analytics permission to interact with your Amazon S3 resources.
- retentionPeriod

  type: RetentionPeriod

  How long, in days, message data is kept for the data store. When "customerManagedS3" storage is selected, this parameter is ignored.
- unlimited

  type: boolean

  If true, message data is kept indefinitely.
- numberOfDays

  type: integer java class: java.lang.Integer range- min:1

  The number of days that message data is kept. The "unlimited" parameter must be false.
- tags

  type: list member: Tag

  Metadata which can be used to manage the data store.
- key

  type: string; (length- max:128 min:1)

  The tag's key.
- value

  type: string; (length- max:128 min:1)

  The tag's value.

Output:

```
{
  "datastoreName": "string",
  "datastoreArn": "string",
  "retentionPeriod": {
    "unlimited": "boolean",
    "numberOfDays": "integer"
```

```
  }
}
```

fields:

- datastoreName

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the data store.
- datastoreArn

  type: string

  The ARN of the data store.
- retentionPeriod

  type: RetentionPeriod

  How long, in days, message data is kept for the data store.
- unlimited

  type: boolean

  If true, message data is kept indefinitely.
- numberOfDays

  type: integer java class: java.lang.Integer range- min:1

  The number of days that message data is kept. The “unlimited” parameter must be false.

Errors:

- InvalidRequestException

  The request was not valid.

  HTTP response code: 400
- ResourceAlreadyExistsException

  A resource with the same name already exists.

  HTTP response code: 409
- InternalFailureException

  There was an internal failure.

  HTTP response code: 500
- ServiceUnavailableException

  The service is temporarily unavailable.

  HTTP response code: 503
- ThrottlingException

  The request was denied due to request throttling.

  HTTP response code: 429

- LimitExceededException

  The command caused an internal limit to be exceeded.

  HTTP response code: 410

CreatePipeline

Creates a pipeline. A pipeline consumes messages from a channel and allows you to process the messages before storing them in a data store. You must specify both a channel and a datastore activity and, optionally, as many as 23 additional activities in the pipelineActivities array.

CLI Synopsis:

```
aws iotanalytics  create-pipeline
 --pipeline-name <value>
 --pipeline-activities <value>
[--tags <value>]
[--cli-input-json <value>]
[--generate-cli-skeleton]
```

cli-input-json format:

```
{
  "pipelineName": "string",
  "pipelineActivities": [
    {
      "channel": {
        "name": "string",
        "channelName": "string",
        "next": "string"
      },
      "lambda": {
        "name": "string",
        "lambdaName": "string",
        "batchSize": "integer",
        "next": "string"
      },
      "datastore": {
        "name": "string",
        "datastoreName": "string"
      },
      "addAttributes": {
        "name": "string",
        "attributes": {
          "string": "string"
        },
        "next": "string"
      },
      "removeAttributes": {
        "name": "string",
        "attributes": [
          "string"
        ],
        "next": "string"
      },
      "selectAttributes": {
        "name": "string",
        "attributes": [
          "string"
        ],
```

```
        "next": "string"
      },
      "filter": {
        "name": "string",
        "filter": "string",
        "next": "string"
      },
      "math": {
        "name": "string",
        "attribute": "string",
        "math": "string",
        "next": "string"
      },
      "deviceRegistryEnrich": {
        "name": "string",
        "attribute": "string",
        "thingName": "string",
        "roleArn": "string",
        "next": "string"
      },
      "deviceShadowEnrich": {
        "name": "string",
        "attribute": "string",
        "thingName": "string",
        "roleArn": "string",
        "next": "string"
      }
    }
  ],
  "tags": [
    {
      "key": "string",
      "value": "string"
    }
  ]
}
```

fields:

- pipelineName

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the pipeline.
- pipelineActivities

  type: list member: PipelineActivity

  A list of “PipelineActivity” objects. Activities perform transformations on your messages, such as removing, renaming or adding message attributes; filtering messages based on attribute values; invoking your Lambda functions on messages for advanced processing; or performing mathematical transformations to normalize device data. The list can be 2-25 PipelineActivity objects and must contain both a channel and a datastore activity. Each entry in the list must contain only one activity, for example: pipelineActivities = [{ “channel”: { … } }, { “lambda”: { … } }, …]
- channel

  type: ChannelActivity

  Determines the source of the messages to be processed.
- name

  type: string; (length- max:128 min:1)

The name of the 'channel' activity.

- channelName

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the channel from which the messages are processed.

- next

  type: string; (length- max:128 min:1)

  The next activity in the pipeline.

- lambda

  type: LambdaActivity

  Runs a Lambda function to modify the message.

- name

  type: string; (length- max:128 min:1)

  The name of the 'lambda' activity.

- lambdaName

  type: string; (length- max:64 min:1); (pattern: ^[a-zA-Z0-9_-]+$)

  The name of the Lambda function that is run on the message.

- batchSize

  type: integer java class: java.lang.Integer range- max:1000 min:1

  The number of messages passed to the Lambda function for processing. The AWS Lambda function must be able to process all of these messages within five minutes, which is the maximum timeout duration for Lambda functions.

- next

  type: string; (length- max:128 min:1)

  The next activity in the pipeline.

- datastore

  type: DatastoreActivity

  Specifies where to store the processed message data.

- name

  type: string; (length- max:128 min:1)

  The name of the 'datastore' activity.

- datastoreName

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the data store where processed messages are stored.

- addAttributes

  type: AddAttributesActivity

Adds other attributes based on existing attributes in the message.

- name

  type: string; (length- max:128 min:1)

  The name of the ‘addAttributes’ activity.

- attributes

  type: map key: AttributeName value: AttributeName

  A list of 1-50 “AttributeNameMapping” objects that map an existing attribute to a new attribute. The existing attributes remain in the message, so if you want to remove the originals, use “RemoveAttributeActivity”.

- next

  type: string; (length- max:128 min:1)

  The next activity in the pipeline.

- removeAttributes

  type: RemoveAttributesActivity

  Removes attributes from a message.

- name

  type: string; (length- max:128 min:1)

  The name of the ‘removeAttributes’ activity.

- attributes

  type: list member: AttributeName

  A list of 1-50 attributes to remove from the message.

- next

  type: string; (length- max:128 min:1)

  The next activity in the pipeline.

- selectAttributes

  type: SelectAttributesActivity

  Creates a new message using only the specified attributes from the original message.

- name

  type: string; (length- max:128 min:1)

  The name of the ‘selectAttributes’ activity.

- attributes

  type: list member: AttributeName

  A list of the attributes to select from the message.

- next

  type: string; (length- max:128 min:1)

The next activity in the pipeline.

- filter

  type: FilterActivity

  Filters a message based on its attributes.

- name

  type: string; (length- max:128 min:1)

  The name of the 'filter' activity.

- filter

  type: string; (length- max:256 min:1)

  An expression that looks like a SQL WHERE clause that must return a Boolean value.

- next

  type: string; (length- max:128 min:1)

  The next activity in the pipeline.

- math

  type: MathActivity

  Computes an arithmetic expression using the message's attributes and adds it to the message.

- name

  type: string; (length- max:128 min:1)

  The name of the 'math' activity.

- attribute

  type: string; (length- max:256 min:1)

  The name of the attribute that contains the result of the math operation.

- math

  type: string; (length- max:256 min:1)

  An expression that uses one or more existing attributes and must return an integer value.

- next

  type: string; (length- max:128 min:1)

  The next activity in the pipeline.

- deviceRegistryEnrich

  type: DeviceRegistryEnrichActivity

  Adds data from the AWS IoT device registry to your message.

- name

  type: string; (length- max:128 min:1)

  The name of the 'deviceRegistryEnrich' activity.

- attribute

type: string; (length- max:256 min:1)

The name of the attribute that is added to the message.

- thingName

  type: string; (length- max:256 min:1)

  The name of the IoT device whose registry information is added to the message.
- roleArn

  type: string; (length- max:2048 min:20)

  The ARN of the role that allows access to the device's registry information.
- next

  type: string; (length- max:128 min:1)

  The next activity in the pipeline.
- deviceShadowEnrich

  type: DeviceShadowEnrichActivity

  Adds information from the AWS IoT Device Shadows service to a message.
- name

  type: string; (length- max:128 min:1)

  The name of the 'deviceShadowEnrich' activity.
- attribute

  type: string; (length- max:256 min:1)

  The name of the attribute that is added to the message.
- thingName

  type: string; (length- max:256 min:1)

  The name of the IoT device whose shadow information is added to the message.
- roleArn

  type: string; (length- max:2048 min:20)

  The ARN of the role that allows access to the device's shadow.
- next

  type: string; (length- max:128 min:1)

  The next activity in the pipeline.
- tags

  type: list member: Tag

  Metadata which can be used to manage the pipeline.
- key

  type: string; (length- max:128 min:1)

The tag's key.

- value

  type: string; (length- max:128 min:1)

  The tag's value.

Output:

```
{
  "pipelineName": "string",
  "pipelineArn": "string"
}
```

fields:

- pipelineName

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the pipeline.

- pipelineArn

  type: string

  The ARN of the pipeline.

Errors:

- InvalidRequestException

  The request was not valid.

  HTTP response code: 400

- ResourceAlreadyExistsException

  A resource with the same name already exists.

  HTTP response code: 409

- InternalFailureException

  There was an internal failure.

  HTTP response code: 500

- ServiceUnavailableException

  The service is temporarily unavailable.

  HTTP response code: 503

- ThrottlingException

  The request was denied due to request throttling.

  HTTP response code: 429

- LimitExceededException

The command caused an internal limit to be exceeded.

HTTP response code: 410

DeleteChannel

Deletes the specified channel.

CLI Synopsis:

```
aws iotanalytics  delete-channel
 --channel-name <value>
[--cli-input-json <value>]
[--generate-cli-skeleton]
```

cli-input-json format:

```
{
  "channelName": "string"
}
```

fields:

- channelName

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the channel to delete.

Output:

None

Errors:

- InvalidRequestException

  The request was not valid.

  HTTP response code: 400
- ResourceNotFoundException

  A resource with the specified name could not be found.

  HTTP response code: 404
- InternalFailureException

  There was an internal failure.

  HTTP response code: 500
- ServiceUnavailableException

  The service is temporarily unavailable.

  HTTP response code: 503

- ThrottlingException

  The request was denied due to request throttling.

  HTTP response code: 429

DeleteDataset

Deletes the specified data set. You do not have to delete the content of the data set before you perform this operation.

CLI Synopsis:

```
aws iotanalytics  delete-dataset
 --dataset-name <value>
[--cli-input-json <value>]
[--generate-cli-skeleton]
```

cli-input-json format:

```
{
  "datasetName": "string"
}
```

fields:

- datasetName

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the data set to delete.

Output:

None

Errors:

- InvalidRequestException

  The request was not valid.

  HTTP response code: 400
- ResourceNotFoundException

  A resource with the specified name could not be found.

  HTTP response code: 404
- InternalFailureException

  There was an internal failure.

  HTTP response code: 500
- ServiceUnavailableException

  The service is temporarily unavailable.

HTTP response code: 503

- ThrottlingException

  The request was denied due to request throttling.

  HTTP response code: 429

DeleteDatasetContent

Deletes the content of the specified data set.

CLI Synopsis:

```
aws iotanalytics  delete-dataset-content
 --dataset-name <value>
[--version-id <value>]
[--cli-input-json <value>]
[--generate-cli-skeleton]
```

cli-input-json format:

```
{
  "datasetName": "string",
  "versionId": "string"
}
```

fields:

- datasetName

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the data set whose content is deleted.
- versionId

  type: string; (length- max:36 min:7)

  The version of the data set whose content is deleted. You can also use the strings “$LATEST” or “$LATEST_SUCCEEDED” to delete the latest or latest successfully completed data set. If not specified, “$LATEST_SUCCEEDED” is the default.

