Cloudera Data Engineering 1.25.0

Orchestrating workflows and pipelines with Apache Airflow in Cloudera Data Engineering

Date published: 2020-07-30
Date modified: 2025-11-20

https://docs.cloudera.com/

Legal Notice

Contents

Apache Airflow in Cloudera Data Engineering

Learn about how Apache Airflow is integrated with Cloudera Data Engineering and how to automate a workflow or data pipeline using Apache Airflow Python DAG files in Cloudera Data Engineering.

Cloudera Data Engineering enables you to automate a workflow or data pipeline using Apache Airflow Python DAG files. Each Cloudera Data Engineering Virtual Cluster includes an embedded instance of Apache Airflow. You can also use Cloudera Data Engineering with your own Airflow deployment. For more information about using your own Cloudera Data Engineering Airflow deployment, see Using Cloudera Data Engineering with an external Apache Airflow deployment.

Cloudera Data Engineering currently supports multiple Airflow operators. For example, one for running Cloudera Data Engineering jobs, or one for accessing and executing SQL commands on Cloudera Data Warehouse. For more information about the complete list of installed and supported operators, see Supported Airflow operators and hooks.

You can create and manage Apache Airflow jobs by writing or creating Python DAG files and uploading them using the UI. For more information about Cloudera Data Engineering Airflow job management, see Creating and managing Cloudera Data Engineering Airflow Jobs using the Cloudera Data Engineering UI.

Cloudera Data Engineering provides various ways to connect to Cloudera Data Warehouses or to other Cloudera Data Engineering Virtual Clusters. You must create an Airflow connection containing the target system access details and then refer it in the associated operators to execute the required workloads.

- Creating and managing Airflow connections: To connect to a Cloudera Data Warehouse or Cloudera Data Engineering Virtual Cluster, you must create an Airflow connection.
- Executing SQL queries on Cloudera Data Warehouse: To execute SQL queries on Hive or Impala Virtual Warehouses, you can use the installed SQLExecuteQueryOperator or the CdwExecuteQueryOperator.
- Running jobs on other Cloudera Data Engineering Virtual Clusters: With an existing Cloudera Data Engineering connection, you can use the CDERunJobOperator to execute jobs on other Cloudera Data Engineering Virtual clusters.

You can also install and use custom operators and libraries (Python packages) for Airflow with Cloudera Data Engineering. Cloudera provides a way to extend the installed default packages with the third party or custom Python packages using the Custom Operators and Libraries feature using the Cloudera Data Engineering user interface (UI).

Important:

- Direct access to the native Airflow API is not supported because bypassing the Cloudera Data Engineering API prevents job tracking, centralized monitoring, and metadata tracking.
- The Cloudera Data Engineering API and the Airflow UI do not support clearing an Airflow task or a DAG run. You can perform this action only by using the Airflow admin command (for example, from the airflow-scheduler container). However, clearing an Airflow task or a DAG run causes a re-run of that task or DAG run and creates a new Cloudera Data Engineering Airflow job run for the re-run within Cloudera Data Engineering. Because this re-run occurs, the Airflow UI only updates the original job run's details in Airflow. Hence, there can be multiple Cloudera Data Engineering job runs pointing to the same Airflow job run.

Related Information

Provider packages

Supported Airflow operators and hooks

Enabling SSO to a Virtual Warehouse

Creating jobs in Cloudera Data Engineering

Using Cloudera Data Engineering with an external Apache Airflow deployment

Creating and managing Airflow jobs using Cloudera Data Engineering

Learn about how to create and manage the Apache Airflow jobs using Cloudera Data Engineering.

Creating Airflow jobs using Cloudera Data Engineering

Learn about how to create Airflow jobs using Cloudera Data Engineering.

About this task

An Airflow job in Cloudera Data Engineering consists of an Airflow DAG file and various optional resources. Jobs can be run on demand or scheduled.

Important: The default Airflow worker pod resource requests limits are as follows:

- CPU requests = 1
- CPU limits = No limit
- Memory requests = 2 Gi
- Memory limits = 2 Gi

Also, you can set the custom resource requests for an Airflow task through executor_config.

For example, sample DAG file is as follows:

```
from airflow import DAG
from airflow.operators.python import PythonOperator
from pendulum import datetime
from kubernetes.client import models as k8s

config_resource_requirements = {
    "pod_override": k8s.V1Pod(
        spec=k8s.V1PodSpec(
            containers=[
                k8s.V1Container(
                    name="base",
                    resources=k8s.V1ResourceRequirements(
                        requests={"cpu": 0.5, "memory": "1024Mi"},
                        limits={"cpu": 0.5, "memory": "1024Mi"}
                    )
                )
            ]
        )
    )
}

with DAG(
        dag_id="python_operator_custom_resources",
        start_date=datetime(2024, 1, 1),
        schedule=None,
        catchup=False,
):
    PythonOperator(
        task_id="hello_task",
        python_callable=lambda: print("Hello!"),
        executor_config=config_resource_requirements
    )
```

Procedure

1. In the Cloudera Management Console, click the Data Engineering tile and click Overview.
2. In the Cloudera Data Engineering Services column, select the service that contains the virtual cluster that you want to create a job for.
3. In the Virtual Clusters column, locate the virtual cluster that you want to use and click the View Jobs icon.
4. In the left navigation menu, click Jobs.
5. Click the Create Job button.
6. Provide the Job Details:
   a) Select Airflow for the job type. The available fields on the user interface updates automatically.
   b) Specify the Name.
   c) Click the File option and select the way you want to provide the DAG file. Using the File option, you can do the following:
      - Upload the DAG file to a new resource.
      - Use the DAG file from a previously created resource.
      - Upload any other resources required for the job.

      For more information about the Editor, see Creating an Airflow DAG using the Pipeline UI.
7. If you do not want to run the job immediately, click the Create and Run drop-down menu and then click Create. Otherwise, click Create and Run to run the job immediately.

   Note: If you select the Create and Run option from the Create and Run drop-down menu, the DAG file must not be paused while creating and provide a False value to the is_paused_upon_creation field.

Creating an Airflow DAG using the Pipeline UI

With the Cloudera Data Engineering Pipeline UI, you can create multi-step pipelines with a combination of available operators.

About this task

Note: Cloudera supports all major browsers (Google Chrome, Firefox and Safari) for this feature. If you are using a browser in incognito mode, you have to allow all cookies in your browser settings so that you can view Pipelines, Spark, and Airflow pages.

