

Top Tasks

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.

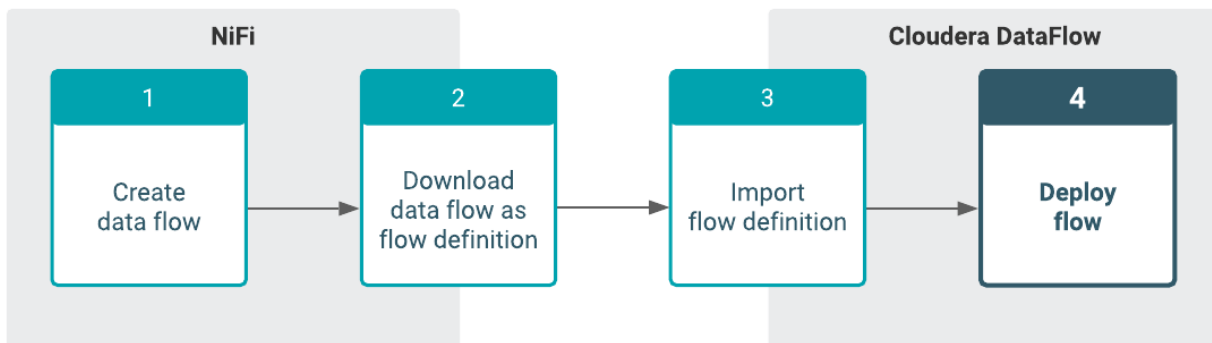
Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES 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

Deploying a flow definition using the wizard.....	4
Select the flow definition version you want to deploy from the catalog.....	4
Launch the deployment wizard.....	4
Name your flow deployment and assign it to a project.....	5
Configure NiFi.....	6
Provide parameter values.....	7
Configure sizing and scaling.....	7
Set Key performance indicators.....	7
Verify your settings and initiate deployment.....	8
 Creating a Cloudera DataFlow function in AWS.....	 8
Import a flow definition.....	8
Retrieving data flow CRN.....	9
Creating Cloudera service account.....	10
Generating Access Key ID and Private Key.....	10
Creating a Lambda function.....	11
Configuring your Lambda function.....	13
 Tutorial: Building a new flow from scratch.....	 18
Create a new flow.....	19
Create controller services.....	23
Build your draft flow.....	28
Start a test session.....	40
Publish your flow definition to the Catalog.....	42

Deploying a flow definition using the wizard

Deploy a flow definition to run Apache NiFi flows as flow deployments in Cloudera DataFlow. To do this, launch the Deployment wizard and specify your environment, parameters, sizing, and KPIs.



Before you begin

- You have an enabled and healthy Cloudera DataFlow environment.
- You have been assigned the DFCatalogAdmin or DFCatalogViewer role granting you access to the Catalog.
- The flow definition you want to deploy has been added to the Catalog by someone with DFCatalogAdmin role.
- You have been assigned the DFFlowAdmin role for the environment to which you want to deploy the flow definition.
- You have been assigned DFProjectMember role for the Project where you want to deploy the flow definition.
- If you are deploying custom processors or controller services, you may need to meet additional prerequisites.

Select the flow definition version you want to deploy from the catalog

The Catalog is where you manage the flow definition lifecycle, from initial import, to versioning, to deploying a flow definition.

Procedure

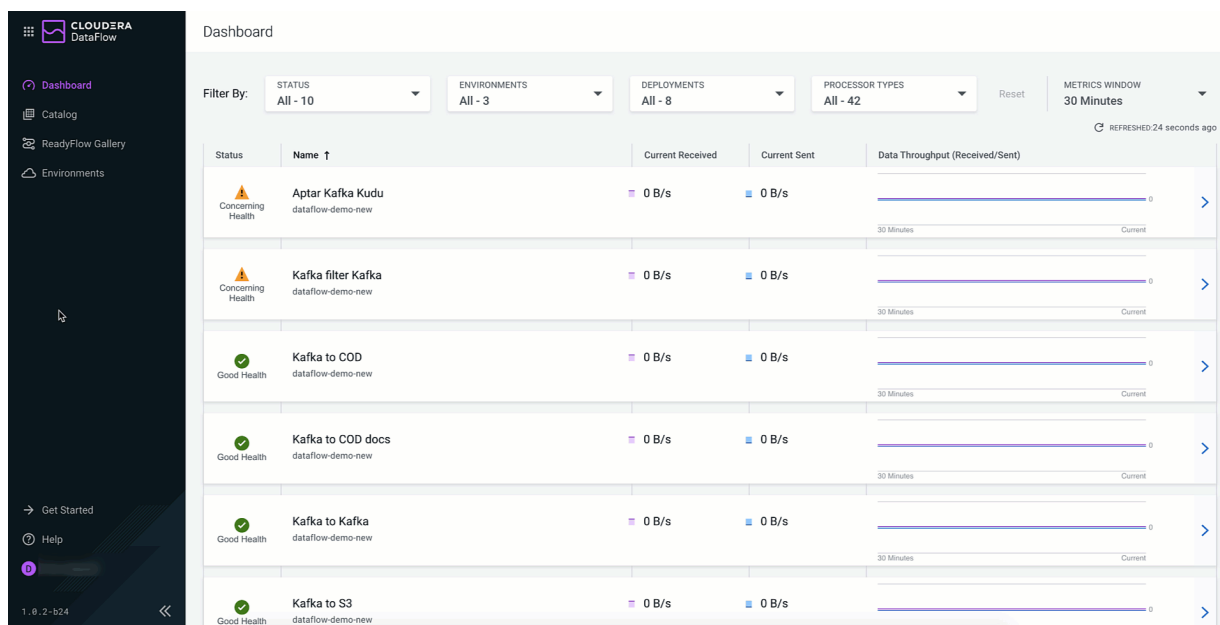
1. In Cloudera DataFlow, select Catalog from the left navigation pane.
Flow definitions available for you to deploy are displayed, one definition per row.
2. Select a row to display the flow definition details and available versions.
The flow details pane opens on the right.

Launch the deployment wizard

After selecting a flow definition version from the catalog, you need to select an environment, provide a deployment name and assign it to a project using the deployment wizard.

Procedure

1. Click Deploy to launch the Deployment wizard.



2. Select the environment where you want to deploy the flow.



Note:

Only environments which have been enabled for Cloudera DataFlow, are in a healthy state, and to which you have access show up in the dropdown. Once you have selected the target environment, you cannot change it later in the flow deployment process without starting over.

3. Click Deploy.

Name your flow deployment and assign it to a project

After selecting the flow version and an environment, the deployment wizard takes you to the Overview page. Here you need to provide a name for your flow deployment and assign it to a project. At this point you can also import a previously exported deployment configuration, auto-filling configuration values and thus speeding up deployment.

Procedure

1. Give your flow a unique Deployment Name.

You can use this name to distinguish between different versions of a flow definition, flow definitions deployed to different environments, and similar.



Note:

Flow Deployment names need to be unique within an Environment. The Deployment wizard indicates whether a name is valid by displaying a green check below the Deployment Name text box.

2. Select a Target Project for your flow deployment from the list of Projects available to you.

- If you do not want to assign the deployment to any of the available Projects, select Unassigned. Unassigned deployments are accessible to every user with DFFlowUser role in the environment.
- This field is automatically populated if you import a configuration and the Project referenced there exists in your environment, and you have access to it.

3. If you have previously exported a deployment configuration that closely aligns with the one you are about to deploy, you can import it under Import Configuration to auto-fill as much of the wizard as possible.
You can later manually modify auto-filled configuration values during deployment.
4. Click Next.

Configure NiFi

After selecting the target environment, project, and naming your flow, you need to set Apache NiFi version, possible inbound connections, and custom processors. Depending on the flow definition, you may also need to provide values for a number of configuration parameters. Finally, you need to set the capacity of the NiFi cluster servicing your deployment.

Procedure

1. Pick a NiFi Runtime Version for your flow deployment.

Select if you want to use Apache NiFi 1.x or 2.x with your deployment.



Important: NiFi 2.x is currently provided as a technical preview feature, do not use it for deployments in production environments.

Cloudera recommends that you always use the latest available version within the 1.x and 2.x lines, if possible.

2. Specify whether you want the flow deployment to auto-start once deployed.
3. Specify whether you want to use Inbound Connections that allow your flow deployment receiving data from an external data source.

If yes, specify the endpoint host name and listening port(s) where your flow deployment listens to incoming data.

See *Creating an inbound connection endpoint* for complete information on endpoint configuration options.

4. Specify whether you want to use NiFi Archives (NARs) to deploy custom NiFi processors or controller services.

If yes, specify the CDP Workload Username, password, and cloud storage location you used when preparing to deploy custom processors.



Tip: If you want to provide a machine user as CDP Workload Username during flow deployment, make sure to note the full workload user name including the `srv_` prefix.

Make sure that you click the Apply button specific to Custom NAR Configuration before proceeding.

5. If you selected to run your flow with NiFi 2.x [Technical Preview], specify whether you want to use custom Python processors with your flow deployment.

If yes, specify the CDP Workload Username, password, and cloud storage location where the processors are stored.



Tip: Create a dedicated directory in your object store where you keep all your Python processors. Create one Python script per processor and store it in this directory.



Tip: If you want to provide a machine user as CDP Workload Username during flow deployment, make sure to note the full workload user name including the `srv_` prefix.

Make sure that you click the Apply button specific to Custom Python Processors before proceeding.

6. Click Next.

Related Information

[Inbound connections](#)

Provide parameter values

Depending on the flow you deploy, you may need to specify parameter values like connection strings, usernames and similar, and upload files like truststores, JARs, and similar.

Procedure

1. Provide values to parameters required for your flow deployment.

You have to provide values for all parameters. You can filter for the still empty fields by selecting the No value checkbox.



Tip: If you are deploying a ReadyFlow, you can learn about required parameters and instructions on how to obtain parameter values by checking *Prerequisites* and *Required parameters* in the documentation of the respective ReadyFlow.

2. When you finished setting configuration parameters, click Next.

Configure sizing and scaling

Set the size and number of Apache NiFi nodes, auto-scaling, and the type of storage to be used.

Procedure

1. Specify NiFi node size.

Select one of the following options:

- Extra Small: 2 vCores per Node, 4 GB per Node
- Small: 3 vCores per Node, 6 GB per Node
- Medium: 6 vCores per Node, 12 GB per Node
- Large: 12 vCores per Node, 24 GB per Node

2. Set the number of NiFi nodes and auto-scaling.

- You can set whether you want to automatically scale your cluster according to flow deployment capacity requirements. When you enable auto-scaling, the minimum number of NiFi nodes are used for initial size and the workload scales up or down depending on resource demands.
- You can set the number of nodes between 1 and 32.
- You can set whether you want to enable Flow Metrics Scaling.

3. Select storage type.

Select whether you want your deployment to use storage optimized for cost or for performance.

- Standard: 512 GB Content Repo Size, 512 GB Provenance Repo Size, 256 GB Flow File Repo Size, 2300 IOPS, 150 MB/s Max Throughput
- Performance: 1024 GB Content Repo Size, 1024 GB Provenance Repo Size, 256 GB Flow File Repo Size, 5000 IOPS, 200 MB/s Max Throughput

4. Click Next.

Related Information

[Auto-scaling](#)

Set Key performance indicators

Optionally add key performance indicators to help you track the performance of your flow deployment then review your settings and launch the deployment process.

