Cloudera Data Flow

Getting Ready to Run Functions

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

CLOUDERA

Legal Notice

Contents

Developing a NiFi flow for Cloudera Data Flow Functions

A flow definition represents the data flow logic that you can download from NiFi, import to the Cloudera Data Flow Catalog and run in serverless mode. You can develop this data flow for your function in any development environment using Apache NiFi and then deploy the function on a Function as a Service (FAAS) solution on AWS, Azure, or the Google Cloud Platform (GCP).

You have three main options for developing your flow:

- Use Cloudera Data Flow Flow Designer and publish your draft as a flow definition in the Catalog. For more information on using the Flow Designer, see *Creating a new draft*. In this case you can skip the downloading flow and importing flow to Catalog steps as you have already added your flow to the Cloudera Data Flow Catalog.
- Use Cloudera on cloud Data Hub with the Flow Management template, if you are a Cloudera customer who has a Cloudera on cloud Data Lake.

  For more information on how to set up a managed and secured Flow Management cluster in Cloudera on cloud, see *Setting up your Flow Management cluster*.
- Develop the data flow in your local development environment using open source Apache NiFi.

Once you have developed and tested your NiFi flow, you can deploy it as a Cloudera Data Flow function in serverless mode using one of the three cloud providers function services: AWS Lambda, Google Cloud Functions, and Azure Functions.

To make sure that your NiFi data flow can be deployed as a function in Cloudera Data Flow, review your traditional NiFi flow development process and follow the best practices outlined in the next sections.

Related Information

Creating a new draft

Setting up your Flow Management cluster

Flow design principles

To make sure that your Apache NiFi data flow can be deployed as a function in Cloudera Data Flow, follow the best practices outlined in this section.

Procedure

1. Create a Process Group and associate a Parameter Context to it.

   The Parameter Context can itself inherit or not, from other Parameter Contexts. Make sure that the name of your Parameter Contexts starts with a letter and it only consists of letters, numbers, and the underscore (_) character.

2. Go into the Process Group and add an Input Port.

   This is the start of your flow where the Cloudera Data Flow Functions code will "generate" a FlowFile containing the payload generated by the function's trigger. See the corresponding sections of each cloud provider in the Cloudera Data Flow Functions documentation to see what the FlowFile's content would be.

3. Add your processors to extract the required data from the FlowFile and perform the required actions.

4. Once the flow is designed, parameterize all processor and controller service properties that should be externalized for configuration during function creation.

   Make sure that all parameters have names that start with a letter and only consist of letters, numbers, and the underscore (_) character.

