

DataFlow Functions Telemetry Tutorial

Date published: 2021-04-06

Date modified: 2024-06-03

The Cloudera logo is displayed in a bold, orange, sans-serif font. The word "CLOUDERA" is written in all caps, with the letter "E" stylized as three horizontal bars.

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

| | |
|--|----------|
| Scope and goals..... | 4 |
| Terminology..... | 4 |
| Assets..... | 4 |
| Telemetry tutorial steps..... | 5 |
| Review the prerequisites..... | 5 |
| Register for 60-day trial..... | 5 |
| Develop and test a flow in Apache NiFi..... | 9 |
| 1. Install Apache NiFi..... | 9 |
| 2. Import the Telemetry Tutorial NiFi Flow..... | 10 |
| 3. Configure and start the Telemetry Tutorial NiFi Flow..... | 14 |
| 4. Test the Telemetry Tutorial NiFi Flow..... | 15 |
| 5. Download the Telemetry Tutorial NiFi Flow..... | 18 |
| Add the Telemetry Tutorial NiFi Flow as a function in | 20 |
| 1. Upload the Telemetry Tutorial NiFi Flow to the Catalog..... | 20 |
| 2. Download the Lambda Function binaries zip and upload to S3..... | 21 |
| 3. Create a Service Account..... | 22 |
| Run the function in serverless mode in AWS Lambda..... | 24 |
| 1. Create the function..... | 24 |
| 2. Test the function..... | 28 |
| 3. Create S3 trigger for the function..... | 31 |
| 4. Monitor the function..... | 33 |

Scope and goals

This tutorial walks you through the process of going from zero – no tenant and no data flow – to a serverless Apache NiFi flow on AWS Lambda in under 30 minutes without any Sales or technical assistance.

You can deploy a serverless NiFi flow using without the need to create a Data Lake and to perform the corresponding cloud prerequisite steps. Following this end-to-end workflow, you will:

1. Register for a 60-day trial for Trial which will provision a tenant in .
2. Implement a data distribution use case on a local development workstation using Apache NiFi.
3. Register the NiFi flow in Service as a function.
4. Deploy the function in serverless mode using AWS Lambda.

Terminology

This tutorial uses the following terms and concepts that you should be familiar with.

| Term | Definition |
|-------------|--|
| Apache NiFi | Low-code data ingestion tool built to automate the flow of data between systems |
| Flow | Represents data flow logic that was developed using Apache NiFi |
| Processor | Component in the data flow that perform work combining data routing, transformation, and mediation between systems |
| | Cloudera's data management platform in the cloud |
| Service | data service that enables self-serve deployments of Apache NiFi |
| function | Flow that is uploaded into the Catalog that can be run in serverless mode by serverless cloud provider services |
| Catalog | Inventory of flow definitions from which you can initiate new deployments |
| AWS Lambda | Serverless, event-driven compute service that lets you run a function without provisioning or managing servers |

Assets

This tutorial relies on the following assets:

| Asset | Description |
|---|---|
| Telemetry Tutorial NiFi Flow | The NiFi flow that implements the requirements of the use case described in this guide and that is run in serverless mode on AWS Lambda |
| Cloudera DataFlow Function Definition | The function definition be used with the AWS CLI command to create the function on AWS Lambda |
| trust-policy.json | Trust policy used with the AWS CLI command to create the IAM role required to build the function in AWS Lambda |
| Sample Trigger Event | Sample trigger event used to test the function in the AWS Lambda |
| Sample Telemetry File | Raw sample telemetry file to be uploaded into S3 to test the NiFi Flow and Lambda function |
| Second Sample Telemetry File | Another raw sample telemetry file to be uploaded into S3 to test the function in AWS Lambda with the S3 trigger enabled |

Telemetry tutorial steps

Review the prerequisites

This section helps you to examine the list of actions you must perform before you start working on your function.

- AWS user account is required with access policies that has permissions to list and create buckets, roles, and Lambda functions.
- Access to AWS console is required.
- Access key is required for the AWS user account to use AWS CLI.
- AWS CLI client needs to be installed and configured to use the access key.
- AWS bucket is needed for the telemetry events files:
 1. Create two folders in the bucket: truck-telemetry-raw and truck-telemetry-processed.
 2. Download the [sample telemetry file](#) and upload it to the truck-telemetry-raw folder. This file will be used during the tutorial to run tests.

Register for 60-day trial

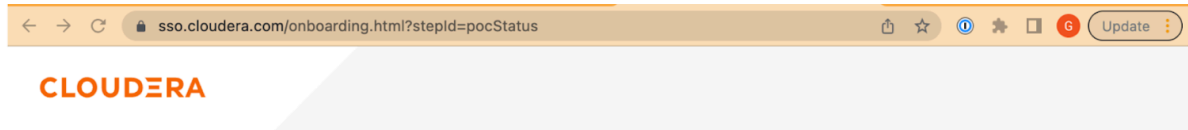
This section provides information on the free trial.

Procedure

1. Log into your account.

If you do not have an account on cloudera.com, register for a new account [here](#).

2. Once you have a cloudera.com account created and you logged into the account, register for the [60-day trial](#).
 - If your company has been registered for a trial account, by the end of the registration you will have a newly provisioned tenant on .



Almost there



Generating your License. Creating your CDP
Tenant. Assigning Roles & Permissions.



- If your company has not been registered for a trial account, shortly after completing the registration, the Sales Team will reach out to you to complete your trial registration.



Thank you for signing up for the CDP Public Cloud trial

We will contact you shortly to provide access to CDP in
your existing environment or to set up a trial.

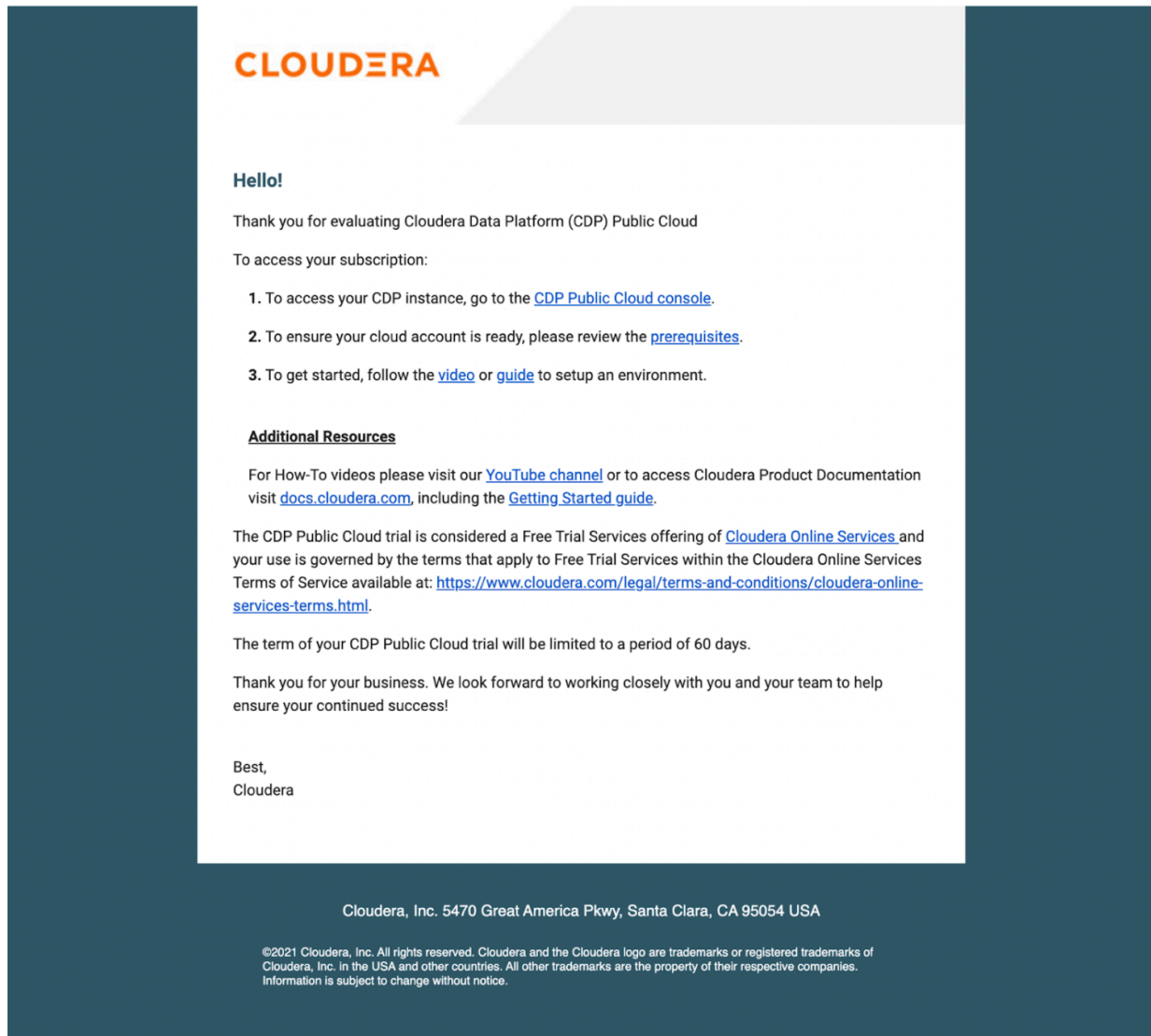
In the meantime, sign up for free training on CDP via an
[on-demand course](#).

Try CDP Public Cloud for free for 60-days using your own data and workloads

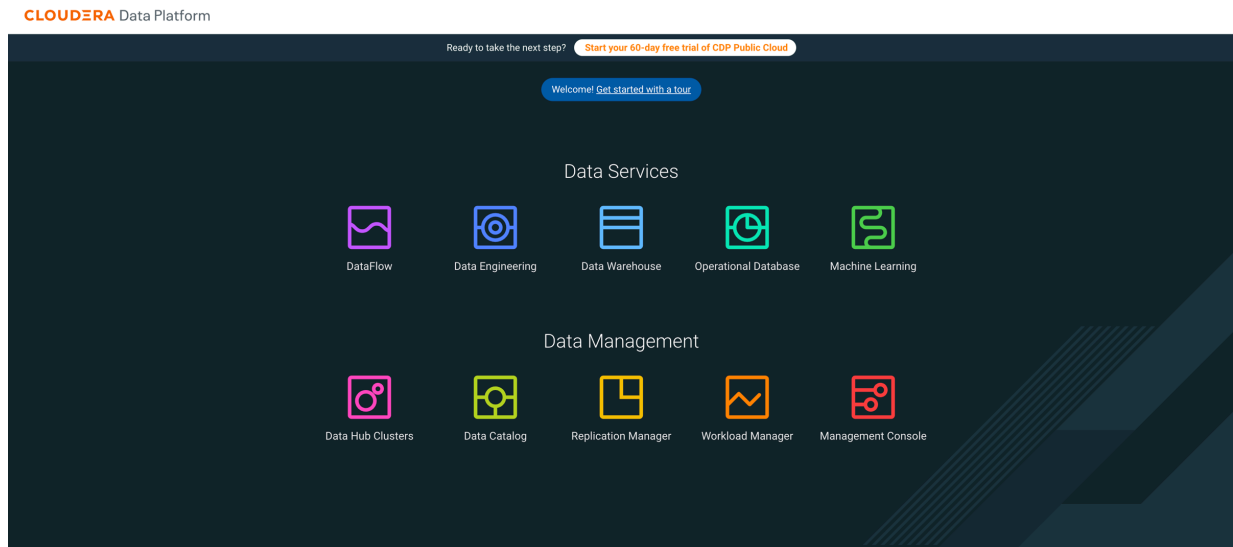
Experience:

