Hands-on Lab: Create a DAG for Apache Airflow with BashOperator

Estimated time needed: **40** minutes

Introduction

In this lab, you will create workflows using BashOperator in Airflow DAGs and simulate an ETL process using bash commands that are scheduled to run once a day.

Objectives

After completing this lab, you will be able to:

- Explore the Airflow Web UI
- Create a DAG with BashOperator
- Submit a DAG and run it through Web UI

Prerequisites

Please ensure that you have completed the reading on the Airflow DAG Operators before proceeding with this lab. It is highly recommended that you are familiar with `bash` commands to do this lab.

About Skills Network Cloud IDE

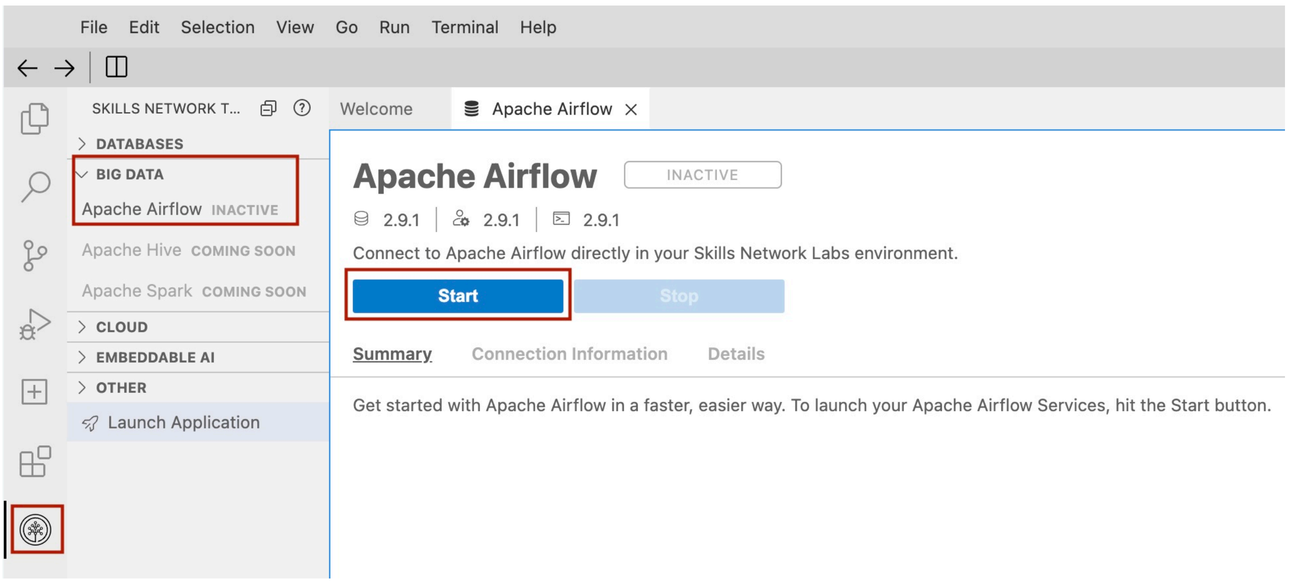
Skills Network Cloud IDE (based on Theia and Docker) provides an environment for hands-on labs for course and project-related labs. Theia is an open-source IDE (Integrated Development Environment) that can be run on a desktop or on the cloud. To complete this lab, you will be using the Cloud IDE based on Theia, running in a Docker container.

Important notice about this lab environment

Please be aware that sessions for this lab environment are not persistent. A new environment is created for you every time you connect to this lab. Any data you may have saved in an earlier session will get lost. To avoid losing your data, please plan to complete these labs in a single session.

Exercise 1: Start Apache Airflow

1. Click on **Skills Network Toolbox**.
2. From the **BIG DATA** section, click **Apache Airflow**.
3. Click **Start** to start the Apache Airflow.