Output:

None

Errors:

- InvalidRequestException

  The request was not valid.

  HTTP response code: 400
- ResourceNotFoundException

  A resource with the specified name could not be found.

HTTP response code: 404

- InternalFailureException

  There was an internal failure.

  HTTP response code: 500
- ServiceUnavailableException

  The service is temporarily unavailable.

  HTTP response code: 503
- ThrottlingException

  The request was denied due to request throttling.

  HTTP response code: 429

DeleteDatastore

Deletes the specified data store.

CLI Synopsis:

```
aws iotanalytics  delete-datastore
 --datastore-name <value>
[--cli-input-json <value>]
[--generate-cli-skeleton]
```

cli-input-json format:

```
{
  "datastoreName": "string"
}
```

fields:

- datastoreName

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the data store to delete.

Output:

None

Errors:

- InvalidRequestException

  The request was not valid.

  HTTP response code: 400
- ResourceNotFoundException

A resource with the specified name could not be found.

HTTP response code: 404

- InternalFailureException

  There was an internal failure.

  HTTP response code: 500
- ServiceUnavailableException

  The service is temporarily unavailable.

  HTTP response code: 503
- ThrottlingException

  The request was denied due to request throttling.

  HTTP response code: 429

DeletePipeline

Deletes the specified pipeline.

CLI Synopsis:

```
aws iotanalytics  delete-pipeline
 --pipeline-name <value>
[--cli-input-json <value>]
[--generate-cli-skeleton]
```

cli-input-json format:

```
{
  "pipelineName": "string"
}
```

fields:

- pipelineName

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the pipeline to delete.

Output:

None

Errors:

- InvalidRequestException

  The request was not valid.

  HTTP response code: 400
- ResourceNotFoundException

A resource with the specified name could not be found.

HTTP response code: 404

- InternalFailureException

  There was an internal failure.

  HTTP response code: 500
- ServiceUnavailableException

  The service is temporarily unavailable.

  HTTP response code: 503
- ThrottlingException

  The request was denied due to request throttling.

  HTTP response code: 429

DescribeChannel

Retrieves information about a channel.

CLI Synopsis:

```
aws iotanalytics  describe-channel
 --channel-name <value>
[--include-statistics | --no-include-statistics]
[--cli-input-json <value>]
[--generate-cli-skeleton]
```

cli-input-json format:

```
{
  "channelName": "string",
  "includeStatistics": "boolean"
}
```

fields:

- channelName

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the channel whose information is retrieved.
- includeStatistics

  type: boolean

  If true, additional statistical information about the channel is included in the response. This feature cannot be used with a channel whose S3 storage is customer-managed.

Output:

```
{
```

```
  "channel": {
    "name": "string",
    "storage": {
      "serviceManagedS3": {
      },
      "customerManagedS3": {
        "bucket": "string",
        "keyPrefix": "string",
        "roleArn": "string"
      }
    },
    "arn": "string",
    "status": "string",
    "retentionPeriod": {
      "unlimited": "boolean",
      "numberOfDays": "integer"
    },
    "creationTime": "timestamp",
    "lastUpdateTime": "timestamp"
  },
  "statistics": {
    "size": {
      "estimatedSizeInBytes": "double",
      "estimatedOn": "timestamp"
    }
  }
}
```

fields:

- channel

  type: Channel

  An object that contains information about the channel.
- name

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the channel.
- storage

  type: ChannelStorage

  Where channel data is stored. You may choose one of “serviceManagedS3” or “customerManagedS3” storage. If not specified, the default is “serviceManagedS3”. This cannot be changed after creation of the channel.
- serviceManagedS3

  type: ServiceManagedChannelS3Storage

  Use this to store channel data in an S3 bucket managed by the AWS IoT Analytics service. The choice of service-managed or customer-managed S3 storage cannot be changed after creation of the channel.
- customerManagedS3

  type: CustomerManagedChannelS3Storage

  Use this to store channel data in an S3 bucket that you manage. If customer managed storage is selected, the “retentionPeriod” parameter is ignored. The choice of service-managed or customer-managed S3 storage cannot be changed after creation of the channel.

- bucket

  type: string; (length- max:255 min:3); (pattern: ^[a-zA-Z0-9.-_]*$)

  The name of the Amazon S3 bucket in which channel data is stored.
- keyPrefix

  type: string; (length- max:255 min:1); (pattern: ^[a-zA-Z0-9!_.*'()/{}:-]*/$)

  [Optional] The prefix used to create the keys of the channel data objects. Each object in an Amazon S3 bucket has a key that is its unique identifier within the bucket (each object in a bucket has exactly one key). The prefix must end with a '/'.
- roleArn

  type: string; (length- max:2048 min:20)

  The ARN of the role which grants AWS IoT Analytics permission to interact with your Amazon S3 resources.
- arn

  type: string

  The ARN of the channel.
- status

  type: string

  The status of the channel. enum: CREATING | ACTIVE | DELETING
- retentionPeriod

  type: RetentionPeriod

  How long, in days, message data is kept for the channel.
- unlimited

  type: boolean

  If true, message data is kept indefinitely.
- numberOfDays

  type: integer java class: java.lang.Integer range- min:1

  The number of days that message data is kept. The "unlimited" parameter must be false.
- creationTime

  type: timestamp

  When the channel was created.
- lastUpdateTime

  type: timestamp

  When the channel was last updated.
- statistics

  type: ChannelStatistics

  Statistics about the channel. Included if the 'includeStatistics' parameter is set to true in the request.

- size

  type: EstimatedResourceSize

  The estimated size of the channel.
- estimatedOn

  type: timestamp

  The time when the estimate of the size of the resource was made.

Errors:

- InvalidRequestException

  The request was not valid.

  HTTP response code: 400
- ResourceNotFoundException

  A resource with the specified name could not be found.

  HTTP response code: 404
- InternalFailureException

  There was an internal failure.

  HTTP response code: 500
- ServiceUnavailableException

  The service is temporarily unavailable.

  HTTP response code: 503
- ThrottlingException

  The request was denied due to request throttling.

  HTTP response code: 429

DescribeDataset

Retrieves information about a data set.

CLI Synopsis:

```
aws iotanalytics  describe-dataset
 --dataset-name <value>
[--cli-input-json <value>]
[--generate-cli-skeleton]
```

cli-input-json format:

```
{
  "datasetName": "string"
}
```

fields:

- datasetName

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the data set whose information is retrieved.

Output:

```
{
  "dataset": {
    "name": "string",
    "arn": "string",
    "actions": [
      {
        "actionName": "string",
        "queryAction": {
          "sqlQuery": "string",
          "filters": [
            {
              "deltaTime": {
                "offsetSeconds": "integer",
                "timeExpression": "string"
              }
            }
          ]
        },
        "containerAction": {
          "image": "string",
          "executionRoleArn": "string",
          "resourceConfiguration": {
            "computeType": "string",
            "volumeSizeInGB": "integer"
          },
          "variables": [
            {
              "name": "string",
              "stringValue": "string",
              "doubleValue": "double",
              "datasetContentVersionValue": {
                "datasetName": "string"
              },
              "outputFileUriValue": {
                "fileName": "string"
              }
            }
          ]
        }
      }
    ],
    "triggers": [
      {
        "schedule": {
          "expression": "string"
        },
        "dataset": {
          "name": "string"
        }
      }
    ],
    "contentDeliveryRules": [
      {
        "entryName": "string",
```

```
        "destination": {
          "iotEventsDestinationConfiguration": {
            "inputName": "string",
            "roleArn": "string"
          },
          "s3DestinationConfiguration": {
            "bucket": "string",
            "key": "string",
            "glueConfiguration": {
              "tableName": "string",
              "databaseName": "string"
            },
            "roleArn": "string"
          }
        }
      }
    ],
    "status": "string",
    "creationTime": "timestamp",
    "lastUpdateTime": "timestamp",
    "retentionPeriod": {
      "unlimited": "boolean",
      "numberOfDays": "integer"
    },
    "versioningConfiguration": {
      "unlimited": "boolean",
      "maxVersions": "integer"
    }
  }
}
```

fields:

- dataset

  type: Dataset

  An object that contains information about the data set.
- name

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the data set.
- arn

  type: string

  The ARN of the data set.
- actions

  type: list member: DatasetAction

  The “DatasetAction” objects that automatically create the data set contents.
- actionName

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the data set action by which data set contents are automatically created.
- queryAction

  type: SqlQueryDatasetAction

An “SqlQueryDatasetAction” object that uses an SQL query to automatically create data set contents.

- sqlQuery

  type: string

  A SQL query string.
- filters

  type: list member: QueryFilter

  Pre-filters applied to message data.
- deltaTime

  type: DeltaTime

  Used to limit data to that which has arrived since the last execution of the action.
- offsetSeconds

  type: integer java class: java.lang.Integer

  The number of seconds of estimated “in flight” lag time of message data. When you create data set contents using message data from a specified time frame, some message data may still be “in flight” when processing begins, and so will not arrive in time to be processed. Use this field to make allowances for the “in flight” time of your message data, so that data not processed from a previous time frame will be included with the next time frame. Without this, missed message data would be excluded from processing during the next time frame as well, because its timestamp places it within the previous time frame.
- timeExpression

  type: string

  An expression by which the time of the message data may be determined. This may be the name of a timestamp field, or a SQL expression which is used to derive the time the message data was generated.
- containerAction

  type: ContainerDatasetAction

  Information which allows the system to run a containerized application in order to create the data set contents. The application must be in a Docker container along with any needed support libraries.
- image

  type: string; (length- max:255)

  The ARN of the Docker container stored in your account. The Docker container contains an application and needed support libraries and is used to generate data set contents.
- executionRoleArn

  type: string; (length- max:2048 min:20)

  The ARN of the role which gives permission to the system to access needed resources in order to run the “containerAction”. This includes, at minimum, permission to retrieve the data set contents which are the input to the containerized application.
- resourceConfiguration

  type: ResourceConfiguration

  Configuration of the resource which executes the “containerAction”.

- computeType

  type: string

  The type of the compute resource used to execute the “containerAction”. Possible values are: ACU_1 (vCPU=4, memory=16GiB) or ACU_2 (vCPU=8, memory=32GiB). enum: ACU_1 | ACU_2
- volumeSizeInGB

  type: integer range- max:50 min:1

  The size (in GB) of the persistent storage available to the resource instance used to execute the “containerAction” (min: 1, max: 50).
- variables

  type: list member: Variable

  The values of variables used within the context of the execution of the containerized application (basically, parameters passed to the application). Each variable must have a name and a value given by one of “stringValue”, “datasetContentVersionValue”, or “outputFileUriValue”.
- name

  type: string; (length- max:256 min:1)

  The name of the variable.
- stringValue

  type: string; (length- max:1024 min:0)

  The value of the variable as a string.
- datasetContentVersionValue

  type: DatasetContentVersionValue

  The value of the variable as a structure that specifies a data set content version.
- datasetName

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the data set whose latest contents are used as input to the notebook or application.
- outputFileUriValue

  type: OutputFileUriValue

  The value of the variable as a structure that specifies an output file URI.
- fileName

  type: string; (pattern: [w.-]{1,255})

  The URI of the location where data set contents are stored, usually the URI of a file in an S3 bucket.
- triggers

  type: list member: DatasetTrigger

  The “DatasetTrigger” objects that specify when the data set is automatically updated.
- schedule

  type: Schedule

The “Schedule” when the trigger is initiated.

- expression

  type: string

  The expression that defines when to trigger an update. For more information, see Schedule Expressions for Rules in the Amazon CloudWatch Events User Guide.

