Cloudera Data Flow

Top Tasks

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

https://docs.cloudera.com/

Legal Notice

Contents

Deploying a flow definition using the wizard

Deploy a flow definition to run Apache NiFi flows as flow deployments in Cloudera Data Flow. 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 Data Flow environment.
- You have been assigned at least the DFCatalogViewer role granting you access to the Catalog.
- You have been assigned at least the DFCollectionViewer role granting you access to the collection where the flow definition is located.
- 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 Data Flow, 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.

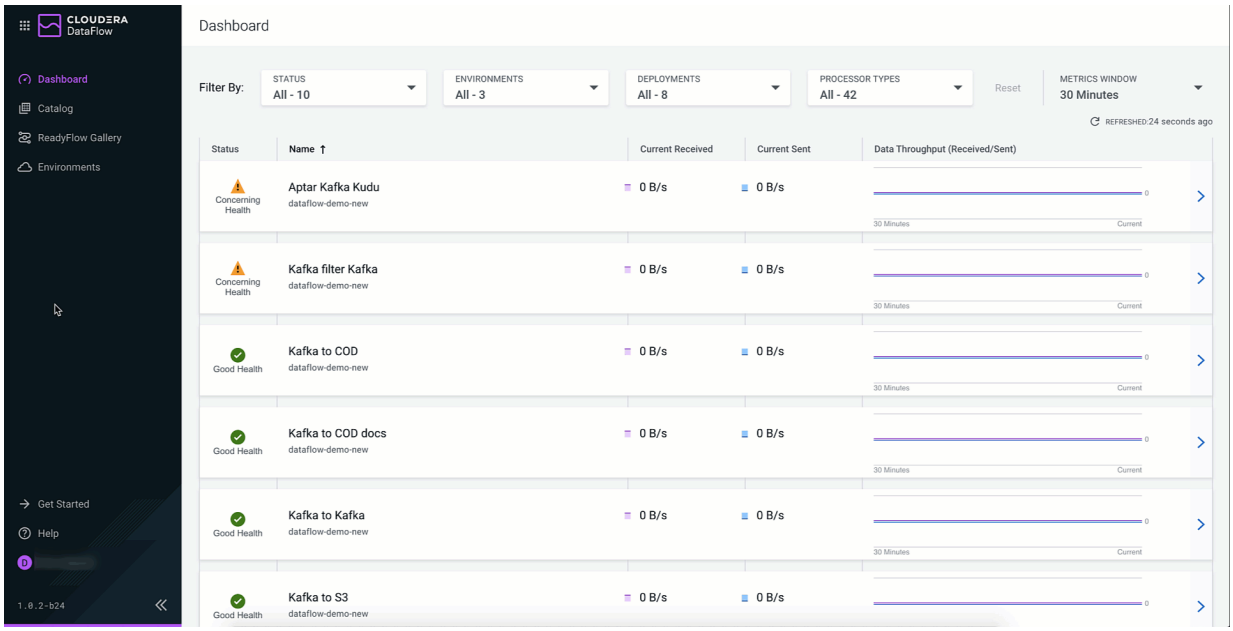
   The flow details pane opens on the right.

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 Data Flow, 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.

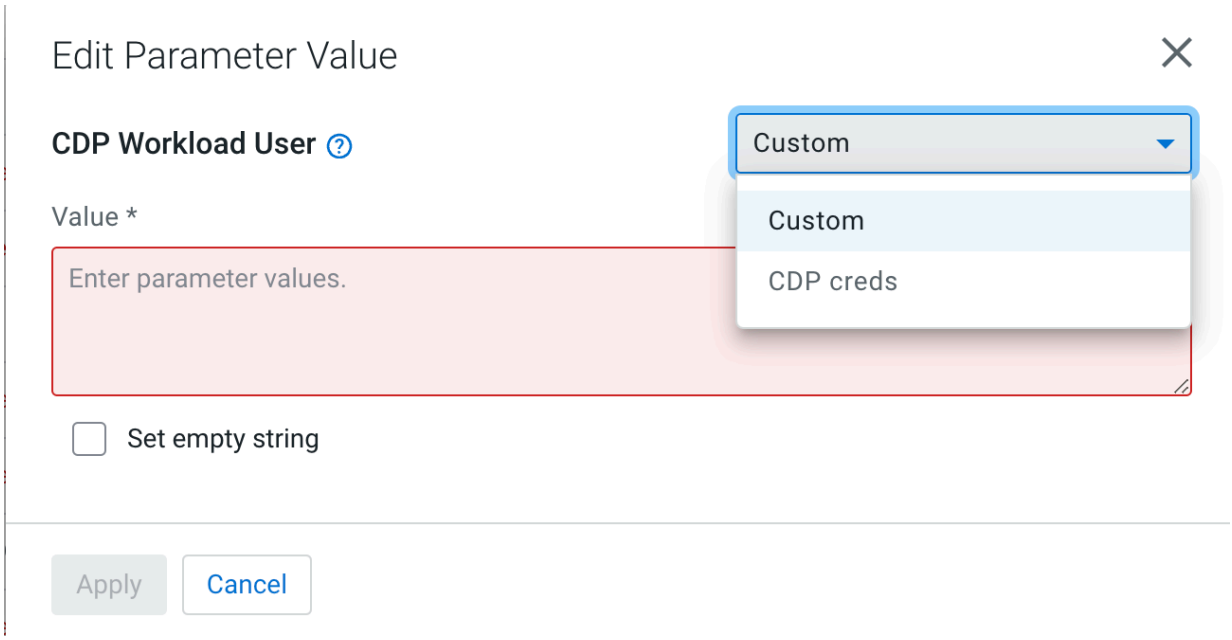
Procedure

- Provide values to parameters required for your flow deployment.

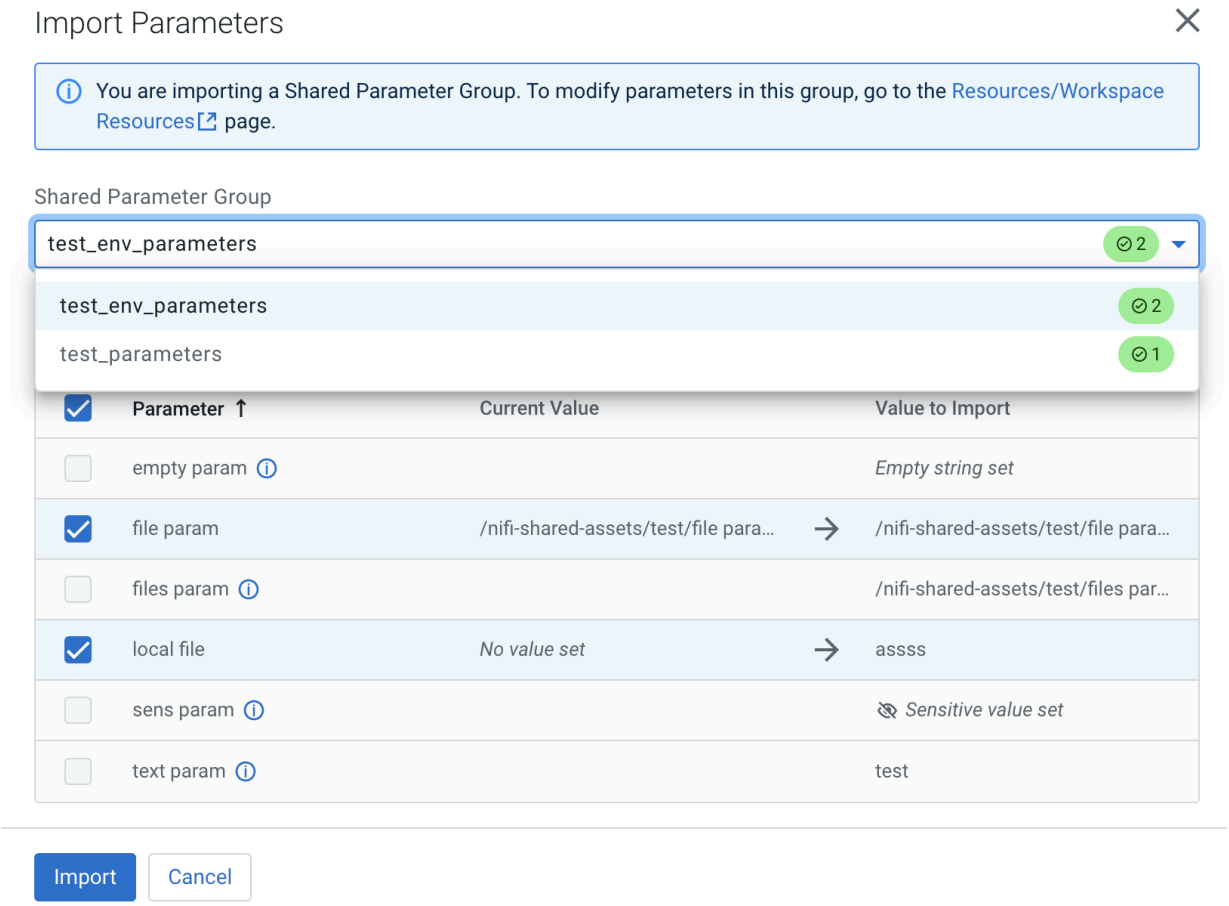
  You have to provide values for all parameters. You can filter for the still empty fields by selecting the No value checkbox.

  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.
- When you finished setting configuration parameters, click Next.
- You may edit parameters individually by clicking the edit icon next to them or you may edit entire parameter groups by selecting the Edit Group option.

  In either case, you have the option to define a custom parameter value that is specific to this deployment or, if available, select the value from a shared parameter group.



- You may import a set of shared parameters from a parameter group to fill in values and facilitate flow deployment. To do so, select the Import Shared Parameters option.



- Parameters in the shared group that match a deployment parameter are preselected, unless you have already modified that parameter in the deployment wizard. In that case that particular parameter is left unselected to protect any changes you have already made and you need to manually select that parameter.

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
   - Custom: If you select this option, you can set the number of vCores between 1-4 and the amount of RAM per node between 2-8 GB. The memory allocated here is the total memory for the NiFi container, of which 50% is allocated to the NiFi JVM heap.

2. Set the number of NiFi nodes and auto-scaling.
   - Optionally select the Auto Scaling checkbox, if you want your cluster to automatically scale based on CPU utilization. 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.
   - Set the number of nodes between 1 and 32.
   - If you have enabled Auto Scaling, you may optionally enable your flow to scale based on flow performance metrics. To do so, select the Flow Metrics Scaling checkbox.
3. Select storage type.

   Select whether you want your deployment to use storage optimized for cost or for performance. You also have the option to set a custom storage size. Only use the custom option if you know how changing individual values affects your deployment.
   - 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
   - Custom:

     Depending on your deployment scenario, you have the following options:
     - On AWS, you can set the repo Size between 50 - 1000 GB, IOPS between 3000 - 7000 in increments of 1000, and Max Throughput between 125 MB/s - 500 MB/s in increments of 100 for both the Content Repo, Provenance Repo, and Flow File Repo. You can set the identical IOPS and Maximum throughput values to all repositories by setting the values for one repository and then selecting the Apply IOPS and Throughput values to all repositories. checkbox.
     - On Azure, you can select one of four predefined custom storage configurations for Content Repo, Provenance Repo, and Flow File Repo. Table *Available Azure storage configurations* describes the predefined custom storage values.

     Table 1: Available Azure storage configurations

Storage configuration	Disk size (GB)	IOPS	Throughput (MB/s)
P6	64	240	50
P10	128	500	100
P15	256	1100	125
P30	1024	5000	200

4. If applicable, configure additional storage for Python processors.

   The wizard automatically detects if the flow definition being deployed contains any Python processors and suggests the amount of extra disk space to add because of them.

   Note:

   If the flow definition does not contain Python processors and you do not plan to introduce Python processors to this flow in the future, you need to take no action. Skip this step.

   If you plan to introduce Python processors to this flow in the future, you must add the extra storage now as you cannot change storage size after deployment.
5. 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 Data Flow function in AWS

You are all set up for creating your first Cloudera Data Flow function: you have a flow definition created in and downloaded from NiFi, a Cloudera Data Flow 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 Data Flow 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.

