# **CSYE 7245 - Big Data Sys and Int Analytics**

## **Lab 8 - Airflow TFX**

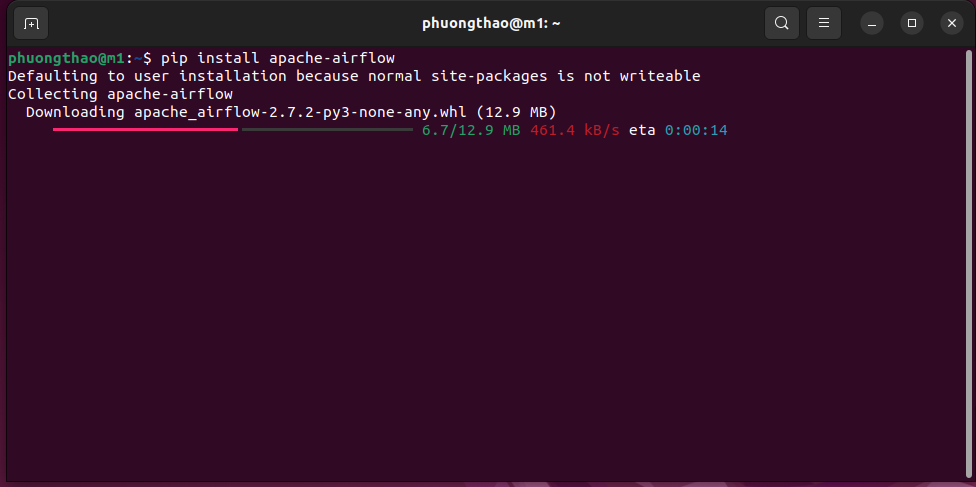


* **Airflow** is a platform to programmatically automate, schedule and monitor workflows.
* In Airflow, a **DAG – or a Directed Acyclic Graph** – is a collection of all the tasks you want to run, organized in a way that reflects their relationships and dependencies.
* The **Airflow scheduler** executes your tasks on an array of workers while following the specified dependencies.
* Rich command line utilities make performing complex surgeries on DAGs a snap.
* The rich user interface makes it easy to visualize pipelines running in production, monitor progress, and troubleshoot issues when needed.

# Experiment Setup

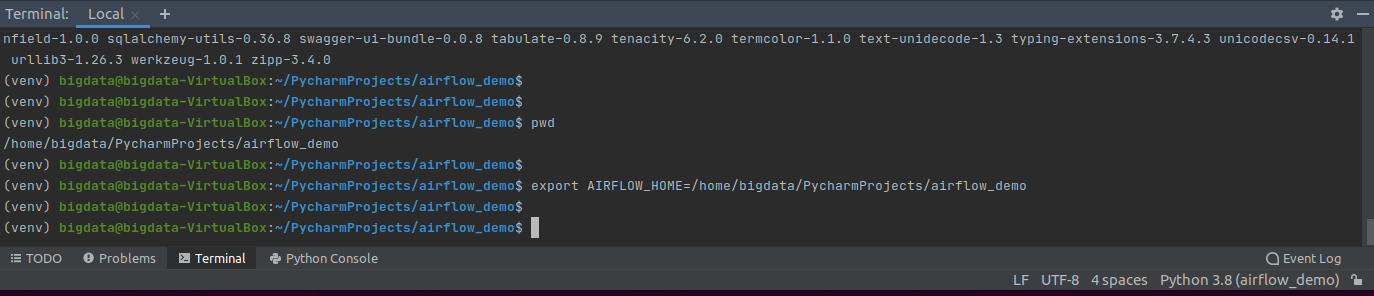
1. Create a new project and install the required dependencies.

pip install apache-airflow



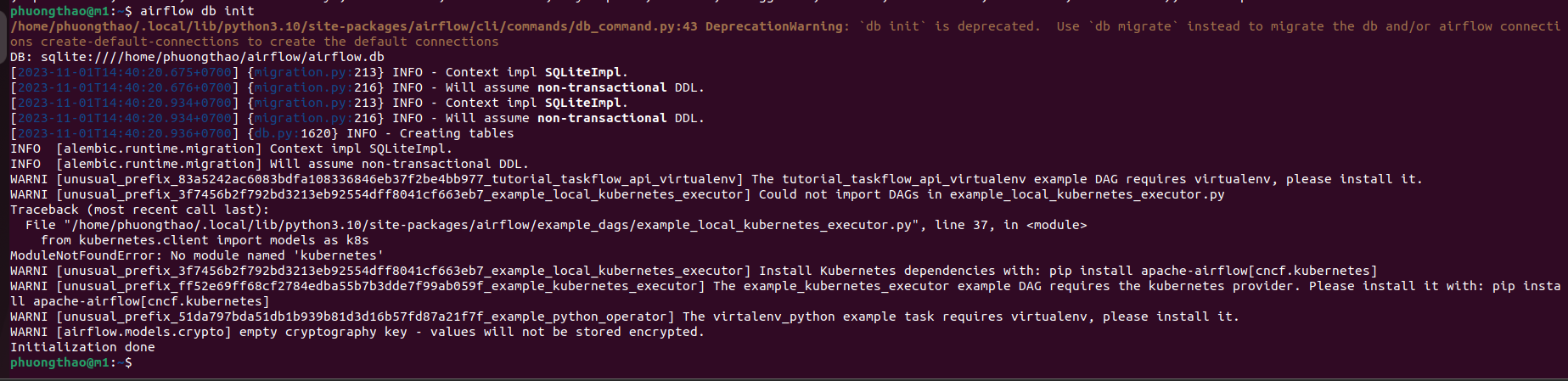
1. EXPORT AIRFLOW\_HOME

Enter the path of the present working directory



1. Initialize the instance

airflow db init



1. Create an admin user

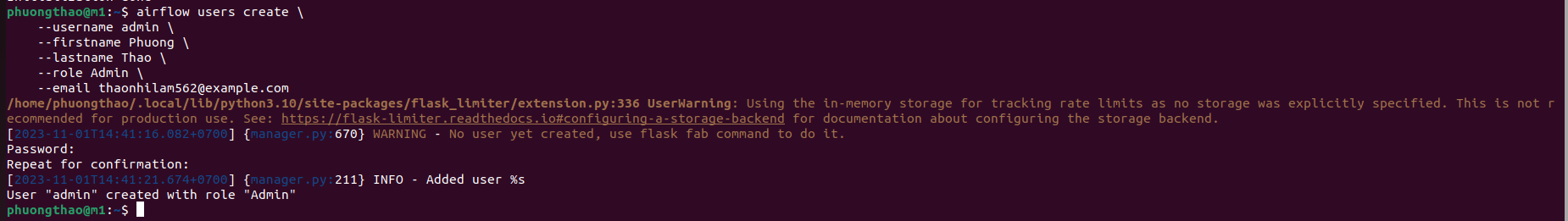
airflow users create\

--username admin\

--firstname Phuong\

--lastname Thao\

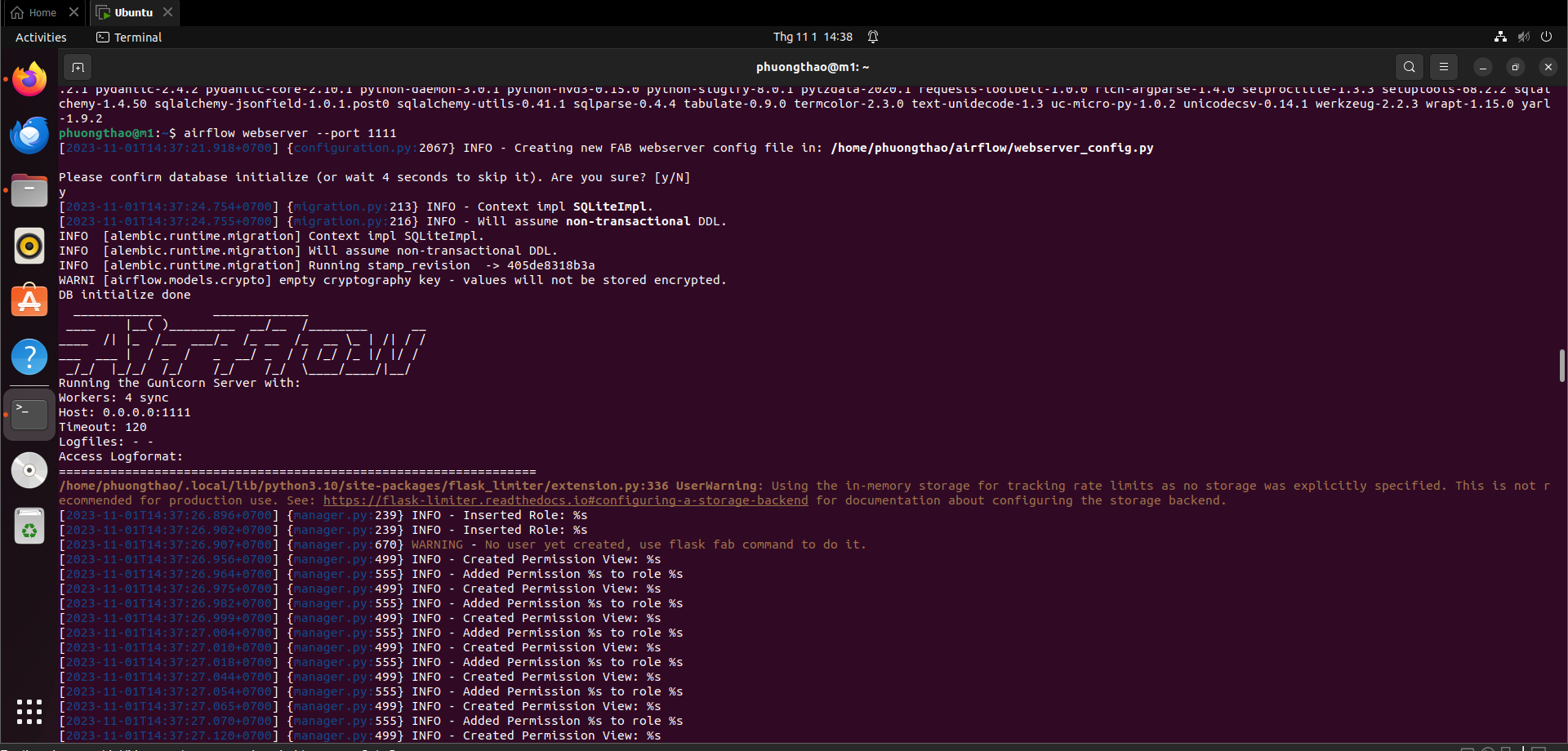
--role Admin\

--email thaonhilam562@example.com

1. Start the Daemon in the background.

airflow webserver -D

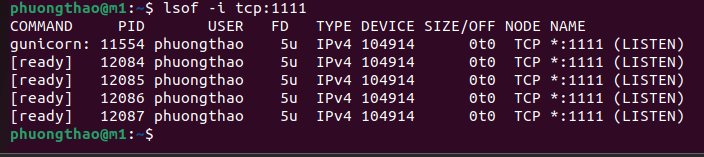
It usually runs on port 1111



1. To check whether Airflow Daemon is running:

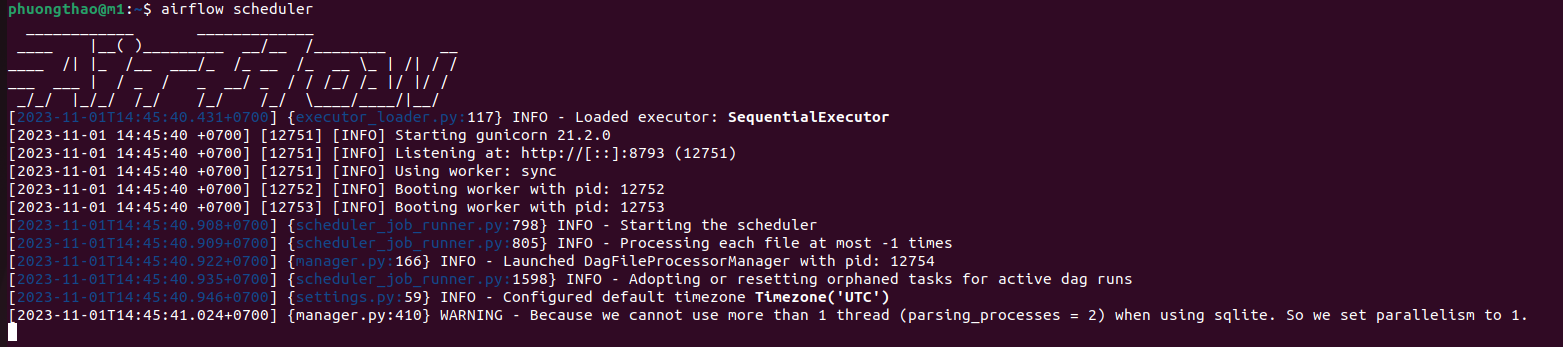
List the services running on port 1111

**lsof -i tcp:1111**

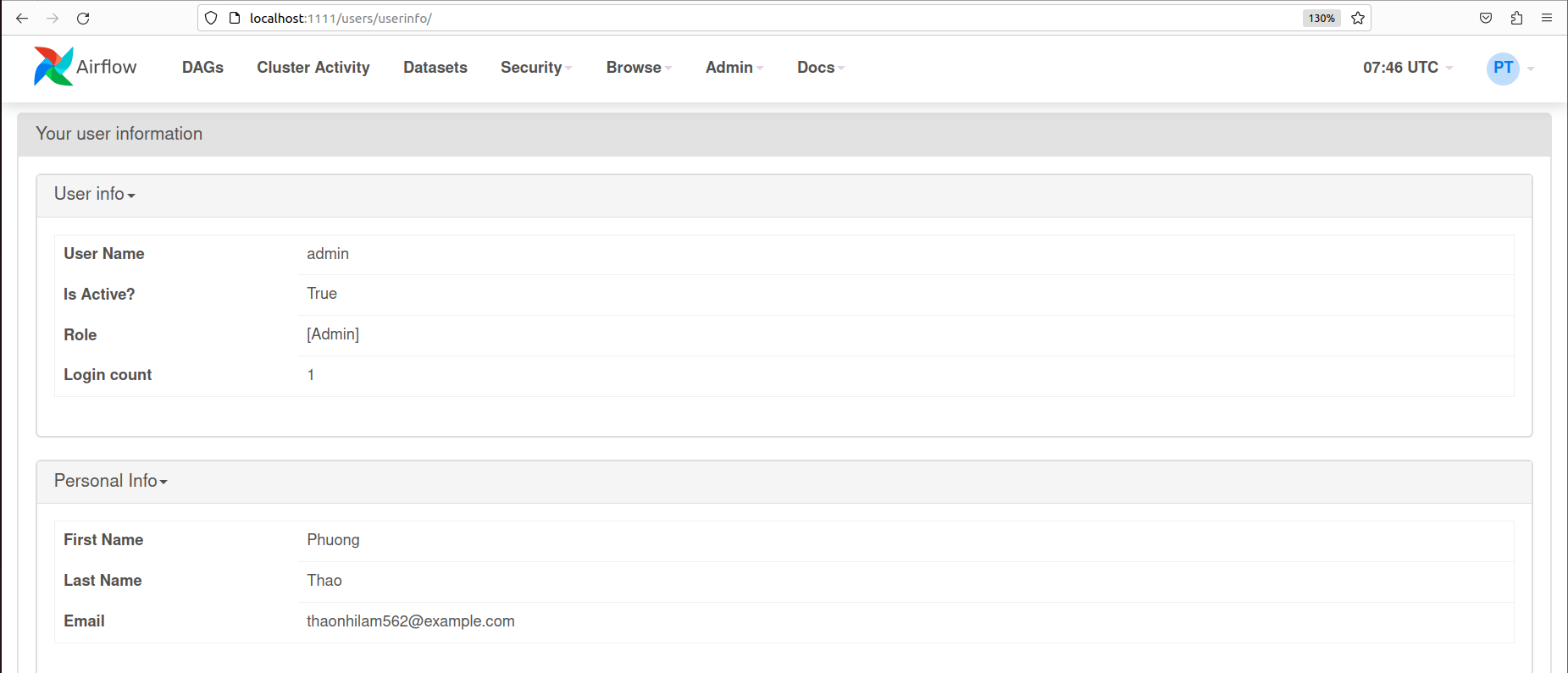
****

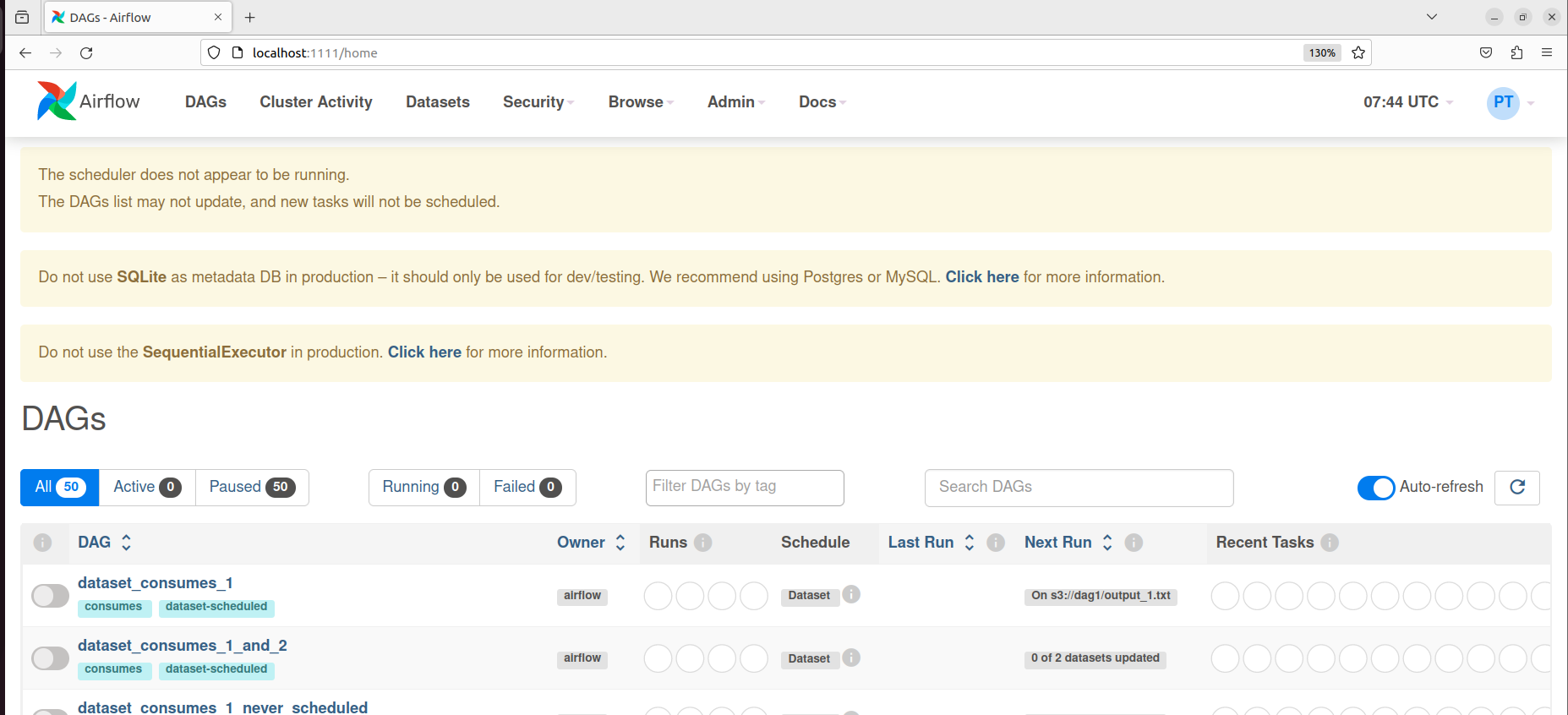
1. Start the scheduler

**airflow scheduler**

****

**Localhost:1111**

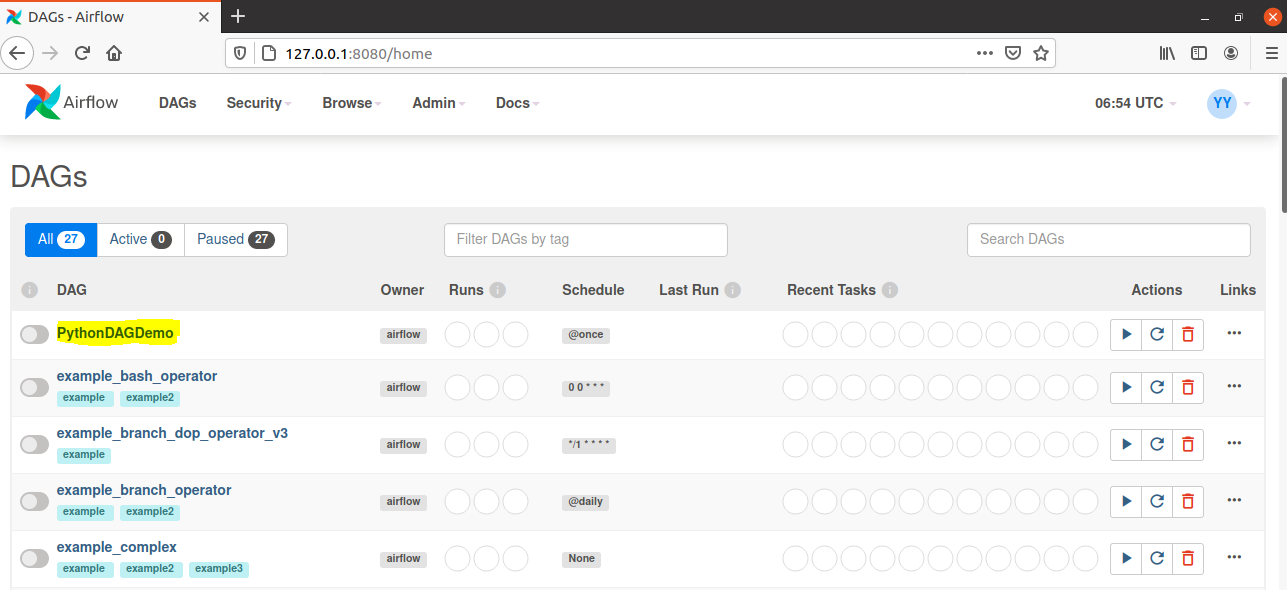


****

1. Create folder dags inside AIRFLOW\_HOME

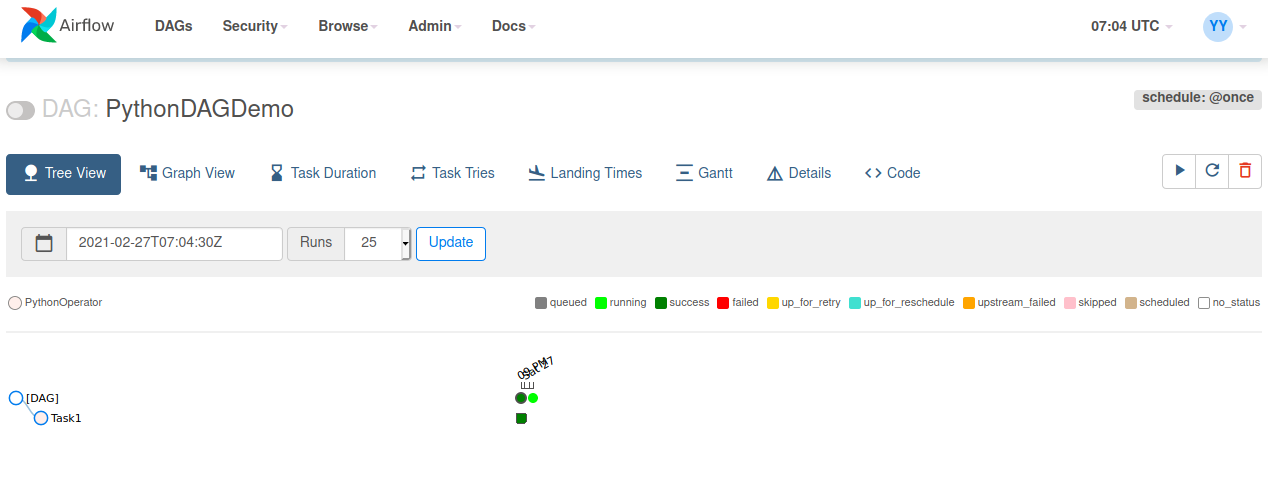
Place the DemoDag python file under the ‘dags’ folder.

1. The file can now be seen under the ‘dags’ folder.

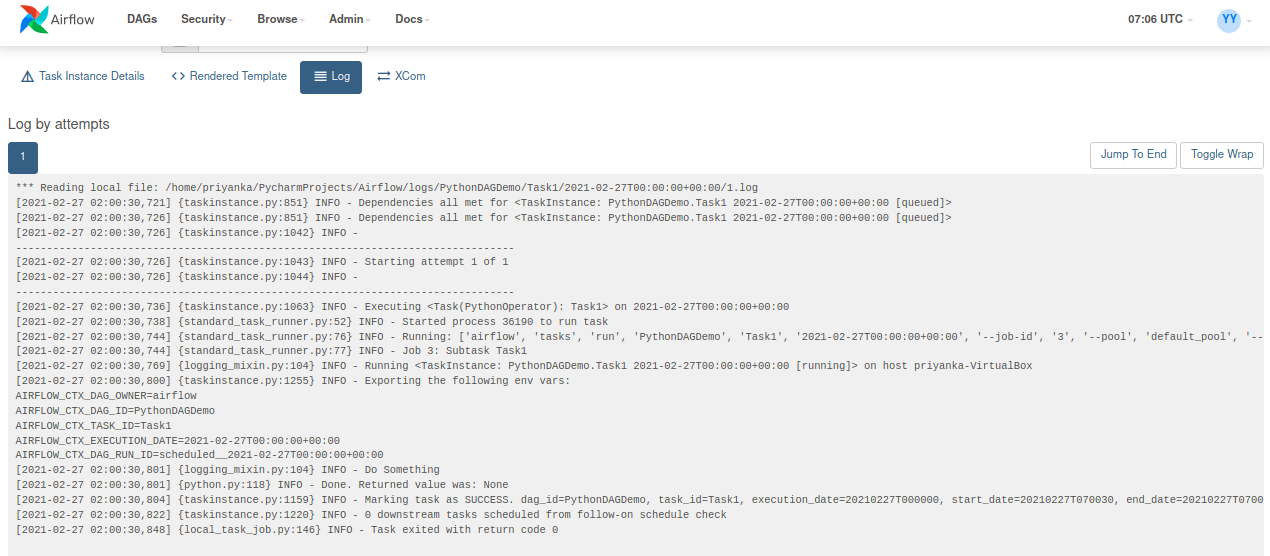


* Dags can be scheduled and run every minute or hourly/daily
* You can also pause/unpause the dag depending on the requirement

1. Trigger your dag

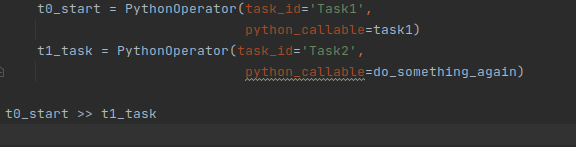


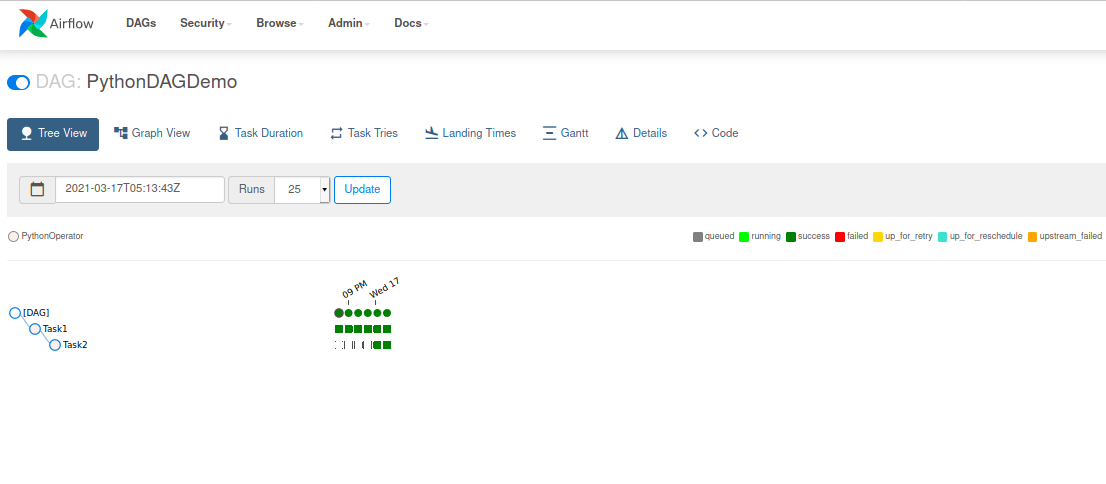
1. Check the logs for additional information



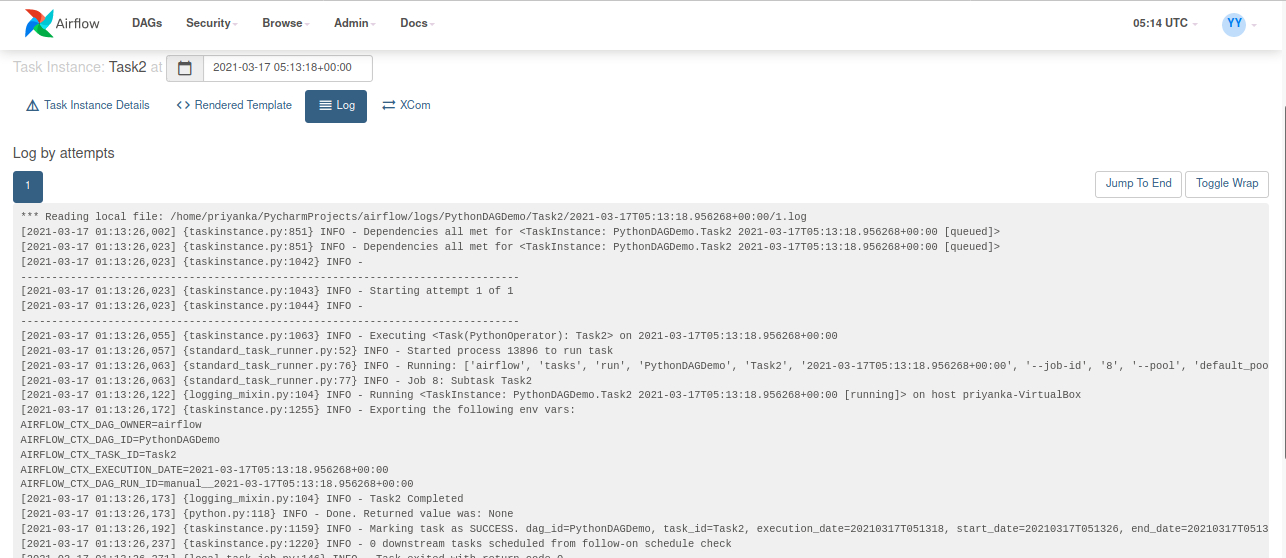
1. Adding tasks to a DAG

Adding task\_2 by making changes in the code and clicking the ‘Update’ button

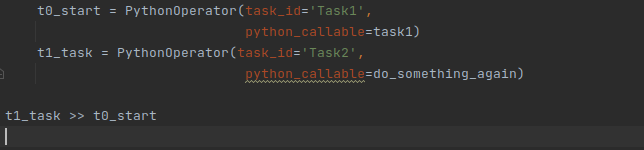


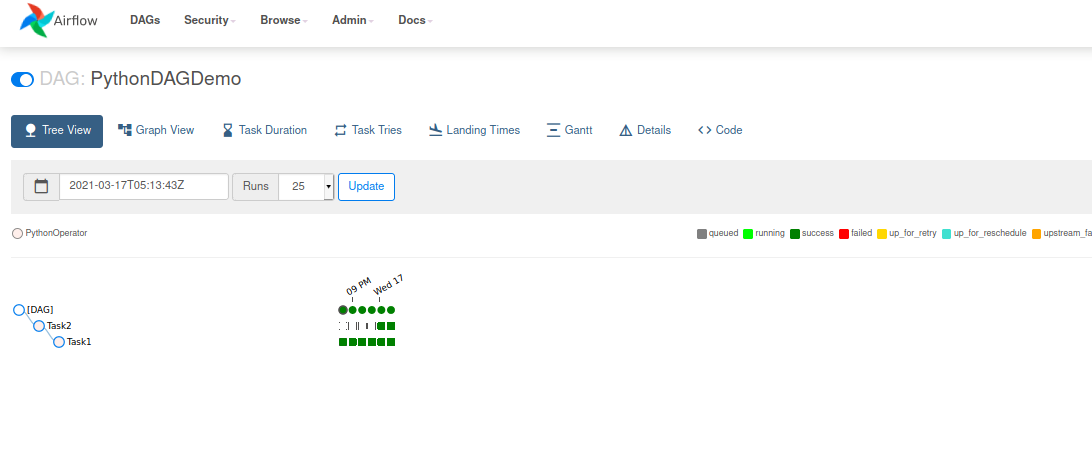


Checking logs for additional information



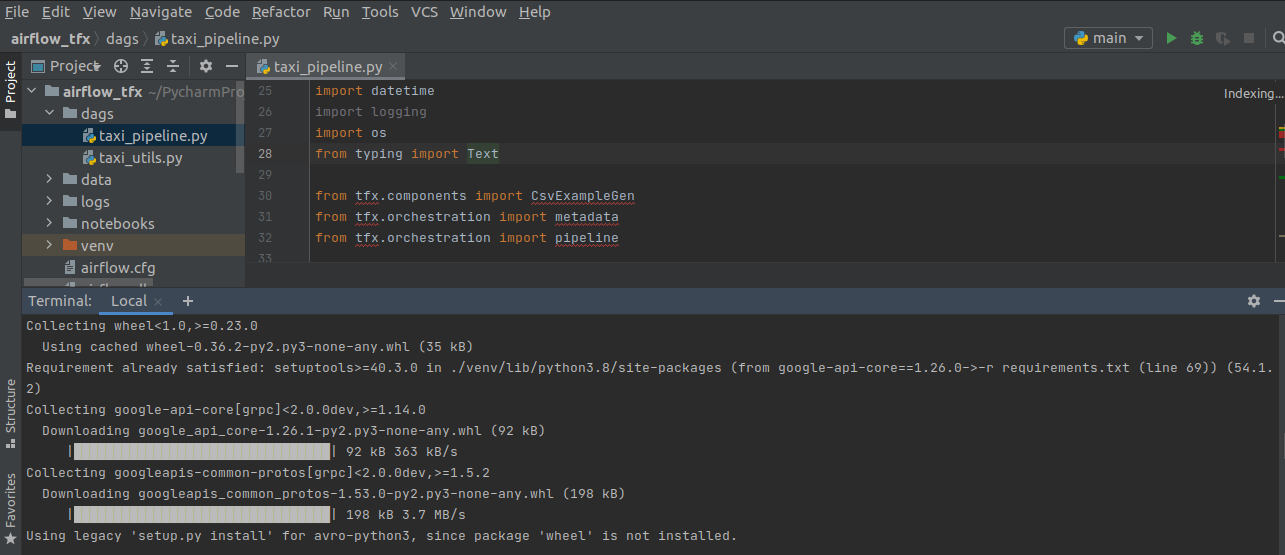
1. Restructuring the code



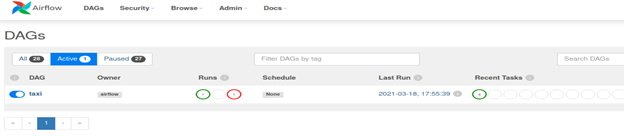


# **Airflow TFX**

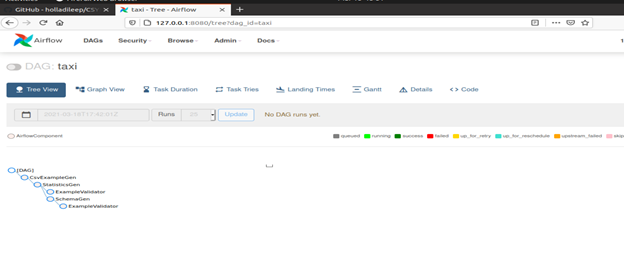
1. Installing ‘requirements.txt’



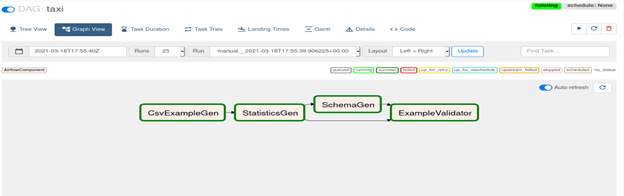
1. You can now see the ‘taxi\_pipeline’ dag in the dags folder.



# The Tree view looks something like this:



1. Successful execution of ‘taxi pipeline’.



* [ExampleGen](https://www.tensorflow.org/tfx/guide/examplegen) ingests and splits the input dataset.
* [StatisticsGen](https://www.tensorflow.org/tfx/guide/statsgen) calculates statistics for the dataset.
* [SchemaGen](https://www.tensorflow.org/tfx/guide/schemagen) SchemaGen examines the statistics and creates a data schema.
* [ExampleValidator](https://www.tensorflow.org/tfx/guide/exampleval) looks for anomalies and missing values in the dataset.

# Lessons learned

1. This lab helps us understand how Airflow allows users to create workflows with high granularity and track the progress as they execute.
2. Airflow enables us to have a platform that can run and automate all the jobs on a schedule.
3. You can also add/transform jobs as and when required.

# 

# References

1. <https://www.tensorflow.org/tfx/tutorials/tfx/airflow_workshop>
2. <https://github.com/tensorflow/tfx/tree/master/tfx/examples/airflow_workshop>