- dataset

  type: TriggeringDataset

  The data set whose content creation triggers the creation of this data set’s contents.

- name

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the data set whose content generation triggers the new data set content generation.

- contentDeliveryRules

  type: list member: DatasetContentDeliveryRule

  When data set contents are created they are delivered to destinations specified here.

- entryName

  type: string

  The name of the data set content delivery rules entry.

- destination

  type: DatasetContentDeliveryDestination

  The destination to which data set contents are delivered.

- iotEventsDestinationConfiguration

  type: IotEventsDestinationConfiguration

  Configuration information for delivery of data set contents to AWS IoT Events.

- inputName

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z][a-zA-Z0-9_]*$)

  The name of the AWS IoT Events input to which data set contents are delivered.

- roleArn

  type: string; (length- max:2048 min:20)

  The ARN of the role which grants AWS IoT Analytics permission to deliver data set contents to an AWS IoT Events input.

- s3DestinationConfiguration

  type: S3DestinationConfiguration

  Configuration information for delivery of data set contents to Amazon S3.

- bucket

  type: string; (length- max:255 min:3); (pattern: ^[a-zA-Z0-9.-_]*$)

The name of the Amazon S3 bucket to which data set contents are delivered.

- key

  type: string; (length- max:255 min:1); (pattern: ^[a-zA-Z0-9!_.*'()/{}:-]*$)

  The key of the data set contents object. Each object in an Amazon S3 bucket has a key that is its unique identifier within the bucket (each object in a bucket has exactly one key). To produce a unique key, you can use "!{iotanalytics:scheduleTime}" to insert the time of the scheduled SQL query run, or "!{iotanalytics:versioned} to insert a unique hash identifying the data set, for example: "/DataSet/!{iotanalytics:scheduleTime}/!{iotanalytics:versioned}.csv".

- glueConfiguration

  type: GlueConfiguration

  Configuration information for coordination with the AWS Glue ETL (extract, transform and load) service.

- tableName

  type: string; (length- max:150 min:1); (pattern: [u0020-uD7FFuE000-uFFFDuD800uDC00-uDBFFuDFFFt]*)

  The name of the table in your AWS Glue Data Catalog which is used to perform the ETL (extract, transform and load) operations. (An AWS Glue Data Catalog table contains partitioned data and descriptions of data sources and targets.)

- databaseName

  type: string; (length- max:150 min:1); (pattern: [u0020-uD7FFuE000-uFFFDuD800uDC00-uDBFFuDFFFt]*)

  The name of the database in your AWS Glue Data Catalog in which the table is located. (An AWS Glue Data Catalog database contains Glue Data tables.)

- roleArn

  type: string; (length- max:2048 min:20)

  The ARN of the role which grants AWS IoT Analytics permission to interact with your Amazon S3 and AWS Glue resources.

- status

  type: string

  The status of the data set. enum: CREATING | ACTIVE | DELETING

- creationTime

  type: timestamp

  When the data set was created.

- lastUpdateTime

  type: timestamp

  The last time the data set was updated.

- retentionPeriod

  type: RetentionPeriod

  [Optional] How long, in days, message data is kept for the data set.

- unlimited

  type: boolean

  If true, message data is kept indefinitely.
- numberOfDays

  type: integer java class: java.lang.Integer range- min:1

  The number of days that message data is kept. The “unlimited” parameter must be false.
- versioningConfiguration

  type: VersioningConfiguration

  [Optional] How many versions of data set contents are kept. If not specified or set to null, only the latest version plus the latest succeeded version (if they are different) are kept for the time period specified by the “retentionPeriod” parameter. (For more information, see https://docs.aws.amazon.com/iotanalytics/latest/userguide/getting-started.html#aws-iot-analytics-dataset-versions)
- unlimited

  type: boolean

  If true, unlimited versions of data set contents will be kept.
- maxVersions

  type: integer java class: java.lang.Integer range- max:1000 min:1

  How many versions of data set contents will be kept. The “unlimited” parameter must be false.

Errors:

- InvalidRequestException

  The request was not valid.

  HTTP response code: 400
- ResourceNotFoundException

  A resource with the specified name could not be found.

  HTTP response code: 404
- InternalFailureException

  There was an internal failure.

  HTTP response code: 500
- ServiceUnavailableException

  The service is temporarily unavailable.

  HTTP response code: 503
- ThrottlingException

  The request was denied due to request throttling.

  HTTP response code: 429

DescribeDatastore

Retrieves information about a data store.

CLI Synopsis:

```
aws iotanalytics  describe-datastore
 --datastore-name <value>
[--include-statistics | --no-include-statistics]
[--cli-input-json <value>]
[--generate-cli-skeleton]
```

cli-input-json format:

```
{
  "datastoreName": "string",
  "includeStatistics": "boolean"
}
```

fields:

- datastoreName

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the data store
- includeStatistics

  type: boolean

  If true, additional statistical information about the data store is included in the response. This feature cannot be used with a data store whose S3 storage is customer-managed.

Output:

```
{
  "datastore": {
    "name": "string",
    "storage": {
      "serviceManagedS3": {
      },
      "customerManagedS3": {
        "bucket": "string",
        "keyPrefix": "string",
        "roleArn": "string"
      }
    },
    "arn": "string",
    "status": "string",
    "retentionPeriod": {
      "unlimited": "boolean",
      "numberOfDays": "integer"
    },
    "creationTime": "timestamp",
    "lastUpdateTime": "timestamp"
  },
  "statistics": {
    "size": {
```

```
      "estimatedSizeInBytes": "double",
      "estimatedOn": "timestamp"
    }
  }
}
```

fields:

- datastore

  type: Datastore

  Information about the data store.
- name

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the data store.
- storage

  type: DatastoreStorage

  Where data store data is stored. You may choose one of “serviceManagedS3” or “customerManagedS3” storage. If not specified, the default is “serviceManagedS3”. This cannot be changed after the data store is created.
- serviceManagedS3

  type: ServiceManagedDatastoreS3Storage

  Use this to store data store data in an S3 bucket managed by the AWS IoT Analytics service. The choice of service-managed or customer-managed S3 storage cannot be changed after creation of the data store.
- customerManagedS3

  type: CustomerManagedDatastoreS3Storage

  Use this to store data store data in an S3 bucket that you manage. When customer managed storage is selected, the “retentionPeriod” parameter is ignored. The choice of service-managed or customer-managed S3 storage cannot be changed after creation of the data store.
- bucket

  type: string; (length- max:255 min:3); (pattern: ^[a-zA-Z0-9.-_]*$)

  The name of the Amazon S3 bucket in which data store data is stored.
- keyPrefix

  type: string; (length- max:255 min:1); (pattern: ^[a-zA-Z0-9!_.*'()/{}:-]*/$)

  [Optional] The prefix used to create the keys of the data store data objects. Each object in an Amazon S3 bucket has a key that is its unique identifier within the bucket (each object in a bucket has exactly one key). The prefix must end with a ‘/’.
- roleArn

  type: string; (length- max:2048 min:20)

  The ARN of the role which grants AWS IoT Analytics permission to interact with your Amazon S3 resources.
- arn

type: string

The ARN of the data store.

- status

  type: string

  The status of a data store: CREATING The data store is being created. ACTIVE The data store has been created and can be used. DELETING The data store is being deleted. enum: CREATING | ACTIVE | DELETING

- retentionPeriod

  type: RetentionPeriod

  How long, in days, message data is kept for the data store. When “customerManagedS3” storage is selected, this parameter is ignored.

- unlimited

  type: boolean

  If true, message data is kept indefinitely.

- numberOfDays

  type: integer java class: java.lang.Integer range- min:1

  The number of days that message data is kept. The “unlimited” parameter must be false.

- creationTime

  type: timestamp

  When the data store was created.

- lastUpdateTime

  type: timestamp

  The last time the data store was updated.

- statistics

  type: DatastoreStatistics

  Additional statistical information about the data store. Included if the ‘includeStatistics’ parameter is set to true in the request.

- size

  type: EstimatedResourceSize

  The estimated size of the data store.

- estimatedOn

  type: timestamp

  The time when the estimate of the size of the resource was made.

Errors:

- InvalidRequestException

The request was not valid.

HTTP response code: 400

- ResourceNotFoundException

  A resource with the specified name could not be found.

  HTTP response code: 404
- InternalFailureException

  There was an internal failure.

  HTTP response code: 500
- ServiceUnavailableException

  The service is temporarily unavailable.

  HTTP response code: 503
- ThrottlingException

  The request was denied due to request throttling.

  HTTP response code: 429

DescribeLoggingOptions

Retrieves the current settings of the AWS IoT Analytics logging options.

CLI Synopsis:

```
aws iotanalytics  describe-logging-options
[--cli-input-json <value>]
[--generate-cli-skeleton]
```

cli-input-json format:

```
{
}
```

fields:

Output:

```
{
  "loggingOptions": {
    "roleArn": "string",
    "level": "string",
    "enabled": "boolean"
  }
}
```

fields:

- loggingOptions

type: LoggingOptions

The current settings of the AWS IoT Analytics logging options.

- roleArn

  type: string; (length- max:2048 min:20)

  The ARN of the role that grants permission to AWS IoT Analytics to perform logging.
- level

  type: string

  The logging level. Currently, only “ERROR” is supported. enum: ERROR
- enabled

  type: boolean

  If true, logging is enabled for AWS IoT Analytics.

Errors:

- InvalidRequestException

  The request was not valid.

  HTTP response code: 400
- ResourceNotFoundException

  A resource with the specified name could not be found.

  HTTP response code: 404
- InternalFailureException

  There was an internal failure.

  HTTP response code: 500
- ServiceUnavailableException

  The service is temporarily unavailable.

  HTTP response code: 503
- ThrottlingException

  The request was denied due to request throttling.

  HTTP response code: 429

DescribePipeline

Retrieves information about a pipeline.

CLI Synopsis:

```
aws iotanalytics  describe-pipeline
```

```
 --pipeline-name <value>
[--cli-input-json <value>]
[--generate-cli-skeleton]
```

cli-input-json format:

```
{
  "pipelineName": "string"
}
```

fields:

- pipelineName

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the pipeline whose information is retrieved.

Output:

```
{
  "pipeline": {
    "name": "string",
    "arn": "string",
    "activities": [
      {
        "channel": {
          "name": "string",
          "channelName": "string",
          "next": "string"
        },
        "lambda": {
          "name": "string",
          "lambdaName": "string",
          "batchSize": "integer",
          "next": "string"
        },
        "datastore": {
          "name": "string",
          "datastoreName": "string"
        },
        "addAttributes": {
          "name": "string",
          "attributes": {
            "string": "string"
          },
          "next": "string"
        },
        "removeAttributes": {
          "name": "string",
          "attributes": [
            "string"
          ],
          "next": "string"
        },
        "selectAttributes": {
          "name": "string",
          "attributes": [
            "string"
          ],
          "next": "string"
        },
```

```
        "filter": {
          "name": "string",
          "filter": "string",
          "next": "string"
        },
        "math": {
          "name": "string",
          "attribute": "string",
          "math": "string",
          "next": "string"
        },
        "deviceRegistryEnrich": {
          "name": "string",
          "attribute": "string",
          "thingName": "string",
          "roleArn": "string",
          "next": "string"
        },
        "deviceShadowEnrich": {
          "name": "string",
          "attribute": "string",
          "thingName": "string",
          "roleArn": "string",
          "next": "string"
        }
      }
    ],
    "reprocessingSummaries": [
      {
        "id": "string",
        "status": "string",
        "creationTime": "timestamp"
      }
    ],
    "creationTime": "timestamp",
    "lastUpdateTime": "timestamp"
  }
}
```

fields:

- pipeline

  type: Pipeline

  A “Pipeline” object that contains information about the pipeline.
- name

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the pipeline.
- arn

  type: string

  The ARN of the pipeline.
- activities

  type: list member: PipelineActivity

  The activities that perform transformations on the messages.
- channel

type: ChannelActivity

Determines the source of the messages to be processed.