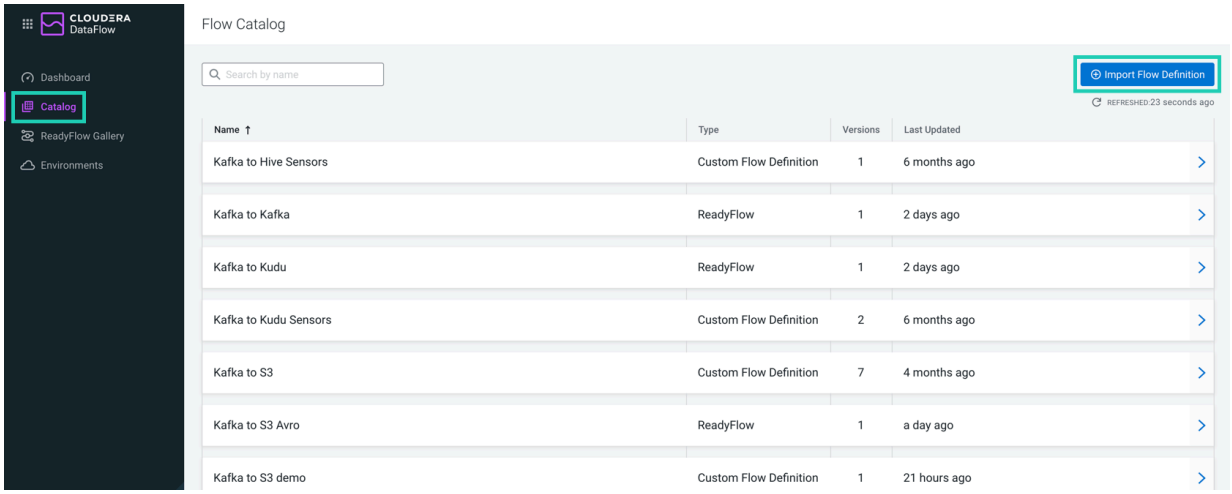
Import a flow definition

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

Steps

1. Open Cloudera Data Flow 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.



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

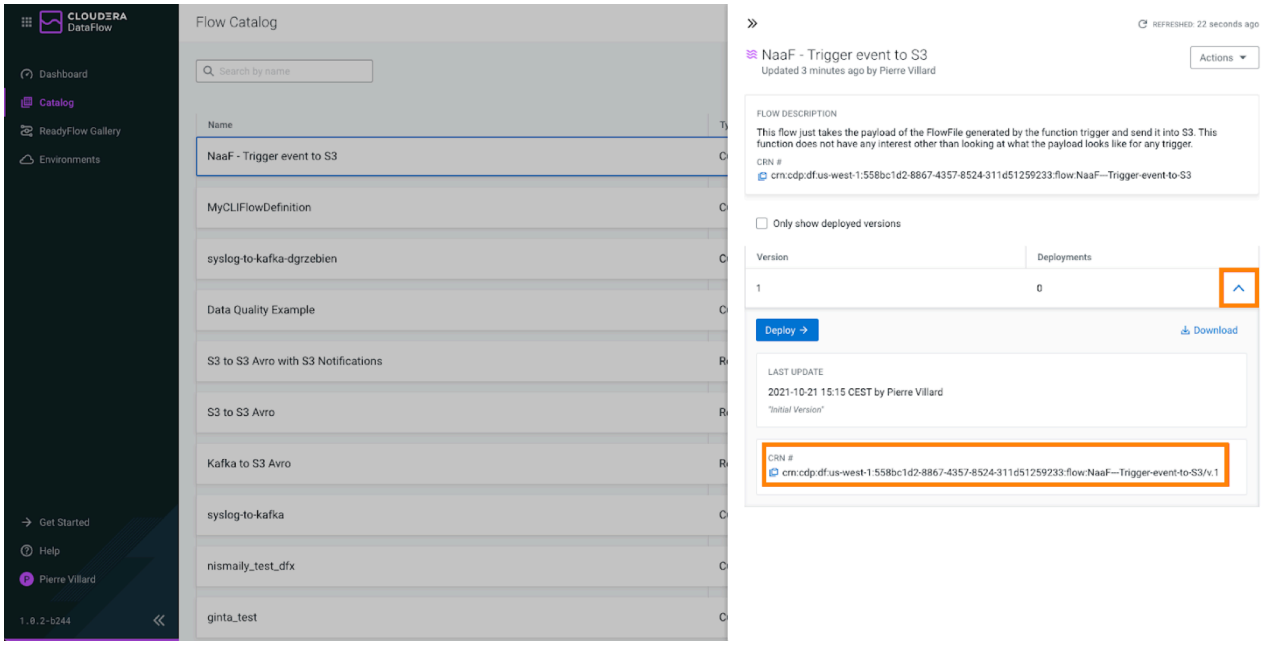
You have successfully imported your NiFi flow to Cloudera Data Flow. It is available in the Flow Catalog as the first version of your flow definition.

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 Data Flow Catalog.

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



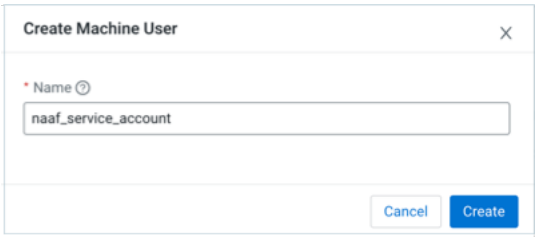
Creating Cloudera service account

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

Procedure

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

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



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

Generating Access Key ID and Private Key

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

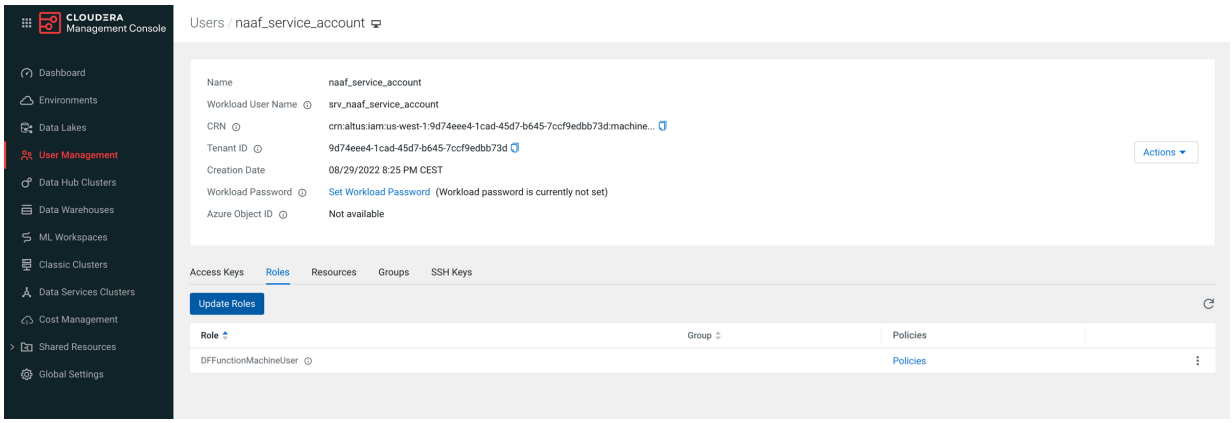
Procedure

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

2. Click Update Roles.

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

   When the role is added, you should see:



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

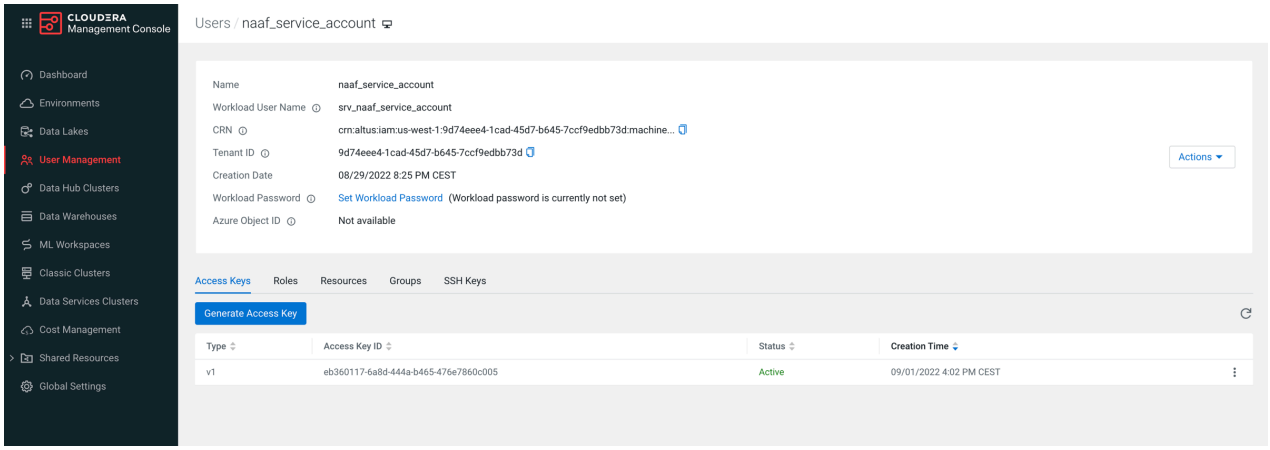
   The Generate Access Key modal window is displayed. it gives you an Access Key ID and a Private Key, which will be required when configuring your function.
7. Click Generate Access Key.

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

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

Results

When ready, you should see something like this:



Creating a Lambda function

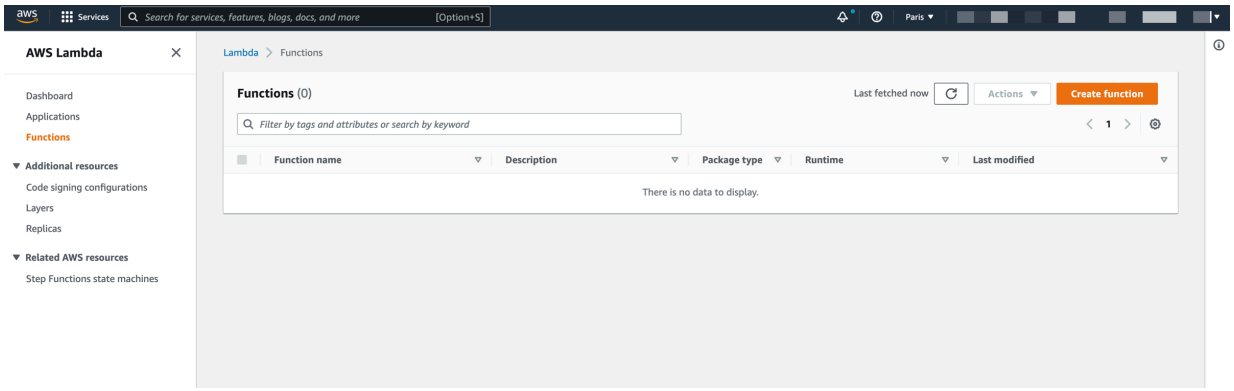
AWS Lambda is a serverless, event-driven compute service that lets you run code for virtually any type of application or backend service without provisioning or managing servers.

About this task

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

Procedure

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

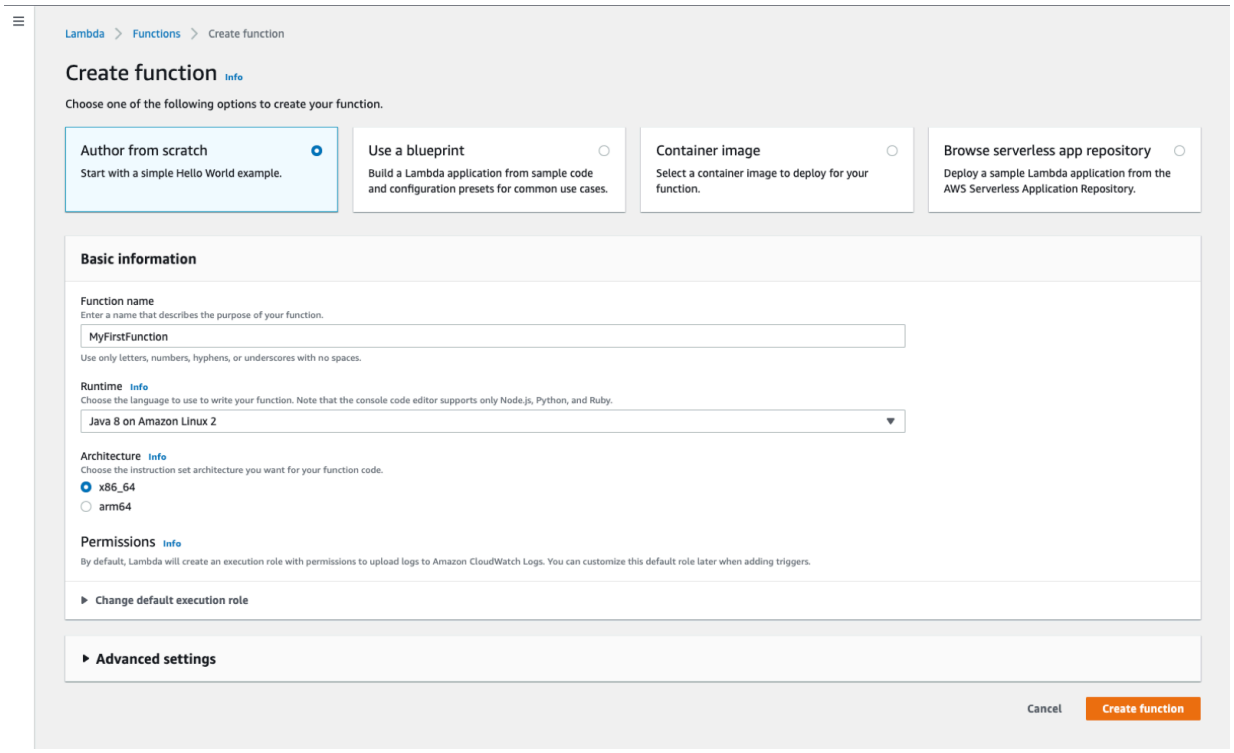


2. Click Create function.

   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.



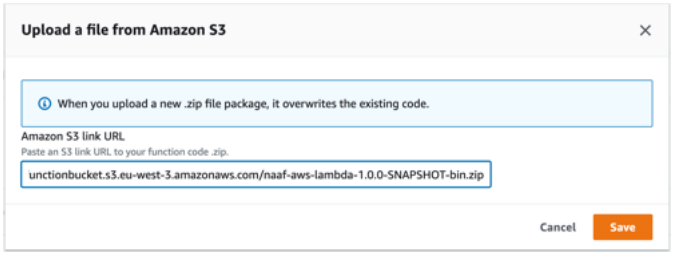
   Your function is created and its details are displayed.

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.

