Cloudera Data Flow

DataFlow Functions Telemetry Tutorial

Date published: 2021-04-06
Date modified: 2025-09-30

CLOUDERA

Legal Notice

Contents

Scope and goals

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

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

1. Register for a 60-day trial for Cloudera on cloud Trial which will provision a tenant in Cloudera on cloud.
2. Implement a data distribution use case on a local development workstation using Apache NiFi.
3. Register the NiFi flow in Cloudera Data Flow Service as a Cloudera Data Flow function.
4. Deploy the Cloudera Data Flow 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 on cloud	Cloudera's data management platform in the cloud
Cloudera Data Flow Service	Cloudera Data Flow data service that enables self-serve deployments of Apache NiFi
Cloudera Data Flow function	Flow that is uploaded into the Cloudera Data Flow Catalog that can be run in serverless mode by serverless cloud provider services
Cloudera Data Flow Catalog	Inventory of flow definitions from which you can initiate new deployments
AWS Lambda	Serverless, event-driven compute service that lets you run a Cloudera Data Flow 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 perfom 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 Cloudera on cloud 60-day trial

This section provides information on the free Cloudera on cloud trial.

Procedure

1. Log into your Cloudera 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 Cloudera on cloud 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 Cloudera on cloud.
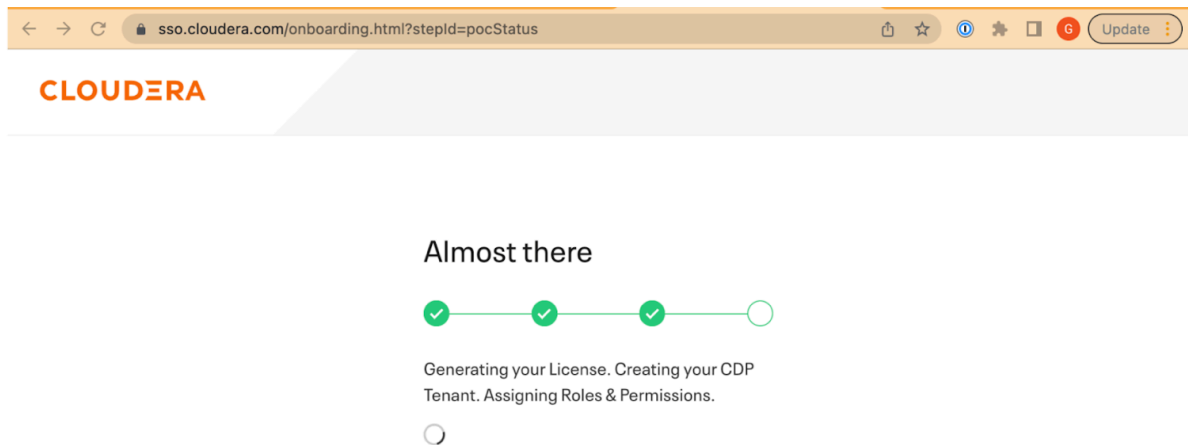

