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

## **Lab9 - Acne Type Classification Pipeline using CNN**

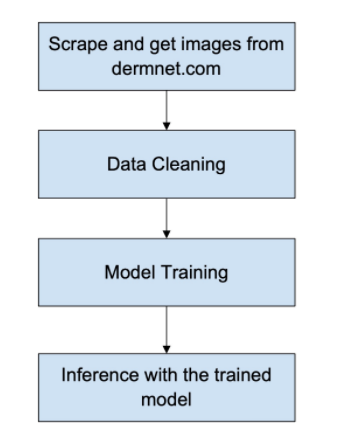
### **Team Information:**

|  |  |
| --- | --- |
| **Name** | **NUID** |
| Tanvi Gurav | 001306848 |
| Keerti Ojha | 001050173 |
| Priyanka Malpekar | 001302741 |

**Lab Completion Date:** **26th February’21**

# **About**

This lab demonstrates how to create a training pipeline that aims to identify the type of Acne-Rosacea, by training a model with images scraped from [dermnet.com](http://www.dermnet.com/) with a confidence score. The front-end application uses Streamlit to predict using the trained model.

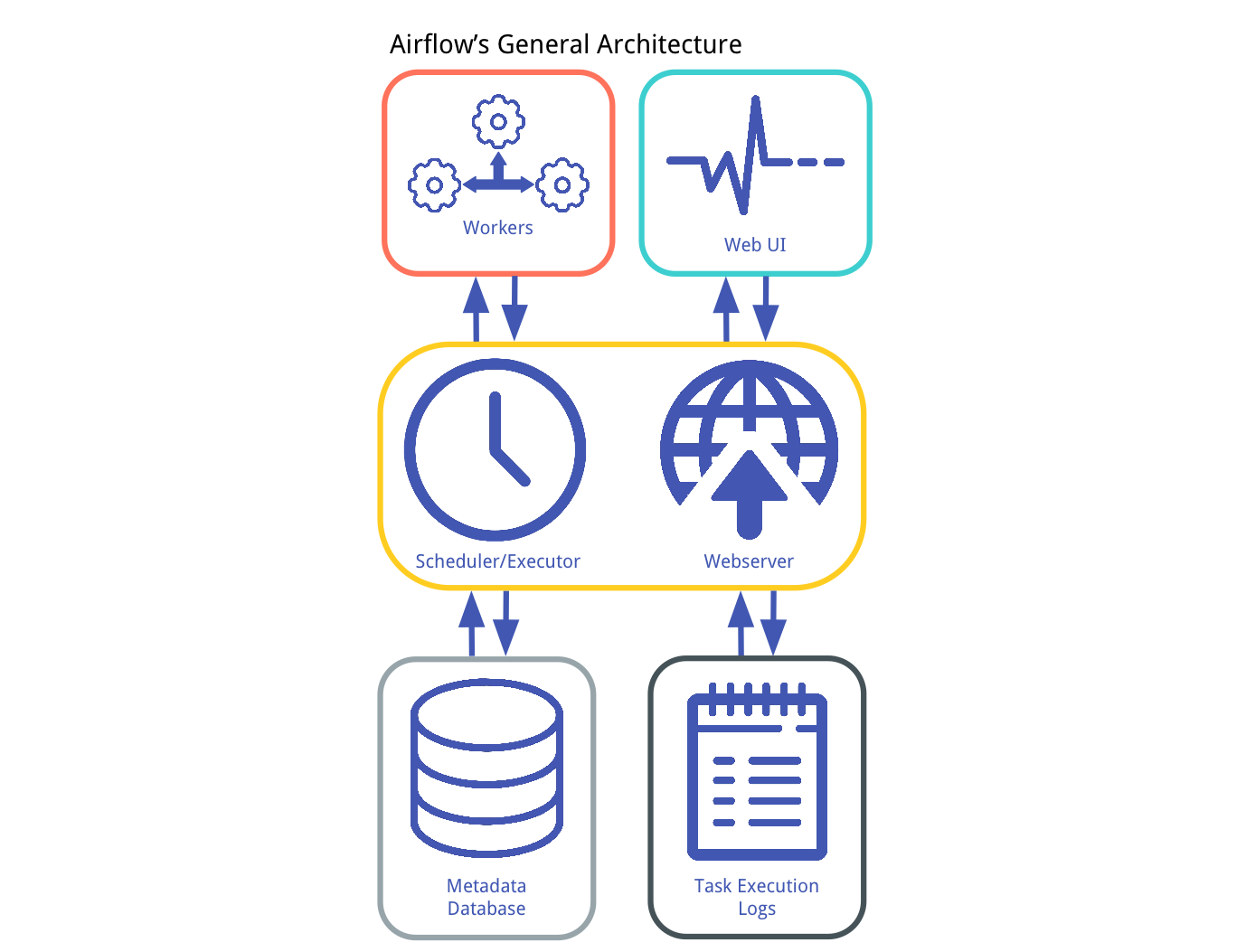


**Orchestration with Apache Airflow**

Airflow is a platform to programmatically author, 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. Use Airflow to author workflows as Directed Acyclic Graphs (DAGs) of tasks. 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. When workflows are defined as code, they become more maintainable, versionable, testable, and collaborative.

# **Dataset**

The dataset used for this lab is from [Dermnet](http://www.dermnet.com/). Dermnet is the largest independent photo dermatology source dedicated to online medical education though articles, photos and video. Dermnet provides information on a wide variety of skin conditions through innovative media. The following are the list of skin problems for which photo dictionary is available:



# **Experiment Setup**

The following are the prerequisite setup we made for the implementation of lab:

**1. Install the dependencies as outlined in the requirements.txt by running**

pip install -r requirements.txt

**2. Install Airflow in the virtual environment**

pip install apache-airflow

**3. Change the bucket name in** s3\_uploader/upload\_models.py

**4. Configure airflow using following commands:**

- Use the current directory as $AIRFLOW\_HOME

export AIRFLOW\_HOME=/home/bigdata/Documents/PyCharmProjects/airflow\_cnn\_pipeline