- A powerful analytics platform that simplifies the management of hybrid and multi-cloud data
- Easy-to-use analytics services such as Data Warehouse and Machine Learning that scale to petabytes of data and thousands of users
- Centralized security and governance for pain-free portability across clouds

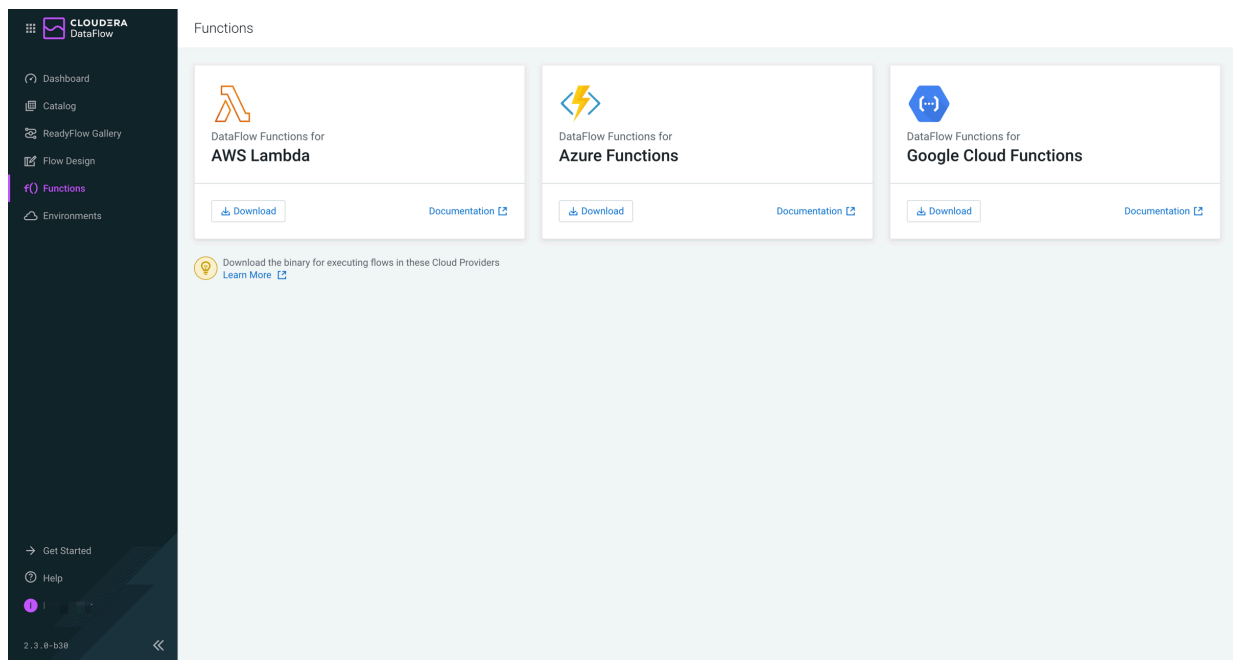
3. After completing the registration, you will receive an email with instructions on how to log into your newly provisioned tenant on . Disregard steps 2 and 3 as Function does not require any of the prerequisites to create an environment or a DataLake.



4. After logging into the console with the credentials that you used to register for the trial, select the service from the homepage.

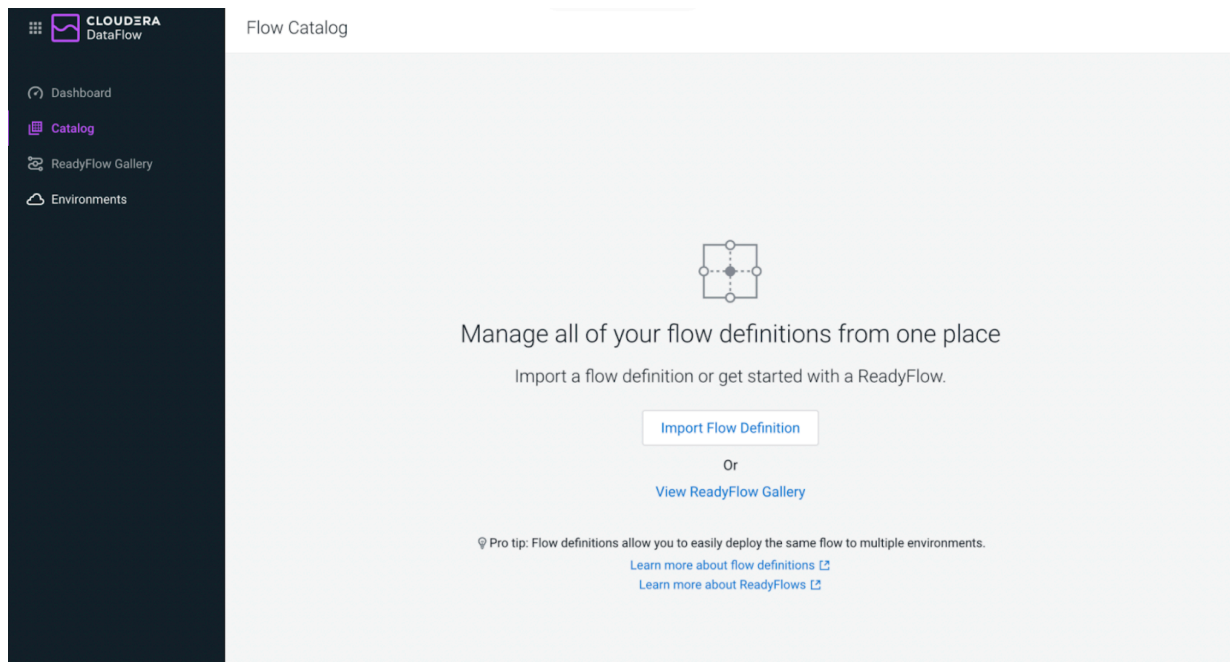


5. If you click Functions in the left navigation pane, you can see the options for cloud provider functions.



This is from where you can Download on the AWS Lambda tile to download the function binaries for AWS Lambda.

6. If you click Catalog in the left navigation pane, you can see that it is empty. You can develop an Apache NiFi flow locally, test it, and then come back to the Catalog to upload it as a function.



Develop and test a flow in Apache NiFi

This section walks you through designing your function by developing a data flow using NiFi on your local development workstation. Once the NiFi flow has been developed and tested, you will deploy it as a function in serverless mode using AWS Lambda.

1. Install Apache NiFi

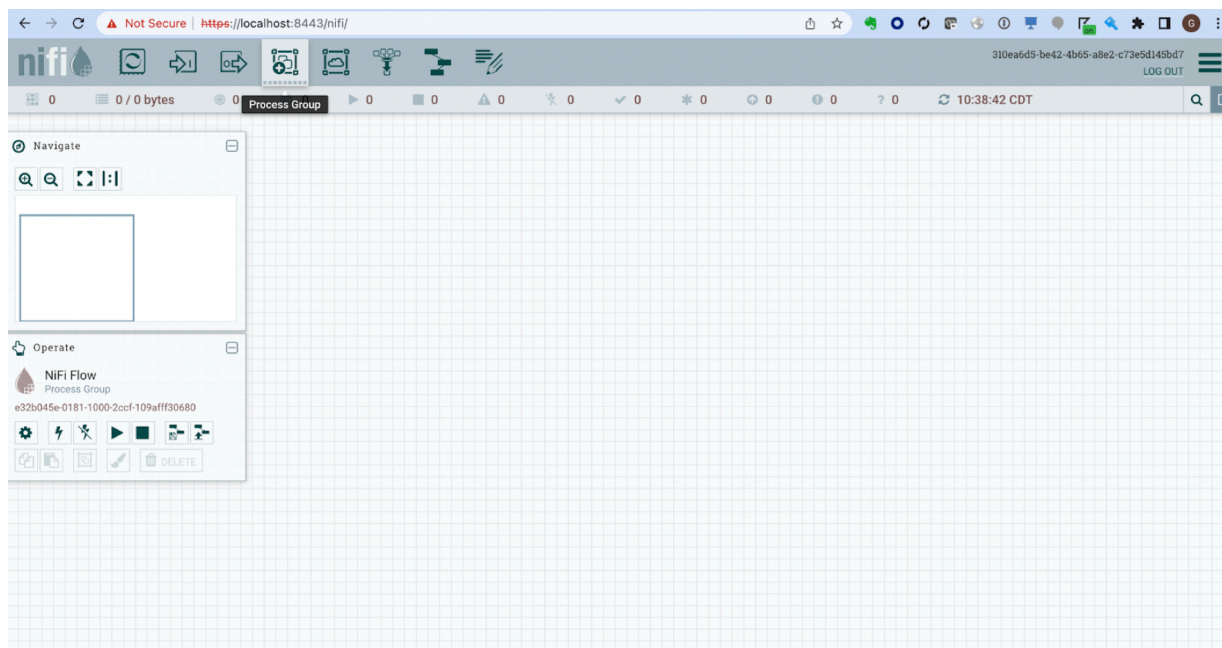
Download and install Apache NiFi 1.16.X on your local development workstation.

Procedure

1. Follow the [instructions](#) to download and install the latest version of Apache NiFi.

2. Log into NiFi with the generated credentials.

You can see the NiFi flow designer canvas.



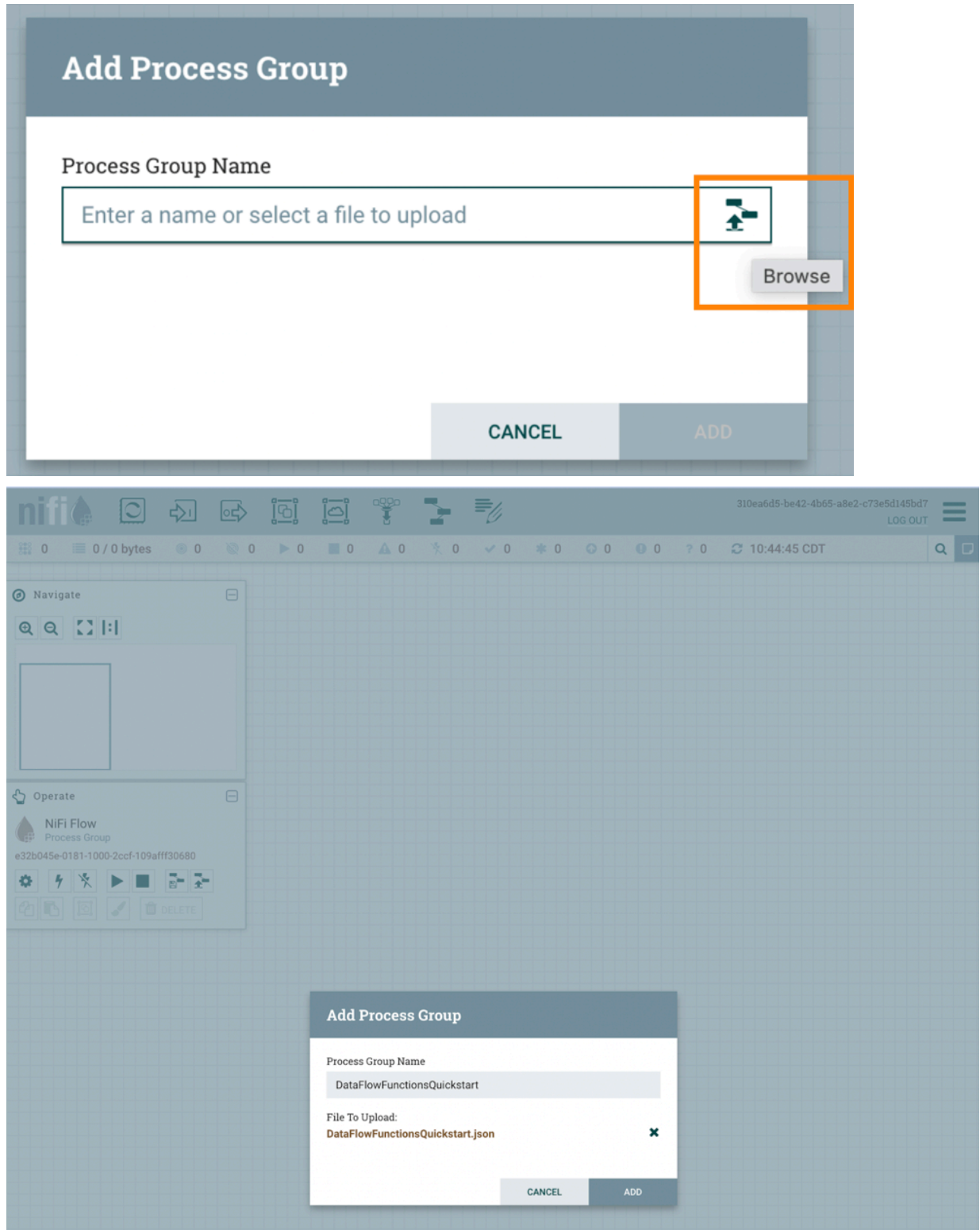
2. Import the Telemetry Tutorial NiFi Flow

Download and add the example data flow to Apache NiFi. This example flow implements the requirements of the use case described in this quickstart.

