

## Getting Ready to Run Functions

Date published: 2021-04-06

Date modified: 2024-06-03

# CLOUDERA

# 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

<b>Developing a NiFi flow for .....</b>	<b>4</b>
Flow design principles.....	4
Flow testing principles.....	7
<b>Downloading a flow from Apache NiFi.....</b>	<b>7</b>
<b>Uploading an Apache NiFi flow to the Catalog.....</b>	<b>8</b>
<b>Downloading Lambda Function binaries and uploading to S3.....</b>	<b>8</b>
<b>Retrieving data flow CRN.....</b>	<b>9</b>
<b>Creating service account.....</b>	<b>10</b>
<b>Generating Access Key ID and Private Key.....</b>	<b>10</b>

## Developing a NiFi flow for

A flow definition represents the data flow logic that you can download from NiFi, import to the 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 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 Catalog.
- Use Data Hub with the Flow Management template, if you are a customer who has a Data Lake.

For more information on how to set up a managed and secured Flow Management cluster in , 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 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 , 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 , 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.

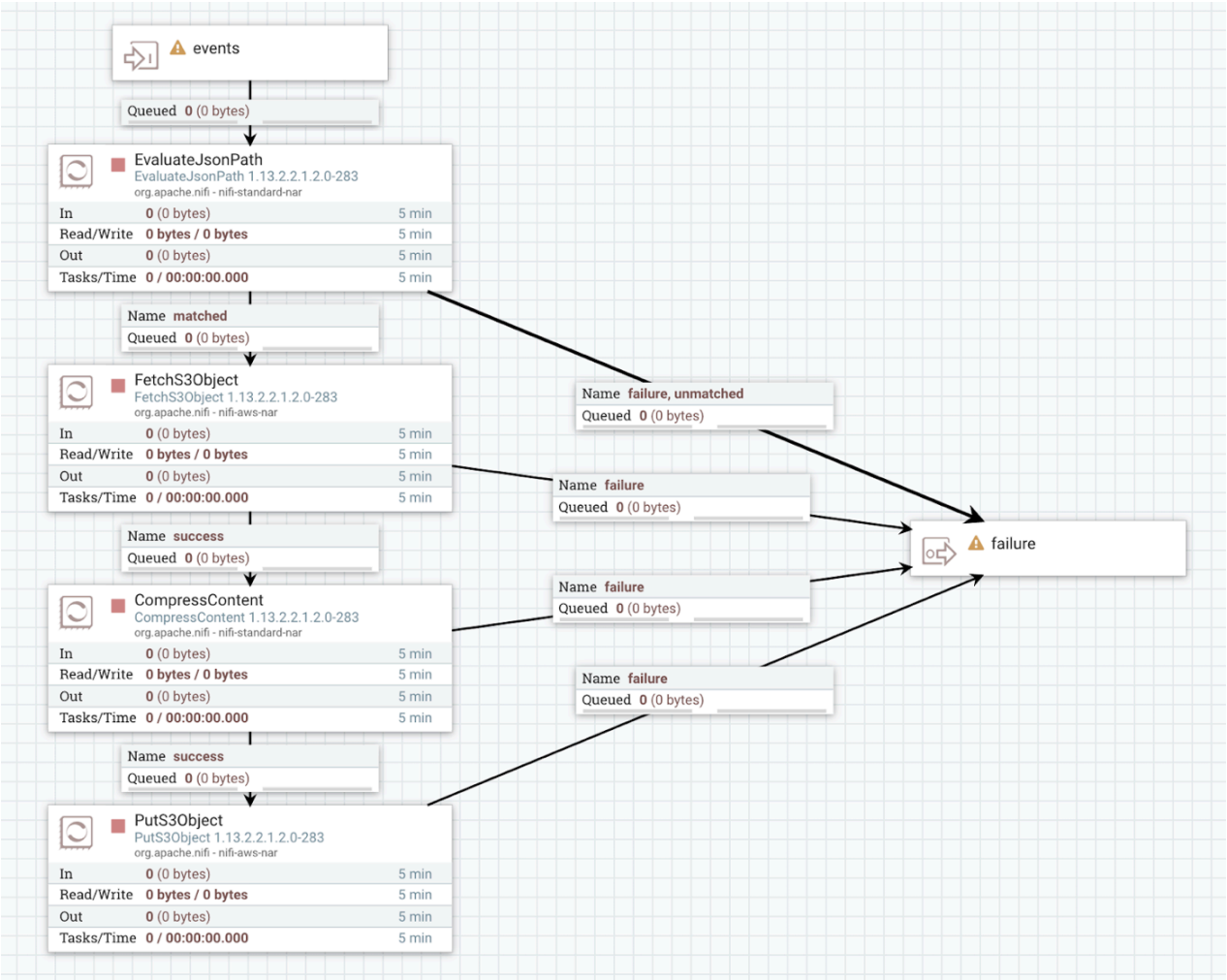
This is the start of your flow where the code will “generate” a FlowFile containing the payload generated by the function’s trigger. See the corresponding sections of each cloud provider in the documentation to see what the FlowFile’s content would be.