- name

  type: string; (length- max:128 min:1)

  The name of the 'channel' activity.
- channelName

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the channel from which the messages are processed.
- next

  type: string; (length- max:128 min:1)

  The next activity in the pipeline.
- lambda

  type: LambdaActivity

  Runs a Lambda function to modify the message.
- name

  type: string; (length- max:128 min:1)

  The name of the 'lambda' activity.
- lambdaName

  type: string; (length- max:64 min:1); (pattern: ^[a-zA-Z0-9_-]+$)

  The name of the Lambda function that is run on the message.
- batchSize

  type: integer java class: java.lang.Integer range- max:1000 min:1

  The number of messages passed to the Lambda function for processing. The AWS Lambda function must be able to process all of these messages within five minutes, which is the maximum timeout duration for Lambda functions.
- next

  type: string; (length- max:128 min:1)

  The next activity in the pipeline.
- datastore

  type: DatastoreActivity

  Specifies where to store the processed message data.
- name

  type: string; (length- max:128 min:1)

  The name of the 'datastore' activity.
- datastoreName

type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

The name of the data store where processed messages are stored.

- addAttributes

  type: AddAttributesActivity

  Adds other attributes based on existing attributes in the message.

- name

  type: string; (length- max:128 min:1)

  The name of the 'addAttributes' activity.

- attributes

  type: map key: AttributeName value: AttributeName

  A list of 1-50 "AttributeNameMapping" objects that map an existing attribute to a new attribute. The existing attributes remain in the message, so if you want to remove the originals, use "RemoveAttributeActivity".

- next

  type: string; (length- max:128 min:1)

  The next activity in the pipeline.

- removeAttributes

  type: RemoveAttributesActivity

  Removes attributes from a message.

- name

  type: string; (length- max:128 min:1)

  The name of the 'removeAttributes' activity.

- attributes

  type: list member: AttributeName

  A list of 1-50 attributes to remove from the message.

- next

  type: string; (length- max:128 min:1)

  The next activity in the pipeline.

- selectAttributes

  type: SelectAttributesActivity

  Creates a new message using only the specified attributes from the original message.

- name

  type: string; (length- max:128 min:1)

  The name of the 'selectAttributes' activity.

- attributes

type: list member: AttributeName

A list of the attributes to select from the message.

- next

  type: string; (length- max:128 min:1)

  The next activity in the pipeline.
- filter

  type: FilterActivity

  Filters a message based on its attributes.
- name

  type: string; (length- max:128 min:1)

  The name of the 'filter' activity.
- filter

  type: string; (length- max:256 min:1)

  An expression that looks like a SQL WHERE clause that must return a Boolean value.
- next

  type: string; (length- max:128 min:1)

  The next activity in the pipeline.
- math

  type: MathActivity

  Computes an arithmetic expression using the message's attributes and adds it to the message.
- name

  type: string; (length- max:128 min:1)

  The name of the 'math' activity.
- attribute

  type: string; (length- max:256 min:1)

  The name of the attribute that contains the result of the math operation.
- math

  type: string; (length- max:256 min:1)

  An expression that uses one or more existing attributes and must return an integer value.
- next

  type: string; (length- max:128 min:1)

  The next activity in the pipeline.
- deviceRegistryEnrich

  type: DeviceRegistryEnrichActivity

Adds data from the AWS IoT device registry to your message.

- name

  type: string; (length- max:128 min:1)

  The name of the 'deviceRegistryEnrich' activity.
- attribute

  type: string; (length- max:256 min:1)

  The name of the attribute that is added to the message.
- thingName

  type: string; (length- max:256 min:1)

  The name of the IoT device whose registry information is added to the message.
- roleArn

  type: string; (length- max:2048 min:20)

  The ARN of the role that allows access to the device's registry information.
- next

  type: string; (length- max:128 min:1)

  The next activity in the pipeline.
- deviceShadowEnrich

  type: DeviceShadowEnrichActivity

  Adds information from the AWS IoT Device Shadows service to a message.
- name

  type: string; (length- max:128 min:1)

  The name of the 'deviceShadowEnrich' activity.
- attribute

  type: string; (length- max:256 min:1)

  The name of the attribute that is added to the message.
- thingName

  type: string; (length- max:256 min:1)

  The name of the IoT device whose shadow information is added to the message.
- roleArn

  type: string; (length- max:2048 min:20)

  The ARN of the role that allows access to the device's shadow.
- next

  type: string; (length- max:128 min:1)

  The next activity in the pipeline.

- reprocessingSummaries

  type: list member: ReprocessingSummary

  A summary of information about the pipeline reprocessing.
- id

  type: string

  The 'reprocessingId' returned by "StartPipelineReprocessing".
- status

  type: string

  The status of the pipeline reprocessing. enum: RUNNING | SUCCEEDED | CANCELLED | FAILED
- creationTime

  type: timestamp

  The time the pipeline reprocessing was created.
- creationTime

  type: timestamp

  When the pipeline was created.
- lastUpdateTime

  type: timestamp

  The last time the pipeline was updated.

Errors:

- InvalidRequestException

  The request was not valid.

  HTTP response code: 400
- ResourceNotFoundException

  A resource with the specified name could not be found.

  HTTP response code: 404
- InternalFailureException

  There was an internal failure.

  HTTP response code: 500
- ServiceUnavailableException

  The service is temporarily unavailable.

  HTTP response code: 503
- ThrottlingException

  The request was denied due to request throttling.

  HTTP response code: 429

GetDatasetContent

Retrieves the contents of a data set as pre-signed URIs.

CLI Synopsis:

```
aws iotanalytics  get-dataset-content
 --dataset-name <value>
[--version-id <value>]
[--cli-input-json <value>]
[--generate-cli-skeleton]
```

cli-input-json format:

```
{
  "datasetName": "string",
  "versionId": "string"
}
```

fields:

- datasetName

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the data set whose contents are retrieved.
- versionId

  type: string; (length- max:36 min:7)

  The version of the data set whose contents are retrieved. You can also use the strings “$LATEST” or “$LATEST_SUCCEEDED” to retrieve the contents of the latest or latest successfully completed data set. If not specified, “$LATEST_SUCCEEDED” is the default.

Output:

```
{
  "entries": [
    {
      "entryName": "string",
      "dataURI": "string"
    }
  ],
  "timestamp": "timestamp",
  "status": {
    "state": "string",
    "reason": "string"
  }
}
```

fields:

- entries

  type: list member: DatasetEntry

  A list of “DatasetEntry” objects.
- entryName

type: string

The name of the data set item.

- dataURI

  type: string

  The pre-signed URI of the data set item.
- timestamp

  type: timestamp

  The time when the request was made.
- status

  type: DatasetContentStatus

  The status of the data set content.
- state

  type: string

  The state of the data set contents. Can be one of “READY”, “CREATING”, “SUCCEEDED” or “FAILED”. enum: CREATING | SUCCEEDED | FAILED
- reason

  type: string

  The reason the data set contents are in this state.

Errors:

- InvalidRequestException

  The request was not valid.

  HTTP response code: 400
- ResourceNotFoundException

  A resource with the specified name could not be found.

  HTTP response code: 404
- InternalFailureException

  There was an internal failure.

  HTTP response code: 500
- ServiceUnavailableException

  The service is temporarily unavailable.

  HTTP response code: 503
- ThrottlingException

  The request was denied due to request throttling.

  HTTP response code: 429

ListChannels

Retrieves a list of channels.

CLI Synopsis:

```
aws iotanalytics  list-channels
[--next-token <value>]
[--max-results <value>]
[--cli-input-json <value>]
[--generate-cli-skeleton]
```

cli-input-json format:

```
{
  "nextToken": "string",
  "maxResults": "integer"
}
```

fields:

- nextToken

  type: string

  The token for the next set of results.
- maxResults

  type: integer java class: java.lang.Integer range- max:250 min:1

  The maximum number of results to return in this request. The default value is 100.

Output:

```
{
  "channelSummaries": [
    {
      "channelName": "string",
      "channelStorage": {
        "serviceManagedS3": {
        },
        "customerManagedS3": {
          "bucket": "string",
          "keyPrefix": "string",
          "roleArn": "string"
        }
      },
      "status": "string",
      "creationTime": "timestamp",
      "lastUpdateTime": "timestamp"
    }
  ],
  "nextToken": "string"
}
```

fields:

- channelSummaries

type: list member: ChannelSummary

A list of “ChannelSummary” objects.

- channelName

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the channel.
- channelStorage

  type: ChannelStorageSummary

  Where channel data is stored.
- serviceManagedS3

  type: ServiceManagedChannelS3StorageSummary

  Used to store channel data in an S3 bucket managed by the AWS IoT Analytics service.
- customerManagedS3

  type: CustomerManagedChannelS3StorageSummary

  Used to store channel data in an S3 bucket that you manage.
- bucket

  type: string; (length- max:255 min:3); (pattern: ^[a-zA-Z0-9.-_]*$)

  The name of the Amazon S3 bucket in which channel data is stored.
- keyPrefix

  type: string; (length- max:255 min:1); (pattern: ^[a-zA-Z0-9!_.*'()/{}:-]*/$)

  [Optional] The prefix used to create the keys of the channel data objects. Each object in an Amazon S3 bucket has a key that is its unique identifier within the bucket (each object in a bucket has exactly one key). The prefix must end with a ‘/’.
- roleArn

  type: string; (length- max:2048 min:20)

  The ARN of the role which grants AWS IoT Analytics permission to interact with your Amazon S3 resources.
- status

  type: string

  The status of the channel. enum: CREATING | ACTIVE | DELETING
- creationTime

  type: timestamp

  When the channel was created.
- lastUpdateTime

  type: timestamp

  The last time the channel was updated.
- nextToken

type: string

The token to retrieve the next set of results, or null if there are no more results.

Errors:

- InvalidRequestException

  The request was not valid.

  HTTP response code: 400
- InternalFailureException

  There was an internal failure.

  HTTP response code: 500
- ServiceUnavailableException

  The service is temporarily unavailable.

  HTTP response code: 503
- ThrottlingException

  The request was denied due to request throttling.

  HTTP response code: 429

ListDatasetContents

Lists information about data set contents that have been created.

CLI Synopsis:

```
aws iotanalytics  list-dataset-contents
 --dataset-name <value>
[--next-token <value>]
[--max-results <value>]
[--scheduled-on-or-after <value>]
[--scheduled-before <value>]
[--cli-input-json <value>]
[--generate-cli-skeleton]
```

cli-input-json format:

```
{
  "datasetName": "string",
  "nextToken": "string",
  "maxResults": "integer",
  "scheduledOnOrAfter": "timestamp",
  "scheduledBefore": "timestamp"
}
```

fields:

- datasetName

type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

The name of the data set whose contents information you want to list.

- nextToken

  type: string

  The token for the next set of results.

- maxResults

  type: integer java class: java.lang.Integer range- max:250 min:1

  The maximum number of results to return in this request.