- If your company has not been registered for a trial account, shortly after completing the registration, the Cloudera Sales Team will reach out to you to complete your trial registration.
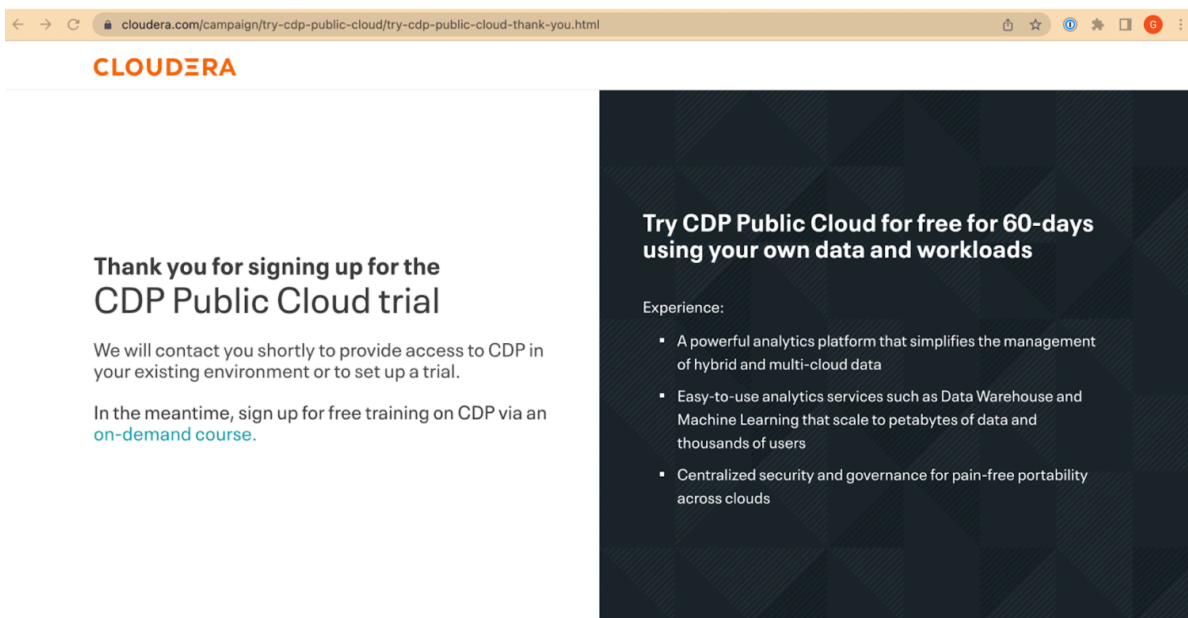
3. After completing the registration, you will receive an email with instructions on how to log into your newly provisioned tenant on Cloudera on cloud. Disregard steps 2 and 3 as Cloudera Data Flow Function does not require any of the prerequisites to create an environment or a DataLake.
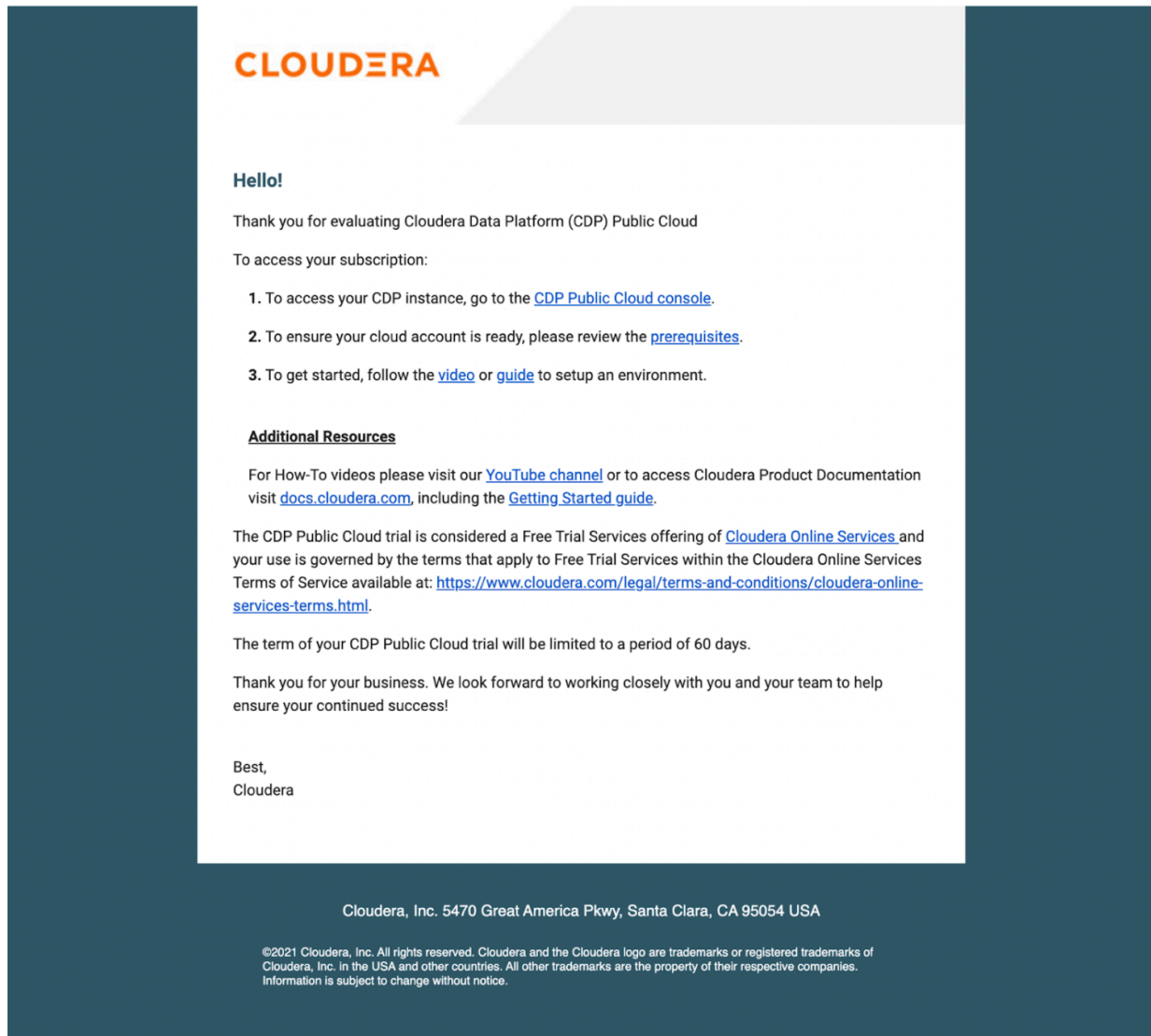
4. After logging into the Cloudera on cloud console with the credentials that you used to register for the trial, select the Cloudera Data Flow service from the homepage.
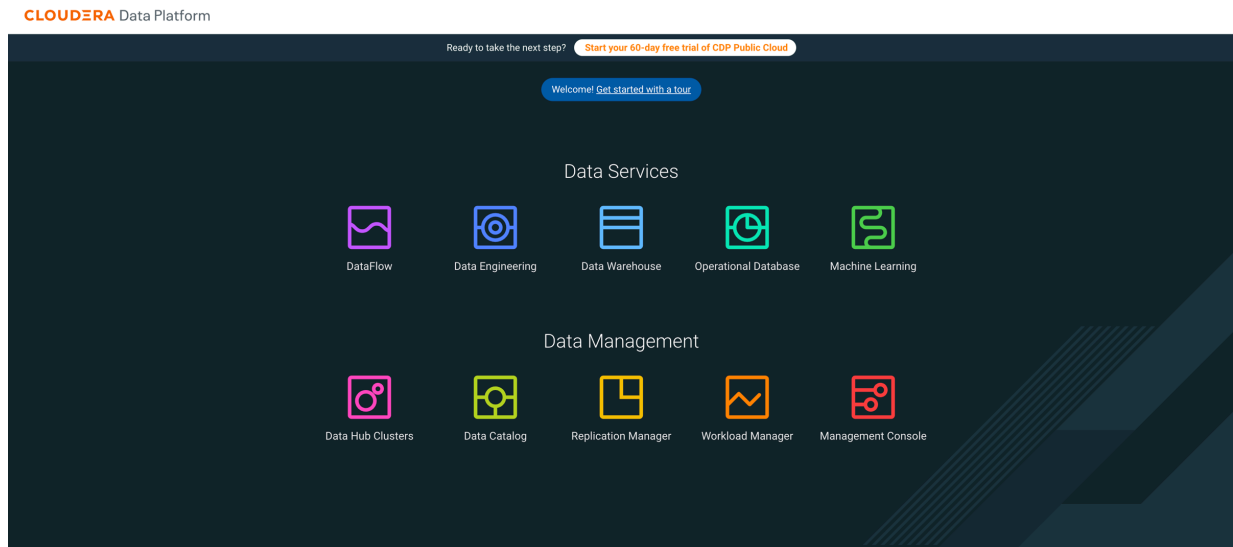

5. If you click Functions in the left navigation pane, you can see the options for cloud provider functions.
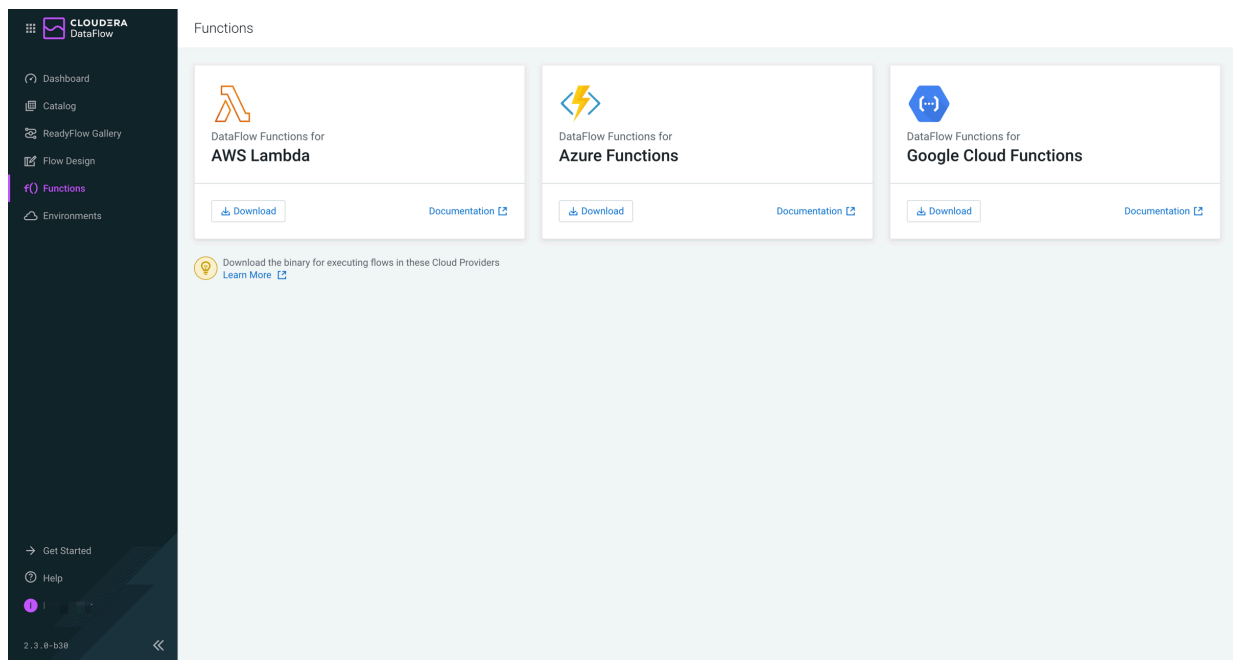
This is from where you can Download on the AWS Lambda tile to download the Cloudera Data Flow function binaries for AWS Lambda.

