Cloudera Data Flow

Creating flow deployments

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

CLOUDƎRA

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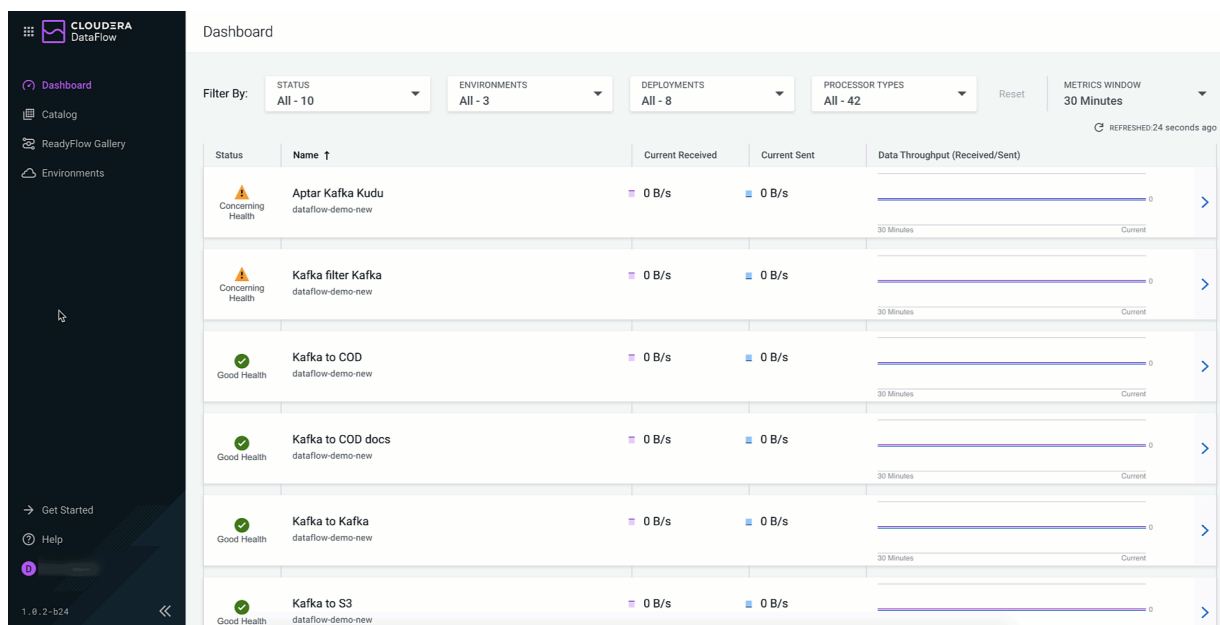
2. Select the environment where you want to deploy the flow.

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.

Import Parameters   ✕

ⓘ You are importing a Shared Parameter Group. To modify parameters in this group, go to the Resources/Workspace Resources↗ page.

Shared Parameter Group

test_env_parameters	⊘2 ▾
test_env_parameters	⊘2
test_parameters	⊘1

☑	Parameter ↑	Current Value	Value to Import
☐	empty param ⓘ		*Empty string set*
☑	file param	/nifi-shared-assets/test/file para...	→ /nifi-shared-assets/test/file para...
☐	files param ⓘ		/nifi-shared-assets/test/files par...
☑	local file	*No value set*	→ assss
☐	sens param ⓘ		👁 *Sensitive value set*
☐	text param ⓘ		test

Import   Cancel

- 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.

Related Information

Shared parameter groups

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.

