

Cloudera DataFlow

DataFlow Functions Quickstart

Date published: 2021-04-06

Date modified: 2024-06-03

CLOUDERA

<https://docs.cloudera.com/>

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.....	4
Use case.....	4
Flow definition overview.....	4
Prerequisites for running the Quickstart.....	8
AWS Lambda.....	9
Azure Functions.....	13
Google Cloud Functions.....	15

Scope

This quickstart helps you to get a better understanding of DataFlow Functions and how it works. It also walks you through deploying your first function on any of the three main cloud providers (AWS Lambda, Azure Functions, and Google Cloud Functions).

Use this guide if you want to try deploying your first serverless NiFi flow. Get started with Cloudera DataFlow Functions by walking through a few simple steps.

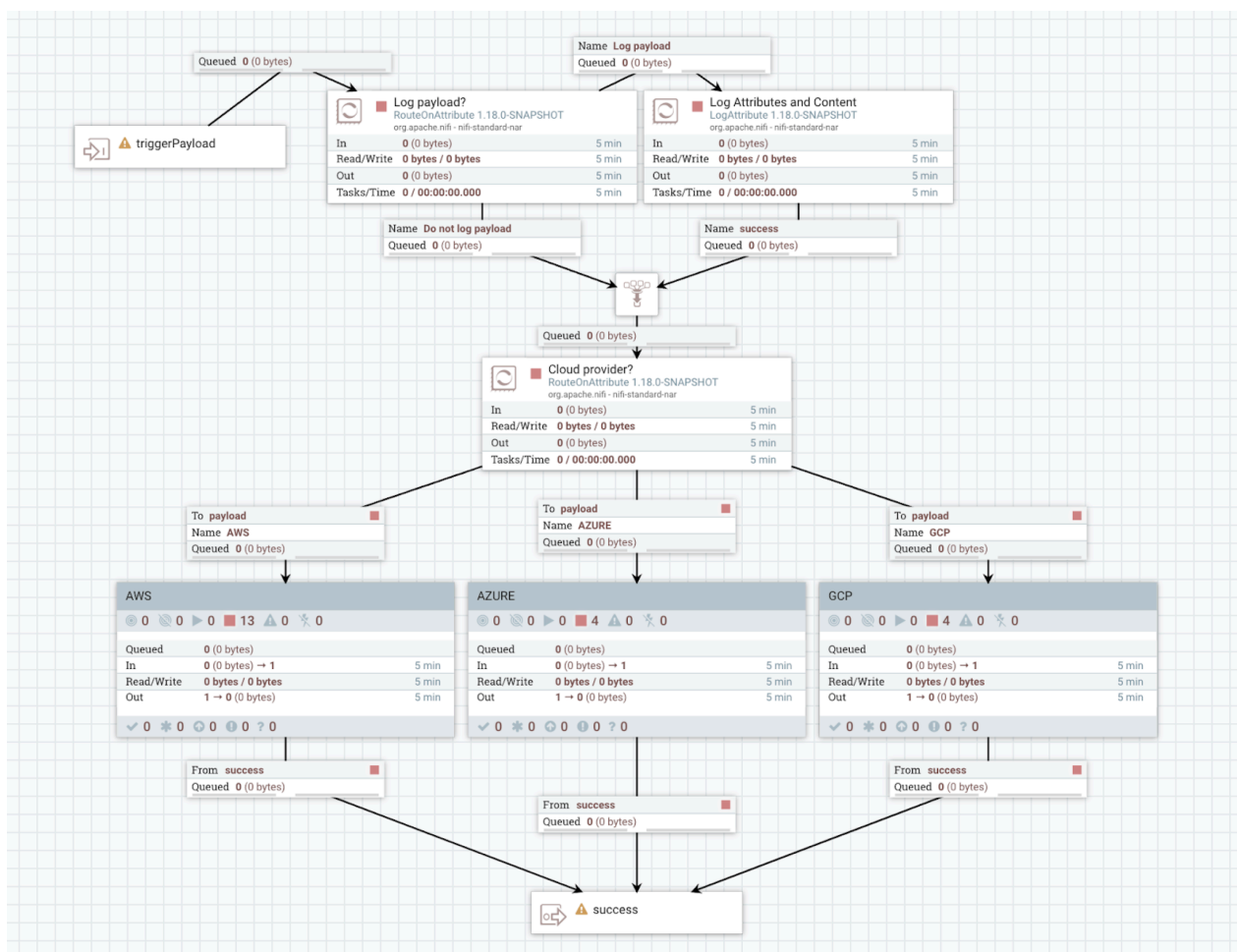
Use case

The quickstart is based on a 'Resize Image web service' use case.