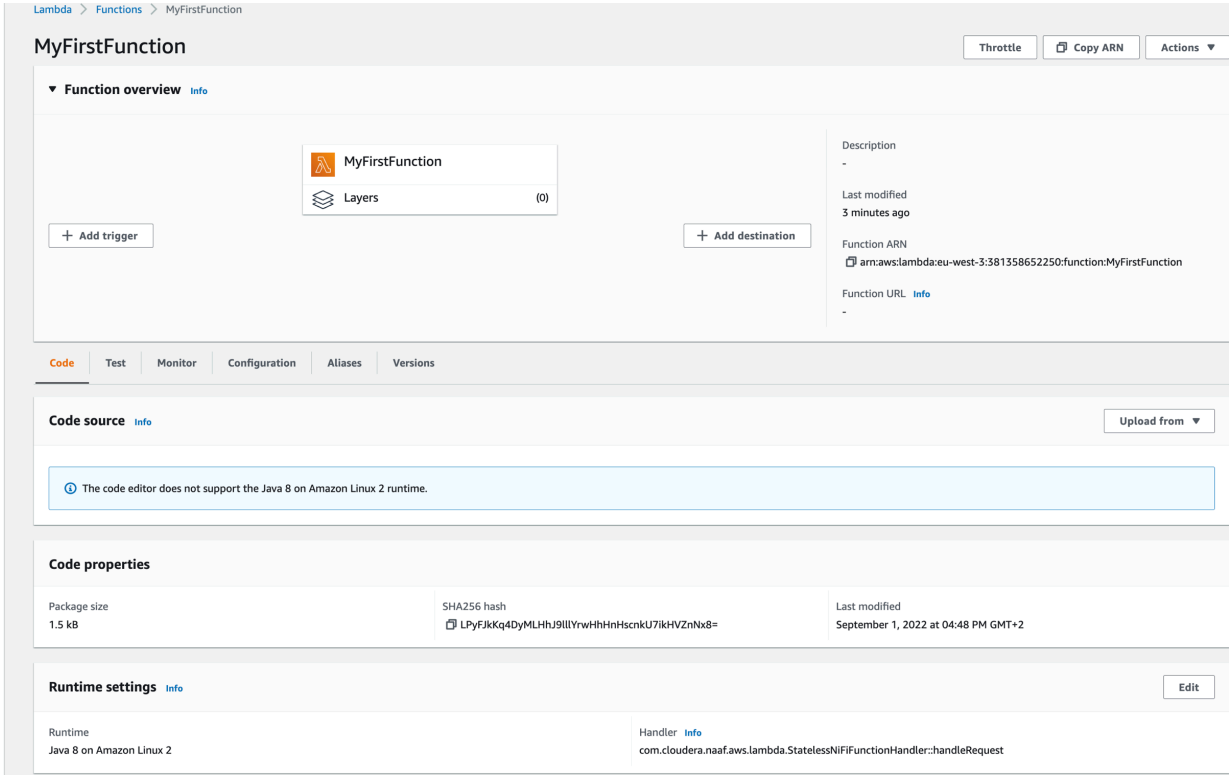
5. Provide the S3 URL of Cloudera Data Flow Function binaries in the textbox.

   Note: You have downloaded the Cloudera Data Flow Function handler libraries as a zip of binaries from Cloudera Data Flow 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 Data Flow Function binaries and uploading to S3*.



6. Click Save.
7. 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

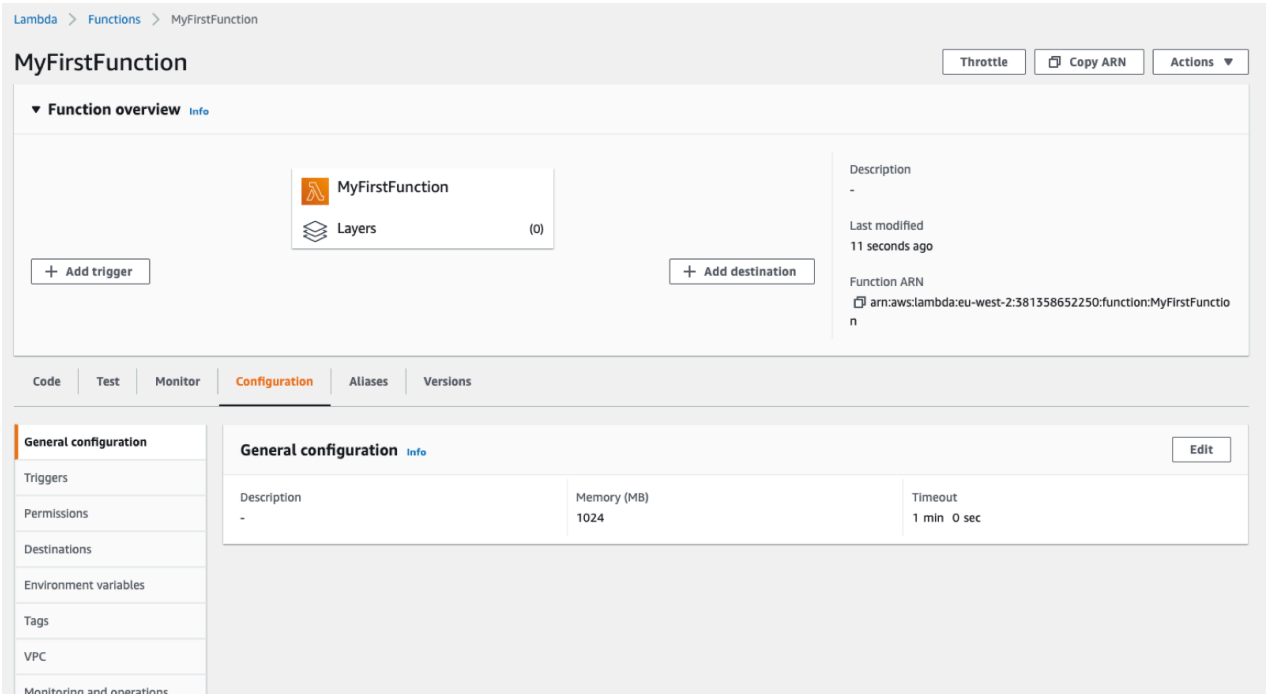
You can now start configuring your Lambda specifically for your flow and for your use case.

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 Data Flow 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	The Cloudera Resource Name (CRN) for the data flow that is to be run. The data flow must be stored in the Cloudera Data Flow Catalog. This CRN should indicate the specific version of the data flow and as such will end with a suffix like /v.1. For more information, see *Retrieving data flow CRN*.	true	--
DF_PRIVATE_KEY	The Private Key for accessing the Cloudera Data Flow service. The Private Key and Access Key are used to authenticate with the Cloudera Data Flow Service and they must provide the necessary authorizations to access the specified data flow. For more information, see *Provisioning Access Key ID and Private Key*.	true	--
DF_ACCESS_KEY	The Access Key for accessing the Cloudera Data Flow service. The Private Key and Access Key are used to authenticate with the Cloudera Data Flow Service and they must provide the necessary authorizations to access the specified data flow. For more information, see *Provisioning Access Key ID and Private Key*.	true	--
INPUT_PORT	The name of the Input Port to use. 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.	false	--

Variable Name	Description	Required	Default Value
OUTPUT_PORT	The name of the Output Port to retrieve the results from. 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. For more information on how the appropriate Output Port is determined, see *Output ports*.	false	--
FAILURE_PORTS	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. For more information, see *Output ports*.	false	--
DEFAULT_PARAM_CONTEXT	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. For more information, see *Parameters*.	false	--
PARAM_CONTEXT_*	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. For more information, see *Parameters*.	false	--
CONTENT_REPO	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. For more information, see *File system for content repository*.	false	--

Variable Name	Description	Required	Default Value
EXTENSIONS_*	The directory to look for custom extensions / NiFi Archives (NARs). For more information, see *Providing custom extensions / NARs*.	false	--
DF_SERVICE_URL	The Base URL for the Cloudera Data Flow 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 *S3 Bucket storage*.	false	--
STORAGE_EXTENSIONS_DIRECTORY	The directory in the S3 bucket to look for custom extensions / NiFi Archives (NARs). For more information, see *S3 Bucket storage*.	false	extensions
STORAGE_RESOURCES_DIRECTORY	The directory in the S3 bucket to look for custom resources. For more information, see *Providing additional resources*.	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 *Providing custom extensions / NARs*.	false	/tmp/extensions
EXTENSIONS_DIR_*	It specifies read-only directories that may contain custom extensions / NiFi Archives (NARs). For more information, see *Providing custom extensions / NARs*.	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 *Configuring Kerberos*.	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. The flow converts JSON events to Avro, then filters and routes the events to two different processors which merge events together, and finally a file is written to local disk.

You will learn about the following actions:

- 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, you must meet the following prerequisites before you can start building your first draft:

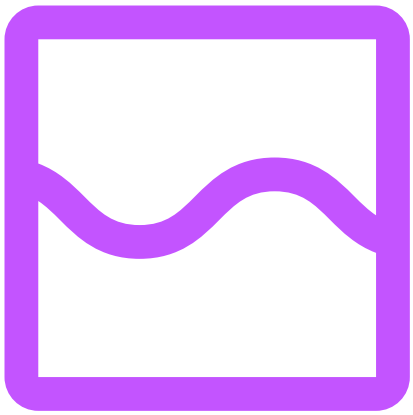
- You must have an enabled and healthy Cloudera Data Flow environment.
- You must be assigned the DFDeveloper role granting you access to the Flow Designer.
- You must be assigned the DFCatalogAdmin or DFCatalogViewer role granting you access to the Catalog. You must have this authorization to publish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

Create a new flow

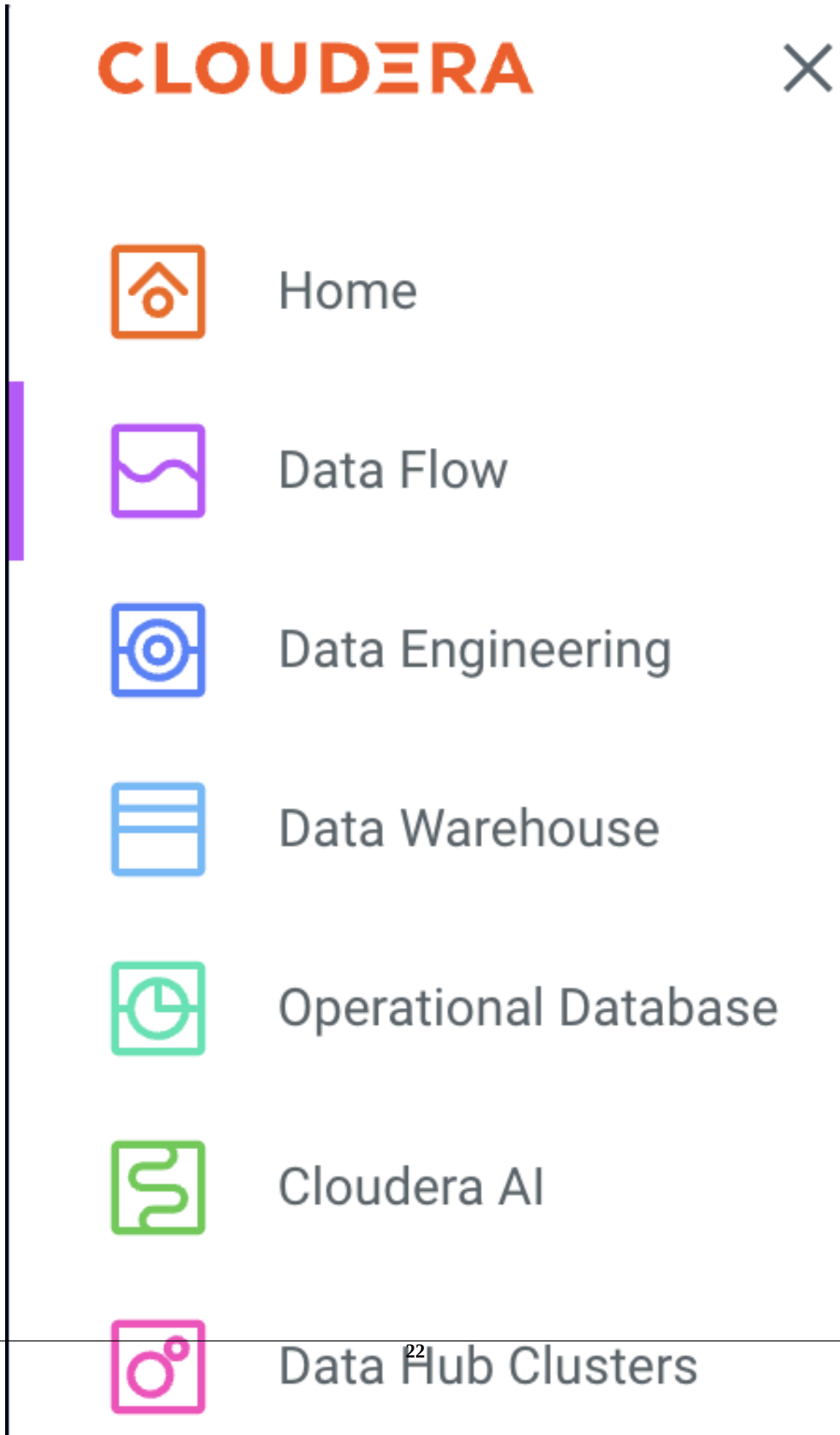
Create and name a new flow in a Flow Designer Workspace.