- Initialize the database

airflow db init

- Create credentials to access airflow server

airflow users create \

--username admin \

--firstname YourName \

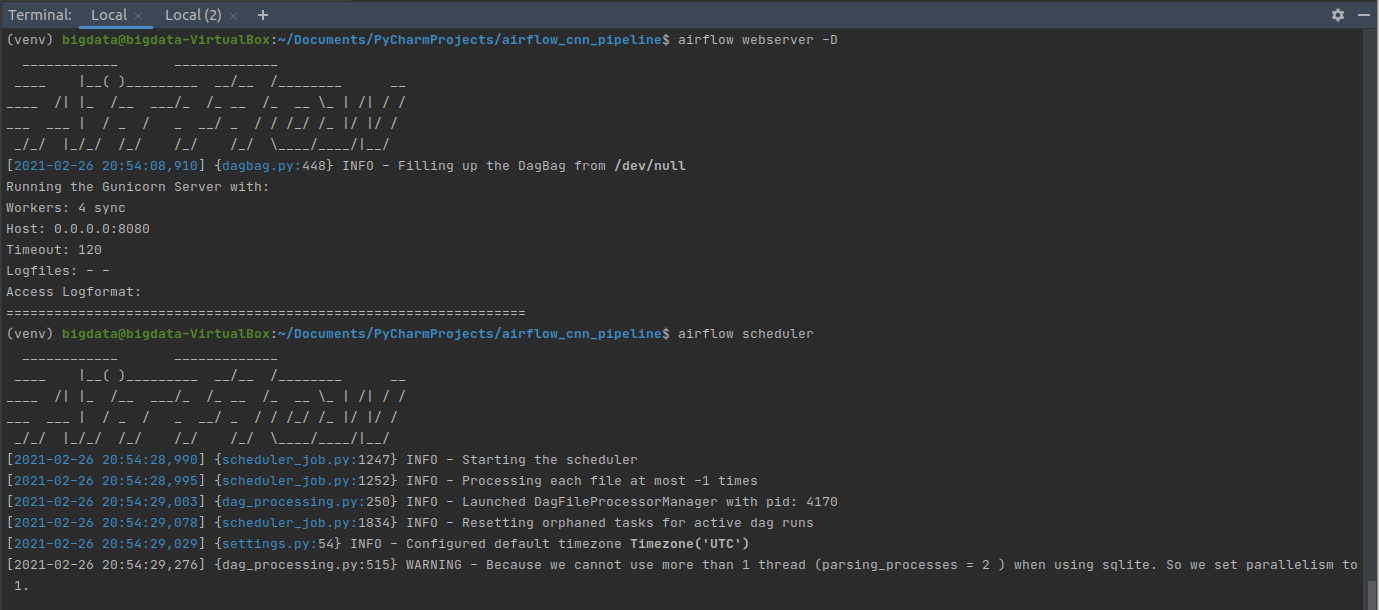
--lastname YourLastName \

--role Admin \

--email [example@example.com](mailto:example@example.com)

- Starting the Webserver to access Airflow server

airflow webserver -D

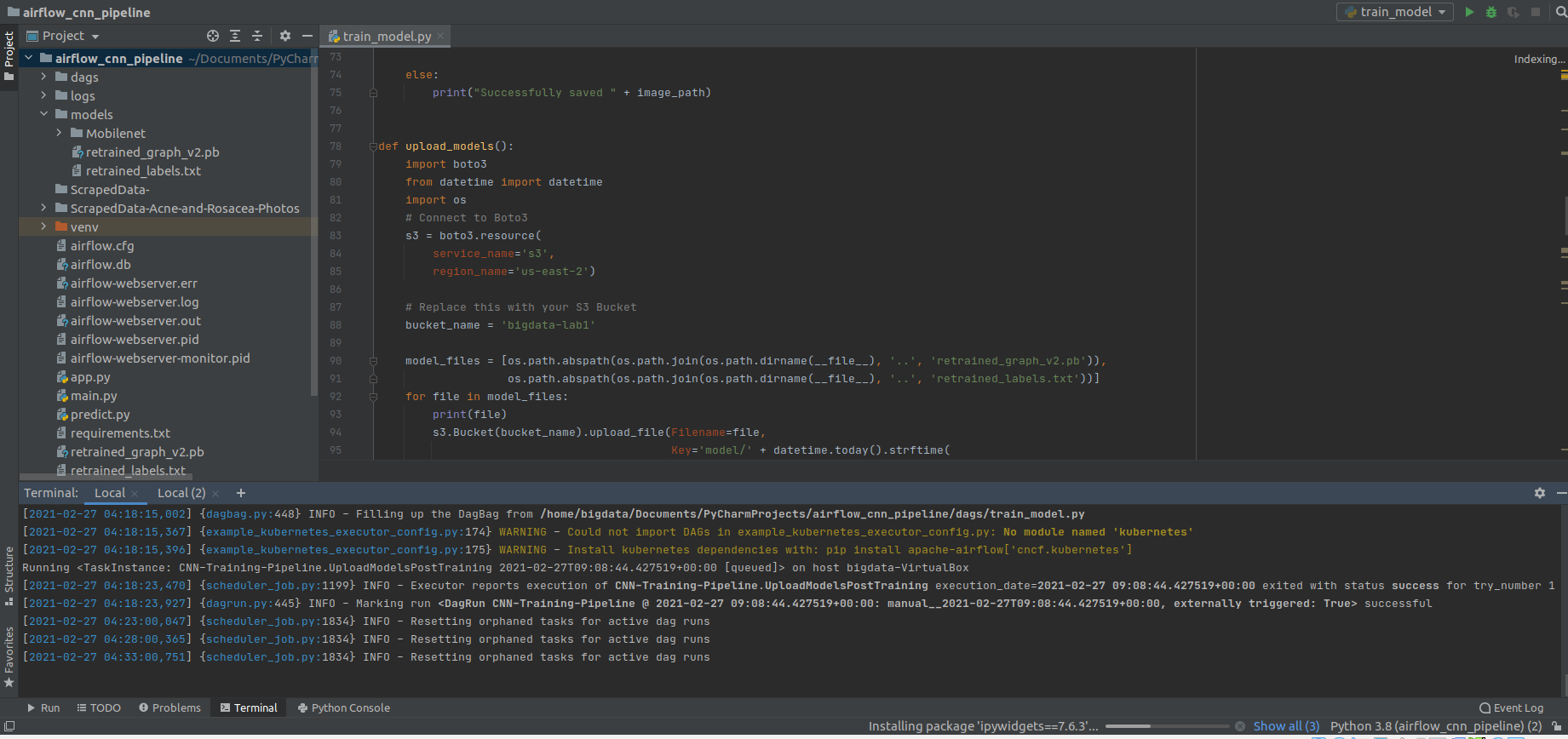


- Before running the scheduler, make sure your DAG code is within the **dags** folder, in our case train\_model.py has our DAG code and hence it should be inside the dags folder. If we have already started the scheduler, get the pid using following command and then kill it using kill -9 <pid>