This use case shows you how to deploy a serverless microservice to resize images using NiFi on any of the three main cloud providers. You deploy a function that is triggered by an HTTP call. In this HTTP call, you send an image with optional HTTP headers to specify the new dimensions of the image. The DataFlow function will return the resized image.

Flow definition overview

The NiFi flow definition used in this quickstart is based on Apache NiFi 1.18.0 and you can download this flow definition from [here](#).



The `triggerPayload` input port is where the DataFlow Functions framework is going to ingest the payload of the trigger that is generated by the cloud provider.

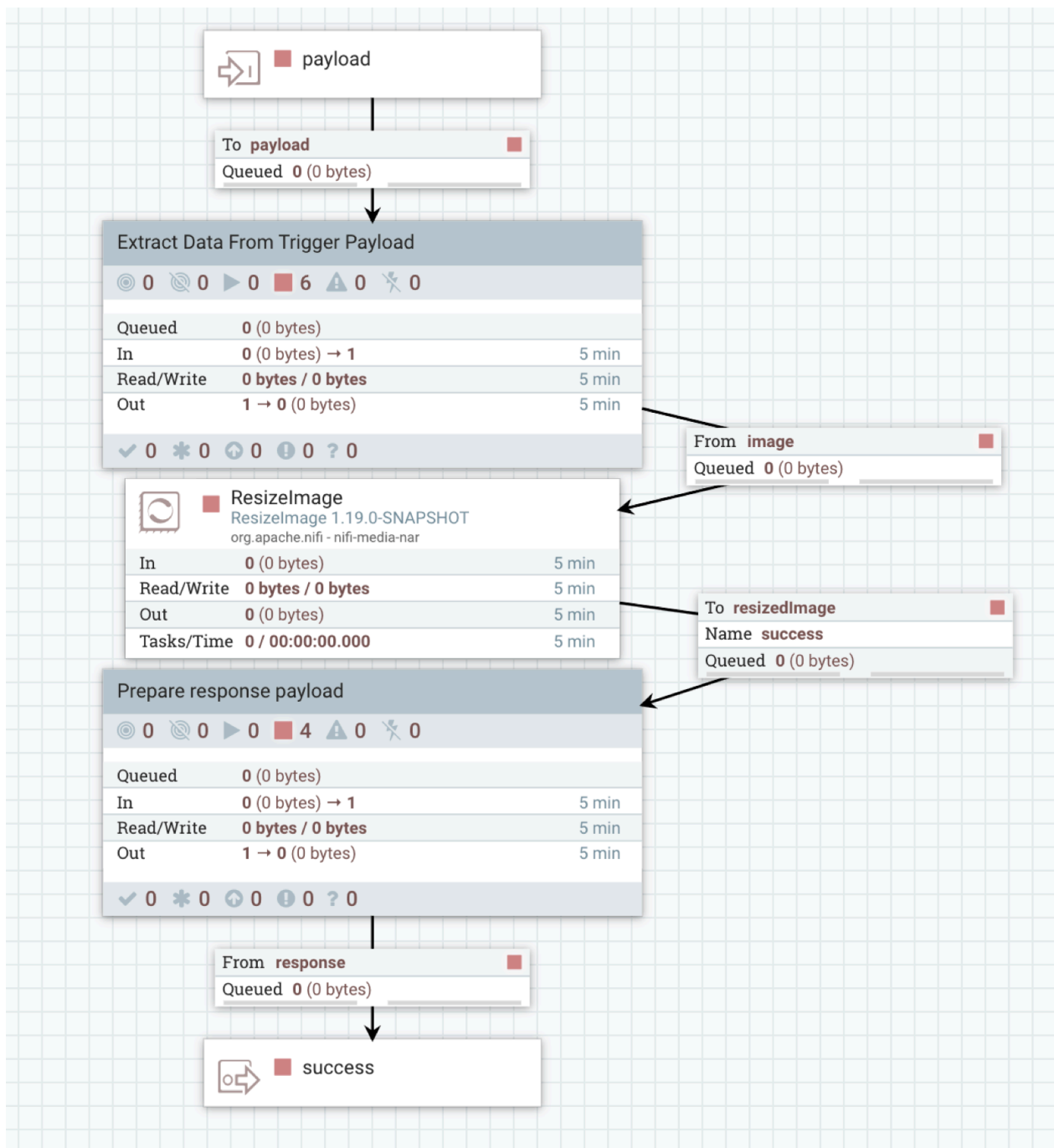
Then you decide whether or not you want to log the attributes and payload of the flowfile in the logs of the function. This is particularly useful when deploying the function for the first time as it helps you to understand what is going on and what is being generated. It would be disabled for a function running in production. The logging is enabled by default but you can turn it off by setting the below variable in the function's configuration: `logPayload = false`