Procedure

1. Go to Jobs Create Job .

   Under Job details, select Airflow.

   The UI refreshes, only Airflow-specific options remain.
2. Specify a name for the job.
3. Under DAG File select the Editor option.
4. Click Create.
   You are redirected to the job Editor tab.

   Note: The Cloudera Data Engineering Airflow Editor's Python Script (using Airflow Pure PythonOperator) is not automatically integrated with the Data Lake. As a result, when you run code through Create Job Type: Airflow Editor Python Script Run , the output does not have connection to the Data Lake resources.

   To enable Airflow Pure PythonOperator integration with the Data Lake, manually configure the required connections and settings for the Data Lake connection.

5. Build your Airflow pipeline.
    - Drag and drop operators to the canvas from the left hand pane.
    - When selecting an operator, you can configure it in the editor pane that opens up.

      On the Configure tab you can provide operator-specific settings. The Advanced tab allows you to make generic settings that are common to all operators, for example execution timeout or retries.
    - Create dependencies between tasks by selecting them and drawing an arrow from one of the four nodes on their edges to another task. If the dependency is valid the task is highlighted in green. If invalid, it is highlighted in red.
    - To modify DAG-level configuration, select Configurations on the upper right.
6. When you are done with building your pipeline, click Save.

Running Airflow jobs using Cloudera Data Engineering

Jobs in Cloudera Data Engineering can be run on demand, or scheduled to run on an ongoing basis. Learn about how to run a job in Cloudera Data Engineering.

Procedure

1. In the Cloudera Management Console, click the Data Engineering tile and click Overview.
2. In the Cloudera Data Engineering Services column, select the environment containing the virtual cluster where you want to run the job.
3. In the Virtual Clusters column on the right, click the View Jobs icon on the virtual cluster containing the job you want to run.
4. To run a job immediately, click ⋮ in the Actions column next to the job, and then click Run Now. The job must not be paused. If the Job is paused, there is an option named Resume Schedule in the menu, which allows you unpause it.

   Job Run Notices: The running jobs provide notifications, in the form of a Bell icon next to the Job Run ID, when certain conditions are met, without having to parse low level logs, or navigating away to a cloud provider or Kubernetes interface. This allows you to identify why certain job is running slow or stuck, and take actions to rectify this.

Deleting Airflow jobs using Cloudera Data Engineering

If you no longer need a job, you can delete it. Deleting a job does not delete the job run history.

Before you begin

An Airflow job can't be deleted if it has active runs. Before deleting a job, make sure that every run is either completed or killed.

Procedure

1. In the Cloudera Management Console, click the Data Engineering tile and click Overview.
2. In the Cloudera Data Engineering Services column, select the environment containing the virtual cluster where you want to delete the job.
3. In the Virtual Clusters column on the right, click the View Jobs icon on the virtual cluster containing the job you want to delete.
4. Click ⋮ in the Actions column next to the job, and then click Delete.

Managing an Airflow Pipeline using the CDE CLI

Based on your business requirement, you can use Cloudera Data Engineering CLI to create basic Airflow pipelines or multi-step pipelines with a combination of available operators, to enable data-driven decisions. You can update these Airflow pipelines by updating the DAG files and job configurations.

Creating a pipeline using the CDE CLI

You can update the following properties in an Airflow pipeline:

Related Information

Using the Cloudera Data Engineering CLI

Creating a basic Airflow pipeline using CDE CLI

By creating a basic pipeline in Cloudera Data Engineering using the CLI, you can create multi-step pipelines with a combination of available operators.

About this task

To create a basic pipeline in Cloudera Data Engineering, you must upload the Airflow (Directed Acyclic Graph) DAG to a Cloudera Data Engineering resource and create a Cloudera Data Engineering Airflow job from this DAG.

Procedure

In the CDE CLI, run the following command:

```
cde resource create --name my_pipeline_resource
cde resource upload --name my_pipeline_resource --local-path my_pipeline_dag
.py
cde job create --name my_pipeline --type airflow --dag-file my_pipeline_dag.
py --mount-1-resource my_pipeline_resource
```

Related Information

Using the Cloudera Data Engineering CLI

Creating a pipeline with additional Airflow configurations using CDE CLI

By creating a pipeline with additional Airflow configurations using the Cloudera Data Engineering CLI, you can create multi-step pipelines with a combination of available operators. There are two ways to create this type of pipeline. The first method detailed below is recommended approach that we highly suggest customers use. The second is the alternative method that customers have used in the past, but is not recommended.

About this task

Airflow DAGs can be defined with parameters at the DAG-level or Task-level. These parameters can be overridden in the case of a manual run. A manual run is triggered explicitly by the user. It is recommended to use the Params approach so that default values can be used by the scheduled job instances as well.

For Params (Recommended)

An example of a DAG definition with additional Airflow configuration is as follows:

1. Create a configuration such as the example shown below:

```
from airflow import DAG
from airflow.models.param import Param

with DAG(
    "my_dag",
    params={
        # an int with a default value
        "int_param": Param(10, type="integer", minimum=0, maximum
=20),

        # a required param which can be of multiple types
        # a param must have a default value
        "dummy": Param(5, type=["null", "number", "string"]),

        # an enum param, must be one of three values
        "enum_param": Param("foo", enum=["foo", "bar", 42]),

        # a param which uses json-schema formatting
        "email": Param(
            default="example@example.com",
            type="string",
            format="idn-email",
            minLength=5,
            maxLength=255,
        ),
    },
):
  # prints <class 'str'> by default
  # prints <class 'int'> if render_template_as_native_obj=True
  my_operator = PythonOperator(
      task_id="template_type",
      op_args=[
          "{{ params.int_param }}",
      ],
      python_callable=(
          lambda x: print(type(x))
      ),
  )
```

In this case, nothing needs to be done on the cde job create step. Values can be additionally overridden in a manual run, through the --config flag of the cde job run command. For example:

```
cde job run --name my_pipeline --config key1=my_new_value1
```

For Dag run conf (Not recommended)

Note: This is an alternative method to create a pipeline with additional Airflow configurations, but it is not recommended. This method does not work with scheduled job instances.

For historical reasons Cloudera Data Engineering supports the {{ dag_run.conf }} object as well. In this case, the option, --config key=value in the cde job create command, is used to define default values whenever the user triggers a manual run using cde job run without specifying these parameters in the run command. This config option can be repeated to define multiple parameters.