Procedure

1. From KPIs, you may choose to identify key performance indicators (KPIs), the metrics to track those KPIs, and when and how to receive alerts about the KPI metrics tracking.

**Tip:**

You can reorder the KPIs by dragging them up or down the list. The order you configure here will be reflected in the detailed monitoring view of the resulting deployment.

See *Working with KPIs* for complete information about the KPIs available to you and how to monitor them.

2. Click Next.

Related Information

[Working with KPIs](#)

Verify your settings and initiate deployment

Review deployment settings, make any necessary changes, and start deployment.

Procedure

1. Review a summary of the information provided and make any necessary edits by clicking Previous.
2. When you are finished, complete your flow deployment by clicking Deploy.

**Tip:**

Click View CLI Command to see the equivalent Cloudera CLI syntax in a help pane.

Results

After you click Deploy, you are redirected to the **Alerts** tab in the **Flow Details** where you can track how the deployment progresses.

Creating a Cloudera DataFlow function in AWS

You are all set up for creating your first Cloudera DataFlow function: you have a flow definition created in and downloaded from NiFi, a Cloudera DataFlow environment, and the appropriate rights to that environment. All you need now is to follow these steps to get your function up and running.

Prerequisites

- You have a data flow created in Apache NiFi and downloaded as a flow definition JSON file.
- You have an enabled and healthy Cloudera DataFlow environment.
- You have been assigned the DFCatalogAdmin role granting you access to the Catalog.
- You have been assigned the DFFlowAdmin role for the environment to which you want to deploy the flow definition.
- An AWS user account is required with access policies that has permissions to list and create buckets, roles, and Lambda functions.

Import a flow definition

If you want to use an Apache NiFi flow in Cloudera DataFlow, you must import it as a flow definition. When imported, the flow definition is added to the Flow Catalog.

Steps

1. Open Cloudera DataFlow by clicking the DataFlow tile in the Cloudera sidebar.
2. Click Catalog from the left navigation pane.

The Flow Catalog page is displayed. Previously imported flow definitions are displayed, one definition per row.

3. Click Import Flow Definition.

Flow Catalog

Search by name

Import Flow Definition

REFRESHED 23 seconds ago

Name ↑	Type	Versions	Last Updated
Kafka to Hive Sensors	Custom Flow Definition	1	6 months ago
Kafka to Kafka	ReadyFlow	1	2 days ago
Kafka to Kudu	ReadyFlow	1	2 days ago
Kafka to Kudu Sensors	Custom Flow Definition	2	6 months ago
Kafka to S3	Custom Flow Definition	7	4 months ago
Kafka to S3 Avro	ReadyFlow	1	a day ago
Kafka to S3 demo	Custom Flow Definition	1	21 hours ago

4. Provide a name (Flow Name) and a description (Flow Description) for the flow definition.
5. In the NiFi Flow Configuration field, select the JSON file you downloaded from NiFi.

Alternatively, you can drop the JSON file into this field to select it.

6. Optional: You can add comments to the flow definition version you are importing (Version Comments).



Note: When you are importing the first version of a flow definition, the Version Comments field contains Initial Version by default. You can change this comment if needed.

7. Click Import.

Result

You have successfully imported your NiFi flow to Cloudera DataFlow. It is available in the Flow Catalog as the first version of your flow definition.

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 DataFlow 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, and Environments. 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. It includes a description: 'This flow just takes the payload of the FlowFile generated by the function trigger and send it into S3. This function does not have any interest other than looking at what the payload looks like for any trigger.' It also shows the CRN # and a table of deployments. The 'Deploy' button is highlighted with an orange box, and the 'Download' button is also highlighted. The 'CRN #' field is also highlighted with an orange box.

Creating Cloudera service account

The first step of executing Cloudera DataFlow Functions code in your cloud environment is fetching the flow definition to be executed from the Cloudera DataFlow 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 screenshot shows a 'Create Machine User' dialog box. It has a close button (X) in the top right corner. Below the title, there is a label '* Name' followed by a help icon. A text input field contains the value 'naaf_service_account'. At the bottom right, there 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 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:

The screenshot shows the Cloudera Management Console interface. On the left is a navigation menu with options like Dashboard, Environments, Data Lakes, User Management (highlighted), Data Hub Clusters, Data Warehouses, ML Workspaces, Classic Clusters, Data Services Clusters, Cost Management, Shared Resources, and Global Settings. The main content area is titled 'Users / naaf_service_account'. It displays user details for 'naaf_service_account', including Workload User Name, CRN, Tenant ID, Creation Date, Workload Password, and Azure Object ID. Below this, there are tabs for Access Keys, Roles, Resources, Groups, and SSH Keys. The 'Roles' tab is active, showing a table with one role: 'DFFunctionMachineUser'. An 'Update Roles' button is located above the table.

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:

The screenshot shows the Cloudera Management Console interface. On the left is a navigation menu with options like Dashboard, Environments, Data Lakes, User Management (highlighted), Data Hub Clusters, Data Warehouses, ML Workspaces, Classic Clusters, Data Services Clusters, Cost Management, Shared Resources, and Global Settings. The main content area is titled 'Users / naaf_service_account'. It displays user details for 'naaf_service_account'. Below this, there are tabs for Access Keys, Roles, Resources, Groups, and SSH Keys. The 'Access Keys' tab is active, showing a table with one access key: 'v1'. A 'Generate Access Key' button is located above the table.

Creating a Lambda function

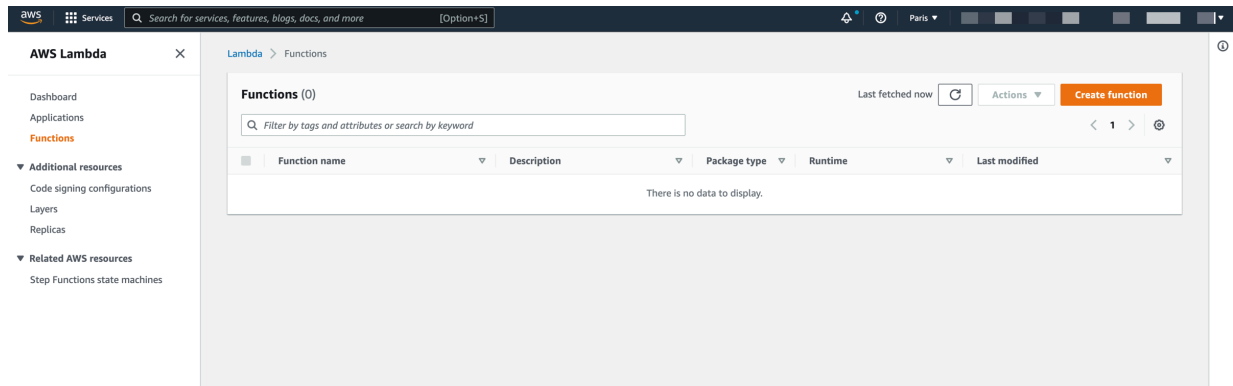
AWS Lambda is a serverless, event-driven compute service that lets you run code for virtually any type of application or backend service without provisioning or managing servers.

About this task

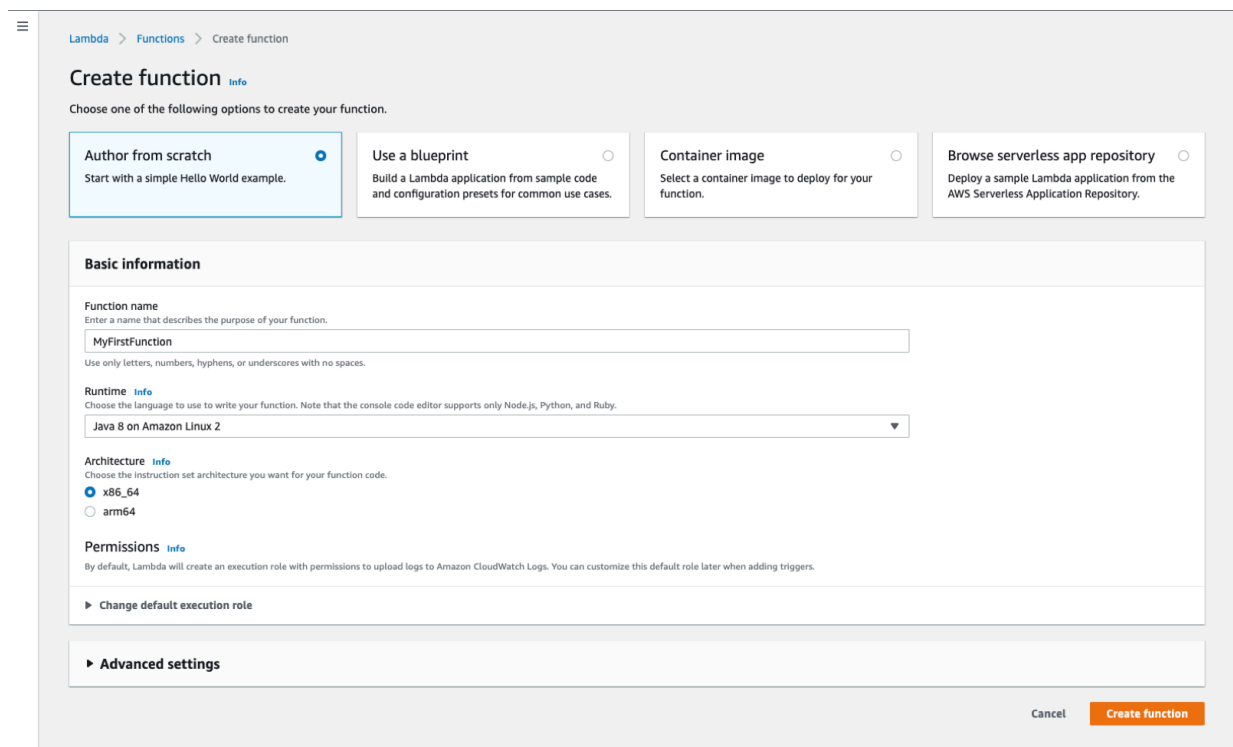
Follow these steps to create an AWS Lambda function that is able to run Cloudera DataFlow Functions:

Procedure

1. Navigate to the AWS Lambda page in your cloud account in the region of your choice.



2. Click Create function.
The Create function page opens.
3. Fill in the Basic information section:
 - a) Keep the default Author from scratch option selected.
 - b) Give your function a meaningful name for your use case.
 - c) Select Java 8 on Amazon Linux 2 for Runtime.
 - d) Choose x86_64 for Architecture.
 - e) Click Create function.



Your function is created and its details are displayed.

4. In the Code source section of the Code tab, click Upload from Amazon S3 location .
The Upload a file from Amazon S3 modal window appears.

- Provide the S3 URL of Cloudera DataFlow Function binaries in the textbox.



Note: You have downloaded the Cloudera DataFlow Function handler libraries as a zip of binaries from Cloudera DataFlow and uploaded them to an S3 bucket when getting ready to run functions. AWS Lambda will use these binaries to run the NiFi flow. For instructions, see *Downloading Lambda Cloudera DataFlow Function binaries and uploading to S3*.