Procedure

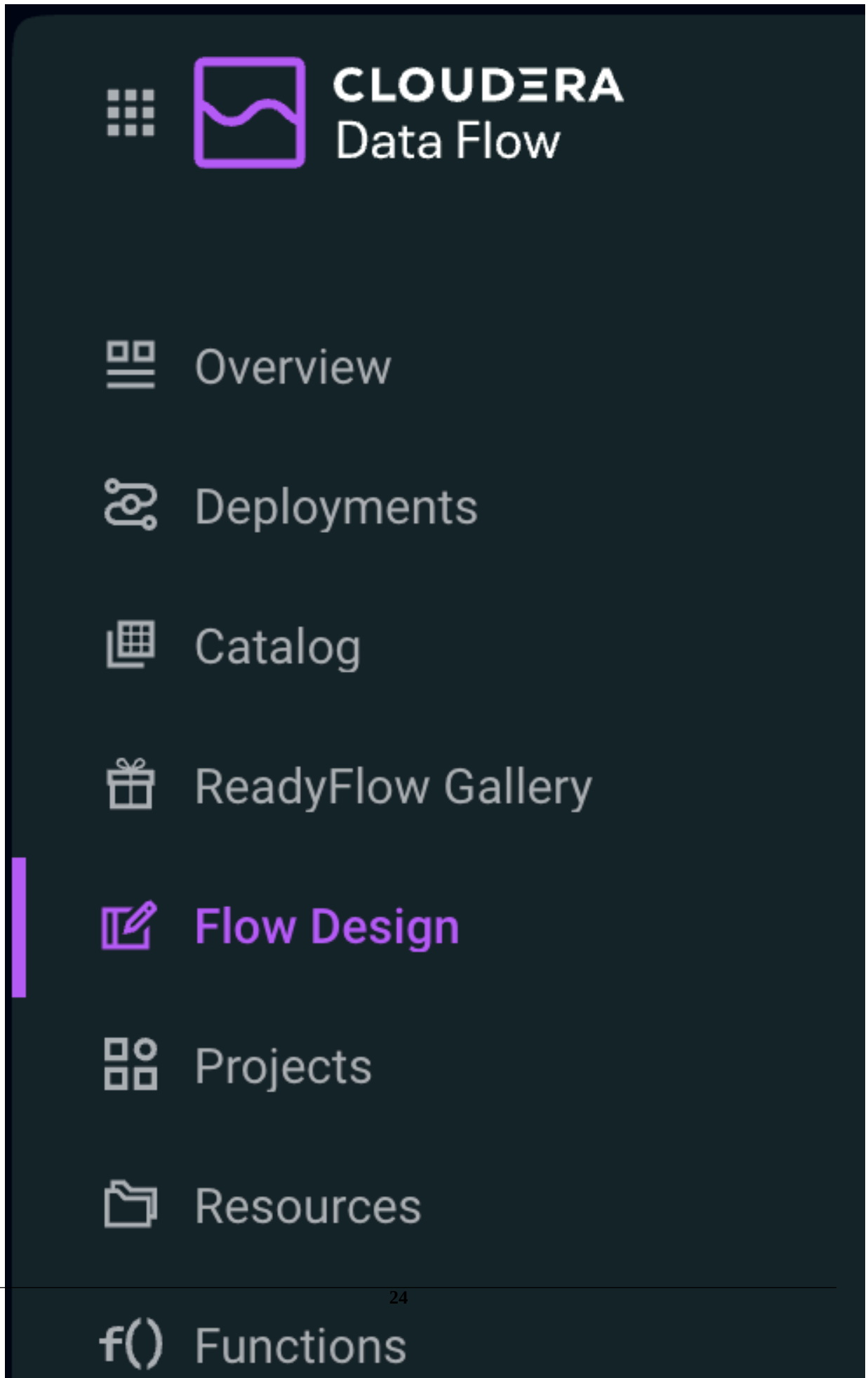
1.



Open Cloudera Data Flow by clicking the Data Flow tile in the Cloudera sidebar.

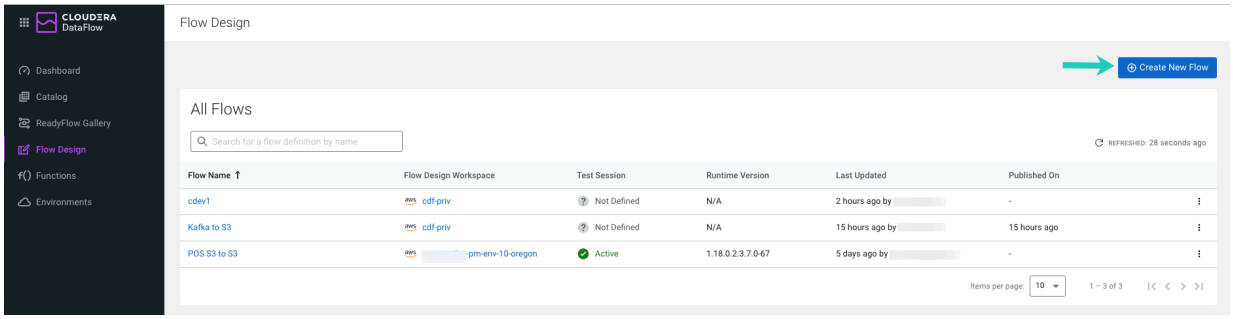

CLOUDERA
Home
Data Flow
Data Engineering
Data Warehouse
Operational Database
Cloudera AI
Data Hub Clusters

2. Click Flow Design in the left navigation pane.

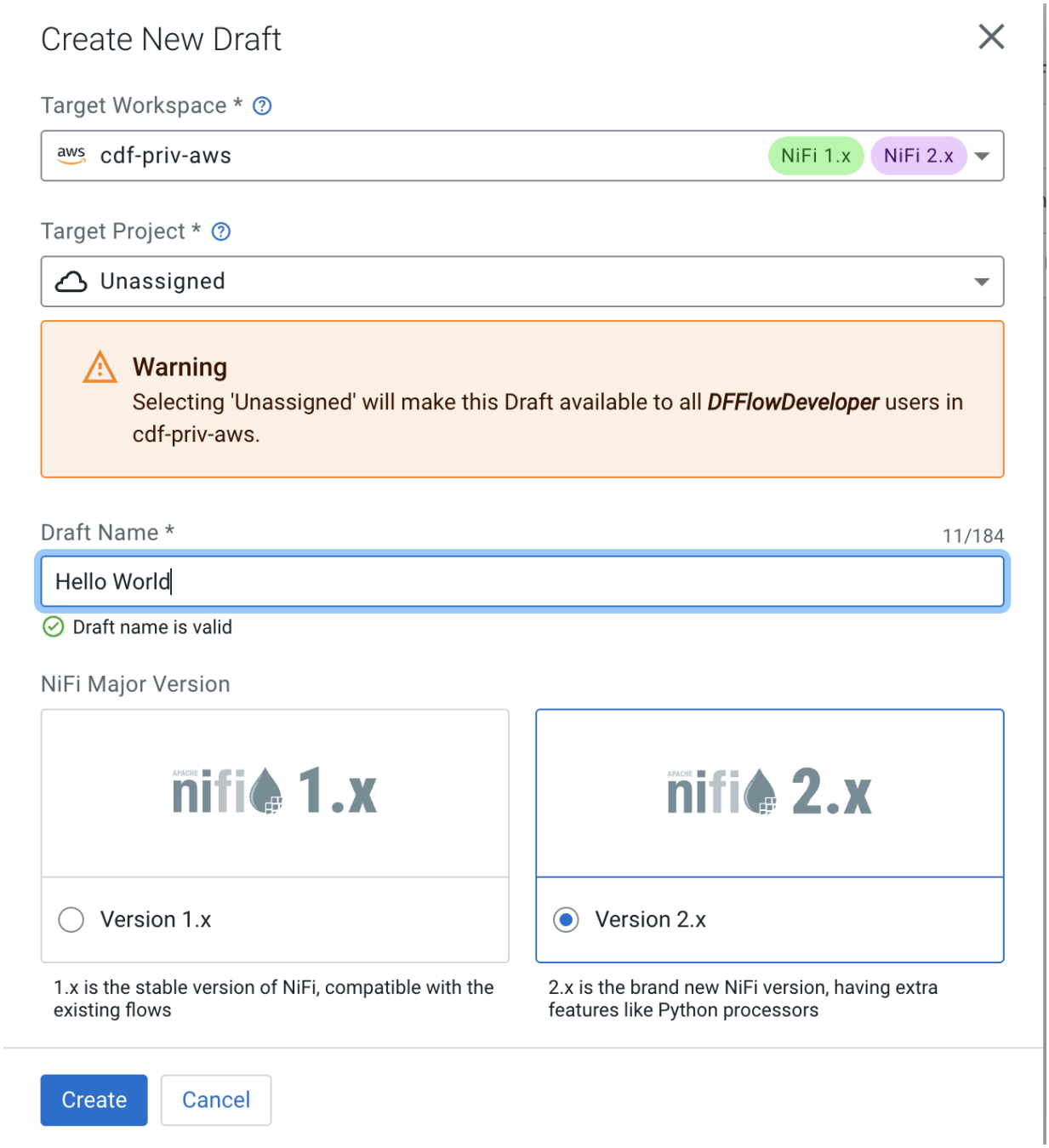

CLOUDERA
Data Flow
Overview
Deployments
Catalog
ReadyFlow Gallery
Flow Design
Projects
Resources
Functions

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

3. Click the ⊕ Create Draft button.

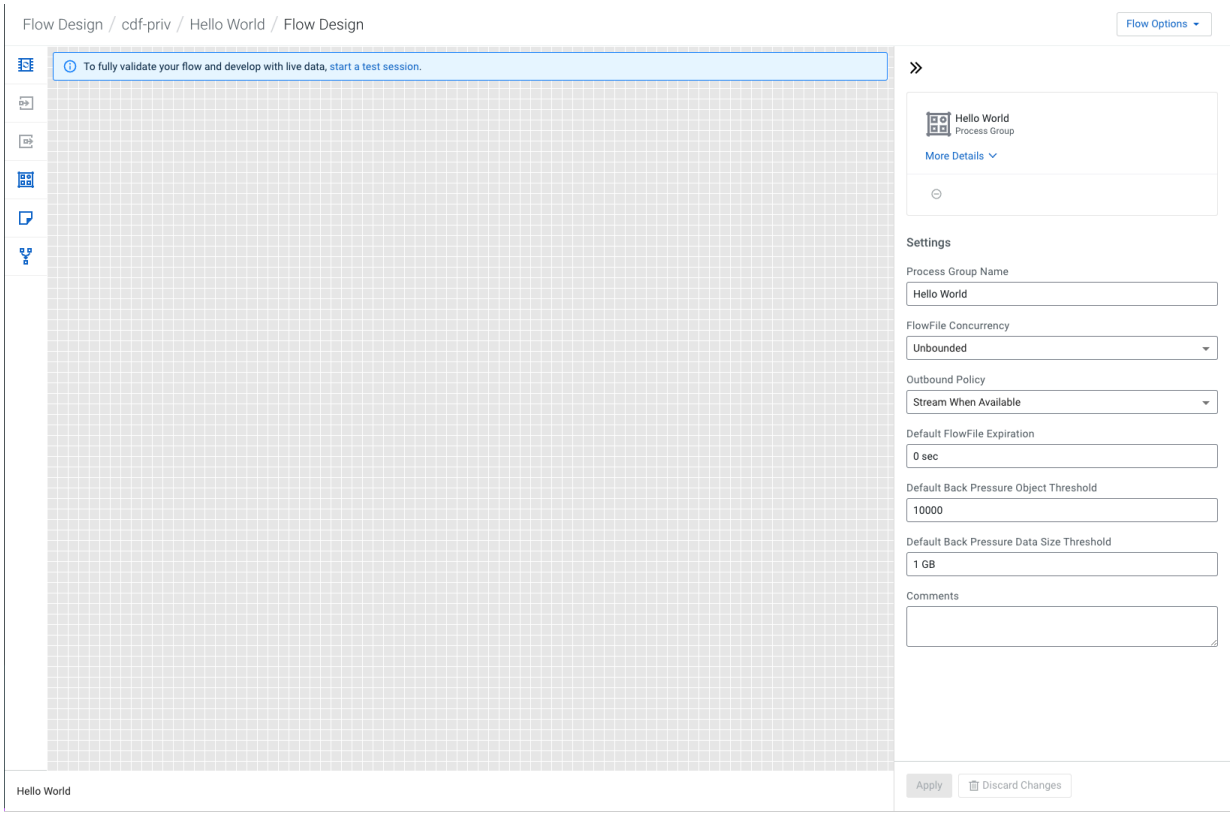


4. Select a Target Workspace where you want to create the draft.



5. In the Target Project field, select the Unassigned option.
6. Provide a Draft Name.
   For example, provide Hello World.
7. Select the Version 2.x radio card for the NiFi Major Version field.

8. Click the Create button.
   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 Data Flow Flow Designer.

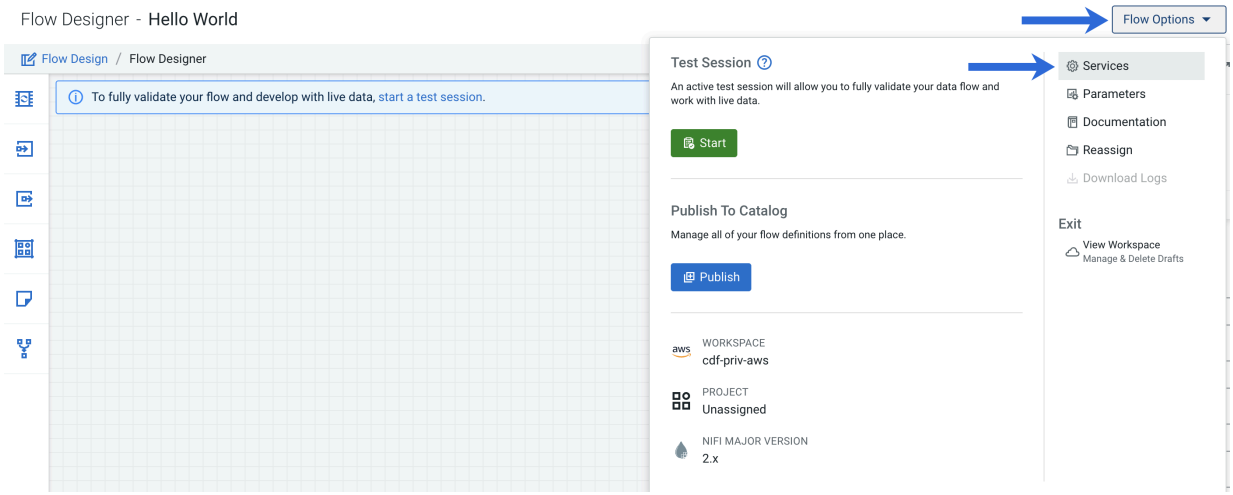
About this task

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 configuring this information in every processor that might need it, the controller service provides the information for any processor to use as needed.