1. Create a configuration such as the example shown below:

```
cde resource create --name my_pipeline_resource
cde resource upload --name my_pipeline_resource --local-path my_pipeline
_dag.py
```

```
cde job create --name my_pipeline --type airflow --dag-file my_pipeline_
dag.py --mount-1-resource my_pipeline_resource --config key1=value1 —-co
nfig key2=value2
```

The configuration can be used in a DAG as shown below:

```
my_bash_task = BashOperator(
 task_id="my_bash_task",
 bash_command="echo key1_value: {{ dag_run.conf[‘key1’] }} key2_value:
 {{ dag_run.conf[‘key2’] }}",
 dag=dag,
)
```

The configuration can also be overridden for manual runs in the same manner as described in the Recommended section on this page.

Related Information

Params

Using the Cloudera Data Engineering CLI

Creating an Airflow pipeline with custom files using CDE CLI [Technical Preview]

By creating a pipeline in Cloudera Data Engineering using the CLI, you can add custom files that are available for tasks. This is a technical preview.

Before you begin

This feature is available in Cloudera Data Engineering 1.19 and above in new Virtual Cluster installations only.

About this task

For use cases where custom files need to be accessed within an Airflow task, you need to first upload the custom files to a Cloudera Data Engineering resource, and then specify it in the job creation parameter using the --airflow-file-mount-<n>-resource option. These files are available only to the jobs in which they are linked.

The general form of the command is:

```
cde job create \
  --name <my_job_name> \
  --type airflow \
  --mount-1-resource <my_dag_resource> \
  --dag-file <my_dag_file.py> \
  --airflow-file-mount-n-resource <my_file_resource> \
  --airflow-file-mount-n-prefix <my_custom_prefix>   # Optional
```

In the --airflow-file-mount-n-resource parameter, n is an integer number (beginning at 1). This allows you to specify multiple ...-resource parameters, to mount multiple resources.

Each resource is mounted at /app/mount/<prefix>. If you do not need to specify a custom prefix, the mount point of your resource will be based on the resource name. For example, if the name of your Cloudera Data Engineering resource is 'my_resource', the files in the resource will be made available within Airflow under /app/mount/my_resource.

If you do want to specify a custom prefix for your resource's mount point, use the optional --airflow-file-mount-n-prefix parameter, specifying n as the same number as the corresponding --airflow-file-mount-n-resource parameter.

Note: Note that all resource mounts to Airflow are read-only.

Procedure

Run the following commands to upload the custom files to a Cloudera Data Engineering resource, and then create the job:

```
cde resource create --name my_pipeline_resource
cde resource upload --name my_pipeline_resource --local-path my_pipeline_dag
.py
cde resource create --name my_file_resource
cde resource upload --name my_file_resource --local-path my_file.conf

cde job create --name my_pipeline --type airflow --dag-file my_pipeline_dag.
py --mount-1-resource my_pipeline_resource --airflow-file-mount-1-resource m
y_file_resource
```

Example

The files can be reached in Airflow DAGs with the following pattern: /app/mount/<resource_name or resource_al ias>/<file_name>, like in the following example:

```
read_conf = BashOperator(
     task_id=read_conf,
     bash_command=”cat /app/mount/my_file_resource/my_file.conf”
 )
```

Note: It is possible to change the mount path by specifying the --airflow-file-mount-N-prefix my_custom_ prefix option in the job creation command, like in the following example:

```
cde job create --name my_pipeline --type airflow --dag-file my_pipeline_dag.
py --mount-1-resource my_pipeline_resource --airflow-file-mount-1-resource m
y_file_resource --airflow-file-mount-1-prefix my_custom_prefix
```

In this case, the file is available at:

```
read_conf = BashOperator(
     task_id=read_conf,
     bash_command=”cat /app/mount/my_custom_prefix/my_file.conf”
 )
```

Note: As a best practice, Cloudera recommends to use the same resource name because it is simpler to follow the DAG without having to look at the job definition. Also, it is possible to use “/” as a value to mount to / app/mount if there is only one Airflow file mounted in the job definition; however, this is not recommended.

Related Information

Using the Cloudera Data Engineering CLI

Updating a pipeline using the CDE CLI

You can update the following properties in an Airflow pipeline:

Updating a DAG file using the CDE CLI

You can update a Directed Acyclic Graph (DAG) file using the CDE CLI for instances where the DAG needs to be overridden. For use cases where the DAG needs to be overridden, first the DAG needs to be uploaded to the resource to override the previous version, then you must update the job.

About this task

Unlike a Spark job, the Airflow job does not automatically pull in the updated resource. Airflow jobs require a forced update by calling the `job update` command, such that the required files are uploaded to Airflow server for processing.

Choose one of the following options in step 1:

Important: Before updating a DAG, it is recommended to ensure the DAG is paused and is not currently running when the update is being done.

Updating a DAG file that needs to be overridden

Run the following command in the CDE CLI:

```
cde resource upload --name my_pipeline_resource --local-path my_pipeline_dag
.py
cde job update --name my_pipeline --dag-file my_pipeline_dag.py --mount-1-r
esource my_pipeline_resource
```

Updating a DAG in scenarios where the DAG must be set to a different one:

First upload the DAG to any resource and then the job needs to be updated. Run the following command in the CDE CLI:

```
cde resource upload --name my_other_pipeline_resource --local-path my_other_
pipeline_dag.py
cde job update --name my_pipeline --dag-file my_other_pipeline_dag.py --mou
nt-1-resource my_other_pipeline_resource
```

Updating the Airflow job configurations using the CDE CLI

In the case where the Airflow job was created with the --config option, the Airflow job configuration can be updated with the following command below. For more information, see Creating a pipeline using the CDE CLI linked below.

Procedure

Run the following command in the CDE CLI:

```
cde job update --name my_pipeline --config key1=new_value1 —-config key2=new
_value2
```

The new configuration is merged with the existing job configuration.

Related Information

Creating a pipeline using the CDE CLI

Updating the Airflow file mounts using the CDE CLI [Technical Preview]

You can update or delete an existing file mount, or add new Airflow file mounts for your pipeline with these commands.

