

Orchestrating workflows and pipelines with Apache Airflow in Cloudera Data Engineering

Date published: 2020-07-30

Date modified: 2024-02-26

CLOUDERA

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Automating data pipelines using Apache Airflow in Cloudera Data Engineering.....	4
Creating an Airflow DAG using the Pipeline UI.....	8
Managing an Airflow Pipeline using the CDE CLI.....	8
Creating a pipeline using the CDE CLI.....	8
Creating a basic Airflow pipeline using CDE CLI.....	9
Creating a pipeline with additional Airflow configurations using CDE CLI.....	9
Creating an Airflow pipeline with custom files using CDE CLI [technical preview].....	11
Updating a pipeline using the CDE CLI.....	12
Updating a DAG file using the CDE CLI.....	12
Updating the Airflow job configurations using the CDE CLI.....	13
Updating the Airflow file mounts using the CDE CLI [technical preview].....	13
Deleting an Airflow pipeline using the CDE CLI.....	13
Using CDE with an external Apache Airflow deployment.....	14
Supporting Airflow operators and hooks.....	16
Using custom operators and libraries for Apache Airflow.....	16
Adding custom operators and libraries.....	16
Updating custom operators and libraries.....	17
Deleting custom operators and libraries.....	17
Troubleshooting custom operators and libraries using API.....	18
Viewing logs for custom operators and libraries.....	18

Automating data pipelines using Apache Airflow in Cloudera Data Engineering

Cloudera Data Engineering (CDE) enables you to automate a workflow or data pipeline using Apache Airflow Python DAG files. Each CDE virtual cluster includes an embedded instance of Apache Airflow. You can also use CDE with your own Airflow deployment. CDE currently supports two Airflow operators; one to run a CDE job and one to access Cloudera Data Warehouse (CDW).

Before you begin



Important:

- Cloudera supports certain Airflow core operators and hooks but does not provide support for any custom extension through provider packages. See [Supporting Airflow operators and hooks](#) linked below. Cloudera Support may require you to remove any installed provider packages during troubleshooting. See [Provider packages](#) linked below for more information.
- Direct access to the native Airflow API is unsupported because bypassing the CDE API prevents job tracking, centralized monitoring, and metadata tracking.
- To complete these steps, you must have a running CDE virtual cluster and have access to a CDW virtual warehouse. CDE currently supports CDW operations for ETL workloads in Apache Hive virtual warehouses.

To determine the CDW hostname to use for the connection:

1. Navigate to the Cloudera Data Warehouse Overview page by clicking the Data Warehouse tile in the Cloudera Data Platform (CDP) management console.
2. In the Virtual Warehouses column, find the warehouse you want to connect to.
3. Click the three-dot menu for 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, the hostname portion of the following JDBC URL is emphasized in italics:



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

About this task

The following instructions are for using the Airflow service provided with each CDE virtual cluster. For instructions on using your own Airflow deployment, see [Using the Cloudera provider for Apache Airflow](#).

Procedure

1. Create a connection to an existing CDW virtual warehouse using the embedded Airflow UI:
 - a) In the Cloudera Data Platform (CDP) console, click the Data Engineering tile. The CDE Home page displays.
 - b) Click Administration in the left navigation menu, select the service containing the virtual cluster you are using,


 and then in the Virtual Clusters column, click **Cluster Details** for the virtual cluster.
 - c) Click AIRFLOW UI.
 - d) From the Airflow UI, click the Connection link from the Admin menu.
 - e)  **Warning:** The connection credentials you specify here can be used by any DAG created by any user in the same Virtual Cluster. Cloudera recommends using a CDP machine user for the connection to CDW and limiting access to the CDE virtual cluster to trusted users using virtual cluster ACLs.

Click the plus sign to add a new record, and then fill in the fields:

To create a connection to an existing CDW virtual warehouse:

Conn Id

Create a unique connection identifier, such as cdw-hive-demo.

Conn Type

Select Impala or HiveServer2, depending on the use-case.

Host

Enter the hostname from the JDBC connection URL. Do not enter the full JDBC URL.

Schema

default

Login/Password

Impala supports JWT token based authentication. If JWT token is used Login and Password should be left empty. Enter your [workload](#) username and password.

Extra

Extra arguments should contain the following options:

If JWT token based authentication is used:

```
{"auth_mechanism": "JWT", "use_ssl": "True", "use_http_transport": "True", "http_path": "cliservice", "jwt": "JWT_TOKEN"}
```

If workload username and password is used:

```
{"auth_mechanism": "LDAP", "use_ssl": "True", "use_http_transport": "True", "http_path": "cliservice"}
```

To create a connection to an existing virtual cluster in a different environment:

Conn Id

Create a unique connection identifier.