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:

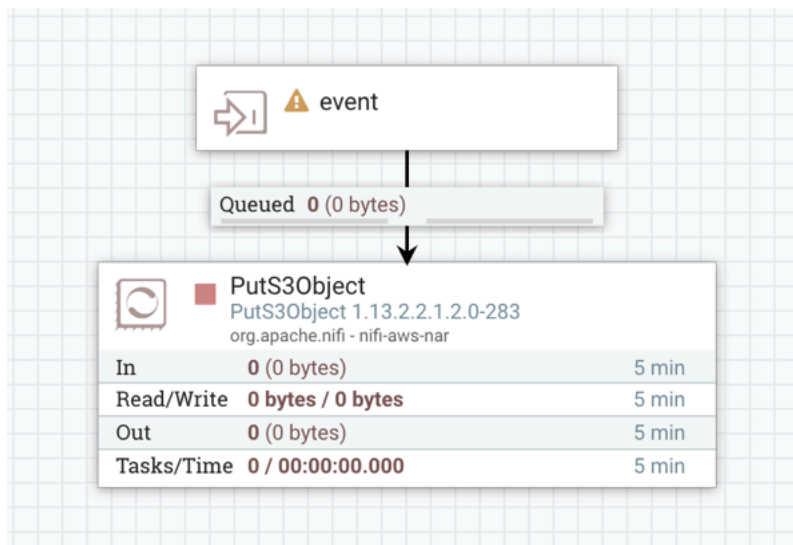
```

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:AIDAVRSV7HNNHFQQ0E2F4"
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+RSLx9syJmL40MCn06poZv1dJQJg5TgRx"
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": "A339RKRO79D9V"
26          },
27          "arn": "arn:aws:s3:::pvillard-naaf-input"
28        },
29        "object": {
30          "key": "04f56bcf-cb15-4244-9379-2abc7ac0cd8d",
31          "size": 1024,
32          "eTag": "265bc4c12162ffa179a58975728a033",
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 .

## Downloading a flow from Apache NiFi

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

If you want to use a NiFi flow in , you must import it to 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](#)

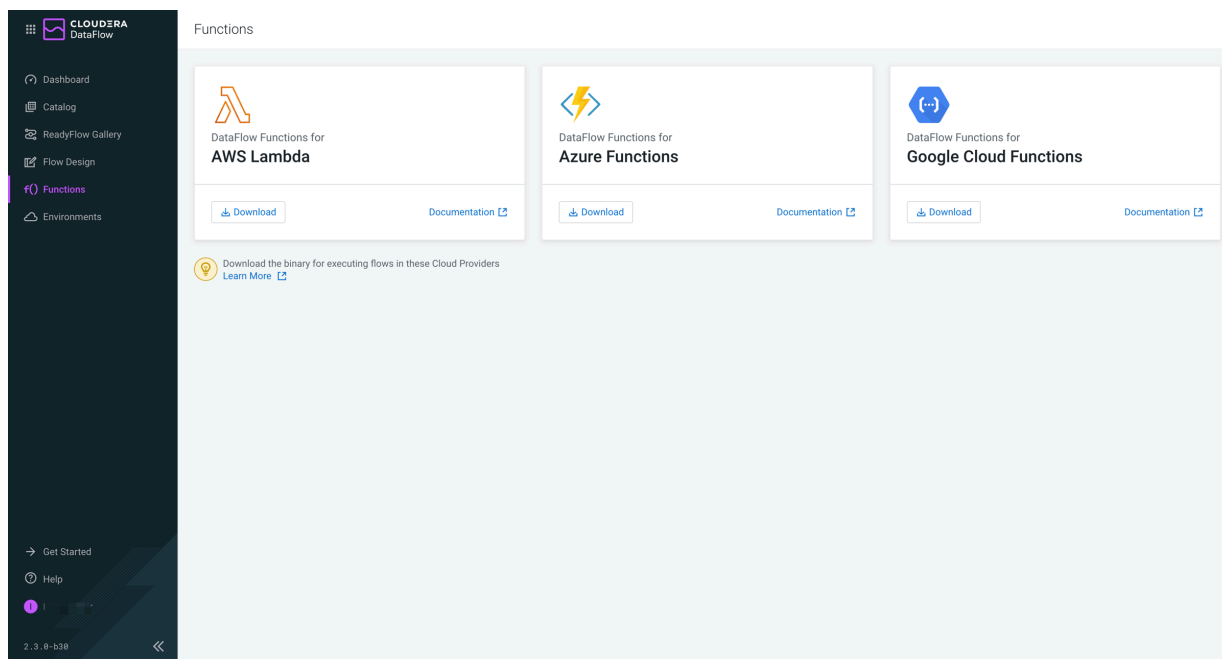
## Downloading Lambda Function binaries and uploading to S3

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

### 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.



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

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

## 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 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.