Changing an existing file mount

```
cde job update --name my_pipeline --airflow-file-mount-1-resource my_new_pip
eline_resource
```

Changing a file mount prefix

```
cde job update --name my_pipeline --airflow-file-mount-1-prefix my_new_pipel
ine_resource_prefix
```

Adding a new file mount

Add a new file mount when one already exists:

```
cde job update --name my_pipeline --airflow-file-mount-2-resource my_new_pip
eline_resource
```

Removing an existing file mount

```
cde job update --name my_pipeline --unset-airflow-file-mount-index 2
```

Note: You can only delete one mount at a time. After the mount is deleted, they will be re-indexed.

Deleting an Airflow pipeline using the CDE CLI

You can delete a pipeline in Cloudera Data Engineering using the CLI.

Procedure

To delete a pipeline, give the job name to the delete command:

```
cde job delete --name my_pipeline
```

Creating and managing Airflow connections

You can create multiple connections to various services using the embedded Airflow UI.

Creating a connection to Cloudera Data Warehouse or Cloudera Data Hub instance for SQL Operator

Learn how to create an Airflow connection to an existing Cloudera Data Warehouse or a Cloudera Data Hub instance before running the workloads using the supported Airflow SQL Operators.

About this task

The following steps are for using the Airflow service provided with each Cloudera Data Engineering Virtual Cluster. For information about using your own Airflow deployment, see Using Cloudera Data Engineering with an external Apache Airflow deployment.

Before you begin

To determine the Cloudera Data Warehouse hostname to use for the connection, perform the following steps:

1. In the Cloudera Cloudera Management Console, click the Data Warehouse tile and click Overview.
2. In the Virtual Warehouses column, locate the Hive or Impala warehouse you want to connect to.

3. Click ⋮ next to the selected Warehouse, and then click Copy JDBC URL.
4. Paste the URL into a text editor, and make note of the hostname and the httpPath.

   For example,

   ```
   jdbc:hive2://hs2-aws-2-hive.env-k5ip0r.dw.ylcu-atmi.cloudera.site/defaul
   t;transportMode=http;httpPath=cliservice;ssl=true;retries=3;
   ```

   In this JDBC URL, the hostname is `hs2-aws-2-hive.env-k5ip0r.dw.ylcu-atmi.cloudera.site` and the httpPath is `cliservice`.

To determine the Cloudera Data Hub hostname to use for the connection, perform the following steps:

1. In the Cloudera Management Console, navigate to a suitable Cloudera Data Hub instance.
2. Click the Endpoints tab.
3. Select and copy the JDBC URL for the Warehouse.
4. Paste the URL into a text editor, and make note of the hostname and the httpPath parameters. The httpPath parameter is different for each Cloudera Data Hub.

   For example,

   ```
   jdbc:hive2://man-az-1-master0.man-azur.xcu2-8y8x.dev.cldr.work/;ssl=true
   ;transportMode=http;httpPath=man-az-1/cdp-proxy-api/hive
   ```

   In this JDBC URL, the hostname is `man-az-1-master0.man-azur.xcu2-8y8x.dev.cldr.work` and the httpPath is `man-az-1/cdp-proxy-api/hive`.

Procedure

To create a connection to an existing Cloudera Data Warehouse virtual warehouse using the embedded Airflow UI, perform the following steps:

1. In the Cloudera console, click the Data Engineering tile. The Cloudera Data Engineering Home page displays.
2. Click Administration in the left navigation menu and select the service containing the Virtual Cluster that you are using.
3. In the Virtual Clusters column, click Cluster Details for the Virtual Cluster.
4. Click AIRFLOW UI.
5. From the Airflow UI, click the Connection link from the Admin menu.

   Important: The connection credentials that you specify here can be used by any DAG created by any user in the same Virtual Cluster. Cloudera recommends you to use a Cloudera machine user for the connection to Cloudera Data Warehouse and limiting access to the Cloudera Data Engineering Virtual Cluster to trusted users using Virtual Cluster ACLs.

6. Click the plus sign to add a new record and fill the following fields:
   - Conn Id: Create a unique connection identifier. For example, cdw-hive-sql.
   - Conn Type: Select Impala, regardless of the Data Warehouse type whether it is Impala or Hive.
   - Host: Enter the hostname copied from the JDBC connection URL. Do not enter the full JDBC URL.
   - Schema: Enter the schema to be used. The default value is default.
   - Login/Password: Both Impala and Hive support JWT token based authentication if configured accordingly. If JWT token is used, make sure that you have left the Username and Password fields blank. If not, enter your workload username and password.
   - Port: 443.
   - Extra: Extra arguments must contain the following options:

     Note: The httpPath parameter must be the same that was copied in the prerequisites steps. From the preceding examples in the Prerequisites steps, for Cloudera Data Warehouse, the httpPath parameter is cliserviceand for Cloudera Data Hub, the httpPath parameter is man-az-1/cdp-proxy-api/hive.

     - If JWT token based authentication is used, then enter the following arguement:

```
{"auth_mechanism": "JWT", "use_ssl": "True",
 "use_http_transport": "True", "http_path":
"<***HTTPPATH_COPIED_FROM_THE_PREQUISITES_STEP***>", "jwt": "JWT_TOKE
N"}
```

     - If the workload username and password are used, then enter the following arguement:

```
{"auth_mechanism": "LDAP", "use_ssl": "True"
, "use_http_transport": "True", "http_path":
 "<***HTTPPATH_COPIED_FROM_THE_PREQUISITES_STEP***>"}
```

7. Click Save.

(Deprecated) Creating a connection to Cloudera Data Warehouse for Cloudera Data Warehouse Operator

Learn how to create an Airflow connection to an existing Cloudera Data Warehouse before running the workloads using the Cloudera Data Warehouse Operator.

About this task

Important: Starting from Cloudera Data Engineering version 1.23.0, Cloudera Data Warehouse Operator is deprecated. Cloudera Data Warehouse Operator supports connections to Apache Hive virtual warehouses only. It does not support connections to Impala virtual warehouses. However, SQL Operator supports connections to both Impala and Hive virtual warehouses. Hence, Cloudera recommends you to use SQL Operator instead of Cloudera Data Warehouse Operator. For more information about creating a connection to Cloudera Data Warehouse using SQL Operator, see Creating a connection to Cloudera Data Warehouse or Cloudera Data Hub instance for SQL Operator.

The following steps are for using the Airflow service provided with each Cloudera Data Engineering virtual cluster. For information about using your own Airflow deployment, see Using Cloudera Data Engineering with an external Apache Airflow deployment.

