User Guide

Amazon IoT Analytics

Amazon IoT Analytics: User Guide

Table of Contents

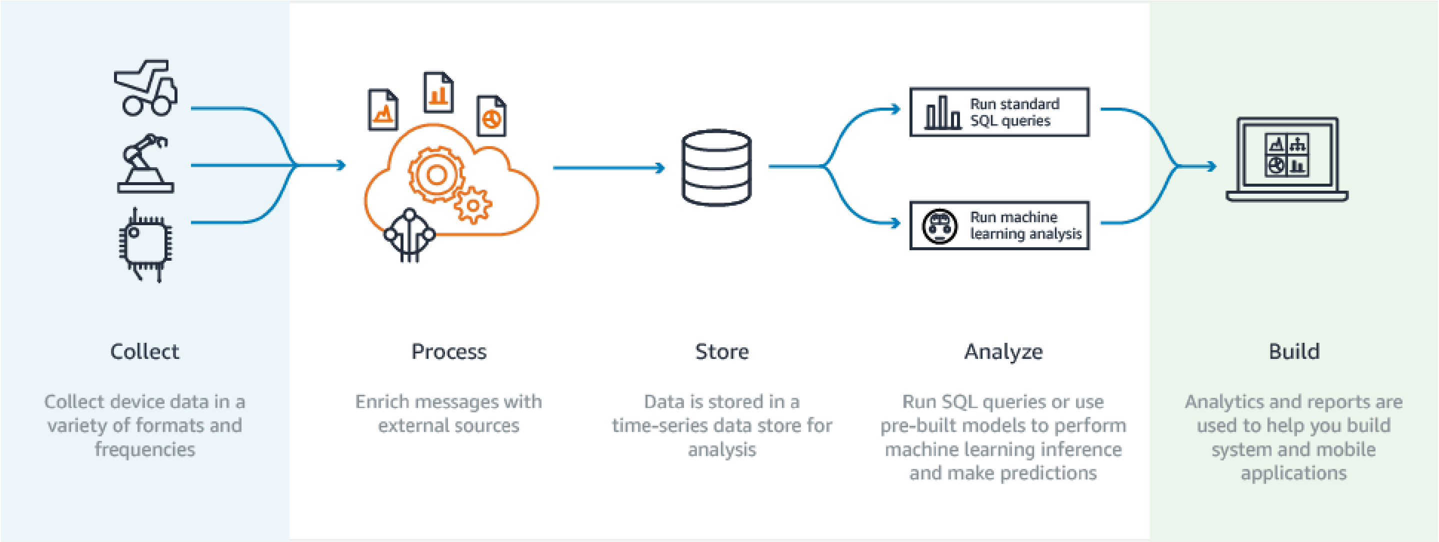
What is Amazon IoT Analytics?

Amazon IoT Analytics automates the steps required to analyze data from IoT devices. Amazon IoT Analytics 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. Amazon IoT Analytics enables advanced data exploration through integration with Jupyter Notebook. Amazon IoT Analytics also enables data visualization through integration with QuickSight. QuickSight is available in the following Regions.

Traditional analytics and business intelligence tools are designed to process structured data. Raw IoT data often comes from devices that record less structured data (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. Amazon IoT Analytics lets 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. Amazon IoT Analytics includes pre-built models for common IoT use cases so that 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 Amazon IoT Analytics

The following graphic shows an overview of how you can use Amazon IoT Analytics.


Run standard SQL queries
Run machine learning analysis
Collect
Collect device data in a variety of formats and frequencies
Process
Enrich messages with external sources
Store
Data is stored in a time-series data store for analysis
Analyze
Run SQL queries or use pre-built models to perform machine learning inference and make predictions
Build
Analytics and reports are used to help you build system and mobile applications

Key features

Collect

- Integrated with Amazon IoT Core—Amazon IoT Analytics is fully integrated with Amazon IoT Core so it can receive messages from connected devices as they stream in.
- Use a batch API to add data from any source—Amazon 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 Amazon IoT Analytics. For more information, see BatchPutMessage in the *Amazon IoT Analytics API Reference*.
- Collect only the data you want to store and analyze—You can use the Amazon IoT Analytics console to configure Amazon IoT Analytics to receive messages from devices through MQTT topic filters in various formats and frequencies. Amazon 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—Amazon IoT Analytics lets you define Amazon Lambda functions that are triggered when Amazon IoT Analytics detects missing data, so you can run code to estimate and fill gaps. You can also define maximum and minimum filters and percentile thresholds to remove outliers in your data.
- Transform—Amazon IoT Analytics can transform messages using mathematical or conditional logic you define, so that you can perform common calculations like Celsius into Fahrenheit conversion.
- Enrich—Amazon IoT Analytics can enrich data with external data sources such as a weather forecast, and then route the data to the Amazon IoT Analytics data store.

Store

- Time-series data store—Amazon 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—Amazon 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—Amazon IoT Analytics provides a SQL query engine so you can run ad-hoc queries and get results quickly. 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.

- Time-series analysis—Amazon 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—Amazon IoT Analytics includes support for hosted notebooks in Jupyter Notebook for statistical analysis and machine learning. The service includes a set of notebook templates that contain Amazon-authored machine learning models and visualizations. You can use the templates to 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 and execute it on a schedule that you specify. For more information, see Automating your workflow.
- 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 contained and executed on a schedule.

Build and visualize

- QuickSight integration—Amazon IoT Analytics provides a connector to QuickSight so that 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 Notebook in the Amazon IoT Analytics' console.

Amazon IoT Analytics components and concepts

Channel

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 using the BatchPutMessage API. The unprocessed messages are stored in an Amazon Simple Storage Service (Amazon S3) bucket that you or Amazon IoT Analytics manage.

Pipeline

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), 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

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 that you or Amazon IoT Analytics manage.

Data set

You retrieve data from a data store by creating a data set. Amazon 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 using QuickSight. You can also perform more advanced analytical functions through integration with Jupyter Notebook. Jupyter Notebook provides powerful data science tools that can perform machine learning and a range of statistical analyses. For more information, see Notebook templates.

You can send data set contents to an Amazon 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 Amazon 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

A SQL data set is similar to a materialized view from a SQL database. You can 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

A container data set enables you to automatically run your analysis tools and generate results. For more information, see Automating your workflow. 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

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 using CreateDatasetContent API.

Docker container

You can create your own Docker container to package your analysis tools or use options that SageMaker AI provides. For more information, see Docker container. You can create your own Docker container to package your analysis tools or use options provided by SageMaker AI. You can store a container in an Amazon ECR registry that you specify so it is available to install on your desired platform. 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. For more information, see Containerizing a notebook.

Delta windows

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. For more information, see the `CreateDataset` API. Usually, you'll want to create the data set content automatically by also setting up a time interval trigger (`triggers:schedule:expression`). This lets you 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. For more information, see Example 6 -- creating a SQL dataset with a Delta window (CLI).

Access Amazon IoT Analytics

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

Amazon Command Line Interface (Amazon CLI)

Run commands for Amazon 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 Amazon Command Line Interface User Guide. For more information about the commands for Amazon IoT, see iot in the *Amazon Command Line Interface Reference*.

> ⚠ **Important**
>
> Use the `aws iotanalytics` command to interact with Amazon IoT Analytics. Use the `aws iot` command to interact with other parts of the IoT system.

Amazon 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, see Actions in the *Amazon IoT API Reference*.

Amazon SDKs

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

Amazon IoT Device SDKs

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

Amazon IoT Analytics Console

You can build the components to visualize the results in the Amazon IoT Analytics console.

Use cases

Predictive maintenance

Amazon IoT Analytics provides templates to build predictive maintenance models and apply them to your devices. For example, you can use Amazon 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

Amazon 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 Amazon IoT Analytics, you can build IoT 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 Amazon 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 leading guidelines to improve efficiency.

Smart agriculture

Amazon IoT Analytics can enrich IoT device data with contextual metadata using Amazon IoT registry data or public data sources so that your analysis 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.

Amazon IoT Analytics end of support

After careful consideration, we decided to end support for Amazon IoT Analytics, effective December 15, 2025. Amazon IoT Analytics will no longer accept new customers beginning July 24, 2024. As an existing customer with an account signed up for the service before July 23, 2024, you can continue to use Amazon IoT Analytics features. After December 15, 2025, you will no longer be able to use Amazon IoT Analytics.

As Amazon IoT Analytics approaches end-of-service on December 15, 2025, it's important for customers to understand their migration options. This page provides an overview of the key features of Amazon IoT Analytics and maps them to alternative Amazon services that are used to replicate the functionality. By understanding the capabilities of these alternate services, customers can plan and execute a smooth migration, ensuring their Amazon IoT data analytics workflows continue uninterrupted.

Topics

- Migration options
- Migration guide

Migration options

When considering a migration from Amazon IoT Analytics, it’s important to understand the benefits and reasons behind this shift. The table below provides alternate options and a mapping to existing Amazon IoT Analytics features.

Action	Amazon IoT Analytics	Alternate Service	Reason
Collect	Amazon IoT Analytics makes it easy to ingest data directly from Amazon IoT Core or other sources using the `BatchPutMessage` API. This integration ensures a	• Amazon Kinesis Data Streams • Amazon Data Firehose	Amazon Kinesis Data Streams offers a robust solution. Kinesis streams data in real-time, enabling immediate processing and analysis, which is crucial for applicati

Action	Amazon IoT Analytics	Alternate Service	Reason
	seamless flow of data from your devices to the analytics platform.		ons needing real-time insights and anomaly detection. Amazon Data Firehose simplifies the process of capturing and transforming streaming data before it lands in Amazon S3, automatically scaling to match your data throughput.

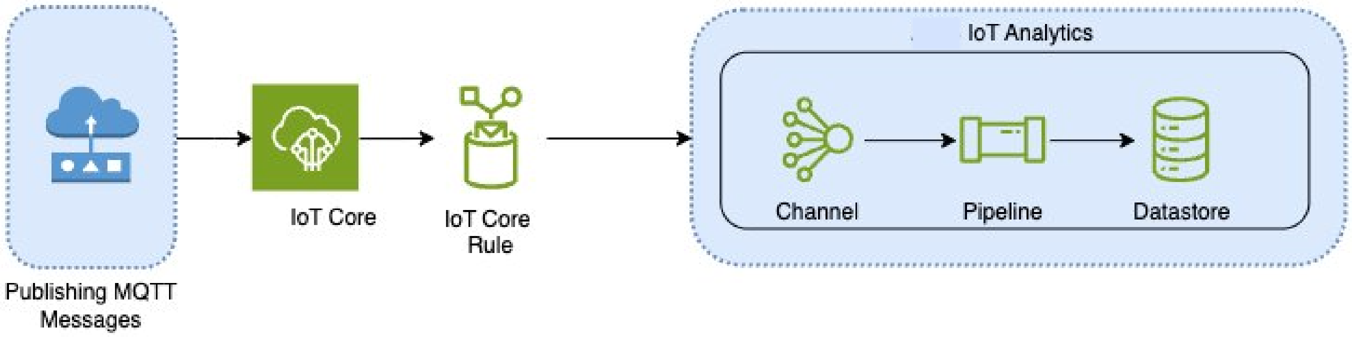
Action	Amazon IoT Analytics	Alternate Service	Reason
Process	Processing data in Amazon IoT Analytics involves cleansing, filtering, transforming, and enriching it with external sources.	- Amazon Managed Service for Apache Flink - Amazon Data Firehose	Amazon Managed Service for Apache Flink supports complex event processing, such as pattern matching and aggregations, which are essential for sophisticated Amazon IoT Analytics scenarios. Amazon Data Firehose handles simpler transformations and can invoke Amazon Lambda functions for custom processing, providing flexibility without the complexity of Flink.

Action	Amazon IoT Analytics	Alternate Service	Reason
Store	Amazon IoT Analytics uses a time-series data store optimized for Amazon IoT data, which includes features like data retention policies and access management.	• Amazon S3 • Amazon Timestream	Amazon S3 offers a scalable, durable, and cost-effective storage solution. Amazon S3's integration with other Amazon services makes it an excellent choice for long-term storage and analysis of massive datasets. Amazon Timestream is a purpose-built time series database. You can batch load data from Amazon S3.

Action	Amazon IoT Analytics	Alternate Service	Reason
Analyze	Amazon IoT Analytics provides built-in SQL query capabilities, time-series analysis, and support for hosted Jupyter Notebooks, making it easy to perform advanced analytics and machine learning.	• Amazon Glue • Amazon Athena	Amazon Glue simplifies the ETL process, making it easy to extract, transform, and load data, while also providing a data catalog that integrates with Athena to facilitate querying. Amazon Athena takes this a step further by allowing you to run SQL queries directly on data stored in Amazon S3 without needing to manage any infrastructure.
Visualize	Amazon IoT Analytics integrates with QuickSight, enabling the creation of rich visualizations and dashboards.	• Amazon QuickSight	Continue to use QuickSight depending on the alternate datastore you decide to use, like Amazon S3.

Migration guide

In the current architecture, Amazon IoT data flows from Amazon IoT Core to Amazon IoT Analytics through an Amazon IoT Core rule. Amazon IoT Analytics handles ingestion, transformation, and storage.



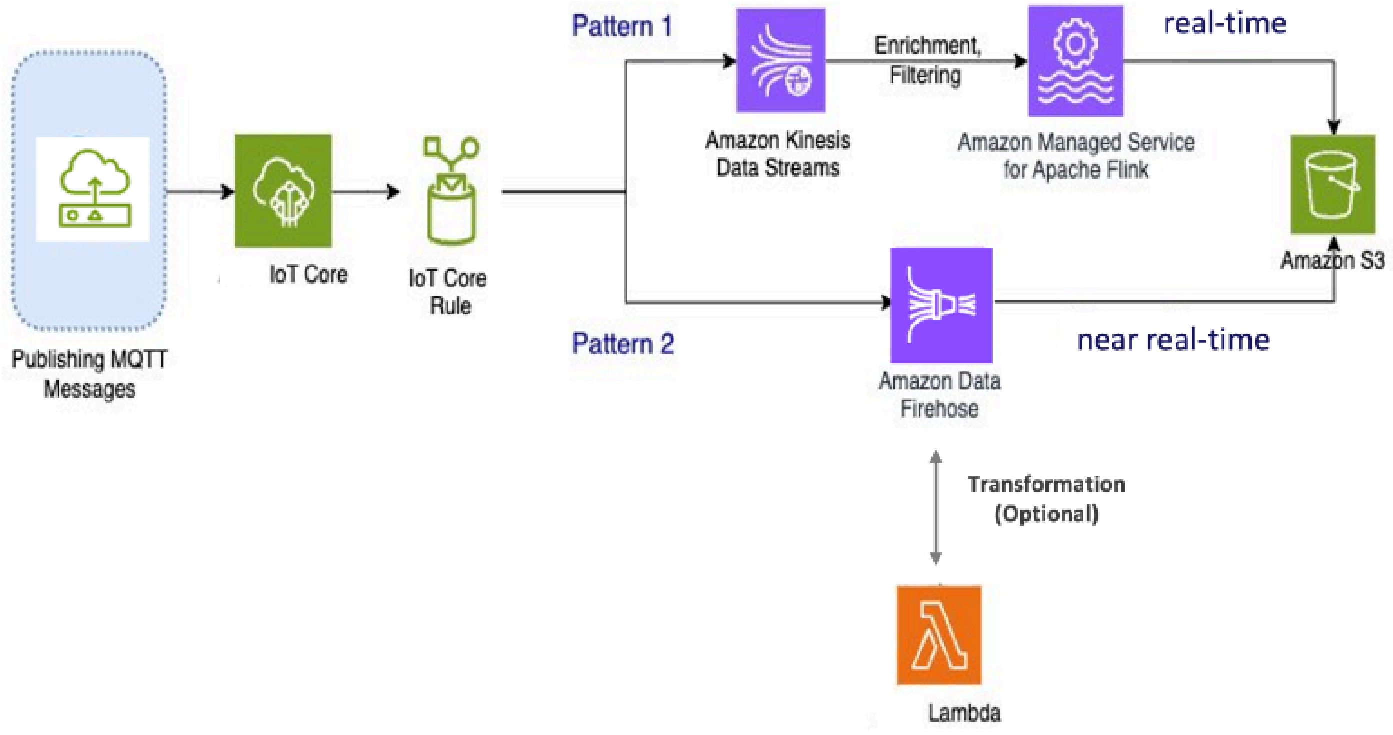
To complete the migration follow two steps:

Topics

- Step 1: Redirect ongoing data ingestion
- Step 2: Export previously ingested data
- Run on-demand queries for both patterns
- Summary

Step 1: Redirect ongoing data ingestion

The first step in your migration is to redirect your ongoing data ingestion to a new service. We recommend two patterns based on your specific use case:



Pattern 1: Amazon Kinesis Data Streams with Amazon Managed Service for Apache Flink

In this pattern, you start by publishing data to Amazon IoT Core which integrates with Amazon Kinesis Data Streams allowing you to collect, process, and analyze large bandwidth of data in real time.

Metrics and Analytics

1. **Ingest Data**: Amazon IoT data is ingested into a Amazon Kinesis Data Streams in real-time. Amazon Kinesis Data Streams can handle a high throughput of data from millions of Amazon IoT devices, enabling real-time analytics and anomaly detection.
2. **Process Data**: Use Amazon Managed Service for Apache Flink to process, enrich, and filter the data from the Amazon Kinesis Data Streams. Flink provides robust features for complex event processing, such as aggregations, joins, and temporal operations.
3. **Store Data**: Flink outputs the processed data to Amazon S3 for storage and further analysis. This data can then be queried using Amazon Athena or integrated with other Amazon analytics services.

Use this pattern if your application involves high-bandwidth streaming data and requires advanced processing, such as pattern matching or windowing, this pattern is the best fit.

Pattern 2: Use Amazon Data Firehose

In this pattern, data is published to Amazon IoT Core, which integrates with Amazon Data Firehose, allowing you to store data directly in Amazon S3. This pattern also supports basic transformations using Amazon Lambda.

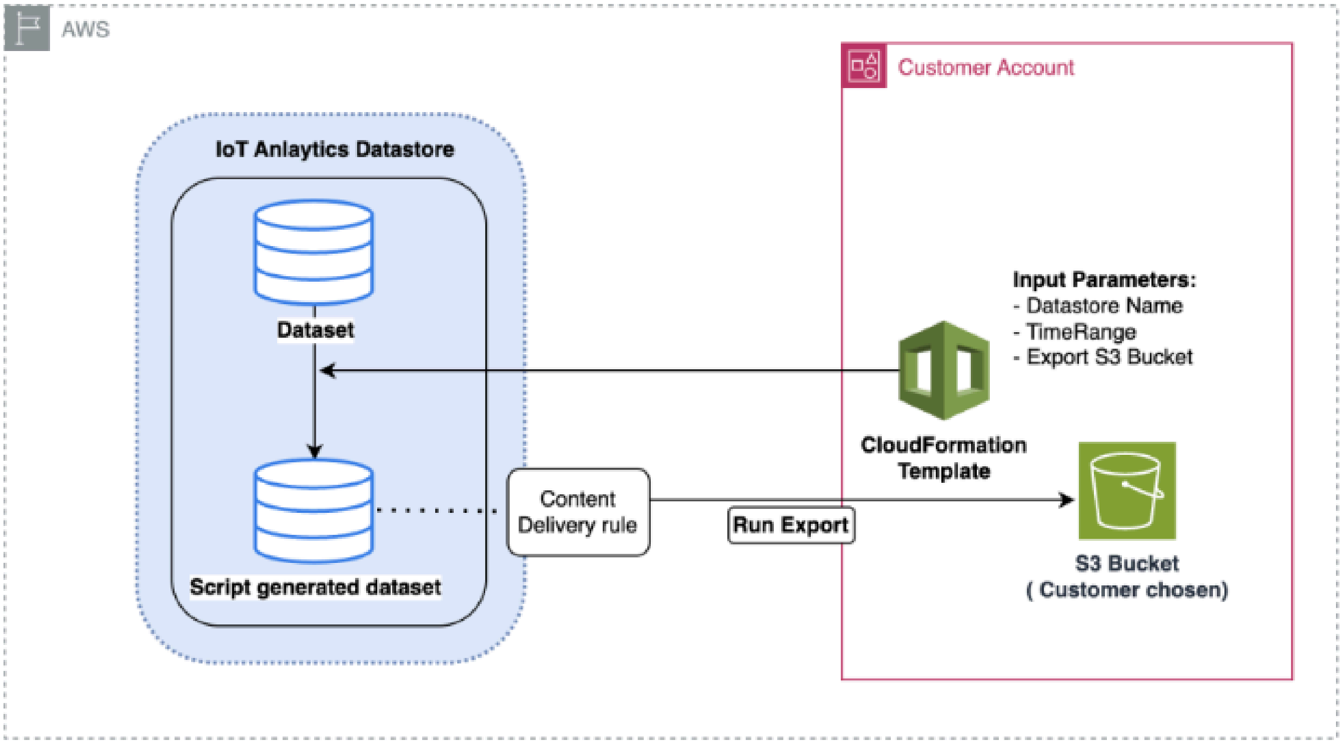
Metrics and Analytics

1. **Ingest Data**: Amazon IoT data is ingested directly from your devices or Amazon IoT Core into Amazon Data Firehose.
2. **Process Data**: Amazon Data Firehose performs basic transformations and processing on the data, such as format conversion and enrichment. You can enable Firehose data transformation by configuring it to invoke Amazon Lambda functions to transform the incoming source data before delivering it to destinations.
3. **Store Data**: The processed data is delivered to Amazon S3 in near real-time. Amazon Data Firehose automatically scales to match the throughput of incoming data, ensuring reliable and efficient data delivery.

Use this pattern for workloads that need basic transformations and processing. In addition, Amazon Data Firehose simplifies the process by offering data buffering and dynamic partitioning capabilities for data stored in Amazon S3.

Step 2: Export previously ingested data

For data previously ingested and stored in Amazon IoT Analytics, you’ll need to export it to Amazon S3. To simplify this process, you can use a Amazon CloudFormation template to automate the entire data export workflow. You can use the script for partial (time range-based) data extraction.



Amazon CloudFormation template to export data to Amazon S3

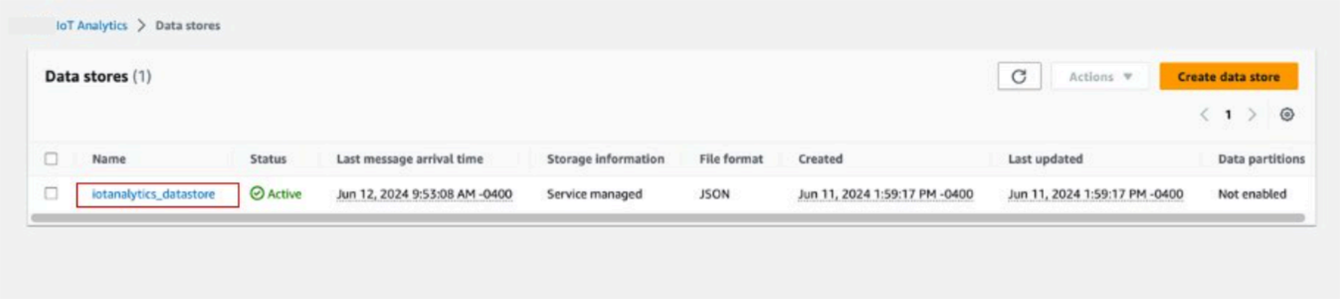
The diagram above illustrates the process of using a Amazon CloudFormation template to create a dataset within the same Amazon IoT Analytics datastore, enabling selection based on a timestamp. This allows users to retrieve specific data points within a desired timeframe. Additionally, a Content Delivery Rule is created to export the data into an Amazon S3 bucket.

The procedure below illustrates the steps.

1. Prepare the Amazon CloudFormation template and save it as a YAML file. For example, `migrate-datasource.yaml`.

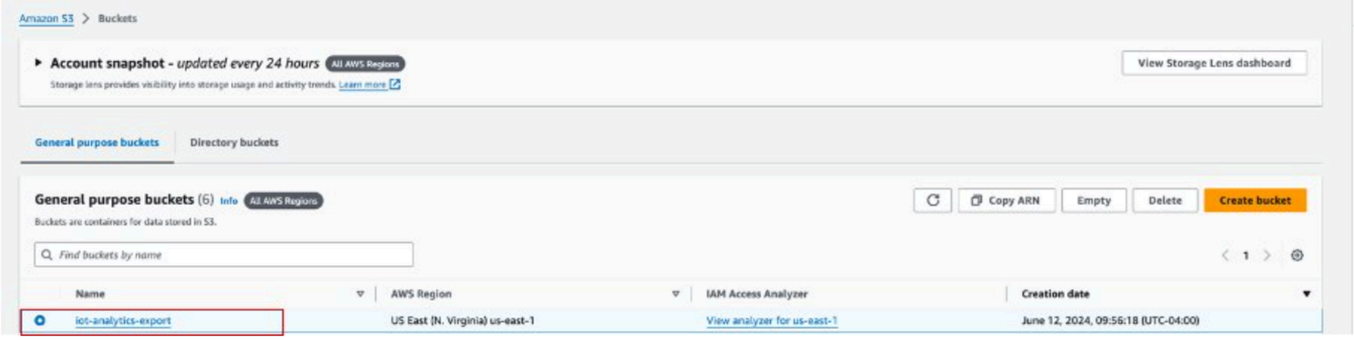
```
# Cloudformation Template to migrate an AWS IoT Analytics datastore to an external
 dataset
AWSTemplateFormatVersion: 2010-09-09
Description: Migrate an AWS IoT Analytics datastore to an external dataset
Parameters:
  DatastoreName:
    Type: String
    Description: The name of the datastore to migrate.
    AllowedPattern: ^[a-zA-Z0-9_]+$
  TimeRange:
    Type: String
    Description: |
      This is an optional argument to split the source data into multiple files.
      The value should follow the SQL syntax of WHERE clause.
      E.g. WHERE DATE(Item_TimeStamp) BETWEEN '09/16/2010 05:00:00' and '09/21/2010
 09:00:00'.
    Default: ''
  MigrationS3Bucket:
    Type: String
    Description: The S3 Bucket where the datastore will be migrated to.
    AllowedPattern: (?!(^xn--|.+-s3alias$))^[a-z0-9][a-z0-9-]{1,61}[a-z0-9]$
  MigrationS3BucketPrefix:
    Type: String
    Description: The prefix of the S3 Bucket where the datastore will be migrated
 to.
    Default: ''
    AllowedPattern: (^([a-zA-Z0-9.\-_]*\/)*$)|(^$)
Resources:
  # IAM Role to be assumed by the AWS IoT Analytics service to access the external
 dataset
  DatastoreMigrationRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Principal:
              Service: iotanalytics.amazonaws.com
            Action: sts:AssumeRole
      Policies:
        - PolicyName: AllowAccessToExternalDataset
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              - Effect: Allow
                Action:
                  - s3:GetBucketLocation
                  - s3:GetObject
                  - s3:ListBucket
                  - s3:ListBucketMultipartUploads
                  - s3:ListMultipartUploadParts
                  - s3:AbortMultipartUpload
                  - s3:PutObject
                  - s3:DeleteObject
                Resource:
                  - !Sub arn:aws:s3:::${MigrationS3Bucket}
                  - !Sub arn:aws:s3:::${MigrationS3Bucket}/
${MigrationS3BucketPrefix}*

  # This dataset that will be created in the external S3 Export
  MigratedDataset:
    Type: AWS::IoTAnalytics::Dataset
    Properties:
      DatasetName: !Sub ${DatastoreName}_generated
      Actions:
        - ActionName: SqlAction
          QueryAction:
            SqlQuery: !Sub SELECT * FROM ${DatastoreName} ${TimeRange}
      ContentDeliveryRules:
        - Destination:
            S3DestinationConfiguration:
              Bucket: !Ref MigrationS3Bucket
              Key: !Sub ${MigrationS3BucketPrefix}${DatastoreName}/!
{iotanalytics:scheduleTime}/!{iotanalytics:versionId}.csv
              RoleArn: !GetAtt DatastoreMigrationRole.Arn
      RetentionPeriod:
        Unlimited: true
      VersioningConfiguration:
        Unlimited: true
```

2. Determine the Amazon IoT Analytics datastore that requires data to be exported. For this guide, we will use a sample datastore named `iot_analytics_datastore`.



□	Name	Status	Last message arrival time	Storage information	File format	Created	Last updated	Data partitions
□	iotanalytics_datastore	Active	Jun 12, 2024 9:53:08 AM -0400	Service managed	JSON	Jun 11, 2024 1:59:17 PM -0400	Jun 11, 2024 1:59:17 PM -0400	Not enabled

3. Create or identify an Amazon S3 bucket where the data will be exported. For this guide, we will use the `iot-analytics-export` bucket.