The screenshot shows the Cloudera DataFlow interface. On the left is a sidebar with navigation links: Dashboard, Catalog, ReadyFlow Gallery, Environments, Get Started, Help, and Pierre Villard. The main area is titled 'Flow Catalog' and contains a search bar and a list of flows. The flow 'NaaF - Trigger event to S3' is selected. To the right, the details of this flow are shown, including its description, CRN, and a table of versions. The 'Deploy' button is highlighted with an orange box, and the 'CRN #' field is also highlighted with an orange box.

## Creating service account

The first step of executing code in your cloud environment is fetching the flow definition to be executed from the 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 'Create Machine User' dialog box is shown. It has a title bar with a close button (X). Below the title bar is a text input field labeled 'Name' with the value 'naaf\_service\_account'. At the bottom right are two buttons: 'Cancel' and 'Create'.

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

## Generating Access Key ID and Private Key

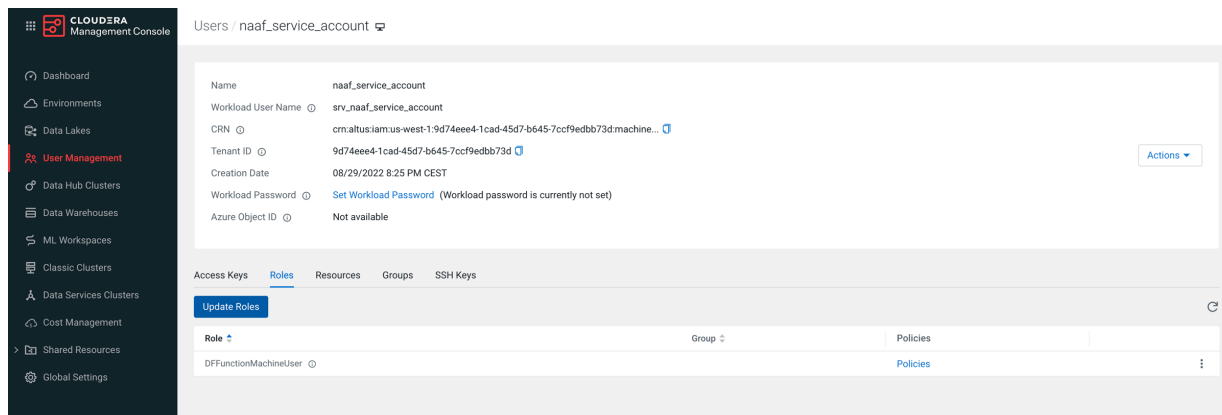
A machine user must have API access credentials to access services through the CLI or API.

### Procedure

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

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

When the role is added, you should see:



Users / naaf\_service\_account

Name: naaf\_service\_account  