Before you begin

To determine the Cloudera Data Warehouse hostname to use for the connection, perform the following steps:

1. In the Cloudera Management Console, click the Data Warehouse tile and click Overview.
2. In the Virtual Warehouses column, locate the Hive or Impala warehouse you want to connect to.
3. Click ⋮ next to the selected Warehouse, and then click Copy JDBC URL.

4. Paste the URL into a text editor, and make note of the hostname.

   For example,

   ```
   jdbc:hive2://hs2-aws-2-hive.env-k5ip0r.dw.ylcu-atmi.cloudera.site/defaul
   t;transportMode=http;httpPath=cliservice;ssl=true;retries=3;
   ```

   In this JDBC URL, the hostname is `hs2-aws-2-hive.env-k5ip0r.dw.ylcu-atmi.cloudera.site`.

Procedure

To create a connection to an existing Cloudera Data Warehouse virtual warehouse using the embedded Airflow UI, perform the following steps:

1. In the Cloudera console, click the Data Engineering tile. The Cloudera Data Engineering Home page displays.
2. Click Administration in the left navigation menu and select the service containing the virtual cluster that you are using.
3. In the Virtual Clusters column, click Cluster Details for the virtual cluster.
4. Click AIRFLOW UI.
5. From the Airflow UI, click the Connection link from the Admin menu.

   Important: The connection credentials that you specify here can be used by any DAG created by any user in the same Virtual Cluster. Cloudera recommends you to use a Cloudera machine user for the connection to Cloudera Data Warehouse and limiting access to the Cloudera Data Engineering virtual cluster to trusted users using virtual cluster ACLs.
6. Click the plus sign to add a new record and fill the following fields:

   - Conn Id: Create a unique connection identifier. For example, cdw-hive-demo.
   - Conn Type: Select Hive Client Wrapper.
   - Host: Enter the hostname copied from the JDBC connection URL. Do not enter the full JDBC URL.
   - Schema: Enter the schema to be used. The default value is default.
   - Login/Password: Enter your workload username and password.
7. Click Save.

Creating a connection to run jobs on other Cloudera Data Engineering Virtual Clusters

Learn how to create a Cloudera Data Engineering type connection to run jobs on other Cloudera Data Engineering Virtual Clusters.

About this task

The following steps are for using the Airflow service provided with each Cloudera Data Engineering Virtual Cluster. For information about using your own Airflow deployment, see Using Cloudera Data Engineering with an external Apache Airflow deployment.

To create a connection to an existing Cloudera Data Engineering Virtual Cluster using the embedded Airflow UI, perform the following steps:

Procedure

1. In the Cloudera console, click the Data Engineering tile. The Cloudera Data Engineering Home page displays.
2. Click Administration in the left navigation menu and select the service containing the Virtual Cluster that you are using.
3. In the Virtual Clusters column, click Cluster Details for the Virtual Cluster.
4. Click AIRFLOW UI.

5. From the Airflow UI, click the Connection link from the Admin menu.
6. Click the plus sign to add a new record and fill the following fields:

    - Conn Id: Create a unique connection identifier. For example, cde_runtime_api.
    - Conn Type: Select Cloudera Data Engineering.
    - Virtual Cluster API endpoint: Enter the target Virtual Clusters Jobs API URL.
    - Cloudera Access Key: Enter the Cloudera access key of the account for running jobs on the Cloudera Data Engineering Virtual Cluster.
    - Cloudera Private Key: Enter the Cloudera private key associated to the Cloudera Access Key that you have entered.
    - Extra: Extra arguments must in JSON format. Available extra parameters are as follows:

        - Proxy: Optional. Translates to https_proxy/HTTPS_PROXY environment variables. The default value is None.
        - Region: Optional. Cloudera Control Plane region ("us-west-1", "eu-1" or "ap-1") is inferred automatically, if not specified.

        For more information about the available extra arguments, see the GitHub page.
7. Click Save.

Executing SQL queries on Cloudera Data Warehouse or Cloudera Data Hub instance using Apache Airflow in Cloudera Data Engineering

About this task

The following steps are for using the Airflow service provided with each Cloudera Data Engineering virtual cluster. For information about using your own Airflow deployment, see Using Cloudera Data Engineering with an external Apache Airflow deployment.

Before you begin

To run this job, an existing Airflow connection is required. For more information about how to create an airflow connection, see the following topics:

- For SQL Operators, see Creating a connection to Cloudera Data Warehouse or Cloudera Data Hub instance for SQL Operator.
- For Cloudera Data Warehouse Operators, see Creating a connection to Cloudera Data Warehouse for Cloudera Data Warehouse Operator.

Procedure

Create an Airflow DAG file in Python. Import the required operators and define the tasks and dependencies.

- The following example DAG file uses the connection named “impala-test”, and executes a “`SHOW TABLES`” query on the Impala Virtual Warehouse using the SQLExecuteQueryOperator:

```
import logging
from airflow import DAG
from datetime import datetime, timedelta
from airflow.providers.common.sql.operators.sql import SQLExecuteQueryOp
erator

with DAG(
dag_id="imp_dag",
start_date=datetime(2024, 2, 9),
schedule_interval=timedelta(days=1),
catchup=False,
) as dag:
  execute_query = SQLExecuteQueryOperator(
    task_id="execute_query",
    conn_id="impala-test",
    sql=f"SHOW TABLES",
    split_statements=True,
    return_last=False,
)
  execute_query
```

For more advanced use cases, see Airflow documentation about the SQL Operators.

- The following example DAG file uses the connection named “cdw-hive”, and executes a “`SHOW TABLES`” query on the Impala Virtual Warehouse using the CdwExecuteQueryOperator:

```
from airflow import DAG
from cloudera.airflow.providers.operators.cdw import CdwExecuteQueryOper
ator
from pendulum import datetime

default_args = {
    'owner': 'dag_owner',
    'depends_on_past': False,
    'start_date':datetime(2024, 2, 9)
}

example_dag = DAG(
    'example-cdwoperator',
    default_args=default_args,
    schedule_interval=None,
    catchup=False,
    is_paused_upon_creation=False
)

cdw_query = """
USE default;
SHOW TABLES;
"""

cdw_step = CdwExecuteQueryOperator(
    task_id='cdw-test',
    dag=example_dag,
    cli_conn_id='cdw-hive',
    hql=cdw_query,
    # The following values `schema`, `query_isolation`
    # are the default values, just presented here for the example.
    schema='default',
    query_isolation=True
)

cdw_step
```

For more information about the CdwExecuteQueryOperator, see the GitHub page.