- Click Save.
- Navigate to the Runtime Settings on the Code tab and click Edit to change the Handler property to: `com.cloudera.naaf.aws.lambda.StatelessNiFiFunctionHandler::handleRequest`

The details of your first function:

What to do next

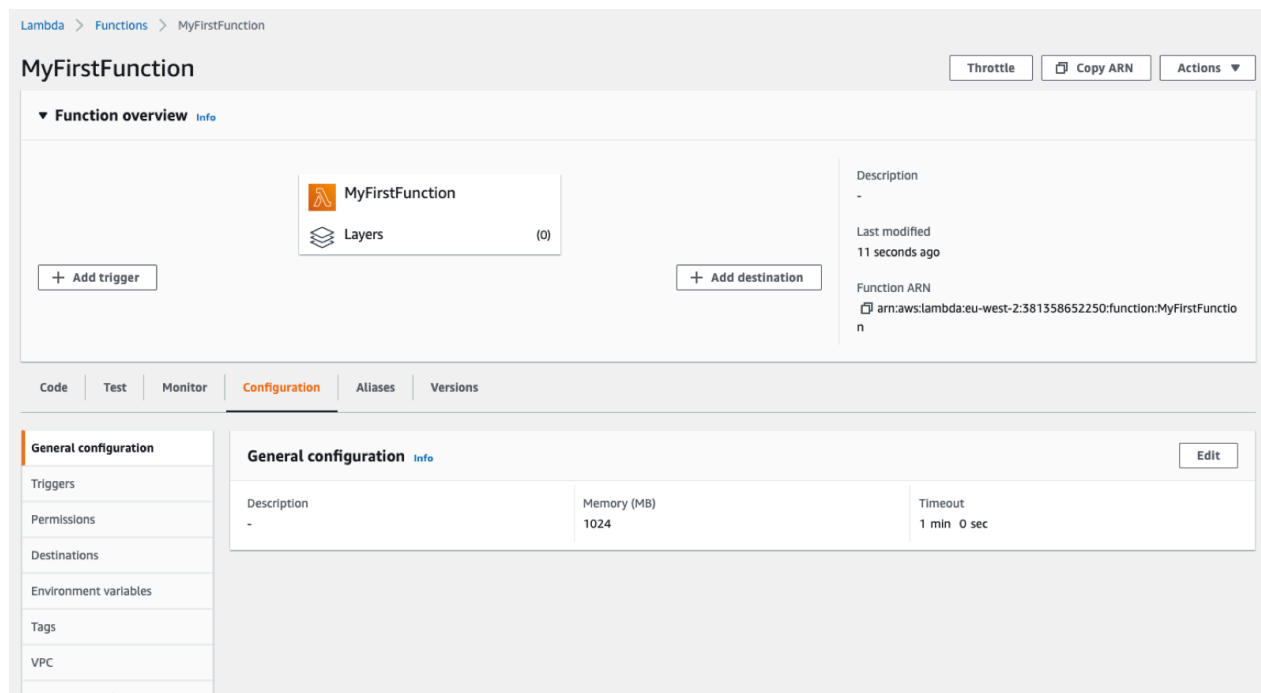
You can now start configuring your Lambda specifically for your flow and for your use case.

Configuring your Lambda function

After you create a function, you can use the built-in configuration options to control its behavior. You can configure additional capabilities, adjust resources associated with your function, such as memory and timeout, or you can also create and edit test events to test your function using the console.

Memory and timeout in runtime configuration

You can configure two very important elements, memory and timeout on the Configuration General configuration tab of the Lambda function details screen.



The appropriate values for these two settings depend on the data flow to be deployed. Typically, it is recommended to start with 1536-2048 MB (1.5 - 2 GB) memory allocated to Lambda. This amount of memory may be more than necessary or less than required. The default value of 512 MB may be perfectly fine for many use cases. You can adjust this value later as you see how much memory your function needs.

While Cloudera DataFlow Functions tend to perform very well during a warm start, a cold start that must source extensions (processors, controller services, and so on) may take several seconds to initialize. So it is recommended to set the Timeout to at least 30 seconds. If the data flow to run reaches out to many other services or performs complex computation, it may be necessary to use a larger value, even several minutes.

Click Edit if you want to change the default values.

See the corresponding [AWS documentation](#) to learn more about timeout duration and memory sizing, which determines the amount of resources provisioned for your function.

Runtime environment variables

You must configure the function to specify which data flow to run and add any necessary runtime configuration using environment variables.

Procedure

1. Navigate to the Configuration tab and select Environment variables from the menu on the left.
2. Click Edit on the Environment variables pane.

3. Define your environment variables as key-value pairs.

You can add one or more environment variables to configure the function. The following environment variables are supported:

Variable Name	Description	Required	Default Value
FLOW_CRN	<p>The Cloudera Resource Name (CRN) for the data flow that is to be run.</p> <p>The data flow must be stored in the Cloudera DataFlow Catalog. This CRN should indicate the specific version of the data flow and as such will end with a suffix like /v.1.</p> <p>For more information, see <i>Retrieving data flow CRN</i>.</p>	true	--
DF_PRIVATE_KEY	<p>The Private Key for accessing the Cloudera DataFlow service.</p> <p>The Private Key and Access Key are used to authenticate with the Cloudera DataFlow Service and they must provide the necessary authorizations to access the specified data flow.</p> <p>For more information, see <i>Provisioning Access Key ID and Private Key</i>.</p>	true	--
DF_ACCESS_KEY	<p>The Access Key for accessing the Cloudera DataFlow service.</p> <p>The Private Key and Access Key are used to authenticate with the Cloudera DataFlow Service and they must provide the necessary authorizations to access the specified data flow.</p> <p>For more information, see <i>Provisioning Access Key ID and Private Key</i>.</p>	true	--
INPUT_PORT	<p>The name of the Input Port to use.</p> <p>If the specified data flow has more than one Input Port at the root group level, this environment variable must be specified, indicating the name of the Input Port to queue up the AWS Lambda notification. If there is only one Input Port, this variable is not required. If it is specified, it must properly match the name of the Input Port.</p>	false	--

Variable Name	Description	Required	Default Value
OUTPUT_PORT	<p>The name of the Output Port to retrieve the results from.</p> <p>If no Output Port exists, the variable does not need to be specified and no data will be returned. If at least one Output Port exists in the data flow, this variable can be used to determine the name of the Output Port whose data will be sent along as the output of the function.</p> <p>For more information on how the appropriate Output Port is determined, see <i>Output ports</i>.</p>	false	--
FAILURE_PORTS	<p>A comma-separated list of Output Ports that exist at the root group level of the data flow. If any FlowFile is sent to one of these Output Ports, the function invocation is considered a failure.</p> <p>For more information, see <i>Output ports</i>.</p>	false	--
DEFAULT_PARAM_CONTEXT	<p>If the data flow uses a Parameter Context, the Lambda function will look for a Secret in the AWS Secrets Manager with the same name. If no Secret can be found with that name, this variable specifies the name of the Secret to default to.</p> <p>For more information, see <i>Parameters</i>.</p>	false	--
PARAM_CONTEXT_*	<p>If the data flow makes use of Parameter Contexts, this environment variable provides a mechanism for mapping a Parameter Context to one or more alternate Secrets in the AWS Secrets Manager, in a comma-separated list.</p> <p>For more information, see <i>Parameters</i>.</p>	false	--
CONTENT_REPO	<p>The contents of the FlowFiles can be stored either in memory, on the JVM heap, or on disk. If this environment variable is set, it specifies the path to a directory where the content should be stored. If it is not specified, the content is held in memory.</p> <p>For more information, see <i>File system for content repository</i>.</p>	false	--

Variable Name	Description	Required	Default Value
EXTENSIONS_*	The directory to look for custom extensions / NiFi Archives (NARs). For more information, see <i>Providing custom extensions / NARs</i> .	false	--
DF_SERVICE_URL	The Base URL for the Cloudera DataFlow Service.	false	https://api.us-west-1.cdp.cloudera.com/
NEXUS_URL	The Base URL for a Nexus Repository for downloading any NiFi Archives (NARs) needed for running the data flow.	false	https://repository.cloudera.com/artifactory/cloudera-repos/
STORAGE_BUCKET	An S3 bucket in which to look for custom extensions / NiFi Archives (NARs) and resources. For more information, see <i>S3 Bucket storage</i> .	false	--
STORAGE_EXTENSIONS_DIRECTORY	The directory in the S3 bucket to look for custom extensions / NiFi Archives (NARs). For more information, see <i>S3 Bucket storage</i> .	false	extensions
STORAGE_RESOURCES_DIRECTORY	The directory in the S3 bucket to look for custom resources. For more information, see <i>Providing additional resources</i> .	false	resources
WORKING_DIR	The working directory, where NAR files will be expanded.	false	/tmp/working
EXTENSIONS_DOWNLOAD_DIR	The directory to which missing extensions / NiFi Archives (NARs) will be downloaded. For more information, see <i>Providing custom extensions / NARs</i> .	false	/tmp/extensions
EXTENSIONS_DIR_*	It specifies read-only directories that may contain custom extensions / NiFi Archives (NARs). For more information, see <i>Providing custom extensions / NARs</i> .	false	--
DISABLE_STATE_PROVIDER	If true, it disables the DynamoDB data flow state provider, even if the data flow has stateful processors.	false	--
DYNAMODB_STATE_TABLE	The DynamoDB table name where the state will be stored	false	nifi_state
DYNAMODB_BILLING_MODE	It sets the DynamoDB Billing Mode (PROVISIONED or PAY_PER_REQUEST).	false	PAY_PER_REQUEST

Variable Name	Description	Required	Default Value
DYNAMODB_WRITE_CAPACITY_UNITS	It sets the DynamoDB Write Capacity Units, when using PROVISIONED Billing Mode.	true if using PROVISIONED Billing Mode	--
DYNAMODB_READ_CAPACITY_UNITS	It sets the DynamoDB Read Capacity Units, when using PROVISIONED Billing Mode.	true if using PROVISIONED Billing Mode	--
KRB5_FILE	It specifies the filename of the krb5.conf file. This is necessary only if connecting to a Kerberos-protected endpoint. For more information, see <i>Configuring Kerberos</i> .	false	/etc/krb5.conf

4. When ready, click Save.

Tutorial: Building a new flow from scratch

If you are new to flow design and have never used NiFi before, this tutorial is for you. Learn how to build a draft adding and configuring components, connecting them, creating Controller Services, and testing your flow while creating it.

About this task

This tutorial walks you through the creation of a simple flow design that retrieves the latest changes from Wikipedia through invoking the Wikipedia API. It converts JSON events to Avro, before filtering and routing them to two different processors which merge events together before a file is written to local disk.

You will learn about:

- Creating a draft
- Creating a Controller Service
- Adding processors to your draft
- Configuring processors
- Adding a user-defined property to a processor configuration
- Connecting processors to create relationships between them
- Running a Test Session
- Publishing a draft to the Catalog as a flow definition.