Then you have a `RouteOnAttribute` processor that redirects the flowfile into a process group dedicated to each cloud provider depending on where the function is being deployed. The attribute containing the cloud provider name is automatically generated by the framework.



Note: This flow definition has been designed so that the same flow can be used on any cloud provider. If the function is deployed on a single cloud provider, the flow design could be greatly simplified by removing the parts related to the other cloud providers.

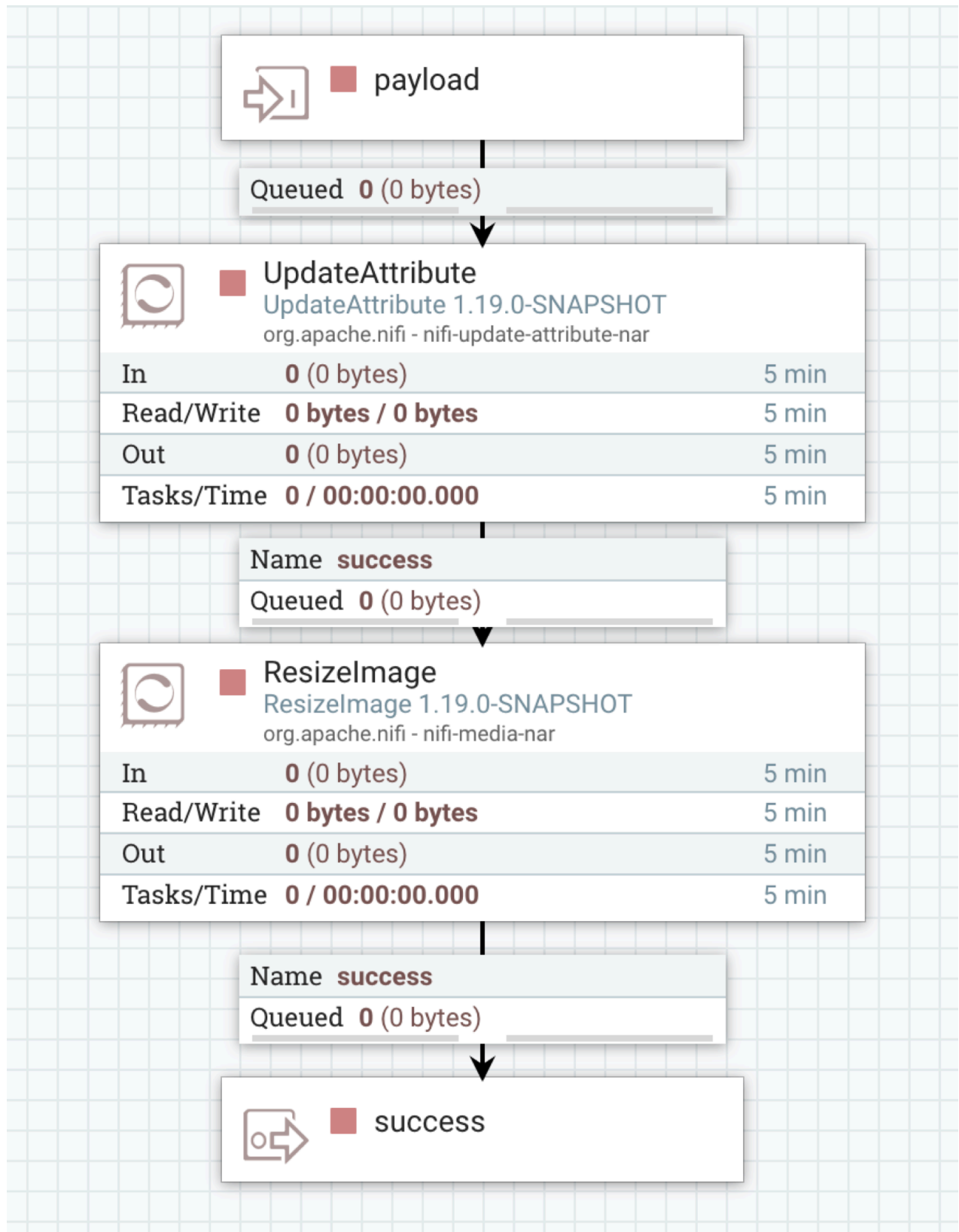
AWS Process Group



The payload of the flowfile generated by the API Gateway trigger is a JSON for which one of the fields will be the binary data of your image with base 64 encoding.

This is why you first need to extract the fields you are interested in, then do the base 64 decoding to have the binary data representing the image as the flow file content. After that you execute the `ResizeImage` processor, then re-encode the result with base 64 encoding and finally you generate the JSON payload that is expected by the AWS API Gateway to send back the response to the HTTP client.