- scheduledOnOrAfter

  type: timestamp

  A filter to limit results to those data set contents whose creation is scheduled on or after the given time. See the field triggers.schedule in the CreateDataset request. (timestamp)

- scheduledBefore

  type: timestamp

  A filter to limit results to those data set contents whose creation is scheduled before the given time. See the field triggers.schedule in the CreateDataset request. (timestamp)

Output:

```
{
  "datasetContentSummaries": [
    {
      "version": "string",
      "status": {
        "state": "string",
        "reason": "string"
      },
      "creationTime": "timestamp",
      "scheduleTime": "timestamp",
      "completionTime": "timestamp"
    }
  ],
  "nextToken": "string"
}
```

fields:

- datasetContentSummaries

  type: list member: DatasetContentSummary

  Summary information about data set contents that have been created.

- version

  type: string; (length- max:36 min:7)

  The version of the data set contents.

- status

type: DatasetContentStatus

The status of the data set contents.

- state

  type: string

  The state of the data set contents. Can be one of “READY”, “CREATING”, “SUCCEEDED” or “FAILED”. enum: CREATING | SUCCEEDED | FAILED

- reason

  type: string

  The reason the data set contents are in this state.

- creationTime

  type: timestamp

  The actual time the creation of the data set contents was started.

- scheduleTime

  type: timestamp

  The time the creation of the data set contents was scheduled to start.

- completionTime

  type: timestamp

  The time the dataset content status was updated to SUCCEEDED or FAILED.

- nextToken

  type: string

  The token to retrieve the next set of results, or null if there are no more results.

Errors:

- InvalidRequestException

  The request was not valid.

  HTTP response code: 400

- InternalFailureException

  There was an internal failure.

  HTTP response code: 500

- ServiceUnavailableException

  The service is temporarily unavailable.

  HTTP response code: 503

- ThrottlingException

  The request was denied due to request throttling.

  HTTP response code: 429

- ResourceNotFoundException

  A resource with the specified name could not be found.

  HTTP response code: 404

ListDatasets

Retrieves information about data sets.

CLI Synopsis:

```
aws iotanalytics  list-datasets
[--next-token <value>]
[--max-results <value>]
[--cli-input-json <value>]
[--generate-cli-skeleton]
```

cli-input-json format:

```
{
  "nextToken": "string",
  "maxResults": "integer"
}
```

fields:

- nextToken

  type: string

  The token for the next set of results.
- maxResults

  type: integer java class: java.lang.Integer range- max:250 min:1

  The maximum number of results to return in this request. The default value is 100.

Output:

```
{
  "datasetSummaries": [
    {
      "datasetName": "string",
      "status": "string",
      "creationTime": "timestamp",
      "lastUpdateTime": "timestamp",
      "triggers": [
        {
          "schedule": {
            "expression": "string"
          },
          "dataset": {
            "name": "string"
          }
        }
      ],
      "actions": [
        {
```

```
          "actionName": "string",
          "actionType": "string"
        }
      ]
    }
  ],
  "nextToken": "string"
}
```

fields:

- datasetSummaries

  type: list member: DatasetSummary

  A list of “DatasetSummary” objects.
- datasetName

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the data set.
- status

  type: string

  The status of the data set. enum: CREATING | ACTIVE | DELETING
- creationTime

  type: timestamp

  The time the data set was created.
- lastUpdateTime

  type: timestamp

  The last time the data set was updated.
- triggers

  type: list member: DatasetTrigger

  A list of triggers. A trigger causes data set content to be populated at a specified time interval or when another data set is populated. The list of triggers can be empty or contain up to five DataSetTrigger objects
- schedule

  type: Schedule

  The “Schedule” when the trigger is initiated.
- expression

  type: string

  The expression that defines when to trigger an update. For more information, see Schedule Expressions for Rules in the Amazon CloudWatch Events User Guide.
- dataset

  type: TriggeringDataset

  The data set whose content creation triggers the creation of this data set’s contents.

- name

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the data set whose content generation triggers the new data set content generation.
- actions

  type: list member: DatasetActionSummary

  A list of “DataActionSummary” objects.
- actionName

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the action which automatically creates the data set’s contents.
- actionType

  type: string

  The type of action by which the data set’s contents are automatically created. enum: QUERY | CONTAINER
- nextToken

  type: string

  The token to retrieve the next set of results, or null if there are no more results.

Errors:

- InvalidRequestException

  The request was not valid.

  HTTP response code: 400
- InternalFailureException

  There was an internal failure.

  HTTP response code: 500
- ServiceUnavailableException

  The service is temporarily unavailable.

  HTTP response code: 503
- ThrottlingException

  The request was denied due to request throttling.

  HTTP response code: 429

ListDatastores

Retrieves a list of data stores.

CLI Synopsis:

```
aws iotanalytics  list-datastores
[--next-token <value>]
[--max-results <value>]
[--cli-input-json <value>]
[--generate-cli-skeleton]
```

cli-input-json format:

```
{
  "nextToken": "string",
  "maxResults": "integer"
}
```

fields:

- nextToken

  type: string

  The token for the next set of results.
- maxResults

  type: integer java class: java.lang.Integer range- max:250 min:1

  The maximum number of results to return in this request. The default value is 100.

Output:

```
{
  "datastoreSummaries": [
    {
      "datastoreName": "string",
      "datastoreStorage": {
        "serviceManagedS3": {
        },
        "customerManagedS3": {
          "bucket": "string",
          "keyPrefix": "string",
          "roleArn": "string"
        }
      },
      "status": "string",
      "creationTime": "timestamp",
      "lastUpdateTime": "timestamp"
    }
  ],
  "nextToken": "string"
}
```

fields:

- datastoreSummaries

  type: list member: DatastoreSummary

  A list of “DatastoreSummary” objects.
- datastoreName

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

The name of the data store.

- datastoreStorage

  type: DatastoreStorageSummary

  Where data store data is stored.

- serviceManagedS3

  type: ServiceManagedDatastoreS3StorageSummary

  Used to store data store data in an S3 bucket managed by the AWS IoT Analytics service.

- customerManagedS3

  type: CustomerManagedDatastoreS3StorageSummary

  Used to store data store data in an S3 bucket that you manage.

- bucket

  type: string; (length- max:255 min:3); (pattern: ^[a-zA-Z0-9.-_]*$)

  The name of the Amazon S3 bucket in which data store data is stored.

- keyPrefix

  type: string; (length- max:255 min:1); (pattern: ^[a-zA-Z0-9!_.*'()/{}:-]*/$)

  [Optional] The prefix used to create the keys of the data store data objects. Each object in an Amazon S3 bucket has a key that is its unique identifier within the bucket (each object in a bucket has exactly one key). The prefix must end with a '/'.

- roleArn

  type: string; (length- max:2048 min:20)

  The ARN of the role which grants AWS IoT Analytics permission to interact with your Amazon S3 resources.

- status

  type: string

  The status of the data store. enum: CREATING | ACTIVE | DELETING

- creationTime

  type: timestamp

  When the data store was created.

- lastUpdateTime

  type: timestamp

  The last time the data store was updated.

- nextToken

  type: string

  The token to retrieve the next set of results, or null if there are no more results.

Errors:

- InvalidRequestException

  The request was not valid.

  HTTP response code: 400
- InternalFailureException

  There was an internal failure.

  HTTP response code: 500
- ServiceUnavailableException

  The service is temporarily unavailable.

  HTTP response code: 503
- ThrottlingException

  The request was denied due to request throttling.

  HTTP response code: 429

ListPipelines

Retrieves a list of pipelines.

CLI Synopsis:

```
aws iotanalytics  list-pipelines
[--next-token <value>]
[--max-results <value>]
[--cli-input-json <value>]
[--generate-cli-skeleton]
```

cli-input-json format:

```
{
  "nextToken": "string",
  "maxResults": "integer"
}
```

fields:

- nextToken

  type: string

  The token for the next set of results.
- maxResults

  type: integer java class: java.lang.Integer range- max:250 min:1

  The maximum number of results to return in this request. The default value is 100.

Output:

```
{
```

```
  "pipelineSummaries": [
    {
      "pipelineName": "string",
      "reprocessingSummaries": [
        {
          "id": "string",
          "status": "string",
          "creationTime": "timestamp"
        }
      ],
      "creationTime": "timestamp",
      "lastUpdateTime": "timestamp"
    }
  ],
  "nextToken": "string"
}
```

fields:

- pipelineSummaries

  type: list member: PipelineSummary

  A list of “PipelineSummary” objects.
- pipelineName

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the pipeline.
- reprocessingSummaries

  type: list member: ReprocessingSummary

  A summary of information about the pipeline reprocessing.
- id

  type: string

  The ‘reprocessingId’ returned by “StartPipelineReprocessing”.
- status

  type: string

  The status of the pipeline reprocessing. enum: RUNNING | SUCCEEDED | CANCELLED | FAILED
- creationTime

  type: timestamp

  The time the pipeline reprocessing was created.
- creationTime

  type: timestamp

  When the pipeline was created.
- lastUpdateTime

  type: timestamp

  When the pipeline was last updated.
- nextToken

type: string

The token to retrieve the next set of results, or null if there are no more results.

Errors:

- InvalidRequestException

  The request was not valid.

  HTTP response code: 400
- InternalFailureException

  There was an internal failure.

  HTTP response code: 500
- ServiceUnavailableException

  The service is temporarily unavailable.

  HTTP response code: 503
- ThrottlingException

  The request was denied due to request throttling.

  HTTP response code: 429

ListTagsForResource

Lists the tags (metadata) which you have assigned to the resource.

CLI Synopsis:

```
aws iotanalytics  list-tags-for-resource
 --resource-arn <value>
[--cli-input-json <value>]
[--generate-cli-skeleton]
```

cli-input-json format:

```
{
  "resourceArn": "string"
}
```

fields:

- resourceArn

  type: string; (length- max:2048 min:20)

  The ARN of the resource whose tags you want to list.

Output:

```
{
```

```
  "tags": [
    {
      "key": "string",
      "value": "string"
    }
  ]
}
```

fields:

- tags

  type: list member: Tag

  The tags (metadata) which you have assigned to the resource.
- key

  type: string; (length- max:128 min:1)

  The tag's key.
- value

  type: string; (length- max:128 min:1)

  The tag's value.

Errors:

- InvalidRequestException

  The request was not valid.

  HTTP response code: 400
- InternalFailureException

  There was an internal failure.

  HTTP response code: 500
- ServiceUnavailableException

  The service is temporarily unavailable.

  HTTP response code: 503
- ThrottlingException

  The request was denied due to request throttling.

  HTTP response code: 429
- LimitExceededException

  The command caused an internal limit to be exceeded.

  HTTP response code: 410
- ResourceNotFoundException

  A resource with the specified name could not be found.

  HTTP response code: 404

PutLoggingOptions

Sets or updates the AWS IoT Analytics logging options. Note that if you update the value of any loggingOptions field, it takes up to one minute for the change to take effect. Also, if you change the policy attached to the role you specified in the roleArn field (for example, to correct an invalid policy) it takes up to 5 minutes for that change to take effect.

CLI Synopsis:

```
aws iotanalytics  put-logging-options
 --logging-options <value>
[--cli-input-json <value>]
[--generate-cli-skeleton]
```

cli-input-json format:

```
{
  "loggingOptions": {
    "roleArn": "string",
    "level": "string",
    "enabled": "boolean"
  }
}
```

fields:

- loggingOptions

  type: LoggingOptions

  The new values of the AWS IoT Analytics logging options.
- roleArn

  type: string; (length- max:2048 min:20)

  The ARN of the role that grants permission to AWS IoT Analytics to perform logging.
- level

  type: string

  The logging level. Currently, only “ERROR” is supported. enum: ERROR
- enabled

  type: boolean

  If true, logging is enabled for AWS IoT Analytics.

Output:

None

Errors:

- InvalidRequestException

  The request was not valid.

  HTTP response code: 400

- InternalFailureException

  There was an internal failure.

  HTTP response code: 500
- ServiceUnavailableException

  The service is temporarily unavailable.