Before you begin

The flow you are about to build can be deployed without any external dependencies and does not require any parameter values during deployment. Still, there are prerequisites you have to meet before you can start building your first draft.

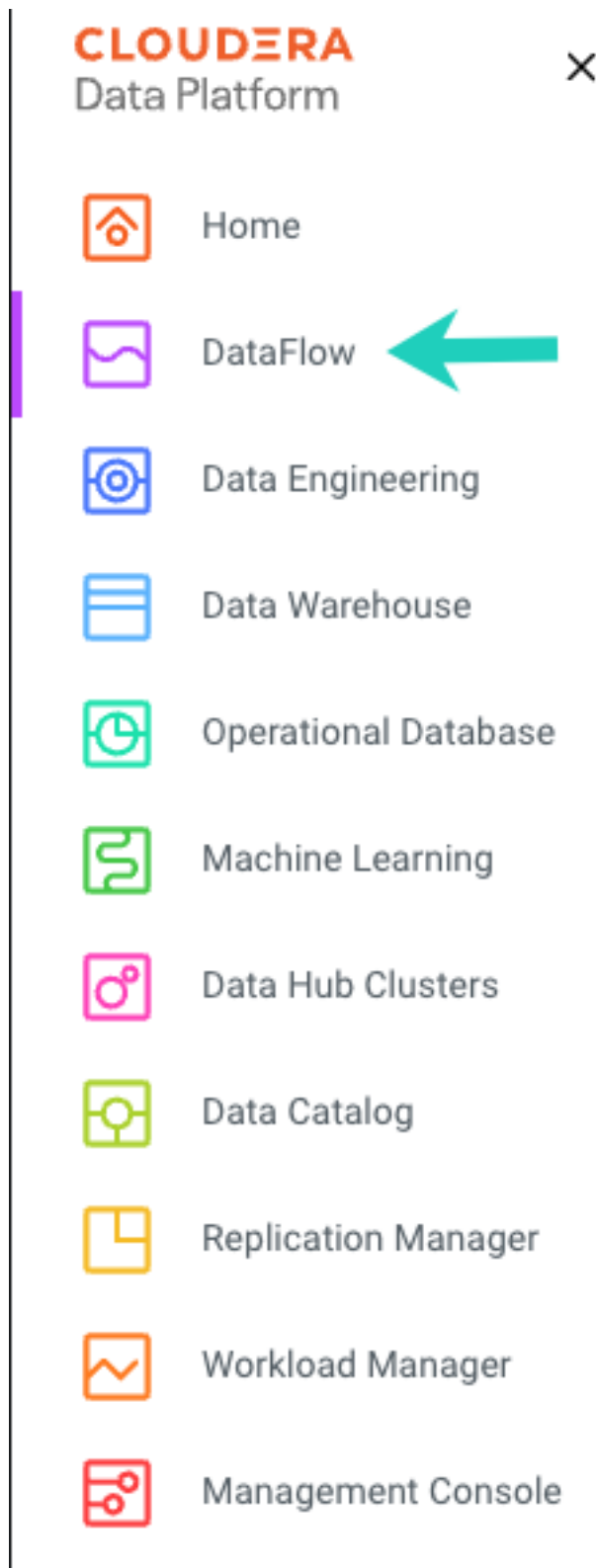
- You have an enabled and healthy Cloudera DataFlow environment.
- You have been assigned the DFDeveloper role granting you access to the Flow Designer.
- You have been assigned the DFCatalogAdmin or DFCatalogViewer role granting you access to the Catalog. You will need this authorization to publish your draft as a flow definition to the Catalog.
- You have been assigned the DFFlowAdmin role for the environment to which you want to deploy the flow definition.

Create a new flow

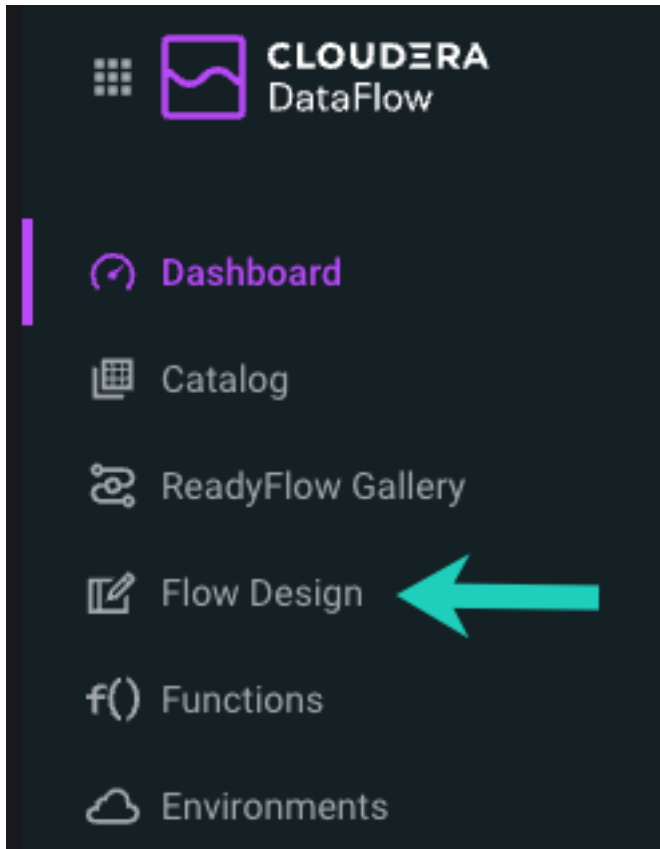
Create a new flow in a Flow Designer Workspace and give it a name.

Procedure

1. Open Cloudera DataFlow by clicking the DataFlow tile in the Cloudera sidebar.

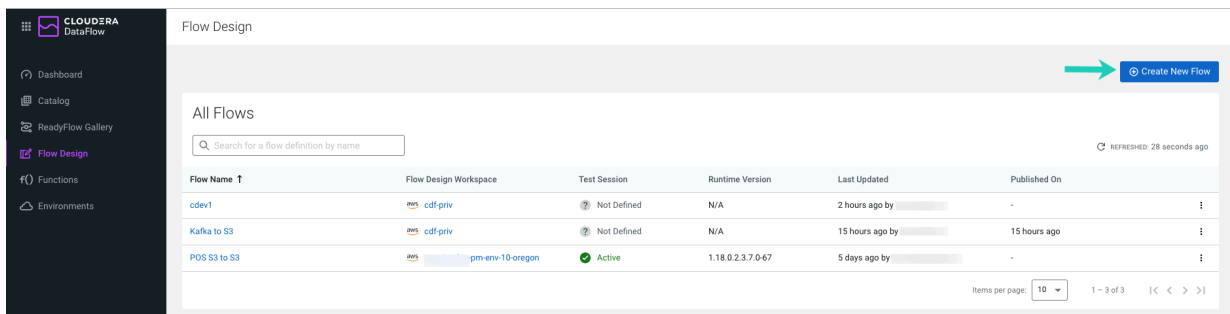


- Click Flow Design in the left navigation pane.



You are redirected to the **Flow Design** page, where previously created draft flows are displayed, one flow per row.


- Click Create New Flow.



4. Select a Target Workspace where you want to create the draft.

Create New Draft ✕

Target Workspace [?](#)

 cdf-priv ▼

Target Project [?](#)


Select a project ▼

Draft Name

Draft Name

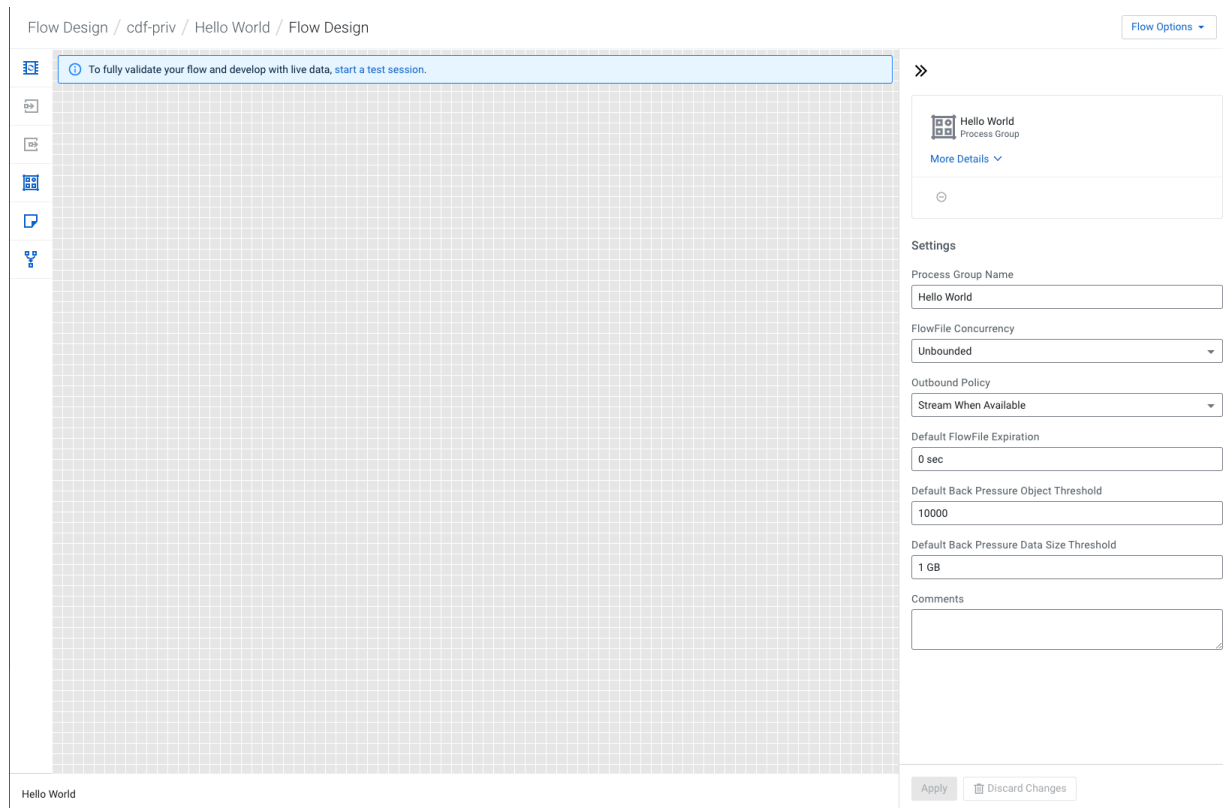
Create

Cancel

5. Under Target Project select  Unassigned.
6. Provide a Draft Name.
Provide Hello World.

7. Click Create.

Flow Designer creates a default Process Group with the Draft Name you provided, 'Hello World' in this case, and you are redirected to the **Flow Design** canvas. The **Configuration** pane on the right displays configuration options for the default Process Group.



What to do next

Proceed to creating controller services.

Create controller services

Learn about creating Controller Services in Cloudera DataFlow Flow Designer.