Azure and GCP Process Groups



The binary data is the content of the generated flow file, so there is no need to do any conversion and you can directly use the ResizeImage processor based on the flow file attributes.



Note: For Azure Functions, the HTTP request has to contain the HTTP header Content-Type = application/octet-stream, so you have to override the mime.type attribute of the flow file before sending it to the ResizeImage to use an image friendly MIME type.

Prerequisites for running the Quickstart

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

Procedure

1. Get the CRN (Customer Resource Number) of the flow definition.
 - a. Download the flow definition for this use case and upload it into your DataFlow Catalog.
 - b. Once you have the flow definition in the Catalog, copy its CRN with its version.

For example: `crn:cdp:df:us-west-1:558bc1d3-8867-4357-8524-311d51289233:flow:Resize-Image/v.1`



Note: Make sure you have the full CRN with the flow version as the end. For more information, see [Retrieving data flow CRN](#).

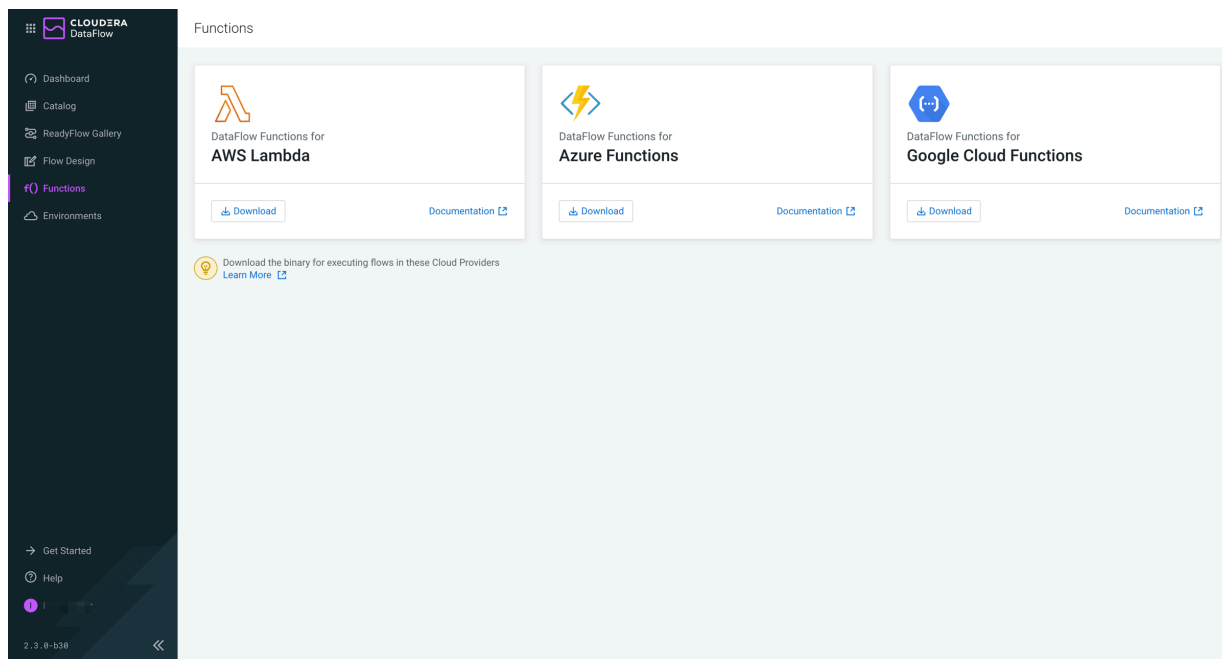
2. Make sure that you have a CDP machine user created.