  HTTP response code: 503
- ThrottlingException

  The request was denied due to request throttling.

  HTTP response code: 429

RunPipelineActivity

Simulates the results of running a pipeline activity on a message payload.

CLI Synopsis:

```
aws iotanalytics  run-pipeline-activity
 --pipeline-activity <value>
 --payloads <value>
[--cli-input-json <value>]
[--generate-cli-skeleton]
```

cli-input-json format:

```
{
  "pipelineActivity": {
    "channel": {
      "name": "string",
      "channelName": "string",
      "next": "string"
    },
    "lambda": {
      "name": "string",
      "lambdaName": "string",
      "batchSize": "integer",
      "next": "string"
    },
    "datastore": {
      "name": "string",
      "datastoreName": "string"
    },
    "addAttributes": {
      "name": "string",
      "attributes": {
        "string": "string"
      },
      "next": "string"
    },
    "removeAttributes": {
      "name": "string",
      "attributes": [
        "string"
      ],
      "next": "string"
```

```
    },
    "selectAttributes": {
      "name": "string",
      "attributes": [
        "string"
      ],
      "next": "string"
    },
    "filter": {
      "name": "string",
      "filter": "string",
      "next": "string"
    },
    "math": {
      "name": "string",
      "attribute": "string",
      "math": "string",
      "next": "string"
    },
    "deviceRegistryEnrich": {
      "name": "string",
      "attribute": "string",
      "thingName": "string",
      "roleArn": "string",
      "next": "string"
    },
    "deviceShadowEnrich": {
      "name": "string",
      "attribute": "string",
      "thingName": "string",
      "roleArn": "string",
      "next": "string"
    }
  },
  "payloads": [
    "blob"
  ]
}
```

fields:

- pipelineActivity

  type: PipelineActivity

  The pipeline activity that is run. This must not be a 'channel' activity or a 'datastore' activity because these activities are used in a pipeline only to load the original message and to store the (possibly) transformed message. If a 'lambda' activity is specified, only short-running Lambda functions (those with a timeout of less than 30 seconds or less) can be used.

- channel

  type: ChannelActivity

  Determines the source of the messages to be processed.

- name

  type: string; (length- max:128 min:1)

  The name of the 'channel' activity.

- channelName

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

The name of the channel from which the messages are processed.

- next

  type: string; (length- max:128 min:1)

  The next activity in the pipeline.

- lambda

  type: LambdaActivity

  Runs a Lambda function to modify the message.

- name

  type: string; (length- max:128 min:1)

  The name of the 'lambda' activity.

- lambdaName

  type: string; (length- max:64 min:1); (pattern: ^[a-zA-Z0-9_-]+$)

  The name of the Lambda function that is run on the message.

- batchSize

  type: integer java class: java.lang.Integer range- max:1000 min:1

  The number of messages passed to the Lambda function for processing. The AWS Lambda function must be able to process all of these messages within five minutes, which is the maximum timeout duration for Lambda functions.

- next

  type: string; (length- max:128 min:1)

  The next activity in the pipeline.

- datastore

  type: DatastoreActivity

  Specifies where to store the processed message data.

- name

  type: string; (length- max:128 min:1)

  The name of the 'datastore' activity.

- datastoreName

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the data store where processed messages are stored.

- addAttributes

  type: AddAttributesActivity

  Adds other attributes based on existing attributes in the message.

- name

  type: string; (length- max:128 min:1)

The name of the ‘addAttributes’ activity.

- attributes

  type: map key: AttributeName value: AttributeName

  A list of 1-50 “AttributeNameMapping” objects that map an existing attribute to a new attribute. The existing attributes remain in the message, so if you want to remove the originals, use “RemoveAttributeActivity”.

- next

  type: string; (length- max:128 min:1)

  The next activity in the pipeline.

- removeAttributes

  type: RemoveAttributesActivity

  Removes attributes from a message.

- name

  type: string; (length- max:128 min:1)

  The name of the ‘removeAttributes’ activity.

- attributes

  type: list member: AttributeName

  A list of 1-50 attributes to remove from the message.

- next

  type: string; (length- max:128 min:1)

  The next activity in the pipeline.

- selectAttributes

  type: SelectAttributesActivity

  Creates a new message using only the specified attributes from the original message.

- name

  type: string; (length- max:128 min:1)

  The name of the ‘selectAttributes’ activity.

- attributes

  type: list member: AttributeName

  A list of the attributes to select from the message.

- next

  type: string; (length- max:128 min:1)

  The next activity in the pipeline.

- filter

  type: FilterActivity

Filters a message based on its attributes.

- name

  type: string; (length- max:128 min:1)

  The name of the ‘filter’ activity.
- filter

  type: string; (length- max:256 min:1)

  An expression that looks like a SQL WHERE clause that must return a Boolean value.
- next

  type: string; (length- max:128 min:1)

  The next activity in the pipeline.
- math

  type: MathActivity

  Computes an arithmetic expression using the message’s attributes and adds it to the message.
- name

  type: string; (length- max:128 min:1)

  The name of the ‘math’ activity.
- attribute

  type: string; (length- max:256 min:1)

  The name of the attribute that contains the result of the math operation.
- math

  type: string; (length- max:256 min:1)

  An expression that uses one or more existing attributes and must return an integer value.
- next

  type: string; (length- max:128 min:1)

  The next activity in the pipeline.
- deviceRegistryEnrich

  type: DeviceRegistryEnrichActivity

  Adds data from the AWS IoT device registry to your message.
- name

  type: string; (length- max:128 min:1)

  The name of the ‘deviceRegistryEnrich’ activity.
- attribute

  type: string; (length- max:256 min:1)

  The name of the attribute that is added to the message.
- thingName

type: string; (length- max:256 min:1)

The name of the IoT device whose registry information is added to the message.

- roleArn

  type: string; (length- max:2048 min:20)

  The ARN of the role that allows access to the device's registry information.

- next

  type: string; (length- max:128 min:1)

  The next activity in the pipeline.

- deviceShadowEnrich

  type: DeviceShadowEnrichActivity

  Adds information from the AWS IoT Device Shadows service to a message.

- name

  type: string; (length- max:128 min:1)

  The name of the 'deviceShadowEnrich' activity.

- attribute

  type: string; (length- max:256 min:1)

  The name of the attribute that is added to the message.

- thingName

  type: string; (length- max:256 min:1)

  The name of the IoT device whose shadow information is added to the message.

- roleArn

  type: string; (length- max:2048 min:20)

  The ARN of the role that allows access to the device's shadow.

- next

  type: string; (length- max:128 min:1)

  The next activity in the pipeline.

- payloads

  type: list member: MessagePayload

  The sample message payloads on which the pipeline activity is run.

Output:

```
{
  "payloads": [
    "blob"
  ],
  "logResult": "string"
```

```
}
```

fields:

- payloads

  type: list member: MessagePayload

  The enriched or transformed sample message payloads as base64-encoded strings. (The results of running the pipeline activity on each input sample message payload, encoded in base64.)
- logResult

  type: string

  In case the pipeline activity fails, the log message that is generated.

Errors:

- InvalidRequestException

  The request was not valid.

  HTTP response code: 400
- InternalFailureException

  There was an internal failure.

  HTTP response code: 500
- ServiceUnavailableException

  The service is temporarily unavailable.

  HTTP response code: 503
- ThrottlingException

  The request was denied due to request throttling.

  HTTP response code: 429

SampleChannelData

Retrieves a sample of messages from the specified channel ingested during the specified timeframe. Up to 10 messages can be retrieved.

CLI Synopsis:

```
aws iotanalytics  sample-channel-data
 --channel-name <value>
[--max-messages <value>]
[--start-time <value>]
[--end-time <value>]
[--cli-input-json <value>]
[--generate-cli-skeleton]
```

cli-input-json format:

```
{
```

```
  "channelName": "string",
  "maxMessages": "integer",
  "startTime": "timestamp",
  "endTime": "timestamp"
}
```

fields:

- channelName

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the channel whose message samples are retrieved.
- maxMessages

  type: integer java class: java.lang.Integer range- max:10 min:1

  The number of sample messages to be retrieved. The limit is 10, the default is also 10.
- startTime

  type: timestamp

  The start of the time window from which sample messages are retrieved.
- endTime

  type: timestamp

  The end of the time window from which sample messages are retrieved.

Output:

```
{
  "payloads": [
    "blob"
  ]
}
```

fields:

- payloads

  type: list member: MessagePayload

  The list of message samples. Each sample message is returned as a base64-encoded string.

Errors:

- InvalidRequestException

  The request was not valid.

  HTTP response code: 400
- ResourceNotFoundException

  A resource with the specified name could not be found.

  HTTP response code: 404
- InternalFailureException

There was an internal failure.

HTTP response code: 500

- ServiceUnavailableException

  The service is temporarily unavailable.

  HTTP response code: 503

- ThrottlingException

  The request was denied due to request throttling.

  HTTP response code: 429

StartPipelineReprocessing

Starts the reprocessing of raw message data through the pipeline.

CLI Synopsis:

```
aws iotanalytics  start-pipeline-reprocessing
 --pipeline-name <value>
[--start-time <value>]
[--end-time <value>]
[--cli-input-json <value>]
[--generate-cli-skeleton]
```

cli-input-json format:

```
{
  "pipelineName": "string",
  "startTime": "timestamp",
  "endTime": "timestamp"
}
```

fields:

- pipelineName

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the pipeline on which to start reprocessing.

- startTime

  type: timestamp

  The start time (inclusive) of raw message data that is reprocessed.

- endTime

  type: timestamp

  The end time (exclusive) of raw message data that is reprocessed.

Output:

```
{
```

```
  "reprocessingId": "string"
}
```

fields:

- reprocessingId

  type: string

  The ID of the pipeline reprocessing activity that was started.

Errors:

- ResourceNotFoundException

  A resource with the specified name could not be found.

  HTTP response code: 404
- ResourceAlreadyExistsException

  A resource with the same name already exists.

  HTTP response code: 409
- InvalidRequestException

  The request was not valid.

  HTTP response code: 400
- InternalFailureException

  There was an internal failure.

  HTTP response code: 500
- ServiceUnavailableException

  The service is temporarily unavailable.

  HTTP response code: 503
- ThrottlingException

  The request was denied due to request throttling.

  HTTP response code: 429

TagResource

Adds to or modifies the tags of the given resource. Tags are metadata which can be used to manage a resource.

CLI Synopsis:

```
aws iotanalytics  tag-resource
 --resource-arn <value>
 --tags <value>
[--cli-input-json <value>]
[--generate-cli-skeleton]
```

cli-input-json format:

```
{
  "resourceArn": "string",
  "tags": [
    {
      "key": "string",
      "value": "string"
    }
  ]
}
```

fields:

- resourceArn

  type: string; (length- max:2048 min:20)

  The ARN of the resource whose tags you want to modify.
- tags

  type: list member: Tag

  The new or modified tags for the resource.
- key

  type: string; (length- max:128 min:1)

  The tag's key.
- value

  type: string; (length- max:128 min:1)

  The tag's value.

Output:

None

Errors:

- InvalidRequestException

  The request was not valid.

  HTTP response code: 400
- InternalFailureException

  There was an internal failure.

  HTTP response code: 500
- ServiceUnavailableException

  The service is temporarily unavailable.

  HTTP response code: 503
- ThrottlingException

  The request was denied due to request throttling.

HTTP response code: 429

- LimitExceededException

  The command caused an internal limit to be exceeded.

  HTTP response code: 410

- ResourceNotFoundException

  A resource with the specified name could not be found.

  HTTP response code: 404

UntagResource

Removes the given tags (metadata) from the resource.

