Final Assignment (Part 1) - Creating ETL Data Pipelines using Apache Airflow

Estimated time needed: **90** minutes.

About This SN Labs Cloud IDE

This Skills Network Labs Cloud IDE provides a hands-on environment for course and project related labs. It utilizes Theia, an open-source IDE (Integrated Development Environment) platform, that can be run on desktop or on the cloud. To complete this lab, we will be using the Cloud IDE based on Theia and Apache Airflow and MySQL database running in a Docker container. You will also need an instance of DB2 running in IBM Cloud.

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.

Scenario

You are a data engineer at a data analytics consulting company. You have been assigned to a project that aims to de-congest 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 author 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 - Screenshots

Throughout this lab you will be prompted to take screenshots and save them on your own device. These screenshots will need to be uploaded for peer review in the next section of the course. You can use various free

screengrabbing tools or your operating system's shortcut keys (Alt + PrintScreen in Windows, for example) to capture the required screenshots. The screenshots can be saved with either the `.jpg` or `.png` extension.

Exercise 1 - Prepare the lab environment

1. Start Apache Airflow.

Open Apache Airflow in IDE

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

```
1. 1
```

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

Copied! Executed!

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

   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/Final%20Assignment/tolldata.tgz

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

4. Change to the staging directory.

```
1. 1
```

```
1. cd /home/project/airflow/dags/finalassignment/staging
```

Copied!

Exercise 2 - Create a DAG

Task 1.1 - Define DAG arguments

Define the DAG arguments as per the following details:

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`.

Task 1.2 - Define the DAG

Create a DAG as per 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 you used and the output.

Name the screenshot `dag_definition.jpg`.

Task 1.3 - Create a task to unzip data

Create a task named `unzip_data`.

Use the downloaded data from the url given in the first part of this assignment in exercise 1 and uncompress it into the destination directory.

Take a screenshot of the task code.

Name the screenshot `unzip_data.jpg`.

Read through the file `fileformats.txt` to understand the column details.

Task 1.4 - Create a task to extract data from csv file

Create a task named `extract_data_from_csv`.

This task should 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`.

Task 1.5 - Create a task to extract data from tsv file

Create a task named `extract_data_from_tsv`.

This task should 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`.

Task 1.6 - Create a task to extract data from fixed width file

Create a task named `extract_data_from_fixed_width`.

This task should 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`.

Task 1.7 - Create a task to consolidate data extracted from previous tasks

Create a task named `consolidate_data`.

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.

`paste` command merges lines of files.

Example : `paste file1 file2 > newfile`

The above command merges the columns of the files file1 and file2 and sends the output to newfile.

You can use the command `man paste` to explore more.

Take a screenshot of the command you used and the output.

Name the screenshot `consolidate_data.jpg`.

Task 1.8 - Transform and load the data

Create a task named `transform_data`.

This task should 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 you used and the output.

Name the screenshot `transform.jpg`.

Task 1.9 - Define the task pipeline

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`.

Exercise 3 - Getting the DAG operational.

Save the DAG you defined into a file named `ETL_toll_data.py`.

Task 1.10 - Submit the DAG

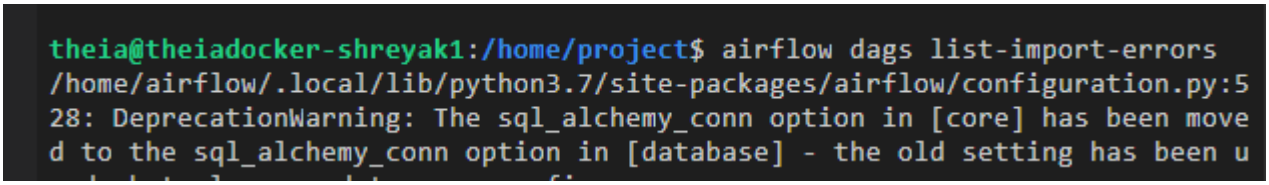
Take a screenshot of the command you used and the output.

Name the screenshot `submit_dag.jpg`.

Note:

If you face issues while submitting the DAG, you can check for errors using the following command in terminal:

airflow dags list-import-errors



Task 1.11 - Unpause the DAG

Take a screenshot of the command you used and the output.

Name the screenshot `unpause_dag.jpg`.

Task 1.12 - Monitor the DAG

Take a screenshot of the DAG runs for the Airflow console.

Name the screenshot `dag_runs.jpg`.

This concludes the assignment.

Authors

Ramesh Sannareddy

Other Contributors

Rav Ahuja

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2021-08-06	0.1	Ramesh Sannareddy	Created initial version
2022-08-26	0.2	Lakshmi Holla	Updated sudo commands