lsof -i tcp:8080

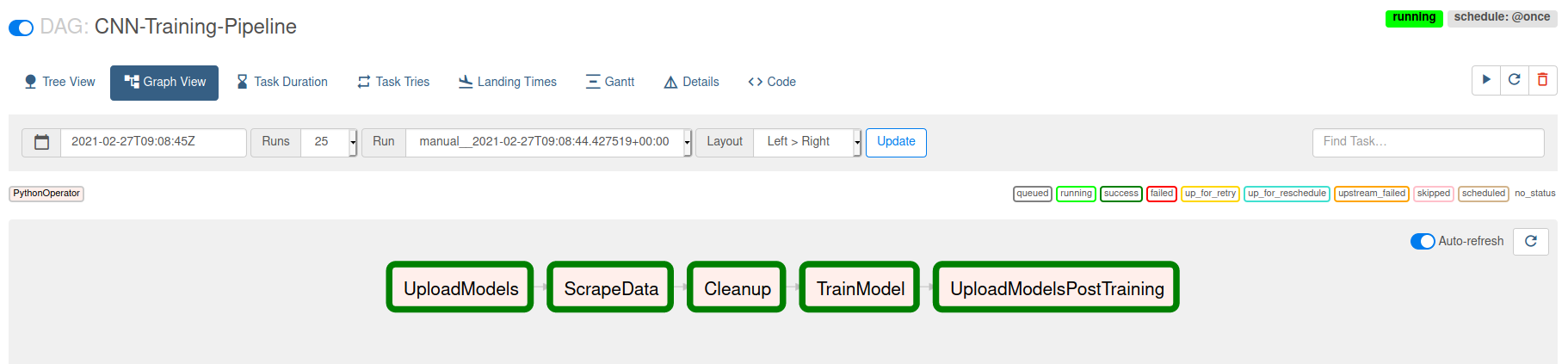
- Once these configurations are done, start the airflow scheduler

airflow scheduler

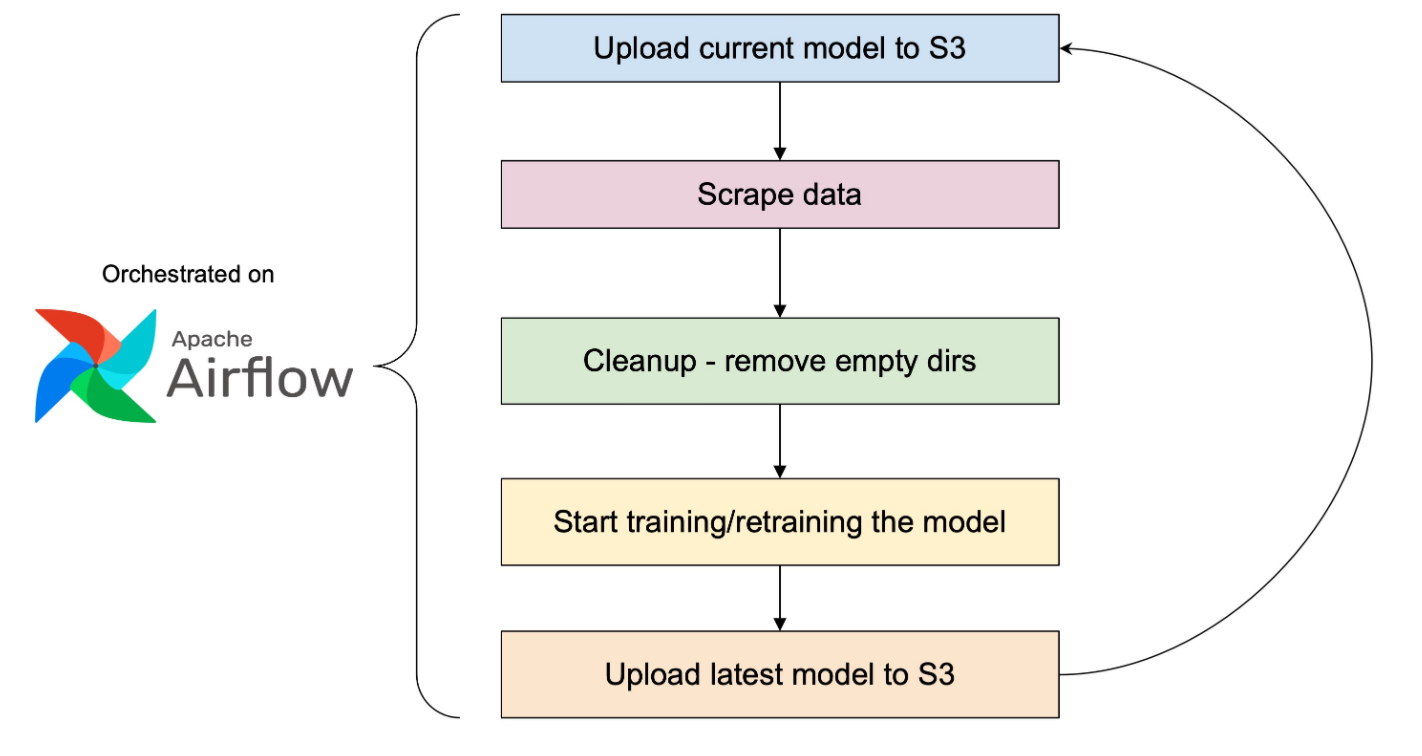


# **Test Cases**

1. After we login to the Airflow webserver on <http://localhost:8080/login/>, we can see the CNN-Training-Pipeline in the list of DAGs, we can check the graph view to a detailed view of the workflow tasks. We further trigger the workflow to start running the sequenced tasks:

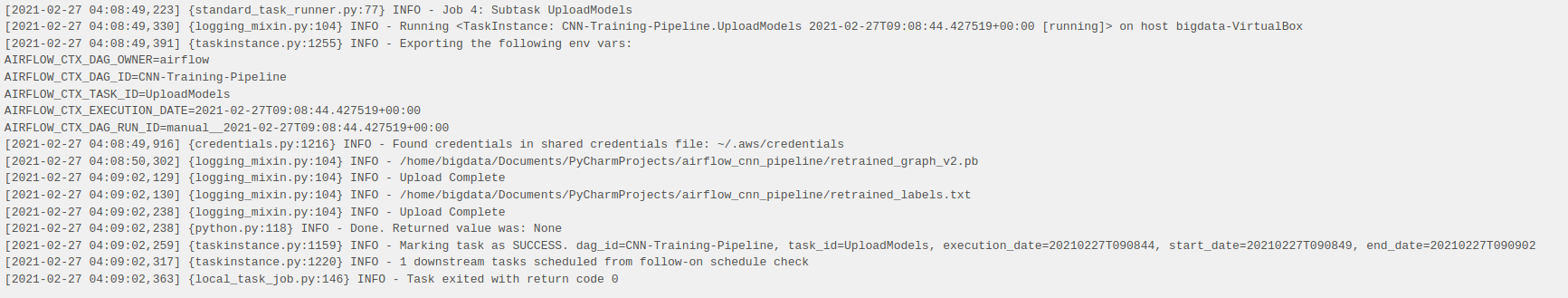


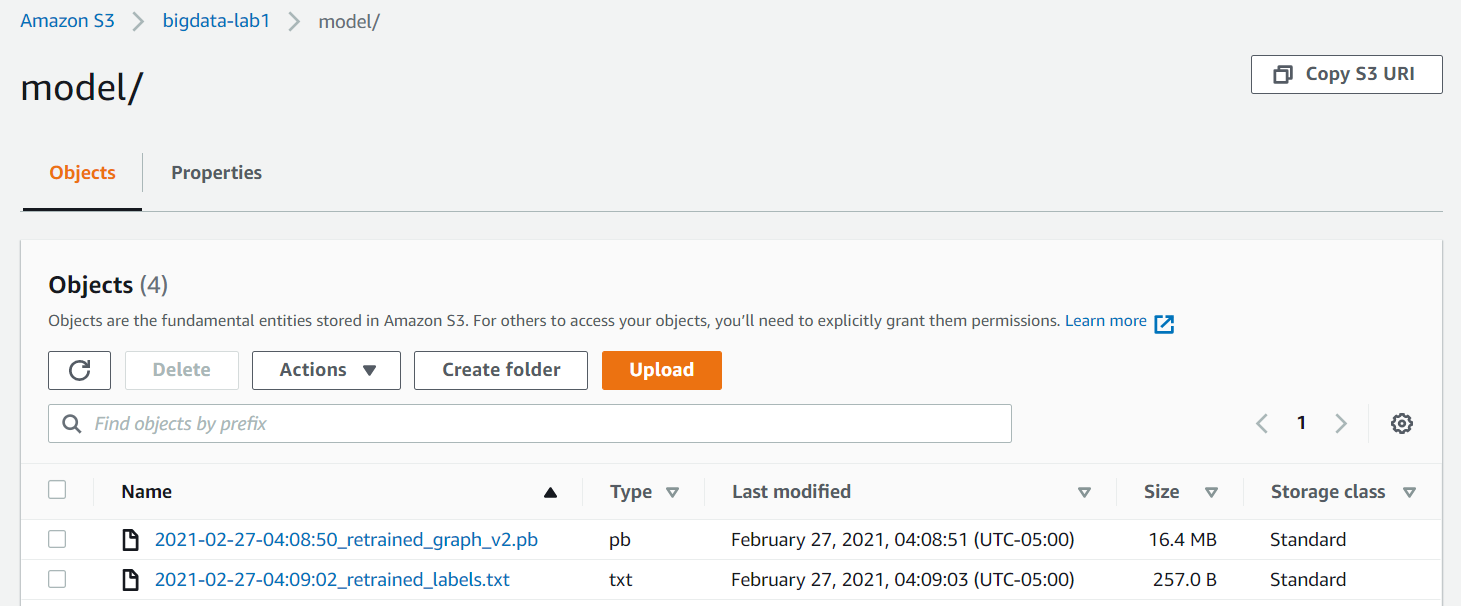
Airflow chains all the individual processes (tasks). The pipeline is scheduled to run at a predefined cadence and is constantly retraining the model using scraped data and continuously upload the trained graph and labels to S3



**Task 1: UploadModels**

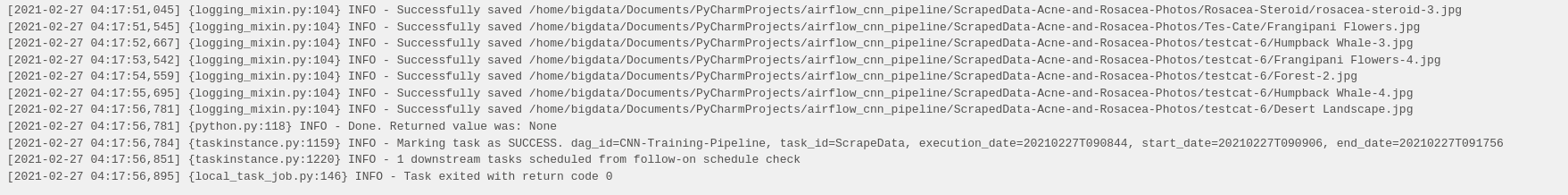
This task uploads the retrained graph (retrained\_graph\_v2.pb) and label (retrained\_labels.txt) from our system to AWS S3 Bucket mentioned in the bucket\_name using boto3 service inside /model folder