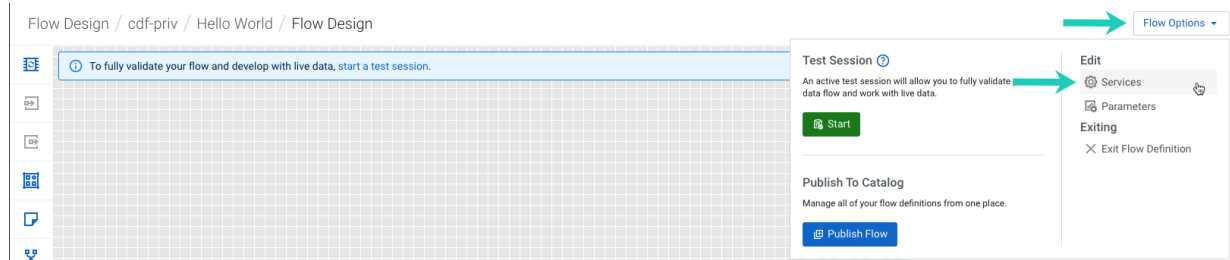
About this task

Controller Services are extension points that provide information for use by other components (such as processors or other controller services). The idea is that, rather than configure this information in every processor that might need it, the controller service provides it for any processor to use as needed.

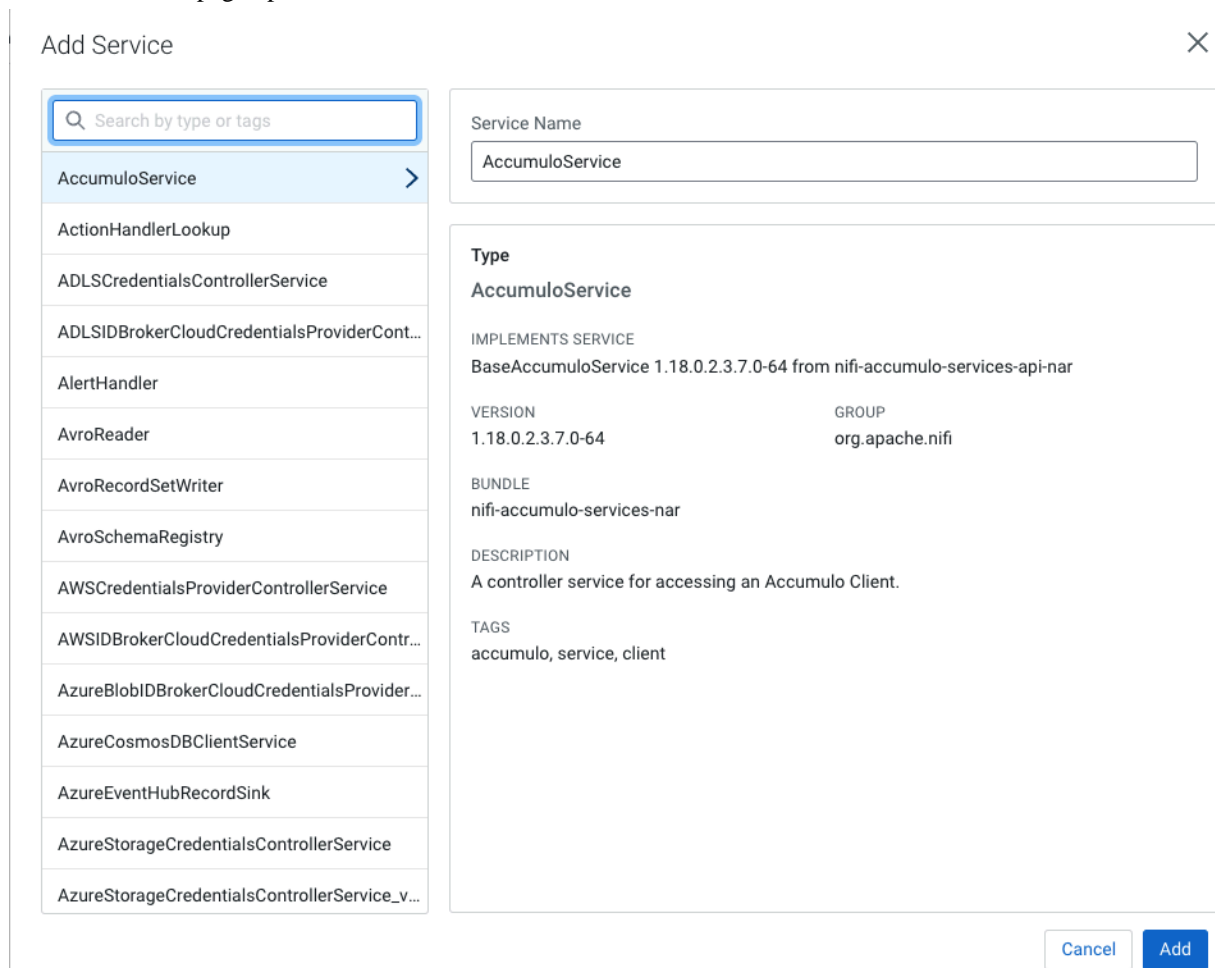
You will use the controller services you create now to configure the behavior of several processors you will add to your flow as you are building it.

Procedure

1. Go to Flow Options Services .



2. Click Add Service.
The Add Service page opens.



3. In the text box, filter for JsonTreeReader.

Add Service

Q json

AvroSchemaRegistry

JsonConfigBasedBoxClientService

JsonPathReader

JsonRecordSetWriter

JsonTreeReader

RestLookupService

Service Name

JsonTreeReader

Type

JsonTreeReader

IMPLEMENTS SERVICE

RecordReaderFactory 1.18.0.2.3.7.0-89 from nifi-standard-services-api-nar

VERSION

1.18.0.2.3.7.0-89

GROUP

org.apache.nifi

BUNDLE

nifi-record-serialization-services-nar

DESCRIPTION

Parses JSON into individual Record objects. While the reader expects each record to be well-formed JSON, the content of a FlowFile may consist of many records, each as a well-formed JSON array or JSON object with optional whitespace between them, such as the common 'JSON-per-line' format. If an array is encountered, each element in that array will be treated as a separate record. If the schema that is configured contains a field that is not present in the JSON, a null value will be used. If the JSON contains a field that is not present in the schema, that field will be skipped. See the Usage of the Controller Service for more information and examples.

TAGS

parser, reader, record, tree, json

Cancel

Add

4. Provide Service Name: JSON_Reader_Recent_Changes.
5. Click Add.

6. Configure the JSON_Reader_Recent_Changes service.

Set the following Properties:

Starting Field Strategy

Nested Field

Starting Field Name

recentchanges

[Flow Options ▾](#)**JSON_Reader_Recent_Changes**
JsonTreeReader V.1.18.0.2.3.7.0-89[More Details ▾](#)

Settings

*Service Name

Comments

Properties

Property	Value	
Schema Access Strategy ?	Infer Schema	...
Schema Inference Cache ?	No value set	...
Starting Field Strategy ?	Root Node ▾	...
Date Format ?	Root Node?	...
Time Format ?	Nested Field?	...
Timestamp Format ?	No value set	...

Referencing Components

No referencing Processors to display.

No referencing Services to display.

[Apply](#)[Discard Changes](#)

7. Click Apply.

8. Click Add Service to create another service.

9. In the text box, filter for AvroRecordSetWriter.

10. Provide Service Name: AvroWriter_Recent_Changes.

11. Click Add.

You do not need to configure the AvroWriter_Recent_Changes service. You can leave all properties with their default values.

12. Click Add Service to create a third service.

13. In the text box, filter for AvroReader.

14. Provide Service Name: AvroReader_Recent_Changes.

15. Click Add.

You do not need to configure the AvroReader_Recent_Changes service. You can leave all properties with their default values.

16. Click Back To Flow Designer to return to the flow design Canvas.

What to do next

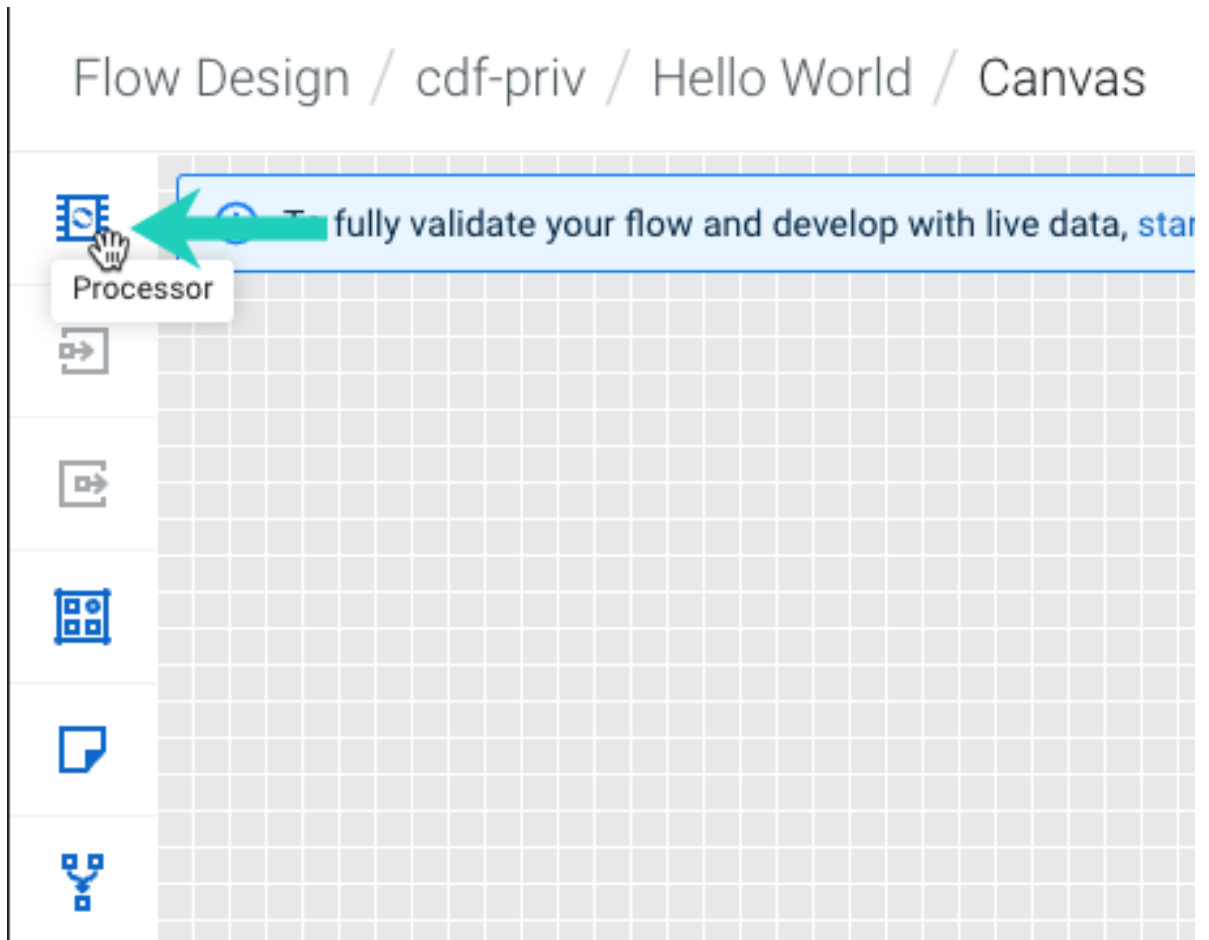
After creating the necessary Controller Services, you can start building and configuring your flow.