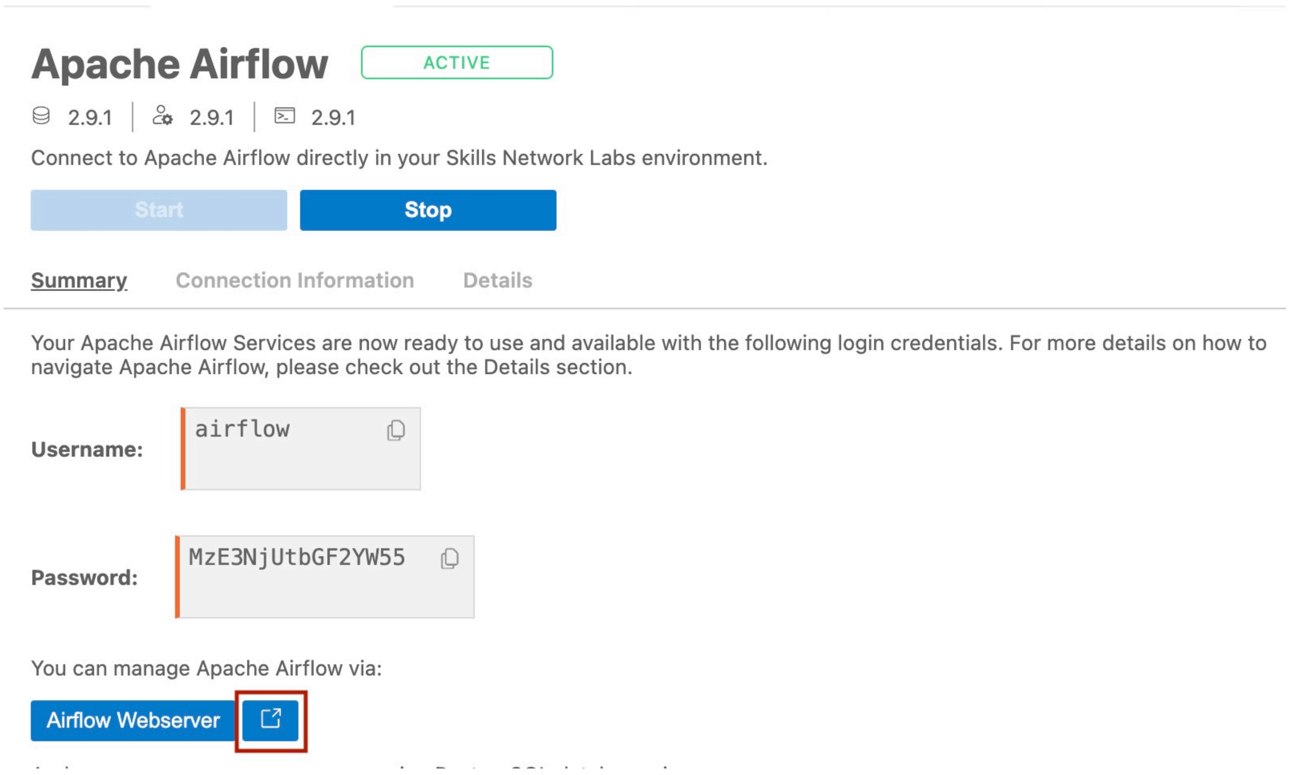
Connect to Apache Airflow directly in your Skills Network Labs environment.

Get started with Apache Airflow in a faster, easier way. To launch your Apache Airflow Services, hit the Start button.

Note: Please be patient, it will take a few minutes for Airflow to start.

Exercise 2: Open the Airflow Web UI

1. When Airflow starts successfully, you should see an output similar to the one below. Once **Apache Airflow** has started, click on the highlighted icon to open **Apache Airflow Web UI** in the new window.


Apache Airflow ACTIVE

2.9.1 | 2.9.1 | 2.9.1

Connect to Apache Airflow directly in your Skills Network Labs environment.