Account snapshot - *updated every 24 hours* All AWS Regions

Storage lens provides visibility into storage usage and activity trends. Learn more

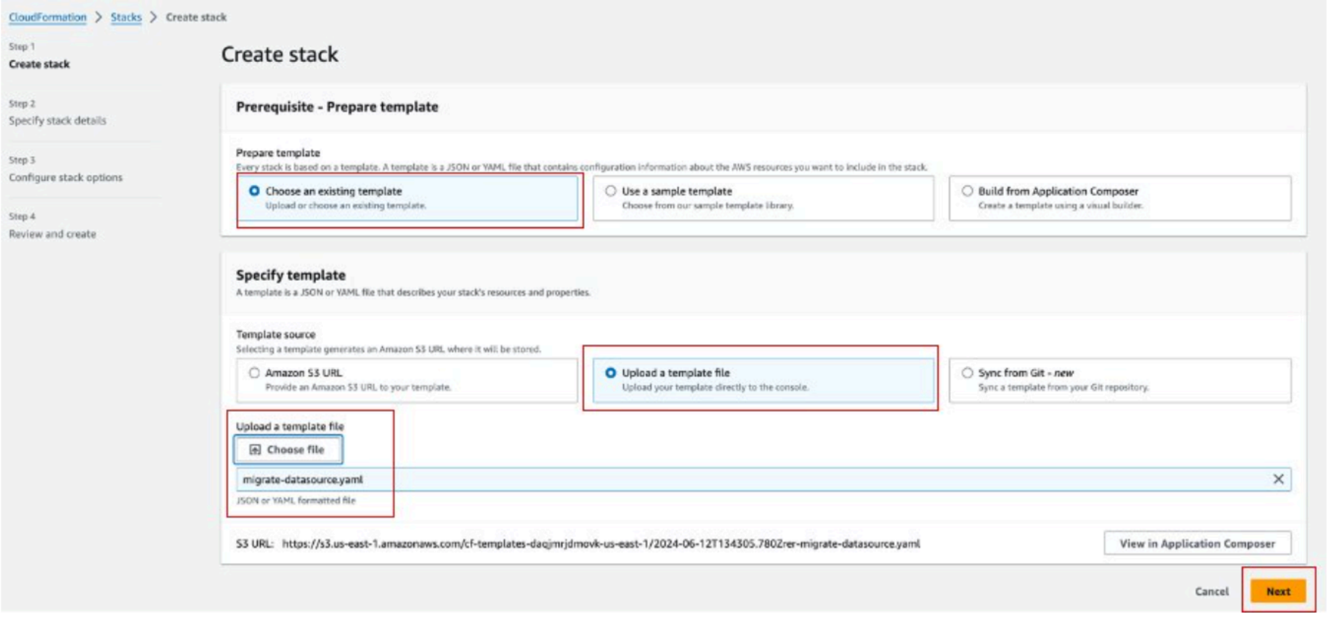
General purpose buckets (6)

Buckets are containers for data stored in S3.

	Name	AWS Region	IAM Access Analyzer	Creation date
●	iot-analytics-export	US East (N. Virginia) us-east-1	View analyzer for us-east-1	June 12, 2024, 09:56:18 (UTC-04:00)

4. Create the Amazon CloudFormation stack.

   - Navigate to the https://console.amazonaws.cn/cloudformation.
   - Click on **Create stack** and select **With new resources (standard)**.
   - Upload the `migrate-datasource.yaml` file.



Create stack

Prerequisite - Prepare template

Prepare template

Every stack is based on a template. A template is a JSON or YAML file that contains configuration information about the AWS resources you want to include in the stack.

● Choose an existing template – Upload or choose an existing template.

○ Use a sample template – Choose from our sample template library.

○ Build from Application Composer – Create a template using a visual builder.

Specify template

A template is a JSON or YAML file that describes your stack's resources and properties.

Template source

Selecting a template generates an Amazon S3 URL where it will be stored.

○ Amazon S3 URL – Provide an Amazon S3 URL to your template.

● Upload a template file – Upload your template directly to the console.

○ Sync from Git - *new* – Sync a template from your Git repository.

Upload a template file

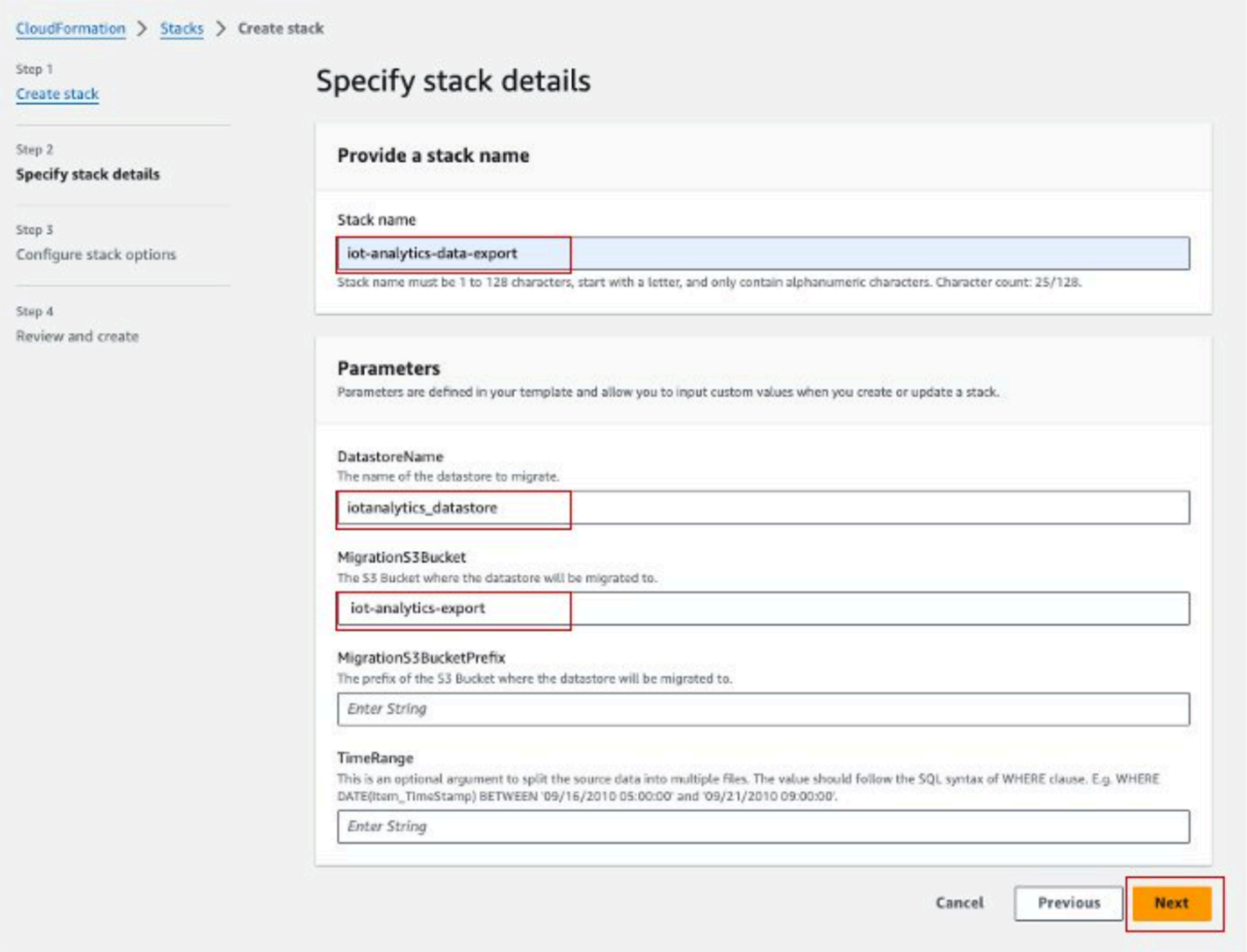
migrate-datasource.yaml

JSON or YAML formatted file

S3 URL: https://s3.us-east-1.amazonaws.com/cf-templates-daqjmrjdmovk-us-east-1/2024-06-12T134305.780Zrer-migrate-datasource.yaml

5. Enter a stack name and provide the following parameters:

   - **DatastoreName**: The name of the Amazon IoT Analytics datastore you want to migrate.
   - **MigrationS3Bucket**: The Amazon S3 bucket where the migrated data is stored.
   - **MigrationS3BucketPrefix** (Optional) : The prefix for the Amazon S3 bucket.
   - **TimeRange** (Optional) : A `SQL WHERE` clause to filter the data being exported, allowing for splitting the source data into multiple files based on the specified time range.



Specify stack details

Provide a stack name

Stack name: iot-analytics-data-export

Stack name must be 1 to 128 characters, start with a letter, and only contain alphanumeric characters. Character count: 25/128.

Parameters

Parameters are defined in your template and allow you to input custom values when you create or update a stack.

DatastoreName
The name of the datastore to migrate.
iotanalytics_datastore

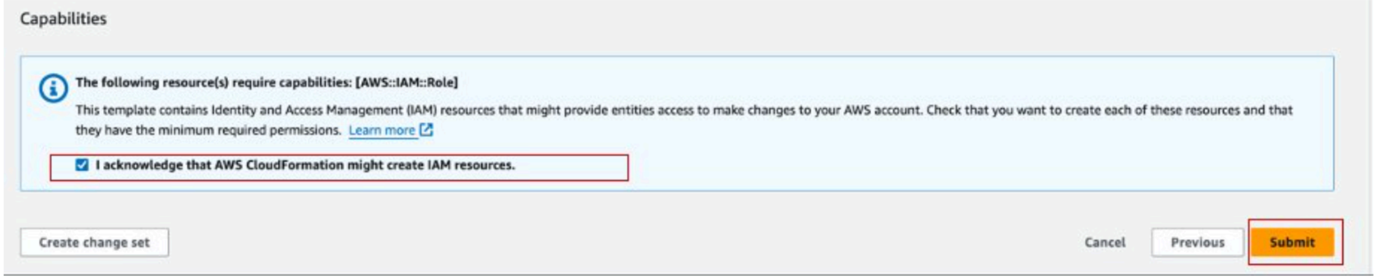
MigrationS3Bucket
The S3 Bucket where the datastore will be migrated to.
iot-analytics-export

MigrationS3BucketPrefix
The prefix of the S3 Bucket where the datastore will be migrated to.
Enter String

TimeRange
This is an optional argument to split the source data into multiple files. The value should follow the SQL syntax of WHERE clause. E.g. WHERE DATE(Item_TimeStamp) BETWEEN '09/16/2010 05:00:00' and '09/21/2010 09:00:00'.
Enter String

6. Click **Next** on the Configure stack options screen.

7. Selecting the checkbox to acknowledge creating IAM resources, and click **Submit**.



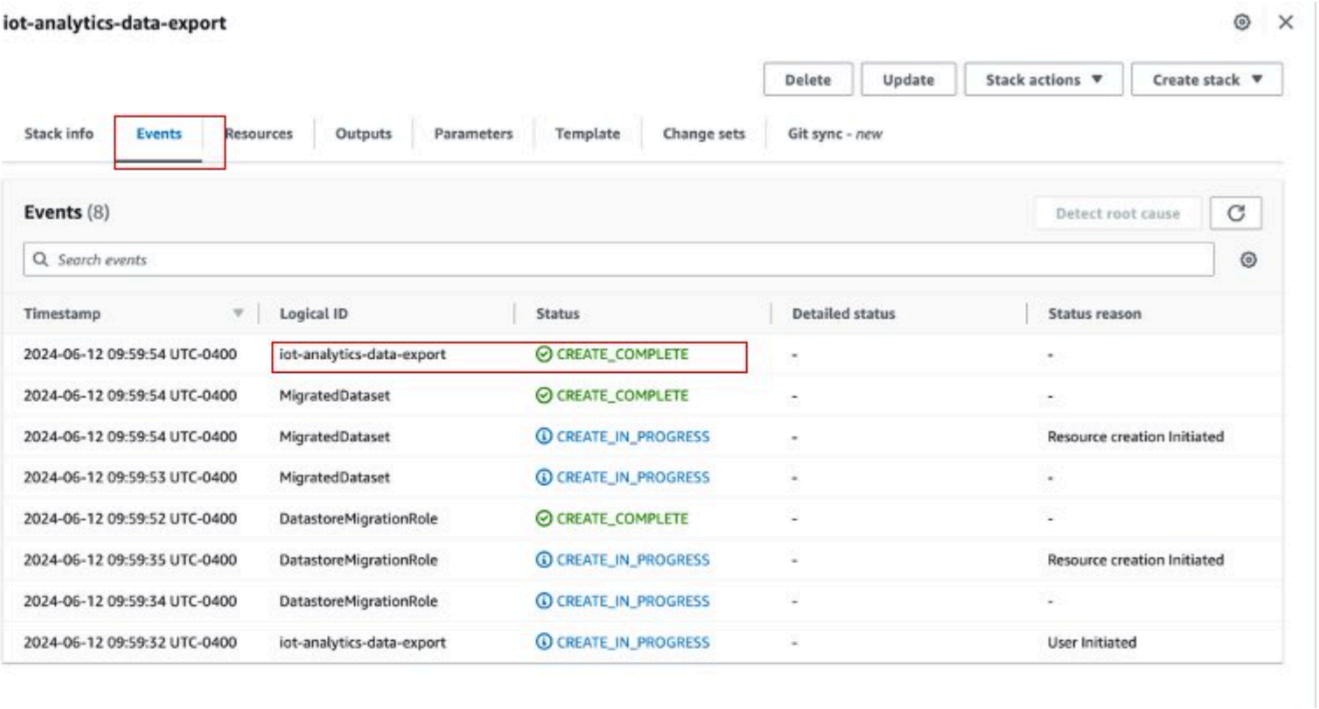
Capabilities

The following resource(s) require capabilities: [AWS::IAM::Role]

This template contains Identity and Access Management (IAM) resources that might provide entities access to make changes to your AWS account. Check that you want to create each of these resources and that they have the minimum required permissions. Learn more

☑ I acknowledge that AWS CloudFormation might create IAM resources.

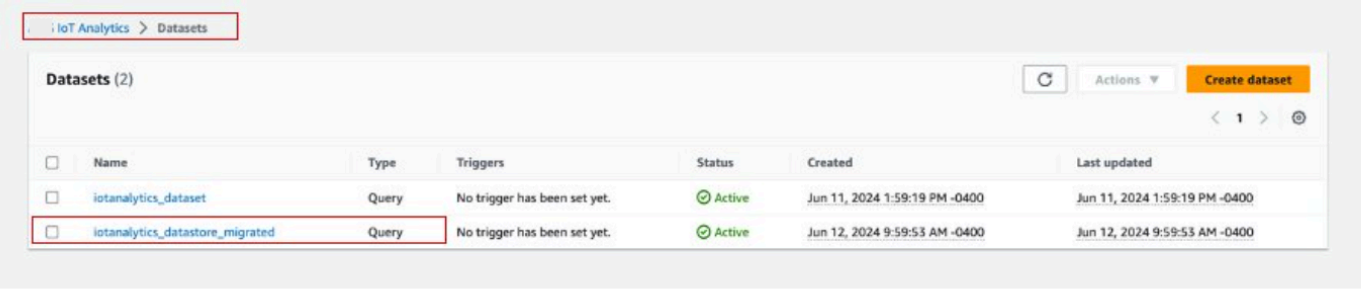
8. Review stack creation on the **Events** tab for completion.



Events (8)

Timestamp	Logical ID	Status	Detailed status	Status reason
2024-06-12 09:59:54 UTC-0400	iot-analytics-data-export	CREATE_COMPLETE	-	-
2024-06-12 09:59:54 UTC-0400	MigratedDataset	CREATE_COMPLETE	-	-
2024-06-12 09:59:54 UTC-0400	MigratedDataset	CREATE_IN_PROGRESS	-	Resource creation Initiated
2024-06-12 09:59:53 UTC-0400	MigratedDataset	CREATE_IN_PROGRESS	-	-
2024-06-12 09:59:52 UTC-0400	DatastoreMigrationRole	CREATE_COMPLETE	-	-
2024-06-12 09:59:35 UTC-0400	DatastoreMigrationRole	CREATE_IN_PROGRESS	-	Resource creation Initiated
2024-06-12 09:59:34 UTC-0400	DatastoreMigrationRole	CREATE_IN_PROGRESS	-	-
2024-06-12 09:59:32 UTC-0400	iot-analytics-data-export	CREATE_IN_PROGRESS	-	User Initiated

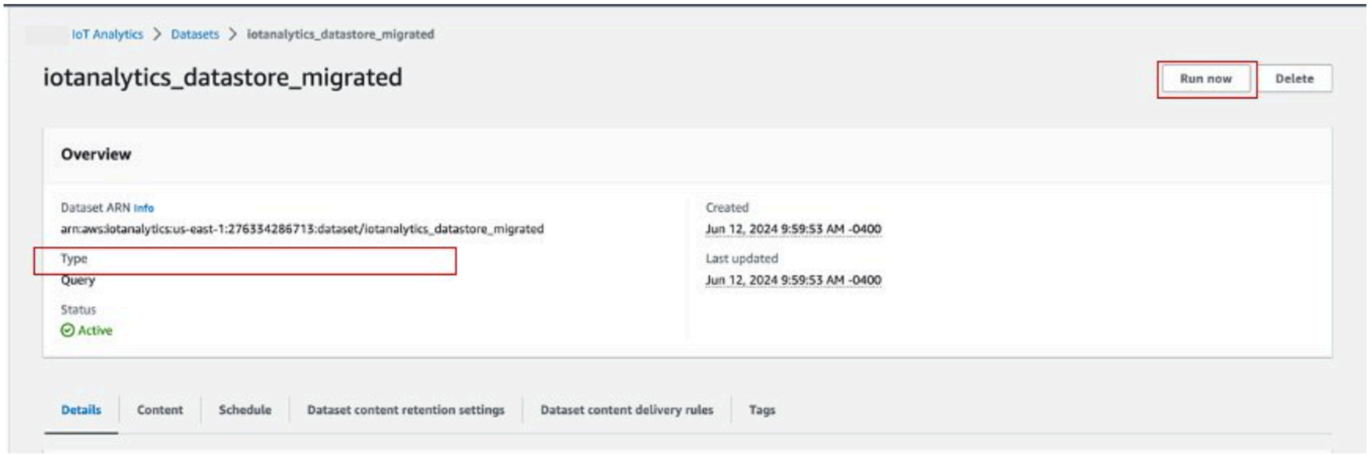
9. On successful stack completion, navigate to **Amazon IoT Analytics** → **Datasets** to view the migrated dataset.



Datasets (2)

□	Name	Type	Triggers	Status	Created	Last updated
□	iotanalytics_dataset	Query	No trigger has been set yet.	Active	Jun 11, 2024 1:59:19 PM -0400	Jun 11, 2024 1:59:19 PM -0400
□	iotanalytics_datastore_migrated	Query	No trigger has been set yet.	Active	Jun 12, 2024 9:59:53 AM -0400	Jun 12, 2024 9:59:53 AM -0400

10. Select the generated dataset, and click **Run now** to export the dataset.



IoT Analytics > Datasets > iotanalytics_datastore_migrated
Run now
Delete
Details
Content
Schedule
Dataset content retention settings
Dataset content delivery rules
Tags

iotanalytics_datastore_migrated

Overview

Dataset ARN Info
arn:aws:iotanalytics:us-east-1:276334286713:dataset/iotanalytics_datastore_migrated

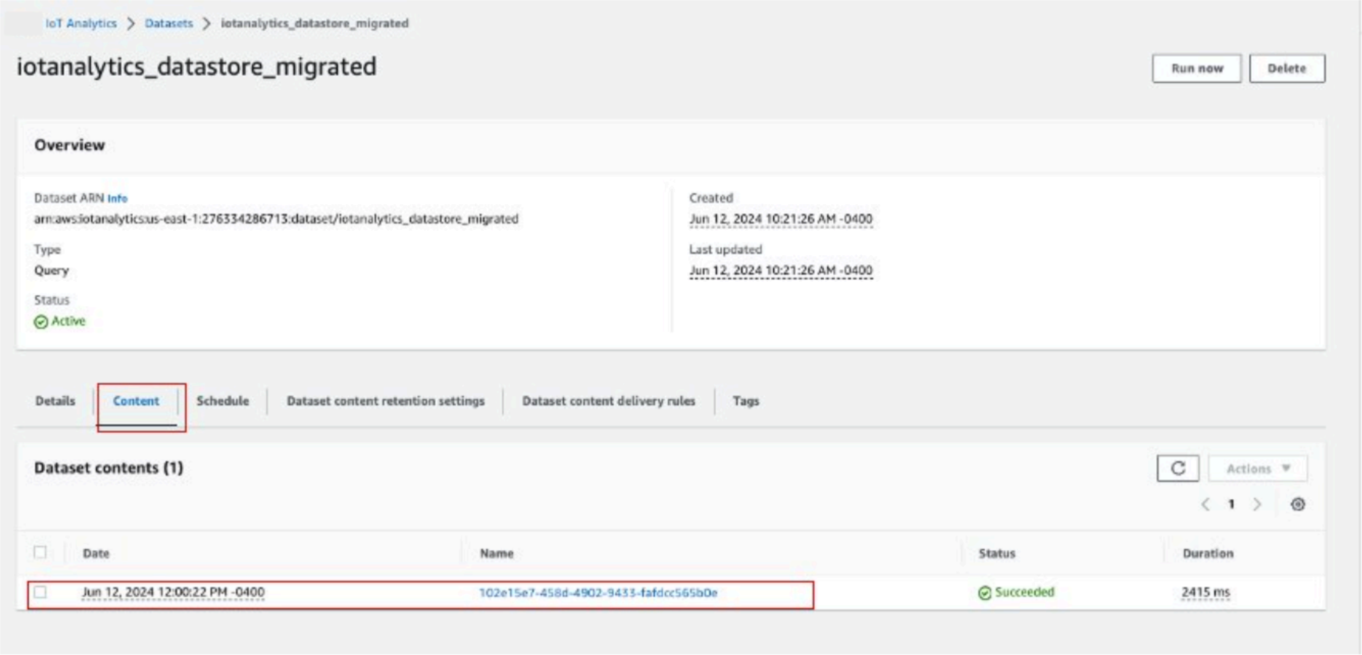
Type
Query

Status
Active

Created
Jun 12, 2024 9:59:53 AM -0400

Last updated
Jun 12, 2024 9:59:53 AM -0400

11. The content can be viewed on the **Content** tab of the dataset.


IoT Analytics > Datasets > iotanalytics_datastore_migrated
Run now
Delete
Details
Content
Schedule
Dataset content retention settings
Dataset content delivery rules
Tags
Actions ▼

iotanalytics_datastore_migrated

Overview

Dataset ARN Info
arn:aws:iotanalytics:us-east-1:276334286713:dataset/iotanalytics_datastore_migrated

Type
Query

Status
Active

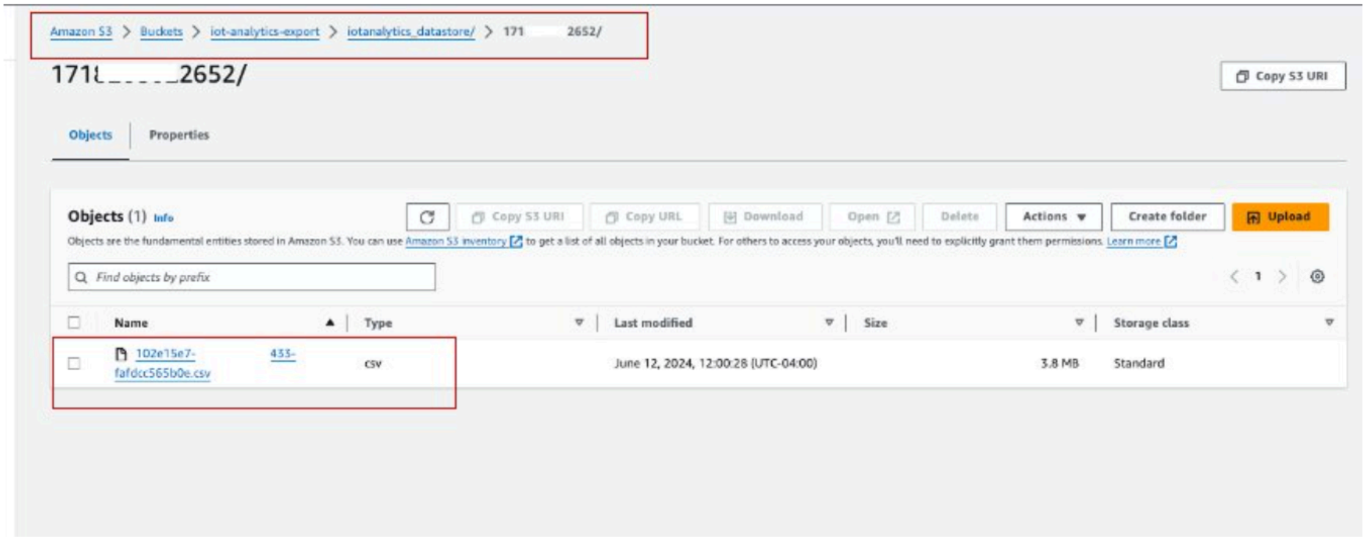
Created
Jun 12, 2024 10:21:26 AM -0400

Last updated
Jun 12, 2024 10:21:26 AM -0400

Dataset contents (1)

	Date	Name	Status	Duration
□	Jun 12, 2024 12:00:22 PM -0400	102e15e7-458d-4902-9433-fafdcc565b0e	Succeeded	2415 ms

12. Finally, review the exported content by opening the **iot-analytics-export** bucket in the Amazon S3 console.



171[redacted]2652/

Objects (1) Info

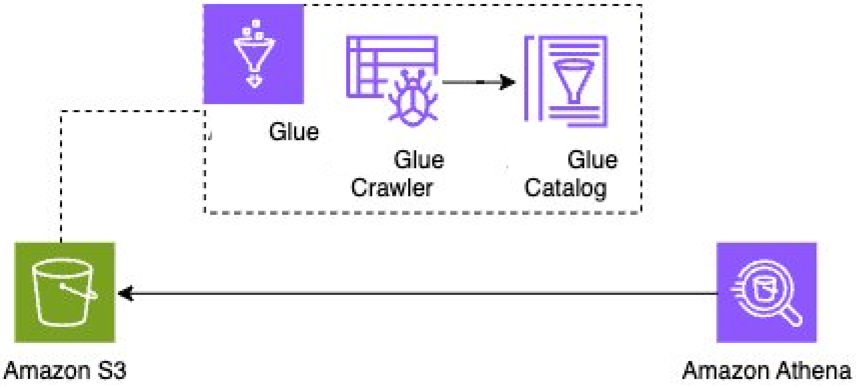
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

	Name	Type	Last modified	Size	Storage class
□	102e15e7-[redacted]433-fafdcc565b0e.csv	csv	June 12, 2024, 12:00:28 (UTC-04:00)	3.8 MB	Standard

Run on-demand queries for both patterns

As you migrate your Amazon IoT Analytics workloads to Amazon Kinesis Data Streams, or Amazon Data Firehose, leveraging Amazon Glue and Amazon Athena can further streamline your data analysis process. Amazon Glue simplifies data preparation and transformation, while Amazon

Athena enables quick, serverless querying of your data. Together, they provide a powerful, scalable, and cost-effective solution for analyzing Amazon IoT data.



Summary

Migrating your Amazon IoT Analytics workload from Amazon IoT Analytics to Amazon Kinesis Data Streams, Amazon S3, and enhances your ability to handle large-scale, complex Amazon IoT data. This architecture provides scalable, durable storage and powerful analytics capabilities, enabling you to gain deeper insights from your IoT data in real-time.

Cleaning up resources created using Amazon CloudFormation is essential to avoid unexpected costs once the migration has completed.

Refer to the Amazon IoT Analytics pricing page for costs involved in data migration. Consider deleting the newly created dataset when finished to avoid any unnecessary expenses.

Full dataset export: To export the complete dataset without any time-based splitting, you can also use Amazon IoT Analytics console and set a content delivery rule accordingly.

By following the migration guide, you can seamlessly transition your data ingestion and processing pipelines, ensuring continuous and reliable data flow. Leveraging Amazon Glue and Amazon Athena further simplifies data preparation and querying, allowing you to perform sophisticated analyzes without managing any infrastructure.

This approach empowers you to scale your Amazon IoT Analytics efforts effectively, making it easier to adapt to the growing demands of your business and extract maximum value from your Amazon IoT data.

Getting started with Amazon IoT Analytics (console)

Use this tutorial to create the Amazon IoT Analytics resources (also known as components) that you need to discover useful insights about your IoT device data.