Conn Type

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

Host/Virtual API Endpoint

The JOBS API URL of the host where you want the job to run.

Login/CDP Access Key

Provide the CDP access key of the account for running jobs on the CDE virtual cluster.

Password/CDP Private Key

Provide the CDP private key of the account for running jobs on the CDE virtual cluster.

- f) Click Save.
2. Create an Airflow DAG file in Python. Import the CDE and CDW operators and define the tasks and dependencies.
Here is a complete DAG file:

```
from dateutil import parser
from datetime import datetime, timedelta
from datetime import timezone
from airflow import DAG
from cloudera.airflow.providers.operators.cde import CdeRunJobOperator
from airflow.providers.common.sql.operators.sql import
    SQLExecuteQueryOperator

default_args = {
    'owner': 'psherman',
    'retry_delay': timedelta(seconds=5),
    'depends_on_past': False,
    'start_date': datetime(2024, 2, 10, tz="UTC"),
}

example_dag = DAG(
    'airflow-pipeline-demo',
    default_args=default_args,
    schedule_interval='@daily',
    catchup=False,
    is_paused_upon_creation=False
)

ingest_step1 = CdeRunJobOperator(
    connection_id='cde-vc01-dev',
    task_id='ingest',
    retries=3,
    dag=example_dag,
    job_name='etl-ingest-job'
)

prep_step2 = CdeRunJobOperator(
    task_id='data_prep',
    job_name='insurance-claims-job'
)

cdw_query = """
show databases;
"""

dw_step3 = SQLExecuteQueryOperator(
    task_id="dataset-etl-cdw",
    conn_id="cde-vw-demo",
    sql=cdw_query
)

ingest_step1 >> prep_step2 >> dw_step3
```

Here are some examples of things you can define in the DAG file:

CDE job run operator

Use `CdeRunJobOperator` to specify a CDE job to run. This job must already exist in the virtual cluster specified by the `connection_id`. If no `connection_id` is specified, CDE looks for the job in the virtual cluster where the Airflow job runs.

```
from cloudera.cdp.airflow.operators.cde_operator import CdeRunJobOperator
```

```
...
ingest_step1 = CdeRunJobOperator(
    connection_id='cde-vc01-dev',
    task_id='ingest',
    retries=3,
    dag=example_dag,
    job_name='etl-ingest-job'
)
```

Query definition

You can define a query string or template within the DAG file for later reference as follows:

```
cdw_query = """
show databases;
"""
```

SQLExecuteQueryOperator

Use `SQLExecuteQueryOperator` to execute a query against the CDW virtual warehouse. Specify the connection ID you configured in Airflow, and the query you want to run.

```
from airflow.providers.common.sql.operators.sql import SQLExecuteQueryOperator
...
dw_step3 = SQLExecuteQueryOperator(
    task_id='dataset-etl-cdw',
    conn_id='cde-vw-demo',
    sql=cdw_query
)
```

Task dependencies

After you have defined the tasks, specify the dependencies as follows:

```
ingest_step1 >> prep_step2 >> dw_step3
```

For more information on task dependencies, see [Task Dependencies](#) in the Apache Airflow documentation.

For a tutorial on creating Apache Airflow DAG files, see the [Apache Airflow documentation](#).

3. Create a CDE job.

- a) In the Cloudera Data Platform (CDP) console, click the Data Engineering tile. The CDE Home page displays.
 - b) In the left navigation menu click Jobs. The Jobs page is displayed.
 - c) Click Create Job. The Job Details page is displayed.
 - d) Select the Airflow job type.
 - e) Name: Provide a name for the job.
 - f) DAG File: Use an existing file or add a DAG file to an existing resource or create a resource and upload it.
 1. Select from Resource: Click Select from Resource to select a DAG file from an existing resource.
 2. Upload: Click Upload to upload a DAG file to an existing resource or to a new resource that you can create by selecting Create a resource from the Select a Resource dropdown list. Specify the resource name and upload the DAG file to it.
4. Click Create and Run to create the job and run it immediately, or click the dropdown button and select Create to create the job.