CLI Synopsis:

```
aws iotanalytics  untag-resource
 --resource-arn <value>
 --tag-keys <value>
[--cli-input-json <value>]
[--generate-cli-skeleton]
```

cli-input-json format:

```
{
  "resourceArn": "string",
  "tagKeys": [
    "string"
  ]
}
```

fields:

- resourceArn

  type: string; (length- max:2048 min:20)

  The ARN of the resource whose tags you want to remove.

- tagKeys

  type: list member: TagKey

  The keys of those tags which you want to remove.

Output:

None

Errors:

- InvalidRequestException

  The request was not valid.

  HTTP response code: 400

- InternalFailureException

  There was an internal failure.

  HTTP response code: 500
- ServiceUnavailableException

  The service is temporarily unavailable.

  HTTP response code: 503
- ThrottlingException

  The request was denied due to request throttling.

  HTTP response code: 429
- LimitExceededException

  The command caused an internal limit to be exceeded.

  HTTP response code: 410
- ResourceNotFoundException

  A resource with the specified name could not be found.

  HTTP response code: 404

UpdateChannel

Updates the settings of a channel.

CLI Synopsis:

```
aws iotanalytics  update-channel
 --channel-name <value>
[--channel-storage <value>]
[--retention-period <value>]
[--cli-input-json <value>]
[--generate-cli-skeleton]
```

cli-input-json format:

```
{
  "channelName": "string",
  "channelStorage": {
    "serviceManagedS3": {
    },
    "customerManagedS3": {
      "bucket": "string",
      "keyPrefix": "string",
      "roleArn": "string"
    }
  },
  "retentionPeriod": {
    "unlimited": "boolean",
    "numberOfDays": "integer"
  }
}
```

fields:

- channelName

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the channel to be updated.
- channelStorage

  type: ChannelStorage

  Where channel data is stored. You may choose one of “serviceManagedS3” or “customerManagedS3” storage. If not specified, the default is “serviceManagedS3”. This cannot be changed after creation of the channel.
- serviceManagedS3

  type: ServiceManagedChannelS3Storage

  Use this to store channel data in an S3 bucket managed by the AWS IoT Analytics service. The choice of service-managed or customer-managed S3 storage cannot be changed after creation of the channel.
- customerManagedS3

  type: CustomerManagedChannelS3Storage

  Use this to store channel data in an S3 bucket that you manage. If customer managed storage is selected, the “retentionPeriod” parameter is ignored. The choice of service-managed or customer-managed S3 storage cannot be changed after creation of the channel.
- bucket

  type: string; (length- max:255 min:3); (pattern: ^[a-zA-Z0-9.-_]*$)

  The name of the Amazon S3 bucket in which channel data is stored.
- keyPrefix

  type: string; (length- max:255 min:1); (pattern: ^[a-zA-Z0-9!_.*'()/{}:-]*/$)

  [Optional] The prefix used to create the keys of the channel data objects. Each object in an Amazon S3 bucket has a key that is its unique identifier within the bucket (each object in a bucket has exactly one key). The prefix must end with a ‘/’.
- roleArn

  type: string; (length- max:2048 min:20)

  The ARN of the role which grants AWS IoT Analytics permission to interact with your Amazon S3 resources.
- retentionPeriod

  type: RetentionPeriod

  How long, in days, message data is kept for the channel. The retention period cannot be updated if the channel’s S3 storage is customer-managed.
- unlimited

  type: boolean

  If true, message data is kept indefinitely.
- numberOfDays

type: integer java class: java.lang.Integer range- min:1

The number of days that message data is kept. The “unlimited” parameter must be false.

Output:

None

Errors:

- InvalidRequestException

  The request was not valid.

  HTTP response code: 400
- ResourceNotFoundException

  A resource with the specified name could not be found.

  HTTP response code: 404
- InternalFailureException

  There was an internal failure.

  HTTP response code: 500
- ServiceUnavailableException

  The service is temporarily unavailable.

  HTTP response code: 503
- ThrottlingException

  The request was denied due to request throttling.

  HTTP response code: 429

UpdateDataset

Updates the settings of a data set.

CLI Synopsis:

```
aws iotanalytics  update-dataset
 --dataset-name <value>
 --actions <value>
[--triggers <value>]
[--content-delivery-rules <value>]
[--retention-period <value>]
[--versioning-configuration <value>]
[--cli-input-json <value>]
[--generate-cli-skeleton]
```

cli-input-json format:

```
{
  "datasetName": "string",
  "actions": [
```

```
    {
      "actionName": "string",
      "queryAction": {
        "sqlQuery": "string",
        "filters": [
          {
            "deltaTime": {
              "offsetSeconds": "integer",
              "timeExpression": "string"
            }
          }
        ]
      },
      "containerAction": {
        "image": "string",
        "executionRoleArn": "string",
        "resourceConfiguration": {
          "computeType": "string",
          "volumeSizeInGB": "integer"
        },
        "variables": [
          {
            "name": "string",
            "stringValue": "string",
            "doubleValue": "double",
            "datasetContentVersionValue": {
              "datasetName": "string"
            },
            "outputFileUriValue": {
              "fileName": "string"
            }
          }
        ]
      }
    }
  ],
  "triggers": [
    {
      "schedule": {
        "expression": "string"
      },
      "dataset": {
        "name": "string"
      }
    }
  ],
  "contentDeliveryRules": [
    {
      "entryName": "string",
      "destination": {
        "iotEventsDestinationConfiguration": {
          "inputName": "string",
          "roleArn": "string"
        },
        "s3DestinationConfiguration": {
          "bucket": "string",
          "key": "string",
          "glueConfiguration": {
            "tableName": "string",
            "databaseName": "string"
          },
          "roleArn": "string"
        }
      }
    }
  ],
```

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

fields:

- datasetName

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the data set to update.
- actions

  type: list member: DatasetAction

  A list of “DatasetAction” objects.
- actionName

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the data set action by which data set contents are automatically created.
- queryAction

  type: SqlQueryDatasetAction

  An “SqlQueryDatasetAction” object that uses an SQL query to automatically create data set contents.
- sqlQuery

  type: string

  A SQL query string.
- filters

  type: list member: QueryFilter

  Pre-filters applied to message data.
- deltaTime

  type: DeltaTime

  Used to limit data to that which has arrived since the last execution of the action.
- offsetSeconds

  type: integer java class: java.lang.Integer

  The number of seconds of estimated “in flight” lag time of message data. When you create data set contents using message data from a specified time frame, some message data may still be “in flight” when processing begins, and so will not arrive in time to be processed. Use this field to make allowances for the “in flight” time of your message data, so that data not processed from a previous time frame will be included with the next time frame. Without this, missed message data would be excluded from processing during the next time frame as well, because its timestamp places it within the previous time frame.
- timeExpression

type: string

An expression by which the time of the message data may be determined. This may be the name of a timestamp field, or a SQL expression which is used to derive the time the message data was generated.

- containerAction

  type: ContainerDatasetAction

  Information which allows the system to run a containerized application in order to create the data set contents. The application must be in a Docker container along with any needed support libraries.
- image

  type: string; (length- max:255)

  The ARN of the Docker container stored in your account. The Docker container contains an application and needed support libraries and is used to generate data set contents.
- executionRoleArn

  type: string; (length- max:2048 min:20)

  The ARN of the role which gives permission to the system to access needed resources in order to run the “containerAction”. This includes, at minimum, permission to retrieve the data set contents which are the input to the containerized application.
- resourceConfiguration

  type: ResourceConfiguration

  Configuration of the resource which executes the “containerAction”.
- computeType

  type: string

  The type of the compute resource used to execute the “containerAction”. Possible values are: ACU_1 (vCPU=4, memory=16GiB) or ACU_2 (vCPU=8, memory=32GiB). enum: ACU_1 | ACU_2
- volumeSizeInGB

  type: integer range- max:50 min:1

  The size (in GB) of the persistent storage available to the resource instance used to execute the “containerAction” (min: 1, max: 50).
- variables

  type: list member: Variable

  The values of variables used within the context of the execution of the containerized application (basically, parameters passed to the application). Each variable must have a name and a value given by one of “stringValue”, “datasetContentVersionValue”, or “outputFileUriValue”.
- name

  type: string; (length- max:256 min:1)

  The name of the variable.
- stringValue

  type: string; (length- max:1024 min:0)

  The value of the variable as a string.

- datasetContentVersionValue

  type: DatasetContentVersionValue

  The value of the variable as a structure that specifies a data set content version.
- datasetName

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the data set whose latest contents are used as input to the notebook or application.
- outputFileUriValue

  type: OutputFileUriValue

  The value of the variable as a structure that specifies an output file URI.
- fileName

  type: string; (pattern: [w.-]{1,255})

  The URI of the location where data set contents are stored, usually the URI of a file in an S3 bucket.
- triggers

  type: list member: DatasetTrigger

  A list of “DatasetTrigger” objects. The list can be empty or can contain up to five DataSetTrigger objects.
- schedule

  type: Schedule

  The “Schedule” when the trigger is initiated.
- expression

  type: string

  The expression that defines when to trigger an update. For more information, see Schedule Expressions for Rules in the Amazon CloudWatch Events User Guide.
- dataset

  type: TriggeringDataset

  The data set whose content creation triggers the creation of this data set’s contents.
- name

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the data set whose content generation triggers the new data set content generation.
- contentDeliveryRules

  type: list member: DatasetContentDeliveryRule

  When data set contents are created they are delivered to destinations specified here.
- entryName

  type: string

  The name of the data set content delivery rules entry.
- destination

type: DatasetContentDeliveryDestination

The destination to which data set contents are delivered.

- iotEventsDestinationConfiguration

  type: IotEventsDestinationConfiguration

  Configuration information for delivery of data set contents to AWS IoT Events.

- inputName

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z][a-zA-Z0-9_]*$)

  The name of the AWS IoT Events input to which data set contents are delivered.

- roleArn

  type: string; (length- max:2048 min:20)

  The ARN of the role which grants AWS IoT Analytics permission to deliver data set contents to an AWS IoT Events input.

- s3DestinationConfiguration

  type: S3DestinationConfiguration

  Configuration information for delivery of data set contents to Amazon S3.

- bucket

  type: string; (length- max:255 min:3); (pattern: ^[a-zA-Z0-9.-_]*$)

  The name of the Amazon S3 bucket to which data set contents are delivered.

- key

  type: string; (length- max:255 min:1); (pattern: ^[a-zA-Z0-9!_.*'()/{}:-]*$)

  The key of the data set contents object. Each object in an Amazon S3 bucket has a key that is its unique identifier within the bucket (each object in a bucket has exactly one key). To produce a unique key, you can use "!{iotanalytics:scheduleTime}" to insert the time of the scheduled SQL query run, or "!{iotanalytics:versioned} to insert a unique hash identifying the data set, for example: "/DataSet/!{iotanalytics:scheduleTime}/!{iotanalytics:versioned}.csv".

- glueConfiguration

  type: GlueConfiguration

  Configuration information for coordination with the AWS Glue ETL (extract, transform and load) service.

- tableName

  type: string; (length- max:150 min:1); (pattern: [u0020-uD7FFuE000-uFFFDuD800uDC00-uDBFFuDFFFt]*)

  The name of the table in your AWS Glue Data Catalog which is used to perform the ETL (extract, transform and load) operations. (An AWS Glue Data Catalog table contains partitioned data and descriptions of data sources and targets.)

- databaseName

  type: string; (length- max:150 min:1); (pattern: [u0020-uD7FFuE000-uFFFDuD800uDC00-uDBFFuDFFFt]*)

The name of the database in your AWS Glue Data Catalog in which the table is located. (An AWS Glue Data Catalog database contains Glue Data tables.)