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 Cloudera Data Flow function.
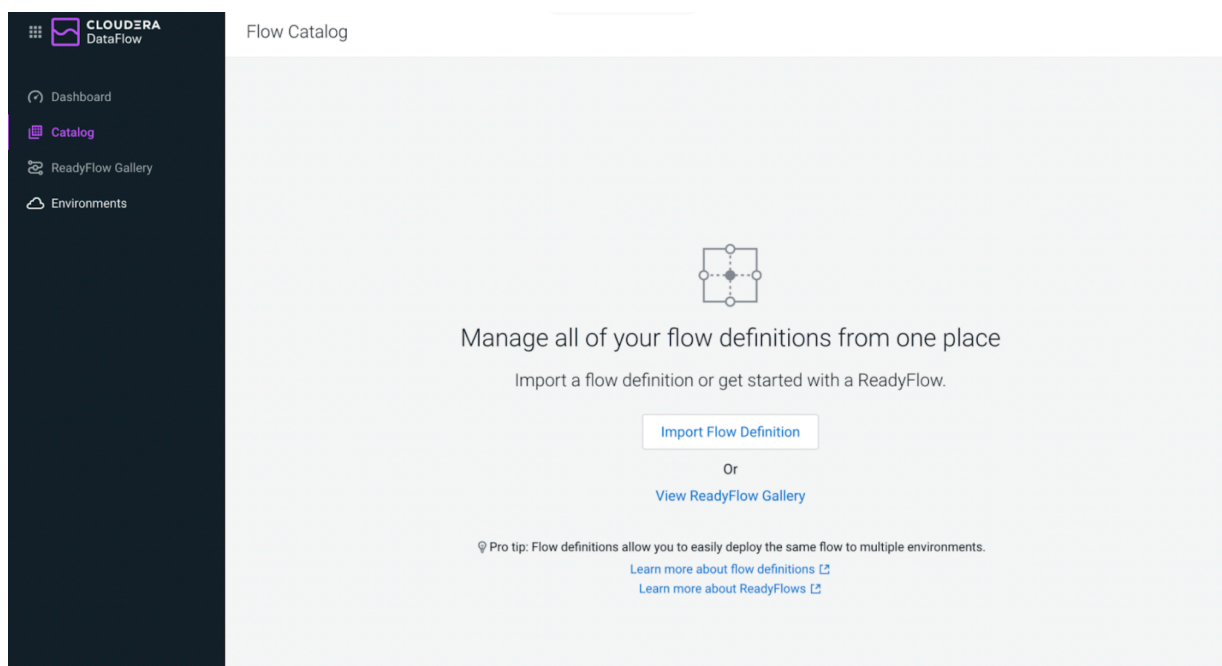
Develop and test a flow in Apache NiFi

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 Cloudera Data Flow function in serverless mode using AWS Lambda.
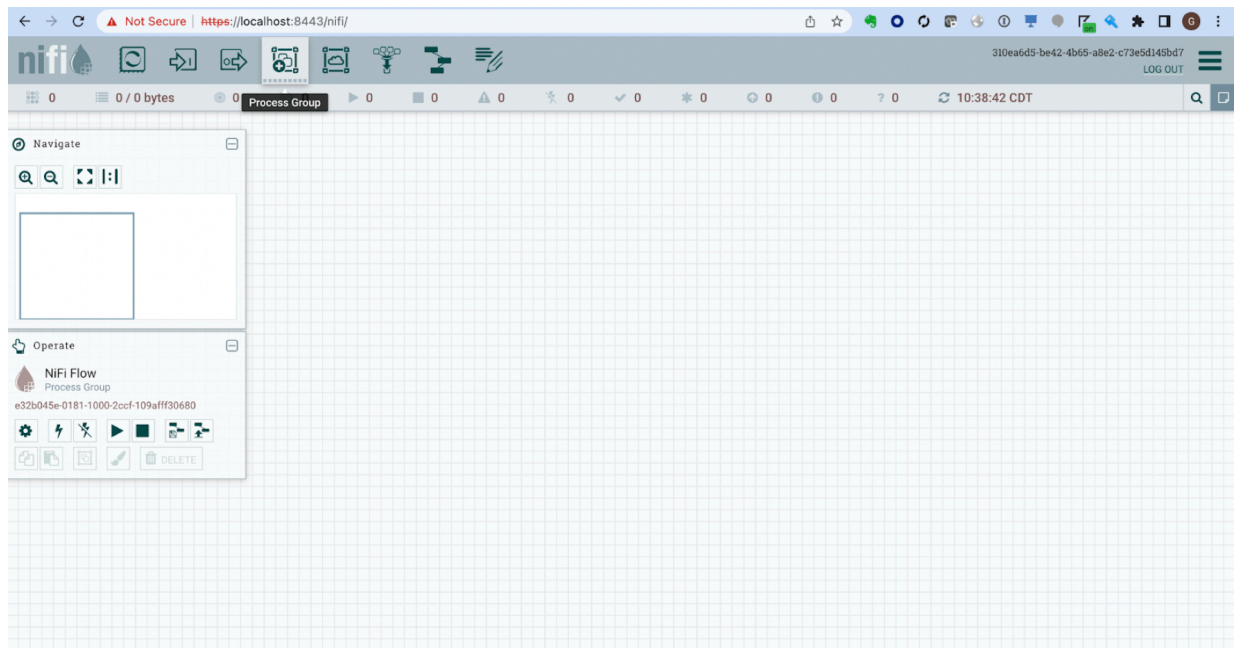
1. Install Apache NiFi

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.
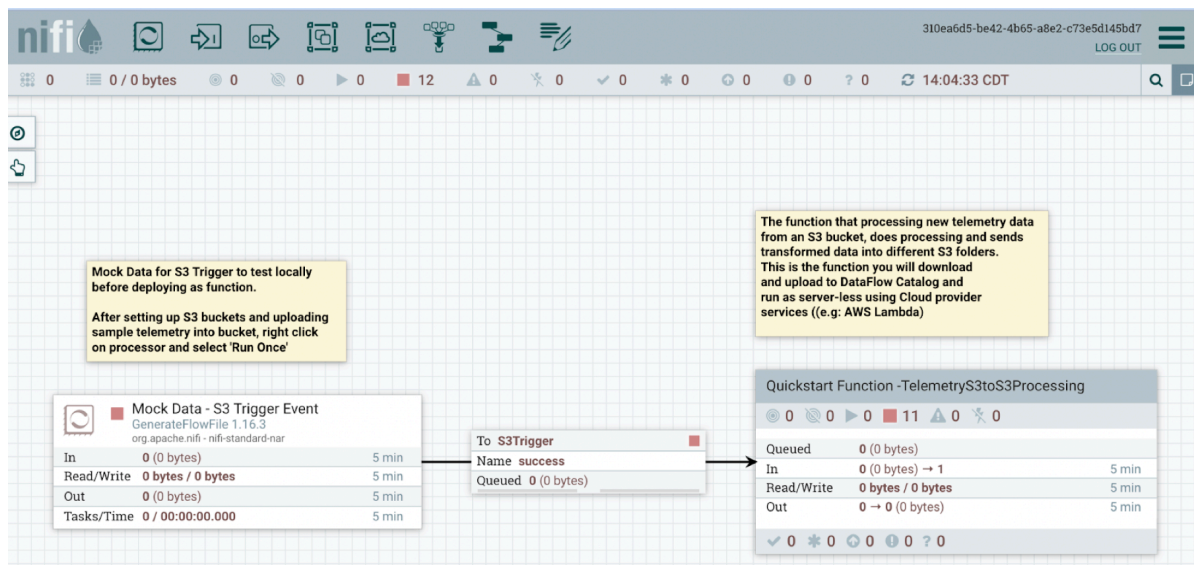
Procedure