This machine user is used when the function is triggered to retrieve the flow definition to be executed. For more information, see [Creating CDP service account](#).

3. Give the CDP machine user the appropriate role and generate the necessary keys.

For more information, see [Provisioning Access Key ID and Private Key](#).

4. Download the DataFlow Functions binary from the Functions page of Cloudera DataFlow for the cloud provider of your choice.



Note: This quickstart leverages the CLI of the cloud providers to make deploying the function straightforward. If you want to deploy your function using the control plane UI of the cloud provider, see the appropriate section of the DataFlow Functions documentation. The instructions provided in this quickstart are based on using the CLI of the cloud provider.

AWS Lambda

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

Before you begin

- You have the latest version of the AWS Lambda CLI.
- You have jq installed to extract some information from the JSON responses when executing commands.

Procedure

1. Start by setting all the parameters required for deploying your function using the AWS CLI.

```
# Name of the binary file you downloaded from Cloudera DataFlow. Change it
# based on the version you downloaded.
FILEKEY="naaf-aws-lambda-1.0.0.2.3.6.0-35-bin.zip"

# Location where the function will be deployed. Change this value to ma
# tch your needs/requirements.
LOCATION=eu-west-2

# Name of the bucket where the binary and NAR will be uploaded. Change thi
# s value to match your needs/requirements.
BUCKET_NAME=mydffresizeimagebucketpvillard

# Name of the function (only lower case letters). Change this value to mat
# ch your needs/requirements.
FUNCTIONNAME=resizeimagepvillard

# Flow CRN. Add the Customer Resource Number you obtained from the DataFlo
# w Catalog.
FLOW_CRN=***YOUR VALUE HERE***

# Credentials of the machine user
DF_ACCESS_KEY=***YOUR VALUE HERE***
DF_PRIVATE_KEY=***YOUR VALUE HERE***
```

2. Download the function template from [here](#) and make it available locally.
3. Execute the below commands:
 - a. Create your S3 bucket.

```
aws s3api create-bucket \
  --bucket $BUCKET_NAME \
  --region $LOCATION \
  --create-bucket-configuration LocationConstraint=$LOCATION
```

- b. Copy the Lambda DataFlow Function binary zip file to your bucket.

```
aws s3 cp $FILEKEY s3://$BUCKET_NAME/$FILEKEY
```

- c. Create the role for the Lambda.

```
ROLEARN=`aws iam create-role --role-name $FUNCTIONNAME\_lambda\_role --a
# ssume-role-policy-document '{"Version": "2012-10-17","Statement": [{ "Ef
# fect": "Allow", "Principal": {"Service": "lambda.amazonaws.com"}, "Actio
# n": "sts:AssumeRole"}]}' | jq -r ".Role.Arn" `
```

```
aws iam attach-role-policy --role-name $FUNCTIONNAME\_lambda\_role --
policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionR
ole
sleep 10
```

d. Create the function.

```
cp my-function-definition.json template.json

sed -i '.bak' -e 's|BUCKETNAME|"$BUCKET_NAME"|g' \
-e 's|FUNCTIONNAME|"$FUNCTIONNAME"|g' \
-e 's|FLOWCRN|"$FLOW_CRN"|g' \
-e 's|DFACCESSKEY|"$DF_ACCESS_KEY"|g' \
-e 's|DFPRIVATEKEY|"$DF_PRIVATE_KEY"|g' \
-e 's|LAYERARN|"$LAYERARNVERSION"|g' \
-e 's|FILEKEY|"$FILEKEY"|g' \
-e 's|ROLEARN|"$ROLEARN"|g' template.json
aws lambda create-function --cli-input-json file://template.json

echo "please wait a bit more (30 sec)..."
sleep 30

aws lambda publish-version --function-name $FUNCTIONNAME
```

When the execution is complete (it takes a few minutes), the function will be created and a version will be published but without a trigger.