S3 trigger

In this example, use the EvaluateJsonPath processor to extract the bucket name and the object key, and continue the flow as you wish, for example fetch the data, compress it and send the result to a different location.
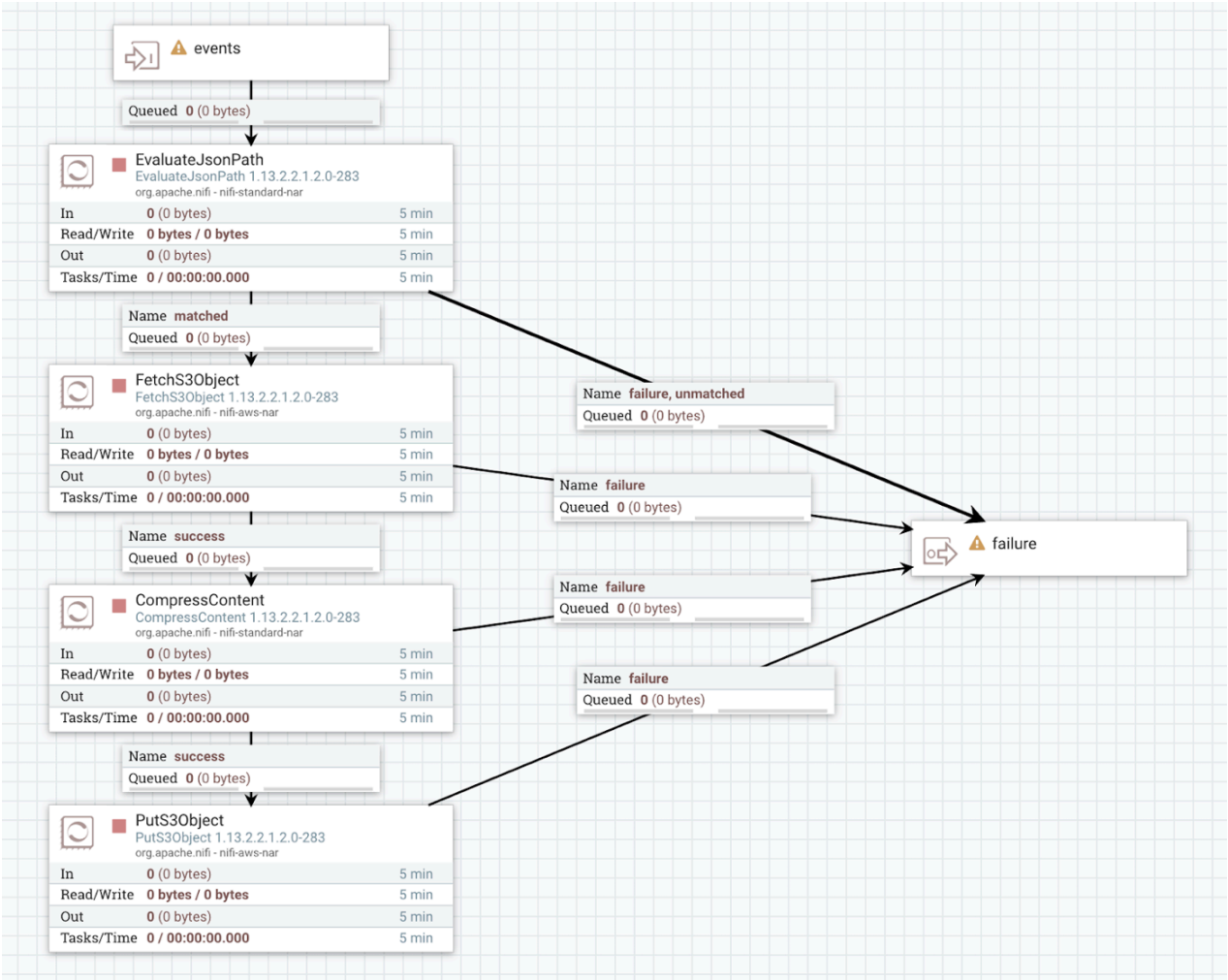
When using an AWS Lambda with an S3 trigger, the generated FlowFile would have a payload similar to this:

```
1  {
2    "Records": [
3      {
4        "eventVersion": "2.1",
5        "eventSource": "aws:s3",
6        "awsRegion": "us-east-2",
7        "eventTime": "2021-08-06T09:53:57.377Z",
8        "eventName": "ObjectCreated:Put",
9        "userIdentity": {
10         "principalId": "AWS:AIDAVRSV7HNNHFQQOE2F4"
11       },
12       "requestParameters": {
13         "sourceIPAddress": "18.222.152.52"
14       },
15       "responseElements": {
16         "x-amz-request-id": "WBPM5W0026J3NWEH",
17         "x-amz-id-2": "Cco/DVtbHFYtb+rXQrEMUFJnZSxtebZL61KG79emsACUDqUIyrAziY3frha9AzX+RSLx9syJMl4OMCn06poZv1dJQJg5TgRx"
18       },
19       "s3": {
20         "s3SchemaVersion": "1.0",
21         "configurationId": "ad0ec124-9c9f-476e-9dd1-9e513d9af822",
22         "bucket": {
23           "name": "pvillard-naaf-input",
24           "ownerIdentity": {
25             "principalId": "A339RKRQ79D9V"
26           },
27           "arn": "arn:aws:s3:::pvillard-naaf-input"
28         },
29         "object": {
30           "key": "04f56bcf-cb15-4244-9379-2abc7ac0cd8d",
31           "size": 1024,
32           "eTag": "265bc4c12162ffca179a58975728a033",
33           "sequencer": "00610D06B8BE365D41"
34         }
35       }
36     }
37   ]
38 }
```