1. Download the Telemetry Tutorial NiFi Flow to your local machine.

2. Drag a Process Group (fourth icon from the left) onto the canvas.

3. Click the browse link, select the Telemetry Tutorial NiFi Flow you have downloaded, and click Add.
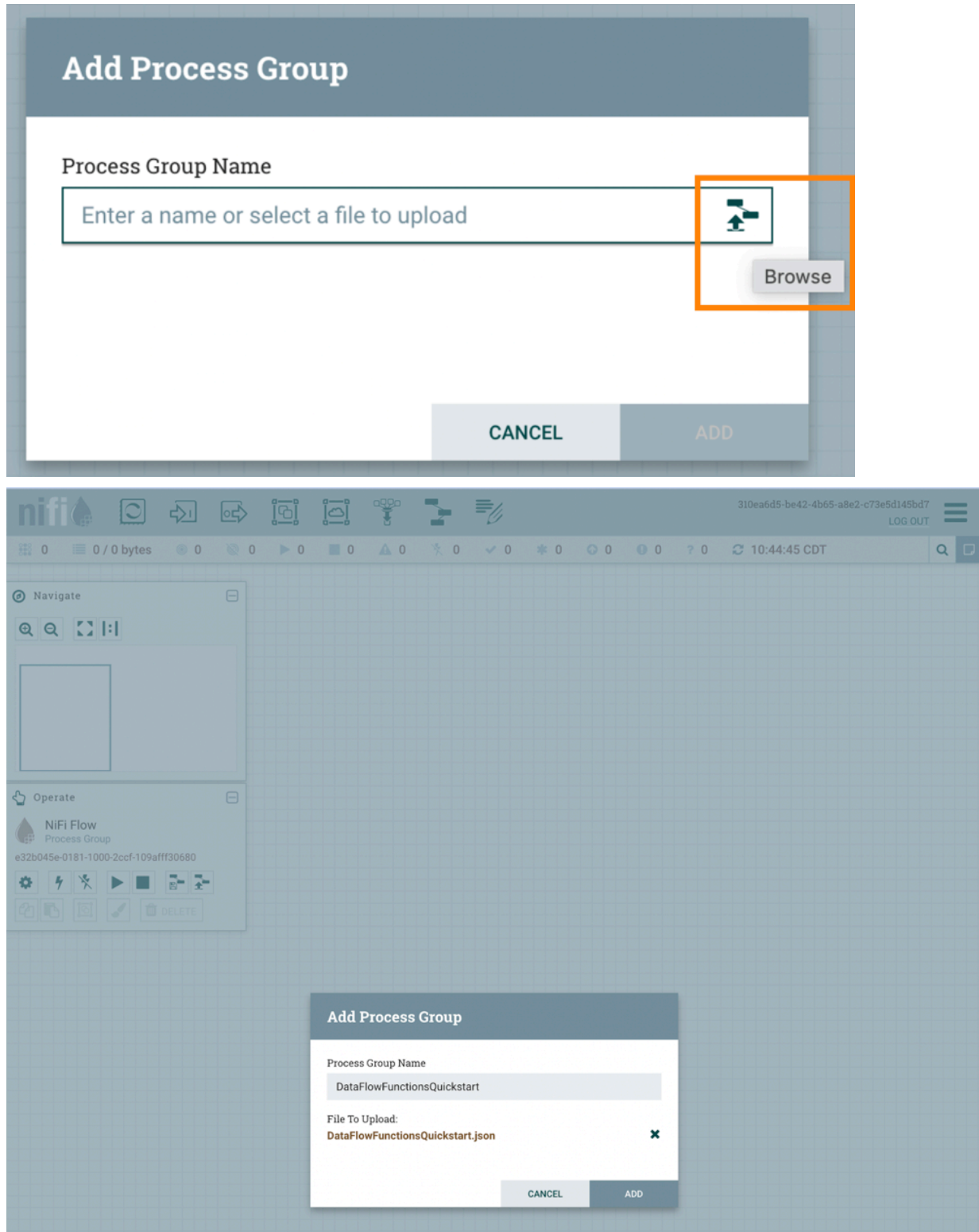
4. Explore the flow that you have uploaded to the canvas.

 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:
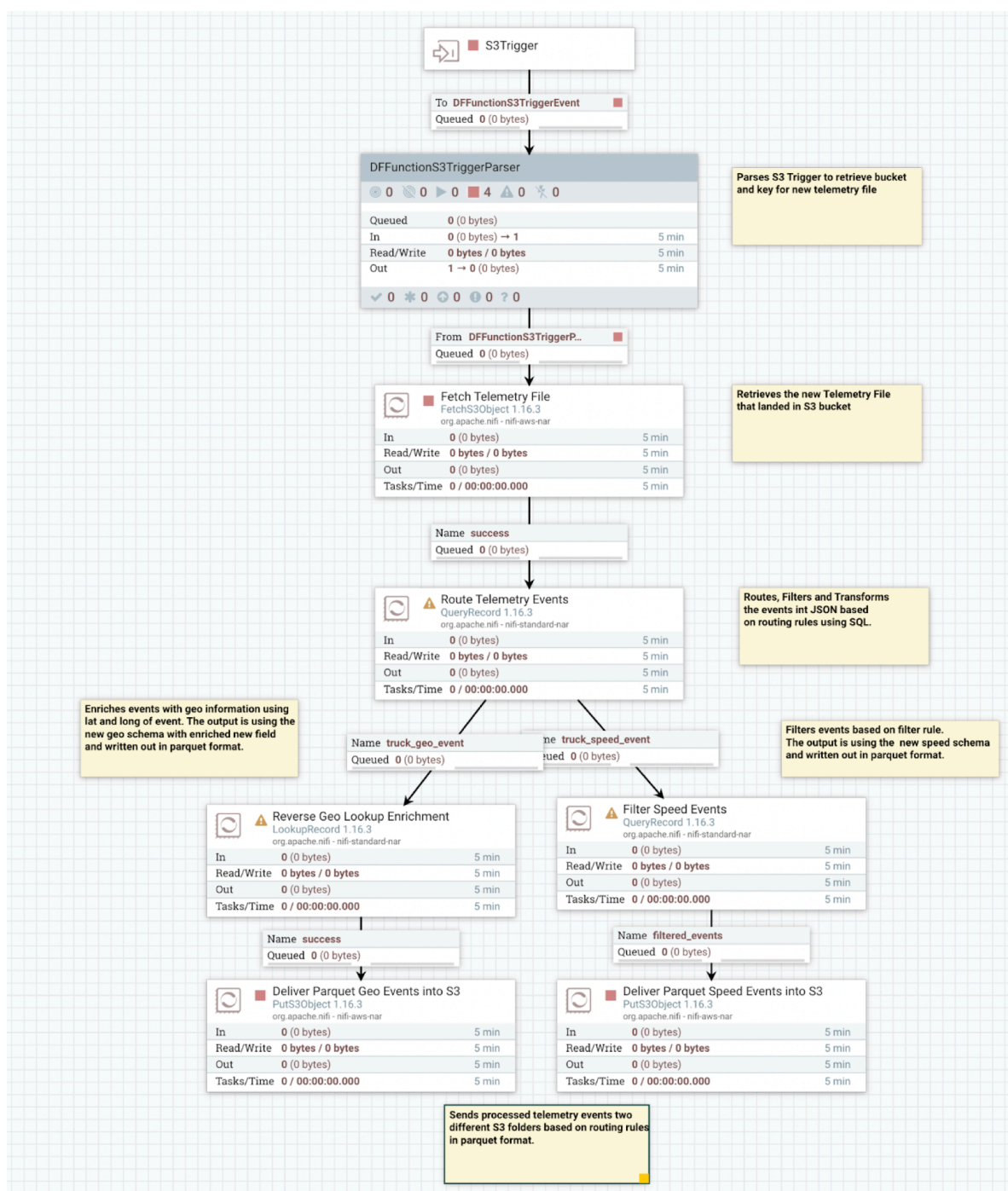
- S3Trigger is an input port that the AWS Lambda handler sends the S3 trigger event to. All functions need to start with the input port for the corresponding trigger that will be used.
- DFFunctionS3TriggerParser parses the json trigger event to extract the bucket, key and region and store in flow attributes 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