4. Open the AWS Console UI and go to your Lambda.

5. Find to the published version of the function and add the API Gateway trigger:

Lambda > Add trigger

Add trigger

Trigger configuration

 **API Gateway**
api application-services aws serverless

Add an API to your Lambda function to create an HTTP endpoint that invokes your function. API Gateway supports two types of RESTful APIs: HTTP APIs and REST APIs. [Learn more](#)

Intent
Use an existing api or have us create one for you.

Create a new API

Use existing API

API type

HTTP API
Build low-latency and cost-effective REST APIs with built-in features such as OIDC and OAuth2, and native CORS support.

REST API
Develop a REST API where you gain complete control over the request and response along with API management capabilities.

Security
Configure the security mechanism for your API endpoint.

Open

▶ **Additional settings**

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

Cancel **Add**

This gives you the endpoint on which the Lambda is listening.

For example: `https://aa2q99qo3i.execute-api.eu-west-2.amazonaws.com/default/resizeimage`

You can now use a tool like Postman to send your picture.

You can add two custom headers (`resize-width` and `resize-height`) to specify the resizing dimensions. If not specified, the default values of the NiFi flow parameters are used.

The screenshot shows the Postman interface for a POST request. The URL is `https://cc2q4qz3l.execute-api.eu-west-2.amazonaws.com/default/resizeimage`. The Headers tab is active, showing the following headers:

Header	Value
Postman-Token	<calculated when request is sent>
Content-Type	image/png
Content-Length	<calculated when request is sent>
Host	<calculated when request is sent>
User-Agent	PostmanRuntime/7.29.2
Accept	*/*
Accept-Encoding	gzip, deflate, br
Connection	keep-alive
resize-width	400
resize-height	200
Key	Description

The Body tab shows a diagram of a pod architecture. It includes a 'MY-POD' container with an 'APP-CONTAINER' and a 'LOGGING-AGENT'. An external 'LOGGING-BACKEND' is also shown, connected to the 'LOGGING-AGENT'.



Note: The first attempt may fail due to a cold start and timeout (30 sec timeout on the API Gateway). A subsequent retry after a minute should work and should return the resized image much faster. For more information, see [Cold start](#).

It is possible to send a request to your function using curl.

For example:

```
curl -X POST https://3bftvfa66m.execute-api.us-east-2.amazonaws.com/default/cpresizeimage \
-H "Content-Type: image/png" \
-H "resize-width: 400" \
-H "resize-height: 200" \
--data-binary "@/Users/pierre/Desktop/test.png" \
--output /tmp/my_resized_image.png
```



Note: The function will be open to the internet. Make sure to secure it if you want to keep it running (this is done when creating the API Gateway trigger) or delete everything once you are done with this quickstart.

You can delete the function using the following command:

```
aws lambda delete-function --function-name $FUNCTIONNAME --region $LOCATION
aws iam detach-role-policy --role-name $FUNCTIONNAME\_lambda\_role --policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole
aws iam delete-role --role-name $FUNCTIONNAME\_lambda\_role

aws s3 rm s3://$BUCKET_NAME --recursive --region $LOCATION
aws s3api delete-bucket --bucket $BUCKET_NAME --region $LOCATION
rm template.json template.json.bak
```

You may also want to manually delete the API Gateway that you created earlier.

Azure Functions

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

Before you begin

- You have the latest version of the Azure Functions CLI.
- You have jq installed to extract some information from the JSON responses when executing commands.

Procedure

1. Start by setting all the parameters required for deploying your function using the Azure CLI.

```
# Name of the binary you download from the CDP Public Cloud control plane.
FILEKEY="NaafAzureFunctions.zip"

# Location where the function will be deployed. Change this value to match
your needs/requirements.
LOCATION=francecentral

# Name of the file share where the binary will be uploaded. Change this va
lue to match your needs/requirements.
BUCKET_NAME=mydffresizeimagebucket

# Name of the function (only lower case letters). Change this value to mat
ch your needs/requirements.
FUNCTIONNAME=resizeimageabc

# Flow CRN. Add the Customer Resource Number you obtained from the DataFlo
w Catalog.
FLOW_CRN=***YOUR VALUE HERE***

# Credentials of the machine user
DF_ACCESS_KEY=***YOUR VALUE HERE***
DF_PRIVATE_KEY=***YOUR VALUE HERE***

# Name of the resource group that is going to be created for the function.
Change this value to match your needs/requirements.
RESOURCEGROUP=MyDFFResourceGroup

# Subscription ID to use. Change this value to match your needs/requiremen
ts.
SUBSCRIPTIONID=***YOUR VALUE HERE***

# Name of the hosting plan that is going to be created for the function. C
hange this value to match your needs/requirements.
HOSTINGPLANNAME=MyHostingPlanDFF

# Name of the storage account that is going to be created (only lower case
letters). Change this value to match your needs/requirements.
STORAGEACCOUNTNAME=mydffstorageaccountname
```

