

**University of Europe for Applied Sciences**Department of Data Science

**Data Engineering Project**

*Project Name: Analyzing Stock Market Trends with Automated Data Pipeline and AI-driven Insights*

**Project Report**

Ömer Onat Postacı

Tolga Özalevli

Ahmet Karakus

Mehmet Taşkesen

Jihye Kim

Kazim Ilhan

Ceren Berzenc

Savas Basar Trak

Hameed Rakhooy

Khaing Zin Thant

**Supervisor:** Sergey Burkov

# **Table of Contents**

[**Table of Contents 2**](#_q26wi3hkjdo0)

[**1. Introduction 3**](#_rapltmtrosfm)

[1.1 Description 3](#_6ek19tkp0jx7)

[1.2 Used Technologies / Libraries 4](#_fjq1ohotnzai)

[1.3 Environment Details 4](#_95oe12xwq68)

[1.3.1 What is Airflow? 4](#_3sjoiocru0r)

[1.3.2 What is a virtual machine? 5](#_nnob8z93arr2)

[1.3.3 How to run Airflow on a Google Virtual Machine? 5](#_yslqegreqw5a)

[Step 1: Getting on the Google Cloud 5](#_q0b16448g457)

[Step 2: Create a virtual machine 6](#_1wc0wqcnxylo)

[Step 3: Setting up the Environment 6](#_7j9sk77q0vt3)

[Step 4: Configuring Airflow 7](#_xhsls7mpq3i0)

[Step 6: Deploying workflows 7](#_j49gri2aqg7m)

[Step 7: Monitor and manage 8](#_xycgreqxmmbc)

[A Look at Google Looker 8](#_jpy8wen5efe2)

[**2 Flow of Executing the Pipeline(s) 8**](#_z2uavzpbl6vb)

[2.1 First Pipeline 8](#_lb1v1qc9r8l1)

[2.2 Second Pipeline 12](#_9s1muklhnja0)

[**3 Visualizing the Pipeline(s) Through Looker Studio 14**](#_icoqt3c7m084)

[3.1 First Pipeline 14](#_63bhx1ji3mbs)

[3.2 Second Pipeline 16](#_5vcatveh4bka)

[**4 Teamwork and Conclusion 16**](#_lqosbf55wnhc)

# 

# **1. Introduction**

In the ever-evolving landscape of financial markets, timely and data-driven insights are paramount for effective decision-making. Our project, "Analyzing Stock Market Trends with Automated Data Pipeline and AI-driven Insights" represents a concerted effort to harness the power of advanced technologies and streamline the process of extracting, analyzing, and visualizing stock market data. By integrating the Alpha Vantage API, Apache Airflow, Python, and Google Looker, our objective is to provide stakeholders with a comprehensive and user-friendly platform for understanding stock market trends, identifying patterns, and deriving actionable insights. Therefore, Analyzing Stock Market Trends with Automated Data project tries to analyze stock symbols, and provide insights and suggestions from GPT on a daily basis. In this direction, this report expresses the technologies used behind the development of the data pipeline and also, gives the complete flow for triggering the pipeline using a fully-deployed Apache Airflow Web User Interface.

## **1.1 Description**

The project uses an API provider for Stock Market Data, and specifically the Stock symbol that project utilizes is IBM; however, the project can be scaled to more generic structure of providing data pipeline for specified stock symbols. In the project, for the infrastructural development, Google Cloud Services were utilized such as Google Compute Engine (VM Instance) and BigQuery. Furthermore, for the orchestration of the tasks inside of the workflow of the data pipeline, Apache Airflow was utilized, and lastly, for the development of the pipeline, Python was utilized along with the libraries of Pandas, Airflow, Google Libraries, and necessary operators. The main aim of this data pipeline is to automate the process of analyzing the stock symbols and receiving the suggestions of a generative AI. Therefore, the data pipeline is designed to first, extract a data from a source of Stock Market Data API, then apply transformations, data analysis functions, and GPT request to transform the data coming from the source and store it into structured tables inside of a dataset of the BigQuery service of Google. After this automated process, a Looker Dashboard can be provided with great visuals and charts.

## **1.2 Used Technologies / Libraries**

There are precise technologies and libraries that were used to develop the pipeline. First of all, the libraries and technologies can be classified into three different categories respectively;

* Infrastructure Libraries & Technologies
* Google Compute Engine Virtual Machine Instance for Hosting the Airflow
* BigQuery for Data Warehouse
* Airflow for Workflow Orchestration
* Software & Development Libraries & Technologies
* Python
* Pandas
* Airflow
* Openai
* Additional Services
* Looker Studio

## **1.3 Environment Details**

### **1.3.1 What is Airflow?**

Airflow, developed by Apache, is an open source platform that helps manage complex computing workflows and data processing pipelines. Airflow's primary function is to create, schedule and monitor workflows in a programmable manner. It uses directed acyclic graphs (DAGs) for efficient task dependency management and scheduling. Each DAG outlines a sequence of tasks and their interrelationships, providing powerful and flexible control of data processing pipelines.