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

- 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

 | | 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... | ⚙ ⚡ 🗑 |
   | | RawTelemetryJsonReaderJson | JsonTreeReader 1.16.3 | org.apache.nifi - nifi-record-serialization-... | Disabled | Quickstart Function -TelemetryS3toS3Pr... | ⚙ ⚡ 🗑 |
   | | RawTelemetryJsonWriterJson | 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.
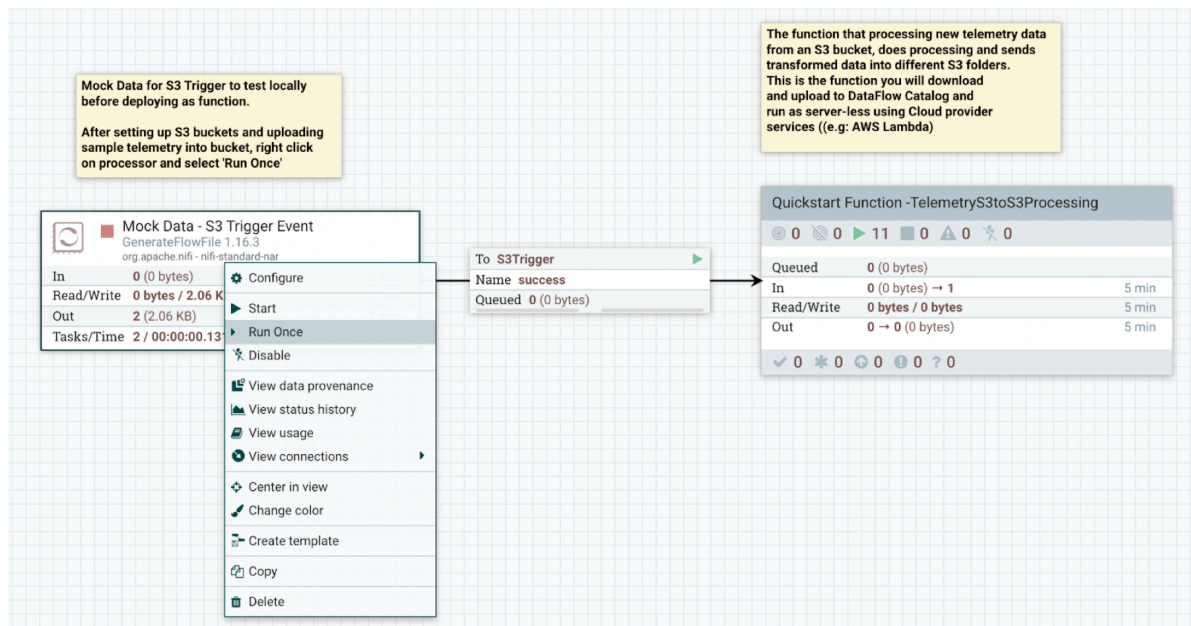
   Note: This is the file is that you uploaded to <<your_bucket>>/truck-telemetry-raw in the prerequisite step.

   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.

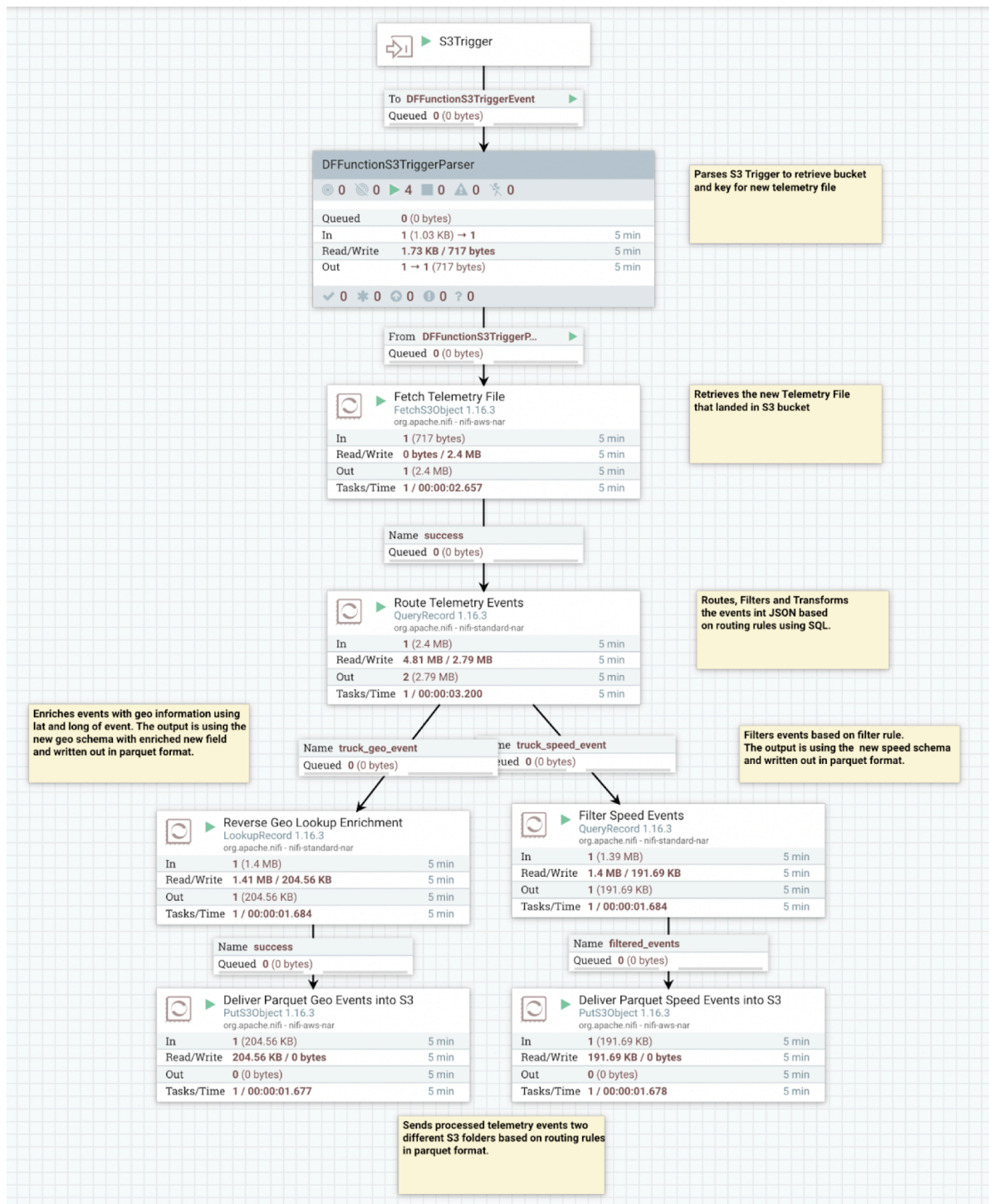a) Right-click the 'Mock Data - S3 Trigger Event' processor group and select 'Run Once'.