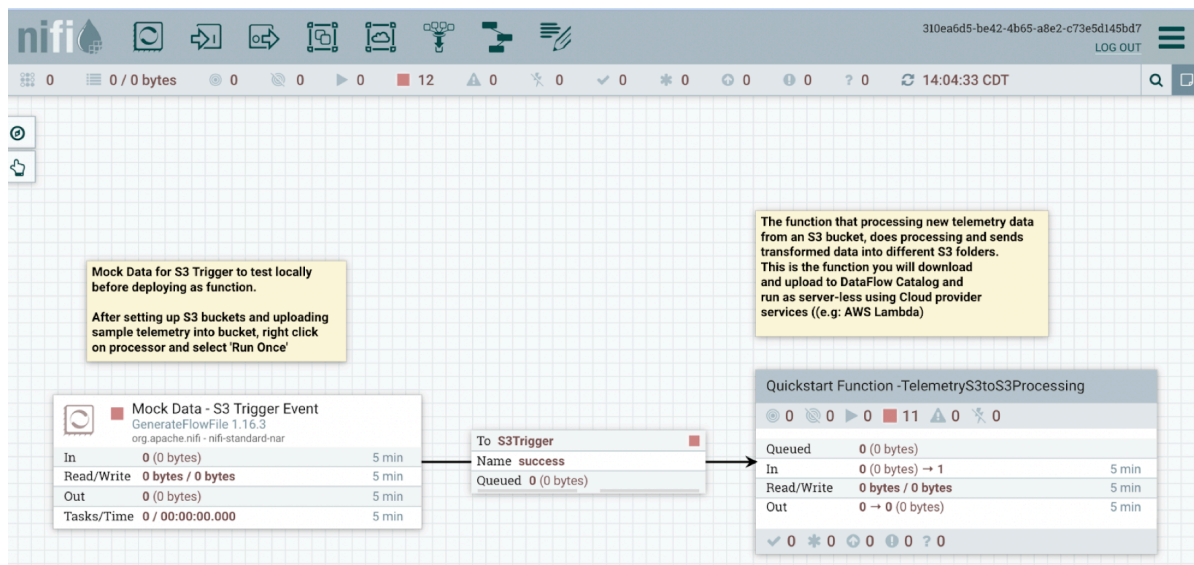
Procedure

1. Download the [Telemetry Tutorial NiFi Flow](#) to your local machine.
2. Drag a Process Group (fourth icon from the left) onto the canvas.

- Click the browse link, select the Telemetry Tutorial NiFi Flow you have downloaded, and click Add.



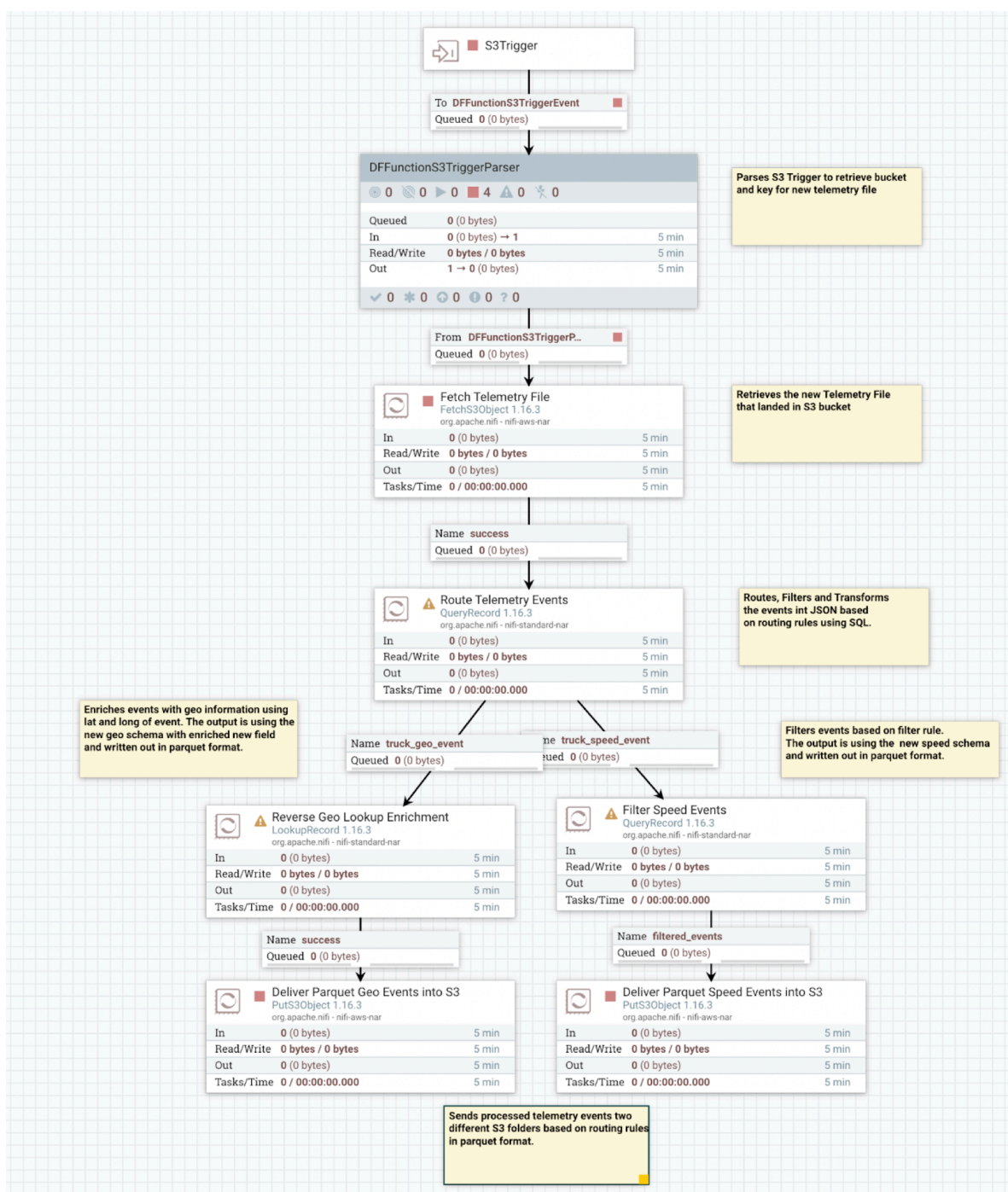
4. Explore the flow that you have uploaded to the canvas.
 - a) Double-click the DataFlowFunctionsQuickstart process group.



The Mock Data - S3 Trigger Event processor will generate a test Lambda S3 trigger event. You can use this processor to test the function locally before deploying it on AWS Lambda.

- b) Double-click the Quickstart Function -TelemetryS3toS3Processing process group that contains the flow for processing the telemetry data.

Note the details of this flow:



- S3Trigger is an input port that the AWS Lambda handler sends the S3 trigger event to. All functions need to start with the input port for the corresponding trigger that will be used.
- DFunctionS3TriggerParser parses the json trigger event to extract the bucket, key and region and store in flow attributes so it can be easily used by the downstream processors.
- Fetch Telemetry File fetches the new telemetry file that landed in S3 and stores it as a flowfile.
- Route Telemetry Events routes sensor events in the file to different paths based on the eventSource value in the event. Events with eventSource of 'truck_geo_event' will be routed to one path and events with value 'truck_speed_event' be routed to another path.
- For the events sent to the truck_geo_event path, the event will be enriched using custom groovy code that looks up the geo address based on the lat and long values in the event. The enriched events are then converted to parquet format using the supplied schema and the file is stored in a new s3 folder.

- For the events sent to the truck_speed_event path, the events are filtered for any events with speed > 40 and then converted into parquet format and stored in a different s3 folder.

3. Configure and start the Telemetry Tutorial NiFi Flow

Apache NiFi uses the concept of parameter context to store properties of the flow that need to change when deploying to different environments. The parameter context for the function in this tutorial is called DataFlowFunctionsQuickstart.

Procedure

- Configure the parameter context.
 - Right-click the canvas and select Parameters.
 - Update the following four parameter values based on your environment.

Update Parameter Context

SETTINGS PARAMETERS INHERITANCE

| Name | Value |
|--------------------------|--|
| aws_access_key_id | Sensitive value set |
| aws_access_key_password | Sensitive value set |
| filter_rule_speed_event | SELECT * FROM FLOWFILE where sp... |
| routing_rule_geo_event | SELECT * FROM FLOWFILE where ev... |
| routing_rule_speed_event | SELECT * FROM FLOWFILE where ev... |
| s3_bucket | dataflowfunctionsquickstart |
| s3_dest_path_geo_event | truck-telemetry-processed/truck-geo... |
| s3_dest_path_speed_event | truck-telemetry-processed/truck-spe... |
| s3_region | us-west-2 |
| telemetry_geo_schema | {...} |
| telemetry_raw_schema | {...} |
| telemetry_speed_schema | {...} |

Parameter: aws_access_key_id

Referencing Components

- Quickstart Function - TelemetryS3toS3Processing (3)
 - Referencing Processors
 - Deliver Parquet Geo Events into S3
 - Deliver Parquet Speed Events into S3
 - Fetch Telemetry File

Referencing Controller Services

None

Unauthorized Referencing Components

None

CANCEL APPLY

- aws_access_key_id - AWS access key id to fetch and write objects to S3 bucket
- aws_access_key_password - AWS secret access key secret to fetch and write objects to S3
- s3_bucket - S3 bucket you created where telemetry data will be processed
- s3_region - the region where you created the bucket

2. Start the NiFi flow.

a) Start Controller Services.

Controller Services are shared services that can be used by processors and other services to utilize for configuration or task execution

b) Within the process group called Quickstart Function -TelemetryS3toS3Processing, select the canvas to bring mouse focus to it, right-click the canvas, select Configure Controller Services .