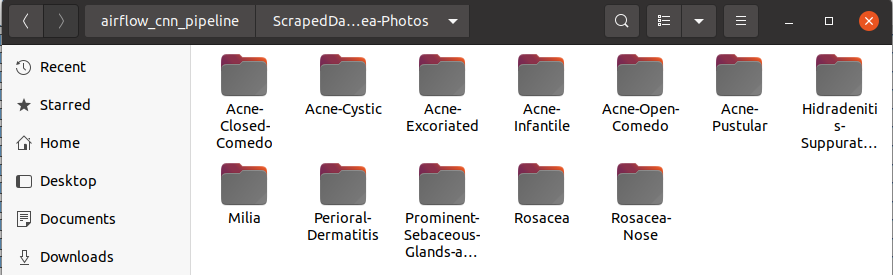




**Task 2: ScrapeData**

This task scrapes data from dermnet.com and downloads the scraped images in ScrapedData-Acne-and-Rosacea-Photos directory using BeautifulSoup in get\_data() function





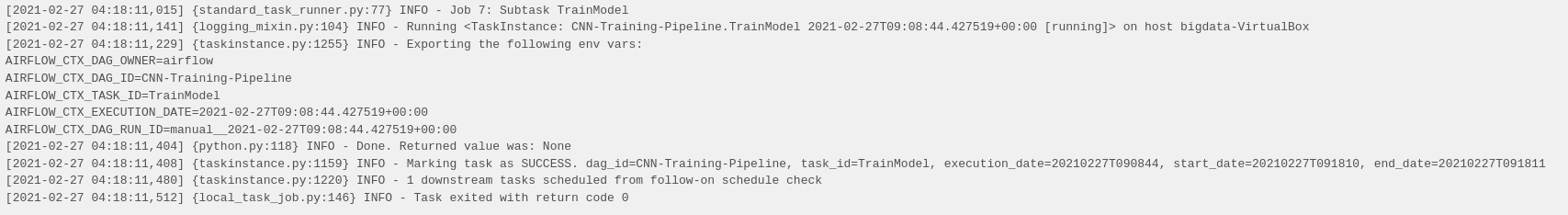
**Task 3: Cleanup**

This task cleans all the empty directories in ScrapedData-Acne-and-Rosacea-Photos folder which does have any images in it.



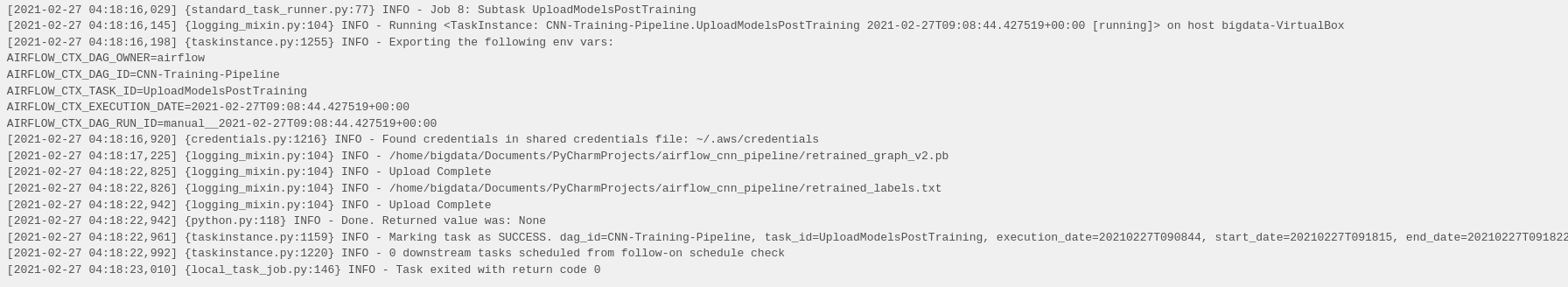
**Task 4: TrainModel**

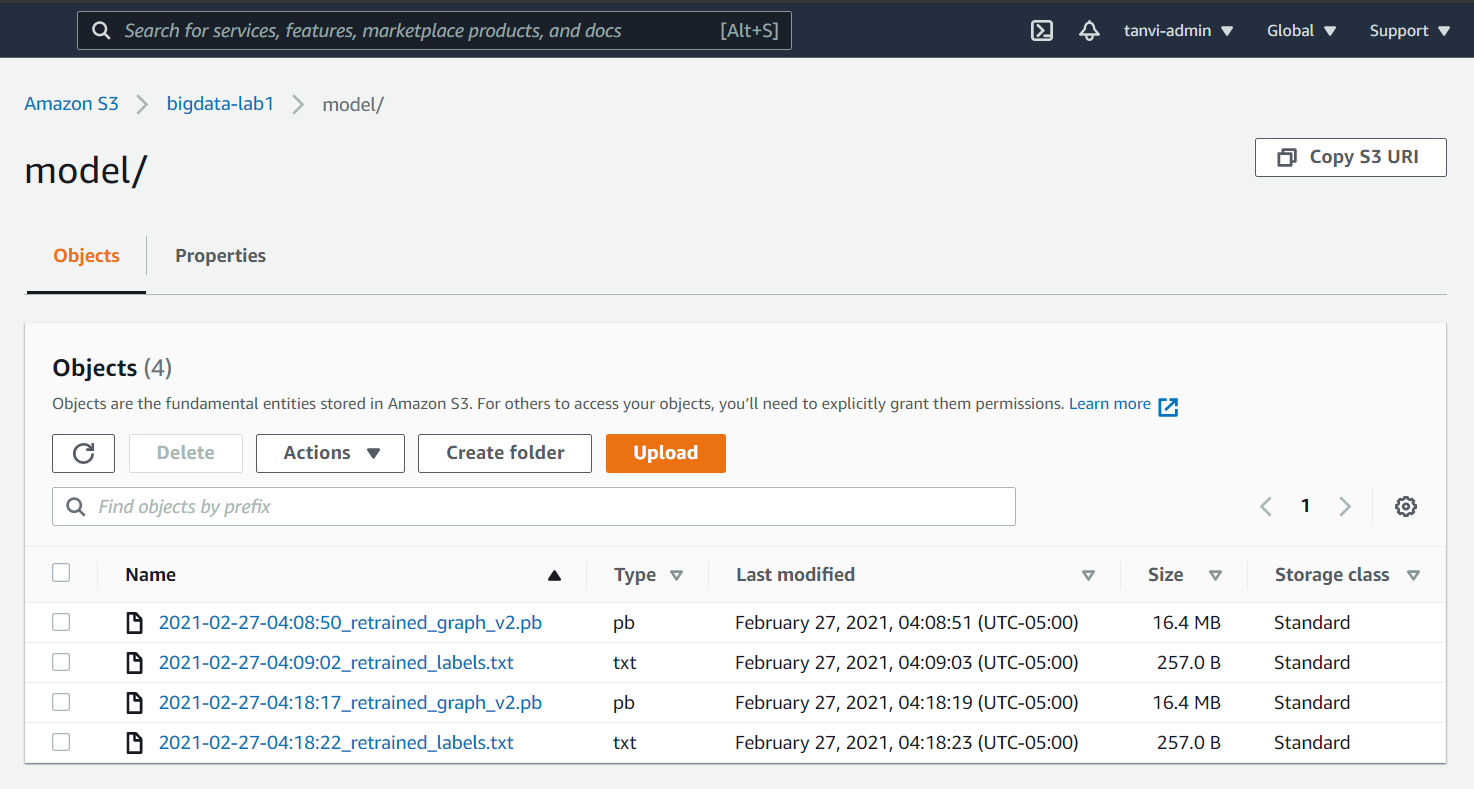
This task trains the retrained model uploaded in S3 with the newly scraped images from dermnet.com