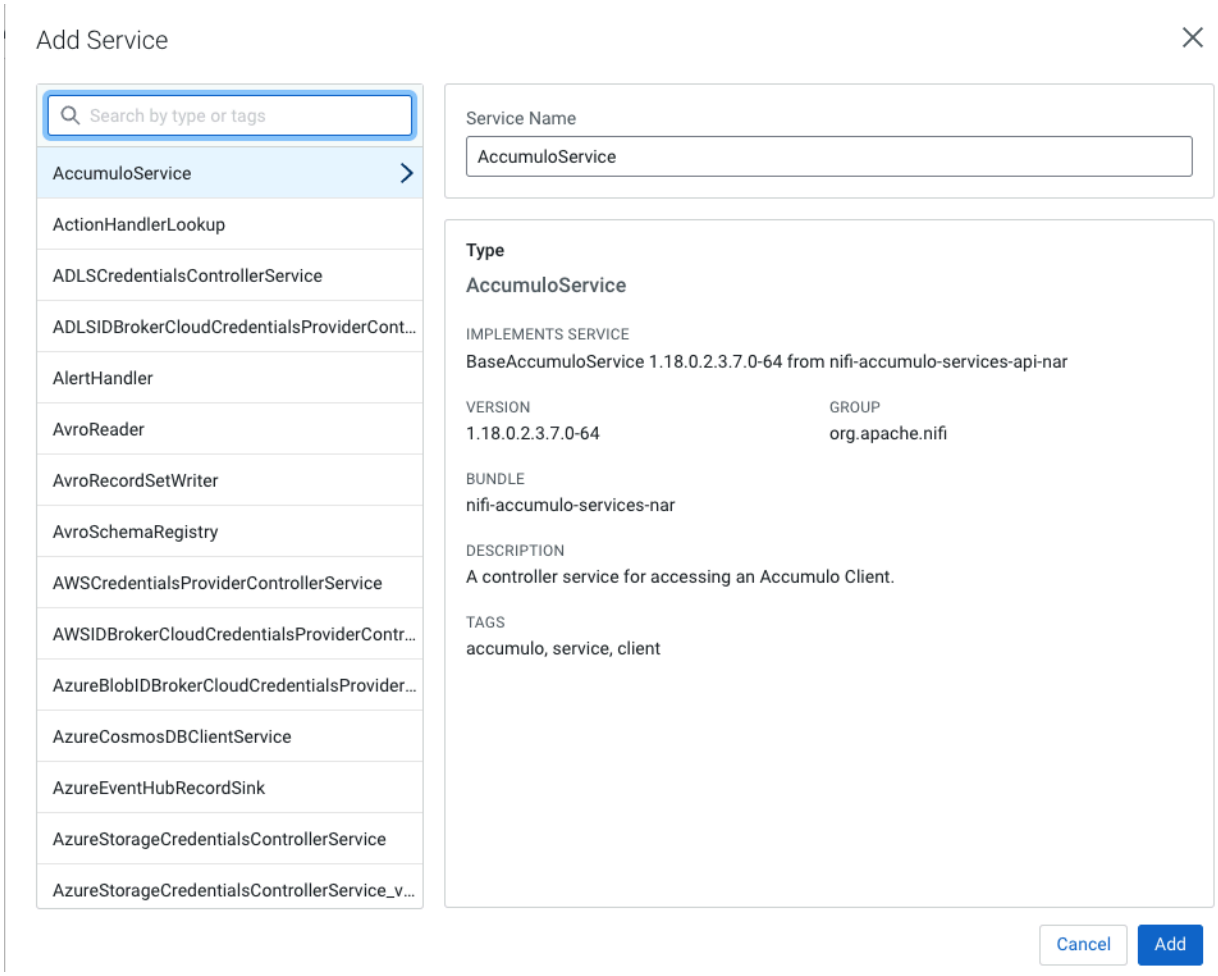
You will use the controller services you create now to configure the behavior of several processors you will add to your flow as you are building it.

Procedure

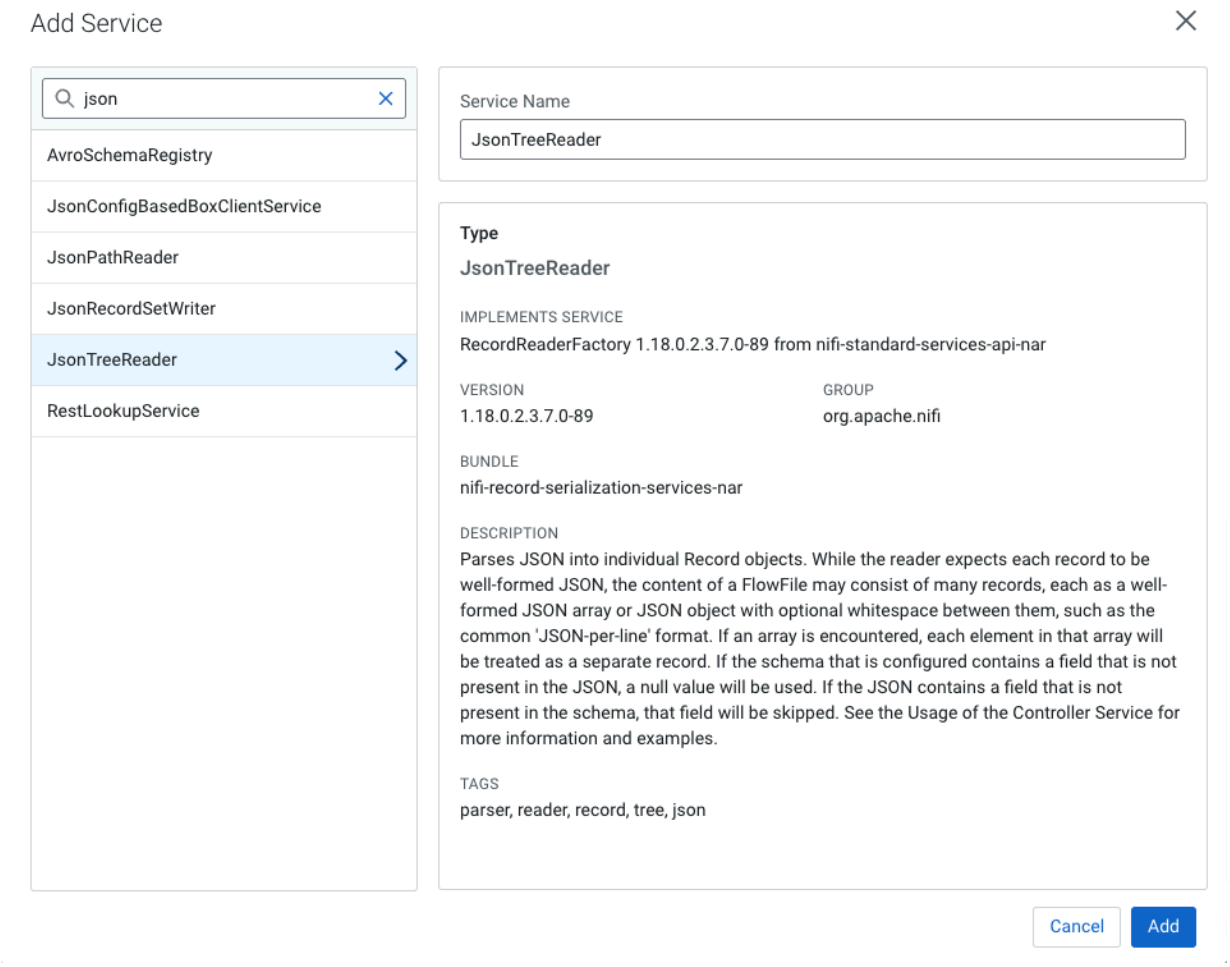
1. Go to Flow Options Services .



2. Click the Add Service button.
The **Add Service** page displays.



3. In the Search field, filter for JsonTreeReader.



4. Provide the Service Name as JSON_Reader_Recent_Changes.
5. Click the Add button.

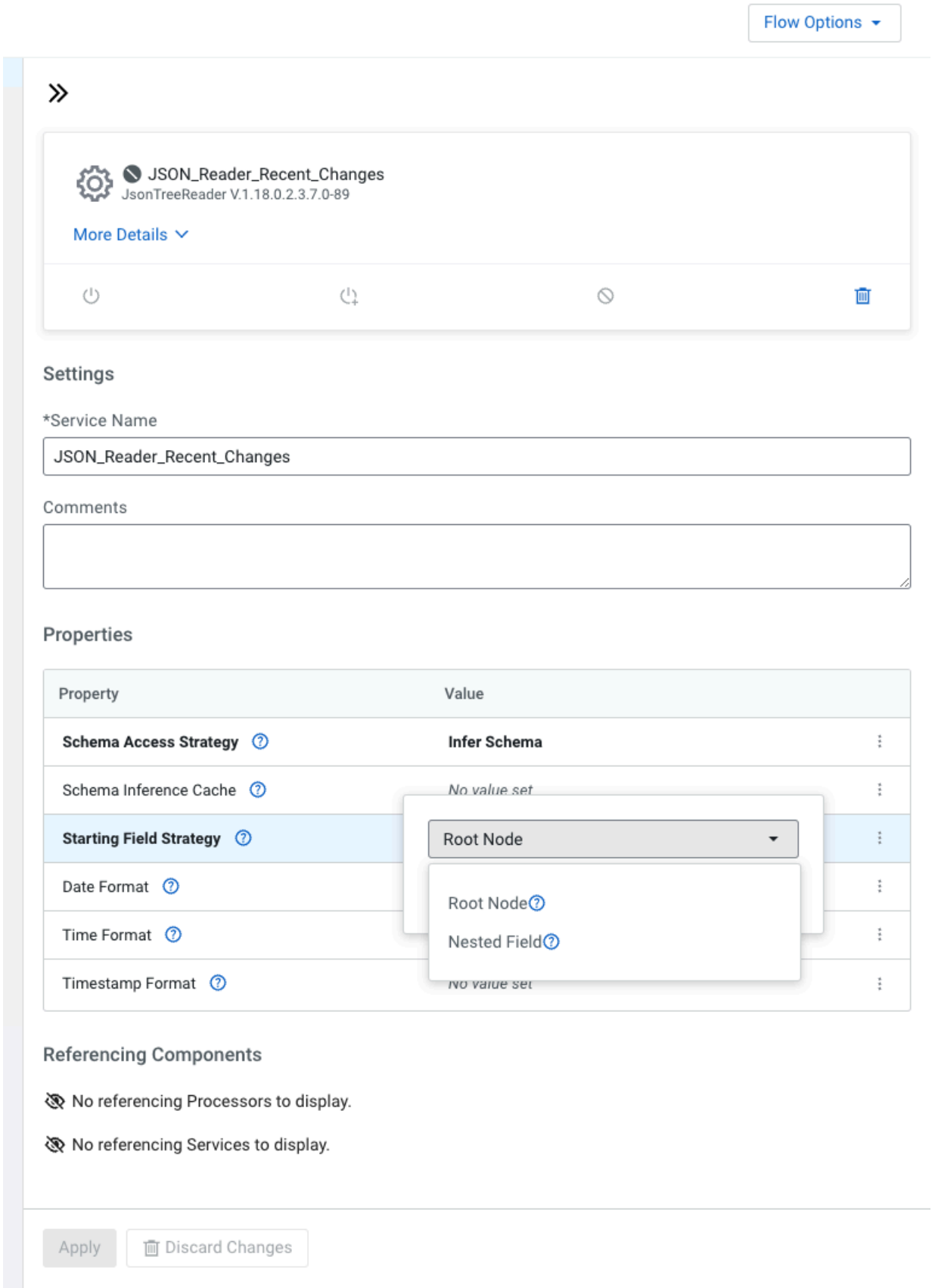
6. Configure the JSON_Reader_Recent_Changes service by setting the following Properties:

   Starting Field Strategy

   Set to Nested Field.

   Starting Field Name

   Set to recentchanges.



7. Click the Apply button.

8. Click the Add Service button to create another service.
9. In the Search field, filter for AvroRecordSetWriter.
10. Provide the Service Name as AvroWriter_Recent_Changes.
11. Click the Add button.

    You do not need to configure the AvroWriter_Recent_Changes service. You can leave all properties with their default values.
12. Click the Add Service button to create a third service.
13. In the Search field, filter for AvroReader.
14. Provide the Service Name as AvroReader_Recent_Changes.
15. Click the Add button.

    You do not need to configure the AvroReader_Recent_Changes service. You can leave all properties with their default values.
16. In the breadcrumbs on top, click 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

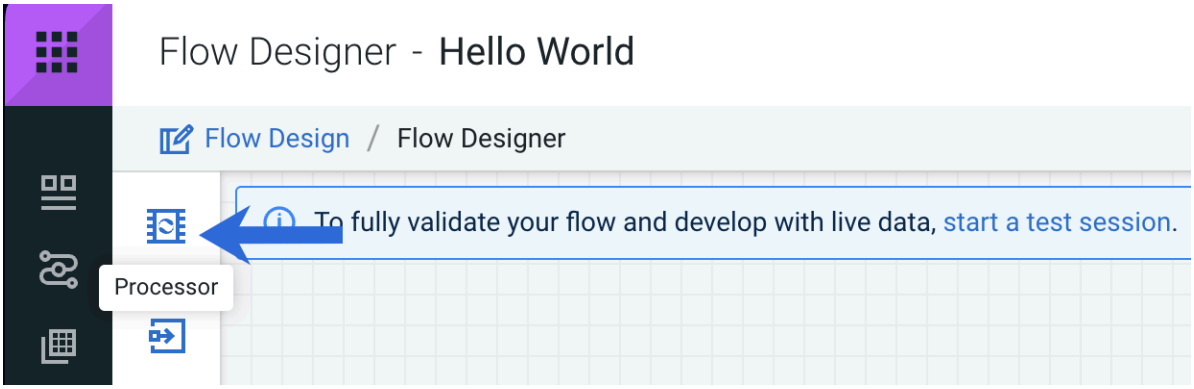
Begin creating your draft flow by adding components to the canvas and setting them up.