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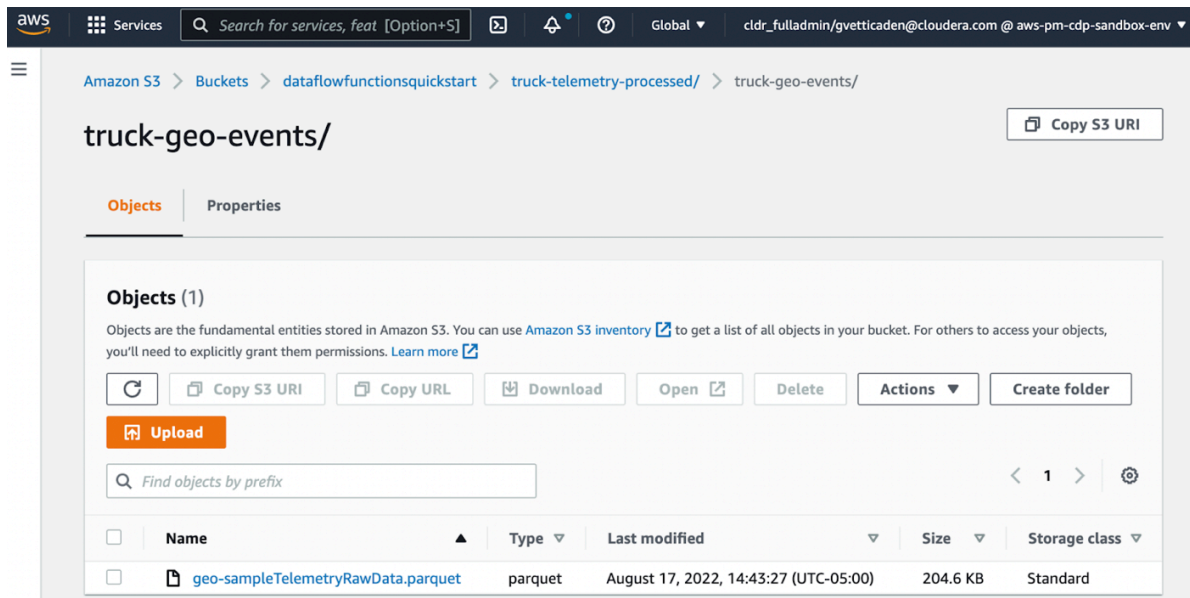
The In metric should show '1' across all processors.

S3Trigger

To **DFFunctionS3TriggerEvent** ▶
Queued **0** (0 bytes)

DFFunctionS3TriggerParser

◉ 0   ◎ 0   ▶ 4   ■ 0   ⚠ 0   ⚡ 0

Queued	**0** (0 bytes)	
In	**1** (1.03 KB) → **1**	5 min
Read/Write	**1.73 KB / 717 bytes**	5 min
Out	**1 → 1** (717 bytes)	5 min

✓ 0   ✳ 0   ⟳ 0   ⓘ 0   ? 0

Parses S3 Trigger to retrieve bucket
and key for new telemetry file

From **DFFunctionS3TriggerP...** ▶
Queued **0** (0 bytes)

Fetch Telemetry File
FetchS3Object 1.16.3
org.apache.nifi - nifi-aws-nar

In	**1** (717 bytes)	5 min
Read/Write	**0 bytes / 2.4 MB**	5 min
Out	**1** (2.4 MB)	5 min
Tasks/Time	**1 / 00:00:02.657**	5 min

Retrieves the new Telemetry File
that landed in S3 bucket

Name **success**
Queued **0** (0 bytes)

Route Telemetry Events
QueryRecord 1.16.3
org.apache.nifi - nifi-standard-nar

In	**1** (2.4 MB)	5 min
Read/Write	**4.81 MB / 2.79 MB**	5 min
Out	**2** (2.79 MB)	5 min
Tasks/Time	**1 / 00:00:03.200**	5 min

Routes, Filters and Transforms
the events int JSON based
on routing rules using SQL.

Enriches events with geo information using
lat and long of event. The output is using the
new geo schema with enriched new field
and written out in parquet format.

Name **truck_geo_event**
Queued **0** (0 bytes)

ne **truck_speed_event**
ued **0** (0 bytes)

Filters events based on filter rule.
The output is using the new speed schema
and written out in parquet format.

Reverse Geo Lookup Enrichment
LookupRecord 1.16.3
org.apache.nifi - nifi-standard-nar

In	**1** (1.4 MB)	5 min
Read/Write	**1.41 MB / 204.56 KB**	5 min
Out	**1** (204.56 KB)	5 min
Tasks/Time	**1 / 00:00:01.684**	5 min

Filter Speed Events
QueryRecord 1.16.3
org.apache.nifi - nifi-standard-nar

In	**1** (1.39 MB)	5 min
Read/Write	**1.4 MB / 191.69 KB**	5 min
Out	**1** (191.69 KB)	5 min
Tasks/Time	**1 / 00:00:01.684**	5 min

Name **success**
Queued **0** (0 bytes)

Name **filtered_events**
Queued **0** (0 bytes)

Deliver Parquet Geo Events into S3
PutS3Object 1.16.3
org.apache.nifi - nifi-aws-nar

In	**1** (204.56 KB)	5 min
Read/Write	**204.56 KB / 0 bytes**	5 min
Out	**0** (0 bytes)	5 min
Tasks/Time	**1 / 00:00:01.677**	5 min

Deliver Parquet Speed Events into S3
PutS3Object 1.16.3
org.apache.nifi - nifi-aws-nar

In	**1** (191.69 KB)	5 min
Read/Write	**191.69 KB / 0 bytes**	5 min
Out	**0** (0 bytes)	5 min
Tasks/Time	**1 / 00:00:01.678**	5 min

Sends processed telemetry events two
different S3 folders based on routing rules
in parquet format.

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 Cloudera Data Flow Catalog and run it in serverless mode.

