Hands-on Lab: Build ETL Data Pipelines with BashOperator using Apache Airflow

Estimated time needed: **90** minutes

Project Scenario

You are a data engineer at a data analytics consulting company. You have been assigned a project to decongest the national highways by analyzing the road traffic data from different toll plazas. Each highway is operated by a different toll operator with a different IT setup that uses different file formats. Your job is to collect data available in different formats and consolidate it into a single file.

Objectives

In this assignment, you will develop an Apache Airflow DAG that will:

- Extract data from a csv file
- Extract data from a tsv file
- Extract data from a fixed-width file
- Transform the data
- Load the transformed data into the staging area

Note about screenshots

Throughout this lab, you will be prompted to take screenshots and save them on your device. You will need to upload the screenshots for peer review. You can use various free screen grabbing tools or your operating system's shortcut keys (Alt + PrintScreen in Windows, for example) to capture the required screenshots. You can save the screenshots with the `.jpg` or `.png` extension.

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: Set up the lab environment

1. Start Apache Airflow.

Open Apache Airflow in IDE

2. Open a terminal and create a directory structure for the staging area as follows: **/home/project/airflow/dags/finalassignment/staging**.

```
1
```

```
sudo mkdir -p /home/project/airflow/dags/finalassignment/staging
```

Copied! Executed!

3. Execute the following commands to give appropriate permission to the directories.

```
1
```

```
sudo chmod -R 777 /home/project/airflow/dags/finalassignment
```

Copied! Executed!

4. Download the data set from the source to the following destination using the `curl` command.

```
1
```

```
sudo curl https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DB0250EN-SkillsNetwork/labs/Final%20Assignment/tolldata.tgz -o /home
```

Copied! Executed!

Exercise 2: Create imports, DAG argument, and definition

Please use `BashOperator` for all the tasks in this assignment.

1. Create a new file named `ETL_toll_data.py` in `/home/project` directory and open it in the file editor.
2. Import all the packages you need to build the DAG.
3. Define the DAG arguments as per the following details in the `ETL_toll_data.py` file:

Parameter	Value
owner	<You may use any dummy name>
start_date	today
email	<You may use any dummy email>
email_on_failure	True
email_on_retry	True
retries	1
retry_delay	5 minutes

   Take a screenshot of the task code. Name the screenshot `dag_args.jpg`.

4. Define the DAG in the `ETL_toll_data.py` file using the following details.

Parameter	Value
DAG id	`ETL_toll_data`
Schedule	Daily once
default_args	As you have defined in the previous step
description	Apache Airflow Final Assignment

   Take a screenshot of the command and output you used. Name the screenshot `dag_definition.jpg`.

At the end of this exercise, you should have the following screenshots with `.jpg` or `.png` extension:
1. *dag_args.jpg*
2. *dag_definition.jpg*

Exercise 3: Create the tasks using BashOperator

1. Create a task named `unzip_data` to unzip data. Use the data downloaded in the first part of this assignment in `Set up the lab environment` and uncompress it into the destination directory using `tar`.

   Take a screenshot of the task code. Name the screenshot `unzip_data.jpg`.

   You can locally untar and read through the file `fileformats.txt` to understand the column details.

2. Create a task named `extract_data_from_csv` to extract the fields `Rowid`, `Timestamp`, `Anonymized Vehicle number`, and `Vehicle type` from the `vehicle-data.csv` file and save them into a file named `csv_data.csv`.

   Take a screenshot of the task code. Name the screenshot `extract_data_from_csv.jpg`.

3. Create a task named `extract_data_from_tsv` to extract the fields `Number of axles`, `Tollplaza id`, and `Tollplaza code` from the `tollplaza-data.tsv` file and save it into a file named `tsv_data.csv`.

   Take a screenshot of the task code. Name the screenshot `extract_data_from_tsv.jpg`.

4. Create a task named `extract_data_from_fixed_width` to extract the fields `Type of Payment code`, and `Vehicle Code` from the fixed width file `payment-data.txt` and save it into a file named `fixed_width_data.csv`.

   Take a screenshot of the task code. Name the screenshot `extract_data_from_fixed_width.jpg`.

5. Create a task named `consolidate_data` to consolidate data extracted from previous tasks. This task should create a single csv file named `extracted_data.csv` by combining data from the following files:

   - `csv_data.csv`
   - `tsv_data.csv`
   - `fixed_width_data.csv`

   The final csv file should use the fields in the order given below:

   - `Rowid`
   - `Timestamp`
   - `Anonymized Vehicle number`
   - `Vehicle type`
   - `Number of axles`
   - `Tollplaza id`
   - `Tollplaza code`
   - `Type of Payment code`, and
   - `Vehicle Code`

Hint: Use the bash `paste` command that merges the columns of the files passed as a command-line parameter and sends the output to a new file specified. You can use the command `man paste` to explore more.

Example: `paste file1 file2 > newfile`

Take a screenshot of the command and output you used. Name the screenshot `consolidate_data.jpg`.

6. Create a task named `transform_data` to transform the vehicle_type field in `extracted_data.csv` into capital letters and save it into a file named `transformed_data.csv` in the staging directory.

   Take a screenshot of the command and output you used. Name the screenshot `transform.jpg`.

7. Define the task pipeline as per the details given below:

Task	Functionality
First task	`unzip_data`
Second task	`extract_data_from_csv`
Third task	`extract_data_from_tsv`
Fourth task	`extract_data_from_fixed_width`
Fifth task	`consolidate_data`
Sixth task	`transform_data`

Take a screenshot of the task pipeline section of the DAG. Name the screenshot `task_pipeline.jpg`.

At the end of this exercise, you should have the following screenshots with `.jpg` or `.png` extension:

1. *unzip_data.jpg*
2. *extract_data_from_csv.jpg*
3. *extract_data_from_tsv.jpg*
4. *extract_data_from_fixed_width.jpg*
5. *consolidate_data.jpg*
6. *transform.jpg*
7. *task_pipeline.jpg*

Exercise 4: Getting the DAG operational

1. Submit the DAG. Use CLI or Web UI to show that the DAG has been properly submitted.
   Take a screenshot showing that the DAG you created is in the list of DAGs. Name the screenshot `submit_dag.jpg`.

   Note: If you don't find your DAG in the list, you can check for errors using the following command in the terminal:

```
1
```

```
airflow dags list-import-errors
```

Copied! Executed!

2. Unpause and trigger the DAG through CLI or Web UI.
3. ***Take a screenshot*** of DAG unpaused on CLI or the GUI. Name the screenshot `unpause_trigger_dag.jpg`.
4. ***Take a screenshot*** of the tasks in the DAG run through CLI or Web UI. Name the screenshot `dag_tasks.jpg`.
5. ***Take a screenshot*** the DAG runs for the Airflow console through CLI or Web UI. Name the screenshot `dag_runs.jpg`.

Screenshot checklist

You should have the following screenshots with `.jpg` or `.png` extension:

1. *submit_dag.jpg*
2. *unpause_trigger_dag.jpg*
3. *dag_tasks.jpg*
4. *dag_runs.jpg*

This action concludes the assignment.

Authors

Lavanya T S
Ramesh Sannareddy

Other Contributors

Rav Ahuja