Running Airflow jobs on other Cloudera Data Engineering Virtual Clusters

You can learn about how to run Airflow jobs on other Cloudera Data Engineering Virtual Clusters.

About this task

The following steps are for using the Apache Airflow service provided with each Cloudera Data Engineering virtual cluster. For more information on using your own Airflow deployment, see Using Cloudera Data Engineering with an external Apache Airflow deployment.

Before you begin

To run this job, an existing Airflow connection is required. For more information about how to create an Airflow connection, see Creating a connection to run jobs on other Cloudera Data Engineering Virtual Clusters.

Procedure

Create an Airflow DAG file in Python. Import the required operators and define the tasks and dependencies.

The following example DAG starts a job called “example-scala-pi” on the Cloudera Data Engineering Virtual Cluster defined by the connection named “cde_runtime_api”. The connection “cde_runtime_api” is also the default connection to the same Virtual Cluster the job is launched on. For more information about the operator and its capabilities, see the GitHub page.

```
from datetime import timedelta
from airflow import DAG
from cloudera.airflow.providers.operators.cde import CdeRunJobOperator
import pendulum

default_args = {
    'retry_delay': timedelta(seconds=5),
    'depends_on_past': False,
    'start_date': pendulum.datetime(2021, 1, 1, tz="UTC")
}

example_dag = DAG(
    'example-cdeoperator',
    default_args=default_args,
    schedule_interval='@once',
    catchup=False,
    is_paused_upon_creation=False
)

ingest_step1 = CdeRunJobOperator(
    connection_id='cde_runtime_api',
    task_id='ingest',
    retries=3,
    dag=example_dag,
    job_name='example-scala-pi'
)

prep_step2 = CdeRunJobOperator(
    task_id='data_prep',
    dag=example_dag,
    job_name='example-scala-pi'
)
ingest_step1 >> prep_step2
```

Using Cloudera Data Engineering with an external Apache Airflow deployment

Before you begin

The Cloudera provider for Apache Airflow, available at the Cloudera GitHub repository, provides two Airflow operators for running Cloudera Data Engineering and Cloudera Data Warehouse jobs. You can install the provider on your existing Apache Airflow deployment to integrate.

Important: Cloudera Data Engineering on Cloudera on premises currently supports only the Cloudera Data Engineering job run operator.

- The Cloudera provider for Apache Airflow is for use with existing Airflow deployments. If you want to use the embedded Airflow service provided by Cloudera Data Engineering, see Apache Airflow in Cloudera Data Engineering.
- The provider requires Python 3.6 or higher.
- The provider requires the Python cryptography package version 3.3.2 or higher to address CVE-2020-36242. If an older version is installed, the plugin automatically updates the cryptography library.

About this task

This component provides two Airflow operators to be integrated in your DAGs:

- CdeRunJobOperator, for running Cloudera Data Engineering jobs.
- CDWOperator, for accessing Cloudera Data Warehouse

Procedure

Install Cloudera Airflow provider on your Airflow servers

1. Run the following pip command on each Airflow server: `pip install cloudera-airflow-provider`

Create a connection using the Airflow UI

Before you can run a Cloudera Data Engineering job from your Airflow deployment, you must configure a connection using the Airflow UI.

2. From the Cloudera Data Engineering home page, go to Overview Virtual Clusters Cluster Details of the Virtual Cluster (VC) where you want the Cloudera Data Engineering job to run.

3. Click JOBS API URL to copy the URL.

4. Go to your Airflow web console (where you installed the Cloudera provider).

5. Go to Admin Connection .

6. Click + Add a new record.

7. Fill in connection details:

   Conn Id

   Create a unique connection identifier.

   Conn Type

   The type of the connection. From the drop-down, select

   - HTTP (if you are using Apache Airflow version 1)
   - HTTP or Cloudera Data engineering (if you are using Apache Airflow version 2)

   Host/Virtual API Endpoint

   URL of the host where you want the job to run. Paste here the JOBS API URL you copied in a previous step.

   Login/Cloudera Access Key

   Provide the Cloudera access key of the account for running jobs on the Cloudera Data Engineering VC.

   Password/Cloudera Private Key

   Provide the Cloudera private key of the account for running jobs on the Cloudera Data Engineering VC.
8. Click Save.
9. In the Cloudera Data Engineering Home page, click Jobs in the left navigation menu, and then click Create Job.
10. Fill in the Job Details:

   Job Type

   Select the option matching your use case.

   Name

   Specify a name for the job.

   DAG File

   Provide a DAG file.

   Use the `CdeRunJobOperator` to specify a Cloudera Data Engineering job to run. The job definition in the DAG file must contain:

   connection_id

   The Conn Id you specified on the Airflow UI when creating the connection.

   task_id

   The ID that identifies the job within the DAG.

   dag

   The variable containing the dag object

   job_name

   The name of the Cloudera Data Engineering job to run. This job must exist in the Cloudera Data Engineering virtual cluster you are connecting to.

   For example:

```
from cloudera.cdp.airflow.operators.cde_operator import CdeRunJobOperator
...
t1 = CdeRunJobOperator(
    connection_id='cde-vc01-dev',
    task_id='ingest',
    dag=example_dag,
    job_name='etl-ingest-job'
)
```

11. Click Create and Run to create the job and run it immediately, or click the dropdown button and select Create to create the job.

Supported Airflow operators and hooks

Apache Airflow in Cloudera Data Engineering supports most of the base operators and hooks. Additionally the following third party provider packages and custom operators are installed by default. Additional third party operators can be installed using the Custom Operators and Libraries feature, however their versions must comply with the release-specific constraints file.