Related reference

[Supporting Airflow operators and hooks](#)

Related Information

[Provider packages](#)

[Enabling SSO to a Virtual Warehouse](#)
[Creating jobs in Cloudera Data Engineering](#)

Creating an Airflow DAG using the Pipeline UI

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

About this task

This feature is available starting in CDE 1.16 in new Virtual Cluster installations.



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.
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**.

Managing an Airflow Pipeline using the CDE CLI

Based on your business requirement, you can use Cloudera Data Engineering (CDE) 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 (CDE) using the CLI, you can create multi-step pipelines with a combination of available operators.

About this task

To create a basic pipeline in CDE, you must upload the Airflow (Directed Acyclic Graph) DAG to a CDE resource and create a CDE 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-l-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 (CDE) 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.

For historical reasons CDE 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 --config 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 CDE 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 CDE 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 CDE 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 CDE 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 CDE 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 my_file_resource
```

Example

The files can be reached in Airflow DAGs with the following pattern: `/app/mount/<resource_name> or resource_alias/<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-l-resource my_pipeline_resource --airflow-file-mount-l-resource my_file_resource
--airflow-file-mount-l-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-l-resource 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 --mount-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_pipeline_resource
```

Changing a file mount prefix

```
cde job update --name my_pipeline --airflow-file-mount-1-prefix my_new_pipeline_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_pipeline_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 CDE using the CLI.

Procedure

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

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

Using CDE with an external Apache Airflow deployment

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

Before you begin



Important: CDE on CDP Private Cloud currently supports only the CDE 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 CDE, see [Automating data pipelines with CDE and CDW using Apache Airflow](#).
- 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:

- CDEJobRunOperator, 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 CDE job from your Airflow deployment, you must configure a connection using the Airflow UI.

2. From the CDE home page, go to Overview Virtual Clusters Cluster Details of the Virtual Cluster (VC) where you want the CDE 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/CDP Access Key

Provide the CDP access key of the account for running jobs on the CDE VC.

Password/CDP Private Key

Provide the CDP private key of the account for running jobs on the CDE VC.

8. Click Save.**9. In the CDE 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 `CDEJobRunOperator` to specify a CDE 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 CDE job to run. This job must exist in the CDE virtual cluster you are connecting to.

For example:

```
from cloudera.cdp.airflow.operators.cde_operator import CDEJobRunOperator
...
tl = CDEJobRunOperator(
    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.

Supporting Airflow operators and hooks

Apache Airflow Python DAG files can be used to automate workflows or data pipelines in Cloudera Data Engineering (CDE). CDE currently supports a specified list of Airflow operators and hooks that can be used.

Airflow operators

CDE supports the following Airflow operators:

- `airflow.operators.bash`
- `airflow.operators.branch`
- `airflow.operators.datetime`
- `airflow.operators.dummy`
- `airflow.operators.email`
- `airflow.operators.generic_transfer`
- `airflow.operators.latest_only`
- `airflow.operators.python`
- `airflow.operators.sql`
- `airflow.operators.subdag`
- `airflow.operators.trigger_dagrun`

Airflow hooks

CDE supports the following Airflow hooks:

- `airflow.hooks.filesystem`
- `airflow.hooks.subprocess`

Related Tasks

[Automating data pipelines using 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 (CDE). Cloudera provides access to the open source packages that you can use for your Airflow jobs using the CDE user interface (UI).

Adding custom operators and libraries

You can add custom python packages for Airflow with Cloudera Data Engineering (CDE). 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.

Procedure

1. In the Cloudera Data Platform (CDP) console, click the Data Engineering tile. The CDE **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. Optional: SSL Certificate - Enter the PEM-encoded CA certificate.
 - c. Optional: Enter Authorization Credentials if you are configuring a Private or Protected PyPi Repository that requires authorization for access:
 1. Username
 2. 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.



Note: You can specify any Python package which is compatible with the Airflow python constraints. These constraints can be found here: [https://raw.githubusercontent.com/apache/airflow/constraints-\\${AIRFLOW_VERSION}/constraints-\\${PYTHON_VERSION}.txt](https://raw.githubusercontent.com/apache/airflow/constraints-${AIRFLOW_VERSION}/constraints-${PYTHON_VERSION}.txt) . The Airflow and Python versions depend on your CDE 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 activated.

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 Data Platform (CDP) console, click the Data Engineering tile. The CDE **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 Data Platform (CDP) console, click the Data Engineering tile. The CDE **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 using API

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

Viewing logs for custom operators and libraries

After adding custom operators and libraries, you will see them listed under Installed 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 Data Platform (CDP) console, click the Data Engineering tile. The CDE **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.