Procedure

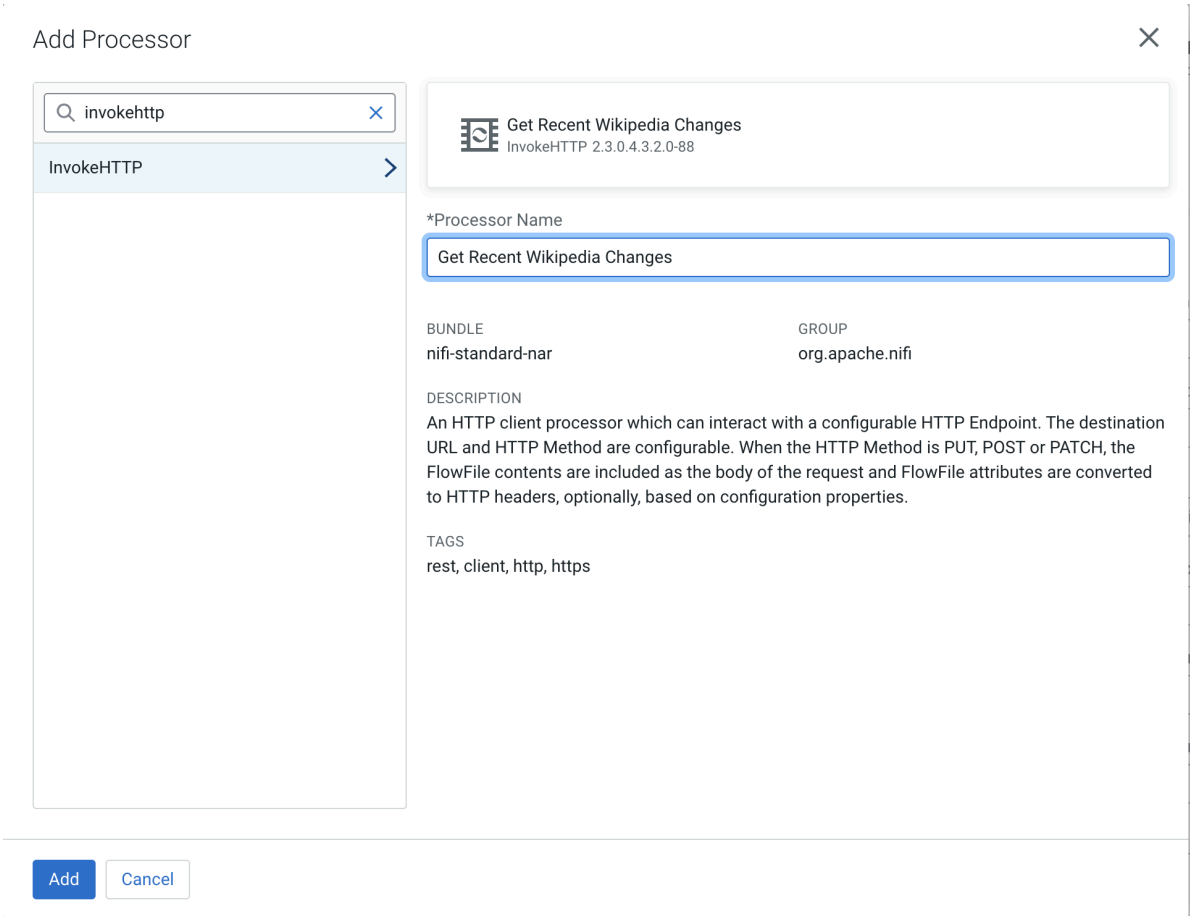
1. Add an InvokeHTTP processor to the canvas.

   Once configured, this processor will call the Wikipedia API to fetch the latest changes.

   a) Drag a Processor from the **Components** sidebar to the canvas.



   b) In the Search field, filter for InvokeHTTP.



   c) Rename the InvokeHTTP processor by changing the Processor Name to Get Recent Wikipedia Changes.

   d) Click the Add button.

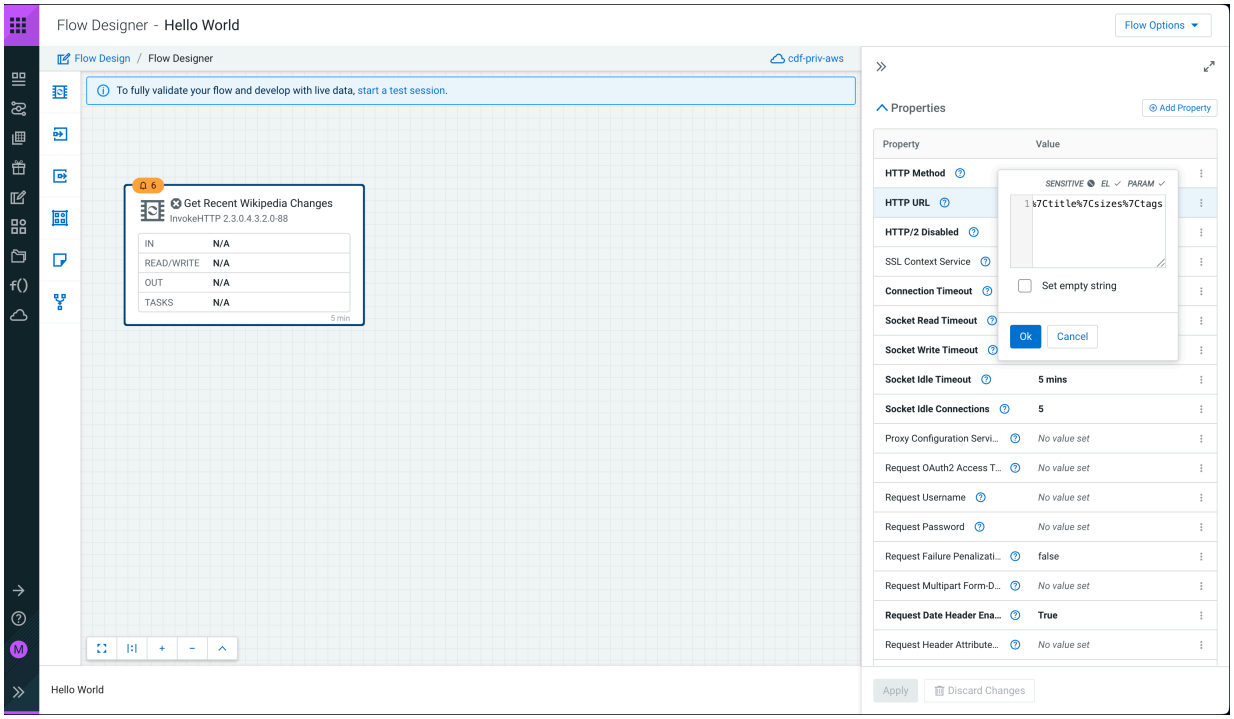
2. Configure the Get Recent Wikipedia Changes processor.

   Properties

   HTTP URL

   Enter the following URL:

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



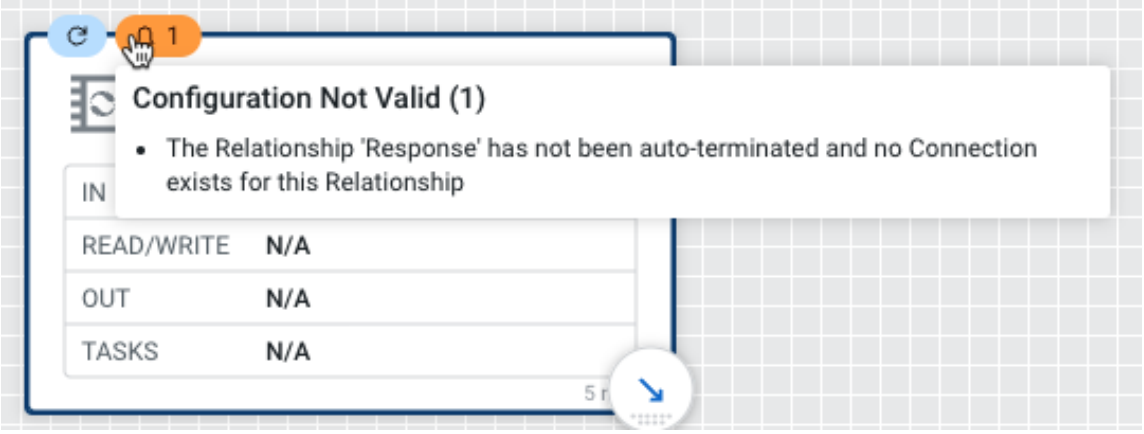
   Relationships

   Select the following relationships:

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

3. Click the Apply button.

Notice: Notice the orange Notification Pill icon in the upper left corner of your processor. The icon warns you about potential issues with a component.



In this example, the Notification Pill icon alerts you about an unaddressed Relationships configuration. The notification will disappear once you connect the processor to another one for the 'Response' relationship.

4. Add a ConvertRecord processor to the canvas.
   a) Drag a Processor from the **Components** sidebar to the canvas.
   b) In the Search field, filter for ConvertRecord.
   c) Change the Processor Name to Convert JSON to AVRO.
   d) Click the Add button.

   This processor converts the JSON response to AVRO format by using RecordReaders and RecordWriters. 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 created earlier from the dropdown list.

   Record Writer

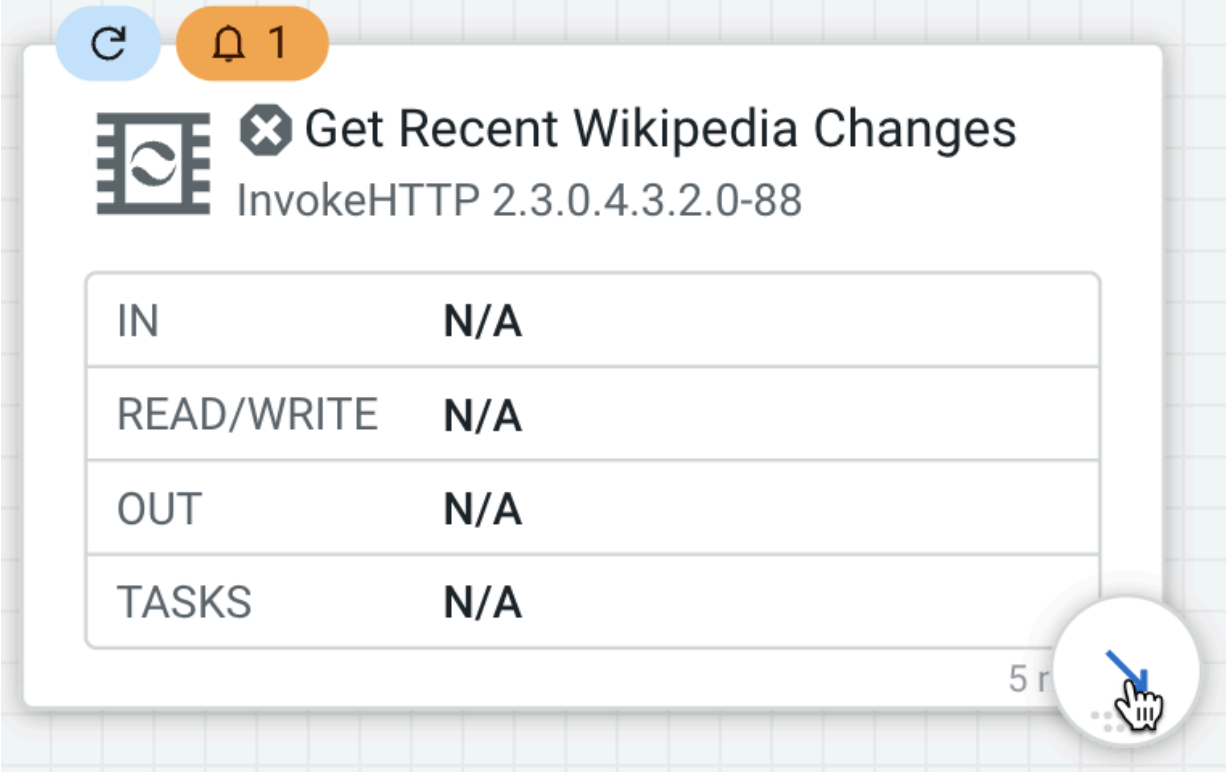
   Select the AvroWriter_Recent_Changes controller service you created earlier from the dropdown list.