- roleArn

  type: string; (length- max:2048 min:20)

  The ARN of the role which grants AWS IoT Analytics permission to interact with your Amazon S3 and AWS Glue resources.
- retentionPeriod

  type: RetentionPeriod

  How long, in days, data set contents are kept for the data set.
- unlimited

  type: boolean

  If true, message data is kept indefinitely.
- numberOfDays

  type: integer java class: java.lang.Integer range- min:1

  The number of days that message data is kept. The “unlimited” parameter must be false.
- versioningConfiguration

  type: VersioningConfiguration

  [Optional] How many versions of data set contents are kept. If not specified or set to null, only the latest version plus the latest succeeded version (if they are different) are kept for the time period specified by the “retentionPeriod” parameter. (For more information, see https://docs.aws.amazon.com/iotanalytics/latest/userguide/getting-started.html#aws-iot-analytics-dataset-versions)
- unlimited

  type: boolean

  If true, unlimited versions of data set contents will be kept.
- maxVersions

  type: integer java class: java.lang.Integer range- max:1000 min:1

  How many versions of data set contents will be kept. The “unlimited” parameter must be false.

Output:

None

Errors:

- InvalidRequestException

  The request was not valid.

  HTTP response code: 400
- ResourceNotFoundException

  A resource with the specified name could not be found.

HTTP response code: 404

- InternalFailureException

  There was an internal failure.

  HTTP response code: 500

- ServiceUnavailableException

  The service is temporarily unavailable.

  HTTP response code: 503

- ThrottlingException

  The request was denied due to request throttling.

  HTTP response code: 429

UpdateDatastore

Updates the settings of a data store.

CLI Synopsis:

```
aws iotanalytics  update-datastore
 --datastore-name <value>
[--retention-period <value>]
[--datastore-storage <value>]
[--cli-input-json <value>]
[--generate-cli-skeleton]
```

cli-input-json format:

```
{
  "datastoreName": "string",
  "retentionPeriod": {
    "unlimited": "boolean",
    "numberOfDays": "integer"
  },
  "datastoreStorage": {
    "serviceManagedS3": {
    },
    "customerManagedS3": {
      "bucket": "string",
      "keyPrefix": "string",
      "roleArn": "string"
    }
  }
}
```

fields:

- datastoreName

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the data store to be updated.

- retentionPeriod

  type: RetentionPeriod

  How long, in days, message data is kept for the data store. The retention period cannot be updated if the data store's S3 storage is customer-managed.
- unlimited

  type: boolean

  If true, message data is kept indefinitely.
- numberOfDays

  type: integer java class: java.lang.Integer range- min:1

  The number of days that message data is kept. The "unlimited" parameter must be false.
- datastoreStorage

  type: DatastoreStorage

  Where data store data is stored. You may choose one of "serviceManagedS3" or "customerManagedS3" storage. If not specified, the default is "serviceManagedS3". This cannot be changed after the data store is created.
- serviceManagedS3

  type: ServiceManagedDatastoreS3Storage

  Use this to store data store data in an S3 bucket managed by the AWS IoT Analytics service. The choice of service-managed or customer-managed S3 storage cannot be changed after creation of the data store.
- customerManagedS3

  type: CustomerManagedDatastoreS3Storage

  Use this to store data store data in an S3 bucket that you manage. When customer managed storage is selected, the "retentionPeriod" parameter is ignored. The choice of service-managed or customer-managed S3 storage cannot be changed after creation of the data store.
- bucket

  type: string; (length- max:255 min:3); (pattern: ^[a-zA-Z0-9.-_]*$)

  The name of the Amazon S3 bucket in which data store data is stored.
- keyPrefix

  type: string; (length- max:255 min:1); (pattern: ^[a-zA-Z0-9!_.*'()/{}:-]*/$)

  [Optional] The prefix used to create the keys of the data store data objects. Each object in an Amazon S3 bucket has a key that is its unique identifier within the bucket (each object in a bucket has exactly one key). The prefix must end with a '/'.
- roleArn

  type: string; (length- max:2048 min:20)

  The ARN of the role which grants AWS IoT Analytics permission to interact with your Amazon S3 resources.

Output:

None

Errors:

- InvalidRequestException

  The request was not valid.

  HTTP response code: 400
- ResourceNotFoundException

  A resource with the specified name could not be found.

  HTTP response code: 404
- InternalFailureException

  There was an internal failure.

  HTTP response code: 500
- ServiceUnavailableException

  The service is temporarily unavailable.

  HTTP response code: 503
- ThrottlingException

  The request was denied due to request throttling.

  HTTP response code: 429

UpdatePipeline

Updates the settings of a pipeline. You must specify both a channel and a datastore activity and, optionally, as many as 23 additional activities in the pipelineActivities array.

CLI Synopsis:

```
aws iotanalytics  update-pipeline
 --pipeline-name <value>
 --pipeline-activities <value>
[--cli-input-json <value>]
[--generate-cli-skeleton]
```

cli-input-json format:

```
{
  "pipelineName": "string",
  "pipelineActivities": [
    {
      "channel": {
        "name": "string",
        "channelName": "string",
        "next": "string"
      },
      "lambda": {
        "name": "string",
```

```
        "lambdaName": "string",
        "batchSize": "integer",
        "next": "string"
      },
      "datastore": {
        "name": "string",
        "datastoreName": "string"
      },
      "addAttributes": {
        "name": "string",
        "attributes": {
          "string": "string"
        },
        "next": "string"
      },
      "removeAttributes": {
        "name": "string",
        "attributes": [
          "string"
        ],
        "next": "string"
      },
      "selectAttributes": {
        "name": "string",
        "attributes": [
          "string"
        ],
        "next": "string"
      },
      "filter": {
        "name": "string",
        "filter": "string",
        "next": "string"
      },
      "math": {
        "name": "string",
        "attribute": "string",
        "math": "string",
        "next": "string"
      },
      "deviceRegistryEnrich": {
        "name": "string",
        "attribute": "string",
        "thingName": "string",
        "roleArn": "string",
        "next": "string"
      },
      "deviceShadowEnrich": {
        "name": "string",
        "attribute": "string",
        "thingName": "string",
        "roleArn": "string",
        "next": "string"
      }
    }
  ]
}
```

fields:

- pipelineName

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the pipeline to update.

- pipelineActivities

  type: list member: PipelineActivity

  A list of “PipelineActivity” objects. Activities perform transformations on your messages, such as removing, renaming or adding message attributes; filtering messages based on attribute values; invoking your Lambda functions on messages for advanced processing; or performing mathematical transformations to normalize device data. The list can be 2-25 PipelineActivity objects and must contain both a channel and a datastore activity. Each entry in the list must contain only one activity, for example: pipelineActivities = [{ “channel”: { … } }, { “lambda”: { … } }, …]
- channel

  type: ChannelActivity

  Determines the source of the messages to be processed.
- name

  type: string; (length- max:128 min:1)

  The name of the ‘channel’ activity.
- channelName

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the channel from which the messages are processed.
- next

  type: string; (length- max:128 min:1)

  The next activity in the pipeline.
- lambda

  type: LambdaActivity

  Runs a Lambda function to modify the message.
- name

  type: string; (length- max:128 min:1)

  The name of the ‘lambda’ activity.
- lambdaName

  type: string; (length- max:64 min:1); (pattern: ^[a-zA-Z0-9_-]+$)

  The name of the Lambda function that is run on the message.
- batchSize

  type: integer java class: java.lang.Integer range- max:1000 min:1

  The number of messages passed to the Lambda function for processing. The AWS Lambda function must be able to process all of these messages within five minutes, which is the maximum timeout duration for Lambda functions.
- next

  type: string; (length- max:128 min:1)

  The next activity in the pipeline.
- datastore

type: DatastoreActivity

Specifies where to store the processed message data.

- name

  type: string; (length- max:128 min:1)

  The name of the 'datastore' activity.

- datastoreName

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the data store where processed messages are stored.

- addAttributes

  type: AddAttributesActivity

  Adds other attributes based on existing attributes in the message.

- name

  type: string; (length- max:128 min:1)

  The name of the 'addAttributes' activity.

- attributes

  type: map key: AttributeName value: AttributeName

  A list of 1-50 "AttributeNameMapping" objects that map an existing attribute to a new attribute. The existing attributes remain in the message, so if you want to remove the originals, use "RemoveAttributeActivity".

- next

  type: string; (length- max:128 min:1)

  The next activity in the pipeline.

- removeAttributes

  type: RemoveAttributesActivity

  Removes attributes from a message.

- name

  type: string; (length- max:128 min:1)

  The name of the 'removeAttributes' activity.

- attributes

  type: list member: AttributeName

  A list of 1-50 attributes to remove from the message.

- next

  type: string; (length- max:128 min:1)

  The next activity in the pipeline.

- selectAttributes

type: SelectAttributesActivity

Creates a new message using only the specified attributes from the original message.

- name

  type: string; (length- max:128 min:1)

  The name of the 'selectAttributes' activity.
- attributes

  type: list member: AttributeName

  A list of the attributes to select from the message.
- next

  type: string; (length- max:128 min:1)

  The next activity in the pipeline.
- filter

  type: FilterActivity

  Filters a message based on its attributes.
- name

  type: string; (length- max:128 min:1)

  The name of the 'filter' activity.
- filter

  type: string; (length- max:256 min:1)

  An expression that looks like a SQL WHERE clause that must return a Boolean value.
- next

  type: string; (length- max:128 min:1)

  The next activity in the pipeline.
- math

  type: MathActivity

  Computes an arithmetic expression using the message's attributes and adds it to the message.
- name

  type: string; (length- max:128 min:1)

  The name of the 'math' activity.
- attribute

  type: string; (length- max:256 min:1)

  The name of the attribute that contains the result of the math operation.
- math

  type: string; (length- max:256 min:1)

An expression that uses one or more existing attributes and must return an integer value.

- next

  type: string; (length- max:128 min:1)

  The next activity in the pipeline.
- deviceRegistryEnrich

  type: DeviceRegistryEnrichActivity

  Adds data from the AWS IoT device registry to your message.
- name

  type: string; (length- max:128 min:1)

  The name of the 'deviceRegistryEnrich' activity.
- attribute

  type: string; (length- max:256 min:1)

  The name of the attribute that is added to the message.
- thingName

  type: string; (length- max:256 min:1)

  The name of the IoT device whose registry information is added to the message.
- roleArn

  type: string; (length- max:2048 min:20)

  The ARN of the role that allows access to the device's registry information.
- next

  type: string; (length- max:128 min:1)

  The next activity in the pipeline.
- deviceShadowEnrich

  type: DeviceShadowEnrichActivity

  Adds information from the AWS IoT Device Shadows service to a message.
- name

  type: string; (length- max:128 min:1)

  The name of the 'deviceShadowEnrich' activity.
- attribute

  type: string; (length- max:256 min:1)

  The name of the attribute that is added to the message.
- thingName

  type: string; (length- max:256 min:1)

  The name of the IoT device whose shadow information is added to the message.
- roleArn

type: string; (length- max:2048 min:20)

The ARN of the role that allows access to the device's shadow.

- next

  type: string; (length- max:128 min:1)

  The next activity in the pipeline.

Output:

None

Errors:

- InvalidRequestException

  The request was not valid.

  HTTP response code: 400
- ResourceNotFoundException

  A resource with the specified name could not be found.

  HTTP response code: 404
- InternalFailureException

  There was an internal failure.

  HTTP response code: 500
- ServiceUnavailableException

  The service is temporarily unavailable.

  HTTP response code: 503
- ThrottlingException

  The request was denied due to request throttling.

  HTTP response code: 429
- LimitExceededException

  The command caused an internal limit to be exceeded.

  HTTP response code: 410