2. Download the deployment template from [here](#).
3. Execute the below commands:
 - a. Create a resource group.

```
az group create -l $LOCATION -n $RESOURCEGROUP
```

```
echo "please
```

b. Deploy the Function App.

```
az deployment group create --resource-group $RESOURCEGROUP \
--template-file template.json \
--parameters \
  subscriptionId=$SUBSCRIPTIONID \
  name=$FUNCTIONNAME \
  location=$LOCATION \
  hostingPlanName=$HOSTINGPLANNAME \
  resourceGroup=$RESOURCEGROUP \
  storageAccountName=$STORAGEACCOUNTNAME \
  FLOW_CRN=$FLOW_CRN \
  DF_ACCESS_KEY=$DF_ACCESS_KEY \
  DF_PRIVATE_KEY=$DF_PRIVATE_KEY

echo "please wait a bit more (10 sec)..."
sleep 10
```

c. Retrieve the file share name created in the previous step.

```
FILESHARE=`az storage share list --account-name $STORAGEACCOUNTNAME | jq
-r '.[0].name'`
```

d. Create the right directories for the binary.

```
az storage directory create --account-name $STORAGEACCOUNTNAME --share-n
ame $FILESHARE --name data
az storage directory create --account-name $STORAGEACCOUNTNAME --share-
name $FILESHARE --name data/SitePackages
```

e. Upload the binary and the metadata file.

```
az storage file upload -s $FILESHARE --source $FILEKEY --path data/SiteP
ackages/ --account-name $STORAGEACCOUNTNAME
echo "$FILEKEY" > /tmp/packageName.txt && az storage file upload -s $FI
LESHARE --source /tmp/packageName.txt --path data/SitePackages/ --accoun
t-name $STORAGEACCOUNTNAME && rm /tmp/packageName.txt
```

When the execution is complete (it takes a few minutes), the function will be publicly listening on:

```
https://<function
```

```
name>.azurewebsites.net/api/StatelessNiFiHttpBinaryTriggerFunction
```

You can now use a tool like Postman to send your picture.

Make sure to override the Content-Type header with application/octet-stream.

You can add two custom headers (resize-width and resize-height) to specify the resizing dimensions. If not specified, the default values of the NiFi flow parameters are used.

The screenshot shows the Postman interface for a POST request. The URL is `https://dffresizeimage.azurewebsites.net/api/StatelessNiFiHttpBinaryTriggerFunction`. The Headers tab is selected, displaying the following headers:

Key	Value	Description
Content-Type	image/png	
Content-Length	<calculated when request is sent>	
Host	<calculated when request is sent>	
User-Agent	PostmanRuntime/7.29.2	
Accept	*	
Accept-Encoding	gzip, deflate, br	
Connection	keep-alive	
resize-width	400	
resize-height	200	
Content-Type	application/octet-stream	

The Body tab shows a diagram with a 'MY-POD' box containing 'APP-CONTAINER' and 'LOGGING AGENT', and a 'LOGGING BACKEND' cloud icon. Arrows indicate connections between the logging agent and the backend, and between the app container and the logging agent.



Note: The first attempt may fail due to a cold start and timeout. A subsequent retry after a minute should work and should return the resized image much faster. For more information, see [Cold start](#).

It is possible to send a request to your function using curl.

For example:

```
curl -X POST https://myresizeimage.azurewebsites.net/api/StatelessNiFiHttpBinaryTriggerFunction \
-H "Content-Type: application/octet-stream" \
-H "resize-width: 400" \
-H "resize-height: 200" \
--data-binary "@/Users/pierre/Desktop/test.png" \
--output /tmp/my_resized_image.png
```



Note: The function will be open to the internet. Make sure to secure it if you want to keep it running or delete everything once you are done with this quickstart. For more information, see [Securing your Azure Function App](#).