Build your draft flow

Start building your draft flow by adding components to the Canvas and configuring them.

Procedure

1. Add an InvokeHTTP processor to the canvas.
 - a) Drag a Processor from the Components sidebar to the canvas.



- b) In the text box, filter for InvokeHTTP.

Add Processor

🔍 invokeH

InvokeHTTP

Type

InvokeHTTP

IMPLEMENTS SERVICE

VERSION

1.18.0.2.3.7.0-89

GROUP

org.apache.nifi

BUNDLE

nifi-standard-nar

DESCRIPTION

An HTTP client processor which can interact with a configurable HTTP Endpoint. The destination URL and HTTP Method are configurable. FlowFile attributes are converted to HTTP headers and the FlowFile contents are included as the body of the request (if the HTTP Method is PUT, POST or PATCH).

TAGS

rest, http, client, https

Processor Name

InvokeHTTP

Cancel

Add

c) Change the Processor Name to Get Recent Wikipedia Changes.

d) Click Add.

After configuration, this processor calls the Wikipedia API to retrieve the latest changes.

2. Configure the Get Recent Wikipedia Changes processor.

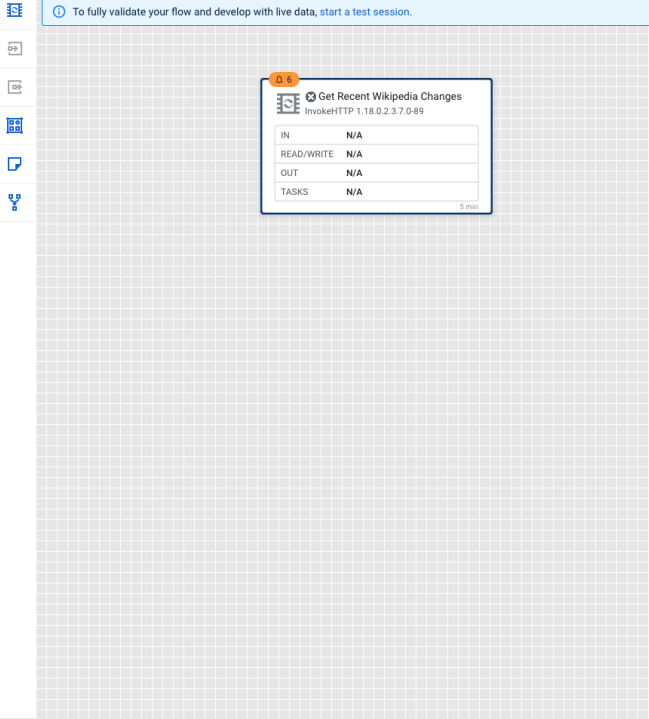
Properties

HTTP URL

provide `https://en.wikipedia.org/w/api.php?action=query&list=recentchanges&format=json&rcprop=user%7Ccomment%7Cparsedcomment%7Ctimestamp%7Ctitle%7Csize%7Ctags`

Flow Design / cdf-priv / Hello World / Canvas Flow Options ▾

To fully validate your flow and develop with live data, start a test session.



The canvas displays a grid with a single processor node titled 'Get Recent Wikipedia Changes' (version 1.18.0.2.3.7.0-89). The node's properties are listed as follows:

Property	Value
IN	N/A
READ/WRITE	N/A
OUT	N/A
TASKS	N/A

WARN

Comments

Scheduling

*Scheduling Strategy: Timer Driven

*Concurrent Tasks: 1

*Run Duration: 0ms

*Run Schedule: 0 sec

*Execution: All Nodes

Properties Add Property

Property	Value
HTTP Method	
HTTP URL	https://en.wikipedia.org/w/api.php?
HTTP/2 Disabled	
SSL Context Service	
Socket Connect Timeout	
Socket Read Timeout	
Socket Idle Timeout	5 mins
Socket Idle Connections	5
Proxy Configuration Service	No value set
Proxy Host	No value set

Apply Discard Changes

Relationships

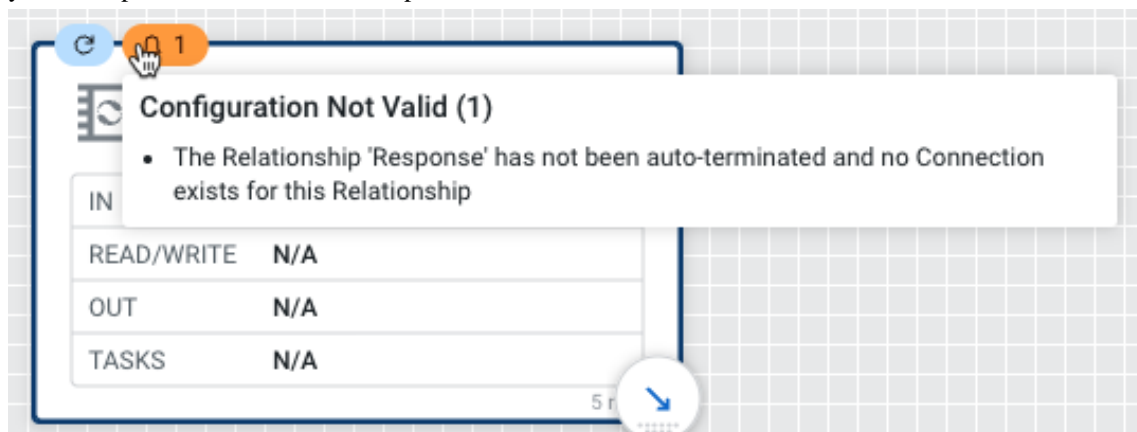
Select the following relationships:

- Original - Terminate
- Failure - Terminate, Retry
- Retry - Terminate
- No Retry - Terminate

3. Click Apply.



Notice: Note the orange Notification Pill in the upper left corner of your processor. It is there to warn you about possible issues with a component.



In this particular case, it warns you about one of the Relationships not being addressed when configuring your processor. Do not worry, it will disappear when you create a connection to another processor for the 'Response' relationship.

4. Add a ConvertRecord processor to the canvas.

- Drag a Processor from the Components sidebar to the canvas.
- In the text box, filter for ConvertRecord.
- Change the Processor Name to Convert JSON to AVRO.
- Click Add.

This processor converts the JSON response to AVRO format. It uses RecordReaders and RecordWriters to accomplish this. It infers the JSON schema starting from the recent changes field.

5. Configure the Convert JSON to AVRO processor.

Properties

Record Reader

Select the JSON_Reader_Recent_Changes controller service you have created from the drop-down list.

Record Writer

Select the AvroWriter_Recent_Changes controller service you have created from the drop-down list.

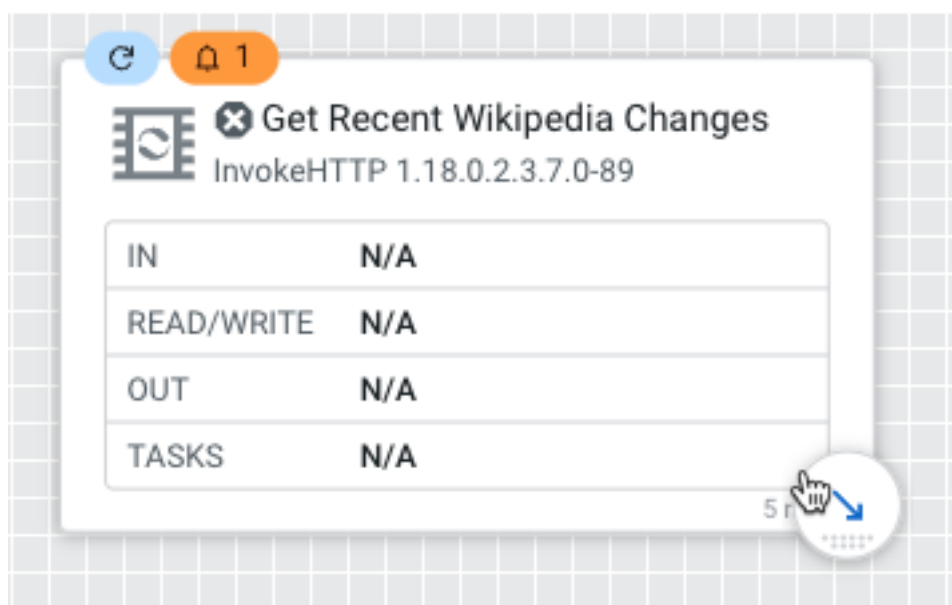
Relationships

Select the following relationships:

failure - Terminate, Retry

6. Click Apply.

7. Connect the Get Recent Wikipedia Changes and Convert JSON to AVRO processors by hovering over the lower-right corner of the Get Recent Wikipedia Changes processor, clicking the arrow that appears and dragging it to the Convert JSON to AVRO processor.