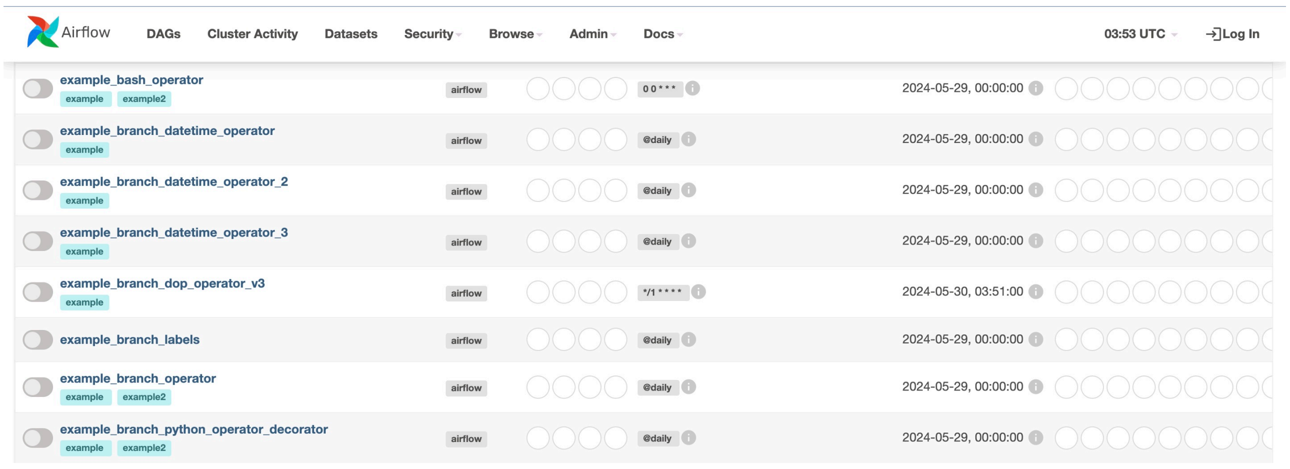
Your Apache Airflow Services are now ready to use and available with the following login credentials. For more details on how to navigate Apache Airflow, please check out the Details section.

Username: airflow

Password: MzE3NjUtbGF2YW55

You can manage Apache Airflow via:

You should land on a page that looks like this.



example_bash_operator (example, example2)	airflow	0 0 * * *	2024-05-29, 00:00:00
example_branch_datetime_operator (example)	airflow	@daily	2024-05-29, 00:00:00
example_branch_datetime_operator_2 (example)	airflow	@daily	2024-05-29, 00:00:00
example_branch_datetime_operator_3 (example)	airflow	@daily	2024-05-29, 00:00:00
example_branch_dop_operator_v3 (example)	airflow	*/1 * * * *	2024-05-30, 03:51:00
example_branch_labels	airflow	@daily	2024-05-29, 00:00:00
example_branch_operator (example, example2)	airflow	@daily	2024-05-29, 00:00:00
example_branch_python_operator_decorator (example, example2)	airflow	@daily	2024-05-29, 00:00:00

Exercise 3: Create a DAG

Let's create a DAG that runs daily, and extracts user information from */etc/passwd* file, transforms it, and loads it into a file.

This DAG will have two tasks `extract` that extracts fields from `/etc/passwd` file and `transform_and_load` that transforms and loads data into a file.