Quickstart Function -TelemetryS3toS3Processing Configuration

| GENERAL | | CONTROLLER SERVICES | | | | |
|--|-------------------------------|--|--|---|---|---|
| Name | Type | Bundle | State | Scope | | |
|  Enrich-ReverseGeoCodeLookupService | ScriptedLookupService 1.16.3 | org.apache.nifi-nifi-scripting-nar |  Disabled | Quickstart Function -TelemetryS3toS3Pr... |  |  |
|  GeoTelemetryParquetWriter | ParquetRecordSetWriter 1.16.3 | org.apache.nifi-nifi-parquet-nar |  Disabled | Quickstart Function -TelemetryS3toS3Pr... |  |  |
|  RawTelemetryJsonReader.Json | JsonTreeReader 1.16.3 | org.apache.nifi-nifi-record-serialization... |  Disabled | Quickstart Function -TelemetryS3toS3Pr... |  |  |
|  RawTelemetryJsonWriter.Json | JsonRecordSetWriter 1.16.3 | org.apache.nifi-nifi-record-serialization... |  Disabled | Quickstart Function -TelemetryS3toS3Pr... |  |  |
|  SpeedTelemetryParquetWriter | ParquetRecordSetWriter 1.16.3 | org.apache.nifi-nifi-parquet-nar |  Disabled | Quickstart Function -TelemetryS3toS3Pr... |  |  |

There are five controller services defined for this function that are responsible for parsing the incoming JSON telemetry data, writing the data in Parquet format and doing the geo address lookup.

c) Click the bolt icon next to each service to enable it.

d) Close the configuration dialog.

e) Right-click the canvas and select Start to start the process group called Quickstart Function - TelemetryS3toS3Processing.

You should see that each processor have a green play button which indicates that all the processors are started and ready to receive data.

4. Test the Telemetry Tutorial NiFi Flow

With the flow configured and started, you can test it locally with a sample trigger event before deploying it as a serverless function on AWS Lambda.

Procedure

1. Configure the test S3 trigger event.

a) From within the process group Quickstart Function -TelemetryS3toS3Processing, right-click the canvas and select Leave Group to go to the parent processor group.

b) Right-click the Mock Data - S3 Trigger Event processor group and select Configure.

c) Click the Properties tab and edit the Custom Text property value.

This property value represents a mock trigger event that Lambda would create when a new file called sampleTelemetryRawData.txt is added to the bucket folder.



Note: This is the file is that you uploaded to <<your_bucket>>/truck-telemetry-raw in the prerequisite step.

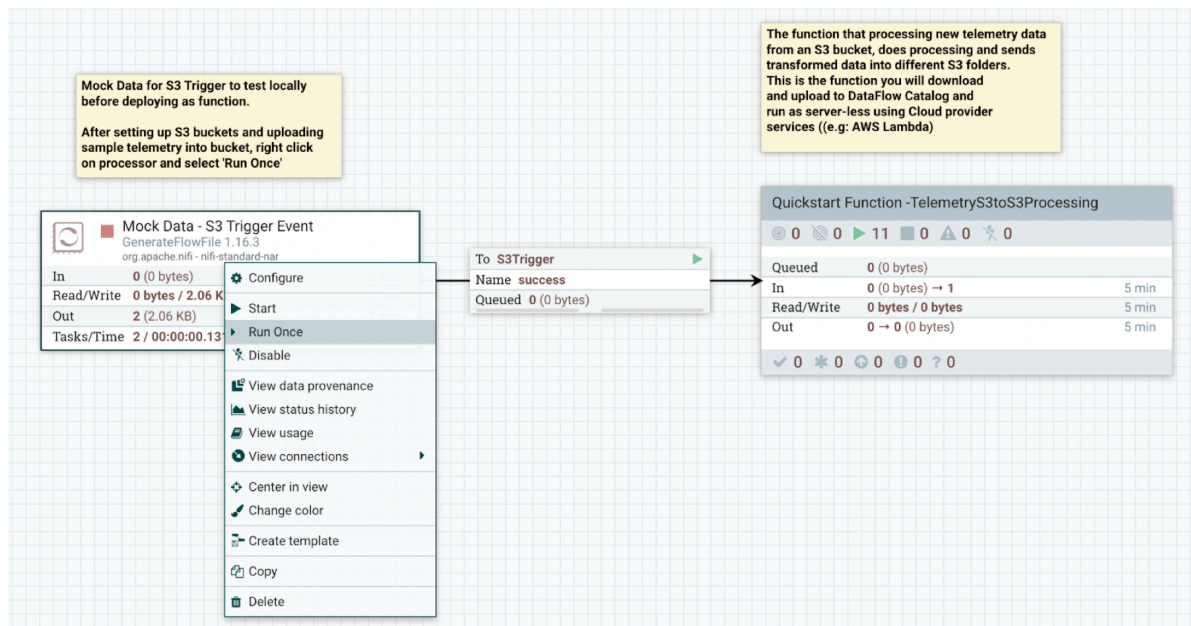
d) Update the bucket.name JSON field with the name of the bucket you created earlier.

e) Update the awsRegion JSON field to match the region in which you earlier created the bucket.

f) Click OK and then Apply to save the change.

2. Run the NiFi flow with the test trigger event.

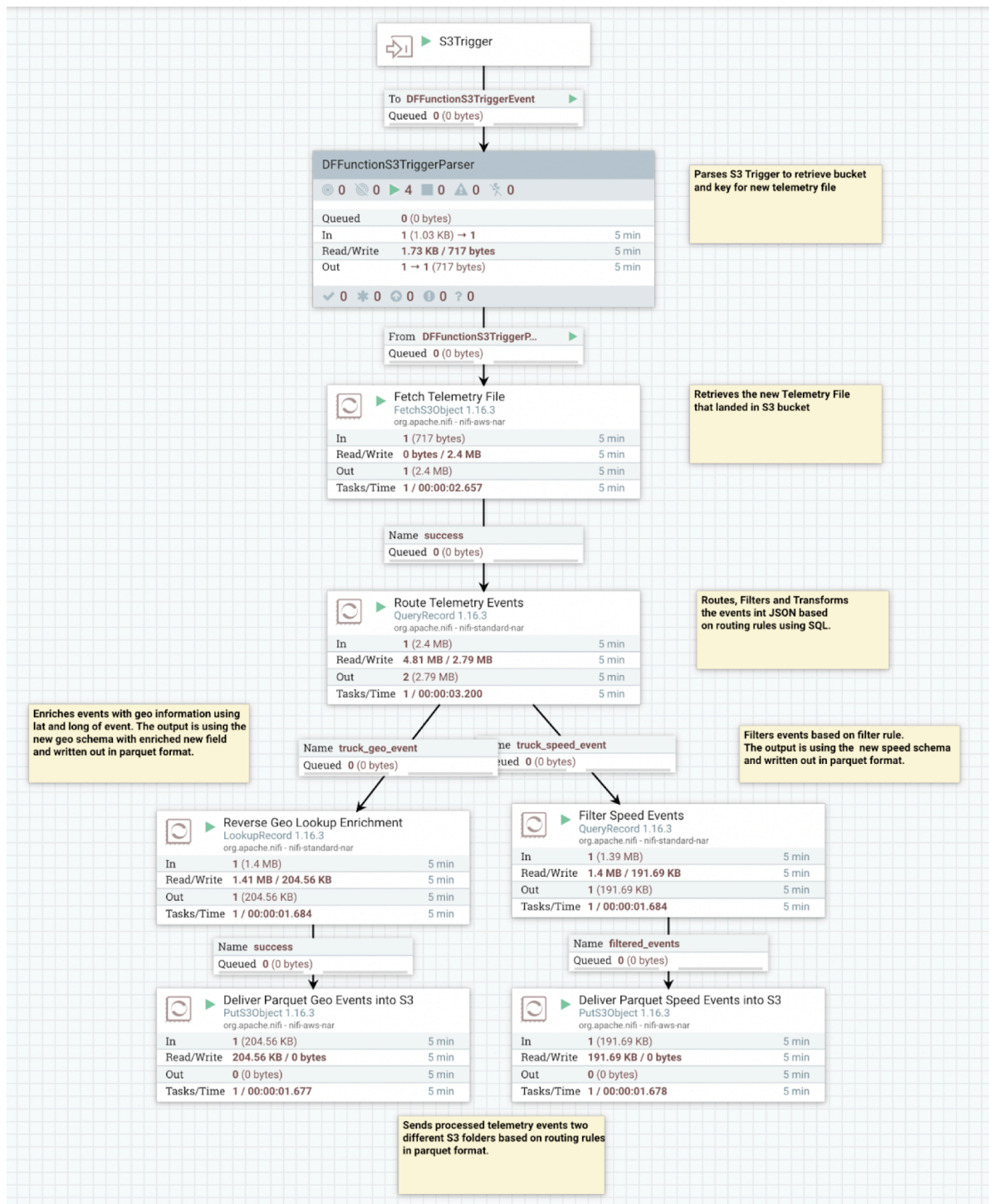
- a) Right-click the 'Mock Data - S3 Trigger Event' processor group and select 'Run Once'.



This will create a mock trigger event flow file.

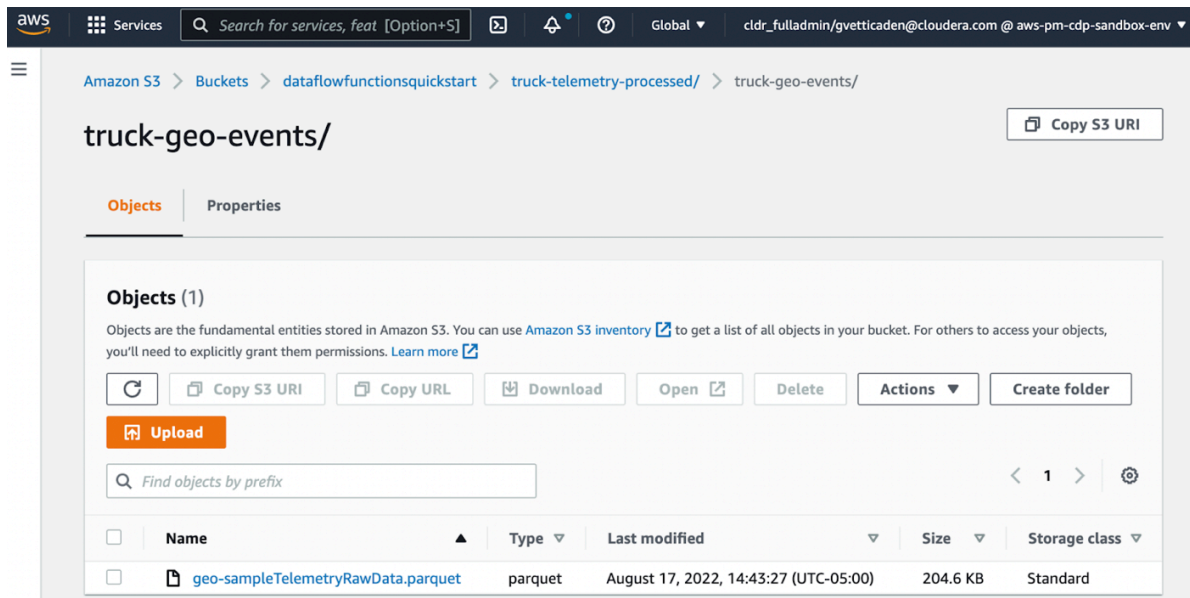
- b) If you double-click the process group Quickstart Function -TelemetryS3toS3Processing, you can see the flow file processed by the processors.

The In metric should show '1' across all processors.



3. Validate the output of the test.

- The output of the test should be a parquet file that contains telemetry data that is filtered and enriched based on the above requirements stored in the following s3 folders: <<your_bucket>>/processed/truck-geo-events and <<your_bucket>>/processed/truck-speed-events.