You can delete the function using the following command: `az group delete --name $RESOURCEGROUP --yes`

Google Cloud Functions

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

Before you begin

- You have the latest version of the Google Cloud Functions (GCP) CLI.
- The scripts assume that a default project has been associated to your local client: `gcloud config set project PROJECT_ID`.

Procedure

1. Start by setting all the parameters required for deploying your function using the GCP CLI.

```
# Name of the binary file you downloaded from the CDP Public Cloud control
# plane. Change it based on the version you downloaded.
FILEKEY="naaf-gcp-cloud-functions-1.0.0.2.3.6.0-35-bin.zip"

# Location where the function will be deployed. Change this value to match
# your needs/requirements.
LOCATION=europe-west1

# Name of the bucket where the binary and NAR will be uploaded. Change thi
# s value to match your needs/requirements.
BUCKET_NAME=mydffresizeimagebucket

# Name of the function (only lower case letters). Change this value to matc
# h your needs/requirements.
FUNCTIONNAME=resizeimage

# Flow CRN. Add the Customer Resource Number you obtained from the DataFlo
# w Catalog.
FLOW_CRN=***YOUR VALUE HERE***

# Credentials of the machine user
DF_ACCESS_KEY=***YOUR VALUE HERE***
DF_PRIVATE_KEY=***YOUR VALUE HERE***
```

2. Create a bucket where you will upload the DataFlow Functions binary.

```
gcloud alpha storage buckets create gs://$BUCKET_NAME --location $LOCATION
gsutil cp $FILEKEY gs://$BUCKET_NAME/$FILEKEY
```

3. Deploy the function.

```
gcloud functions deploy $FUNCTIONNAME \
  --gen2 \
  --region=$LOCATION \
  --runtime=java11 \
  --source=gs://$BUCKET_NAME/$FILEKEY \
  --memory=1024MB \
  --timeout=300 \
  --trigger-http \
  --allow-unauthenticated \
  --entry-point=com.cloudera.naaf.gcp.cloud.functions.StatelessNiFiHttpFu
  nction \
```



```
--set-env-vars=FLOW_CRN=$FLOW_CRN,DF_PRIVATE_KEY=$DF_PRIVATE_KEY,DF_ACCESS_KEY=$DF_ACCESS_KEY,NEXUS_URL=https://maven-central.storage-download.googleapis.com/maven2,STORAGE_BUCKET=$BUCKET_NAME,OUTPUT_PORT=success
```

After a couple of minutes, the function will be deployed and you will receive an HTTP endpoint exposed by Google Cloud to call your function.

For example: `https://resizeimage-vqz99abcdea-ew.a.run.app`

You can now use a tool like Postman to send your picture.

You can add two custom headers (`resize-width` and `resize-height`) to specify the resizing dimensions. If not specified, the default values of the NiFi flow parameters are used.

The screenshot shows the Postman interface for a POST request to `https://resizeimage-vqz99abcdea-ew.a.run.app`. The Headers tab is active, showing a list of headers with checkboxes. The headers include: Postman-Token, Content-Type (image/png), Content-Length, Host, User-Agent, Accept, Accept-Encoding, Connection, resize-width (400), and resize-height (200). The Body tab is also visible, showing a diagram of a container architecture with components like MY-POD, APP-CONTAINER, LOGGING AGENT, and LOGGING BACKEND.



Note: The first attempt may fail due to cold start and timeout. A subsequent retry should work and should return the resized image much faster. For more information, see [Cold start](#).

It is possible to send a request to your function using curl.

For example:

```
curl -X POST https://resizeimage-vqz99abcdea-ew.a.run.app \
-H "Content-Type: image/png" \
-H "resize-width: 400" \
-H "resize-height: 200" \
--data-binary "@/Users/pierre/Desktop/test.png" \
--output /tmp/my_resized_image.png
```



Note: The function will be open to the internet. Make sure to secure it if you want to keep it running or delete everything once you have completed this quickstart.

You can secure the function by deploying it without the `--allow-unauthenticated` parameter.

You can delete the function using the following command:

```
gcloud functions delete $FUNCTIONNAME --gen2 --region=$LOCATION --quiet
gcloud alpha storage rm -r gs://$BUCKET_NAME
```