Procedure

1. 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)

Quickstart Function -TelemetryS3toS3Processing

◎ 0   ⊘ 0   ▶ 0   ■ 11   ⚠ 0   ✗ 0

Queued	**0** (0 bytes)	
In	**0** (0 bytes) → **1**	5 min
Read/Write		5 min
Out		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
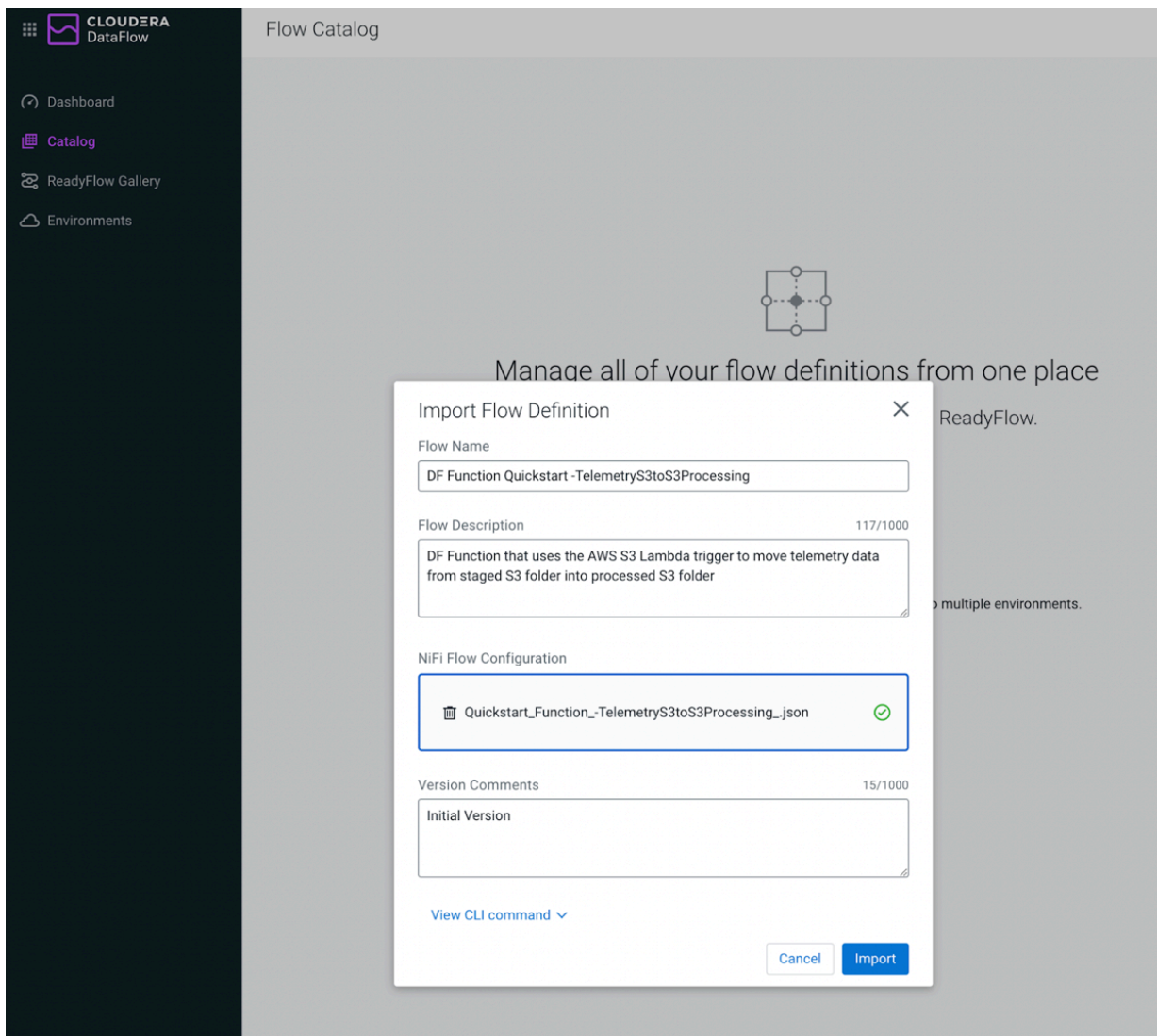
Add the Telemetry Tutorial NiFi Flow as a function in Cloudera Data Flow

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

1. Upload the Telemetry Tutorial NiFi Flow to the Cloudera Data Flow Catalog

Procedure

1. After logging into the Cloudera on cloud console with the credentials that you used to register for the trial, select the Cloudera Data Flow service on the Cloudera on cloud 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.

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.

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

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

Procedure

1. Click Functions in the left navigation pane and download the Cloudera Data Flow Function binaries for AWS Lambda.

AWS Lambda will use these binaries to run the NiFi flow.

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.

3. Create a Cloudera on cloud Service Account

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

Procedure

1. Go to the Cloudera on cloud homepage, select Management Console User Management .

2. Under Actions, select Create Machine User and provide a name for the service account.
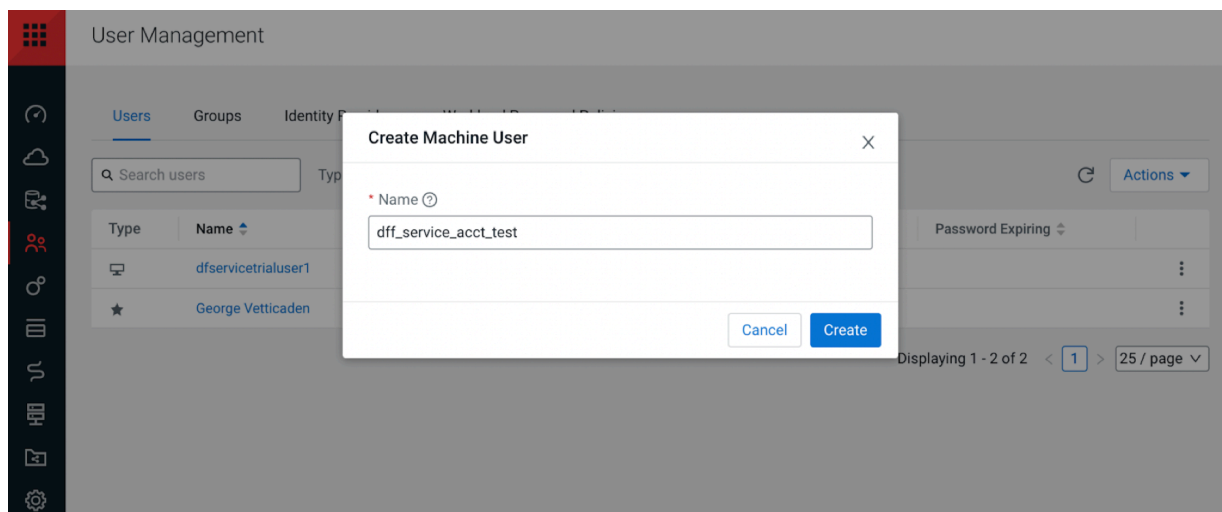
3. Click Roles Update Roles and select the DFFunctionMachineUser.

