Data Pipelines Using Apache AirFlow

Estimated time needed: **30** 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 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.

Scenario

Write a pipeline that analyzes the web server log file, extracts the required lines(ending with html) and fields(time stamp, size) and transforms (bytes to mb) and load (append to an existing file.)

Objectives

In this assignment you will author an Apache Airflow DAG that will:

- Extract data from a web server log file
- Transform the data
- Load the transformed data into a tar file

Tools / Software

- Apache AirFlow

Note - Screenshots

Throughout this lab you will be prompted to take screenshots and save them on your own device. You will need these screenshots to either answer graded quiz questions or to upload as your submission for peer review at the

end of this course. You can use various free screengrabbing tools to do this or use your operating system's shortcut keys to do this (for example Alt+PrintScreen in Windows).

Exercise 1 - Prepare the lab environment

Before you start the assignment:

- Start Apache Airflow.
- Download the dataset from the source to the destination mentioned below. wget link

Source : https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DB0321EN-SkillsNetwork/ETL/accesslog.txt

Destination : /home/project/airflow/dags/capstone

Exercise 2 - Create a DAG

Task 1 - Define the DAG arguments

Create a DAG with these arguments.

- owner
- start_date
- email

You may define any suitable additional arguments.

Take a screenshot of the code you used clearly showing the above arguments.

Name the screenshot `dag_args.jpg`. (Images can be saved with either the .jpg or .png extension.)

Task 2 - Define the DAG

Create a DAG named `process_web_log` that runs daily.

Use suitable description.

Take a screenshot of the code you used to define the DAG.

Name the screenshot `dag_definition.jpg`. (Images can be saved with either the .jpg or .png extension.)

Task 3 - Create a task to extract data

Create a task named `extract_data`.

This task should extract the ipaddress field from the web server log file and save it into a file named extracted_data.txt

Take a screenshot of the task code.

Name the screenshot `extract_data.jpg`. (Images can be saved with either the .jpg or .png extension.)

Task 4 - Create a task to transform the data in the txt file

Create a task named `transform_data`.

This task should filter out all the occurrences of ipaddress "198.46.149.143" from extracted_data.txt and save the output to a file named `transformed_data.txt`.

Take a screenshot of the task code.

Name the screenshot `transform_data.jpg`. (Images can be saved with either the .jpg or .png extension.)

Task 5 - Create a task to load the data

Create a task named `load_data`.

This task should archive the file `transformed_data.txt` into a tar file named `weblog.tar`.

Take a screenshot of the task code.

Name the screenshot `load_data.jpg`. (Images can be saved with either the .jpg or .png extension.)

Task 6 - Define the task pipeline

Define the task pipeline as per the details given below:

Task	Functionality
First task	`extract_data`
Second task	`transform_data`
Third task	`load_data`

Take a screenshot of the task pipeline section of the DAG.

Name the screenshot `pipeline.jpg`. (Images can be saved with either the .jpg or .png extension.)

Exercise 3 - Getting the DAG operational.

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

Task 7 - Submit the DAG

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

Name the screenshot `submit_dag.jpg`. (Images can be saved with either the .jpg or .png extension.)

Task 8 - Unpause the DAG

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

Name the screenshot `unpause_dag.jpg`. (Images can be saved with either the .jpg or .png extension.)

Task 9 - Monitor the DAG

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

Name the screenshot `dag_runs.jpg`. (Images can be saved with either the .jpg or .png extension.)

End of the assignment.

Authors

Ramesh Sannareddy

Other Contributors

Rav Ahuja

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2021-13-12	0.1	Ramesh Sannareddy	Created initial version
2022-30-01	0.2	Alison Woolford	Updated version
2022-04-14	0.2	Alison Woolford	Updated version


```
# 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': 'Sam Smith',
'start_date': days_ago(0),
'email': ['samsmith@xyz.com'],
'email_on_failure': True,
'email_on_retry': True,
'retries': 1,
'retry_delay': timedelta(minutes=5),
}
# defining the DAG
dag = DAG(
dag_id = 'Capstone-etl-dag',
default_args = default_args,
description = 'Capstone Project ETL DAG',
schedule_interval = timedelta(days=1),
)
# defining the first task to extract_data by calling shell script
extract_data = BashOperator(
task_id = "extract_data",
bash_command = 'cut -d" " -f1 /home/project/airflow/dags/capstone/accesslog.txt > /home/project/airflow/dags/capstone/extracted_data.txt'
dag=dag,
)
# defining the second task to transform_data
# filter out all the occurrences of ipaddress “198.46.149.143” from extracted_data.txt and save toutput to transformed_data.txt.
transform_data = BashOperator(
task_id = "transform_data",
bash_command = 'grep -v "198.46.149.143" /home/project/airflow/dags/capstone/extracted_data.txt > /home/project/airflow/dags/capstone/transformed_data.txt'
dag=dag,
)
# defining the third task to load_data
# archive the file 'transformed_data.txt' into a tar file named 'weblog.tar'.
load_data = BashOperator(
task_id = "load_data",
bash_command = 'tar -cvf /home/project/airflow/dags/capstone/weblog.tar /home/project/airflow/dags/capstone/transformed_data.txt'
dag=dag,
)
# task Pipeline
extracted_data >> transformed_data >> load_data
```