Notes

- If you enter uppercase characters in the following tutorial, Amazon IoT Analytics automatically changes them to lowercase.
- The Amazon IoT Analytics console has a one-click getting started feature to create a channel, pipeline, data store, and dataset. You can find this feature when you sign in to the Amazon IoT Analytics console.
  - This tutorial walks you through each step to create your Amazon IoT Analytics resources.

Follow the instructions below to create an Amazon IoT Analytics channel, pipeline, data store, and dataset. The tutorial also shows you how to use the Amazon IoT Core console to send messages that will be ingested into Amazon IoT Analytics.

Topics

Sign in to the Amazon IoT Analytics console

To get started, you must have an Amazon account. If you already have an Amazon account, navigate to the https://console.amazonaws.cn/iotanalytics/.

If you don't have an Amazon account, follow these steps to create one.

To create an Amazon account

1. Open https://portal.amazonaws.cn/billing/signup.
2. Follow the online instructions.

   Part of the sign-up procedure involves receiving a phone call or text message and entering a verification code on the phone keypad.

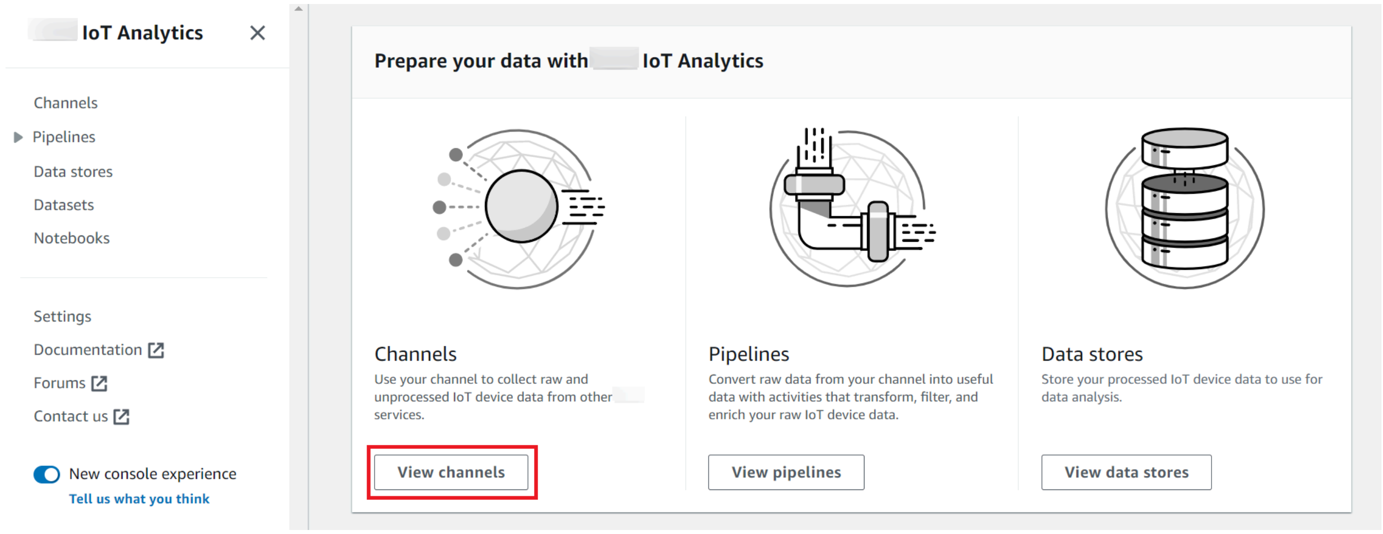
   When you sign up for an Amazon Web Services account, an *Amazon Web Services account root user* is created. The root user has access to all Amazon Web Services services and resources in the account. As a security best practice, assign administrative access to a user, and use only the root user to perform tasks that require root user access.
3. Sign in to the Amazon Web Services Management Console and navigate to the https://console.amazonaws.cn/iotanalytics/.

Create a channel

A channel collects and archives raw, unprocessed, and unstructured IoT device data. Follow these steps to create your channel.

To create a channel

1. In the https://console.amazonaws.cn/iotanalytics/, in the **Prepare your data with Amazon IoT Analytics** section, choose **View channels**.



Prepare your data with IoT Analytics

Channels

Use your channel to collect raw and unprocessed IoT device data from other services.

Pipelines

Convert raw data from your channel into useful data with activities that transform, filter, and enrich your raw IoT device data.

Data stores

Store your processed IoT device data to use for data analysis.

> **Tip**
>
> You can also choose **Channels** from the navigation pane.

2. On the **Channels** page, choose **Create channel**.
3. On the **Specify channel details** page, enter the details about your channel.

   a. Enter a channel name that is unique and that you can easily identify.

   b. (Optional) For **Tags**, add one or more custom tags (key-value pairs) to your channel. Tags can help you identify your resources that you create for Amazon IoT Analytics.

   c. Choose **Next**.
4. Amazon IoT Analytics stores your raw, unprocessed IoT device data in an Amazon Simple Storage Service (Amazon S3) bucket. You can choose your own Amazon S3 bucket, which you can access and manage, or Amazon IoT Analytics can manage the Amazon S3 bucket for you.

   a. In this tutorial, for **Storage type**, choose **Service managed storage**.

   b. For **Choose how long to store your raw data**, choose **Indefinitely**.

   c. Choose **Next**.
5. On the **Configure source** page, enter information for Amazon IoT Analytics to collect message data from Amazon IoT Core.

   a. Enter an Amazon IoT Core topic filter, for example, `update/environment/dht1`. Later in this tutorial, you will use this topic filter to send message data to your channel.

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

   c. Choose **Next**.

6. Review your choices and then choose **Create channel**.
7. Verify that your new channel appears on the **Channels** page.

Create a data store

A data store receives and stores your message data. A data store isn't a database. Instead, a data store is a scalable and queryable repository in an Amazon S3 bucket. You can use multiple data stores for messages from different devices or locations. Or, you can filter message data depending on your pipeline configuration and requirements.

Follow these steps to create a data store.

To create a data store

1. In the https://console.amazonaws.cn/iotanalytics/, in the **Prepare your data with Amazon IoT Analytics** section, choose **View data stores**.
2. On the **Data stores** page, choose **Create data store**.
3. On the **Specify data store details** page, enter basic information about your data store.

   a. For **Data store ID**, enter a unique data store ID. You can't change this ID after you create it.

   b. (Optional) For **Tags**, choose **Add new tag** to add one or more custom tags (key-value pairs) to your data store. Tags can help you identify your resources that you create for Amazon IoT Analytics.

   c. Choose **Next**.

4. On the **Configure storage type** page, specify how to store your data.

   a. For **Storage type**, choose **Service managed storage**.

   b. For **Configure how long you want to keep your processed data**, choose **Indefinitely**.

   c. Choose **Next**.

5. Amazon IoT Analytics data stores support JSON and Parquet file formats. For your data store data format, choose **JSON** or **Parquet**. See File formats for more information about Amazon IoT Analytics supported file types.

Choose **Next**.

6. (Optional) Amazon IoT Analytics supports custom partitions in your data store so you can query on pruned data to improve latency. For more information about supported custom partitions, see Custom partitions.

   Choose **Next**.
7. Review your choices and then choose **Create data store**.
8. Verify that your new data store appears on the **Data stores** page.

Create a pipeline

You must create a pipeline to connect a channel to a data store. A basic pipeline only specifies the channel that collects the data and identifies the data store to which the messages are sent. For more information, see Pipeline activities.

For this tutorial, you create a pipeline that only connects a channel to a data store. Later, you can add pipeline activities to process this data.

Follow these steps to create a pipeline.

To create a pipeline

1. In the https://console.amazonaws.cn/iotanalytics/, in the **Prepare your data with Amazon IoT Analytics** section, choose **View pipelines**.

   > **Tip**
   >
   > You can also choose **Pipelines** from the navigation pane.
2. On the **Pipelines** page, choose **Create pipeline**.
3. Enter the details about your pipeline.

   a. In **Setup pipeline ID and sources**, enter a pipeline name.

   b. Choose your pipeline's source, which is an Amazon IoT Analytics channel that your pipeline will read messages from.

   c. Specify your pipeline's output, which is the data store where your processed message data is stored.

   d. (Optional) For **Tags**, add one or more custom tags (key-value pairs) to your pipeline.

   e. On the **Infer message attributes** page, enter an attribute name and an example value, choose a data type from the list, and then choose **Add attribute**.

   f. Repeat the previous step for as many attributes as you need, and then choose **Next**.

   g. You won't add any pipeline activities right now. On the **Enrich, transform, and filter messages** page, choose **Next**.

4. Review your choices and then choose **Create pipeline**.

5. Verify that your new pipeline appears on the **Pipelines** page.

> ⓘ **Note**
>
> You created Amazon IoT Analytics resources so that they can do the following:
>
> - Collect raw, unprocessed IoT device message data with a *channel*.
> - Store your IoT device message data in a *data store*.
> - Clean, filter, transform, and enrich your data with a *pipeline*.
>
> Next, you will create an Amazon IoT Analytics SQL dataset to discover useful insights about your IoT device.

Create a dataset

> ⓘ **Note**
>
> A dataset is typically a collection of data that might or might not be organized in tabular form. In contrast, Amazon IoT Analytics creates your dataset by applying a SQL query to data in your data store.

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

To create a dataset

1. In the https://console.amazonaws.cn/iotanalytics/, in the **left navigation pane**, choose **Datasets**.
2. On the **Create dataset** page, choose **Create SQL**.
3. On the **Specify dataset details** page, specify the details of your dataset.
   a. Enter a name for your dataset.
   b. For **Data store source**, choose the the unique ID that identifies the data store that you created earlier.
   c. (Optional) For **Tags**, add one or more custom tags (key-value pairs) to your dataset.
4. Use SQL expressions to query your data and answer analytical questions. The results of your query are stored in this dataset.
   a. In the **Author query** field, enter a SQL query that uses a wildcard to show up to five rows of data.

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

      For more information about supported SQL functionality in Amazon IoT Analytics, see SQL expressions in Amazon IoT Analytics.
   b. You can choose **Test query** to validate that your input is correct and display the results in a table following the query.

      Note

      - At this point in the tutorial your datastore might be empty. Running a SQL query on an empty datastore won't return results, so you might see only `__dt`.
      - You must be careful to limit your SQL query to a reasonable size so that it does not run for an extended period because Athena limits the maximum number of running queries. Because of this, you must be careful to limit the SQL query to a reasonable size.

> We suggest using a `LIMIT` clause in your query during testing. After the test succeeds, you can remove this clause.

5. (Optional) When you create dataset contents using data from a specified time frame, some data might not arrive in time for processing. To allow for a delay, you can specify an offset, or delta. For more information, see Getting late data notifications through Amazon CloudWatch Events.

   You won't configure a data selection filter at this point. On the **Configure data selection filter** page, choose **Next**.
6. (Optional) You can schedule this query to run regularly to refresh the dataset. Dataset schedules can be created and edited at any time.

   You won't schedule a recurring run of the query at this point, so on the **Set query schedule** page choose **Next**.
7. Amazon IoT Analytics will create versions of this dataset content and store your analytics results for the specified period. We recommend 90 days, however you can opt to set your custom retention policy. You may also limit the number of stored versions of your dataset content.

   You can use the default dataset retention period as **Indefinitely** and keep **Versioning** disabled. On the **Configure the results of your analytics** page, choose **Next**.
8. (Optional) You can configure the delivery rules of your dataset results to a specific destination, such as Amazon IoT Events.

   You won't deliver your results elsewhere in this tutorial, so on the **Configure dataset content delivery rules** page, choose **Next**.
9. Review your choices and then choose **Create dataset**.
10. Verify that your new dataset appears on the **Datasets** page.

Send message data with Amazon IoT

If you have a channel that routes data to a pipeline, which stores data in a data store where it can be queried, then you're ready to send IoT device data into Amazon IoT Analytics. You can send data into Amazon IoT Analytics by using the following options:

- Use the Amazon IoT message broker.
- Use the Amazon IoT Analytics BatchPutMessage API operation.

In the following steps, you send message data from the Amazon IoT message broker in the Amazon IoT Core console so that Amazon IoT Analytics can ingest this data.

> **Note**
>
> When you create topic names for your messages, note the following:
>
> - Topic names are not case sensitive. Fields named `example` and `EXAMPLE` in the same payload are considered duplicates.
> - Topic names can't begin with the $ character. Topics that begin with $ are reserved topics and can only be used by Amazon IoT.
> - Don't include personally identifiable information in your topic names because this information can appear in unencrypted communications and reports.
> - Amazon IoT Core can't send messages between Amazon accounts or Amazon Regions.

To send message data with Amazon IoT

1. Sign in to the Amazon IoT console.
2. In the navigation pane, choose **Test**, and then choose **MQTT test client**.
3. On the **MQTT test client** page, choose **Publish to a topic**.
4. For **Topic name**, enter a name that will match the topic filter that you entered when you created a channel. This example uses `update/environment/dht1`.
5. For **Message payload**, enter the following JSON contents.

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

6. (Optional) Choose **Add Configuration** for additional message protocol options.
7. Choose **Publish**.

This publishes a message that is captured by your channel. Your pipeline then routes the message to your data store.

Check the progress of Amazon IoT messages

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

To check the progress of Amazon IoT messages

1. Sign in to the https://console.amazonaws.cn/iotanalytics/.
2. In the navigation pane, choose **Channels**, and then choose the channel name that you created earlier.
3. On the **Channel's details** page, scroll down to the **Monitoring** section, and then adjust the displayed time frame (**1h 3h 12h 1d 3d 1w**). Choose a value such as **1w** to view data for the last week.

You can use a similar feature to monitor for pipeline activity runtime and errors on the **Pipeline's details** page. In this tutorial, you haven't specified activities as part of the pipeline, so you shouldn't see any runtime errors.

To monitor pipeline activity

1. In the navigation pane, choose **Pipelines**, and then choose the name of the pipeline that you created earlier.
2. On the **Pipeline's details** page, scroll down to the **Monitoring** section, and then adjust the displayed time frame by choosing one of the time frame indicators (**1h 3h 12h 1d 3d 1w**).

Access query results

The dataset content is a file containing the result of your query, in CSV format.

1. In the https://console.amazonaws.cn/iotanalytics/, in the left navigation pane, choose **Datasets**.
2. On the **Datasets** page, choose the name of the dataset that you created previously.
3. On the dataset information page, in the upper-right corner, choose **Run now**.

4. To check if the dataset is ready, look under the dataset for a message similar to **You've successfully started the query for your dataset**. The **Dataset content** tab contains the query results and displays **Succeeded**.
5. To preview the results of your successful query, on the **Dataset contents** tab, select the query name. To view or save the CSV file that contains the query results, choose **Download**.

> **Note**
>
> Amazon IoT Analytics can embed the HTML portion of a Jupyter Notebook on the **Dataset contents** page. For more information, see Visualizing Amazon IoT Analytics data with the console.

Explore your data

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

Amazon Simple Storage Service

You can send dataset contents to an Amazon 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 the CreateDataset operation.

Amazon IoT Events

You can send dataset contents as an input to Amazon IoT Events, a service that enables you to monitor devices or processes for failures or changes in operation, and to initiate additional actions when such events occur.

To do this, create a dataset using the CreateDataset operation and specify an Amazon IoT Events input in the field `contentDeliveryRules :: destination :: iotEventsDestinationConfiguration :: inputName`. You must also specify the `roleArn` of a role, which grants Amazon IoT Analytics permissions to run `iotevents:BatchPutMessage`. Whenever the datasets contents are created, Amazon IoT Analytics will send each dataset content entry as a message to the specified Amazon IoT Events input. For example, if your dataset contains the following content.

```
"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 Amazon IoT Analytics sends messages that contain fields like the following.

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

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

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

Jupyter Notebook

Jupyter Notebook is an open source solution for using scripting languages to run ad-hoc data exploration and advanced analyses. You can dive deep and apply more complex analyses and use machine learning methods, such as k-means clustering and regression models for prediction, on your IoT device data.

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

1. Navigate to the SageMaker AI 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.
2. Go to the IAM console and modify the SageMaker AI 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 enter **`GetDatasetContent`** in the search box and choose it. Choose **Review Policy**.
   c. Review the policy for accuracy, enter a name, and then choose **Create policy**.

This gives the newly created role permission to read a dataset from Amazon IoT Analytics.

1. Return to the https://console.amazonaws.cn/iotanalytics/, and in the left navigation pane, choose **Notebooks**. On the **Notebooks** page, choose **Create notebook**.
2. On the **Select a template** page, choose **IoTA blank template**.
3. On the **Set up notebook** page, enter a name for your notebook. In **Select dataset source**, choose and then choose the dataset you created earlier. In **Select a notebook instance**, choose the notebook instance you created in SageMaker AI.
4. After you review your choices, choose **Create Notebook**.
5. On the **Notebooks** page, your notebook instance will open in the Amazon SageMaker AI console.

Notebook templates

The Amazon IoT Analytics notebook templates contain Amazon authored machine learning models and visualizations to help you get started with Amazon IoT Analytics use cases. You can use these notebook templates to learn more or reuse them to fit your IoT device data and deliver immediate value.

You can find the following notebook templates in the Amazon IoT Analytics console:

- **Detecting contextual anomalies** – Application of contextual anomaly detection in measured wind speed with a Poisson Exponentially Weighted Moving Average (PEWMA) model.
- **Solar panel output forecasting** – Application of piecewise, seasonal, and linear time series models to predict the output of solar panels.
- **Predictive maintenance on jet engines –** Application of multivariate Long Short-Term Memory (LSTM) neural networks and logistic regression to predict jet engine failure.
- **Smart home customer segmentation –** Application of k-means and Principal Component Analysis (PCA) analysis to detect different customer segments in data of smart home usage.
- **Smart city congestion forecasting** – Application of LSTM to predict the utilization rates for city highways.
- **Smart city air quality forecasting** – Application of LSTM to predict particulate pollution in city centers.

Getting started with Amazon IoT Analytics

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

⚠ **Important**

Use the `aws iotanalytics` command to interact with Amazon IoT Analytics using the Amazon CLI. Use the `aws iot` command to interact with other parts of the IoT system using the Amazon CLI.

ⓘ **Note**

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

Creating a channel

A channel collects and archives raw, unprocessed message data before publishing this data to a pipeline. 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 Amazon IoT messages to be ingested into Amazon IoT Analytics, you can create an Amazon IoT Rules Engine rule to send the messages to this channel. This is shown later in Ingesting data to Amazon IoT Analytics. Another way to get the data in to a channel is to use the Amazon 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 Amazon 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 Amazon 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).

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 might need to reprocess channel data to ensure that previously ingested data is included in dataset contents. See Reprocessing channel data.

Creating a data store

A data store receives and stores your messages. It is not a database but a scalable and queryable repository of your messages. You can create multiple data stores to store messages that comes from different devices or locations, or your can use a single data store to receive all of your Amazon 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
```

Amazon S3 policies for Amazon IoT Analytics resources

You can store processed data store messages in an Amazon S3 bucket managed by Amazon IoT Analytics or in one that you manage. When you create a data store, select the Amazon S3 bucket you want by using the `datastoreStorage` API parameter. 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 Amazon IoT Analytics permission to perform these actions on your Amazon S3 bucket for you:

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

If you use the data store as a source for an SQL query dataset, set up an Amazon S3 bucket policy that grants Amazon IoT Analytics permission to invoke Amazon Athena queries on the contents of your bucket.

> **Note**
>
> We recommend that you specify `aws:SourceArn` in your bucket policy to help prevent the confused deputy security problem. This restricts access by allowing only those requests that come from a specified account. For more information about the confused deputy problem, see the section called “Cross-service confused deputy prevention”.

The following is an example of a bucket policy that grants these 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:::DOC-EXAMPLE-BUCKET",
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
            ],
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": [
            "arn:aws:iotanalytics:us-east-1:123456789012:dataset/DOC-EXAMPLE-DATASET",
            "arn:aws:iotanalytics:us-east-1:123456789012:datastore/DOC-EXAMPLE-DATASTORE"
          ]
        }
      }
    }
  ]
}
```

For more information, see Cross-account access in the *Amazon Athena User Guide*.

Note

If you update the options or permissions of your customer managed data store, you might need to reprocess channel data to ensure that any previously ingested data is included in dataset contents. For more information, see Reprocessing channel data.

File formats

Amazon IoT Analytics data stores currently support JSON and Parquet file formats. The default file format is JSON.

- JSON (JavaScript Object Notation) - A text format that supports name-value pairs and ordered lists of values.
- Apache Parquet - A columnar storage format used to efficiently store and query large volumes of data.

To configure the file format of the Amazon IoT Analytics data store, you can use the `FileFormatConfiguration` object when you create the data store.

`fileFormatConfiguration`

Contains the configuration information of file formats. Amazon IoT Analytics data stores support JSON and Parquet.

The default file format is JSON. You can specify only one format. You can't change the file format after you create the data store.

`jsonConfiguration`

Contains the configuration information of the JSON format.

`parquetConfiguration`

Contains the configuration information of the Parquet format.

`schemaDefinition`

Information needed to define a schema.

`columns`

Specifies one or more columns that store your data.

Each schema can have up to 100 columns. Each column can have up to 100 nested types.

`name`

The name of the column.

Length constraints: 1-255 chars.

`type`

The type of data. For more information about the supported data type, see Common data types in the *Amazon Glue Developer Guide*.

Length constraints: 1-131072 characters.

Amazon IoT Analytics supports all data types listed on the Data Types in Amazon Athena page, except for `DECIMAL(`*`precision`*`, `*`scale`*`)` - *`precision`*.

Create a data store (console)

The following procedure shows you how to create a data store that saves data in Parquet format.

To create a data store

1. Sign in to the https://console.amazonaws.cn/iotanalytics/.
2. In the navigation pane, choose **Data stores**.
3. On the **Data stores** page, choose **Create data store**.

4. On the **Specify data store details** page, enter basic information about your data store.

   a. For **Data store ID**, enter a unique data store ID. You can't change this ID after you create it.

   b. (Optional) For **Tags**, choose **Add new tag** to add one or more custom tags (key-value pairs) to your data store. Tags can help you identify your resources that you create for Amazon IoT Analytics.

   c. Choose **Next**.

5. On the **Configure storage type** page, specify how to store your data.

   a. For **Storage type**, choose **Service managed storage**.

   b. For **Configure how long you want to keep your processed data**, choose **Indefinitely**.

   c. Choose **Next**.

6. On the **Configure data format** page, define the structure and format of your data records.

   a. For **Classification**, choose **Parquet**. You can't change this format after you create the data store.

   b. For **Inference source**, choose **JSON string** for your data store.

   c. For **String**, enter your schema in JSON format, such as the following example.

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

   d. Choose **Infer schema**.

   e. Under **Configure Parquet schema**, confirm that the format matches your JSON example. If the format doesn't match, update the Parquet schema manually.

      - If you want your schema to show more columns, choose **Add new column**, enter a column name, and then choose the data type.

        > **Note**
        >
        > By default, you can have 100 columns for your schema. For more information, see Amazon IoT Analytics quotas.

- You can change the data type for an existing column. For more information about the supported data types, see Common data types in the *Amazon Glue Developer Guide*.

  > **Note**
  >
  > After you create your data store, you can't change the data type for an existing column.

- To remove an existing column, choose **Remove column**.

f. Choose **Next**.

7. (Optional) Amazon IoT Analytics supports custom partitions in your data store so you can query on pruned data to improve latency. For more information about supported custom partitions, see Custom partitions.

   Choose **Next**.

8. On the **Review and create** page, review your choices, and then choose **Create data store**.

   > **Important**
   >
   > You can't change the data store ID, file format, or the data type for a column after you create the data store.

9. Verify that your new data store appears on the **Data stores** page.

Custom partitions

Amazon IoT Analytics supports data partitioning so you can organize the data in your data store. When you use data partitioning to organize data, you can query on pruned data. This decreases the amount of data scanned per query and improves latency.

You can partition your data according to message data attributes or attributes added through pipeline activities.

To get started, enable data partitioning in a data store. Specify one or more data partition dimensions and connect your partitioned data store to an Amazon IoT Analytics pipeline. Then, write queries that leverage the `WHERE` clause to optimize performance.

Create a data store (console)

The following procedure shows you how to create a data store with a custom partition.

To create a data store

1. Sign in to the Amazon IoT Analytics console.
2. In the navigation pane, choose **Data stores**.
3. On the **Data stores** page, choose **Create data store**.
4. On the **Specify data store details** page, enter basic information about your data store.

    a. For **Data store ID**, enter a unique data store ID. You can't change this ID after you create it.

    b. (Optional) For **Tags**, choose **Add new tag** to add one or more custom tags (key-value pairs) to your data store. Tags can help you identify resources that you create for Amazon IoT Analytics.

    c. Choose **Next**.
5. On the **Configure storage type** page, specify how to store your data.

    a. For **Storage type**, choose **Service managed storage**.

    b. For **Configure how long you want to keep your processed data**, choose **Indefinitely**.

    c. Choose **Next**.
6. On the **Configure data format** page, define the structure and format of your data records.

    a. For your data store data format **Classification**, choose **JSON** or **Parquet**. For more information about Amazon IoT Analytics supported file types, see File formats.

    > **Note**
    >
    > You can't change this format after you create the data store.

    b. Choose **Next**.
7. Create custom partitions for this data store.

    a. For **Add data partitions**, select **Enable**.

    b. For **Data partition source**, specify basic information about the source of your partition.

Choose **Sample source**, and select the Amazon IoT Analytics channel that collects messages for this data store.

c. For **Message sample attributes**, select the message attributes you want to use to partition your data store. Then, add your selections as attribute partition dimensions or timestamp partition dimensions under **Actions**.

> **Note**
>
> You can add only one timestamp partition to your data store.

d. For **Custom data store partition dimensions**, define basic information about your partition dimensions. Each message sample attribute you selected in the previous step will become the dimensions of your partition. Customize each dimension with these options:

- **Partition type** - Specify if this partition dimension is an **Attribute** or a **Timestamp** partition type.
- **Attribute name** and **Dimension name** - By default, Amazon IoT Analytics will use the name of the message sample attribute you selected as an identifier for your attribute partition dimension. Edit the attribute name to customize the name of your partition dimension. You can use the dimension name in the `WHERE` clause to optimize query performance.
  - The name of any partition attribute dimension is prefixed with `__partition_`.
  - For timestamp partition types, Amazon IoT Analytics creates the following four dimensions with names `__year`, `__month`, `__day`, `__hour`.
- **Ordering** - Rearrange your partition dimensions to improve the latency for your queries.

For **Timestamp format**, specify the format of your timestamp partition by matching the ingested timestamp from your message data. You can choose one of Amazon IoT Analytics listed format options, or specify one that matches the format of your data. Learn more about specifying date time formatters.

To add a new dimension that isn't a message attribute, choose **Add new partitions**.

e. Choose **Next**.

8. On the **Review and create** page, review your choices, and then choose **Create data store**.

> **Important**
>
> - You can't change the data store ID after you create the data store.
> - To edit existing partitions, you must create another data store and reprocess the data through a pipeline.

9. Verify that your new data store appears on the **Data stores** page.

Creating a pipeline

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

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.

Run the following command to create a pipeline.

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

The `mypipeline.json` file contains the following content.

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

Run the following command to list your existing pipelines.

```
aws iotanalytics list-pipelines
```

Run the following command to view the configuration of an individual pipeline.

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

Ingesting data to Amazon 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 Amazon IoT Analytics. Here we show two methods of getting data into Amazon IoT Analytics. You can send a message using the Amazon IoT message broker or use the Amazon IoT Analytics `BatchPutMessage` API.

Topics

- Using the Amazon IoT message broker
- Using the BatchPutMessage API

Using the Amazon IoT message broker

To use the Amazon IoT message broker, you create a rule using the Amazon IoT rules engine. The rule routes messages with a specific topic into Amazon IoT Analytics. But first, this rule requires you to create a role which grants the required permissions.

Creating an IAM role

To have Amazon IoT messages routed into an Amazon 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 Amazon IoT Analytics channel.

Run the following command to create the role.

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

The contents of the `arpd.json` file should look like the following.

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

Then, attach a policy document to the role.

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

The contents of the `pd.json` file should look like the following.

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

Creating a Amazon IoT rule

Create an Amazon 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 the following.

```
{
    "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.

Sending MQTT messages to Amazon 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 Amazon IoT Analytics to the data store ready to be queried. To test this, you can use the Amazon IoT console to send a message.