 Workload User Name: srv\_naaf\_service\_account  
 CRN: cm.altus.iam.us-west-1:9d74eee4-1cad-45d7-b645-7ccf9edbb73d.machine...  
 Tenant ID: 9d74eee4-1cad-45d7-b645-7ccf9edbb73d  
 Creation Date: 08/29/2022 8:25 PM CEST  
 Workload Password: Set Workload Password (Workload password is currently not set)  
 Azure Object ID: Not available

Access Keys Roles Resources Groups SSH Keys

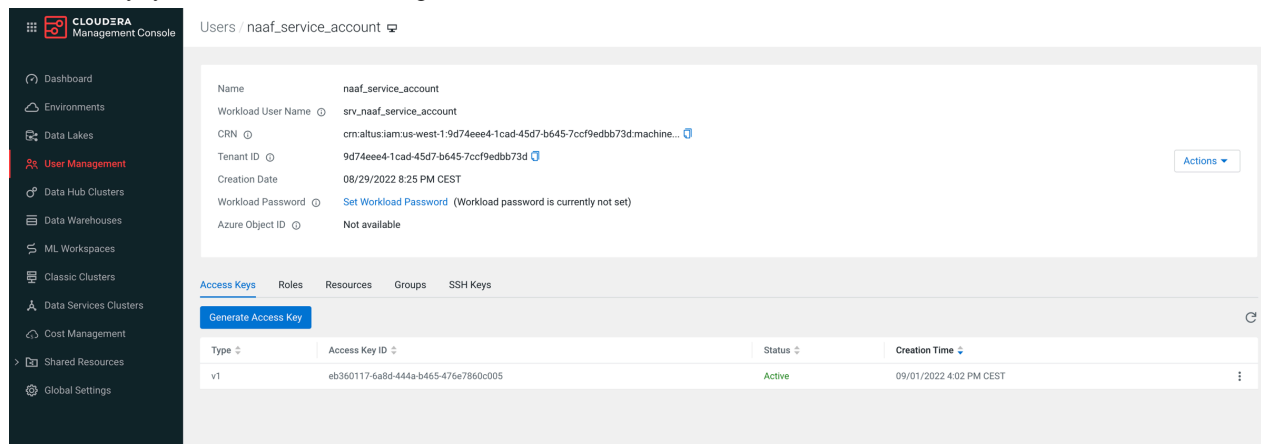
Update Roles

Role	Group	Policies
DFFunctionMachineUser		Policies

- Select the Access Keys tab on the user account details page.
- 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.
- 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.
- 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 credentials file in the same directory.

## Results

When ready, you should see something like this:



Users / naaf\_service\_account

Name: naaf\_service\_account  
 Workload User Name: srv\_naaf\_service\_account  
 CRN: cm.altus.iam.us-west-1:9d74eee4-1cad-45d7-b645-7ccf9edbb73d.machine...  
 Tenant ID: 9d74eee4-1cad-45d7-b645-7ccf9edbb73d  
 Creation Date: 08/29/2022 8:25 PM CEST  
 Workload Password: Set Workload Password (Workload password is currently not set)  
 Azure Object ID: Not available

Access Keys Roles Resources Groups SSH Keys

Generate Access Key

Type	Access Key ID	Status	Creation Time
v1	eb360117-6a8d-444a-b465-476e7860c005	Active	09/01/2022 4:02 PM CEST