**Task 5: UploadModelsPostTraining**

This task uploads the newly retrained model with scraped data to back to S3



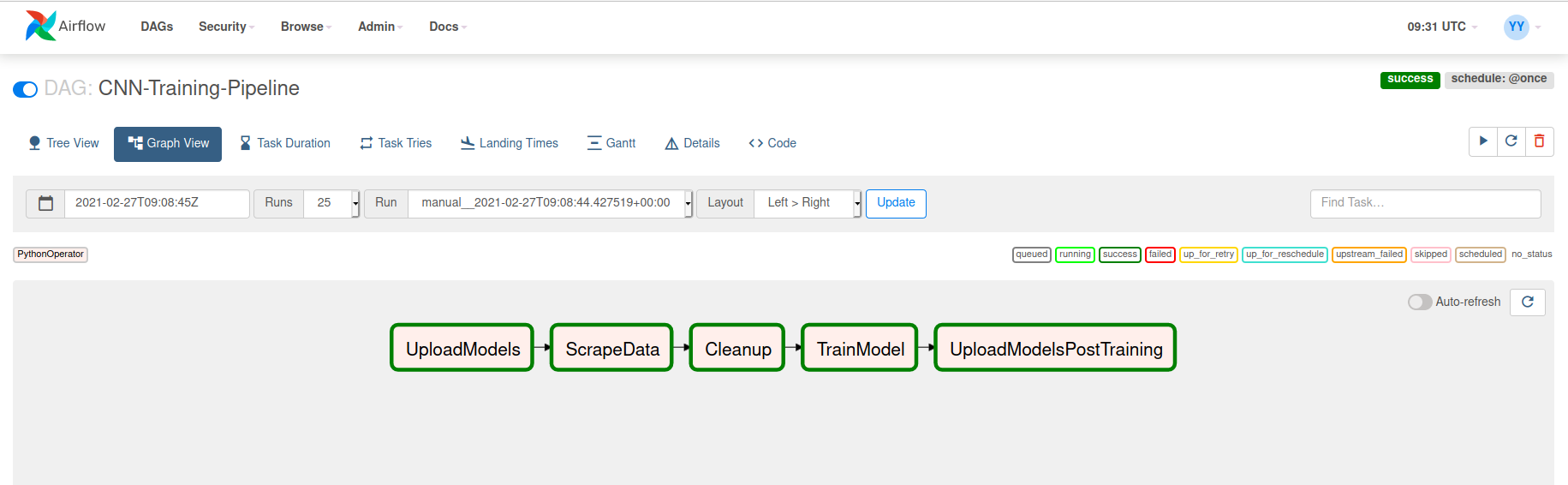


# 

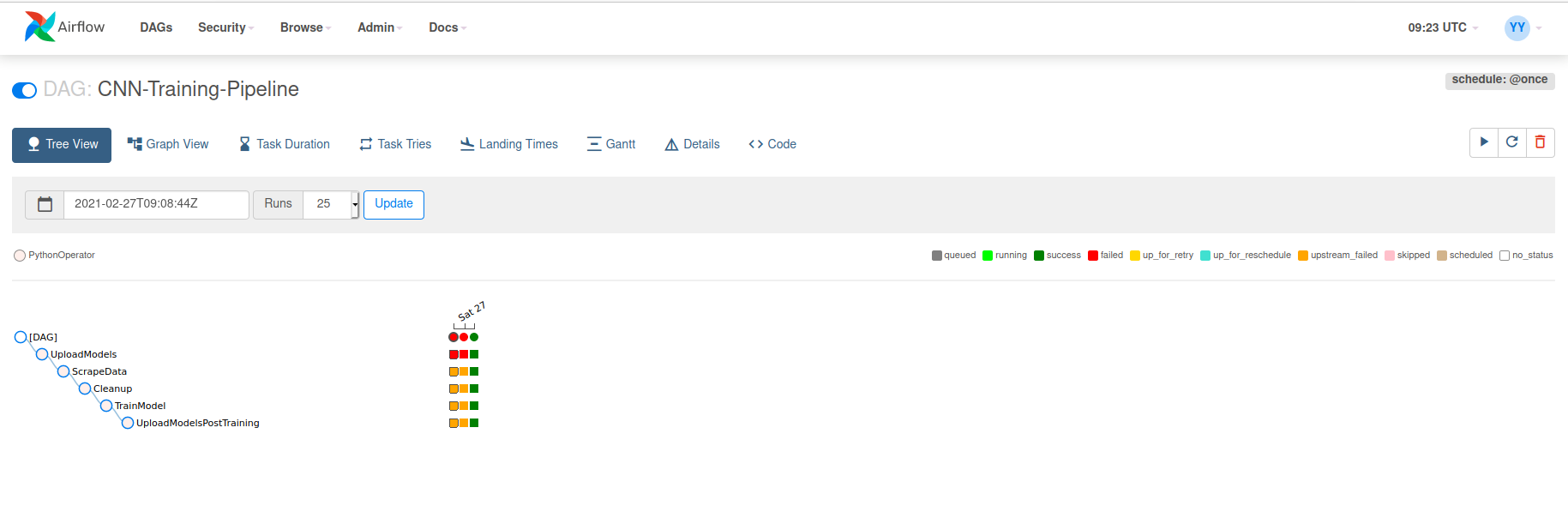
# **Results**

Once the airflow DAG is successfully completed individual tasks are highlighted in dark green color as below:

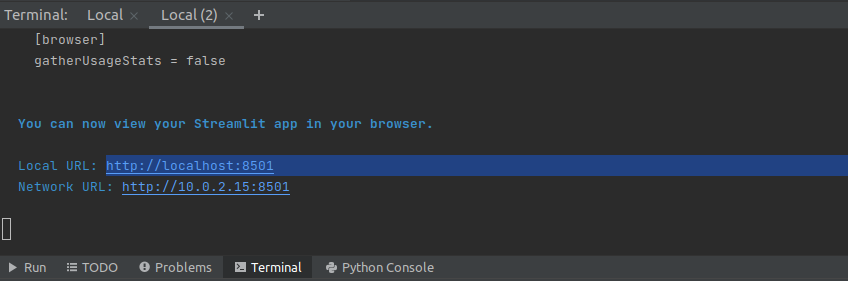
**Graph View:**



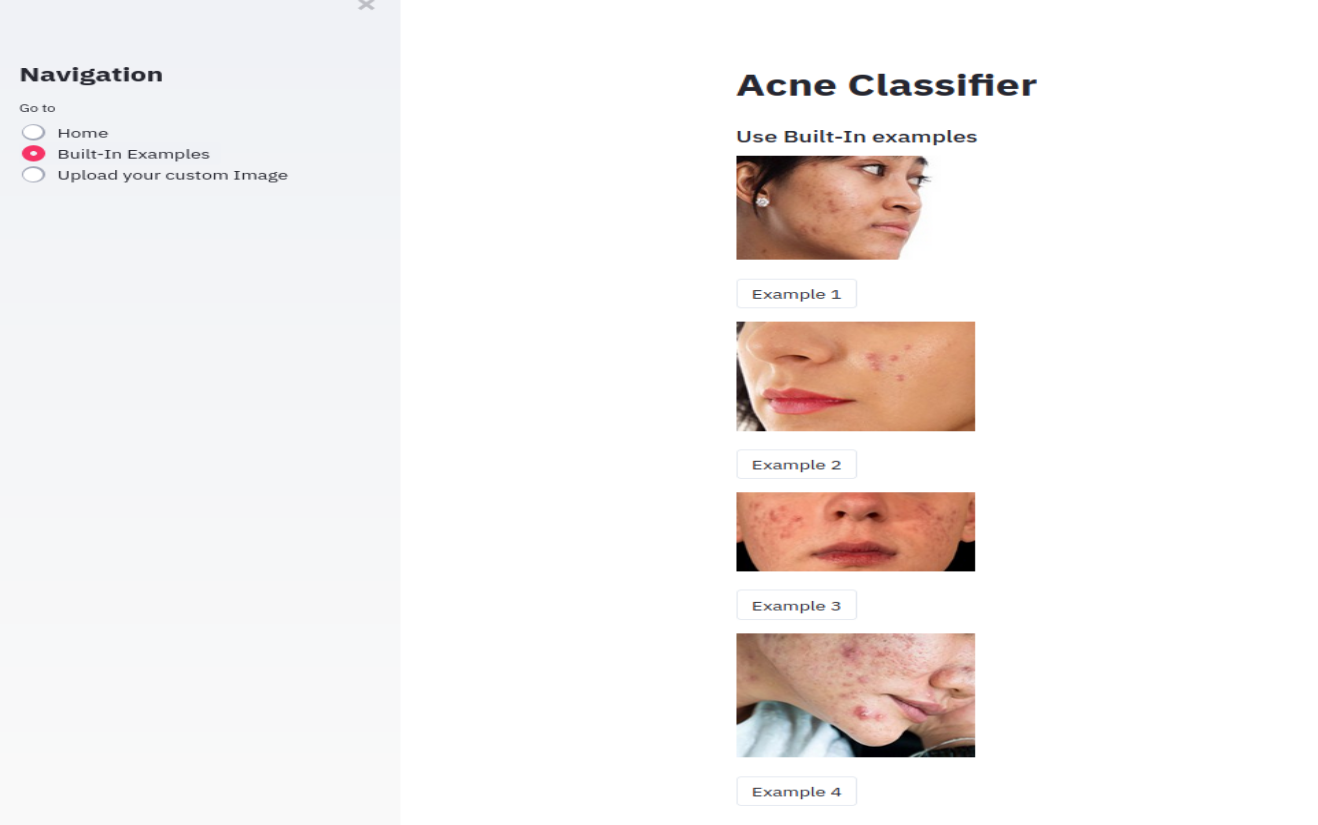
**Tree View:**

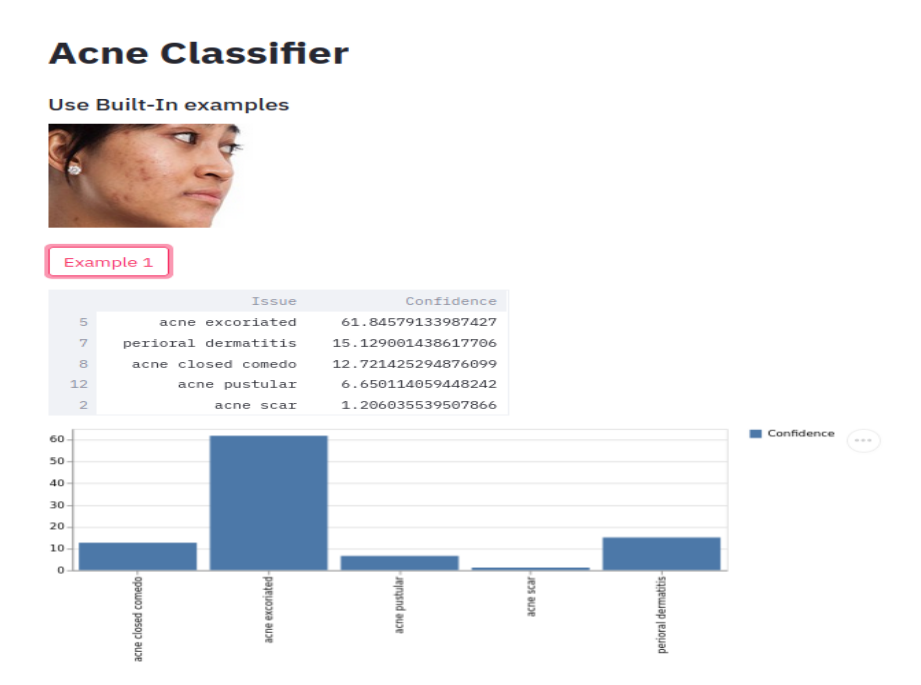


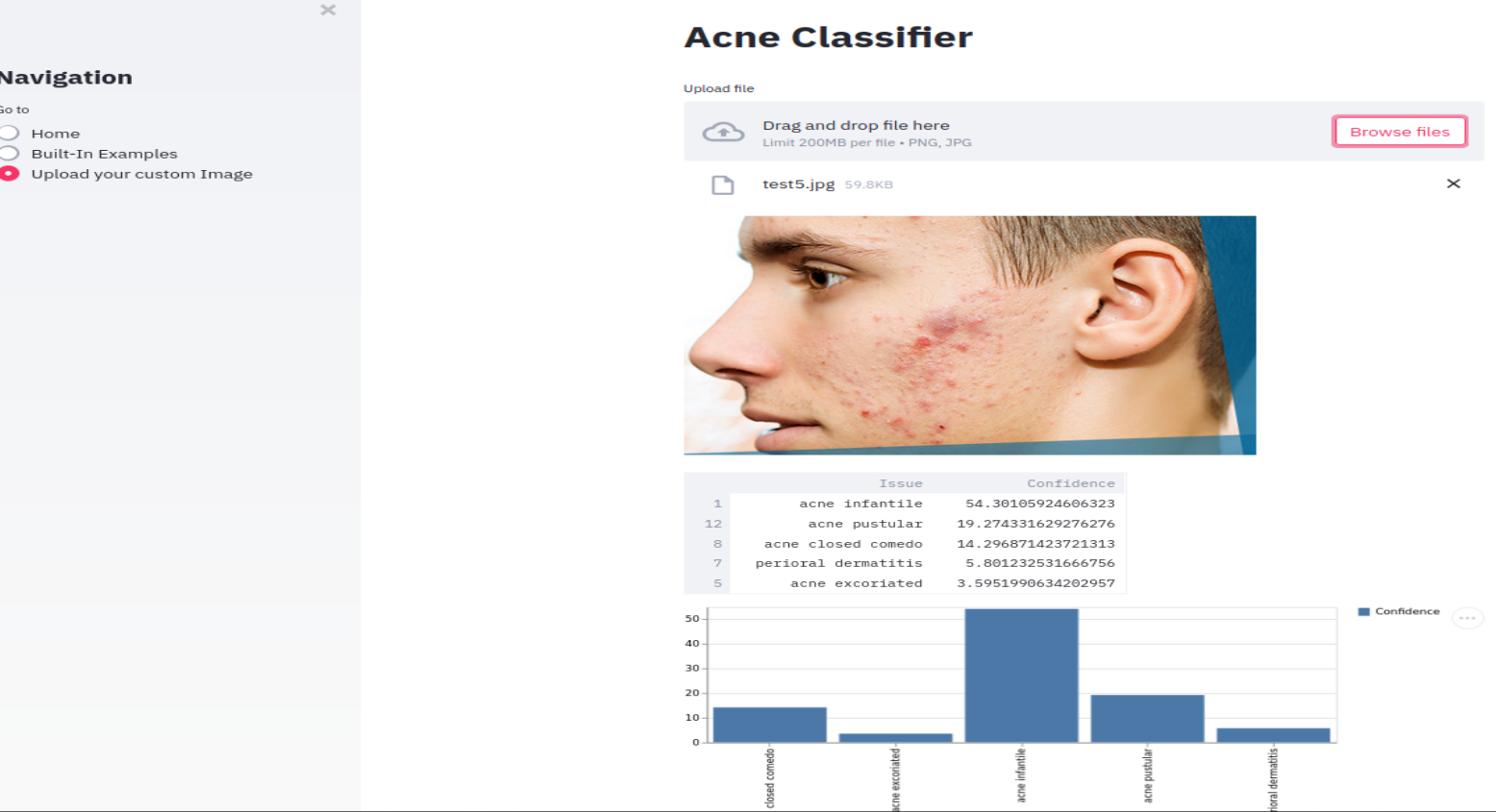
We can validate our retrained model by running the streamlit app (<http://localhost:8501>) which calculates the confidence score for its acne condition for each new uploaded images based on the retrained model with scraped images



# 







# Lessons Learned

1. Learnt how to orchestrate tasks in a pipeline using Apache Airflow
2. Crawling the data from web using BeautifulSoup
3. Using streamlit app as inference and validating retrained model for confident score for new images

# References

<https://airflow.apache.org/docs/apache-airflow/stable/index.html>

[https://medium.com/@dustinstansbury/understanding-apache-airflows-key-concepts-a96efed52b1a](https://airflow.apache.org/docs/apache-airflow/stable/index.html)

[https://towardsdatascience.com/getting-started-with-airflow-locally-and-remotely-d068df7fcb4](https://airflow.apache.org/docs/apache-airflow/stable/index.html)