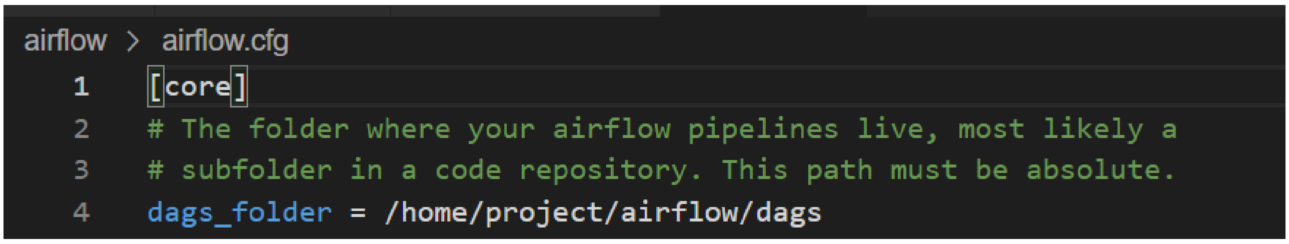
```
# import the libraries
from datetime import timedelta
# The DAG object; we'll need this to instantiate a DAG
from airflow.models import DAG
# Operators; you need this to write tasks!
from airflow.operators.bash_operator import BashOperator
# This makes scheduling easy
from airflow.utils.dates import days_ago
#defining DAG arguments
# You can override them on a per-task basis during operator initialization
default_args = {
    'owner': 'your_name_here',
    'start_date': days_ago(0),
    'email': ['your_email_here'],
    'retries': 1,
    'retry_delay': timedelta(minutes=5),
}
# defining the DAG
# define the DAG
dag = DAG(
    'my-first-dag',
    default_args=default_args,
    description='My first DAG',
    schedule_interval=timedelta(days=1),
)
# define the tasks
# define the first task
extract = BashOperator(
    task_id='extract',
    bash_command='cut -d":" -f1,3,6 /etc/passwd > /home/project/airflow/dags/extracted-data.txt',
    dag=dag,
)
# define the second task
transform_and_load = BashOperator(
    task_id='transform',
    bash_command='tr ":" "," < /home/project/airflow/dags/extracted-data.txt > /home/project/airflow/dags/transformed-data.csv',
    dag=dag,
)
# task pipeline
extract >> transform_and_load
```

1. Create a new file by choosing File->New File and naming it `my_first_dag.py`.
2. Then, copy the code above and paste it into `my_first_dag.py`.

Exercise 4: Submit a DAG

Submitting a DAG is as simple as copying the DAG Python file into the `dags` folder in the `AIRFLOW_HOME` directory.

Airflow searches for Python source files within the specified `DAGS_FOLDER`. The location of `DAGS_FOLDER` can be located in the airflow.cfg file, where it has been configured as `/home/project/airflow/dags`.



airflow > airflow.cfg

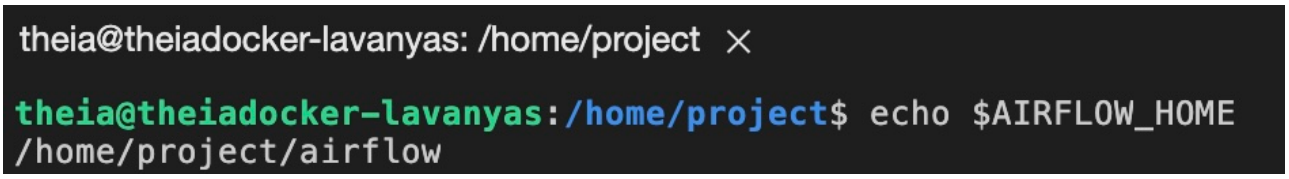
```
[core]
# The folder where your airflow pipelines live, most likely a
# subfolder in a code repository. This path must be absolute.
dags_folder = /home/project/airflow/dags
```

Airflow will load the Python source files from this designated location. It will process each file, execute its contents, and subsequently load any DAG objects present in the file.

Therefore, when submitting a `DAG`, it is essential to position it within this directory structure. Alternatively, the `AIRFLOW_HOME` directory, representing the structure `/home/project/airflow`, can also be utilized for DAG submission.

1. Open a terminal and run the command below to set the `AIRFLOW_HOME`.

   ```
   export AIRFLOW_HOME=/home/project/airflow
   echo $AIRFLOW_HOME
   ```



theia@theiadocker-lavanyas: /home/project ×

```
theia@theiadocker-lavanyas:/home/project$ echo $AIRFLOW_HOME
/home/project/airflow
```

2. Run the command below to submit the DAG that was created in the previous exercise.

   ```
   export AIRFLOW_HOME=/home/project/airflow
    cp my_first_dag.py $AIRFLOW_HOME/dags
   ```

3. Verify that your DAG actually got submitted.

4. Run the command below to list out all the existing DAGs.

   ```
   airflow dags list
   ```

5. Verify that `my-first-dag` is a part of the output.

   ```
   airflow dags list|grep "my-first-dag"
   ```

   You should see your DAG name in the output.