- If the Parquet file is in each of these folders, the local test has completed successfully and the function works as expected and now can run on AWS Lambda.

5. Download the Telemetry Tutorial NiFi Flow

Download the Apache NiFi flow so that you can upload it into the Catalog and run it in serverless mode.

Procedure

- Right-click the 'Quickstart Function -TelemetryS3toS3Processing' process group.

- 2. Select Download Flow Definition and Without external services.

The function that processing new telemetry data from an S3 bucket, does processing and sends transformed data into different S3 folders. This is the function you will download and upload to DataFlow Catalog and run as server-less using Cloud provider services ((e.g: AWS Lambda)

ss
bytes)

Quickstart Function -TelemetryS3toS3Processing

0 0 0 11 0 0

Queued0 (0 bytes)

In0 (0 bytes) → 15 min

Read/Write0 bytes / 0 bytes5 min

Out0 (0 bytes)5 min

0

Configure

Parameters

Variables

Enter group

Start

Stop

Enable

Disable

Enable all controller services

Disable all controller services

View status history

View connections

Center in view

Download flow definition

Create template

Copy

Empty all queues

Delete

Without external services

With external services

19

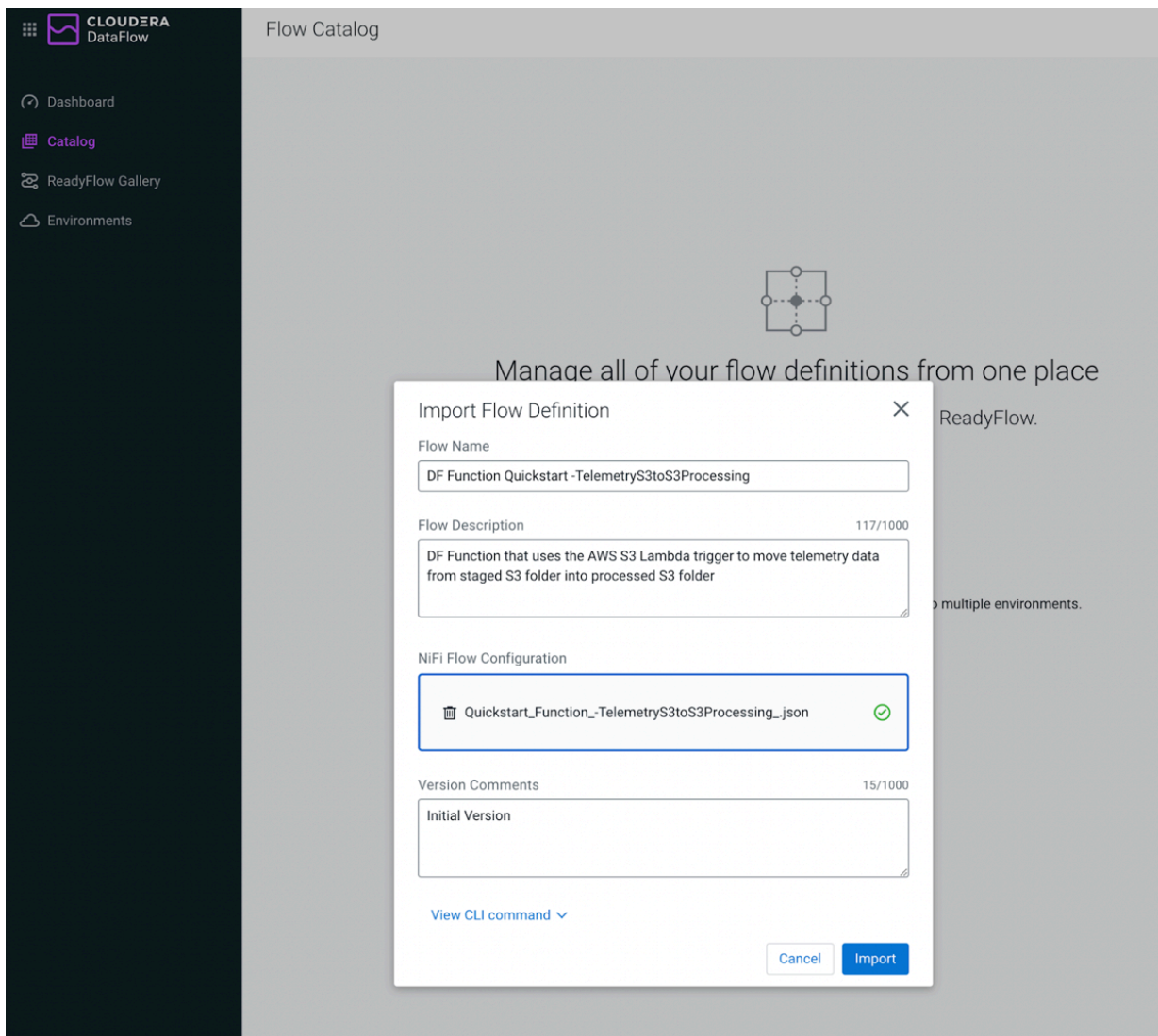
Add the Telemetry Tutorial NiFi Flow as a function in

Before you can run the Apache NiFi flow in serverless mode on AWS Lambda, you must register it in service and download the Lambda DF function handler libraries.

1. Upload the Telemetry Tutorial NiFi Flow to the Catalog

Procedure

1. After logging into the [console](#) with the credentials that you used to register for the trial, select the service on the homepage.
2. Click Catalog in the left navigation pane to display the Flow Catalog.
3. Click Import Flow Definition.
4. Provide a name and a description and upload the data flow you downloaded from your local NiFi instance.



The screenshot shows the Cloudera DataFlow console interface. On the left is a dark sidebar with navigation links: Dashboard, Catalog (highlighted), ReadyFlow Gallery, and Environments. The main area is titled 'Flow Catalog' and features a large graphic with the text 'Manage all of your flow definitions from one place' and 'ReadyFlow.' Below this, a modal dialog titled 'Import Flow Definition' is open. The dialog contains the following fields:

- Flow Name:** A text input field containing 'DF Function Quickstart -TelemetryS3toS3Processing'.
- Flow Description:** A text area with a character count of 117/1000, containing the text: 'DF Function that uses the AWS S3 Lambda trigger to move telemetry data from staged S3 folder into processed S3 folder'.
- NiFi Flow Configuration:** A section with a file upload icon and the text 'Quickstart_Function_-TelemetryS3toS3Processing_.json', followed by a green checkmark icon.
- Version Comments:** A text area with a character count of 15/1000, containing the text: 'Initial Version'.

At the bottom of the dialog, there is a link 'View CLI command' and two buttons: 'Cancel' and 'Import'.

5. Click Import.

6. Copy the CRN # for version 1 of the flow you uploaded.

Make sure to select the version's CRN in the orange box below. You will need it when configuring the function in AWS Lambda.

The screenshot shows the Cloudera DataFlow interface. On the left is a navigation menu with 'Catalog' selected. The main area is titled 'Flow Catalog' and contains a search bar and a list of flows. The flow 'DF Function Quickstart - TelemetryS3toS3Processing' is selected. On the right, the details for this flow are shown. The 'CRN #' is displayed as 'crn:cdp:df:us-west-1:78cbbfcd-e358-406c-bdaa-76afa5c87f79:flow:DF-Function-Quickstart...'. This CRN is highlighted with an orange rectangular box. Below the CRN, there is a table with columns 'Version' and 'Deployments'. Version 1 is listed with 0 deployments. At the bottom of the details panel, the CRN is repeated and highlighted with an orange box.

2. Download the Lambda Function binaries zip and upload to S3

To be able to run the NiFi flow in AWS Lambda, you need the function handler libraries.

Procedure

1. Click Functions in the left navigation pane and download the Function binaries for AWS Lambda.
AWS Lambda will use these binaries to run the NiFi flow.

The screenshot shows the Cloudera DataFlow 'Functions' page. The left navigation menu has 'Functions' selected. The main area displays three cards for different cloud providers: 'DataFlow Functions for AWS Lambda', 'DataFlow Functions for Azure Functions', and 'DataFlow Functions for Google Cloud Functions'. Each card has a 'Download' button and a 'Documentation' link. Below the cards, there is a note: 'Download the binary for executing flows in these Cloud Providers' with a 'Learn More' link.

2. Upload this binary to an S3 bucket that you will later reference when creating the function in AWS Lambda. The S3 bucket needs to be in the same region as where the Lambda is being created/deployed.
3. Copy the S3 URI for later use.

Amazon S3 > Buckets > dataflowfunctionsquickstart > libs/ > naaf-aws-lambda-1.0.0-SNAPSHOT-bin.zip

naaf-aws-lambda-1.0.0-SNAPSHOT-bin.zip Info

Copy S3 URI
Download
Open
Object actions ▼

Properties
Permissions
Versions

Object overview

| | |
|--|---|
| Owner cloud-aws-pm-cdp-sandbox-env | S3 URI s3://dataflowfunctionsquickstart/libs/naaf-aws-lambda-1.0.0-SNAPSHOT-bin.zip |
| AWS Region US West (Oregon) us-west-2 | Amazon Resource Name (ARN) arn:aws:s3:::dataflowfunctionsquickstart/libs/naaf-aws-lambda-1.0.0-SNAPSHOT-bin.zip |
| Last modified August 17, 2022, 22:33:47 (UTC-05:00) | Entity tag (Etag) 39b2af2f20a945593fc0c71a26fbc02f-5 |
| Size 75.6 MB | Object URL https://dataflowfunctionsquickstart.s3.us-west-2.amazonaws.com/libs/naaf-aws-lambda-1.0.0-SNAPSHOT-bin.zip |
| Type zip | |
| Key libs/naaf-aws-lambda-1.0.0-SNAPSHOT-bin.zip | |

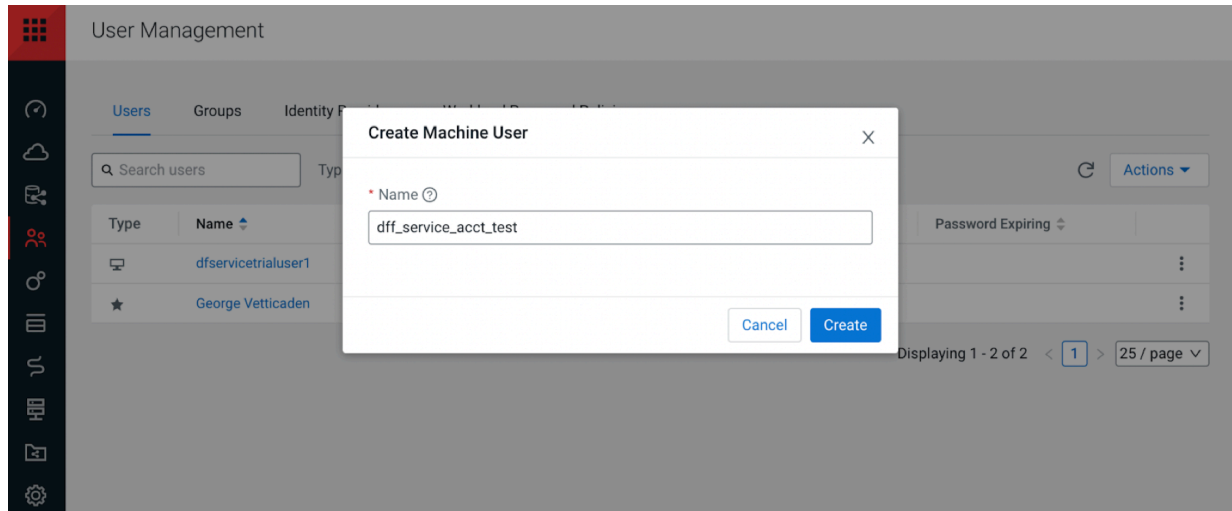
3. Create a Service Account

You need to create a service account for the AWS Lambda function to be able to retrieve the function from the Catalog where you uploaded it earlier.