   Relationships

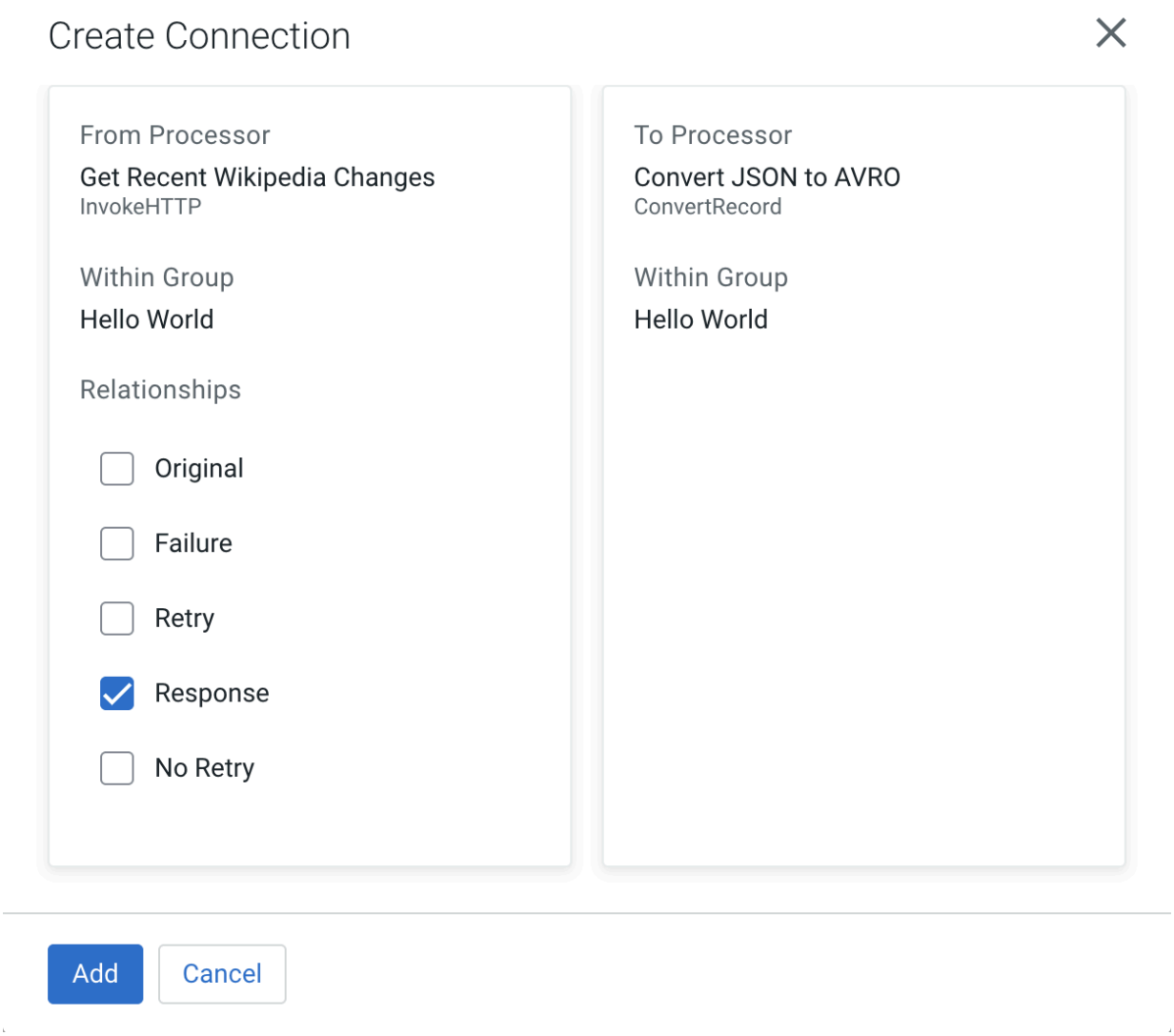
   Select the following relationships:

   Failure - Terminate, Retry
6. Click the Apply button.

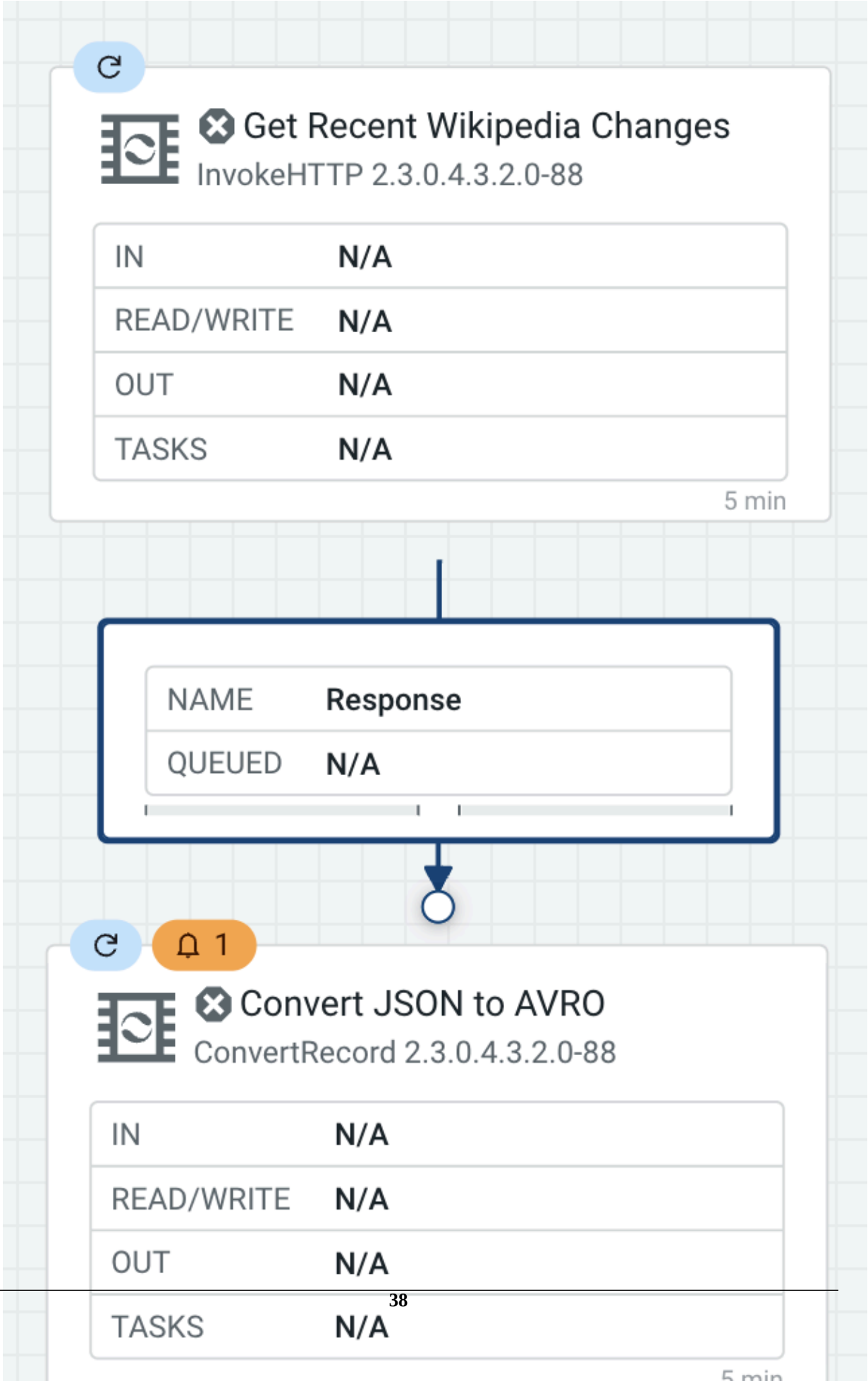
7. Connect the Get Recent Wikipedia Changes and Convert JSON to AVRO processors by hovering over the lower-right corner of the Get Recent Wikipedia Changes processor, clicking the arrow that appears and dragging it to the Convert JSON to AVRO processor.



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



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


Get Recent Wikipedia Changes
InvokeHTTP 2.3.0.4.3.2.0-88
IN N/A
READ/WRITE N/A
OUT N/A
TASKS N/A
5 min
NAME Response
QUEUED N/A
1
Convert JSON to AVRO
ConvertRecord 2.3.0.4.3.2.0-88
IN N/A
READ/WRITE N/A
OUT N/A
TASKS N/A

9. Add a QueryRecord processor.
   a) Drag a Processor from the **Components** sidebar to the canvas.
   b) In the Search field, search for QueryRecord.
   c) Change the processor Name to Filter Edits.
   d) Click the Add button.

   This processor filters out anything except actual page edits. To achieve this, the processor runs a query that selects all FlowFiles (events) of the edit type.
10. Configure the Filter Edits processor.

    Properties

    Record Reader

    Select the AvroReader_Recent_Changes controller service you have created from the dropdown list.

    Record Writer

    Select the AvroWriter_Recent_Changes controller service you have created from the dropdown list.

    Relationships

    Select the following relationships:

    - Failure - Terminate
    - Original - Terminate
11. For the Filter Edits processor you also must add a user-defined property. Click the Add Property button.

    a) Provide the Name as Filtered edits.
    b) Provide the Value as Select * from FLOWFILE where type='edit'.
    c) Click the Apply button.
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 the Add button.
14. Add a second QueryRecord processor.
    a) Drag a Processor from the Components sidebar to the canvas.
    b) In the Search field, filter for QueryRecord.
    c) Change the processor Name to Route on Content Size.
    d) Click the Add button.

    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 dropdown list.

Record Writer

Select the AvroWriter_Recent_Changes controller service you have created from the dropdown list.

Relationships

Select the following relationships:

- Failure - Terminate, Retry
- Original - Terminate

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

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

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 Search field, filter for MergeRecord.
c) Change the processor Name to Merge Edit Events.
d) Click the Add button.
e) Repeat steps a. to d. 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 dropdown list.

Record Writer

Select the AvroWriter_Recent_Changes controller service you have created from the dropdown list.

Max Bin Age

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

Relationships

Select the following relationships:

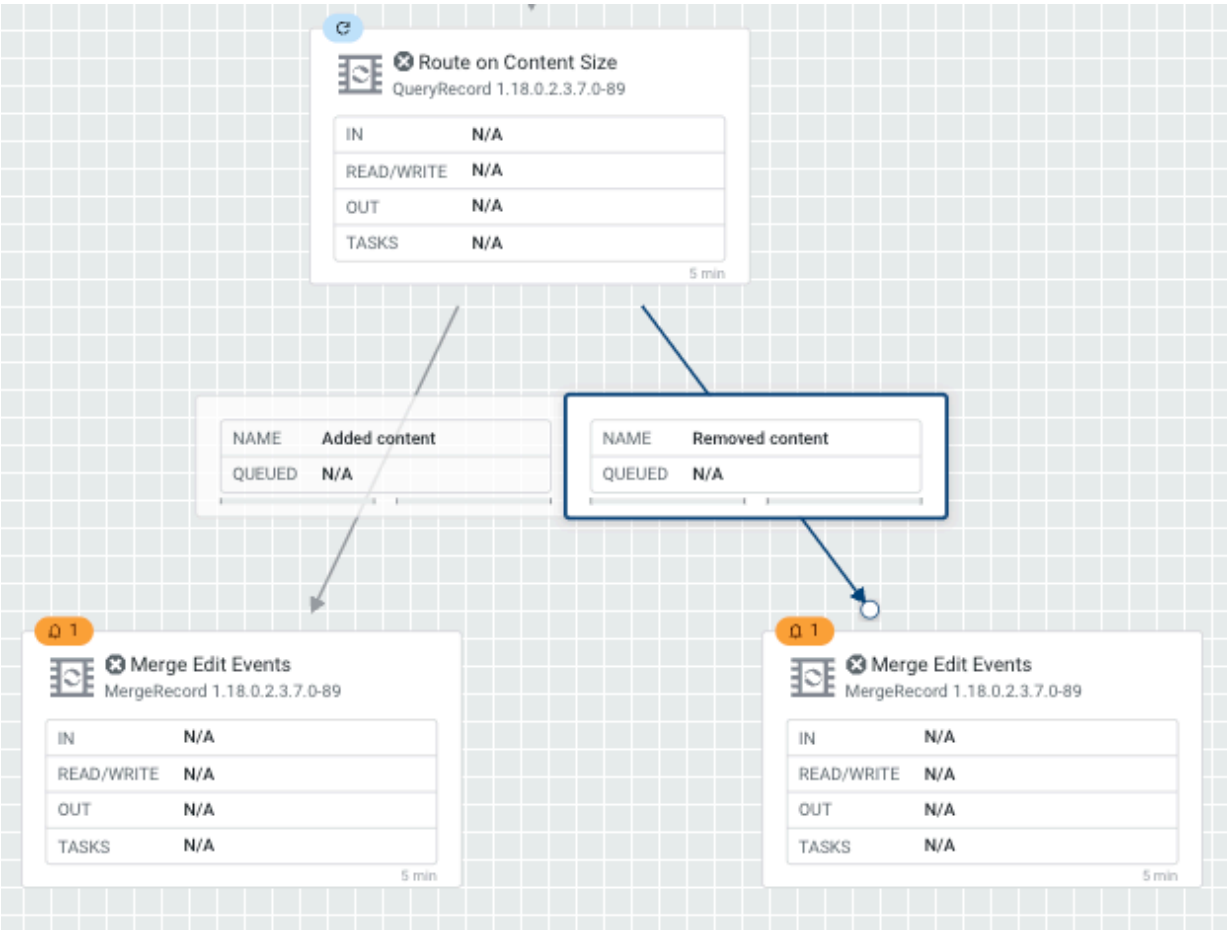
- Failure - Terminate
- Original - Terminate

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.

a) For the first Merge Edit Events processor, select Added content value from the Relationships options and click the Add button.

b) For the second Merge Edit Events processor, select the Removed content value from the Relationships option and click the Add button.



21. Add two PutFile processors to the canvas.
    a) Drag a Processor from the Components sidebar to the canvas.
    b) In the Search field, filter for PutFile.
    c) Change the processor Name to Write "Added Content" Events To File.
    d) Click the Add button.
    e) Repeat steps a. to d. 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 Data Flow, you typically do not write to local disk but replace these processors with processors that resemble your destination system, such as Kafka, Database, or Object Store.
22. Configure the Write "Added Content" Events To File processor.

    Properties:
    Directory

    Set to /tmp/larger_edits.