Getting the payload associated with a trigger

You can design a flow that would just push as-is the event generated by the function's trigger into S3. This can be useful when you do not know beforehand what would be generated by a given trigger.
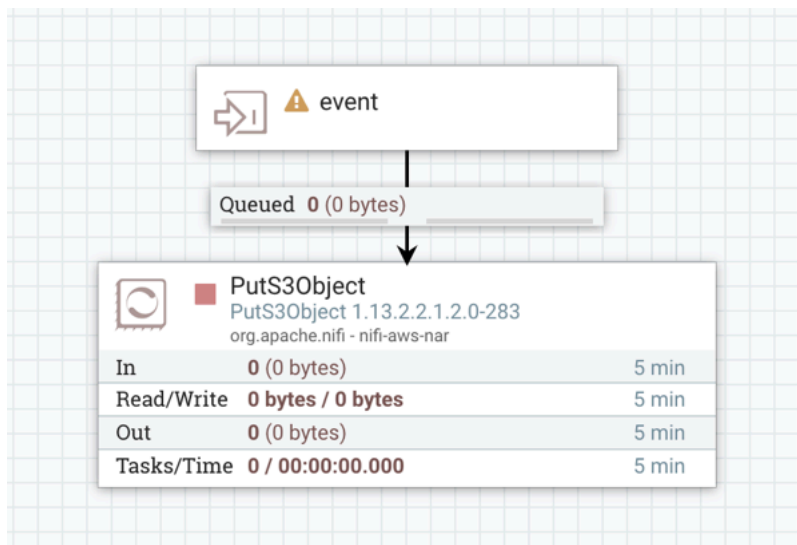
Flow testing principles

As with any data flow that you build, it is important to test it before deployment. Data flows to be used by AWS Lambda require an Input Port and most often will also contain at least one Output Port. This makes it quite easy to test these data flows in NiFi.

Procedure

1. Go to the Parent Process Group within Apache NiFi and add a GenerateFlowFile processor.
2. Set the Custom Text property to any value that you want to feed into your data flow.

   For example, to simulate an event indicating that data was added to an S3 bucket, you would set the Custom Text property to the following:

   ```
   {
      "Records": [
         {
            "awsRegion": "us-east-1",
            "s3": {
               "bucket": {
                  "name": "my-nifi-logs",
                  "arn": "arn:aws:s3:::example-bucket"
               },
               "object": {
                  "key": "nifi-app_2021-10-12_18.36.log.gz",
                  "size": 1024,
               }
            }
         }
      ]
   }
   ```

3. Connect the GenerateFlowFile processor to the Input Port of the Process Group that you will run in Lambda.

   The Run Once feature of NiFi makes it easy to create this FlowFile and send it to the Process Group.
4. Start the Process Group and ensure that the data processes as expected.

   If not, you can fix the data flow and trigger the GenerateFlowFile processor again, until you have properly handled the data.

What to do next
When ready, you can download the Flow Definition from NiFi and upload it to the Catalog in Cloudera Data Flow.

Downloading a flow from Apache NiFi

When ready, you can download the Flow Definition from Apache NiFi, import it to the Cloudera Data Flow Catalog, and deploy the function on AWS, Azure, or the Google Cloud Platform (GCP) to run it in serverless mode. In Cloudera Data Flow Functions, only flows versioned in the Cloudera Data Flow Catalog can be used as functions.

The Download flow definition option in NiFi allows you to download the flow definition of a Process Group as a JSON file. After you have downloaded your flow from NiFi, you can import it into Cloudera Data Flow. For detailed instructions, see *Downloading a flow definition from NiFi* .

Related Information
Download a flow definition from NiFi

Uploading an Apache NiFi flow to the Catalog

Before an Apache NiFi flow can be run in serverless mode on a cloud provider function service, you must register it as a Cloudera Data Flow function by importing it into the Cloudera Data Flow Catalog. With Cloudera Data Flow Functions, only flows versioned in the Cloudera Data Flow Catalog can be used as functions.

If you want to use a NiFi flow in Cloudera Data Flow, you must import it to Cloudera Data Flow as a flow definition. When imported, the flow definition is added to the Catalog and you can start using it. For instructions, see *Importing a flow definition*.

Related Information

Importing a flow definition

Downloading Lambda Cloudera Data Flow Function binaries and uploading to S3

To be able to run the NiFi flow in AWS Lambda, you need to add the Cloudera Data Flow function handler libraries to S3.