Procedure

1. Go to the homepage, select Management Console User Management .

- Under Actions, select Create Machine User and provide a name for the service account.



- Click Roles Update Roles and select the DFFunctionMachineUser.

Update Roles

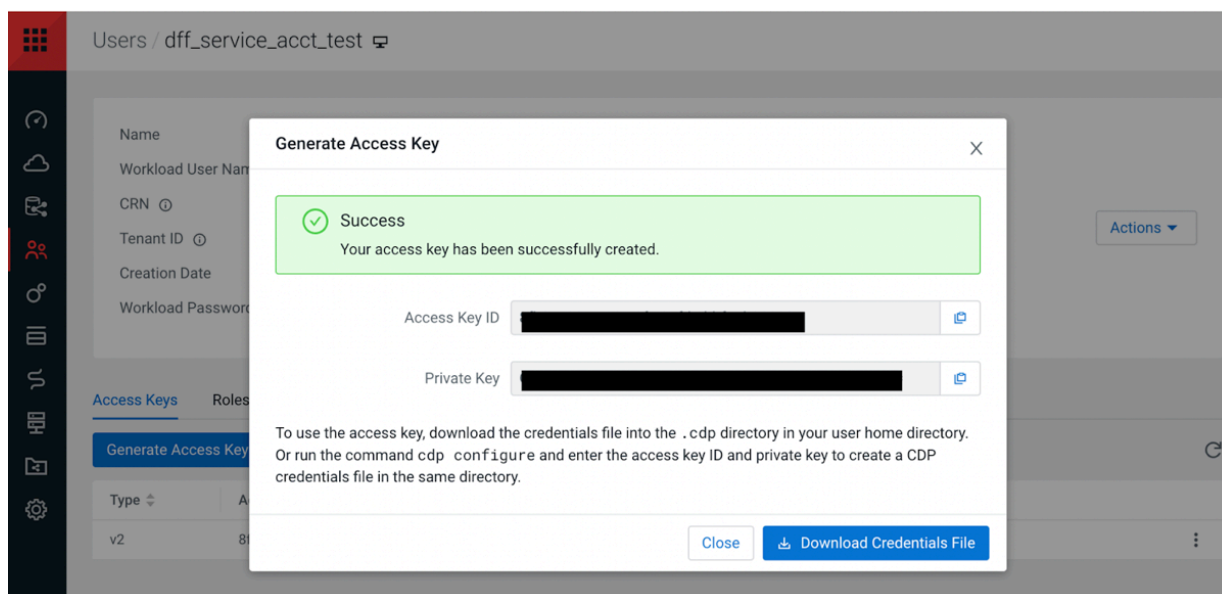
| <input type="checkbox"/> | Role | Description | Policies |
|-------------------------------------|---------------------------|--|--------------------------|
| <input type="checkbox"/> | ClassicClustersCreator | Grant permission to register classic clusters. | Policies |
| <input type="checkbox"/> | DFCatalogAdmin | Grants permission to perform all tasks on objects stored in the DataFlow Catalog. This includes importing and deleting flow definitions, as well as uploading new versions of existing flow definitions. | Policies |
| <input type="checkbox"/> | DFCatalogViewer | Grants permission to search and browse flow definitions stored in the DataFlow Catalog. | Policies |
| <input checked="" type="checkbox"/> | DFFunctionMachineUser | Grants permission to download a flow definition from the DataFlow Catalog and run it as a serverless function instance. | Policies |
| <input type="checkbox"/> | DataCatalogCspRuleManager | Grants permission to perform all task on CSP rules in Data Catalog. | Policies |
| <input type="checkbox"/> | DataCatalogCspRuleViewer | Grants permission to list and view CSP rules in Data Catalog. | Policies |
| <input type="checkbox"/> | EnvironmentAdmin | Deprecated - Grants permission to perform all task on environments, Data Lake and Data Hub clusters. | Policies |
| <input type="checkbox"/> | EnvironmentCreator | Grants permission to credential on Management Console for the specific account. | Policies |
| <input type="checkbox"/> | EnvironmentUser | Deprecated - Grants permission to perform read tasks on environments and Data Lake clusters and perform all tasks on Data Hub clusters. | Policies |
| <input type="checkbox"/> | IamUser | Grants permission to create access keys for the user, view assigned roles, and view all users in the account. | Policies |
| <input type="checkbox"/> | IamViewer | Grants permission to view access keys, view assigned roles and view all users in the account. | Policies |

Cancel

Update

- Click Update.
- Select Access Keys Generate Access Key .

6. Save the Access Key ID and Private Key, which will be used to configure the AWS Lambda function.



Run the function in serverless mode in AWS Lambda

Now that you have developed the NiFi flow and tested locally, registered it as `function` in `service`, you are ready to run the function in serverless mode using AWS Lambda. For this, you will need to create, configure, test and deploy the function in AWS Lambda.

1. Create the function

You can use the AWS CLI to create and configure the `function` in AWS Lambda.

Procedure

1. Create the AWS IAM Role required to create the lambda function.
 - a) When Lambda executes a function, it requires an execution role that grants the function permission to access AWS services and resources. Lambda assumes the role when the `function` is invoked. Assign the most limited permissions/policies for the function to execute.
 - b) Download the [trust-policy.json](#) file.
 - c) Using the below AWS CLI command, create a role called `NiFi_Function_Quickstart_Lambda_Role` that the Lambda service will assume.

The role will be attached to an AWS managed role that provides the limited permissions for the function to execute.

```
aws iam create-role --role-name NiFi_Function_Quickstart_Lambda_Role --assume-role-policy-document file://trust-policy.json
```

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole --role-name NiFi_Function_Quickstart_Lambda_Role
```

- d) These two commands will create the following IAM role. Copy and save the Role ARN which you will need to create the function.

[IAM](#) > [Roles](#) > [NiFi_Function_Quickstart_Lambda_Role](#)

NiFi_Function_Quickstart_Lambda_Role

Delete

Summary

Edit

Creation date
August 18, 2022, 22:37 (UTC-05:00)

ARN
[arn:aws:iam::381358652250:role/NiFi_Function_Quickstart_Lambda_Role](#)

Last activity
None

Maximum session duration
1 hour

Permissions

Trust relationships

Tags

Access Advisor

Revoke sessions

Permissions policies (1)

You can attach up to 10 managed policies.



Simulate

Remove

Add permissions ▼

Filter policies by property or policy name and press enter

< 1 > ⚙

| <input type="checkbox"/> | Policy name ↗ | Type | Description |
|--------------------------|---|-------------|--|
| <input type="checkbox"/> | AWSLambdaBasicExecutionRole | AWS managed | Provides write permissions to CloudWatch Logs. |

2. Create the function in Lambda.

a) Download the [Function Definition](#) JSON file.

- This file has the full definition required to create the function including function, code, environment variable and security configuration.
- The environment variables in this file contain the info for Lambda to fetch the function definition from the Catalog as well as the function's application parameters.

b) Update the following properties in the definition file:

- `DF_ACCESS_KEY` – The access key created for the service account
- `DF_PRIVATE_KEY` – The private key created for the service account
- `FLOW_CRN` – The CRN value you copied from the Catalog page after uploading the function
- `aws_access_key_id` – The AWS access key that has permissions to access (read/write) the S3 bucket you created in the prerequisite section
- `aws_access_key_password` – The AWS access key password that has permissions to access (read/write) the S3 bucket you created in the prerequisite section
- `s3_bucket` – The name of the bucket you created
- `s3_region` – The bucket's region
- `S3Bucket` – The name of the bucket that you uploaded the binaries ZIP file that you downloaded from the page
- `S3Key` – The key to the binaries ZIP file in S3.

For example, if you uploaded the ZIP to S3 with this URI `s3://dataflowfunctionsquickstart/libs/naaf-aws-lambda-1.0.0-SNAPSHOT-bin.zip`, the key would be `libs/naaf-aws-lambda-1.0.0-SNAPSHOT-bin.zip`

- `Role` – The ARN of the role created in the previous step



Note: In the definition file, there is a parameter named `NEXUS_URL`. This parameter defines the location from where NARs will be downloaded when your function is instantiated.

- If you have designed your NiFi flow in a Cloudera Flow Management environment, you need to remove the parameter from the definition file.
- If you have designed your NiFi flow using an Apache NiFi environment as instructed before, you should keep this parameter.

c) Run the following command to create a function called `NiFi_Function_Quickstart` (if the `FunctionName` property was not modified):