Deploying a flow definition using the CLI

Deploy a flow definition to run NiFi flows as flow deployments in Cloudera Data Flow. To do this, launch the CDP CLI and specify your environment, parameters, sizing, and KPIs.
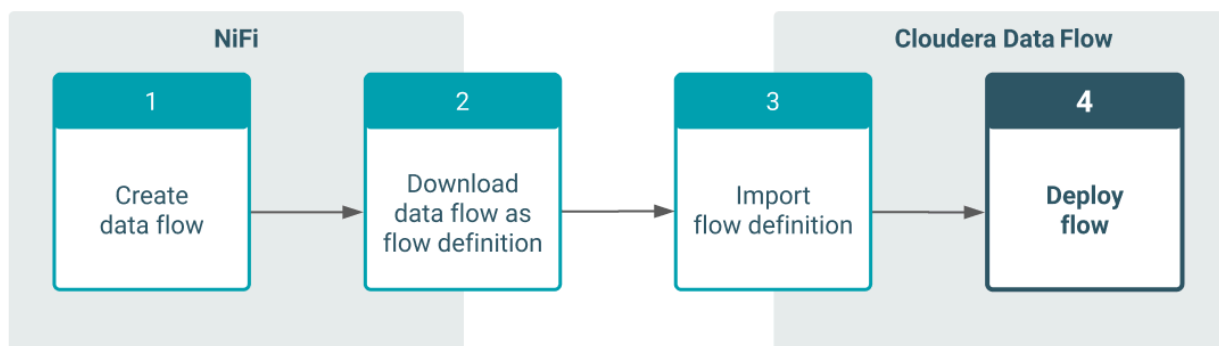
Before you begin

- You have installed CDP CLI.
- Run cdp df list-services to get the service-crn value.
- Run cdp df list-flows to get the flow-version-crn value.

Procedure

To deploy a flow, enter:

```
cdp df create-deployment
        --service-crn [***SERVICE CRN VALUE***]
        --flow-version-crn [***FLOW VERSION CRN VALUE***]
        --deployment-name [***USER SPECIFIED NAME OF THE DEPLOYMENT***]
        [--cluster-size name=[EXTRA_SMALL|SMALL|MEDIUM|LARGE|CUSTOM]]
        [--static-node-count [***NUMBER OF NODES IF AUTOSCALING IS
 DISABLED***]]
        [--auto-scaling-enabled | --no-auto-scaling-enabled]
        [--auto-scale-min-nodes [***MINIMUM NUMBER OF NODES IF AUTOSCALING
 IS ENABLED***]]
        [--auto-scale-max-nodes [***MAXIMUM NUMBER OF NODES IF AUTOSCALING
 IS ENABLED***]]
        [--cfm-nifi-version [***NIFI-VERSION***]]
        [--auto-start-flow | --no-auto-start-flow]
        [--parameter-groups [***LOCATION OF PARAMETERS JSON FILE***]]
        [--kpis [***LOCATION OF KPIS JSON FILE***]]
        [--node-storage-profile-name [STANDARD_AWS|STANDARD_AZURE|
PERFORMANCE_AWS|PERFORMANCE_AZURE|CUSTOM_AWS|CUSTOM_AZURE]]
```

Where:

--service-crn

> Specifies the service-crn value you obtained when completing the prerequisites.

--flow-version-crn

> Specifies the flow-version-crn value you obtained when completing the prerequisites.

--deployment-name

> Specifies a unique name for your flow deployment.

[--cluster-size name]

> Specifies the cluster size. Valid values are:
>
> - EXTRA_SMALL
> - SMALL
> - MEDIUM
> - LARGE
> - CUSTOM
>
>   If you specify CUSTOM, you need to specify the number of virtual cores per node (coresPerNode) and the amount of RAM per node (memoryLimit) in addition.
>
>   For example:
>
>   ```
>   --cluster-size name=CUSTOM,coresPerNode=4,memoryLimit=16
>   ```
>
> The default value is EXTRA_SMALL.

[--static-node-count]

> Specifies the number of NiFi nodes when autoscaling is not enabled. You can select between 1 and 32 nodes. The default value is 1.

[--auto-scaling-enabled | --no-auto-scaling-enabled]

> Specifies whether you want to enable autoscaling. The default is to disable autoscaling.
> **[--auto-scale-min-nodes]**

> Specifies the minimum number of nodes when you have autoscaling enabled. If you have autoscaling enabled, this parameter is required.