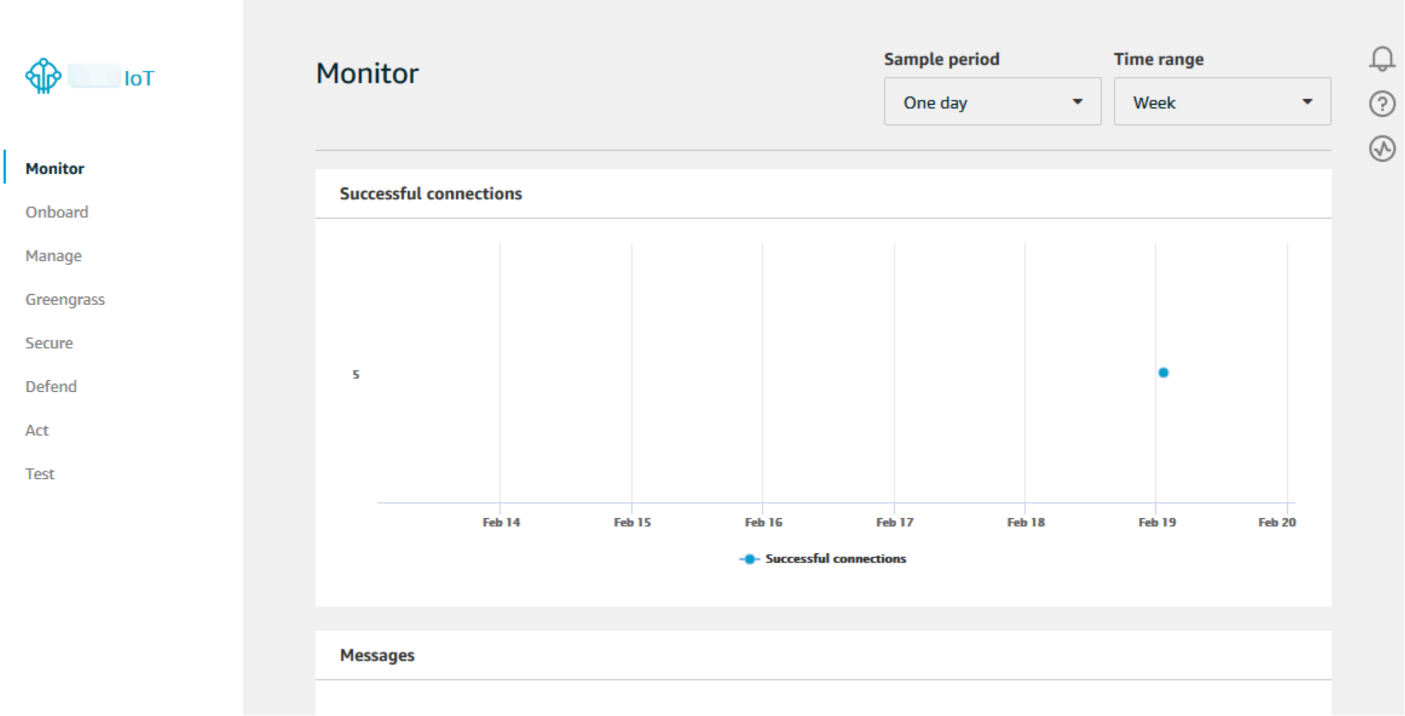
Note

The field names of message payloads (data) that you send to Amazon 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.

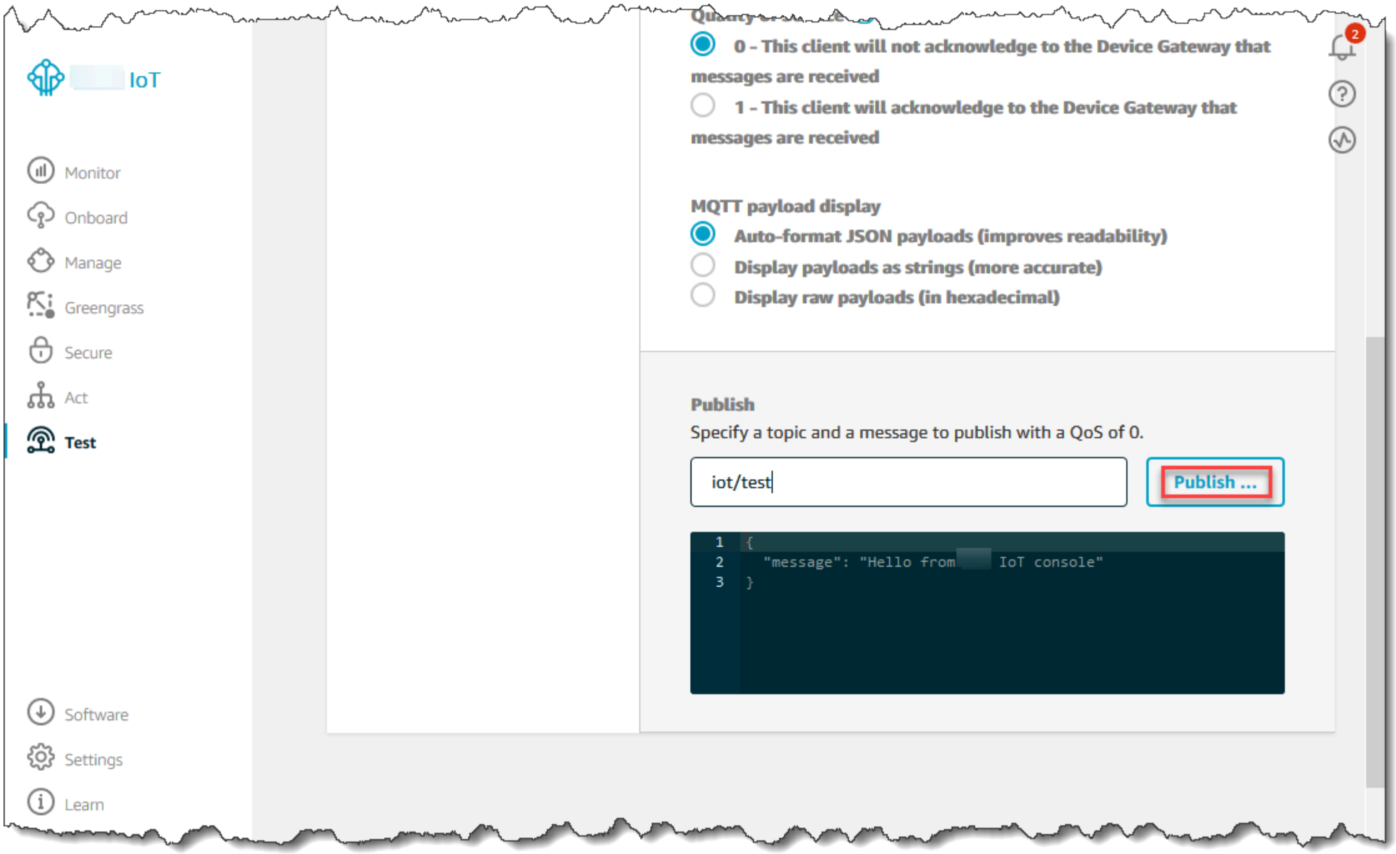
1. In the Amazon IoT console, in the left navigation pane, choose **Test**.



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 the IoT console"
}
```

3. Choose **Publish to topic**.



● 0 - This client will not acknowledge to the Device Gateway that messages are received

○ 1 - This client will acknowledge to the Device Gateway that messages are received

MQTT payload display

● Auto-format JSON payloads (improves readability)

○ Display payloads as strings (more accurate)

○ Display raw payloads (in hexadecimal)

Publish

Specify a topic and a message to publish with a QoS of 0.

iot/test

Publish ...

```
{
  "message": "Hello from [redacted] IoT console"
}
```

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

Using the BatchPutMessage API

Another way to get message data into Amazon IoT Analytics is to use the `BatchPutMessage` API command. This method does not require that you set up an Amazon 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 Amazon SDK or is capable of using the Amazon CLI to call `BatchPutMessage`.

1. Create a file `messages.json` that contains the messages to be sent (in this example only one message is sent).

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

2. Run the `batch-put-message` command.

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

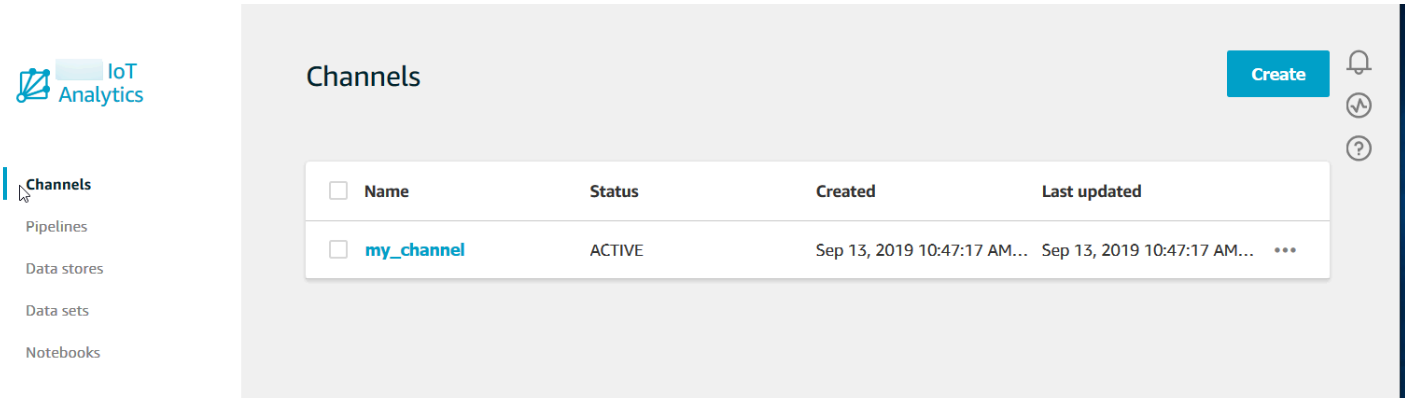
If there are no errors, you see the following output.

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

Monitoring the ingested data

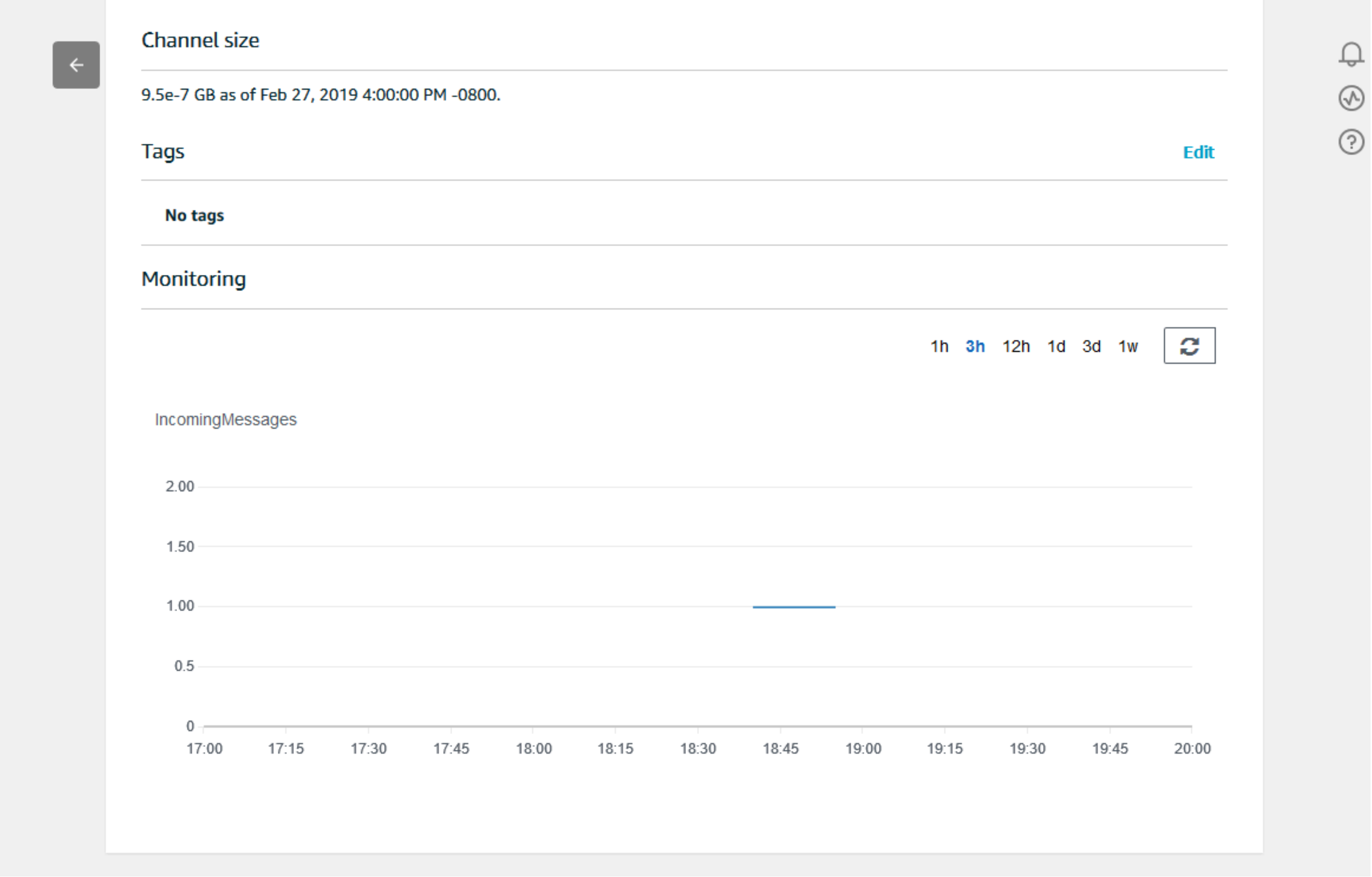
You can check that the messages you sent are being ingested into your channel by using the Amazon IoT Analytics console.

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



Name	Status	Created	Last updated
my_channel	ACTIVE	Sep 13, 2019 10:47:17 AM...	Sep 13, 2019 10:47:17 AM...

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.



Channel size

9.5e-7 GB as of Feb 27, 2019 4:00:00 PM -0800.

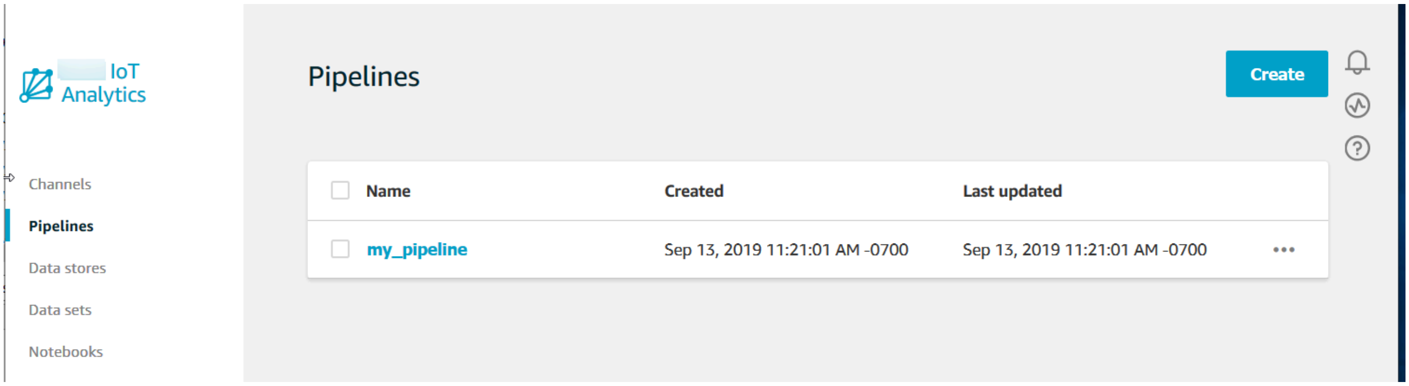
Tags

No tags

Monitoring

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

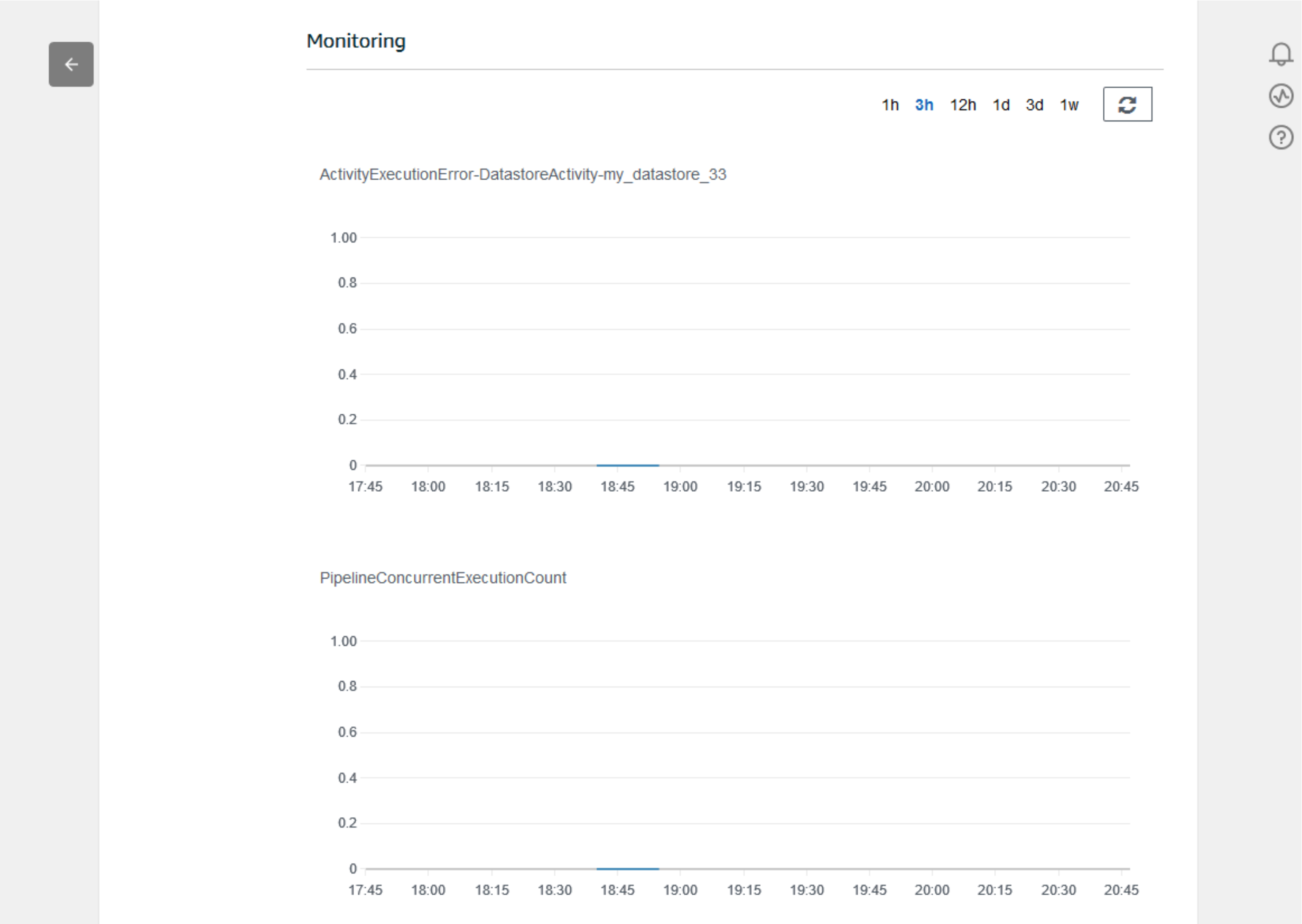


Pipelines

□	Name	Created	Last updated
□	my_pipeline	Sep 13, 2019 11:21:01 AM -0700	Sep 13, 2019 11:21:01 AM -0700

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.



Creating a dataset

You retrieve data from a data store by creating a SQL dataset or a container dataset. Amazon IoT Analytics can query the data 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 dataset.

Topics

- Querying data
- Accessing the queried data

Querying data

To query the data, you create a dataset. A dataset 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.

Run the following command to create a dataset.

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

Where the `mydataset.json` file contains the following content.

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

Run the following command to create the dataset content by executing the query.

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

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

Accessing the queried data

The result of the query is your dataset 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.

Run the following `get-dataset-content` command.

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

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

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

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

`dataURI` 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.

Exploring Amazon IoT Analytics data

You have several options for storing, analyzing and visualizing your Amazon IoT Analytics data.

Topics on this page:

- Amazon S3
- Amazon IoT Events
- Jupyter Notebook

Amazon S3

You can send dataset contents to an Amazon Simple Storage Service (Amazon 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.

Amazon IoT Events

You can send dataset contents as an input to Amazon 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 dataset using CreateDataset and specify an Amazon IoT Events input in the field `contentDeliveryRules :: destination :: iotEventsDestinationConfiguration :: inputName`. You must also specify the `roleArn` of a role which grants Amazon IoT Analytics permission to execute "iotevents:BatchPutMessage". Whenever the dataset's contents are created, Amazon IoT Analytics will send each dataset content entry as a message to the specified Amazon IoT Events input. For example, if your dataset 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 Amazon 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 Amazon IoT Events input that recognized the fields you are interested in (one or more of `what`, `who`, `dt`) and to create an Amazon IoT Events detector model that uses these input fields in events to trigger actions or set internal variables.

Jupyter Notebook

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

Keeping multiple versions of datasets

You can choose how many versions of your dataset contents to retain, and for how long, by specifying values for the dataset `retentionPeriod and versioningConfiguration` fields when invoking the CreateDataset and UpdateDataset 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.

Message payload syntax

The field names of message payloads (data) that you send to Amazon 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.

Working with Amazon IoT SiteWise data

Amazon IoT SiteWise is a managed service that you can use to collect, model, analyze, and visualize data from industrial equipment at scale. The service provides an asset modeling framework for building representations of your industrial devices, processes, and facilities.

With Amazon IoT SiteWise asset models, you can define what industrial equipment data to consume and how to process your data into complex metrics. You can configure asset models to collect and process data in the Amazon Cloud. For more information, see the Amazon IoT SiteWise User Guide.

Amazon IoT Analytics integrates with Amazon IoT SiteWise so you can run and schedule SQL queries on Amazon IoT SiteWise data. To start querying your Amazon IoT SiteWise data, create a data store by following the procedures in Configure storage settings in the *Amazon IoT SiteWise User Guide*. Then, follow the steps in Create a dataset with Amazon IoT SiteWise data (Console) or in Create a dataset with Amazon IoT SiteWise data (Amazon CLI) to create an Amazon IoT Analytics dataset and run a SQL query on your industrial data.

Topics

- Create an Amazon IoT Analytics dataset with Amazon IoT SiteWise data
- Access dataset contents
- Tutorial: Query Amazon IoT SiteWise data in Amazon IoT Analytics

Create an Amazon IoT Analytics dataset with Amazon IoT SiteWise data

An Amazon IoT Analytics dataset contains SQL statements and expressions that you use to query data in your data store along with an optional schedule that repeats the query at a day and time that you specify. You can use expressions similar to Amazon CloudWatch schedule expressions to create the optional schedules.

> **Note**
>
> A dataset is typically a collection of data that might or might not be organized in tabular form. In contrast, Amazon IoT Analytics creates your dataset by applying a SQL query to data in your data store.

Follow these steps to get started with creating a dataset for your Amazon IoT SiteWise data.

Topics

- Create a dataset with Amazon IoT SiteWise data (Console)
- Create a dataset with Amazon IoT SiteWise data (Amazon CLI)

Create a dataset with Amazon IoT SiteWise data (Console)

Use these steps to create a dataset in the Amazon IoT Analytics console for your Amazon IoT SiteWise data.

To create a dataset

1. In the https://console.amazonaws.cn/iotanalytics/, on the **left navigation pane**, choose **Datasets**.
2. On the **Create dataset** page, choose **Create SQL**.
3. On the **Specify dataset details** page, specify the details of your dataset.
   a. Enter a name for your dataset.
   b. For **Data store source**, choose the unique ID that identifies your Amazon IoT SiteWise data store.
   c. (Optional) For **Tags**, add one or more custom tags (key-value pairs) to your dataset.
4. Use SQL expressions to query your data and answer analytical questions.
   a. In the **Author query** field, enter a SQL query that uses a wildcard to display up to five rows of data.

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

   For more information about supported SQL functionality in Amazon IoT Analytics, see SQL expressions in Amazon IoT Analytics. Or, see Tutorial: Query Amazon IoT SiteWise data in Amazon IoT Analytics for examples of statistical queries that can provide insight to your data.

   b. You can choose **Test query** to validate that your input is correct, and to display the results in a table following the query.

   > **Note**
   >
   > Because Amazon Athena limits the maximum number of running queries, you should limit your SQL query to a reasonable size so that it does not run for an extended period.

5. (Optional) When you create dataset contents using data from a specified time frame, some data might not arrive in time for processing. To allow for a delay, you can specify an offset, or delta. For more information, see Getting late data notifications through Amazon CloudWatch Events.

   After configuring a data selection filter on the **Configure data selection filter** page, choose **Next**.

6. (Optional) On the **Set query schedule page**, you can schedule this query to run regularly to refresh the dataset. Dataset schedules can be created and edited at any time.

   > **Note**
   >
   > Data from Amazon IoT SiteWise ingests into Amazon IoT Analytics every six hours. We recommend selecting a frequency that is six hours or more.

   Choose and option for **Frequency** and then choose **Next**.

7. Amazon IoT Analytics will create versions of this dataset content and store your analytics results for the specified period. We recommend 90 days, however you can opt to set your custom retention policy. You may also limit the number of stored versions of your dataset content.

   After selecting your options on the **Configure the results of your dataset** page, choose **Next**.

8. (Optional) You can configure the delivery rules of your dataset results to a specific destination, such as Amazon IoT Events.

   After selecting your options on the **Configure dataset content delivery rules** page, choose **Next**.

9. Review your choices and then choose **Create dataset**.
10. Verify that your new dataset appears on the **Datasets** page.

Create a dataset with Amazon IoT SiteWise data (Amazon CLI)

Run the following Amazon CLI commands to get started querying your Amazon IoT SiteWise data.

The examples shown here use the Amazon Command Line Interface (Amazon CLI). For more information on the Amazon CLI, see the Amazon Command Line Interface User Guide. For more information about the CLI commands available for Amazon IoT Analytics, see iotanalytics in the *Amazon Command Line Interface Reference*.

To create a dataset

1. Run the following `create-dataset` command to create a dataset.

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

   Where the `my_dataset.json` file contains the following content.

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

   For more information about supported SQL functionality in Amazon IoT Analytics, see SQL expressions in Amazon IoT Analytics. Or, see Tutorial: Query Amazon IoT SiteWise data in Amazon IoT Analytics for examples of statistical queries that can provide insight to your data.

2. Run the following `create-dataset-content` command to create your dataset content by running your query.

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

Access dataset contents

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

Topics

- Access dataset content in Amazon IoT Analytics (Console)
- Access dataset content in Amazon IoT Analytics (Amazon CLI)

Access dataset content in Amazon IoT Analytics (Console)

If your dataset contains any data, you can preview and download your SQL query results in Amazon IoT Analytics console.

To access your Amazon IoT Analytics dataset results

1. In the console, on the **Datasets** page, choose the name of the dataset you want to access.
2. On the dataset summary page, choose the **Content** tab.
3. In the **Dataset contents** table, choose the name of the the query you wish to preview the results or download a csv file of the results.

Access dataset content in Amazon IoT Analytics (Amazon CLI)

If your dataset contains any data, you can preview and download your SQL query results.

The examples shown here use the Amazon Command Line Interface (Amazon CLI). For more information on the Amazon CLI, see the Amazon Command Line Interface User Guide. For more information about the CLI commands available for Amazon IoT Analytics, see iotanalytics in the *Amazon Command Line Interface Reference*.

To access your Amazon IoT Analytics dataset results (Amazon CLI)

1. Run the following `get-dataset-content` command to view the result of your query.

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

2. If your dataset contains any data, then the output from `get-dataset-content`, has `"state": "SUCCEEDED"` in the `status` field, such as in the following example.

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

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

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

3. The output from `get-dataset-content` includes a `dataURI`, which is a signed URL to the output results. It is valid for a short period of time (a few hours). Visit the `dataURI` URL to access your SQL query results.

   > **Note**
   >
   > 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.

Tutorial: Query Amazon IoT SiteWise data in Amazon IoT Analytics

This tutorial demonstrates how to query Amazon IoT SiteWise data in Amazon IoT Analytics. The tutorial uses data from a demo in Amazon IoT SiteWise that provides a sample set of data for a wind farm.

> **Important**
>
> You will be charged for the resources that this demo creates and consumes.

Topics

- Cleaning up your tutorial resources

Prerequisites

For this tutorial, you need the following resources:

- You must have an Amazon account to get started with Amazon IoT SiteWise and Amazon IoT Analytics. If you don't have one, follow the procedures in To create an Amazon account.
- A development computer running Windows, macOS, Linux, or Unix to access the Amazon Web Services Management Console. For more information, see Getting Started with the Amazon Web Services Management Console.
- Amazon IoT SiteWise data that defines Amazon IoT SiteWise models and assets and streams data that represents data from wind farm equipment. To create your data, follow the steps in Creating the Amazon IoT SiteWise demo in the *Amazon IoT SiteWise User Guide*.
- Your Amazon IoT SiteWise demo wind farm equipment data in an existing data store that you manage. For more information about how to create a data store for your Amazon IoT SiteWise data, see Configure storage settings in the *Amazon IoT SiteWise User Guide*.

> **Note**
>
> Your Amazon IoT SiteWise metadata appears in your Amazon IoT SiteWise data store soon after creation; however, it can take up to six hours for your raw data to appear. In the meantime, you can create an Amazon IoT Analytics dataset and run queries on your metadata.

Next step

Load and verify data

Load and verify data

The data you query in this tutorial is a sample set of Amazon IoT SiteWise data that models wind engine turbines in a wind farm.

> **Note**
>
> You will query three tables in your data store throughout this tutorial:

- raw - Contains raw, unprocessed data for each asset.
- asset_metadata - Contains general information about each asset.
- asset_hierarchy_metadata - Contains information about the relationships between assets.

To run the SQL queries in this tutorial

1. Follow the steps in Create a dataset with Amazon IoT SiteWise data (Console) or Create a dataset with Amazon IoT SiteWise data (Amazon CLI) to create an Amazon IoT Analytics dataset for your Amazon IoT SiteWise data.
2. To update your dataset query throughout this tutorial, do the following.

   a. In the Amazon IoT Analytics console, on the **Datasets** page, choose the name of the dataset you created on the previous page.

   b. On the dataset summary page, choose **Edit** to edit your SQL query.

   c. To display the results in a table following the query, choose **Test query**.

   Alternatively, you can run the following update-dataset command to modify the SQL query with the Amazon CLI.

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

   Contents of update-query.json:

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

3. In the Amazon IoT Analytics console or with the Amazon CLI, run the following query on your data to verify that your `asset_metadata` table loaded successfully.

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

   Similarly, you can verify that your `asset_hierarchy_metadata` and `raw` tables aren't empty.

Next Step

Data exploration

Data exploration

After your Amazon IoT SiteWise data is created and loaded into a data store, you can create an Amazon IoT Analytics dataset and run SQL queries in Amazon IoT Analytics to discover insights about your assets. The following queries demonstrate how you can explore your data before running statistical queries.

To explore your data with SQL queries

1. View a sample of columns and values in each table, such as in the raw table.

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

2. Use `SELECT DISTINCT` to query your `asset_metadata` table and list the (unique) names of your Amazon IoT SiteWise assets.

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

3. To list information about properties for a particular Amazon IoT SiteWise asset, use the `WHERE` clause.

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

4. With Amazon IoT Analytics, you can join data from two or more tables in your data store, such as in the following example.

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

   To view all relationships between your assets, use the `JOIN` functionality in the following query.

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

Next step

Run statistical queries

Run statistical queries

Now that you've explored your Amazon IoT SiteWise data, you can run statistical queries that provide valuable insights to your industrial equipment. The following queries demonstrate some of the information that you can retrieve.

To run statistical queries on Amazon IoT SiteWise demo wind farm data

1. Run the following SQL command to find the latest values of all properties with numeric values for a particular asset (Demo Turbine Asset 4).

   ```
   SELECT assetName,
   ```

```
    assetPropertyName,
    assetPropertyUnit,
    max_by(value, timeInSeconds) AS Latest
FROM (
    SELECT *,
        CASE assetPropertyDataType
        WHEN 'DOUBLE' THEN
        cast(doubleValue AS varchar)
        WHEN 'INTEGER' THEN
        cast(integerValue AS varchar)
        WHEN 'STRING' THEN
        stringValue
        WHEN 'BOOLEAN' THEN
        cast(booleanValue AS varchar)
        ELSE NULL
        END AS value
    FROM my_iotsitewise_datastore.asset_metadata AS asset_metadata
    JOIN my_iotsitewise_datastore.raw AS raw
        ON raw.seriesId = asset_metadata.timeSeriesId
    WHERE startYear=2021
        AND startMonth=7
        AND startDay=8
        AND assetName='Demo Turbine Asset 4'
)
GROUP BY assetName, assetPropertyName, assetPropertyUnit
```

2. Join both metadata tables and your raw table to identify the maximum wind speed properties for all assets, in addition to their parent assets.