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.
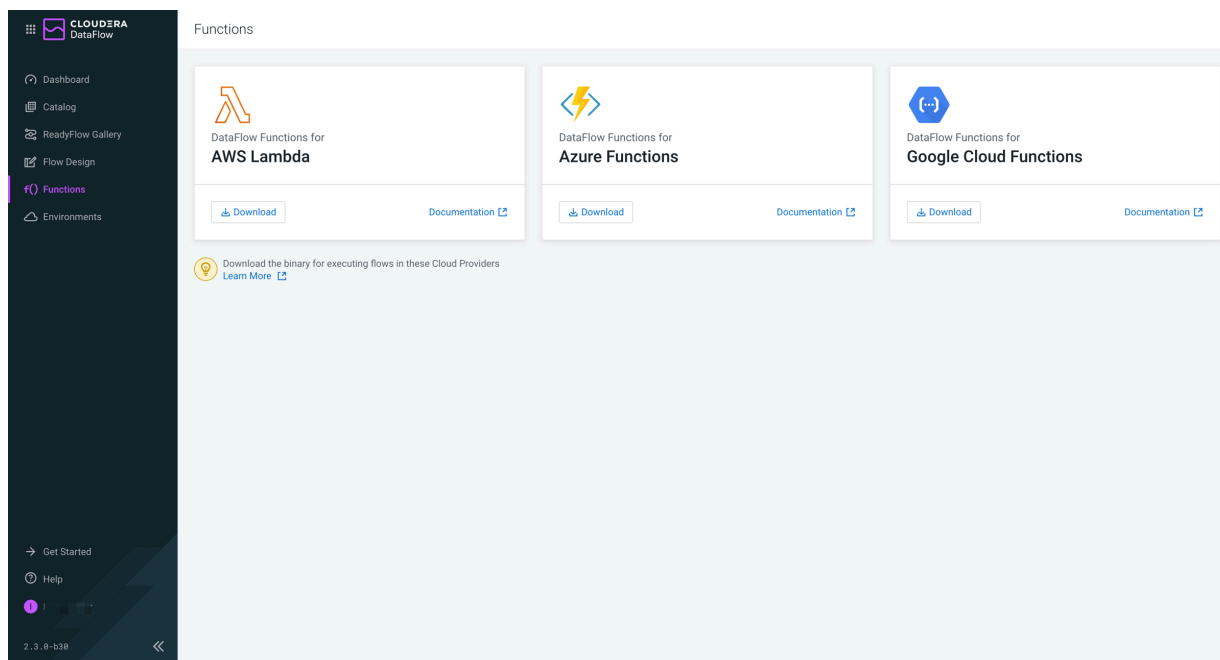
2. Upload this binary to an S3 bucket that you will later reference when creating the function in AWS Lambda.

    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.

Retrieving data flow CRN

When configuring your function on the cloud provider service page, you need to provide the Customer Resource Number (CRN) of the flow to be executed.

You can retrieve the CRN by checking the flow in the Cloudera Data Flow Catalog.

Note: The CRN must include the specific version of the flow that should be executed, so it should end with some version suffix such as /v.1.

Creating Cloudera service account

The first step of executing Cloudera Data Flow Functions code in your cloud environment is fetching the flow definition to be executed from the Cloudera Data Flow Catalog. For this, you need to create a service account and provision an access key.

Procedure

1. Navigate to the Management Console User Management Users .
2. From the Actions menu, select Create Machine User.
3. Provide a name and click Create.

   Note: Machine user names cannot start with a double underscore ("__").

The user details page is displayed showing information about the user.

Generating Access Key ID and Private Key

A Cloudera machine user must have API access credentials to access Cloudera services through the Cloudera CLI or API.

Procedure

1. Click the Roles tab on the user account details page.
2. Click Update Roles.

   The Update Roles pane is displayed.
3. Select the DFFunctionMachineUser role to assign it to your user.
4. Click Update.

   When the role is added, you should see:

5. Select the Access Keys tab on the user account details page.
6. Click Generate Access Key.

   The Generate Access Key modal window is displayed. it gives you an Access Key ID and a Private Key, which will be required when configuring your function.
7. Click Generate Access Key.

   A message is displayed that your access key has been successfully created.

   You can copy your Access Key ID and your Private Key. You will need these when configuring your function.
8. You can also download the credentials file into the .cdp directory in your user home directory. Or run the command cdp configure and enter the access key ID and private key to create a Cloudera credentials file in the same directory.

Results

When ready, you should see something like this: