Hands-on Lab: Create and execute a Shell script from Airflow

Estimated time needed: **40** minutes

Objectives

After completing this lab you will be able to:

- Create a basic shell script
- Explore the anatomy of a DAG.
- Create a DAG.
- Call and execute the shell script
- Submit a DAG.

About Skills Network Cloud IDE

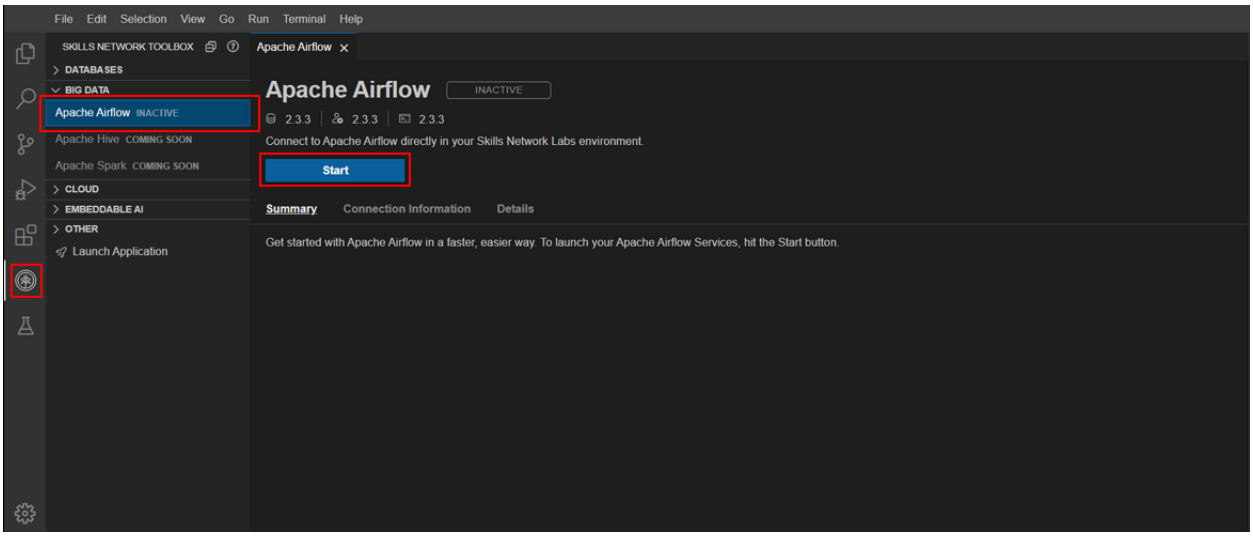
Skills Network Cloud IDE (based on Theia and Docker) provides an environment for hands on labs for course and project related labs. Theia is an open source IDE (Integrated Development Environment), that can be run on desktop or on the cloud. to complete this lab, we will be using the Cloud IDE based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

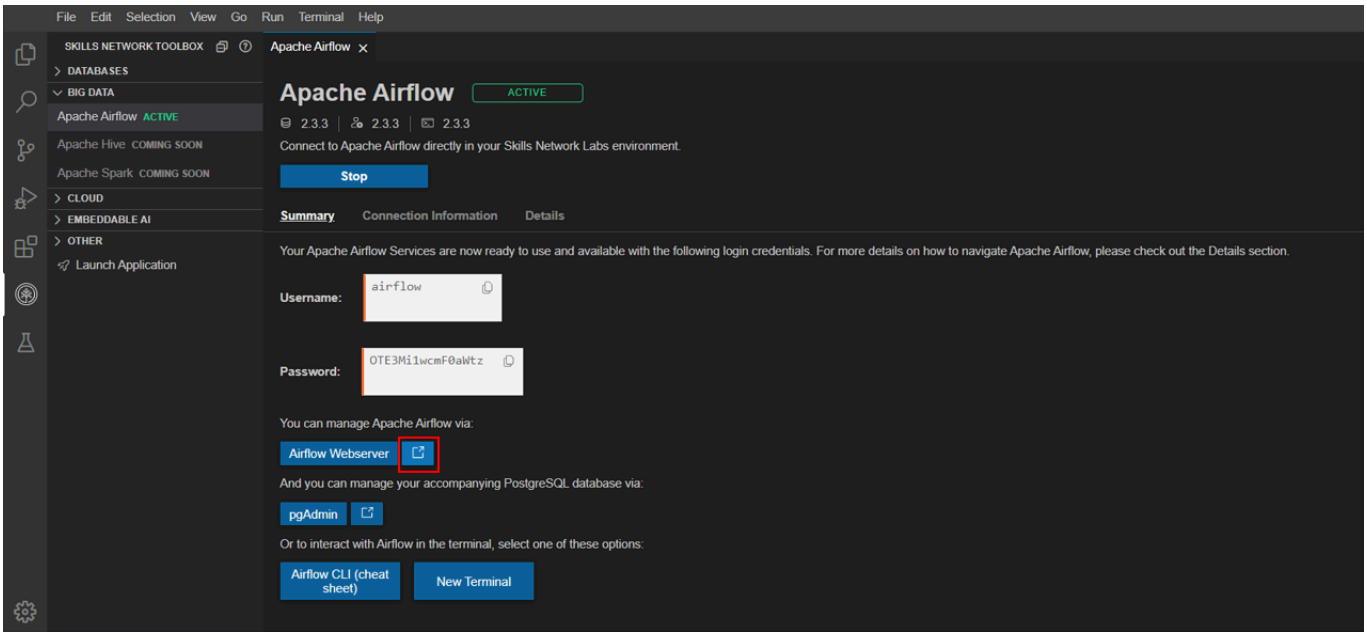
Click on **Skills Network Toolbox**. In **BIG DATA** section, click **Apache Airflow**. To start the Apache Airflow click **Start**.



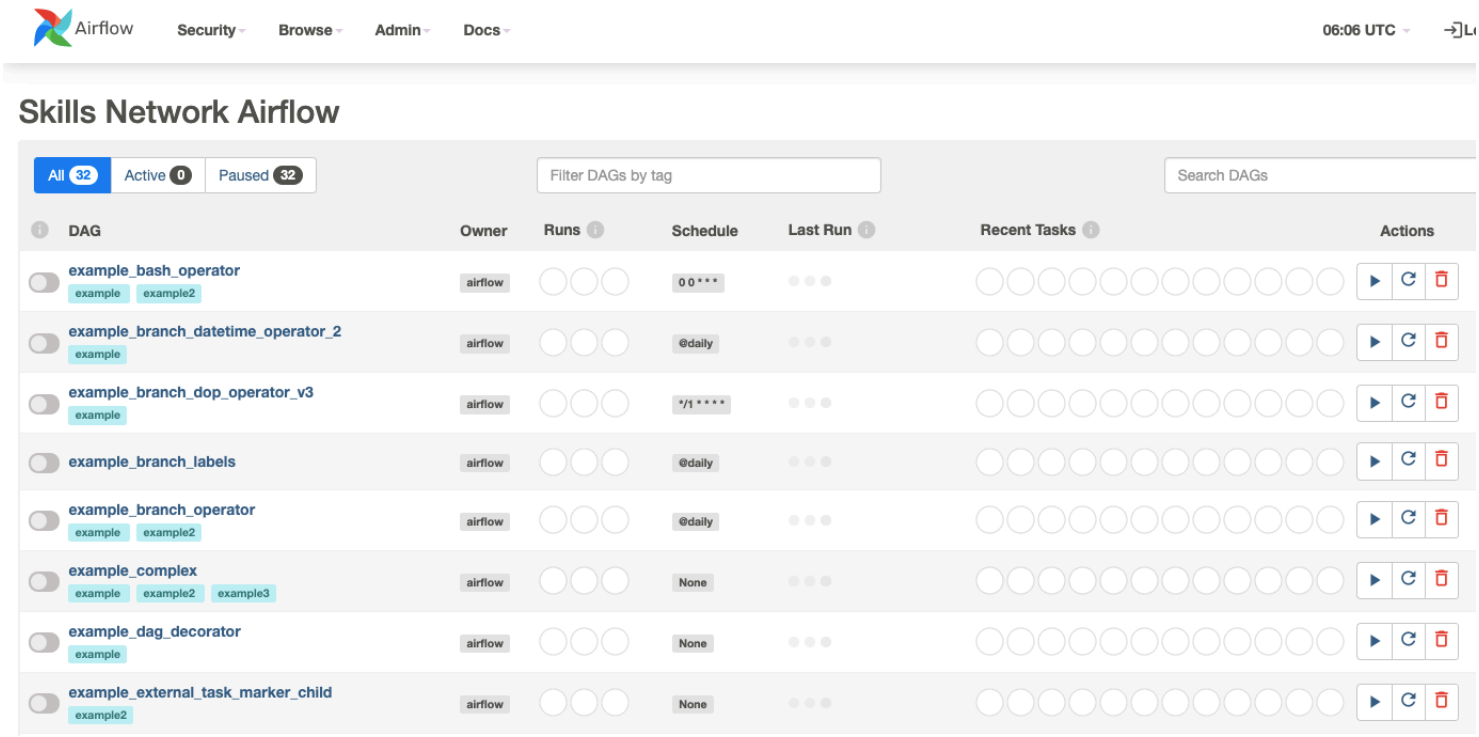
Please be patient, it will take a few minutes for airflow to get started.

Exercise 2 - Open the Airflow Web UI

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.



You should land at a page that looks like this.



Exercise 2 - Create a basic shell script

Here we will define a shell script `extract_transform_load.sh` which will define a pipeline of tasks such as

- extract
- transform
- load

For now, let the shell script have basic echo statements for `extract`,`transform` and `load`.

```
#!/bin/bash
echo "Extract data"
echo "Transform data"
echo "Load data"
```

In the next section we will define a `DAG` to call and execute the shell script.

Exercise 3 - Explore the anatomy of a DAG

An Apache Airflow DAG is a python program. It consists of these logical blocks.

- Imports
- DAG Arguments
- DAG Definition
- Task Definitions

- Task Pipeline

A typical `imports` block looks like this.

```
# import the libraries
from datetime import timedelta
# The DAG object; we'll need this to instantiate a DAG
from airflow import DAG
# Operators; we need this to write tasks!
from airflow.operators.bash_operator import BashOperator
# This makes scheduling easy
from airflow.utils.dates import days_ago
```

A typical `DAG Arguments` block looks like this.

```
#defining DAG arguments
# You can override them on a per-task basis during operator initialization
default_args = {
    'owner': 'Ramesh Sannareddy',
    'start_date': days_ago(0),
    'email': ['ramesh@somemail.com'],
    'email_on_failure': True,
    'email_on_retry': True,
    'retries': 1,
    'retry_delay': timedelta(minutes=5),
}
```

DAG arguments are like settings for the DAG.

The above settings mention

- the owner name,
- when this DAG should run from: days_age(0) means today,
- the email address where the alerts are sent to,
- whether alert must be sent on failure,
- whether alert must be sent on retry,
- the number of retries in case of failure, and
- the time delay between retries.

A typical `DAG definition` block looks like this.

```
# define the DAG
dag = DAG(
    dag_id='sample-etl-dag',
    default_args=default_args,
    description='Sample ETL DAG using Bash',
    schedule_interval=timedelta(days=1),
)
```

Here we are creating a variable named `dag` by instantiating the DAG class with the following parameters.

`sample-etl-dag` is the ID of the DAG. This is what you see on the web console.

We are passing the dictionary `default_args`, in which all the defaults are defined.

`description` helps us in understanding what this DAG does.

`schedule_interval` tells us how frequently this DAG runs. In this case every day. (`days=1`).

A typical `task definitions` block looks like this:

```
# define the tasks
# define the task named extract_transform_and_load to call the shell script
extract_transform_and_load = BashOperator(
    task_id='extract_transform_and_load',
    bash_command='/home/project/airflow/dags/extract_transform_load.sh "',
    dag=dag,
)
```

A task is defined using:

- A task_id which is a string and helps in identifying the task.
- What bash command it represents. Here we are calling the shell script `extract_transform_load.sh`which we previously defined
- Which dag this task belongs to.

A typical `task pipeline` block looks like this:

```
# task pipeline
extract_transform_and_load
```

When we execute the task `extract_transform_and_load` the code in the shell script gets executed.

Exercise 4 - ETL process on a `/etc/passwd` file

Here we will first

- Create a new shell script called `my_first_dag.sh` to perform the ETL process.
- Create a `DAG` file `my_first_dag.py` which will run daily and defines a task `etl` to call the shell script `my_first_etl.sh`.
- Submit the `DAG`

Create a new shell script `my_first_dag.sh` by selecting **File->New File**.

- The shell script extracts the first,third and sixth fields from `/etc/passwd` file using the `cut` command and writes to a new file `extracted-data.txt`
- Next the `extracted-data.txt` is transformed to a `csv` file and loaded into a new file called `transformed-data.csv` using `tr` command.

Copy the code below in the shell script.

```
#!/bin/bash
echo "extract_transform_and_load"
cut -d":" -f1,3,6 /etc/passwd > /home/project/airflow/dags/extracted-data.txt
tr ":" "," < /home/project/airflow/dags/extracted-data.txt > /home/project/airflow/dags/transformed-data.csv
```

Click `Save` to save the shell script.

Create a new `DAG` file `my_first_dag.py`by selecting **File->New File**.

This DAG has one task `etl` that calls the shell script `my_first_dag.sh`

```
# import the libraries
from datetime import timedelta
# The DAG object; we'll need this to instantiate a DAG
from airflow import DAG
# Operators; we 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': 'Ramesh Sannareddy',
    'start_date': days_ago(0),
    'email': ['ramesh@somemail.com'],
    'email_on_failure': False,
    'email_on_retry': False,
    '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 task **extract_transform_and_load** to call shell script
#calling the shell script
extract_transform_load = BashOperator(
    task_id="extract_transform_load",
    bash_command="/home/project/airflow/dags/my_first_dag.sh ",
    dag=dag,
)
# task pipeline
extract_transform_load
```

Exercise 5 - Submit a DAG

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

Open a terminal and run the command below to submit the DAG that was created in the previous exercise.

Note: While submitting the dag that was created in the previous exercise, use **sudo** in the terminal before the command used to submit the dag.

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

Next, run the command below one by one to submit shell script in the dags folder and to change the permission for reading shell script.

```
cp my_first_dag.sh $AIRFLOW_HOME/dags
cd airflow/dags
chmod 777 my_first_dag.sh
```

Verify that our DAG actually got submitted.

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

```
airflow dags list
```

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.

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

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

You should see 1 task in the output.

Practice exercises

1. Problem:

   Download the dataset from the source to the destination mentioned below using `wget` command in terminal.

   Note: While downloading the file in the terminal use the **sudo** command before the command used to download the file.

Source : 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

Destination : `/home/project/airflow/dags`.

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

Write a shell script named `ETL_Server_Access_Log_Processing.sh`.

***Task 1**: Add a command in the shell script to extract the fields `timestamp` and `visitorid` from the `web-server-access-log.txt` and write to a file `extracted.txt`*

***Task 2**: Update the shell script to add a command to capitalize the `visitorid` and write to a new file `capitalized.txt`.*

***Task 3**: Update the shell script to add a command to compress the transformed file `capitalized.txt` into a new file `log.tar.gz`.*

Write a DAG named `ETL_Server_Access_Log_Processing`.

***Task 2**: Create the imports block.*

***Task 3**: Create the DAG Arguments block. You can use the default settings*

***Task 4**: Create the DAG definition block. The DAG should run daily.*

***Task 5**: Create the task extract_transform_and_load to call the shell script.*

***Task 6**: Create the task pipeline block.*

***Task 7**: Submit the DAG.*

***Task 8**: Submit the shell script to dags folder.*

***Task 9**: Change the permission to read shell script.*

***Task 10**. Verify if the DAG is submitted*

▶ Click here for Hint
▶ Click here for Solution

Authors

Ramesh Sannareddy

Pratiksha Verma

Other Contributors

Rav Ahuja

Lakshmi Holla

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2023-03-02	0.6	Pratiksha Verma	Modified instructions to create DAG using shell script
2022-11-10	0.5	Appalabhaktula Hema	Updated instruction
2022-08-22	0.4	Lakshmi Holla	updated bash command
2022-07-29	0.3	Lakshmi Holla	changed dag name
2022-06-28	0.2	Lakshmi Holla	updated DAG path
2021-07-05	0.1	Ramesh Sannareddy	Created initial version of the lab