```
SELECT child_assets_data_set.parentAssetId,
        child_assets_data_set.childAssetId,
        asset_metadata.assetPropertyId,
        asset_metadata.assetPropertyName,
        asset_metadata.timeSeriesId,
        raw_data_set.max_speed
FROM (
   SELECT sourceAssetId AS parentAssetId,
        targetAssetId AS childAssetId
    FROM my_iotsitewise_datastore.asset_hierarchy_metadata
    WHERE associationType = 'CHILD'
)
AS child_assets_data_set
JOIN mls_demo.asset_metadata AS asset_metadata
```

```
    ON asset_metadata.assetId = child_assets_data_set.childAssetId
JOIN (
    SELECT seriesId, MAX(doubleValue) AS max_speed
    FROM my_iotsitewise_datastore.raw
    GROUP BY seriesId
)
AS raw_data_set
ON raw_data_set.seriesId = asset_metadata.timeseriesid
WHERE assetPropertyName = 'Wind Speed'
ORDER BY max_speed DESC
```

3. To find the average value of a particular property (Wind Speed) for an asset (Demo Turbine Asset 2), run the following SQL command. You must replace `my_bucket_id` with the ID of your bucket.

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

Next step

Cleaning up your tutorial resources

Cleaning up your tutorial resources

After you complete the tutorial, clean up your resources to avoid incurring charges.

To delete your Amazon IoT SiteWise demo

The Amazon IoT SiteWise demo deletes itself after a week. If you're done using the demo resources, you can delete the demo earlier. To delete the demo manually, use the following steps.

1. Navigate to the Amazon CloudFormation console.
2. Choose **IoTSiteWiseDemoAssets** from the list of **Stacks**.
3. Choose **Delete**. When you delete the stack, all of the resources created for the demo are deleted.

4. In the confirmation dialog, enter **Delete**.

   The stack takes around 15 minutes to delete. If the demo fails to delete, choose **Delete** in the upper-right corner again. If the demo fails to delete again, follow the steps in the Amazon CloudFormation console to skip the resources that failed to delete, and try again.

To delete your data store

- To delete your managed data store, run the CLI command `delete-datastore`, such as in the following example.

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

To delete your Amazon IoT Analytics dataset

- To delete your dataset, run the CLI command `delete-dataset`, such as in the following example. You don't have to delete the content of the dataset before you perform this operation.

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

> **Note**
>
> This command produces no output.

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.

You can use the RunPipelineActivity operation 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 demonstrates how it is used.

Channel activity

The first activity in a pipeline must be the `channel` activity that 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"
    }
}
```

Amazon Lambda activity

You can use a **`lambda` activity** to perform complex processing on messages. For example, you can enrich messages with data from the output of external API operations, or filter for messages based on logic from Amazon DynamoDB. However, you can't use this pipeline activity to add additional messages, or remove existing messages, before entering a data store.

The Amazon Lambda function used in a **`lambda` activity** must receive and return an array of JSON objects. For an example, see the section called “Lambda function example 1”.

To grant Amazon IoT Analytics permission to invoke your Lambda function, you must add a policy. For example, run the following CLI command and replace *`exampleFunctionName`* with the name of your Lambda function, replace *`123456789012`* with your Amazon Account ID, and use the Amazon Resource Name (ARN) of the pipeline that invokes the given Lambda function.

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

The command returns the following:

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

For more information, see Using resource-based policies for Amazon Lambda in the *Amazon Lambda Developer Guide*.

Lambda function example 1

In this example, the Lambda function adds information based on data in the original message. A device publishes a message with a payload similar to the following example.

```
{
  "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 device has 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 that 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's stored. For example, you can use `addAttributes` to normalize data coming from different generations of device firmware.

Consider the following input message.

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

The `addAttributes` activity looks like the following.

```
{
    "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 value 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 example.

```
{
    "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 Amazon 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 example.

```
{
    "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 the following example.

```
{
    "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 Amazon IoT Device Shadow 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 the following example.

```
{
    "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 the following.

```
{
    "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)
- acos(Decimal)
- asin(Decimal)
- atan(Decimal)
- atan2(Decimal, Decimal)
- ceil(Decimal)
- cos(Decimal)
- cosh(Decimal)
- exp(Decimal)
- ln(Decimal)
- log(Decimal)
- mod(Decimal, Decimal)
- power(Decimal, Decimal)
- round(Decimal)
- sign(Decimal)
- sin(Decimal)

- sinh(Decimal)
- sqrt(Decimal)
- tan(Decimal)
- tanh(Decimal)
- trunc(Decimal, Integer)

abs(Decimal)

Returns the absolute value of a number.

Examples: `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.

Examples: `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` (with double precision) 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.

Examples: `asin(0)` = 0.0

Argument type	Result
`Int`	`Decimal` (with double precision), the inverse sine of the argument. Imaginary results are returned as `Undefined` .

Argument type	Result
`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` .
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.

Examples: `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` .

Argument type	Result
`Boolean`	`Undefined` .
`String`	`Decimal` (with double precision), 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 `Decimal` arguments are rounded to double precision before function application.

Examples: `atan(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
`Int` / `Decimal` / `String`	`Int` / `Decimal` / `String`	`Decimal`, the inverse tangent of the point described. If a string cannot be converted, the result is `Undefined` .

Argument type	Argument type	Result
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 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.

Examples: `cos(0)` = 1

Argument type	Result
`Int`	`Decimal` (with double precision), the cosine of the argument. Imaginary results are returned as `Undefined` .

Argument type	Result
`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 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.

Examples: `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` .

Argument type	Result
`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.

Examples: `exp(1)` = 1

Argument type	Result
`Int`	`Decimal` (with double precision), e^argument.
`Decimal`	`Decimal` (with double precision), e^argument
`String`	`Decimal` (with double precision), e^argumen t. If the `String` cannot be converted to a `Decimal`, the result if `Undefined` .
Other Value	`Undefined` .

ln(Decimal)

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

Examples: `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.

Examples: `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` .
`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 of the second argument. You can also use `%` as an infix operator for the same modulo functionality.

Examples: `mod(8, 3)` = 2

Left operand	Right operand	Output
`Int`	`Int`	`Int`, the first argument modulo of the second argument.

Left operand	Right operand	Output
Int / Decimal	Int / Decimal	Decimal, the first argument modulo of the second argument.
String / Int / Decimal	String / Int / Decimal	If all strings convert to Decimals, the result if 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.

Examples: 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 power. Any strings are converted to Decimals. If any String fails to be converted to Decimal, the result is Undefined .

Argument type 1	Argument type 2	Output
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.

Examples:

`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.
`String`	`Int`, the sign of the `Decimal` value. The string if 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.

Examples: `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` .

Argument type	Result
`String`	`Decimal`, 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.

Examples: `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` .
`String`	`Decimal`, the hyperbolic sine of the argument. If the string cannot be converted to a `Decimal`, the result is `Undefined` .
`Array`	`Undefined` .
`Object`	`Undefined` .

Argument type	Result
`Null`	`Undefined` .
`Undefined`	`Undefined` .

sqrt(Decimal)

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

Examples: `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.

Examples: `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.
`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.

Examples: `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` .

Argument type	Result
`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, Integer)

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 zeros are stripped from the result.

Examples:

`trunc(2.3, 0)` = 2

`trunc(2.3123, 2)` = 2.31

`trunc(2.888, 2)` = 2.88

`trunc(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`,

Argument type 1	Argument type 2	Result
		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

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 `maths.json` file, 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` file, which contains the example payloads that are used to test the pipeline activity.

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

3. Call the `RunPipelineActivities` operation from the command line.

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

This produces the following results.

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

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 messages

Amazon IoT Analytics enables you to reprocess channel data. This can be useful in the following cases:

- 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 want to include data that was ingested before you made changes to the customer managed storage options, permissions for channels, or data store.

Parameters

When you reprocess channel messages through the pipeline with Amazon IoT Analytics, you must specify the following information:

`StartPipelineReprocessing`

Starts reprocessing channel messages through the pipeline.

`ChannelMessages`

Specifies one or more sets of channel messages that you want to reprocess.

If you use the `channelMessages` object, you must not specify a value for `startTime` and `endTime`.

`s3Paths`

Specifies one or more keys that identify the Amazon Simple Storage Service (Amazon S3) objects that save your channel messages. You must use the full path for the key.

Example path: `00:00:00/1582940490000_1582940520000_123456789012_mychannel_0_2118.0.json`

Type: Array of strings

Array members constraints: 1-100 items.

Length constraints: 1-1024 characters.

`endTime`

The end time (exclusive) of the channel data that is reprocessed.

If you specify a value for the `endTime` parameter, you must not use the `channelMessages` object.

Type: Timestamp

`startTime`

The start time (inclusive) of raw message data that is reprocessed.

If you specify a value for the `startTime` parameter, you must not use the `channelMessages` object.

Type: Timestamp

`pipelineName`

The name of the pipeline on which to start reprocessing.

Type: String

Length constraints: 1-128 characters.

Reprocessing channel messages (console)

This tutorial shows you how to reprocess the channel data that is stored in the specified Amazon S3 object in the Amazon IoT Analytics console.

Before you begin, make sure that the channel messages that you want to reprocess are saved in a customer managed Amazon S3 bucket.

1. Sign in to the Amazon IoT Analytics console.
2. In the navigation pane, choose **Pipelines**.
3. Choose your target pipeline.
4. Choose **Reprocess messages** from **Actions**.
5. On the **Pipeline reprocessing** page, choose **S3 objects** for **Reprocess messages**.

   The Amazon IoT Analytics console also provides the following options:

- **All available range** - Reprocess all valid data in the channel.
- **Last 120 days** - Reprocess data that arrived in the last 120 days.
- **Last 90 days** - Reprocess data that arrived in the last 90 days.
- **Last 30 days** - Reprocess data that arrived in the last 30 days.
- **Custom range** - Reprocess data that arrived in the specified time range. You can choose any time range.

6. Enter the key of the Amazon S3 obejct that stores your channel messages.

   To find the key, do the following:

   a. Go to the Amazon S3 console.
   b. Choose the target Amazon S3 object.
   c. Under **Properties**, in the **Object overview** section, copy the key.
7. Choose **Start reprocessing**.

Reprocessing channel messages (API)

When you use the `StartPipelineReprocessing` API, 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.
- You can make up to 1000 `StartPipelineReprocessing` API requests for every 24 hours to reprocess the same channel messages through a pipeline.
- Reprocessing your raw data incurs additional costs.

For more information, see the StartPipelineReprocessing API, in *Amazon IoT Analytics API Reference*.

Canceling channel reprocessing activities

To cancel a pipeline reprocessing activity, use the CancelPipelineReprocessing API or choose **Cancel reprocessing** on the **Activities** page in the Amazon IoT Analytics console. If you cancel the reprocessing, the remaining data won't be reprocessed. You must start another reprocessing request.

Use the DescribePipeline API to check the status of the reprocessing. See the `reprocessingSummaries` field in the response.

Automating your workflow

Amazon IoT Analytics provides advanced data analysis for Amazon 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 Notebook 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 dataset content on a recurring schedule

> Schedule the automatic creation of dataset content by specifying a trigger when you call `CreateDataset` (`triggers:schedule:expression`). Data that has in a data store is used to create the dataset 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 dataset 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 dataset content when the time interval has elapsed. See the section called "Example 6 -- creating a SQL dataset with a delta window (CLI)".

Create dataset content upon completion of another dataset

> Trigger creation of new dataset content when another dataset'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 dataset's content is created. This way, you can feed your application with data from a dataset'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`)

Create dataset content upon completion of another dataset

> Trigger creation of new dataset content when another dataset'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 dataset's content is created. This way, you can feed your application with data from a dataset'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 value opens and closes. With Amazon 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 batch, you 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 Amazon 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

Amazon 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 Amazon 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. Amazon 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.

Using a Docker container

This section includes information 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. Amazon 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 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`)
4. Create a channel where the messages are sent (`iotanalytics: CreateChannel`).
5. Create a pipeline to connect the channel to the data store (`iotanalytics: CreatePipeline`).
6. Create an IAM role that grants permission to send message data to an Amazon 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 visa MQTT, it is routed to your channel. Or, you can use `iotanalytics: BatchPutMessage` to send messages directly into a channel from a device capable of using the Amazon SDK or Amazon CLI.
8. Create a SQL dataset whose creation is triggered by a time schedule (`iotanalytics: CreateDataset,` 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 dataset is created using messages received since the last time the SQL dataset was created. (What about the first time the SQL dataset is created? An estimate of when the last time the dataset would have been created is made based on the schedule and the pre-filter.)

9. Create another dataset that is triggered by the creation of the first (CreateDataset field `trigger:dataset`). For this dataset, you specify a container action (filed `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 if 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 (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 dataset 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 dataset whose latest content is used as input to the application (this is the name of the dataset you created in the previous step) (`datasetContentVersionValue:datasetName`).

With the SQL query and delta window to generate the dataset, and the container with your application, Amazon IoT Analytics creates a scheduled production dataset 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 dataset application and resume it whenever you choose to do so. When you resume your production dataset application, Amazon 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 dataset job window length) by performing a series of consecutive runs. Alternatively, you can resume your production dataset application by capturing only the newly arrived data that fits within the specified size of your delta window.

Please note the following limitations when creating or defining a dataset which is triggered by the creation of another dataset:

- Only container datasets can be triggered by SQL datasets.
- A SQL dataset can trigger at most 10 container datasets.

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

- "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 dataset which is triggered by two different SQL dataset.
- "The triggering dataset <dataset-name> cannot be triggered by a container dataset"

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

This error occurs if you attempt to define another container dataset which is triggered by a SQL dataset which already triggers 10 container datasets.

- "Exactly one trigger type should be provided"

  This error occurs is you attempt to define a dataset which is triggered by both a schedule trigger and a dataset 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 might also store its output in an Amazon S3 location. The output must be loaded with a "bucket-owner-full-control" access control list. The access list grants the Amazon 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 Amazon 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 AI 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 SageMaker AI instance with the Amazon IoT Analytics console, you are given the option to automatically create a new role which grants all privileges necessary to execute SageMaker AI 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 Amazon 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": "*"
        }
    ]
}
```

The following is an example trust policy.

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

Using the CreateDataset API via Java and the Amazon CLI

Creates a dataset. A dataset 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 dataset. The dataset can be populated manually by calling `CreateDatasetContent` or automatically according to a `trigger` you specify. For more information, see CreateDataset and `CreateDatasetContent`.

Topics

Example 1 -- creating a SQL dataset (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 dataset 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 dataset with a SQL dataset 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 continuous time intervals. Delta windows enable you to create dataset 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 dataset (CreateDataset). Usually, you'll want to create the dataset 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 dataset that automatically created new dataset 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 dataset content is created at 10:30AM, the data used (included in the dataset 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 the following.

```
{
  "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

This section includes information 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 Amazon 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 Notebook provides powerful data science tools that can perform machine learning and a range of statistical analyses. For more information, see Notebook templates. (Note that we do not currently support containerization inside JupyterLab.) You can package your Jupyter Notebook and libraries into a container that periodically runs on a new batch of data as it is received by Amazon 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.

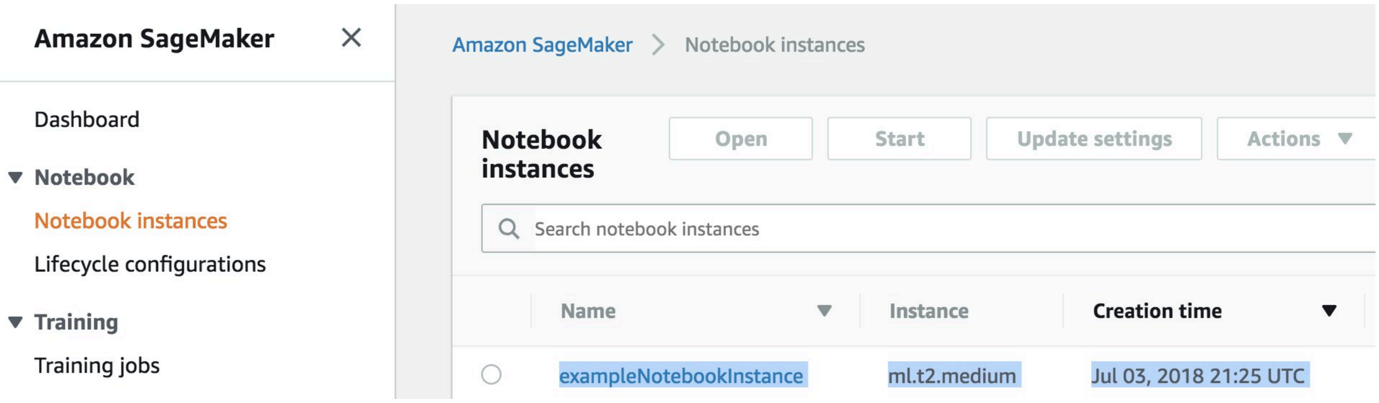
If you have created a SageMaker AI Instance using the Amazon IoT Analytics console after August 23, 2018, then the installation of the containerization extension has been done for you automatically and you can begin creating a containerized image. Otherwise, follow the steps listed in this section to enable notebook containerization on your SageMaker AI instance. In what follows, you modify your SageMaker AI Execution Role to allow you to upload the container image to Amazon EC2 and you install the containerization extension.

Enable containerization of notebook instances not created via Amazon IoT Analytics console

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

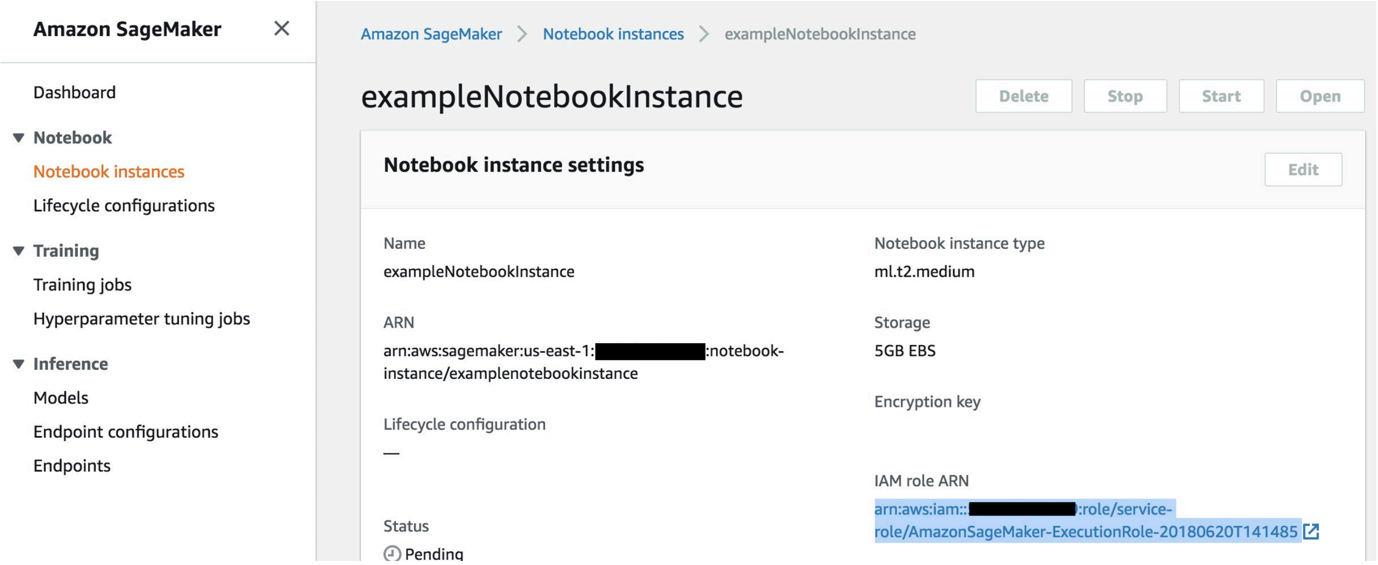
If you restart your SageMaker AI instance after enabling containerization as shown here, you won't have to re-add the IAM roles and policies, but you must re-install the extension, as shown in the final step.

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



	Name	Instance	Creation time
○	exampleNotebookInstance	ml.t2.medium	Jul 03, 2018 21:25 UTC

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



exampleNotebookInstance

Notebook instance settings

Name
exampleNotebookInstance

Notebook instance type
ml.t2.medium

ARN
arn:aws:sagemaker:us-east-1:[redacted]:notebook-instance/examplenotebookinstance

Storage
5GB EBS

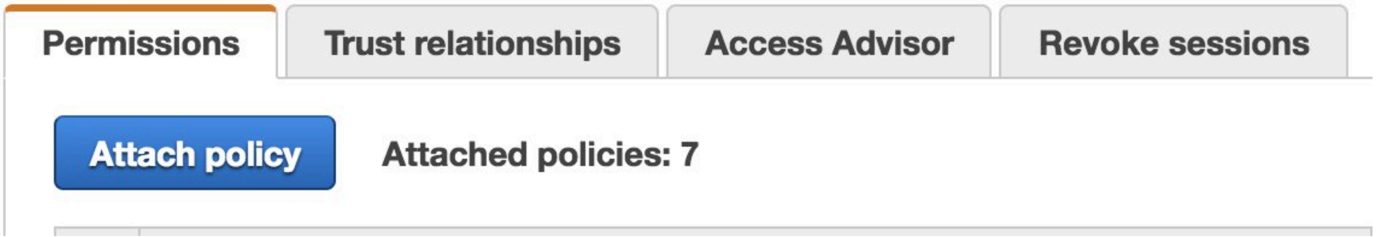
Encryption key

Lifecycle configuration
—

IAM role ARN
arn:aws:iam::[redacted]:role/service-role/AmazonSageMaker-ExecutionRole-20180620T141485

Status
Pending

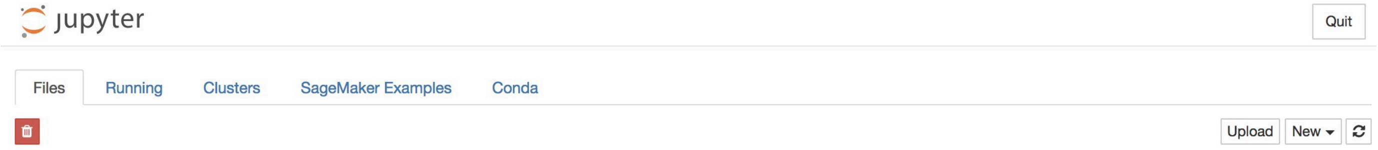
3. Choose **Attach Policy**, then define and attach the policy shown in Permissions. If the `AmazonSageMakerFullAccess` policy is not already attached, attach it as well.



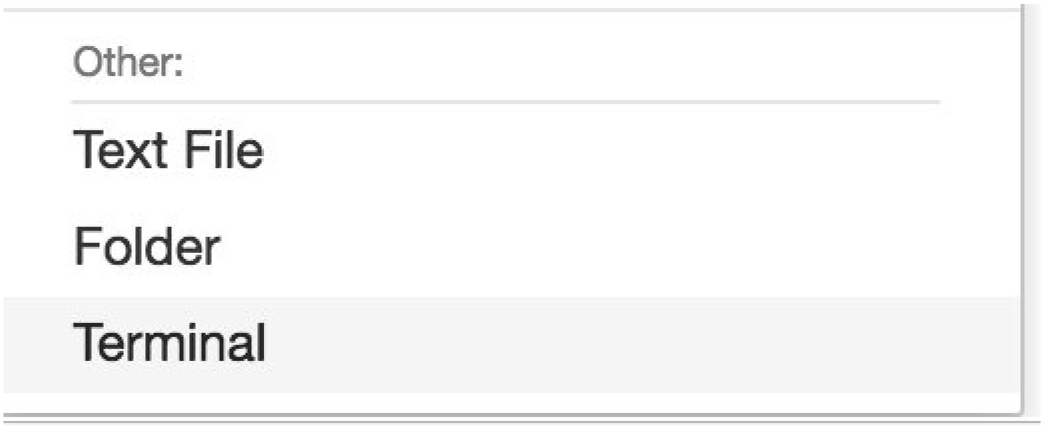
Attached policies: 7

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

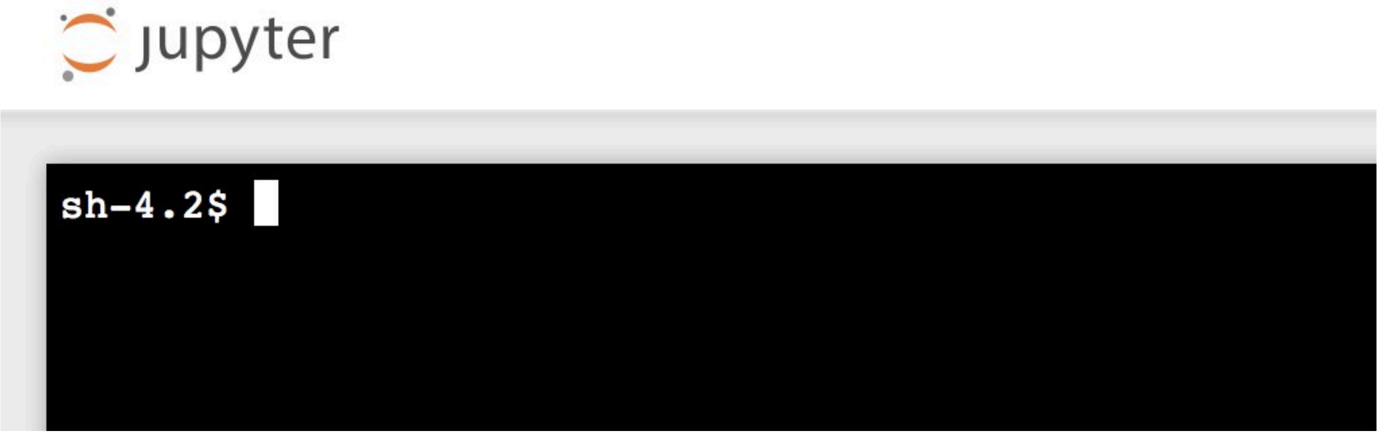
1. Inside Jupyter, choose **New**.



2. In the menu that appears, choose **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 AI instance.