> **[--auto-scale-max-nodes]**

> Specifies the maximum number nodes when autoscaling is enabled. If you have autoscaling enabled, this parameter is required.

[--cfm-nifi-version]

> Specifies the NiFi runtime version. The default is the latest version.

[--auto-start-flow | --no-auto-start-flow]

> Specifies whether you want to automatically start your flow once it has been deployed. The default is to enable the automatic start.

[--parameter-groups]

> Specifies the location of the parameter group JSON file, if you are using one for this flow deployment.

[--kpis]

> Specifies the location of the KPIs JSON file, if you are providing KPIs for this flow.

--node-storage-profile-name

> Specifies the storage profile. Valid values are:

> - STANDARD_AWS
> - STANDARD_AZURE
> - PERFORMANCE_AWS
> - PERFORMANCE_AZURE
> - CUSTOM_AWS
> - CUSTOM_AZURE

> The default values are STANDARD_AWS and STANDARD_AZURE, depending on the cloud provider.

Example parameter group file:

> **Note:** The JSON file you develop for parameter group will be different depending on your flow objectives and requirements. This is an example of the parameter group file format.

```
[
  {
    "name": "kafka-filter-to-kafka",
    "parameters": [
      {
        "name": "CDP Workload User",
        "assetReferences": [],
        "value": "srv_nifi-machine-ingest"
      },
      {
        "name": "CDP Workload User Password",
        "assetReferences": [],
        "value": "<<CDP_MISSING_SENSITIVE_VALUE>>"
      },
      {
        "name": "CSV Delimiter",
        "assetReferences": [],
        "value": ","
      },
```

```
        {
          "name": "Data Input Format",
          "assetReferences": [],
          "value": "CSV"
        },
        {
          "name": "Data Output Format",
          "assetReferences": [],
          "value": "JSON"
        },
        {
          "name": "Filter Rule",
          "assetReferences": [],
          "value": "SELECT * FROM FLOWFILE"
        },
        {
          "name": "Kafka Broker Endpoint",
          "assetReferences": [],
          "value": "streams-messaging-broker0.pm-sandb.a465-9q4k.cloudera.sit
e:9093"
        },
        {
          "name": "Kafka Consumer Group ID",
          "assetReferences": [],
          "value": "cdf"
        },
        {
          "name": "Kafka Destination Topic",
          "assetReferences": [],
          "value": "MachineDataJSON"
        },
        {
          "name": "Kafka Producer ID",
          "assetReferences": [],
          "value": "cdf"
        },
        {
          "name": "Kafka Source Topic",
          "assetReferences": [],
          "value": "MachineDataCSV"
        },
        {
          "name": "Schema Name",
          "assetReferences": [],
          "value": "SensorReading"
        },
        {
          "name": "Schema Registry Hostname",
          "assetReferences": [],
          "value": "streams-messaging-master0.pm-sandb.a465-9q4k.cloudera.site
"
        }
      ]
    }
]
```

Example KPI file:

Note: The JSON file you develop for KPIs will be different depending on your flow objectives and requirements. This is an example of the KPI file format.

```
[
  {
```

```
      "metricId": "rateBytesReceived",
      "alert": {
        "thresholdLessThan": {
          "unitId": "kilobytesPerSecond",
          "value": 150
        },
        "frequencyTolerance": {
          "unit": {
            "id": "MINUTES"
          },
          "value": 1
        }
      }
    },
    {

      "metricId": "processorAmountBytesSent",
      "alert": {},
      "componentId": "a7f7df1c-a32d-3c25-9b09-e1d1036dcc04;a33a1b48-005b-32dd-
bb88-b63230bb8525"
  }
]
```

Results

Successfully deploying a flow results in output similar to:

```
{
  "crn": "[***DEPLOYMENT CRN***]"
}
```