8. In the configuration popup, select the Response relationship and click Add.

Create Connection ✕

From Processor

Get Recent Wikipedia Changes
InvokeHTTP

Within Group

Hello World

Relationships

☐ Original

☐ Failure

☐ Retry

☐ No Retry

☒ Response

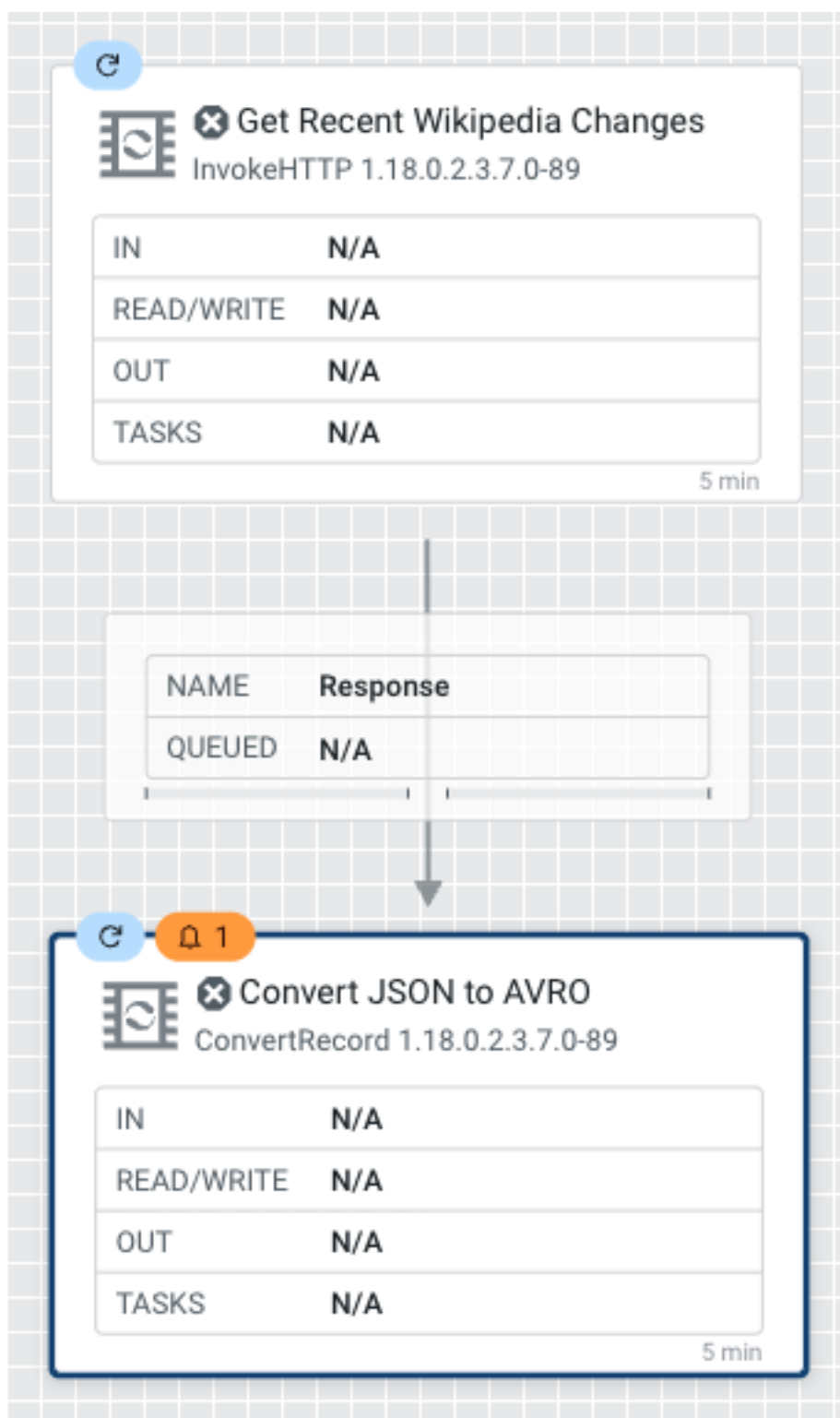
To Processor


Convert JSON to AVRO
ConvertRecord

Within Group

Hello World

Cancel Add



Notice: Note, that the  Notification Pill warning about the unconfigured 'Response' relationship disappeared from your Get Recent Wikipedia Changes processor.

9. Add a QueryRecord processor.

- a) Drag a Processor from the Components sidebar to the canvas.
- b) In the text box, filter for QueryRecord.
- c) Name it Filter Edits.
- d) Click Add.

This processor filters out anything but actual page edits. To achieve this, it's running a query that selects all FlowFiles (events) of the type edit.

10. Configure the Filter Edits processor.

Properties

Record Reader

Select the AvroReader_Recent_Changes controller service you have created from the drop-down list.

Record Writer

Select the AvroWriter_Recent_Changes controller service you have created from the drop-down list.

Relationships

Select the following relationship:

- failure - Terminate
- original - Terminate

11. For the Filter Edits processor you also need to add a user-defined property. Click Add Property.

- a) Provide Filtered edits as Name
- b) Provide Select * from FLOWFILE where type='edit' as Value.
- c) Click Apply.

12. Connect the Convert JSON to AVRO and Filter Edits processors by hovering over the lower-right corner of the Convert JSON to AVRO processor, clicking the arrow that appears and dragging it to the Filter Edits processor.**13. In the configuration pane, select the success and failure relationships and click Add.****14. Add a second QueryRecord processor.**

- a) Drag a Processor from the Components sidebar to the canvas.
- b) In the text box, filter for QueryRecord.
- c) Name it Route on Content Size.
- d) Click Add.

This processor uses two SQL statements to separate edit events that resulted in a longer article from edit events that resulted in a shorter article.

15. Configure the Route on Content Size processor.

Properties

Record Reader

Select the AvroReader_Recent_Changes controller service you have created from the drop-down list.

Record Writer

Select the AvroWriter_Recent_Changes controller service you have created from the drop-down list.

Relationships

Select the following relationships:

- failure - Terminate, Retry
- original - Terminate

16. For the Route on Content Size processor you also need to add two user-defined properties.

- a) Click Add Property.
- b) Provide Added content as Name
- c) Provide Select * from FLOWFILE where newlen>=oldlen as Value.
- d) Click Add.
- e) Click Add Property, to create the second property.
- f) Provide Removed content as Name.
- g) Provide Select * from FLOWFILE where newlen<oldlen as Value.
- h) Click Add.
- i) Click Apply.

17. Connect the Filter Edits and Route on Content Size processors by hovering over the lower-right corner of the Filter Edits processor, clicking the arrow that appears and drawing it to Route on Content Size.

In the Create Connection pop up select the Filtered edits relation and click Add.

18. Add two MergeRecord processors.

- a) Drag a Processor from the Components sidebar to the canvas.
- b) In the text box, filter for MergeRecord.
- c) Name it Merge Edit Events.
- d) Click Add.
- e) Repeat the above steps to add another identical processor.

These processors are configured to merge at least 100 records into one flowfile to avoid writing lots of small files. The MaxBinAge property is set to 2 minutes which makes the processors merge records after two minutes even if less than 100 records have arrived.

19. Configure the two Merge Edit Events processors.

Properties

Record Reader

Select the AvroReader_Recent_Changes controller service you have created from the drop-down list.

Record Writer

Select the AvroWriter_Recent_Changes controller service you have created from the drop-down list.

Max Bin Age

Set to two minutes by providing a value of 2 min.

Relationships

Select the following relationships:

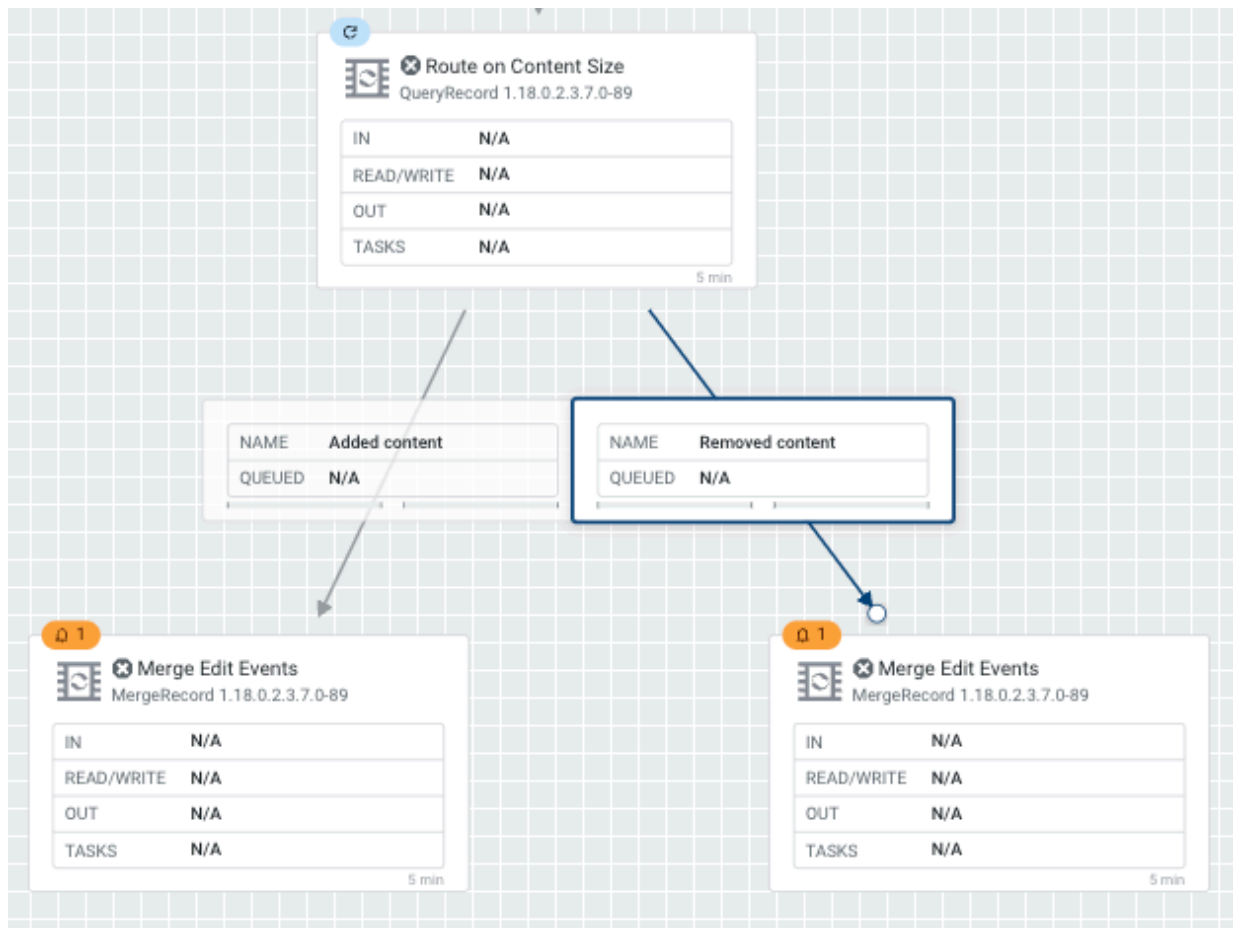
- failure - Terminate
- original - Terminate



Note: Do not forget to perform configuration for both Merge Edit Events processors.

20. Connect the Route on Content Size processor to both of the Merge Edit Events processors.

- For the first Merge Edit Events processor, select Added content from Relationships and click Add.
- For the second Merge Edit Events processor, select Removed content from Relationships and click Add.



21. Add two PutFile processors to the canvas.

- Drag a Processor from the Components sidebar to the canvas.
- In the text box, filter for PutFile.
- Name it Write "Added Content" Events To File.
- Click Add.
- Repeat the above steps to add another identical processor, naming this second PutFile processor Write "Removed Content" Events To File.

These processors write the filtered, routed edit events to two different locations on the local disk. In Cloudera DataFlow, you would typically not write to local disk but replace these processors with processors that resemble your destination system (Kafka, Database, Object Store etc.)

22. Configure the Write "Added Content" Events To File processor.

Properties:

Directory

Provide /tmp/larger_edits

Maximum File Count

Provide 500

Relationships

Select the following relationships:

- SUCCESS - Terminate
- failure - Terminate

23. Click Apply.**24.** Configure the Write "Removed Content" Events To File processor.

Properties:

Directory

Provide /tmp/smaller_edits

Maximum File Count

Provide 500

Relationships

Select the following relationships:

- SUCCESS - Terminate
- failure - Terminate

25. Click Apply.**26.** Connect the Merge Edit Events processor with the Added content connection to the Write "Added Content" Events To File processor.