```
sh-4.2$
```

```
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.

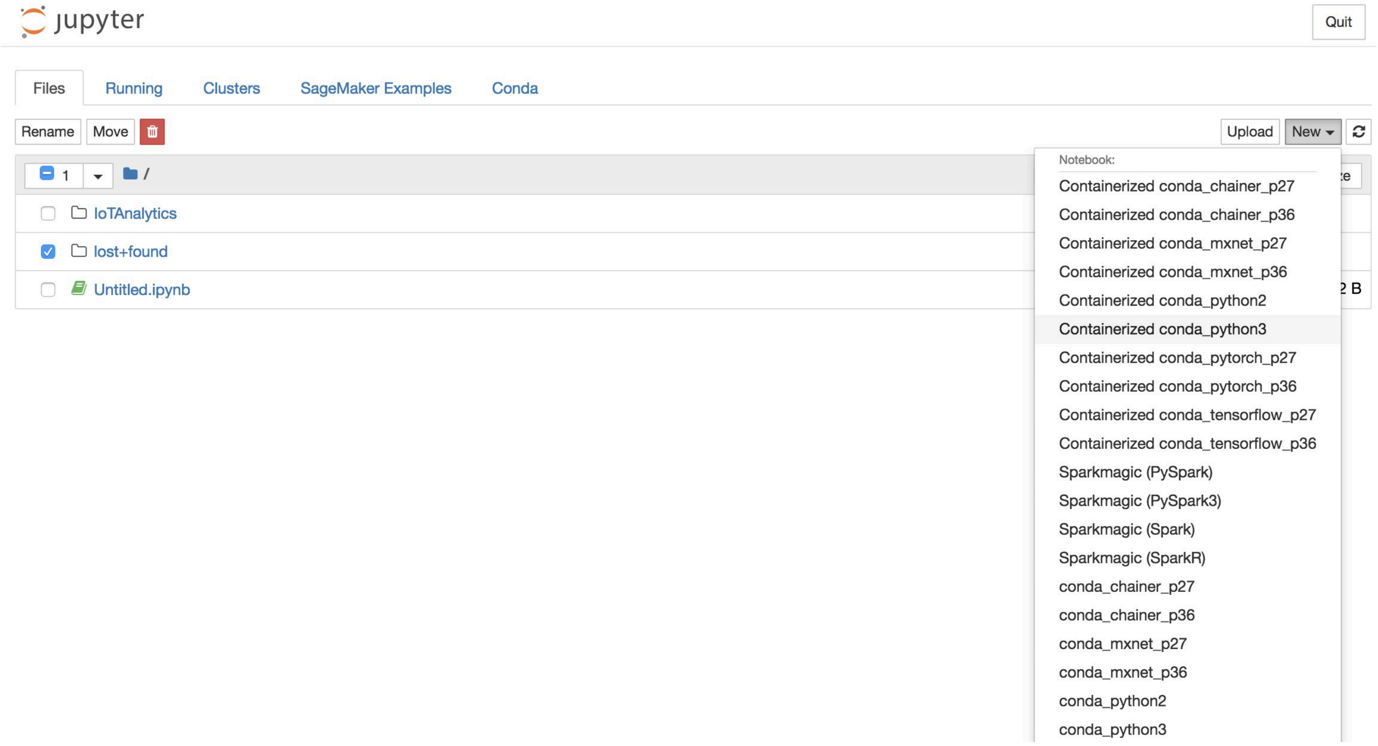
Update your notebook containerization extension

If you created your SageMaker AI Instance via the Amazon IoT Analytics console after August 23, 2018, then the containerization extension was installed automatically. You can update the extension by restarting your instance from SageMaker AI 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 Amazon IoT Analytics Console.

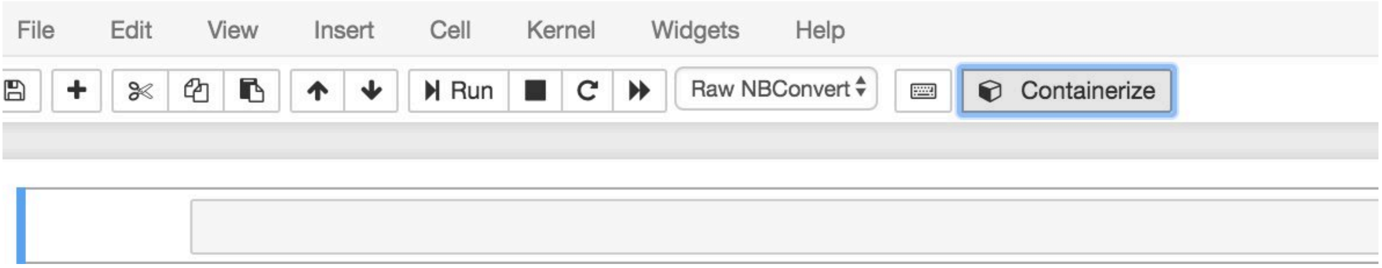
Create a containerized image

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.

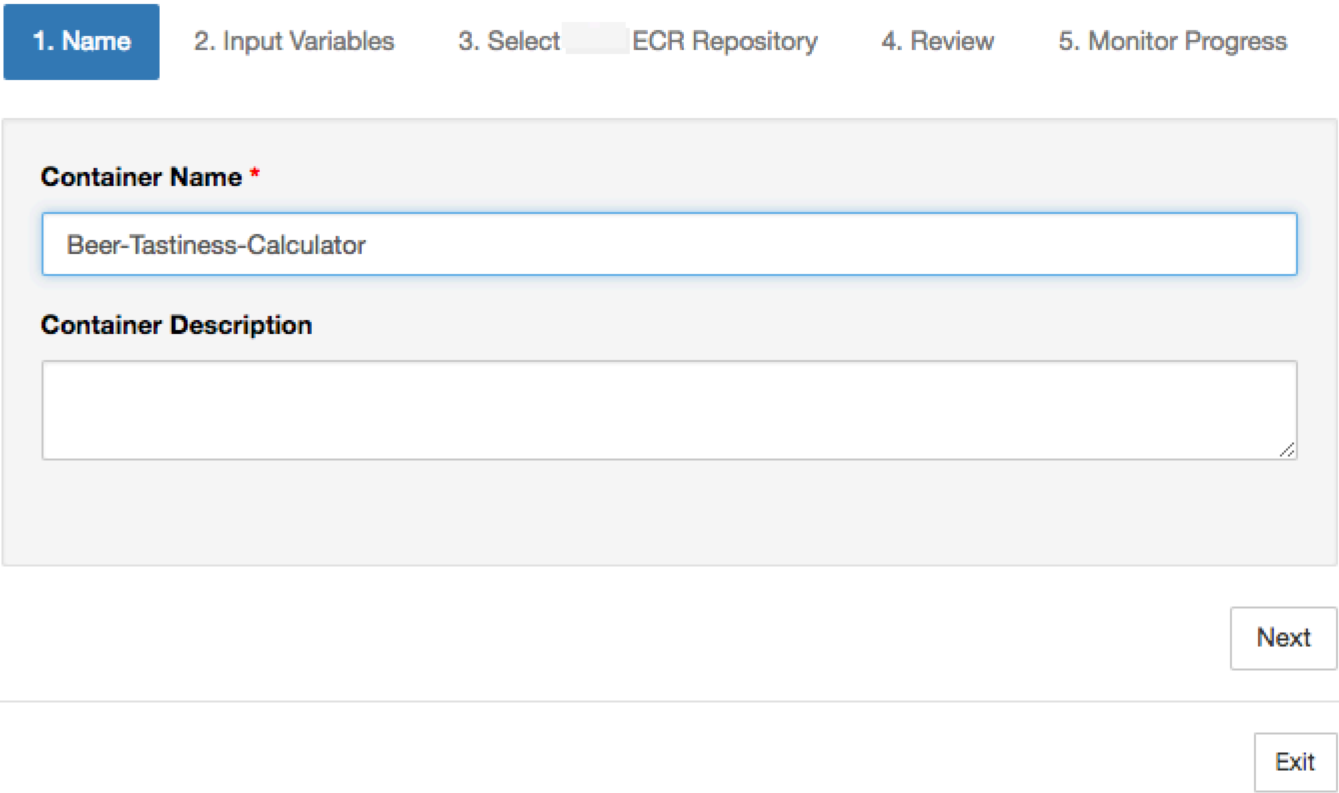
1. In your Jupyter Notebook, choose **New**, then choose the kernel type you want from the dropdown list. (The kernel type should start with "Containerized" and end with whatever kernel you would have otherwise selected. For example, if you just want a plain Python 3.0 environment like "conda_python3", choose "Containerized conda_python3").


jupyter
Quit
Files
Running
Clusters
SageMaker Examples
Conda
Rename
Move
Upload
New
1
/
IoTAnalytics
lost+found
Untitled.ipynb
Notebook:
Containerized conda_chainer_p27
Containerized conda_chainer_p36
Containerized conda_mxnet_p27
Containerized conda_mxnet_p36
Containerized conda_python2
Containerized conda_python3
Containerized conda_pytorch_p27
Containerized conda_pytorch_p36
Containerized conda_tensorflow_p27
Containerized conda_tensorflow_p36
Sparkmagic (PySpark)
Sparkmagic (PySpark3)
Sparkmagic (Spark)
Sparkmagic (SparkR)
conda_chainer_p27
conda_chainer_p36
conda_mxnet_p27
conda_mxnet_p36
conda_python2
conda_python3

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



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

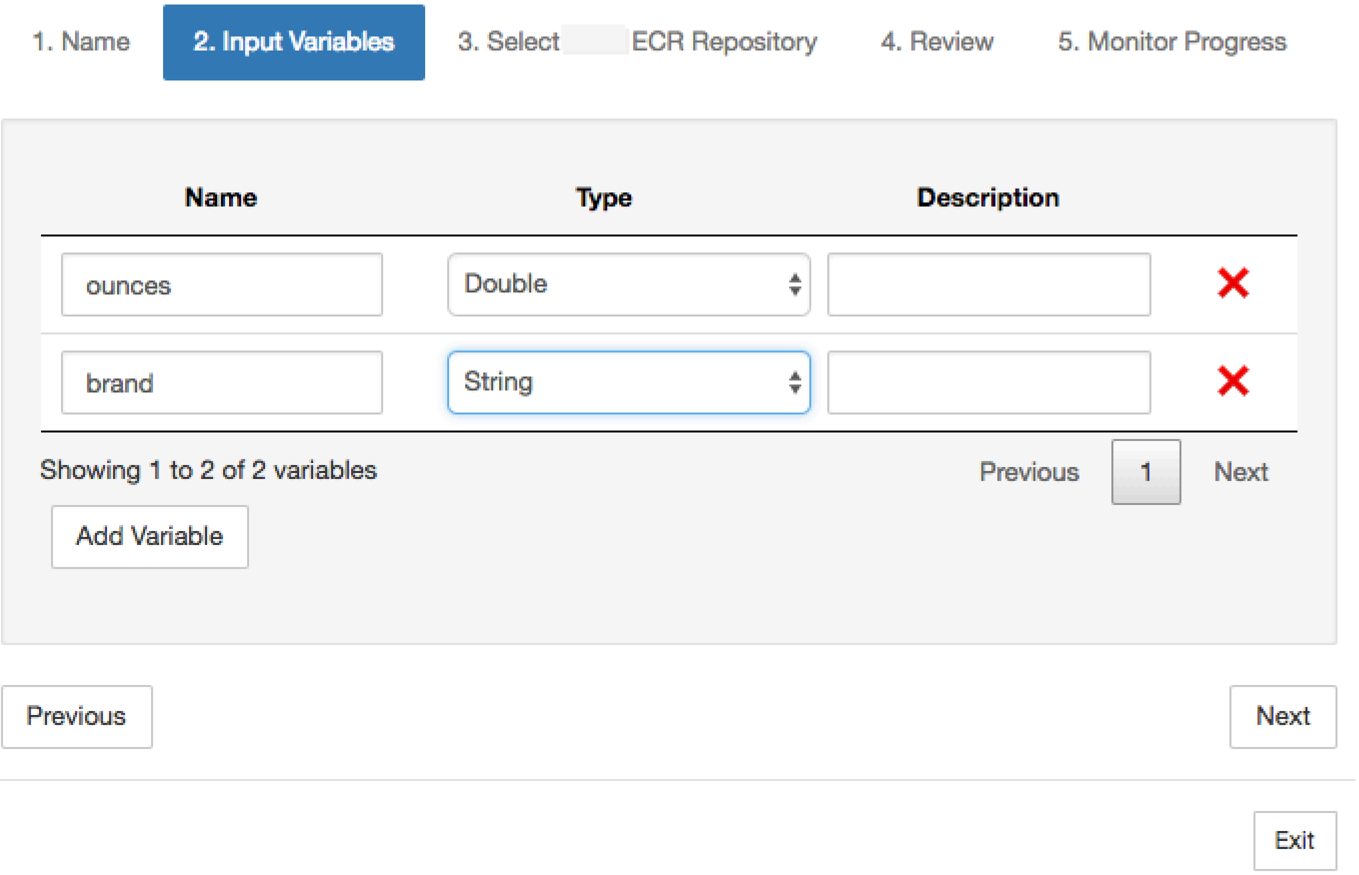


Container Name *

Beer-Tastiness-Calculator

Container Description

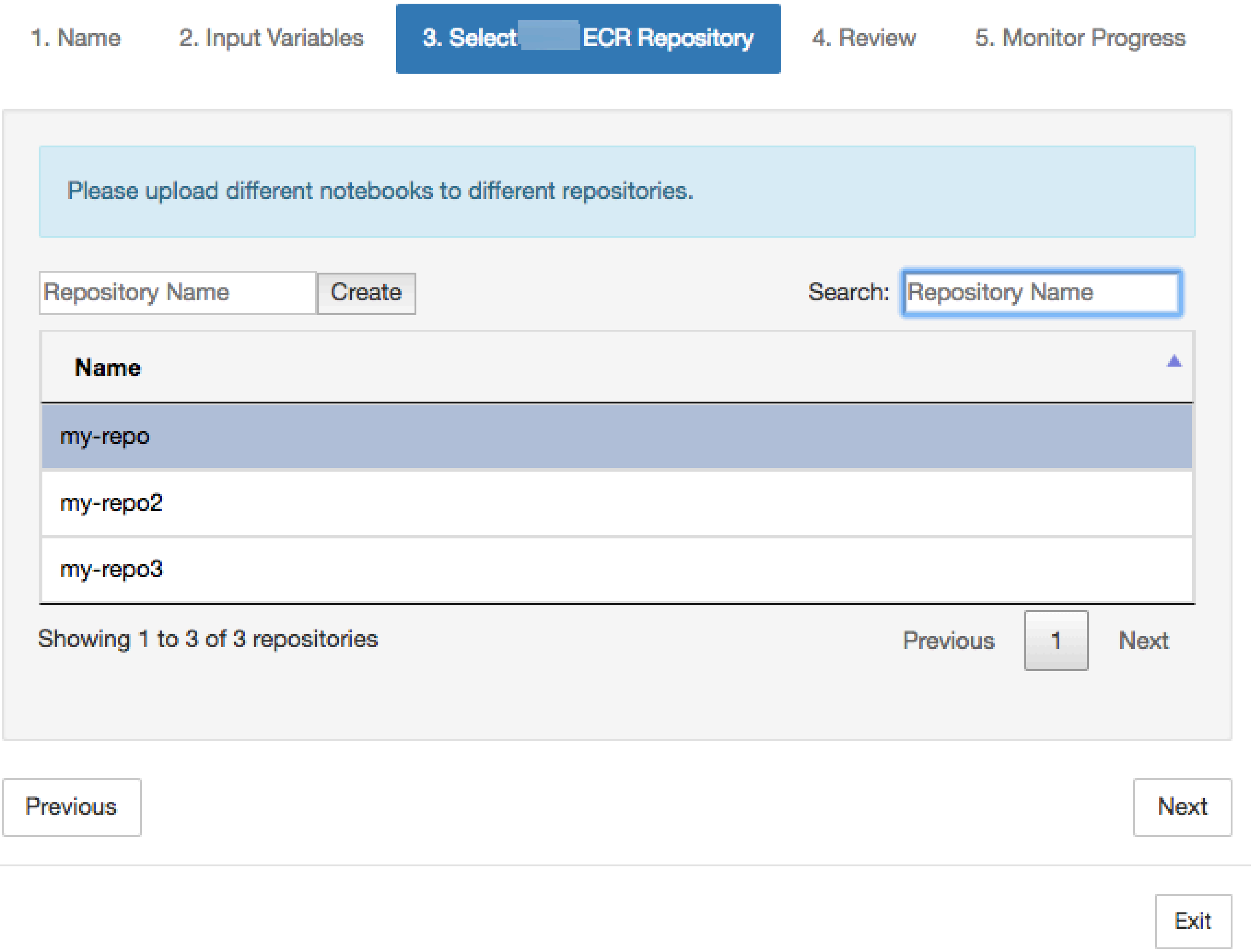
4. Specify the **Input Variables** (parameters) that your notebook should be invoked with. You can select the input variables that are automatically detected from your notebook or define custom variables. (Note that input variables are only detected if you have previously executed your notebook.) For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



Name	Type	Description
ounces	Double	
brand	String	

Showing 1 to 2 of 2 variables

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



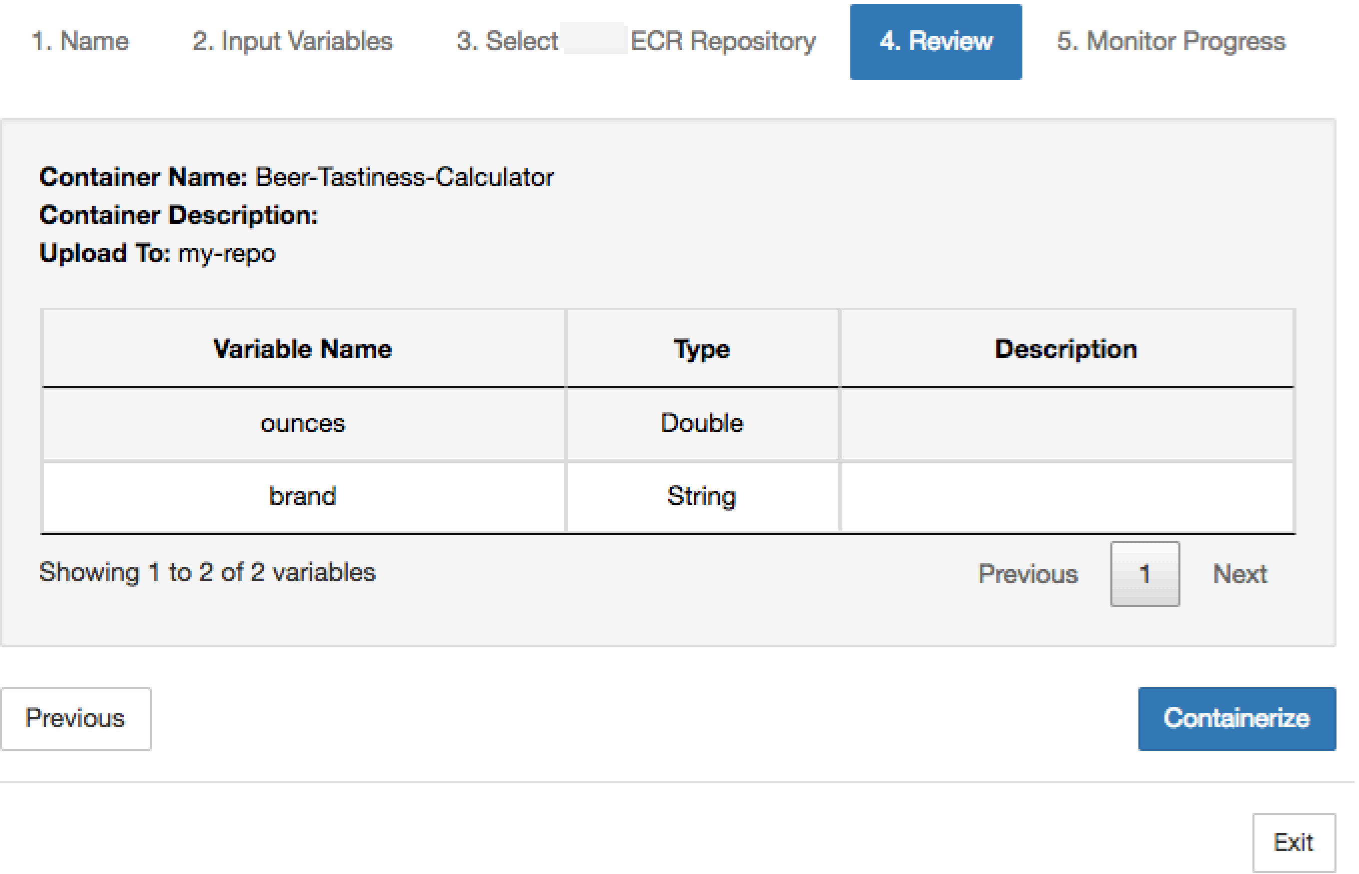
Please upload different notebooks to different repositories.

Name
my-repo
my-repo2
my-repo3

Showing 1 to 3 of 3 repositories

6. Choose **Containerize** to begin the process.

   You are presented with an overview summarizing your input. Note that after you have started the process you can't cancel it. The process might last up to an hour.

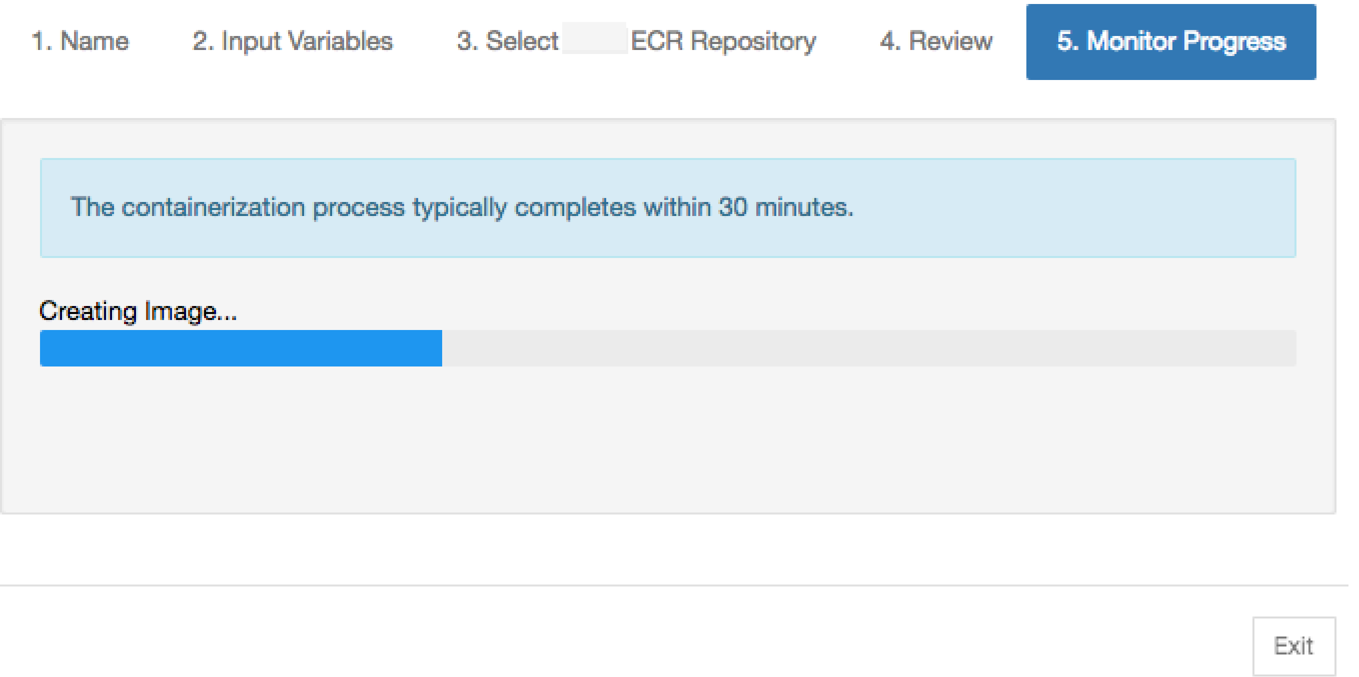


Container Name: Beer-Tastiness-Calculator
Container Description:
Upload To: my-repo

Variable Name	Type	Description
ounces	Double	
brand	String	

Showing 1 to 2 of 2 variables

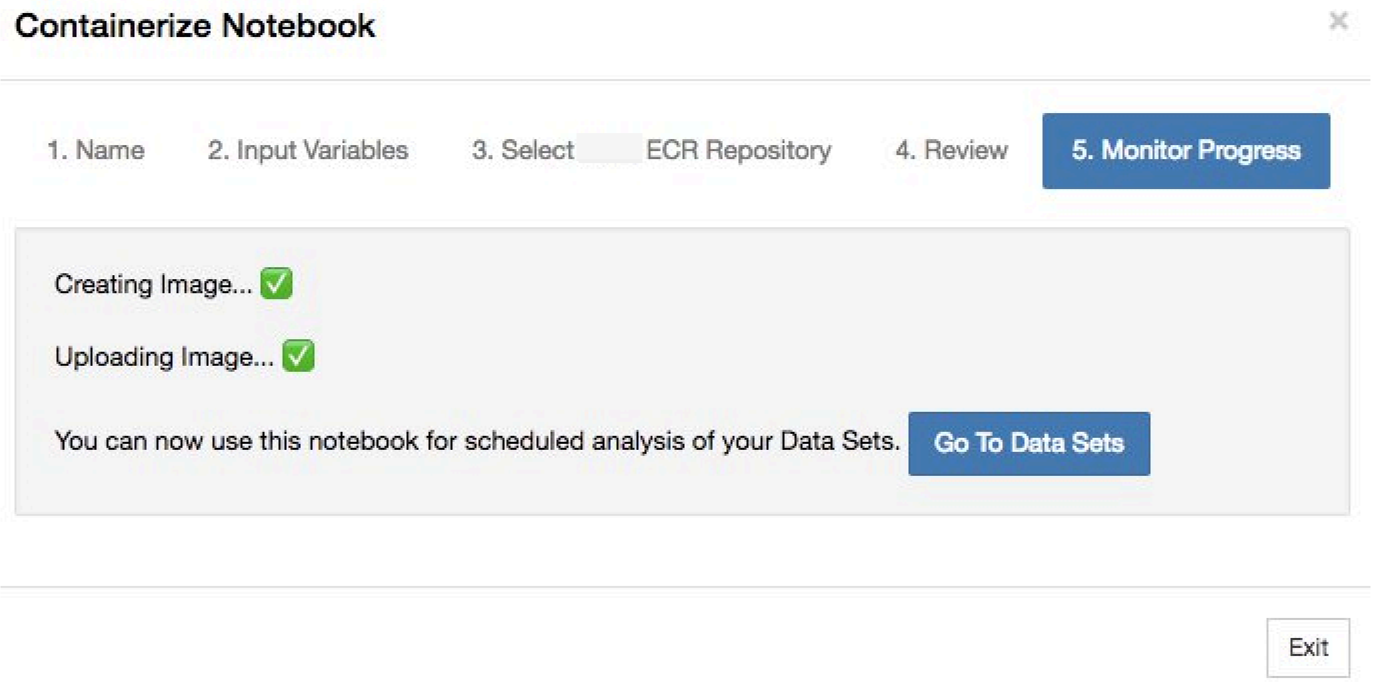
7. The next page shows the progress.