For Cloudera Data Engineering 1.24.1

Cloudera Data Engineering on cloud version 1.24.1 supports the following components:

- Airflow 2.10.4
- Python 3.11.11

Airflow operators:

Cloudera Data Engineering on cloud version 1.24.1 supports the following Airflow operators:

- airflow.operators.bash
- airflow.operators.branch
- airflow.operators.datetime
- airflow.operators.email
- airflow.operators.empty
- airflow.operators.generic_transfer
- airflow.operators.latest_only
- airflow.operators.python
- airflow.operators.smooth
- airflow.operators.trigger_dagrun
- airflow.operators.weekday

Airflow hooks:

Cloudera Data Engineering on cloud version 1.24.1 supports the following Airflow hooks:

- airflow.hooks.dbapi
- airflow.hooks.filesystem
- airflow.hooks.package_index
- airflow.hooks.subprocess

Airflow provider packages:

Cloudera Data Engineering on cloud version 1.24.1 supports the following Airflow provider packages:

- apache-airflow-providers-amazon 9.1.0
- apache-airflow-providers-apache-hive 8.2.1
- apache-airflow-providers-apache-impala 1.5.2
- apache-airflow-providers-celery 3.8.5
- apache-airflow-providers-cncf-kubernetes 10.0.1
- apache-airflow-providers-common-compat 1.2.2
- apache-airflow-providers-common-io 1.4.2
- apache-airflow-providers-common-sql 1.20.0
- apache-airflow-providers-docker 3.14.1
- apache-airflow-providers-elasticsearch 5.5.3

- apache-airflow-providers-fab 1.5.1
- apache-airflow-providers-ftp 3.11.1
- apache-airflow-providers-google 11.0.0
- apache-airflow-providers-grpc 3.6.0
- apache-airflow-providers-hashicorp 3.8.0
- apache-airflow-providers-http 4.13.3
- apache-airflow-providers-imap 3.7.0
- apache-airflow-providers-microsoft-azure 11.1.0
- apache-airflow-providers-mysql 5.7.4
- apache-airflow-providers-odbc 4.8.1
- apache-airflow-providers-openlineage 1.14.0
- apache-airflow-providers-postgres 5.14.0
- apache-airflow-providers-redis 3.8.0
- apache-airflow-providers-sendgrid 3.6.0
- apache-airflow-providers-sftp 4.11.1
- apache-airflow-providers-slack 8.9.2
- apache-airflow-providers-smtp 1.8.1
- apache-airflow-providers-snowflake 5.8.1
- apache-airflow-providers-sqlite 3.9.1
- apache-airflow-providers-ssh 3.14.0
- cloudera-airflow-provider 2.1.4
- cloudera_cdp_airflow_internal 1.0.0
- google-cloud-orchestration-airflow 1.15.1

For Cloudera Data Engineering 1.23.0

Cloudera Data Engineering on cloud version 1.23.0 supports the following components:

- Airflow 2.9.3
- Python 3.11.9

Airflow operators:

Cloudera Data Engineering on cloud version 1.23.0 supports the following Airflow operators:

- airflow.operators.bash
- airflow.operators.branch
- airflow.operators.datetime
- airflow.operators.email
- airflow.operators.empty
- airflow.operators.generic_transfer
- airflow.operators.latest_only
- airflow.operators.python
- airflow.operators.smooth
- airflow.operators.trigger_dagrun
- airflow.operators.weekday

Airflow hooks:

Cloudera Data Engineering on cloud version 1.23.0 supports the following Airflow hooks:

- airflow.hooks.dbapi
- airflow.hooks.filesystem
- airflow.hooks.package_index
- airflow.hooks.subprocess

Important: CDWHook and CdwExecuteQueryOperator are deprecated. Instead, use SQLExecuteQueryOperator.

Airflow provider packages:

Cloudera Data Engineering on cloud version 1.23.0 supports the following Airflow provider packages:

- apache-airflow-providers-amazon 8.25.0
- apache-airflow-providers-apache-hive 8.1.2
- apache-airflow-providers-apache-impala 1.4.1
- apache-airflow-providers-celery 3.7.2
- apache-airflow-providers-cncf-kubernetes 8.3.3
- apache-airflow-providers-common-io 1.3.2
- apache-airflow-providers-common-sql 1.14.2
- apache-airflow-providers-docker 3.12.2
- apache-airflow-providers-elasticsearch 5.4.1
- apache-airflow-providers-fab 1.2.1
- apache-airflow-providers-ftp 3.10.0
- apache-airflow-providers-google 10.21.0
- apache-airflow-providers-grpc 3.5.2
- apache-airflow-providers-hashicorp 3.7.1
- apache-airflow-providers-http 4.12.0
- apache-airflow-providers-imap 3.6.1
- apache-airflow-providers-microsoft-azure 10.3.0
- apache-airflow-providers-mysql 5.6.2
- apache-airflow-providers-odbc 4.6.2
- apache-airflow-providers-openlineage 1.9.1
- apache-airflow-providers-postgres 5.11.2
- apache-airflow-providers-redis 3.7.1
- apache-airflow-providers-sendgrid 3.5.1
- apache-airflow-providers-sftp 4.10.2
- apache-airflow-providers-slack 8.7.1
- apache-airflow-providers-smtp 1.7.1
- apache-airflow-providers-snowflake 5.6.0
- apache-airflow-providers-sqlite 3.8.1
- apache-airflow-providers-ssh 3.11.2
- cloudera-airflow-provider 2.1.4
- cloudera-cdp-airflow-internal 1.0.0
- google-cloud-orchestration-airflow 1.13.0

For Cloudera Data Engineering 1.22.0

Cloudera Data Engineering on cloud version 1.22.0 supports the following components:

- Airflow 2.7.3
- Python 3.8.16

Airflow operators:

Cloudera Data Engineering on cloud version 1.22.0 supports the following Airflow operators:

- airflow.operators.bash
- airflow.operators.branch
- airflow.operators.datetime
- airflow.operators.email

- airflow.operators.empty
- airflow.operators.generic_transfer
- airflow.operators.latest_only
- airflow.operators.python
- airflow.operators.smooth
- airflow.operators.trigger_dagrun
- airflow.operators.weekday

Airflow hooks:

Cloudera Data Engineering on cloud version 1.22.0 supports the following Airflow hooks:

- airflow.hooks.dbapi
- airflow.hooks.filesystem
- airflow.hooks.subprocess

Airflow provider packages:

Cloudera Data Engineering on cloud version 1.22.0 supports the following Airflow provider packages:

- apache-airflow-providers-amazon 8.10.0
- apache-airflow-providers-apache-hive 6.2.0
- apache-airflow-providers-apache-impala 1.2.0
- apache-airflow-providers-celery 3.4.1
- apache-airflow-providers-cncf-kubernetes 7.8.0
- apache-airflow-providers-common-sql 1.8.0
- apache-airflow-providers-daskexecutor 1.1.0
- apache-airflow-providers-docker 3.8.0
- apache-airflow-providers-elasticsearch 5.1.0
- apache-airflow-providers-ftp 3.6.0
- apache-airflow-providers-google 10.11.0
- apache-airflow-providers-grpc 3.3.0
- apache-airflow-providers-hashicorp 3.5.0
- apache-airflow-providers-http 4.6.0
- apache-airflow-providers-imap 3.4.0
- apache-airflow-providers-microsoft-azure 8.1.0
- apache-airflow-providers-mysql 5.4.0
- apache-airflow-providers-odbc 4.1.0
- apache-airflow-providers-openlineage 1.2.0
- apache-airflow-providers-postgres 5.7.1
- apache-airflow-providers-redis 3.4.0
- apache-airflow-providers-sendgrid 3.3.0
- apache-airflow-providers-sftp 4.7.0
- apache-airflow-providers-slack 8.3.0
- apache-airflow-providers-snowflake 5.1.0
- apache-airflow-providers-sqlite 3.5.0
- apache-airflow-providers-ssh 3.8.1
- cloudera-airflow-provider 2.1.2
- google-cloud-orchestration-airflow 1.9.2

Related Information

Apache Airflow in Cloudera Data Engineering

Using custom operators and libraries for Apache Airflow

You can install and use custom python packages for Airflow with Cloudera Data Engineering. Cloudera provides access to the open source packages that you can use for your Airflow jobs using the Cloudera Data Engineering user interface (UI).

Note: The Cloudera Data Engineering API for Airflow Operators and Libraries currently expects Base64-encoded certificates, but the Cloudera Data Engineering UI sends PEM certificates without conversion. As a workaround, go to Cloudera Data Engineering UI Administration Virtual Cluster Details of the selected virtual cluster Configuration tab Airflow Libraries and Operators Configure Repositories SSL Certificate and provide the certificate in Base64 format.

Adding custom operators and libraries

You can add custom python packages for Airflow with Cloudera Data Engineering. Cloudera provides access to the open source packages that you can use for your Airflow jobs using the UI.

About this task

While you can install the operator, if additional runtime dependencies are required such as additional setup with binaries on the path, and environment configuration like Kerberos and Cloud credentials, and so on, then the operator will not work. To use an Airflow third-party operator for your custom library and operator package, you must configure the Airflow connection in the Airflow UI.

Note: You can use all officially supported third-party providers for Apache Airflow. Any provider listed in Apache Airflow's official documentation is fully compatible, excluding potential cases of dependency conflicts with your custom package.

Note: The installed operators and libraries will be available to all jobs defined in the chosen Cloudera Data Engineering Virtual Cluster.

Procedure

1. In the Cloudera console, click the Data Engineering tile. The Cloudera Data Engineering **Home** page displays.
2. Click Administration in the left navigation menu. The **Administration** page displays.
3. Locate the Virtual Cluster that you want to edit, and click Cluster Details.
4. Go to the **Airflow** tab. The **Libraries and Operators** page displays.
5. Under the **Configure Repositories** section, enter the following fields to configure the Python Package Index (PyPi) repositories used to source your custom libraries and operators:
   a) PyPI Repository URL - Enter the Python Package Index (PyPi) URL.
   b) SSL Certificate - Enter the PEM-encoded CA certificate.
   c) Enter Authorization Credentials if you are configuring a Private or Protected PyPi Repository that requires authorization for access.
      - Username
      - Password
6. Click Validate Configurations.

7. Under the **Build** section, upload a requirements.txt file that contains a list of all library and operator packages that you want to enable. Once uploaded, the system will automatically build and install your packages.

   You can specify any Python package which is compatible with the Airflow python constraints, which you can find here:

   https://raw.githubusercontent.com/apache/airflow/constraints-${AIRFLOW_VERSION}/constraints-${PYTHON_VERSION}.txt

   Tip: The Airflow and Python versions (${AIRFLOW_VERSION} and ${PYTHON_VERSION} respectively) depend on your Cloudera Data Engineering version.

8. Click Activate. The activation restarts the Airflow server. This may take a few minutes. Once activation is complete, you will see the Installed Packages listed.

   You can now create and run an Airflow job using the custom library and operators that you have acitvated.

Related Information

Third-party Airflow Operators

Requirement files

Updating custom operators and libraries

After creating your custom operator and libraries, you can update and add more configurations to the installed package.

Procedure

1. In the Cloudera console, click the Data Engineering tile. The Cloudera Data Engineering **Home** page displays.
2. Click Administration in the left navigation menu. The **Administration** page displays.
3. Locate the Virtual Cluster that you want to edit, and click Cluster Details.
4. Go to the **Airflow** tab. The **Libraries and Operators** page displays.
5. Click the Installed Package, and click Actions Update
6. Under **Configure Repositories**, update configurations or click Add Configurations, then click Validate Configurations to confirm your changes.
7. Under **Build**, you can upload a new requirements.txt file if needed.
8. Click Activate to confirm your updates.

Deleting custom operators and libraries

If you no longer need your custom operators and libraries, you can delete it. To delete an installed package containing custom operators and libraries, complete the following steps.

Procedure

1. In the Cloudera console, click the Data Engineering tile. The Cloudera Data Engineering **Home** page displays.
2. Click Administration in the left navigation menu. The **Administration** page displays.
3. Locate the Virtual Cluster that you want to edit, and click Cluster Details.
4. Go to the **Airflow** tab. The **Libraries and Operators** page displays.
5. Under Installed Package , click Actions Delete Deleting may take a few minutes.

Troubleshooting custom operators and libraries

Learn the different ways that you can troubleshoot the setup of your Airflow Python environment in Cloudera Data Engineering.

Viewing logs for custom operators and libraries

After adding custom operators and libraries, you will see them listed under Instsalled Packages. You can view the logs related to the build and activation process of the installed packages. To view logs, complete the following steps:

Procedure

1. In the Cloudera console, click the Data Engineering tile. The Cloudera Data Engineering **Home** page displays.
2. Click Administration in the left navigation menu. The **Administration** page displays.
3. Locate the Virtual Cluster that you want to edit, and click Cluster Details. The **Libraries and Operators** page displays.
4. Under Installed Package, click Actions View Logs . The logs information is displayed in a new window.