6. Run the command below to list out all the tasks in `my-first-dag`.

   ```
   airflow tasks list my-first-dag
   ```

   You should see 2 tasks in the output.

Practice exercise

Write a DAG named `ETL_Server_Access_Log_Processing.py`.

1. Create the imports block.
2. Create the DAG Arguments block. You can use the default settings
3. Create the DAG definition block. The DAG should run daily.
4. Create the download task. The download task must download the server access log file, which is available at the URL:

`https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DB0250EN-SkillsNetwork/labs/Apache%20Airflow/Build%20a%20DAG%20using%20Airflow/web-server-access-log.txt`

5. Create the extract task.

   The server access log file contains these fields.

   a. `timestamp` - TIMESTAMP
   b. `latitude` - float
   c. `longitude` - float
   d. `visitorid` - char(37)
   e. `accessed_from_mobile` - boolean
   f. `browser_code` - int

   The `extract` task must extract the fields `timestamp` and `visitorid`.

6. Create the transform task. The `transform` task must capitalize the `visitorid`.

7. Create the load task. The `load` task must compress the extracted and transformed data.

8. Create the task pipeline block. The pipeline block should schedule the task in the order listed below:

   1. download
   2. extract
   3. transform
   4. load

9. Submit the DAG.

10. Verify if the DAG is submitted.

▼ Click here for **hint**.

Follow the example Python code given in the lab and make necessary changes to create the new DAG.

▼ Click here for the **solution**.

Add to the file the following parts of code to `ETL_Server_Access_Log_Processing.py` to complete the tasks given in the problem.

```
# import the libraries
from datetime import timedelta
# The DAG object; we'll need this to instantiate a DAG
from airflow.models import DAG
# Operators; you need this to write tasks!
from airflow.operators.bash_operator import BashOperator
# This makes scheduling easy
from airflow.utils.dates import days_ago
#defining DAG arguments
# You can override them on a per-task basis during operator initialization
default_args = {
    'owner': 'your_name',
    'start_date': days_ago(0),
    'email': ['your email'],
    'retries': 1,
    'retry_delay': timedelta(minutes=5),
}
# defining the DAG
# define the DAG
dag = DAG(
    'ETL_Server_Access_Log_Processing',
    default_args=default_args,
    description='My first DAG',
    schedule_interval=timedelta(days=1),
)
# define the tasks
# define the task 'download'
download = BashOperator(
    task_id='download',
    bash_command='curl "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DB0250EN-SkillsNetwork/labs/Apache%20Airflow/Build%20a%20DAG%20using%20Airflow/web-server-access-log.txt" -o web-server-access-log.txt',
```

```
    dag=dag,
)
# define the task 'extract'
extract = BashOperator(
    task_id='extract',
    bash_command='cut -f1,4 -d"#" web-server-access-log.txt > /home/project/airflow/dags/extracted.txt',
    dag=dag,
)
# define the task 'transform'
transform = BashOperator(
    task_id='transform',
    bash_command='tr "[a-z]" "[A-Z]" < /home/project/airflow/dags/extracted.txt > /home/project/airflow/dags/capitalized.txt',
    dag=dag,
)
# define the task 'load'
load = BashOperator(
    task_id='load',
    bash_command='zip log.zip capitalized.txt' ,
    dag=dag,
)
# task pipeline
download >> extract >> transform >> load
```

Submit the DAG by running the following command.

```
 cp  ETL_Server_Access_Log_Processing.py $AIRFLOW_HOME/dags
```

Verify if the DAG is submitted on the Web UI or the CLI using the below command.

```
airflow dags list
```

Authors

Lavanya T S
Ramesh Sannareddy