The containerization process typically completes within 30 minutes.

Creating Image...

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



Creating Image... ✓

Uploading Image... ✓

You can now use this notebook for scheduled analysis of your Data Sets.

Using a custom container for analysis

This section includes information 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 Amazon 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 Amazon IoT Analytics service. To do so, you setup a Docker image and upload it to Amazon ECR, then set up a dataset yo 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 Amazon account with Amazon ECR or Amazon 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 Amazon IoT Analytics, runs the Octave script, and uploads artifacts to Amazon S3.

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

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

variables = params['Variables']

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

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

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

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

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

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

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

- `moment` – A simple Octave script which calculates the moment based on an input or 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 and then `cd` to that directory.
2. Run the following command.

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

3. You should see a new image in your Docker repository. Verify it by running the following command.

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

Step 2: Upload the Docker image to an Amazon ECR repository

1. Create a repository in Amazon 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 the following.

```
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 Amazon ECR repository tag.

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

5. Push the image to Amazon ECR.

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

Step 3: Upload your sample data to an Amazon 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 an Amazon S3 bucket called `octave-sample-data-`*`your-aws-account-id`*.
3. Upload the file `input.txt` to the Amazon S3 bucket you just created. You should now have a bucket named `octave-sample-data-`*`your-aws-account-id`* that contains the `input.txt` file.

Step 4: Create a container execution role

1. Copy the following to a file named `role1.json`. Replace *`your-aws-account-id`* with your Amazon account ID and *`aws-region`* with the Amazon region of your Amazon resources.

> **Note**
>
> This example includes a global condition context key to protect against the confused deputy security problem. For more information, see the section called "Cross-service confused deputy prevention".

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

2. Create a role that gives access permissions to SageMaker AI and Amazon IoT Analytics, using the file `role1.json` that you 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 `policy.json` file 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 dataset with a container action

1. Download the following to a fie 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 dataset 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 dataset content generation

1. Run the following command.

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

Step 7: Get dataset content

1. Run the following command.

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

2. You might 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 the following command.

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

Visualizing Amazon IoT Analytics data

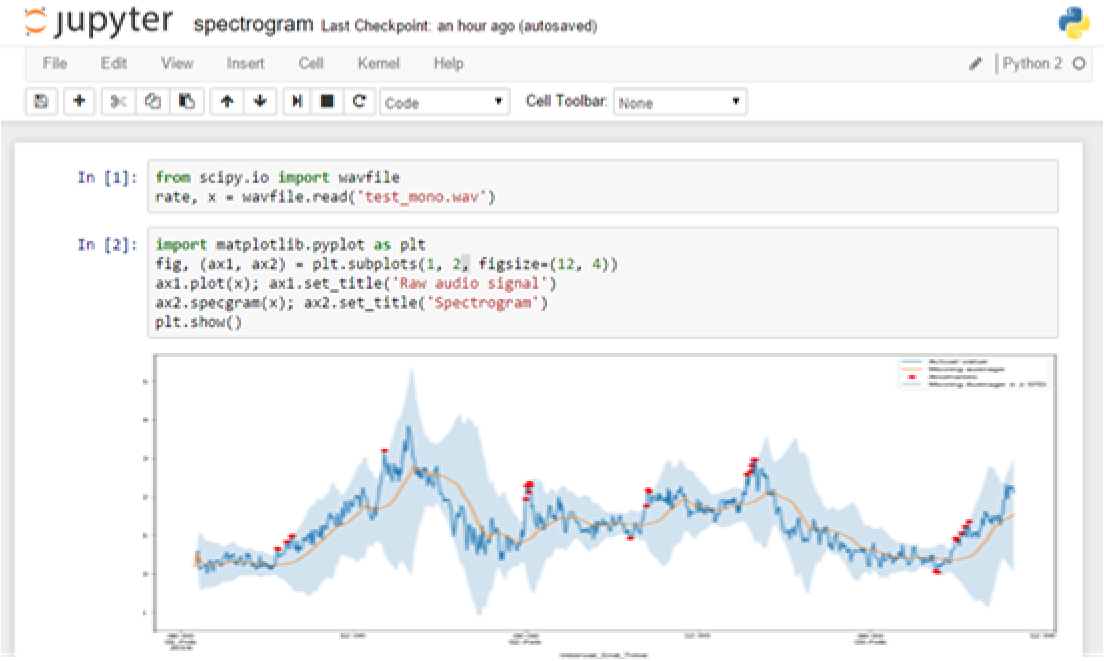
To visualize your Amazon IoT Analytics data, you can use the Amazon IoT Analytics console or QuickSight.

Topics

- Visualizing Amazon IoT Analytics data with the console

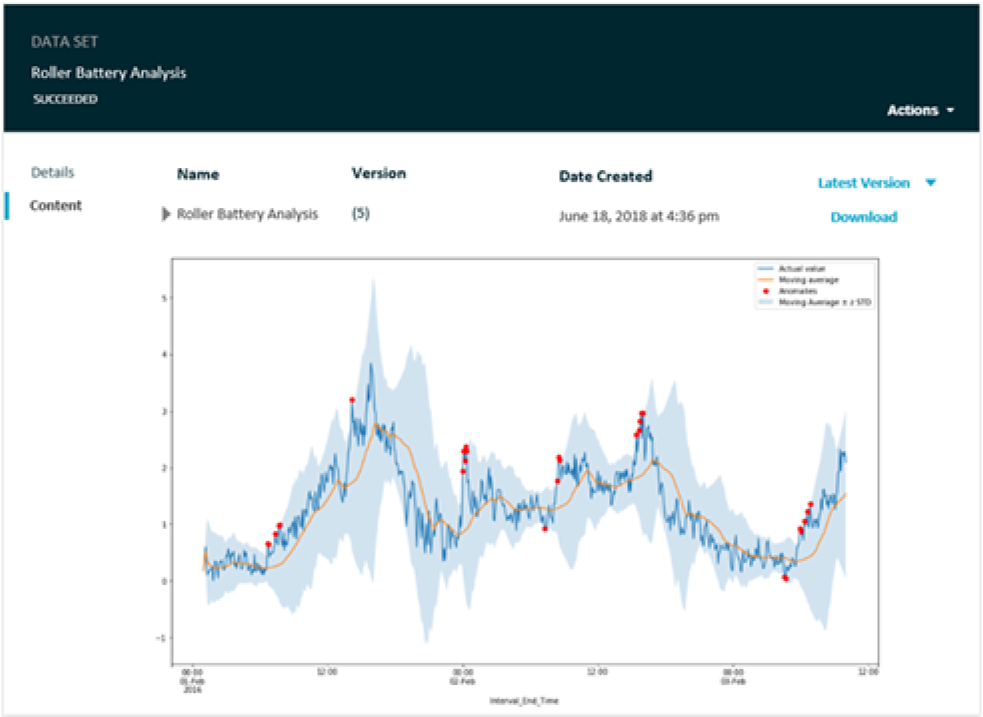
Visualizing Amazon IoT Analytics data with the console

Amazon IoT Analytics can embed the HTML output of your container dataset (found in the file `output.html`) on the container dataset content page of the Amazon IoT Analytics console. For example, if you define a container dataset that runs a Jupyter notebook, and you create a visualization in your Jupyter notebook, your dataset might look like the following.



```
In [1]: from scipy.io import wavfile
        rate, x = wavfile.read('test_mono.wav')

In [2]: import matplotlib.pyplot as plt
        fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 4))
        ax1.plot(x); ax1.set_title('Raw audio signal')
        ax2.specgram(x); ax2.set_title('Spectrogram')
        plt.show()
```

Then, after the container dataset content is created, you can view this visualization on the console's **Data Set** content page.



DATA SET

Roller Battery Analysis

SUCCEEDED

Actions ▾

Details

Content

Name	Version	Date Created	Latest Version ▾
▸ Roller Battery Analysis	(5)	June 18, 2018 at 4:36 pm	Download

For information about creating a container dataset that runs a Jupyter notebook, see Automating your workflow.

Tagging your Amazon 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 describe tags and shows you how to create them.

Topics

Tag basics

Tags enable you to categorize your Amazon 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, Amazon 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 Amazon Billing User Guide.

For ease of use, use the **Tag Editor** in the Amazon Billing and Cost 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 Amazon Web Services Management Console.

You can also work with tags using the Amazon CLI and the Amazon 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
- CreateDataset
- CreateDatastore
- CreatePipeline

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

- TagResource
- ListTagsForResource
- UntagResource

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 olde 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>` yo 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 Amazon IoT Analytics actions where an identifier for a resource capable of being tagged is a required parameter. For example, the use of DescribeLoggingOptions 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.

For more information, see Controlling access using tags in the *IAM User Guide*. 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-based restrictions. A user restricted by this policy:

1. Can't give a resource the tag "env=prod" (see the line `"aws:RequestTag/env" : "prod"` in the example).
2. Can't 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 the following example.

```
"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 Amazon 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 source 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: + - = . _ : / @.

SQL expressions in Amazon IoT Analytics

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

Amazon IoT Analytics supports a subset of ANSI standard SQL syntax.

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

For a description of the parameters, see Parameters in the *Amazon Athena documentation*.

Amazon IoT Analytics and Amazon Athena doesn't support the following:

- `WITH` clauses.
- `CREATE TABLE AS SELECT` statements
- `INSERT INTO` statements
- Prepared statements, you can't run `EXECUTE` with `USING`.
- `CREATE TABLE LIKE`
- `DESCRIBE INPUT` and `DESCRIBE OUTPUT`
- `EXPLAIN` statements
- User-defined functions (UDFs or UDAFs)
- Stored procedures
- Federated connectors

Topics

Supported SQL functionality in Amazon IoT Analytics

Datasets are generated by using SQL expressions on data in a data store. The queries you run in Amazon IoT Analytics are based on Presto 0.217.

Supported data types

Amazon IoT Analytics and Amazon Athena support these 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][,...]>`

Note

Amazon IoT Analytics and Amazon Athena don't support some data types.

Supported functions

Amazon Athena and Amazon IoT Analytics SQL functionality are based on Presto 0.217. For information about related functions, operators, and expressions, see Functions and Operators and the following specific sections from the Presto documentation.

- 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

Note

Amazon IoT Analytics and Amazon Athena don't support user-defined functions (UDFs or UDAFs) or stored procedures.

Troubleshoot common issues with SQL queries in Amazon IoT Analytics

Use the following information to help troubleshoot issues with your SQL queries in Amazon IoT Analytics.

- **To escape a single quote**, precede it with another single quote. Don't confuse this with a double quote.

  Example Example

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

- **To escape underscores**, use backticks to enclose data store column names that begin with an underscore.

  Example Example

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

- **To escape names with numbers**, enclose data store names that include numbers in double quotes.

  Example Example

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

- **To escape reserved keywords**, enclose reserved keywords in double quotes. For more information, see List of Reserved Keywords in *SQL SELECT Statements*.

Security in Amazon IoT Analytics

Cloud security at Amazon is the highest priority. As an Amazon 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 Amazon and you. The shared responsibility model described this as security *of* the cloud and security *in* the cloud:

- **Security of the cloud** - Amazon is responsible for protecting the infrastructure that runs Amazon services in the Amazon Cloud. Amazon 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 Amazon compliance programs. To learn about the compliance programs that apply to Amazon IoT Analytics, see Amazon services in scope by compliance program.
- **Security in the cloud** - Your responsibility is determined by the Amazon 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 Amazon IoT Analytics. The following topics show you how to configure Amazon IoT Analytics to meet your security and compliance objectives. You'll also learn how to use other Amazon services that can help you to monitor and secure your Amazon IoT Analytics resources.

Amazon Identity and Access Management in Amazon IoT Analytics

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

Audience

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

Service user – If you use the Amazon IoT Analytics service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more Amazon 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 Amazon IoT Analytics, see Troubleshooting Amazon IoT Analytics identity and access.

Service administrator – If you're in charge of Amazon IoT Analytics resources at your company, you probably have full access to Amazon IoT Analytics. It's your job to determine which Amazon IoT Analytics features and resources your service users 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 Amazon IoT Analytics, see How Amazon IoT Analytics works with IAM.

IAM administrator – If you're an IAM administrator, you might want to learn details about how you can write policies to manage access to Amazon IoT Analytics. To view example Amazon IoT Analytics identity-based policies that you can use in IAM, see Amazon IoT Analytics identity-based policy examples.

Authenticating with identities

Authentication is how you sign in to Amazon using your identity credentials. You must be *authenticated* (signed in to Amazon) as the Amazon Web Services account root user, as an IAM user, or by assuming an IAM role.

If you access Amazon programmatically, Amazon provides a software development kit (SDK) and a command line interface (CLI) to cryptographically sign your requests by using your credentials. If you don't use Amazon tools, you must sign requests yourself. For more information about using the recommended method to sign requests yourself, see Amazon Signature Version 4 for API requests in the *IAM User Guide*.

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

Amazon Web Services account root user

When you create an Amazon Web Services account, you begin with one sign-in identity that has complete access to all Amazon Web Services services and resources in the account. This identity is called the Amazon Web Services 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 don't use the root user for your everyday tasks. Safeguard your root user credentials and use them to perform the tasks that only the root user can perform. For the complete list of tasks that require you to sign in as the root user, see Tasks that require root user credentials in the *IAM User Guide*.

IAM users and groups

An *IAM user* is an identity within your Amazon Web Services account that has specific permissions for a single person or application. Where possible, we recommend relying on temporary credentials instead of creating IAM users who have long-term credentials such as passwords and access keys. However, if you have specific use cases that require long-term credentials with IAM users, we recommend that you rotate access keys. For more information, see Rotate access keys regularly for use cases that require long-term credentials in the *IAM User Guide*.

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 Use cases for IAM users in the *IAM User Guide*.

IAM roles

An *IAM role* is an identity within your Amazon Web Services account that has specific permissions. It is similar to an IAM user, but is not associated with a specific person. To temporarily assume an IAM role in the Amazon Web Services Management Console, you can switch from a user to an IAM role (console). You can assume a role by calling an Amazon CLI or Amazon API operation or by using a custom URL. For more information about methods for using roles, see Methods to assume a role in the *IAM User Guide*.

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

- **Federated user access** – To assign permissions to a federated identity, you create a role and define permissions for the role. When a federated identity authenticates, the identity is associated with the role and is granted the permissions that are defined by the role. For information about roles for federation, see Create a role for a third-party identity provider (federation) in the *IAM User Guide*.

- **Temporary IAM user permissions** – An IAM user or role can assume an IAM role to temporarily take on different permissions for a specific task.
- **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 Amazon Web Services 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 Cross account resource access in IAM in the *IAM User Guide*.
- **Cross-service access** – Some Amazon Web Services services use features in other Amazon Web Services services. For example, when you make a call in a service, it's common for that service to run applications in Amazon EC2 or store objects in Amazon S3. A service might do this using the calling principal's permissions, using a service role, or using a service-linked role.
  - **Forward access sessions (FAS)** – When you use an IAM user or role to perform actions in Amazon, you are considered a principal. When you use some services, you might perform an action that then initiates another action in a different service. FAS uses the permissions of the principal calling an Amazon Web Services service, combined with the requesting Amazon Web Services service to make requests to downstream services. FAS requests are only made when a service receives a request that requires interactions with other Amazon Web Services services or resources to complete. In this case, you must have permissions to perform both actions. For policy details when making FAS requests, see Forward access sessions.
  - **Service role** – A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Create a role to delegate permissions to an Amazon Web Services service in the *IAM User Guide*.
  - **Service-linked role** – A service-linked role is a type of service role that is linked to an Amazon Web Services service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your Amazon Web Services account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.
- **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 Amazon CLI or Amazon API requests. This is preferable to storing access keys within the EC2 instance. To assign an Amazon 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 Use an IAM role to grant permissions to applications running on Amazon EC2 instances in the *IAM User Guide*.

Managing access using policies

You control access in Amazon by creating policies and attaching them to Amazon identities or resources. A policy is an object in Amazon that, when associated with an identity or resource, defines their permissions. Amazon evaluates these policies when a principal (user, root user, or role session) makes a request. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in Amazon 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*.

Administrators can use Amazon JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

By default, users and roles have no permissions. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

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 Amazon Web Services Management Console, the Amazon CLI, or the Amazon API.

Identity-based policies

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Define custom IAM permissions with customer managed 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 Amazon Web Services account. Managed policies include Amazon managed policies and customer managed policies. To learn how to choose between a managed policy or an inline policy, see Choose between managed policies and inline policies in the *IAM User Guide*.

Other policy types

Amazon 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 an 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 Amazon Organizations. Amazon Organizations is a service for grouping and centrally managing multiple Amazon Web Services accounts that your business owns. If you enable all features in an organization, then you can apply service control policies (SCPs) to any or all of your accounts. The SCP limits permissions for entities in member accounts, including each Amazon Web Services account root user. For more information about Organizations and SCPs, see Service control policies in the *Amazon Organizations User Guide*.
- **Resource control policies (RCPs) –** RCPs are JSON policies that you can use to set the maximum available permissions for resources in your accounts without updating the IAM policies attached to each resource that you own. The RCP limits permissions for resources in member accounts and can impact the effective permissions for identities, including the Amazon Web Services account root user, regardless of whether they belong to your organization. For more information about Organizations and RCPs, including a list of Amazon Web Services services that support RCPs, see Resource control policies (RCPs) in the *Amazon 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 Amazon determines whether to allow a request when multiple policy types are involved, see Policy evaluation logic in the *IAM User Guide*.

How Amazon IoT Analytics works with IAM

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

Topics on this page:

- Amazon IoT Analytics identity-based policies
- Amazon IoT Analytics resource-based policies
- Authorization based on Amazon IoT Analytics tags
- Amazon IoT Analytics IAM roles

Amazon 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. Amazon 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 Amazon API operation. The actions is used in a policy to grant permissions to perform the associated operation.

Policy action in Amazon IoT Analytics use the following prefix before the action: `iotanalytics:` For example, to grant someone permission to create an Amazon IoT Analytics channel with the Amazon IoT Analytics `CreateChannel` API operation, you include the `iotanalytics:BatchPuMessage` action in their policy. Policy statements must include either an

`Action` or `NotAction` element. Amazon 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 Amazon IoT Analytics actions, see Actions defined by Amazon 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 Amazon 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 Amazon service namespaces.

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

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

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

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

Some Amazon 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 Amazon IoT Analytics API actions involve multiple resources. For example, `CreatePipeline` references as a channel and a dataset, so a 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 Amazon IoT Analytics resource types and their ARNs, see Resources defined by Amazon IoT Analytics in the *IAM User Guide*. To learn with which actions you can specify the ARN of each resource, see Actions defined by Amazon 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, Amazon evaluates them using a logical `AND` operation. If you specify multiple values for a single condition key, Amazon 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 a user permission to access a resource only if it is tagged with their user name. For more information, see IAM policy elements: Variables and tags in the *IAM User Guide*.

Amazon IoT Analytics does not provide any sevice-specific condition keys, but it does support using some global condition keys. To see all Amazon global condition keys, see Amazon global condition context keys. in the *IAM User Guide*.

Examples

To view examples of Amazon IoT Analytics identity-based policies, see Amazon IoT Analytics identity-based policy examples.

Amazon IoT Analytics resource-based policies

Amazon IoT Analytics does not support resource-based policies. To view an example of a detailed resource-based policy page, see Using resource-based policies for Amazon Lambda in the *Amazon Lambda Developer Guide*.

Authorization based on Amazon IoT Analytics tags

You can attach tags to Amazon IoT Analytics resources or pass tags in a request to Amazon IoT Analytics. To control access based on tags, your 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 Amazon IoT Analytics resources, see Tagging your Amazon IoT Analytics resources.

To view an example identity-based policy for limiting access to a resource based on the tags on that resource, see Viewing Amazon IoT Analytics channels based on tags.

Amazon IoT Analytics IAM roles

An IAM role is an entity within your Amazon Web Services account that has specific permissions.

Using temporary credential with Amazon 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 Amazon Security Token Service (Amazon STS) API operations such as AssumeRole or GetFederationToken.

Amazon IoT Analytics does not support using temporary credentials.

Service-linked roles

Service-lined roles allow Amazon service 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.

Amazon 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.

Amazon IoT Analytics supports service roles.

Cross-service confused deputy prevention

The confused deputy problem is a security issue where an entity that doesn't have permission to perform an action can coerce a more-privileged entity to perform the action. In Amazon, cross-service impersonation can result in the confused deputy problem. Cross-service impersonation can occur when one service (the *calling service*) calls another service (the *called service*). The calling service can be manipulated to use its permissions to act on another customer's resources in a way it shouldn't otherwise have permission to access. To prevent this, Amazon provides tools that help you protect your data for all services, with service principals that have been given access to resources in your account.

We recommend using the `aws:SourceArn` and `aws:SourceAccount` global condition context keys in resource policies. This limits the permissions that Amazon IoT Analytics gives another service to the resource. If you use both global condition context keys, the `aws:SourceAccount` value and the account in the `aws:SourceArn` value must use the same account ID when used in the same policy statement.

The most effective way to protect against the confused deputy problem is to use the `aws:SourceArn` global condition context key with the full Amazon Resource Name (ARN) of the resource. If you don't know the full ARN of the resource or if you're specifying multiple resources, use the `aws:SourceArn` global context condition key with wildcards (`*`) for the unknown portions of the ARN. For example, `arn:aws:`*`iotanalytics`*`::`*`123456789012`*`:*`.

Topics

- Prevention for Amazon S3 buckets
- Prevention with Amazon CloudWatch Logs
- Confused deputy prevention for customer managed Amazon IoT Analytics resources

Prevention for Amazon S3 buckets

If you use customer managed Amazon S3 storage for your Amazon IoT Analytics data store, the Amazon S3 bucket that stores your data may be exposed to confused deputy issues.

For example, Nikki Wolf uses a customer owned Amazon S3 bucket called *DOC-EXAMPLE-BUCKET*. The bucket stores information for an Amazon IoT Analytics data store that was created in the Region *us-east-1*. She specifies a policy that enables the Amazon IoT Analytics service principal to query *DOC-EXAMPLE-BUCKET* on her behalf. Nikki's coworker, Li Juan, queries *DOC-EXAMPLE-BUCKET* from her own account and creates a dataset with the results. As a result, the Amazon IoT Analytics service principal queried Nikki's Amazon S3 bucket on Li's behalf even though Li ran the query from her account.

To prevent this, Nikki can specify the `aws:SourceAccount` condition or the `aws:SourceArn` condition in the policy for *DOC-EXAMPLE-BUCKET*.

Specify the `aws:SourceAccount` condition - The following example of a bucket policy specifies that only the Amazon IoT Analytics resources from Nikki's account (*123456789012*) can access *DOC-EXAMPLE-BUCKET*.

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

Specify the `aws:SourceArn` condition - Alternatively, Nikki can use the `aws:SourceArn` condition.

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

Prevention with Amazon CloudWatch Logs

You can prevent the confused deputy problem while monitoring with Amazon CloudWatch Logs. The following resource policy shows how to prevent the confused deputy problem with:

- The global condition context key, `aws:SourceArn`
- The `aws:SourceAccount` with your Amazon account ID
- The customer resource that is associated with the `sts:AssumeRole` request in Amazon IoT Analytics

Replace *`123456789012`* with your Amazon account ID, and *`us-east-1`* with the Region of your Amazon IoT Analytics account in the following example.

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

For more information about enabling and configuring Amazon CloudWatch Logs, see [the section called “Logging and monitoring”](#).

Confused deputy prevention for customer managed Amazon IoT Analytics resources

If you grant Amazon IoT Analytics permission to perform actions on your Amazon IoT Analytics resources, the resources may be exposed to confused deputy issues. To prevent the confused deputy problem, you can limit the permissions given to Amazon IoT Analytics with the following example resource policies.

Topics

- Prevention for Amazon IoT Analytics channels and data stores
- Cross-service confused deputy prevention for Amazon IoT Analytics dataset content delivery rules

Prevention for Amazon IoT Analytics channels and data stores

You use IAM roles to control the Amazon resources that Amazon IoT Analytics can access on your behalf. To prevent exposing your role to the confused deputy problem, you can specify the Amazon account in the `aws:SourceAccount` element and the ARN of the Amazon IoT Analytics resource in the `aws:SourceArn` element of the trust policy that you attach to a role.

In the following example, replace *123456789012* with your Amazon account ID and *arn:aws:iotanalytics:aws-region:123456789012:channel/DOC-EXAMPLE-CHANNEL* with the ARN of an Amazon IoT Analytics channel or data store.

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

To learn more about customer managed S3 storage options for channels and data stores, see `CustomerManagedChannelS3Storage` and `CustomerManagedDatastoreS3Storage` in the *Amazon IoT Analytics API Reference*.

Cross-service confused deputy prevention for Amazon IoT Analytics dataset content delivery rules

The IAM role that Amazon IoT Analytics assumes to deliver dataset query results to Amazon S3 or to Amazon IoT Events can be exposed to confused deputy issues. To prevent the confused deputy problem, specify the Amazon account in the `aws:SourceAccount` element and the ARN of the Amazon IoT Analytics resource in the `aws:SourceArn` element of the trust policy that you attach to your role.

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

For more details about configuring dataset content delivery rules, see `contentDeliveryRules` in the *Amazon IoT Analytics API Reference*.

Amazon IoT Analytics identity-based policy examples

By default, users and roles don't have permission to create or modify Amazon IoT Analytics resources. They also can't perform tasks using the Amazon Web Services Management Console, Amazon CLI, or Amazon 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 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*

Topics on this page:

- Policy best practices
- Using the Amazon IoT Analytics console
- Allow users to view their own permissions
- Accessing one Amazon IoT Analytics input
- Viewing Amazon IoT Analytics channels based on tags

Policy best practices

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

- **Get started using Amazon managed policies** - To start using Amazon IoT Analytics quickly, use Amazon managed policies to give your employees the permissions they need. These policies are already available in your account and are maintained and update by Amazon. For more information, see Get started using permissions with Amazon 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 users to use multi-factor authentication (MFA) to access sensitive resources or API operations. For more information, see Using multi-factor authentication (MFA) in Amazon 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 condition 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 Amazon IoT Analytics console

To access the Amazon IoT Analytics console, you must have a minimum set of permissions. These permissions must allow you to list and view details about the Amazon IoT Analytics resources in your Amazon Web Services 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 (users or roles) with that policy.

To ensure that those entities can still use the Amazon IoT Analytics console, also attach the following Amazon 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 Amazon CLI or the Amazon 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 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 Amazon CLI or Amazon API.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": [
                "arn:aws-cn: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 Amazon IoT Analytics input

In this example, you want to grant a user in your Amazon Web Services account access to one of your Amazon IoT Analytics channels, `exampleChannel`. You also want to allow the use 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-cn:iotanalytics:::*"
      },
      {
         "Sid":"ViewSpecificChannelInfo",
         "Effect":"Allow",
         "Action":[
            "iotanalytics:DescribeChannel"
         ],
         "Resource":"arn:aws-cn:iotanalytics:::exampleChannel"
      },
      {
         "Sid":"ManageChannels",
         "Effect":"Allow",
         "Action":[
            "iotanalytics:CreateChannel",
            "iotanalytics:DeleteChannel",
            "iotanalytics:DescribeChannel",
            "iotanalytics:ListChannels",
            "iotanalytics:UpdateChannel"
         ],
         "Resource":"arn:aws-cn:iotanalytics:::exampleChannel/*"
      }
   ]
}
```

Viewing Amazon IoT Analytics channels based on tags

You can use conditions in your identity-based policy to control access to Amazon IoT Analytics resources based on tags. This example shows how you might create a policy that allows viewing a `channel`. However, permissions 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-cn:iotanalytics:*:*:channel/*",
            "Condition": {
                "StringEquals": {"iotanalytics:ResourceTag/Owner": "${aws:username}"}
            }
        }
    ]
}
```

You can attach this policy to the users in your account. If a user named `richard-roe` attempts to view an Amazon 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 Amazon IoT Analytics identity and access

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

Topics

I am not authorized to perform an action in Amazon IoT Analytics

If the Amazon Web Services Management Console tells you that you're not authorized to perform an action, the you must contact your administrator for assistance. You administrator is the person that provided you with your user name and password.

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

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

In this case, Mateo asks his administrator 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, your policies must be updated to allow you to pass a role to Amazon IoT Analytics.

Some Amazon Web Services 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 Amazon IoT Analytics. However, the action requires the service to have permissions that are granted by a service role. Mary does not have permissions to pass the role to the service.

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

In this case, Mary's policies must be updated to allow her to perform the `iam:PassRole` action.