    Maximum File Count

    Set to 500.

    Relationships

    Select the following relationships:

    - Failure - Terminate
    - Success - Terminate
23. Click the Apply button.
24. Configure the Write "Removed Content" Events To File processor.

    Properties:
    Directory

    Set to /tmp/smaller_edits.

    Maximum File Count

    Set to 500.

    Relationships

    Select the following relationships:

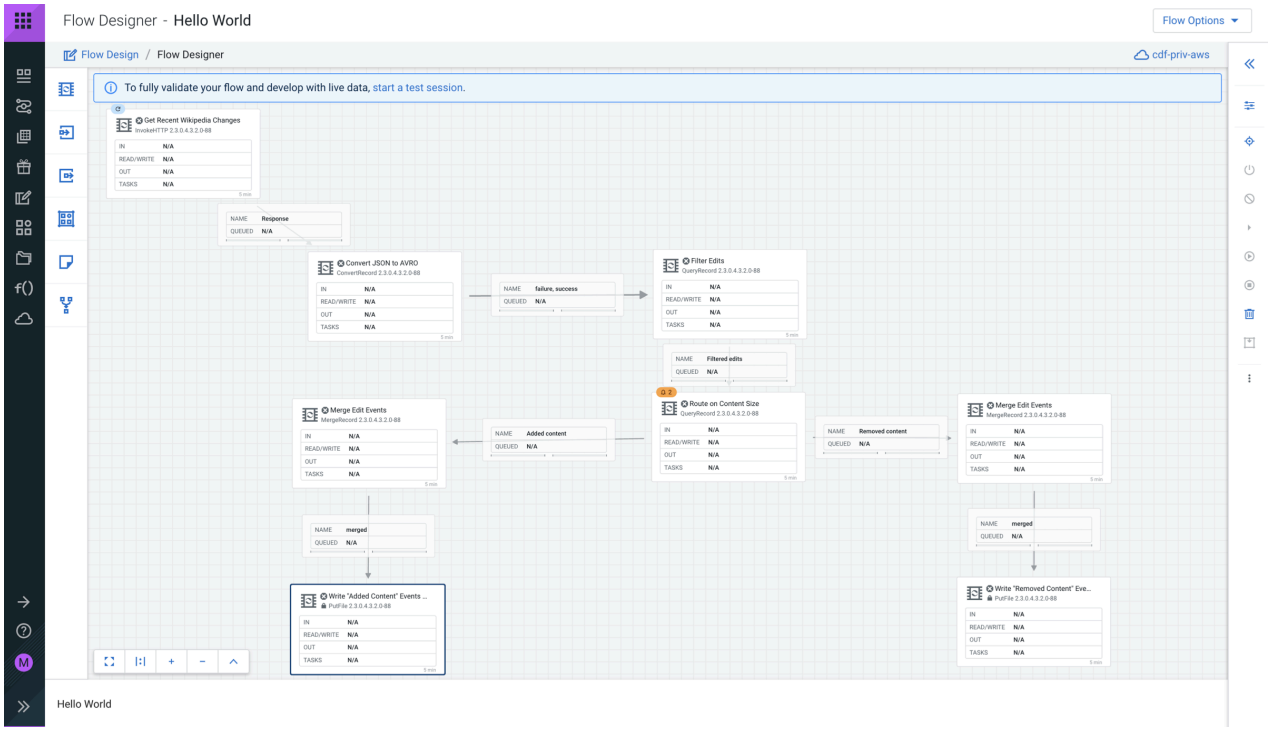
    - Failure - Terminate
    - Success - Terminate
25. Click the Apply button.
26. Connect the Merge Edit Events processor with the Added content connection to the Write "Added Content" Events To File processor.

    In the Create Connection modal select merged and click the Add button.
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 modal select merged and click the Add button.

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.

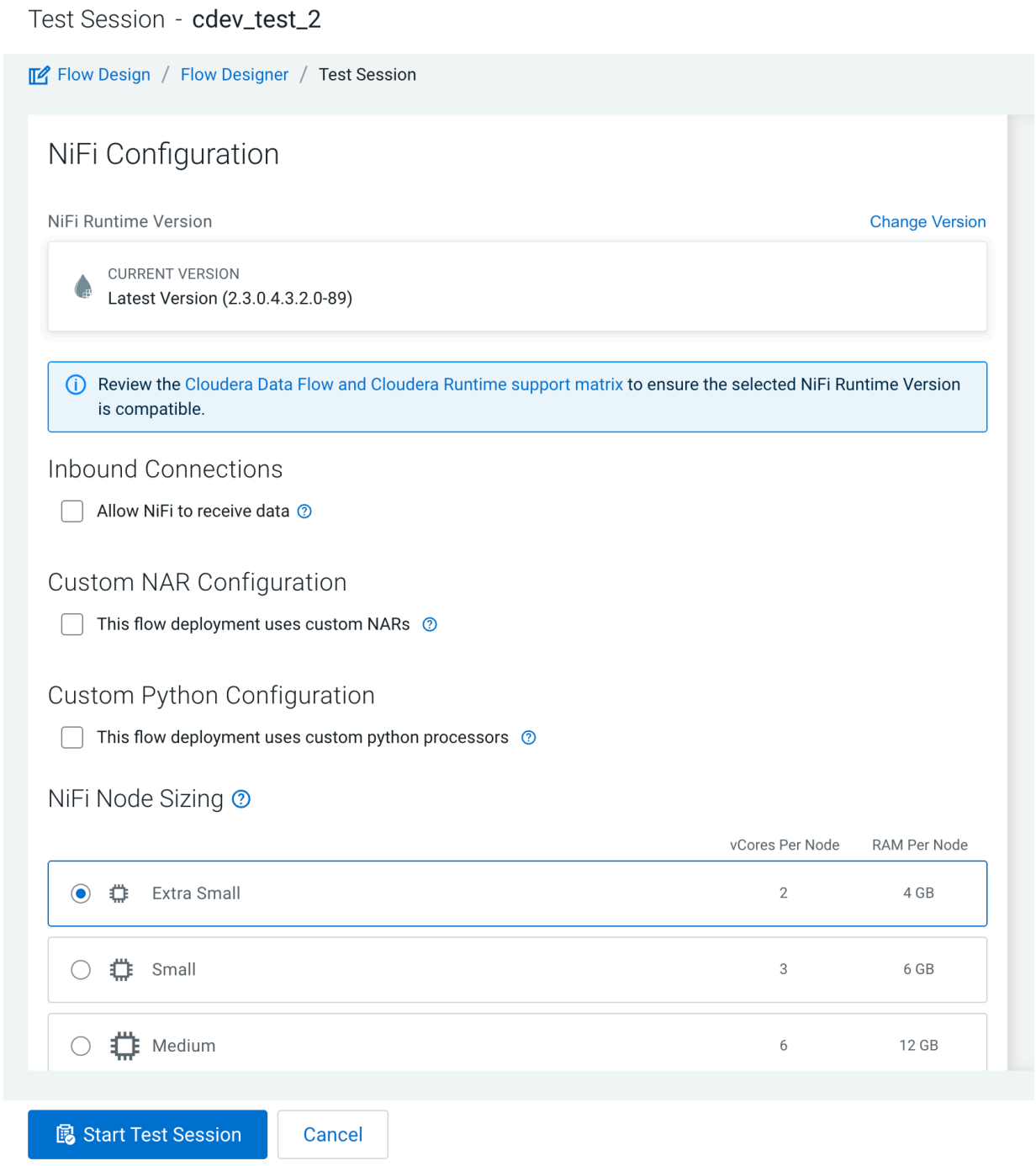
About this task

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

2. Review the suggested NiFi configuration. You do not need to change anything.



3. Click the Start Test Session button.

   Test Session status Initializing Test Session... appears on top of the page.
4. Wait for the status to change to

   Active Test Session.

   This can take several minutes.
5. Go to Flow Options Services to enable Controller Services.

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

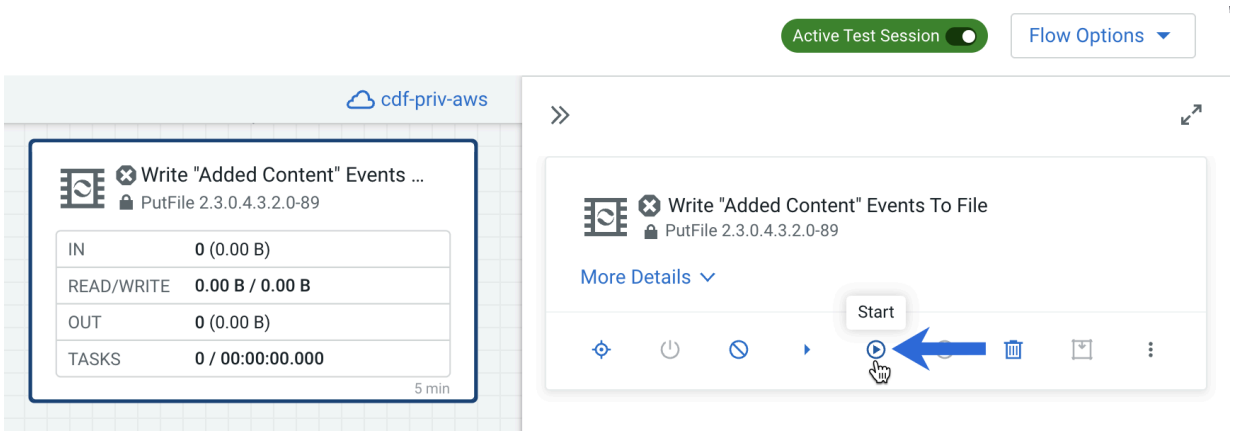
   This option does not only enable the controller service, but also any component that references it. This way, you do not need to enable the component separately to run the test session. In the context of this tutorial, enabling the AvroReader_Recent_Changes controller service will also enable the Filter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

   Repeat this step for all Controller Services.
7. Click the Flow Designer link in the breadcrumbs on the top of the page to return to the Flow Design canvas.
8. Start the remaining inactive processors.

   a) Start the Get Recent Wikipedia Changes processor by selecting it on the canvas, then clicking the [Start] icon in the component details pane.

   b) Start the Write "Added Content" Events To File processor by selecting it on the canvas, then clicking the [Start] icon in the component details pane.

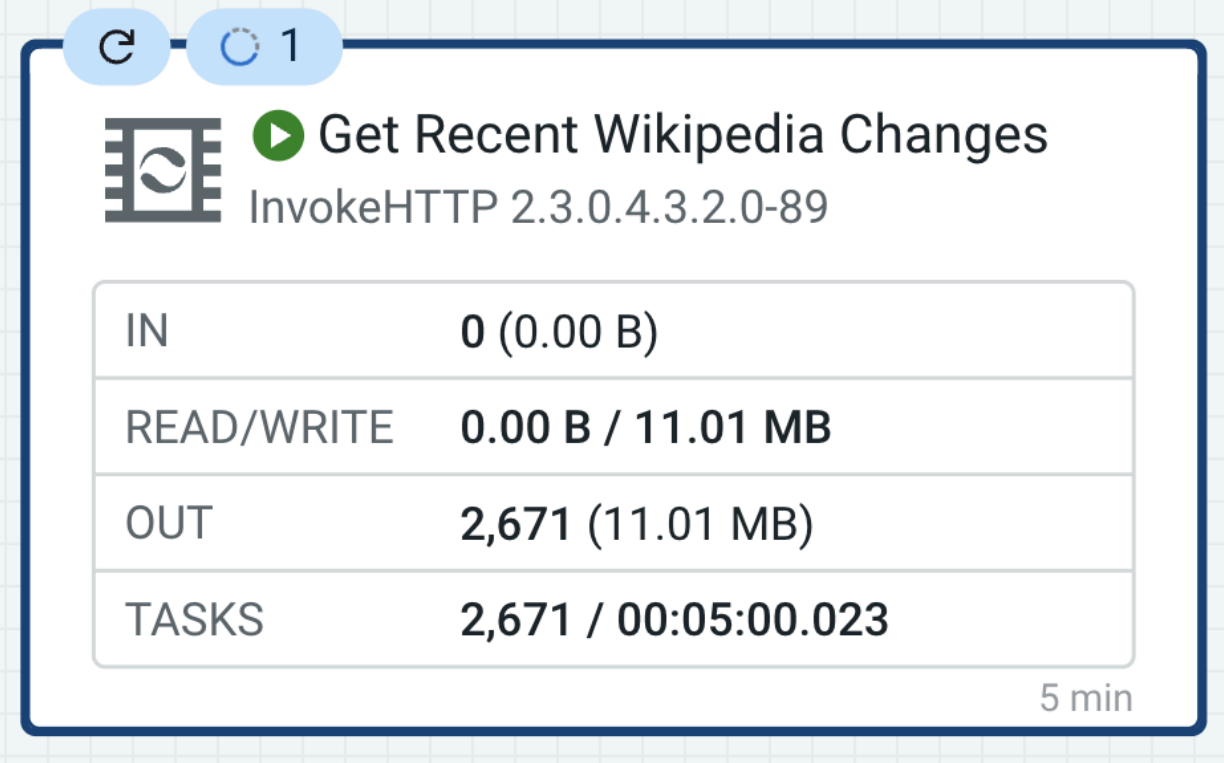
   c) Start the Write "Removed Content" Events To File processor by selecting it on the canvas, then clicking the [Start] icon in the component details pane.



All other components were auto-started when you selected the Enable Service and Referencing Components option in the Flow Options Services view.

9. Observe your first draft flow processing data.

   On the **Flow Designer** 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 Data Flow deployment.

Procedure

1. On the **Flow Designer** canvas, go to Flow Options Publish To Catalog Publish .
2. Fill in the fields in the **Publish A New Flow** modal window.
   - 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 the Publish button.

Results

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