```
aws lambda create-function --cli-input-json file://NiFi_Function_Quickstart-definition.json
```

3. View the Lambda function in AWS Console.

- a) On the AWS console, navigate to the Lambda service and click the function called NiFi_Function_Quickstart (if the FunctionName property was not modified).

You can see the following under the Code tab of the function:

The screenshot displays the AWS Lambda console for the function **NiFi_Function_Quickstart**. The interface includes a breadcrumb trail: **Lambda > Functions > NiFi_Function_Quickstart**. At the top right, there are buttons for **Throttle**, **Copy ARN**, and **Actions**. The main section is titled **Function overview** and contains a visual representation of the function with a trigger icon, the function name, and a layers icon labeled **Layers (0)**. Below this, there are buttons for **+ Add trigger** and **+ Add destination**. To the right, a sidebar provides details: **Description** (NiFi Function that uses the AWS S3 Lambda trigger to move telemetry data from landing zone S3 folder into processed S3 folder), **Last modified** (5 minutes ago), **Function ARN** (arn:aws:lambda:us-west-2:381358652250:function:NiFi_Function_Quickstart), and **Function URL** (Info). Below the overview, a tabbed interface shows **Code**, **Test**, **Monitor**, **Configuration**, **Aliases**, and **Versions**. The **Code** tab is active, showing a message: "The deployment package of your Lambda function 'NiFi_Function_Quickstart' is too large to enable inline code editing. However, you can still invoke your function." Below this, the **Code properties** section lists **Package size** (75.6 MB), **SHA256 hash** (tt6z17+ifXkG+6p2vns93KT6IEGckBRfMtapOdOwsk=), and **Last modified** (August 18, 2022 at 11:44 PM CDT). The **Runtime settings** section shows **Runtime** (Java 8 on Amazon Linux 2), **Handler** (com.cloudera.naaf.aws.lambda.StatelessNiFiFunctionHandler:handleRequest), and **Architecture** (arm64). Buttons for **Upload from** and **Edit** are also visible.


- b) If you click the Configuration tab, you can see all the configured parameters required to run the function under Environment variables.

Lambda > Functions > NiFi_Function_Quickstart

NiFi_Function_Quickstart

Throttle Copy ARN Actions

▼ Function overview [Info](#)


NiFi_Function_Quickstart

Layers (0)

+ Add trigger

+ Add destination

Description
NiFi Function that uses the AWS S3 Lambda trigger to move telemetry data from landing zone S3 folder into processed S3 folder

Last modified
12 minutes ago

Function ARN
[arn:aws:lambda:us-west-2:381358652250:function:NiFi_Function_Quickstart](#)

Function URL [Info](#)

Code Test Monitor **Configuration** Aliases Versions

General configuration

Triggers

Permissions

Destinations

Function URL

Environment variables

Tags

VPC

Monitoring and operations tools

Concurrency

Asynchronous invocation

Code signing

Database proxies

File systems

State machines

Environment variables (16) [Edit](#)

The environment variables below are encrypted at rest with the default Lambda service key.

| Key | Value |
|--------------------------|---|
| DF_ACCESS_KEY | [REDACTED] |
| DF_PRIVATE_KEY | [REDACTED] |
| FLOW_CRN | crn:cdp:df:us-west-1:78cbbfcd-e358-406c-bdaa-76afa5c87f79:flow:DF-Function-Quickstart--TelemetryS3toS3Processing/v.1 |
| NEXUS_URL | https://maven-central.storage-download.googleapis.com/maven2 |
| aws_access_key_id | [REDACTED] |
| aws_access_key_password | [REDACTED] |
| filter_rule_speed_event | SELECT * FROM FLOWFILE where speed > 40 |
| routing_rule_geo_event | SELECT * FROM FLOWFILE where eventSource = 'truck_geo_event' |
| routing_rule_speed_event | SELECT * FROM FLOWFILE where eventSource = 'truck_speed_event' |
| s3_bucket | dataflowfunctionsquickstart2 |
| s3_dest_path_geo_event | truck-telemetry-processed/truck-geo-events |
| s3_dest_path_speed_event | truck-telemetry-processed/truck-speed-events |
| s3_region | us-west-2 |
| telemetry_geo_schema | { "type": "record", "namespace": "cloudera.cdf.csp.schema.refapp.trucking", "name": "TruckGeoEventEnriched", "fields": [{"name": "eventTime", "type": "string"}, {"name": "eventTimeLong", "type": "long", "default": 0}, {"name": "eventSource", "type": "string"}, {"name": "truckId", "type": "int"}, {"name": "driverId", "type": "int"}, {"name": "driverName", "type": "string"}, {"name": "routeId", "type": "int"}, {"name": "route", "type": "string"}, {"name": "eventType", "type": "string"}, {"name": "latitude", "type": "double"}, {"name": "longitude", "type": "double"}, {"name": "correlationId", "type": "long"}, {"name": "geoAddress", "type": "string", "default": "None"}]} |
| telemetry_raw_schema | { "type": "record", "namespace": "cloudera.cdf.csp.schema.refapp.trucking", "name": "TruckEvent", "fields": [{"name": "eventTime", "type": "string"}, {"name": "eventTimeLong", "type": "long", "default": 0}, {"name": "eventSource", "type": "string"}, {"name": "truckId", "type": "int"}, {"name": "driverId", "type": "int"}, {"name": "driverName", "type": "string"}, {"name": "routeId", "type": "int"}, {"name": "route", "type": "string"}, {"name": "eventType", "type": "string"}, {"name": "latitude", "type": "double"}, {"name": "longitude", "type": "double"}, {"name": "correlationId", "type": "long"}, {"name": "speed", "type": "int", "default": 0}]} } |
| telemetry_speed_schema | { "type": "record", "namespace": "cloudera.cdf.csp.schema.refapp.trucking", "name": "TruckSpeedEventEnriched", "fields": [{"name": "eventTime", "type": "string"}, {"name": "eventTimeLong", "type": "long", "default": 0}, {"name": "eventSource", "type": "string"}, {"name": "truckId", "type": "int"}, {"name": "driverId", "type": "int"}, {"name": "driverName", "type": "string"}, {"name": "routeId", "type": "int"}, {"name": "route", "type": "string"}, {"name": "speed", "type": "int"}]} |

2. Test the function

With the function created and fully configured, you can test the function with a test trigger event before configuring the real trigger in the Lambda service.

Procedure

1. Create a Test Event.

Click the Test tab, select Create New Event, and configure the following:

- Provide a name for the test: NiFi_Function_Quickstart.
- Copy the contents from the [sampleTriggerEvent](#) and paste it into the Event JSON field.
- Replace the bucket name with the one you created.
- Click Save.

Test event Save Test

To invoke your function without saving an event, configure the JSON event, then choose Test.

Test event action

☒ Create new event Edit saved event

Event name

MyEventName

Maximum of 25 characters consisting of letters, numbers, dots, hyphens and underscores.

Event sharing settings

☒ Private
This event is only available in the Lambda console and to the event creator. You can configure a total of 10. [Learn more](#)

☐ Shareable
This event is available to IAM users within the same account who have permissions to access and use shareable events. [Learn more](#)

Template - optional

hello-world

Event JSON Format JSON

```

1- {
2-   "Records": [
3-     {
4-       "eventVersion": "2.0",
5-       "eventSource": "aws:s3",
6-       "awsRegion": "us-west-2",
7-       "eventTime": "1970-01-01T00:00:00.000Z",
8-       "eventName": "ObjectCreated:Put",
9-       "userIdentity": {
10-        "principalId": "EXAMPLE"
11-      },
12-       "requestParameters": {
13-        "sourceIPAddress": "127.0.0.1"
14-      },
15-       "responseElements": {
16-        "x-amz-request-id": "EXAMPLE123456789",
17-        "x-amz-id-2": "EXAMPLE123/5678abcdefghijklmbdaisawesom/mnopqrstuvwxyzABCDEFGH"
18-      },
19-       "s3": {
20-        "s3SchemaVersion": "1.0",
21-        "configurationId": "testConfigRule",
22-        "bucket": {
23-          "name": "<<REPLACE>>",
24-          "ownerIdentity": {
25-            "principalId": "EXAMPLE"
26-          }

```

2. Execute the Test Event.

- Reset data for the test to run by deleting the folders under <<your_bucket>>/processed which was created after the test run on your local NiFi instance.
- Click Test.

The initial run of the test is a cold start which will take a few minutes because it requires additional binaries to be downloaded from Nexus. Subsequent runs should be faster.

If the run is successful, the logs should look like this:

DF_Function_Quickstart-TelemetryS3toS3Processing_3 [Throttle] [Copy ARN] [Actions]

Function overview Info

DF_Function_Quickstart-TelemetryS3toS3Processing_3
Layers (0)

+ Add trigger + Add destination

Description
Last modified: 3 hours ago
Function ARN: arn:aws:lambda:us-west-2:381358652250:function:DF_Function_Quickstart-TelemetryS3toS3Processing_3
Function URL: Info

Code | **Test** | Monitor | Configuration | Aliases | Versions

Execution result: succeeded (logs)

Details

The area below shows the last 4 KB of the execution log.

Summary

| | |
|---|--|
| Code SHA-256 tt6z17+iFXkG+6p2vns93KT6IEGckBRfMtapOdOwsk= | Request ID 22940085-0164-4d2b-861e-8d5d6f0c3a3e |
| Duration 77051.60 ms | Billed duration 77052 ms |
| Resources configured 2048 MB | Max memory used 1122 MB |

Log output

The section below shows the logging calls in your code. [Click here](#) to view the corresponding CloudWatch log group.