If you need help, contact your Amazon administrator. Your administrator is the person who provided you with your sign-in credentials.

I want to allow people outside of my Amazon Web Services account to access my Amazon 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 (ACL), you can use those policies to grant people access to your resources.

To learn more, consult the following:

- To learn whether Amazon IoT Analytics supports these features, see How Amazon IoT Analytics works with IAM.
- To learn how to provide access to your resources across Amazon Web Services accounts that you own, see Providing access to an IAM user in another Amazon Web Services account that you own in the *IAM User Guide*.
- To learn how to provide access to your resources to third-party Amazon Web Services accounts, see Providing access to Amazon Web Services 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*.

Logging and monitoring in Amazon IoT Analytics

Amazon provides tools that you can use to monitor Amazon IoT Analytics. You can configure some of these tools to do the monitoring for you. Some of the tools require manual intervention. We recommend that you automate monitoring tasks as much as possible.

Automated monitoring tools

You can use the following automated monitoring tools to watch Amazon IoT and report when something is wrong:

- **Amazon CloudWatch Logs** - Monitor, store, and access your log files from Amazon CloudTrail or other sources. For more information, see What is Amazon CloudTrail Monitoring Log Files in the *Amazon CloudWatch User Guide*.
- **Amazon CloudTrail log monitoring** - Share log files between accounts, monitor CloudTrail log files in real time by sending them to CloudWatch Logs, write log-processing applications in Java, and validate that your log files have not changed after delivery by CloudTrail. For more information, see Working with CloudTrail log files in the *Amazon CloudTrail User Guide*.

Manual monitoring tools

Another important part of monitoring Amazon IoT involves manually monitoring those items that the CloudWatch alarms don't cover. The Amazon IoT, CloudWatch, and other Amazon service console dashboards provide an at-a-glance view of the state of your Amazon environment. We recommend that you also check the log files on Amazon IoT Analytics.

- The Amazon IoT Analytics console shows:
  - Channels
  - Pipelines
  - Data stores
  - Data sets
  - Notebooks
  - Settings
  - Learn
- The CloudWatch home page shows:
  - Current alarms and status
  - Graphs of alarms and resources
  - Service health status

  In addition, you can use CloudWatch to do the following:

  - Create customized dashboards to monitor the services you care about

- Graph metric data to troubleshoot issues and discover trends
- Search and browse all your Amazon resource metrics
- Create and edit alarms to be notified of problems

Monitoring with Amazon CloudWatch Logs

Amazon IoT Analytics supports logging with Amazon CloudWatch. You can enable and configure Amazon CloudWatch logging for Amazon IoT Analytics by using the `PutLoggingOptions` API operation. This section describes how you can use `PutLoggingOptions` with Amazon Identity and Access Management (IAM) to configure and enable Amazon CloudWatch logging for Amazon IoT Analytics.

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

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

Create an IAM role to enable logging

To create an IAM role to enable logging for Amazon CloudWatch

1. Use the Amazon IAM console or the following Amazon IAM CLI command, CreateRole, to create a new IAM role with a trust relationship policy (trust policy). The trust policy grants an entity, such as Amazon CloudWatch, permission to assume the role.

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

   The `exampleTrustPolicy.json` file contains the following content.

> **Note**
>
> This example includes a global condition context key to protect against the confused deputy security problem. Replace *123456789012* with your Amazon account ID and *aws-region* with the Amazon region of your Amazon resources. For more information, see the section called "Cross-service confused deputy prevention".

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

      }
   ]
}
```

You use the ARN of this role later when you call the Amazon IoT Analytics `PutLoggingOptions` command.

2. Use Amazon IAM PutRolePolicy to attach a permissions policy (a `role policy`) to the role you created in Step 1.

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

The exampleRolePolicy.json file contains the following content.

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

3. To give Amazon IoT Analytics permission to put logging events to Amazon CloudWatch, use the Amazon CloudWatch command PutResourcePolicy.

> **Note**
>
> To help prevent the confused deputy security problem, we recommend that you specify `aws:SourceArn` in your resource policy. This restricts access to allow only those requests that come from a specified account. For more information about the confused deputy problem, see the section called "Cross-service confused deputy prevention".

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

The `exampleResourcePolicy.json` file contains the following resource policy.

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

Configure and enable logging

Use the `PutLoggingOptions` command to configure and enable Amazon CloudWatch logging for Amazon 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 `DecribeLoggingOptions` command to check your logging options settings.

PutLoggingOptions

Sets or updates the Amazon IoT Analytics logging options. 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 a policy that isn't valid), it can take up to five minutes for that change to take effect. For more information, see `PutLoggingOptions`.

DescribeLoggingOptions

Retrieves the current settings of the Amazon IoT Analytics logging options. For more information, see `DescribeLoggingOptions`

Namespace, metrics, and dimensions

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

Namespace
Amazon/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.
DatasetName	The name of the dataset 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.

Dimension	Description
PipelineName	The name of the pipeline that is being monitored.

Monitor with Amazon CloudWatch Events

Amazon IoT Analytics automatically publishes an event to Amazon CloudWatch Events when a runtime error occurs during an Amazon Lambda activity. This event contains a detailed error message and the keys of the Amazon Simple Storage Service (Amazon S3) objects that store the unprocessed channel messages. You can use the Amazon S3 keys to reprocess the unprocessed channel messages. For more information, see Reprocessing channel messages, the StartPipelineReprocessing API in the *Amazon IoT Analytics API Reference*, and What Is Amazon CloudWatch Events in the *Amazon CloudWatch Events User Guide*.

You can also configure targets that enable Amazon CloudWatch Events to send notifications or take further actions. For example, you can send the notification to an Amazon Simple Queue Service (Amazon SQS) queue, and then invoke the `StartReprocessingMessage` API to process the channel messages saved in the Amazon S3 objects. Amazon CloudWatch Events supports many types of targets, such as the following:

- Amazon Kinesis streams
- Amazon Lambda functions
- Amazon Simple Notification Service (Amazon SNS) topics
- Amazon Simple Queue Service (Amazon SQS) queues

For the list of supported targets, see Amazon EventBridge Targets in the *Amazon EventBridge User Guide*.

Your CloudWatch Events resources and the associated targets must be in the Amazon Region where you created your Amazon IoT Analytics resources. For more information, see Service endpoints and quotas in the *Amazon Web Services General Reference*.

The notification sent to Amazon CloudWatch Events for runtime errors in the Amazon Lambda activity uses the following format.

```
{
    "version": "version-id",
    "id": "event-id",
    "detail-type": "IoT Analytics Pipeline Failure Notification",
    "source": "aws.iotanalytics",
    "account": "aws-account",
    "time": "timestamp",
    "region": "aws-region",
    "resources": [
        "pipeline-arn"
    ],
    "detail": {
        "event-detail-version": "1.0",
        "pipeline-name": "pipeline-name",
        "error-code": "LAMBDA_FAILURE",
        "message": "error-message",
        "channel-messages": {
            "s3paths": [
                "s3-keys"
            ]
        },
        "activity-name": "lambda-activity-name",
        "lambda-function-arn": "lambda-function-arn"
    }
}
```

Example notification:

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

Getting late data notifications through Amazon CloudWatch Events

When you create dataset contents using data from a specified time frame, some data might not arrive in time for processing. To allow for a delay, you can specify a `deltaTime` offset for the `QueryFilter` when you create a dataset by applying a `queryAction` (a SQL query). Amazon IoT Analytics still processes the data that arrives within the delta time, and your dataset contents have a time lag. The late data notification feature enables Amazon IoT Analytics to send notifications through Amazon CloudWatch Events when data arrives after the delta time.

You can use the Amazon IoT Analytics console, API, Amazon Command Line Interface (Amazon CLI), or Amazon SDK to specify late data rules for a dataset.

In the Amazon IoT Analytics API, the `LateDataRuleConfiguration` object represents the late data rule settings of a dataset. This object is part of the `Dataset` object associated with the `CreateDataset` and `UpdateDataset` API operations.

Parameters

When you create a late data rule for a dataset with Amazon IoT Analytics, you must specify the following information:

`ruleConfiguration` (`LateDataRuleConfiguration`)

A structure that contains the configuration information of a late data rule.

`deltaTimeSessionWindowConfiguration`

A structure that contains the configuration information of a delta time session window.

DeltaTime specifies a time interval. You can use `DeltaTime` to create dataset contents with data that has arrived in the data store since the last execution. For an example of `DeltaTime`, see Creating a SQL dataset with a delta window (CLI).

`timeoutInMinutes`

A time interval. You can use `timeoutInMinutes` so that Amazon IoT Analytics can batch up late data notifications that have been generated since the last execution. Amazon IoT Analytics sends one batch of notifications to CloudWatch Events at one time.

Type: Integer

Valid range: 1-60

`ruleName`

The name of the late data rule.

Type: String

> **Important**
>
> To specify `lateDataRules`, the dataset must use a `DeltaTime` filter.

Configure late data rules (console)

The following procedure shows you how to configure the late data rule of a dataset in the Amazon IoT Analytics console.

To configure late data rules

1. Sign in to the Amazon IoT Analytics console.
2. In the navigation pane, choose **Data sets**.
3. Under **Data sets**, choose the target data set.
4. In the navigation pane, choose **Details**.
5. In the **Delta window** section, choose **Edit**.
6. Under **Configure data selection filter**, do the following:

    a. For **Data selection window**, choose **Delta time**.

b. For **Offset**, enter a time period, and then choose a unit.

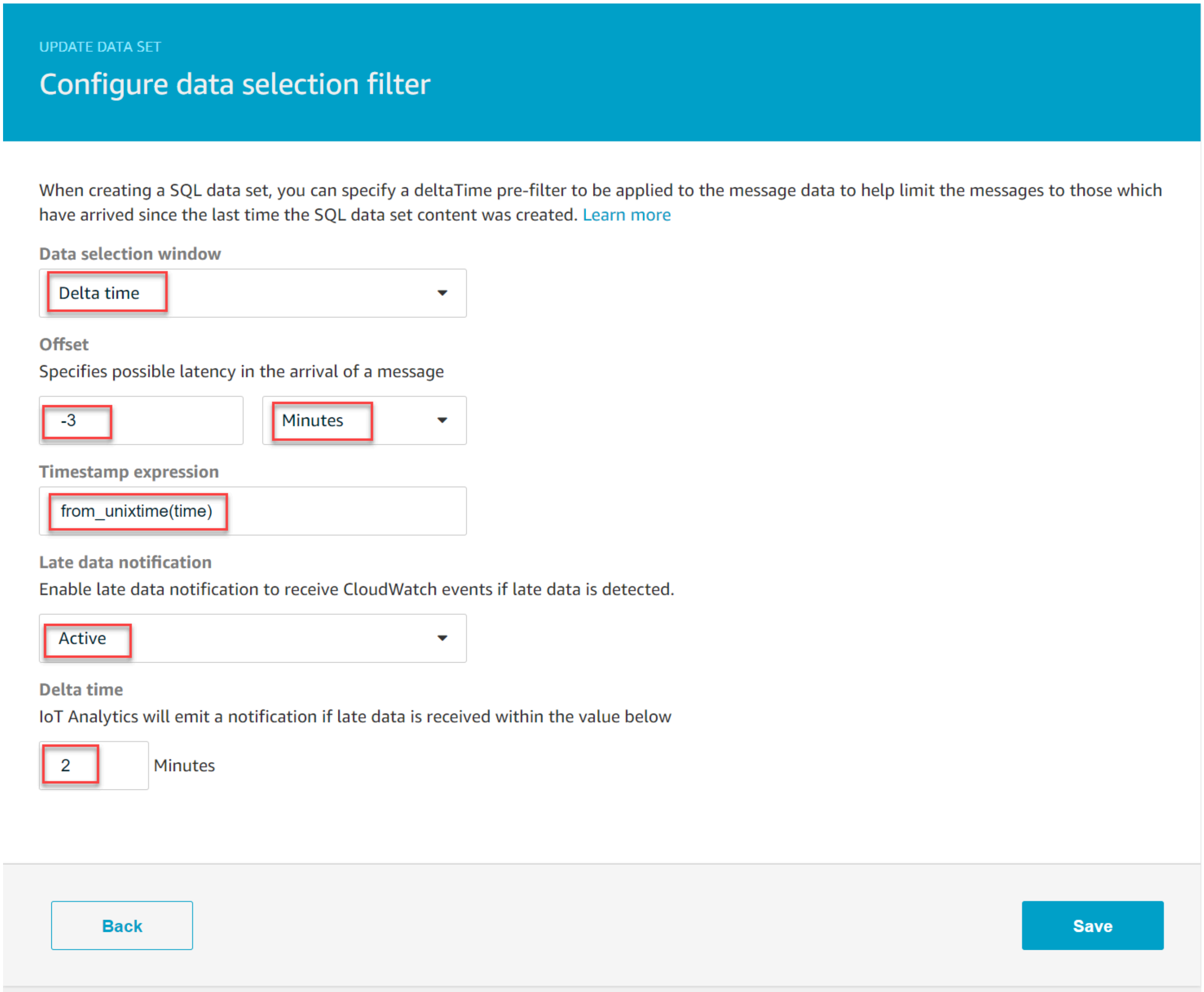
c. For **Timestamp expression**, enter an expression. This can be the name of a timestamp field or a SQL expression that can derive the time, such as `from_unixtime(time)`.

   For more information about how to write a timestamp expression, see Date and Time Functions and Operators in the *Presto 0.172 Documentation*.

d. For **Late data notification**, choose **Active**.

e. For **Delta time**, enter an integer. The valid range is 1-60.

f. Choose **Save**.



Configure data selection filter

When creating a SQL data set, you can specify a deltaTime pre-filter to be applied to the message data to help limit the messages to those which have arrived since the last time the SQL data set content was created. Learn more

Data selection window: Delta time

Offset
Specifies possible latency in the arrival of a message

-3 Minutes

Timestamp expression: from_unixtime(time)

Late data notification
Enable late data notification to receive CloudWatch events if late data is detected.

Active

Delta time
IoT Analytics will emit a notification if late data is received within the value below

2 Minutes

Configure late data rules (CLI)

In the Amazon IoT Analytics API, the `LateDataRuleConfiguration` object represents the late data rule settings of a dataset. This object is part of the `Dataset` object associated with `CreateDataset` and `UpdateDataset`. You can use the API, Amazon CLI, or Amazon SDK to specify late data rules for a dataset. The following example uses the Amazon CLI.

To create your dataset with specified late data rules, run the following command. The command assumes that the `dataset.json` file is in the current directory.

> **Note**
>
> You can use the UpdateDataset API to update an existing dataset.

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

The `dataset.json` file should contain the following:

- Replace *`demo_dataset`* with the target dataset name.
- Replace *`demo_datastore`* with the target data store name.
- Replace *`from_unixtime(time)`* with the name of a timestamp field or a SQL expression that can derive the time.

  For more information about how to write a timestamp expression, see Date and Time Functions and Operators in the *Presto 0.172 Documentation*.
- Replace *`timeout`* with an integer between 1–60.
- Replace *`demo_rule`* with any name.

```
{
    "datasetName": "demo_dataset",
    "actions": [
        {
            "actionName": "myDatasetAction",
            "queryAction": {
                "filters": [
                    {
                        "deltaTime": {
                            "offsetSeconds": -180,
                            "timeExpression": "from_unixtime(time)"
                        }
                    }
                ],
                "sqlQuery": "SELECT * FROM demo_datastore"
            }
        }
    ],
    "retentionPeriod": {
        "unlimited": false,
        "numberOfDays": 90
    },
    "lateDataRules": [
        {
            "ruleConfiguration": {
                "deltaTimeSessionWindowConfiguration": {
                    "timeoutInMinutes": timeout
                }
            },
            "ruleName": "demo_rule"
        }
    ]
}
```

Subscribing to receive late data notifications

You can create rules in CloudWatch Events that defines how to process late data notifications sent from Amazon IoT Analytics. When CloudWatch Events receives the notifications, it invokes specified the target actions defined in your rules.

Prerequisites for creating CloudWatch Events rules

Before you create a CloudWatch Events rule for Amazon IoT Analytics, you should do the following:

- Familiarize yourself with events, rules, and targets in CloudWatch Events.
- Create and configure the targets invoked by your CloudWatch Events rules. Rules can invoke many types of targets, such as the following:
  - Amazon Kinesis streams
  - Amazon Lambda functions
  - Amazon Simple Notification Service (Amazon SNS) topics

- Amazon Simple Queue Service (Amazon SQS) queues

  Your CloudWatch Events rule, and the associated targets must be in the Amazon Region where you created your Amazon IoT Analytics resources. For more information, see Service endpoints and quotas in the *Amazon Web Services General Reference*.

For more information, see What is CloudWatch Events? and Getting started with Amazon CloudWatch Events in the *Amazon CloudWatch Events User Guide*.

Late data notification event

The event for late data notifications uses the following format.

```
{
 "version": "0",
 "id": "7f51dfa7-ffef-97a5-c625-abddbac5eadd",
 "detail-type": "IoT Analytics Dataset Lifecycle Notification",
 "source": "aws.iotanalytics",
 "account": "123456789012",
 "time": "2020-05-14T02:38:46Z",
 "region": "us-east-2",
 "resources": ["arn:aws:iotanalytics:us-east-2:123456789012:dataset/demo_dataset"],
 "detail": {
  "event-detail-version": "1.0",
  "dataset-name": "demo_dataset",
  "late-data-rule-name": "demo_rule",
  "version-ids": ["78244852-8737-4650-aa4d-3071a01338fa"],
  "message": null
 }
}
```

Create a CloudWatch Events rule to receive late data notifications

The following procedure shows you how to create a rule that sends Amazon IoT Analytics late data notifications to an Amazon SQS queue.

To create a CloudWatch Events rule

1. Sign in to the Amazon CloudWatch console.
2. In the navigation pane, under **Events**, choose **Rules**.
3. On the **Rules** page, choose **Create rule**.

4. Under **Event Source**, choose **Event Pattern**.
5. In the **Build event pattern to match events by service** section, do the following:

   a. For **Service Name**, choose **IoT Analytics**

   b. For **Event Type**, choose **IoT Analytics Dataset Lifecycle Notification**.

   c. Choose **Specific dataset name(s)**, and then enter the name of the target dataset.
6. Under **Targets**, choose **Add target***.
7. Choose **SQS queue**, and then do the following:

   - For **Queue***, choose the target queue.
8. Choose **Configure details**.
9. On the **Step 2: Configure rule details** page, enter a name and a description.
10. Choose **Create rule**.

Logging Amazon IoT Analytics API calls with Amazon CloudTrail

Amazon IoT Analytics is integrated with Amazon CloudTrail, a service that provides a record of actions taken by a user, role, or an Amazon service in Amazon IoT Analytics. CloudTrail captures a subset of API calls for Amazon IoT Analytics as events, including calls from the Amazon IoT Analytics console and from code calls to the Amazon IoT Analytics APIs. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for Amazon 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 Amazon 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 Amazon CloudTrail User Guide.

Amazon IoT Analytics information in Amazon CloudTrail

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

For an ongoing record of events in your Amazon account, including events for Amazon 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 Amazon partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other Amazon 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

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

- CancelPipelineReprocessing
- CreateChannel
- CreateDataset
- CreateDatasetContent
- CreateDatastore
- CreatePipeline
- DeleteChannel
- DeleteDataset
- DeleteDatasetContent
- DeleteDatastore
- DeletePipeline
- DescribeChannel
- DescribeDataset
- DescribeDatastore
- DescribeLoggingOptions
- DescribePipeline
- GetDatasetContent
- ListChannels
- ListDatasets

- ListDatastores
- ListPipelines
- PutLoggingOptions
- RunPipelineActivity
- SampleChannelData
- StartPipelineReprocessing
- UpdateChannel
- UpdateDataset
- UpdateDatastore
- UpdatePipeline

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 and Access Management user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another Amazon service.

For more information, see the CloudTrail userIdentity element.

Understanding Amazon 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 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"
}
```

Compliance validation for Amazon IoT Analytics

To learn whether an Amazon Web Services service is within the scope of specific compliance programs, see Amazon Web Services services in Scope by Compliance Program and choose the compliance program that you are interested in. For general information, see Amazon Web Services Compliance Programs.

You can download third-party audit reports using Amazon Artifact. For more information, see Downloading Reports in Amazon Artifact.

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

- Security & Compliance – These solution implementation guides discuss architectural considerations and provide steps for deploying security and compliance features.
- Amazon Compliance Resources – This collection of workbooks and guides might apply to your industry and location.
- Evaluating Resources with Rules in the *Amazon Config Developer Guide* – The Amazon Config service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- Amazon Security Hub – This Amazon Web Services service provides a comprehensive view of your security state within Amazon. Security Hub uses security controls to evaluate your Amazon resources and to check your compliance against security industry standards and best practices. For a list of supported services and controls, see Security Hub controls reference.
- Amazon GuardDuty – This Amazon Web Services service detects potential threats to your Amazon Web Services accounts, workloads, containers, and data by monitoring your environment for suspicious and malicious activities. GuardDuty can help you address various compliance requirements, like PCI DSS, by meeting intrusion detection requirements mandated by certain compliance frameworks.

Resilience in Amazon IoT Analytics

The Amazon global infrastructure is built around Amazon Regions and Availability Zones. Amazon 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 databased 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 Amazon Regions and Availability Zones, see Amazon global infrastructure.

Infrastructure security in Amazon IoT Analytics

As a managed service, Amazon IoT Analytics is protected by Amazon global network security. For information about Amazon security services and how Amazon protects infrastructure, see Amazon Cloud Security. To design your Amazon environment using the best practices for infrastructure security, see Infrastructure Protection in *Security Pillar Amazon Well‐Architected Framework*.

You use Amazon published API calls to access Amazon IoT Analytics through the network. Clients must support the following:

- Transport Layer Security (TLS). We require TLS 1.2 and recommend TLS 1.3.
- Cipher suites with perfect forward secrecy (PFS) such as DHE (Ephemeral Diffie-Hellman) or ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). 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 Amazon Security Token Service (Amazon STS) to generate temporary security credentials to sign requests.

Amazon IoT Analytics quotas

The *Amazon Web Services General Reference Guide* provides the default quotas for Amazon IoT Analytics for an Amazon account. Unless specified, each quota is per Amazon Region. For more information, see Amazon IoT Analytics endpoints and quotas and Amazon service quotas in the *Amazon Web Services General Reference Guide*.

To request a service quota increase, submit a support case in the Support center console. For more information, see Requesting a quota increase in the *Service Quotas User Guide*.

Amazon IoT Analytics commands

Read this topic to learn about the API operations for Amazon IoT Analytics, including sample requests, responses, and errors for the supported web services protocols.

Amazon IoT Analytics actions

You can use Amazon IoT Analytics API commands to collect, process, store, and analyze your IoT data. For more information, see the actions that are supported by Amazon IoT Analytics in the *Amazon IoT Analytics API Reference*.

The Amazon IoT Analytics sections in the *Amazon CLI Command Reference* include the Amazon CLI commands that you can use to administer and manipulate Amazon IoT Analytics.

Amazon IoT Analytics data

You can use the Amazon IoT Analytics Data API commands to perform advanced activities with Amazon IoT Analytics `channel`, `pipeline`, `datastore`, and `dataset`. For more information, see the data types that are supported by Amazon IoT Analytics Data in the *Amazon IoT Analytics API Reference*.

Troubleshooting Amazon IoT Analytics

See the following section to troubleshoot errors and find and possible solutions to resolve issues with Amazon IoT Analytics.

Topics

How do I know if my messages are getting into Amazon 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 the following.

```
{
    "ruleArn": "arn:aws:iot:us-west-2:your-account-id:rule/your-rule-name",
    "rule": {
        "awsIotSqlVersion": "2016-03-23",
        "sql": "SELECT * FROM 'iot/your-rule-name'",
        "ruleDisabled": false,
        "actions": [
            {
                "iotAnalytics": {
                    "channelArn":
 "arn:aws:iotanalytics:region:your_account_id:channel/your-channel-name"
                }
            }
        ],
        "ruleName": "your-rule-name"
    }
}
```

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 Amazon S3 bucket temporarily.

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

- An activity has received an invalid JSON input:

  All activities, except Lambda activities, specially 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. For more information, see Lambda function example 2.
- 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 Amazon IoT Analytics service. You can use the following Amazon 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 if 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.

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 dataset query is incorrect:

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

Why does my dataset just show `__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 dataset contains nothing but this, see the previous question and response.

How do I code an event driven by the dataset completion?

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

How do I correctly configure my notebook instance to use Amazon IoT Analytics?

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 SageMaker AI 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 SageMaker AI can use.
4. Go to the IAM console and modify the newly created SageMaker AI 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 dataset from Amazon IoT Analytics.
7. Go to Amazon 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 datasets.

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.

Why aren't I seeing my datasets in QuickSight?

> ⚠ **Important**
>
> QuickSight isn't available in the China (Beijing) Region. For the list of supported Regions, see QuickSight endpoints and quotas in the *Amazon Web Services General Reference*.

QuickSight might need permission to read your Amazon IoT Analytics dataset content. To give permission, follow these steps.

1. Choose your account name in the upper-right corner of QuickSight and choose **Manage QuickSight**.
2. In the left navigation pane, choose **Security & permissions**. Under **QuickSight access to Amazon services**, verify that access is granted to Amazon IoT Analytics.
   a. If Amazon IoT Analytics doesn't have access, choose **Add or remove**.
   b. Choose the box next to **Amazon IoT Analytics** and then select **Update**. This gives QuickSight permission to read your dataset content.
3. Try again to visualize your data.

Make sure that you choose the same Amazon Region for both Amazon IoT Analytics and QuickSight. Otherwise, you might have issues accessing the Amazon resources. For the list of supported Regions, see Amazon IoT Analytics endpoints and quotas and QuickSight endpoints and quotas in the *Amazon Web Services General Reference*.

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

- This is caused by a missing Amazon IoT Analytics Containerization Plugin. If you created you SageMaker notebook instance before August 23, 2018, you need to manually install the plugin by following the instructions in Containerizing a notebook.
- If you don't see the containerize button after creating the SageMaker notebook instance from the Amazon IoT Analytics console or manually installing it, contact Amazon IoT Analytics technical support.

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 and add the required permissions to the notebook instance role. If the problem persists, create a new notebook instance from the Amazon IoT Analytics console.
- You can safely ignore the following message in the log if it appears during installation of the plugin: "To initialize this extension in the browser every time the notebook (or other app) loads."

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 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.

Why don't I see my variables during the containerization?

- The Amazon 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.

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 your to run the same container with different parameters that need to be supplied at the time of dataset creation. The Amazon 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.

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 `getDatasetContent` API and pick the appropriate output file required for the subsequent run.

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 Amazon 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.

Document history

The following table describes the important changes to the *Amazon IoT Analytics User Guide* after November 3, 2020. For more information about updates to this documentation, you can subscribe to an RSS feed.

Change	Description	Date
End of support notice	End of support notice: On December 15, 2025, Amazon will end support for Amazon IoT Analytics. After December 15, 2025, you will no longer be able to access the Amazon IoT Analytics console or Amazon IoT Analytics resources. For more information, see Amazon IoT Analytics end of support.	May 20, 2025
Amazon IoT Analytics is no longer available to new customers	Amazon IoT Analytics is no longer available to new customers. Existing customers of Amazon IoT Analytics can continue to use the service as normal. Learn more	August 8, 2024
Region launch	Amazon IoT Analytics is now available in the Asia Pacific (Mumbai) region.	August 18, 2021
Query with `JOIN`	This update enables you to use `JOIN` to query an Amazon IoT Analytics dataset.	July 27, 2021

Integration with Amazon IoT SiteWise	You can now use Amazon IoT Analytics to query Amazon IoT SiteWise data.	July 27, 2021
Custom partitions	Amazon IoT Analytics now generally supports partition ing your data according to message attributes or attributes added through pipeline activities.	June 14, 2021
Reprocessing channnel messages	This update enables you to reprocess the channel data in the specified Amazon S3 objects.	December 15, 2020
Parquet schema	Amazon IoT Analytics data stores now support Parquet file format.	December 15, 2020
Monitoring with CloudWatch Events	Amazon IoT Analytics automatically publishes an event to Amazon CloudWatc h Events when a runtime error occurs during an Amazon Lambda activity.	December 15, 2020
Late data notification	You can use this feature to receive notifications through Amazon CloudWatch Events when late data arrives.	November 9, 2020
Region launch	Launched Amazon IoT Analytics in China (Beijing).	November 4, 2020

Earlier updates

The following table describes important changes to the *Amazon IoT Analytics User Guide* before November 4, 2020.

Change	Description	Date
Region launch	Launched Amazon IoT Analytics in the Asia Pacific (Sydney) Region.	July 16, 2020
Update	Reorganized the documenta tion.	May 07, 2020