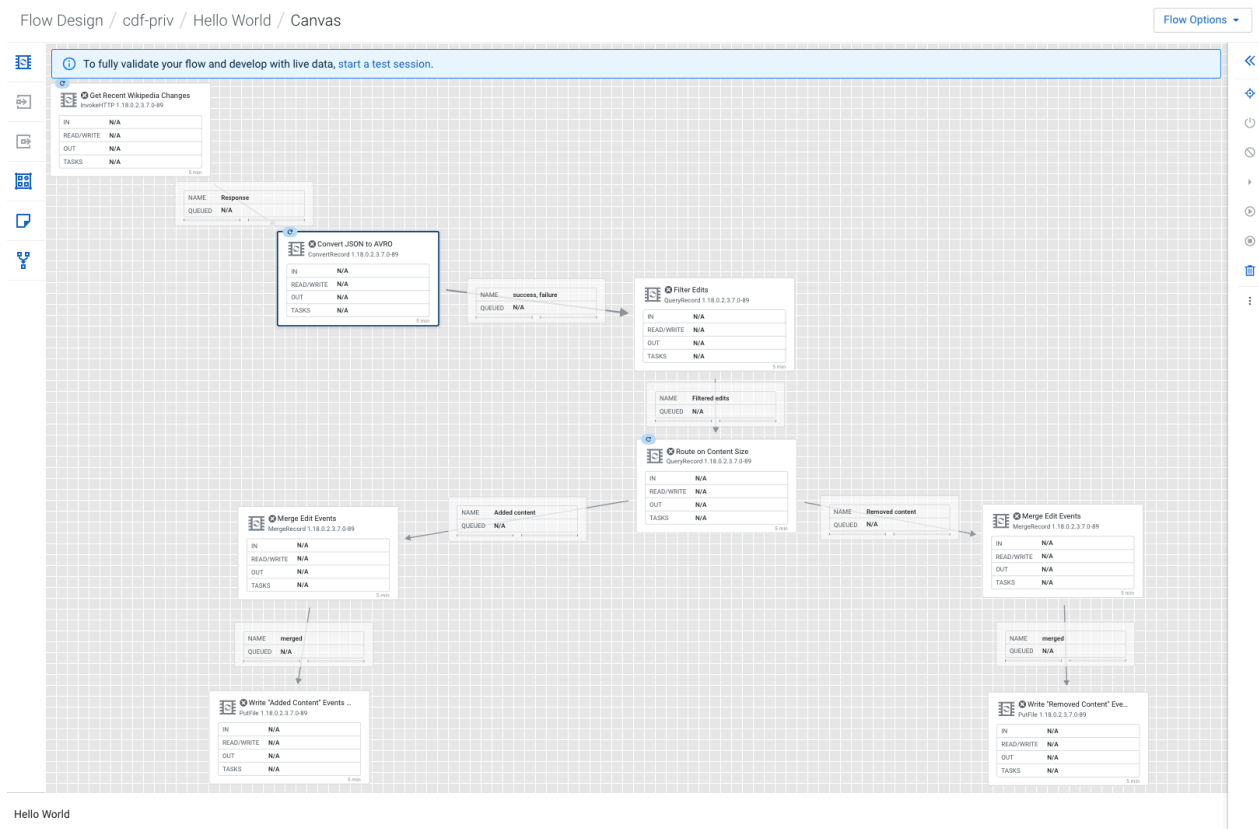
In the Create Connection pop up select merged and click Add.

27. Connect the Merge Edit Events processor with the Removed content connection to the Write "Removed Content" Events To File processor.

In the Create Connection pop up select merged and click Add.

Results

Congratulations, you have created your first draft flow. Now proceed to testing it by launching a Test Session.



Start a test session

To validate your draft, start a test session. This provisions an Apache NiFi cluster where you can test your draft.

About this task

Starting a Test Session provisions NiFi resources, acting like a development sandbox for a particular draft. It allows you to work with live data to validate your data flow logic while updating your draft. You can suspend a test session any time and change the configuration of the NiFi cluster then resume testing with the updated configuration.

Procedure

1. Click start a test session in the banner on top of the Canvas.

 To fully validate your flow and develop with live data, [start a test session](#).

- 2. Click Start Test Session.**

Test Session status

○ Initializing Test Session...

Initializing Test Session... appears on top of the page.

- 3. Wait for the status to change to**


Active Test Session

Test Session.

This may take several minutes.

- 4. Click Flow Options Services to enable Controller Services.**


5.

Select a service you want to enable, then click  Enable Service and Referencing Components.

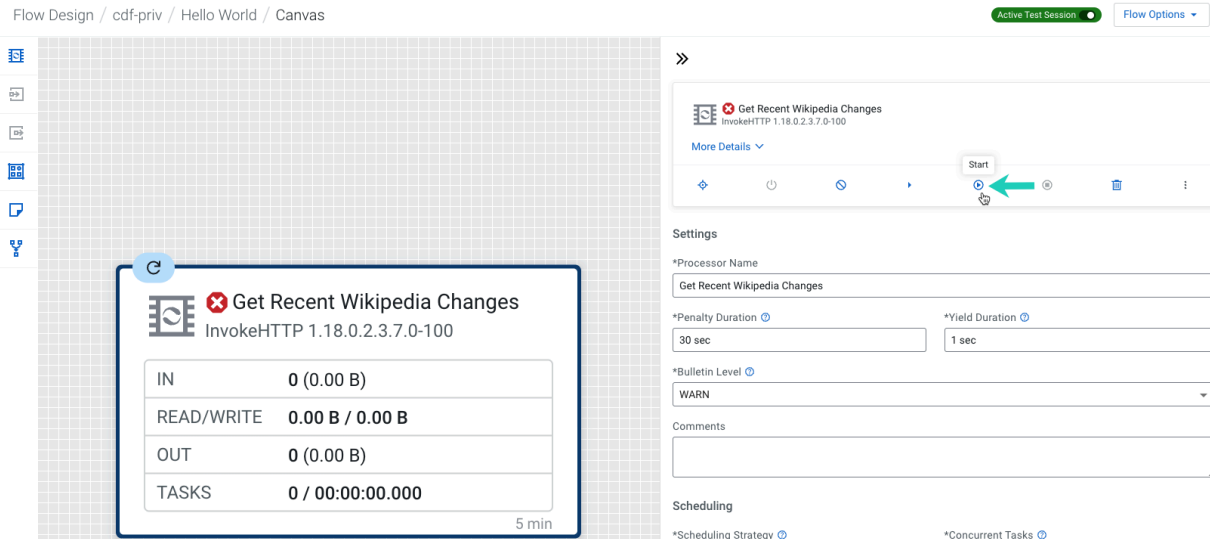
This option does not only enable the controller service, but also any component that references it. This way, you do not need to enable the component separately to run the test session. In the context of this tutorial, enabling the 'AvroReader_Recent_Changes' controller service will also enable 'Filter Edits', 'Route on Content Size', and 'Merge Edit Events' processors as well.

Repeat this step for all Controller Services.

6. Click Back To Flow Designer to return to the flow design Canvas.

7. Start the Get Recent Wikipedia Changes, Write "Added Content" Events To File, and Write "Removed Content" Events To File components by selecting them on the Canvas then clicking  Start.

Flow Design / cdf-priv / Hello World / Canvas Active Test Session Flow Options



The screenshot shows the Cloudera DataFlow interface. On the left is a sidebar with icons for components. The main canvas displays a component named 'Get Recent Wikipedia Changes' (InvokeHTTP 1.18.0.2.3.7.0-100) with a status icon and a '5 min' timer. A table shows component statistics:

IN	0 (0.00 B)
READ/WRITE	0.00 B / 0.00 B
OUT	0 (0.00 B)
TASKS	0 / 00:00:00.000

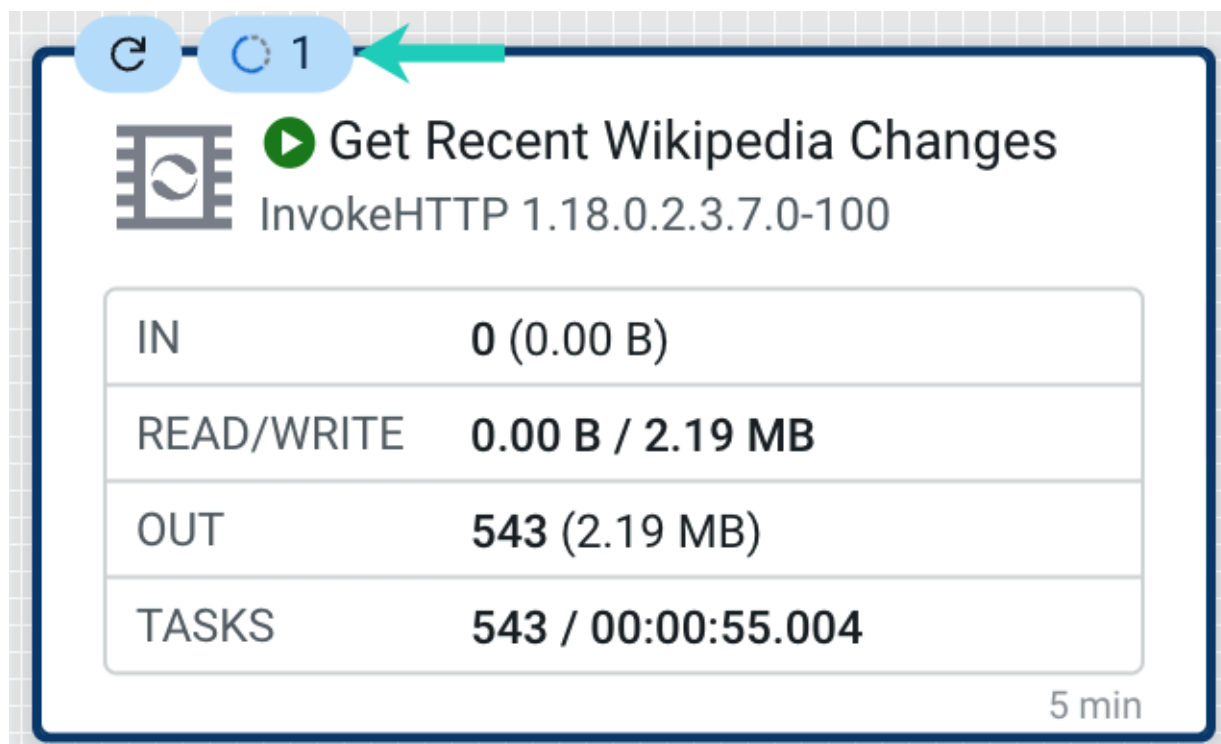
On the right, the settings panel for the component is visible. It includes a 'Start' button with a green arrow pointing to it, and various configuration options:

- Settings:**
 - *Processor Name: Get Recent Wikipedia Changes
 - *Penalty Duration: 30 sec
 - *Yield Duration: 1 sec
 - *Bulletin Level: WARN
- Scheduling:**
 - *Scheduling Strategy
 - *Concurrent Tasks

All other components were auto-started when you selected the Enable Service and Referencing Components option.

8. Observe your first draft flow processing data.

On the Flow Design Canvas you can observe statistics on your processors change as they consume and process data from Wikipedia. You can also observe one or more blue Notification Pills, providing information about the current task.



Publish your flow definition to the Catalog

Now that you have tested your draft and it works fine, you can go on and publish it to the Catalog as a flow definition so that you can create a Cloudera DataFlow deployment.

Procedure

1. On the **Flow Designer** canvas, click **Flow Options Publish To Catalog Publish**.
2. Fill in the fields in the **Publish A New Flow** box.
 - Provide a Flow Name for your flow definition.
You can only provide a name when you publish your flow for the first time.
 - Optionally provide a Flow Description.
You can only provide a description when you publish your flow for the first time.
 - Optionally provide Custom Tags.
You can filter flow definition versions by tags in the Catalog.
 - Optionally provide Version Comments.
3. Click **Publish**.

Results

Your draft is published to the Catalog as a flow definition.