```
b38a-e8cac16bc060,size=731] into 1 FlowFiles
[Run DataFlow Quickstart Function -TelemetryS3toS3Processing ] INFO org.apache.nifi.processors.aws.s3.FetchS3Object - FetchS3Object[id=26f515bf-bbe1-341b-812f-f49cde2e949] Successfully retrieved S3 Object for StandardFlowFileRecord[uuid=74c6d718-6b69-41fa-8c19-712d9ce8d347,claim=org.apache.nifi.stateless.repository.ByteArrayContentRepository$ByteArrayContentClaim@18d0a835,offset=0,name=cfd0964-eaf0-48e2-b38a-e8cac16bc060,size=2519727] in 616 millis; routing to success
[Run DataFlow Quickstart Function -TelemetryS3toS3Processing ] INFO org.apache.nifi.processors.standard.QueryRecord - QueryRecord[id=650cbd6e-1224-34df-9c7e-e7d554a41577] Successfully queried StandardFlowFileRecord[uuid=74c6d718-6b69-41fa-8c19-712d9ce8d347,claim=org.apache.nifi.stateless.repository.ByteArrayContentRepository$ByteArrayContentClaim@18d0a835,offset=0,name=cfd0964-eaf0-48e2-b38a-e8cac16bc060,size=2519727] in 4576 millis
[Run DataFlow Quickstart Function -TelemetryS3toS3Processing ] INFO org.apache.nifi.processors.standard.QueryRecord - QueryRecord[id=5bcbe8d3-b97e-
```

Test event [Save] [Test]

- Validate that the processed Parquet files are under the following directories:

- <<your_bucket>>/processed/truck-geo-events
- <<your_bucket>>/processed/truck-speed-events

Amazon S3 > Buckets > dataflowfunctionsquickstart > truck-telemetry-processed/ > truck-geo-events/

truck-geo-events/

[Copy S3 URI](#)

Objects | Properties


Objects (1)

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](#)

[Refresh](#)
[Copy S3 URI](#)
[Copy URL](#)
[Download](#)
[Open](#)
[Delete](#)
[Actions](#)

[Create folder](#)
[Upload](#)

Find objects by prefix

| <input type="checkbox"/> | Name ▲ | Type ▼ | Last modified ▼ | Size ▼ | Storage class ▼ |
|--------------------------|--|---------|---------------------------------------|----------|-----------------|
| <input type="checkbox"/> |  geo-sampleTelemetryRawData.parquet | parquet | August 18, 2022, 11:45:24 (UTC-05:00) | 204.6 KB | Standard |

3. Create S3 trigger for the function

With the function fully configured in Lambda and tested using a sample trigger event, you can now create a S3 trigger for the function.

Procedure

1. Select the function that you created and click Add Trigger in the Function overview section.


2. Select S3 as the trigger source and configure the following:

- Bucket – Select the bucket you created
- Event type – All object create events
- Prefix – truck-telemetry-raw/
- Click the checkbox to acknowledge recursive invocation

Lambda > Add trigger

Add trigger

Trigger configuration

 **S3**
aws storage

Bucket
Please select the S3 bucket that serves as the event source. The bucket must be in the same region as the function.

Event type
Select the events that you want to have trigger the Lambda function. You can optionally set up a prefix or suffix for an event. However, for each bucket, individual events cannot have multiple configurations with overlapping prefixes or suffixes that could match the same object key.

Prefix - optional
Enter a single optional prefix to limit the notifications to objects with keys that start with matching characters.

Suffix - optional
Enter a single optional suffix to limit the notifications to objects with keys that end with matching characters.

Recursive invocation
If your function writes objects to an S3 bucket, ensure that you are using different S3 buckets for input and output. Writing to the same bucket increases the risk of creating a recursive invocation, which can result in increased Lambda usage and increased costs. [Learn more](#)

☒ I acknowledge that using the same S3 bucket for both input and output is not recommended and that this configuration can cause recursive invocations, increased Lambda usage, and increased costs.

Lambda will add the necessary permissions for AWS S3 to invoke your Lambda function from this trigger. [Learn more](#) about the Lambda permissions model.

3. Click Add.

4. With the trigger created, when any new telemetry file lands in <<your_bucket>>/truck-telemetry-raw, AWS Lambda will execute the function.

With the trigger created, when any new telemetry file lands in <<your_bucket>>/truck-telemetry-raw, AWS Lambda will execute the function.

5. You can test this by uploading the [sample telemetry file](#) into `<<your_bucket>>/truck-telemetry-raw`.

It will generate a trigger event that spins up the function which results in the processed files landing in `<<your_bucket>>/processed/truck-geo-events` and `<<your_bucket>>/processed/truck-speed-events`.

Amazon S3 > Buckets > dataflowfunctionsquickstart > truck-telemetry-processed/ > truck-geo-events/ > geo-telemetry-sensor-readings-0.parquet

geo-telemetry-sensor-readings-0.parquet [Info](#)

[Copy S3 URI](#)
[Download](#)
[Open](#)
[Object actions](#) ▼

[Properties](#)
[Permissions](#)
[Versions](#)

Object overview

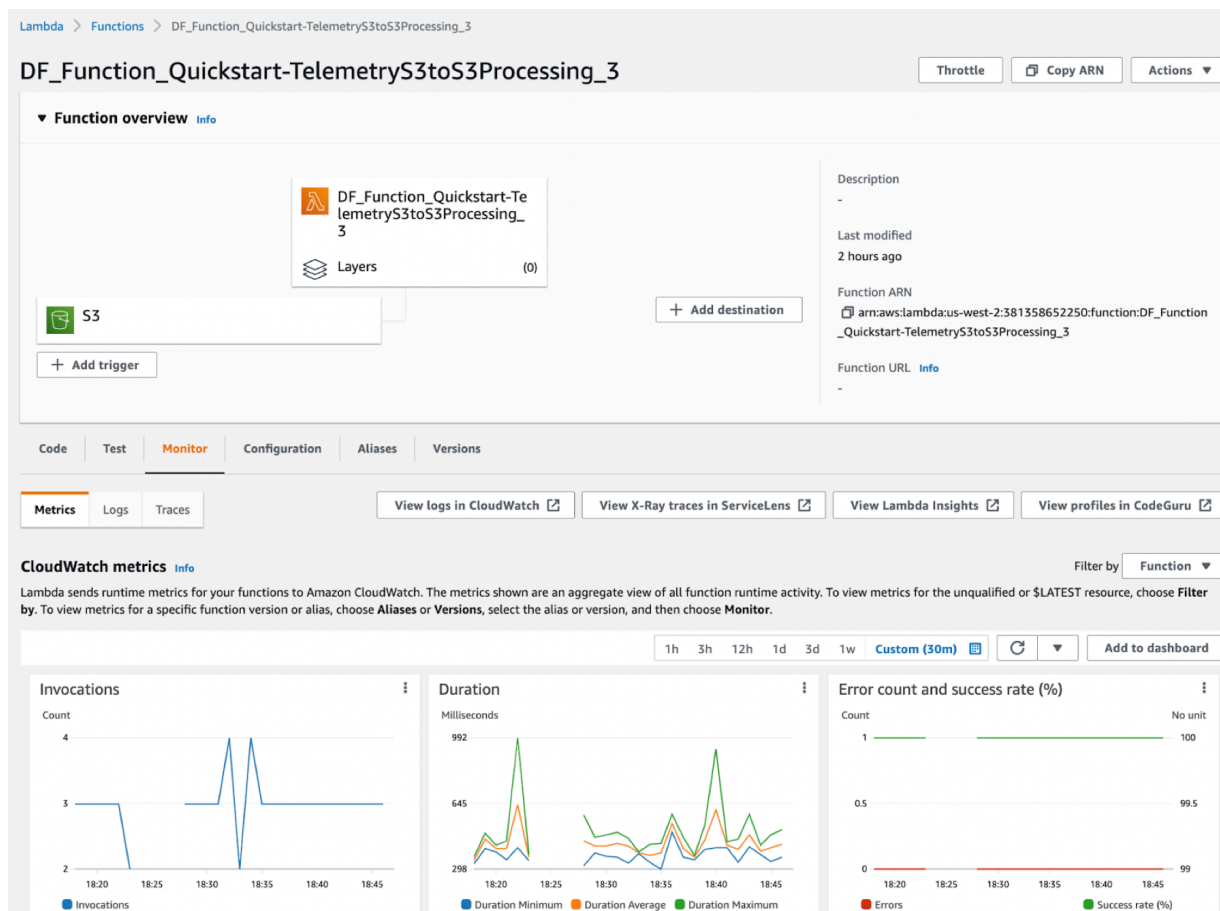
| | |
|--|--|
| Owner cloud-aws-pm-cdp-sandbox-env | S3 URI s3://dataflowfunctionsquickstart/truck-telemetry-processed/truck-geo-events/geo-telemetry-sensor-readings-0.parquet |
| AWS Region US West (Oregon) us-west-2 | Amazon Resource Name (ARN) arn:aws:s3:::dataflowfunctionsquickstart/truck-telemetry-processed/truck-geo-events/geo-telemetry-sensor-readings-0.parquet |
| Last modified August 18, 2022, 12:52:59 (UTC-05:00) | Entity tag (Etag) 972cf3ae0810147366679bda7a973e10 |
| Size 265.6 KB | Object URL https://dataflowfunctionsquickstart.s3.us-west-2.amazonaws.com/truck-telemetry-processed/truck-geo-events/geo-telemetry-sensor-readings-0.parquet |
| Type parquet | |
| Key truck-telemetry-processed/truck-geo-events/geo-telemetry-sensor-readings-0.parquet | |

4. Monitor the function

As more telemetry files are added to the landing S3 folder, you can view the metrics and logs of the serverless in the AWS Lambda monitoring view.

Procedure

- If you want to view metrics on all the function invocations, go to the Monitor tab and select the Metrics sub-tab.



- If you want to view all the function logs for each function invocation, check out the Logs sub-tab under the Monitor tab.

Lambda > Functions > DF_Function_Quickstart-TelemetryS3toS3Processing_3

DF_Function_Quickstart-TelemetryS3toS3Processing_3

Throttle Copy ARN Actions

▼ Function overview Info

DF_Function_Quickstart-TelemetryS3toS3Processing_3

Layers (0)

S3

+ Add destination

+ Add trigger

Description
-

Last modified
2 hours ago

Function ARN
arn:aws:lambda:us-west-2:381358652250:function:DF_Function_Quickstart-TelemetryS3toS3Processing_3

Function URL [Info](#)
-

Code Test **Monitor** Configuration Aliases Versions

Metrics **Logs** Traces

View logs in CloudWatch View X-Ray traces in ServiceLens View Lambda Insights View profiles in CodeGuru

CloudWatch Logs Insights Info

Lambda logs all requests handled by your function and automatically stores logs generated by your code through Amazon CloudWatch Logs. To validate your code, instrument it with custom logging statements. The following tables list the most recent and most expensive function invocations across all function activity. To view logs for a specific function version or alias, visit the **Monitor** section at that level.

1h 3h 12h 1d 3d 1w Custom Refresh Add to dashboard

Recent invocations

| # | Timestamp | RequestID | LogStream | DurationInMS | BilledDurationIn... | MemorySetInMB |
|---|--------------------------|---------------------------------------|---|--------------|---------------------|---------------|
| 1 | 2022-08-18T18:48:04.280Z | 3d35d8a0-546a-42f7-b937-08cb7b0423da | 2022/08/18/[\$LATEST]83c53a1a5f364c7cb19333a564e2bd06 | 390.67 | 391.0 | 2048 |
| 2 | 2022-08-18T18:47:43.554Z | 41aa1ba2-61e2-47f7-a89f-0a291ffaec18 | 2022/08/18/[\$LATEST]83c53a1a5f364c7cb19333a564e2bd06 | 452.19 | 453.0 | 2048 |
| 3 | 2022-08-18T18:47:24.570Z | d12569f7-22f0-4946-aa3a-f973d5201cd2 | 2022/08/18/[\$LATEST]83c53a1a5f364c7cb19333a564e2bd06 | 387.55 | 388.0 | 2048 |
| 4 | 2022-08-18T18:47:05.111Z | b54e06e7-2a62-4449-b152-221cc4b74aea | 2022/08/18/[\$LATEST]83c53a1a5f364c7cb19333a564e2bd06 | 483.92 | 484.0 | 2048 |
| 5 | 2022-08-18T18:46:42.277Z | 348dd3be-3906-4d19-b7fc-3fe8fbcbb0bd | 2022/08/18/[\$LATEST]83c53a1a5f364c7cb19333a564e2bd06 | 416.83 | 417.0 | 2048 |
| 6 | 2022-08-18T18:46:22.487Z | 316f895c-7f98-4a56-bb5a-4336e4bf53a1 | 2022/08/18/[\$LATEST]83c53a1a5f364c7cb19333a564e2bd06 | 508.99 | 509.0 | 2048 |
| 7 | 2022-08-18T18:46:06.242Z | fdff5904-325e-4596-bc11-ed0aefeffbdcf | 2022/08/18/[\$LATEST]83c53a1a5f364c7cb19333a564e2bd06 | 360.72 | 361.0 | 2048 |
| 8 | 2022-08-18T18:45:46.341Z | be9f32e6-02e0-4e9a-a1e9-74c37a188f55 | 2022/08/18/[\$LATEST]83c53a1a5f364c7cb19333a564e2bd06 | 339.88 | 340.0 | 2048 |
| 9 | 2022-08-18T18:45:27.186Z | 1a3ca8e4-ae90-40d1-a708-4a0f081539db | 2022/08/18/[\$LATEST]83c53a1a5f364c7cb19333a564e2bd06 | 419.49 | 420.0 | 2048 |

Most expensive invocations

| # | Timestamp | RequestID | LogStream | BilledDurationIn... | MemorySetInMB | BilledDurationIn... |
|---|--------------------------|---------------------------------------|---|---------------------|---------------|---------------------|
| 1 | 2022-08-18T16:45:23.733Z | 22940085-0164-4d2b-861e-8d5d6f0cba3e | 2022/08/18/[\$LATEST]2b2d0b26396f4214835626ee5da508fc | 77052.0 | 2048 | 154.104 |
| 2 | 2022-08-18T16:22:13.523Z | 6ce44050-394c-4c60-89a3-2a6c307c4dea | 2022/08/18/[\$LATEST]1cde4f1b4e2b477d8f3348dfede09258 | 74497.0 | 2048 | 148.994 |
| 3 | 2022-08-18T17:52:58.883Z | b8fceda6-8612-4d12-b725-91bad5b1ba31 | 2022/08/18/[\$LATEST]83c53a1a5f364c7cb19333a564e2bd06 | 72293.0 | 2048 | 144.586 |
| 4 | 2022-08-18T18:08:03.723Z | c0afe5c6-a569-4b2b-9376-dc056d728fab | 2022/08/18/[\$LATEST]83c53a1a5f364c7cb19333a564e2bd06 | 2242.0 | 2048 | 4.484 |
| 5 | 2022-08-18T18:13:52.095Z | dfe5e0b8-25ad-4dfb-b963-1066166f1c64 | 2022/08/18/[\$LATEST]83c53a1a5f364c7cb19333a564e2bd06 | 1982.0 | 2048 | 3.964 |
| 6 | 2022-08-18T18:14:11.674Z | 9c3f5243-424a-40f0-8831-0940629a6365 | 2022/08/18/[\$LATEST]83c53a1a5f364c7cb19333a564e2bd06 | 1616.0 | 2048 | 3.232 |
| 7 | 2022-08-18T18:05:03.778Z | 0de46428-3031-4db5-9608-cd865ce08f49 | 2022/08/18/[\$LATEST]83c53a1a5f364c7cb19333a564e2bd06 | 1352.0 | 2048 | 2.704 |
| 8 | 2022-08-18T18:22:20.197Z | 5c1e9c33-2483-43a1-8294-91c41c7c3a53 | 2022/08/18/[\$LATEST]83c53a1a5f364c7cb19333a564e2bd06 | 993.0 | 2048 | 1.986 |
| 9 | 2022-08-18T18:40:34.417Z | 1b410b06-be5e-401f-b4a2-a7e5dfe0ab45d | 2022/08/18/[\$LATEST]83c53a1a5f364c7cb19333a564e2bd06 | 934.0 | 2048 | 1.868 |