4. Click Update.
5. 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.
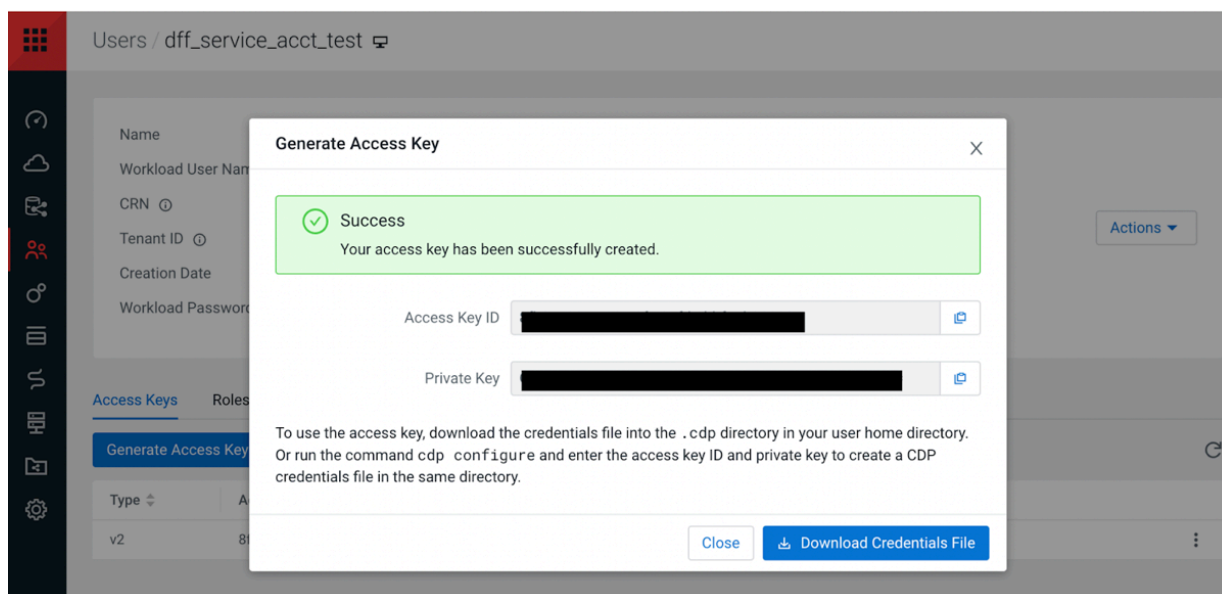
Run the Cloudera Data Flow function in serverless mode in AWS Lambda

Now that you have developed the NiFi flow and tested locally, registered it as Cloudera Data Flow function in Cloudera Data Flow 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 Cloudera Data Flow function

You can use the AWS CLI to create and configure the Cloudera Data Flow 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 Cloudera Data Flow 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 --a
ssume-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_L
ambda_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.

2. Create the Cloudera Data Flow function in Lambda.

a) Download the Cloudera Data Flow Function Definition JSON file.

- This file has the full definition required to create the Cloudera Data Flow 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 Cloudera Data Flow 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 Cloudera on cloud service account
- DF_PRIVATE_KEY – The private key created for the Cloudera on cloud service account
- FLOW_CRN – The CRN value you copied from the Cloudera Data Flow 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 Cloudera Data Flow Functions 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_Quickst
art-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:

    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

λ 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	████████████████
DF_PRIVATE_KEY	████████████████████
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	████████████████
aws_access_key_password	████████████████████
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 Cloudera Data Flow 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:

   a. Provide a name for the test: NiFi_Function_Quickstart.
   b. Copy the contents from the sampleTriggerEvent and paste it into the Event JSON field.
   c. Replace the bucket name with the one you created.
   d. Click Save.

2. Execute the Test Event.

   a) 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.

   b) 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:

   c) Validate that the processed Parquet files are under the following directories:

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

3. Create S3 trigger for the Cloudera Data Flow function

With the Cloudera Data Flow 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

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 Cloudera Data Flow function.

With the trigger created, when any new telemetry file lands in <<your_bucket>>/truck-telemetry-raw, AWS Lambda will execute the Cloudera Data Flow 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 Cloudera Data Flow function which results in the processed files landing in <<your_bucket>>/processed/truck-geo-events and <<your_bucket>>/processed/truck-speed-events.

4. Monitor the Cloudera Data Flow function

As more telemetry files are added to the landing S3 folder, you can view the metrics and logs of the serverless Cloudera Data Flow Functions 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.
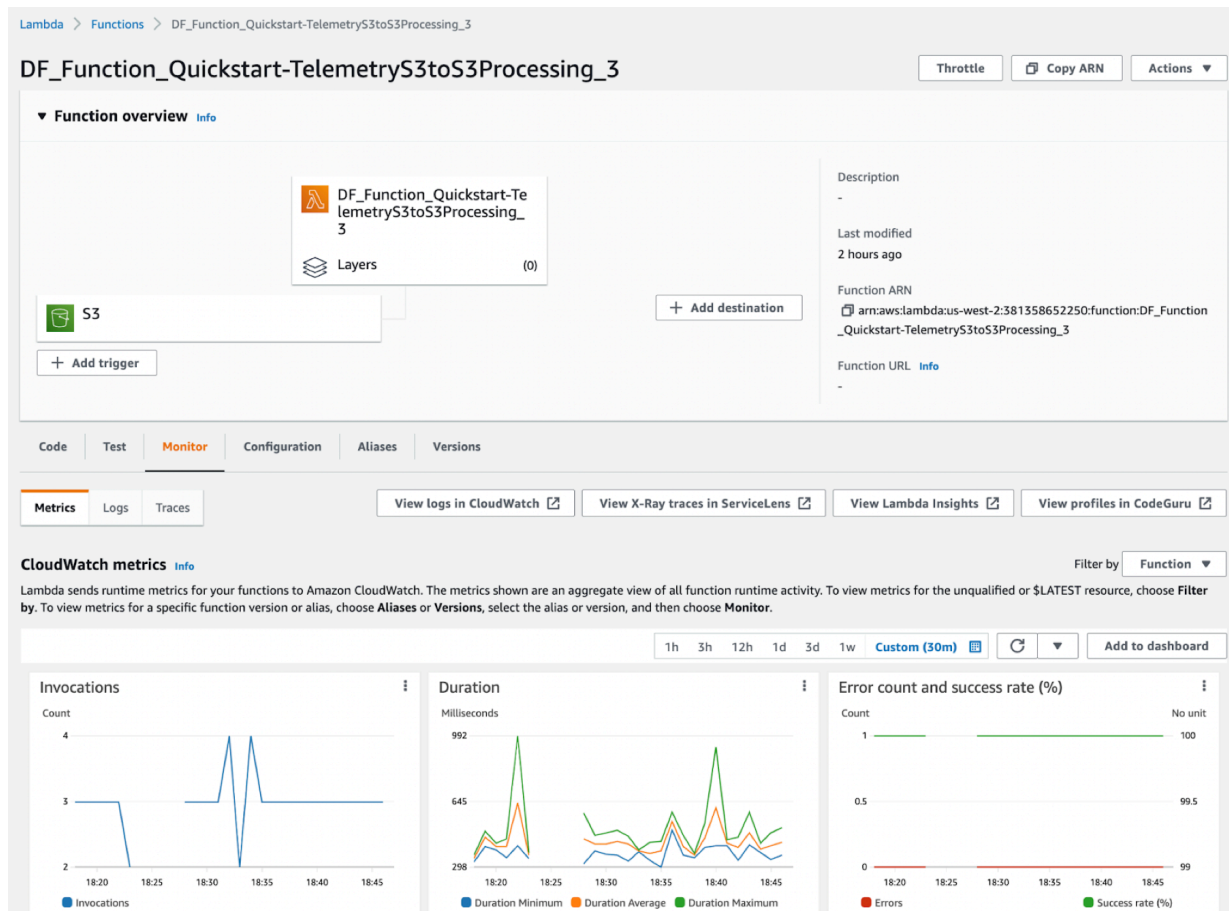
- If you want to view all the Cloudera Data Flow function logs for each function invocation, check out the Logs sub-tab under the Monitor tab.