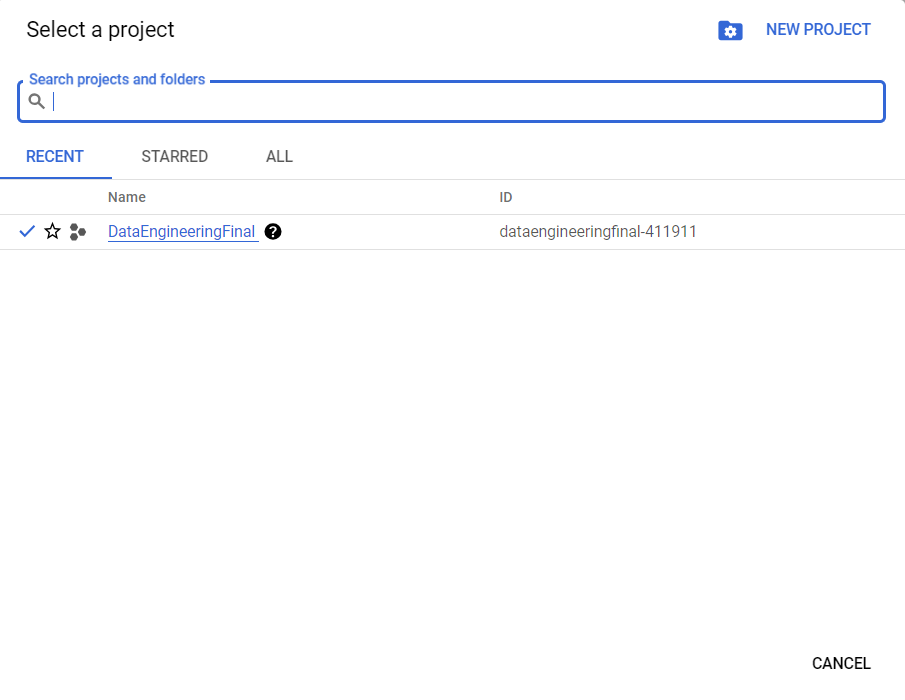
### **1.3.2 What is a virtual machine?**

A virtual machine, or VM, is a software-based replica of a physical computer system. It runs an operating system and applications just like a physical computer, but is maintained within a host system. Based on virtualisation technology, a single physical machine can support multiple VMs, each running in isolation from the others. This configuration is often used to test new applications, run different operating systems, consolidate servers and more, providing a scalable and efficient computing environment.

### **1.3.3 How to run Airflow on a Google Virtual Machine?**

There are several steps involved in using Apache Airflow on a Google Cloud Platform (GCP) virtual machine:

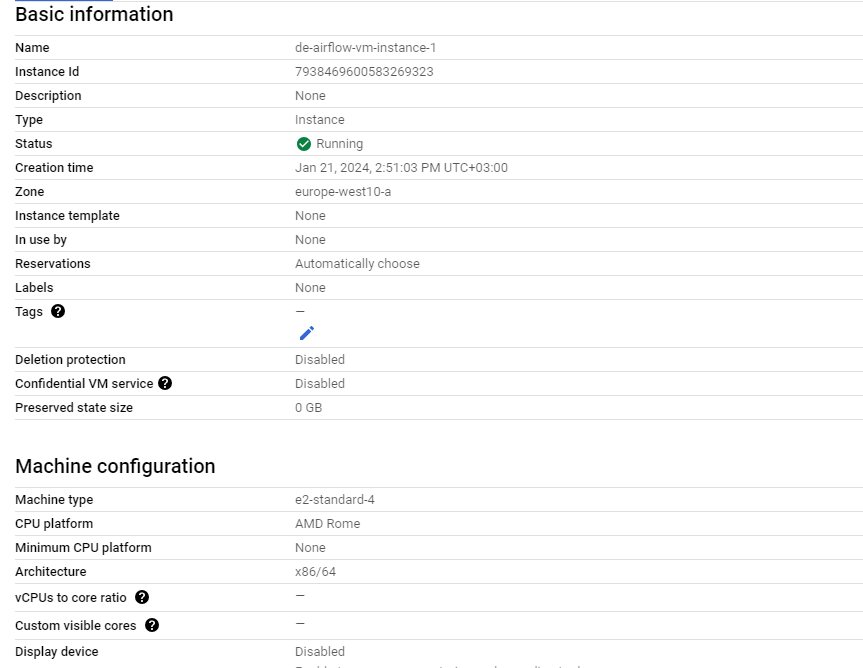
#### **Step 1: Getting on the Google Cloud**

First of all, a Google Cloud Platform account has been created, and in Google Cloud Platform, a new project has been created with the project name and project ID as follows:

#### 

#### **Step 2: Create a virtual machine**

After creating the project, the service for deploying the Apache Airflow was determined, and that service was Compute Machine service and specifically the Virtual Machine of Google Cloud Platform. Then, a Virtual Machine with OS Linux has been created with the necessary resources as follows:



#### 

#### 

#### 

#### 

#### **Step 3: Setting up the Environment**

Once the VM was up and running, using SSH connection, it was connected and then for the environment details of Apache Airflow, first Python was installed and then a Virtual environment was established using miniconda. For this purpose, the commands are as follows:

* **For installing Python:**

sudo apt update

sudo apt -y upgrade

sudo apt-get install wget

sudo apt install -y python3-pip

* **For establishing a Virtual Environment with Miniconda:**

mkdir -p ~/miniconda3

wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86\_64.sh -O ~/miniconda3/miniconda.sh

bash ~/miniconda3/miniconda.sh -b -u -p ~/miniconda3

rm -rf ~/miniconda3/miniconda.sh

~/miniconda3/bin/conda init bash

~/miniconda3/bin/conda init zsh

* **Installing the Airflow:**

AIRFLOW\_VERSION=2.8.0

PYTHON\_VERSION=3.9

CONSTRAINT\_URL="https://raw.githubusercontent.com/apache/airflow/constraints-${AIRFLOW\_VERSION}/constraints-${PYTHON\_VERSION}.txt"

pip install "apache-airflow[gcp]==${AIRFLOW\_VERSION}" --constraint "${CONSTRAINT\_URL}"

pip install pyspark==2.4.5

pip install cryptography==2.9.2

#### **Step 4: Configuring Airflow**

With Airflow installed, we set up our Airflow environment. This involved creating an airflow.cfg file and initializing the database used by Airflow. Here are the steps for initializing the Airflow:

airflow db init

airflow users create -r Admin -u <username> -p <password> -e <email> -f <first name> -l <last name>

#### **Step 6: Deploying workflows**

Next we created and deployed our DAGs to the Airflow instance running on the VM to see our workflows up and running.

#### **Step 7: Monitor and manage**

Finally, we used the Airflow web interface to monitor and manage our workflows.

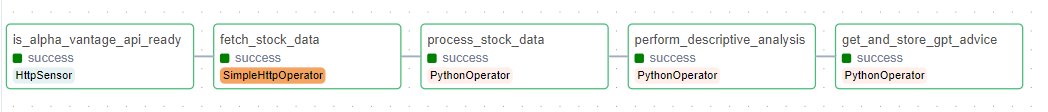
### **A Look at Google Looker**

During this project, we also worked with Google Looker. It's a business intelligence software and big data analytics platform. We found it great for exploring, analyzing, and sharing real-time business analytics. Looker is part of the Google Cloud Platform and offers some powerful data exploration capabilities. We especially liked how it integrates with SQL databases and makes it easy to build customized reports and dashboards. Plus, it's user-friendly, so everyone in our group was able to generate insights and make data-driven decisions without needing extensive technical know-how.

# **2 Flow of Executing the Pipeline(s)**

## **2.1 First Pipeline**

As Data Pipeline(s) are being represented and structured as Directed Acyclic Graphs (DAGs) in Apache Airflow, the following Figure demonstrates the DAG representation of the Data Pipeline of this project.



There are 5 different steps in the pipeline to extract the data from the data source, apply transformations and data analysis methods, and receive GPT’s advice, then store the structured data into BigQuery. Therefore, the necessary tasks for ensuring these functionalities are as follows:

**is\_alpha\_vantage\_api\_ready: HttpSensor**

A HttpSensor task that controls whether the given endpoint is ready to be applied to a GET function. This initial task is essential for the pipeline to ensure the flow and data consistency because when the data cannot be fetched correctly, then the remaining tasks would become meaningless.

**fetch\_stock\_data: SimpleHttpOperator**

A SimpleHttpSensor task that ensures the data fetch from the specified endpoint. Hereby, the fetched data can be utilized for the upcoming tasks. The prior task ensured us that the endpoint is ready to be fetched, and using this task, the data is fetched to be processed for the next task

**process\_stock\_data: PythonOperator**

A PythonOperator task that applies necessary transformations to the data received from the prior task. Using this transformations and analysis data is structured into a form that can be posted to the BigQuery. Using this task, data is structured for a specific table in the BigQuery and posted to BigQuery.

**perform\_descriptive\_analysis: PythonOperator**

A PythonOperator task that applies descriptive analysis to the stock data. After applying the descriptive analysis, this task structures the data to the form of the table in the BigQuery.

**get\_and\_store\_gpt\_advice: PythonOperator**

The last task which is a PythonOperator task that utilizes GPT API for using the data coming from the descriptive analysis and coming up with suggestions generated by GPT.

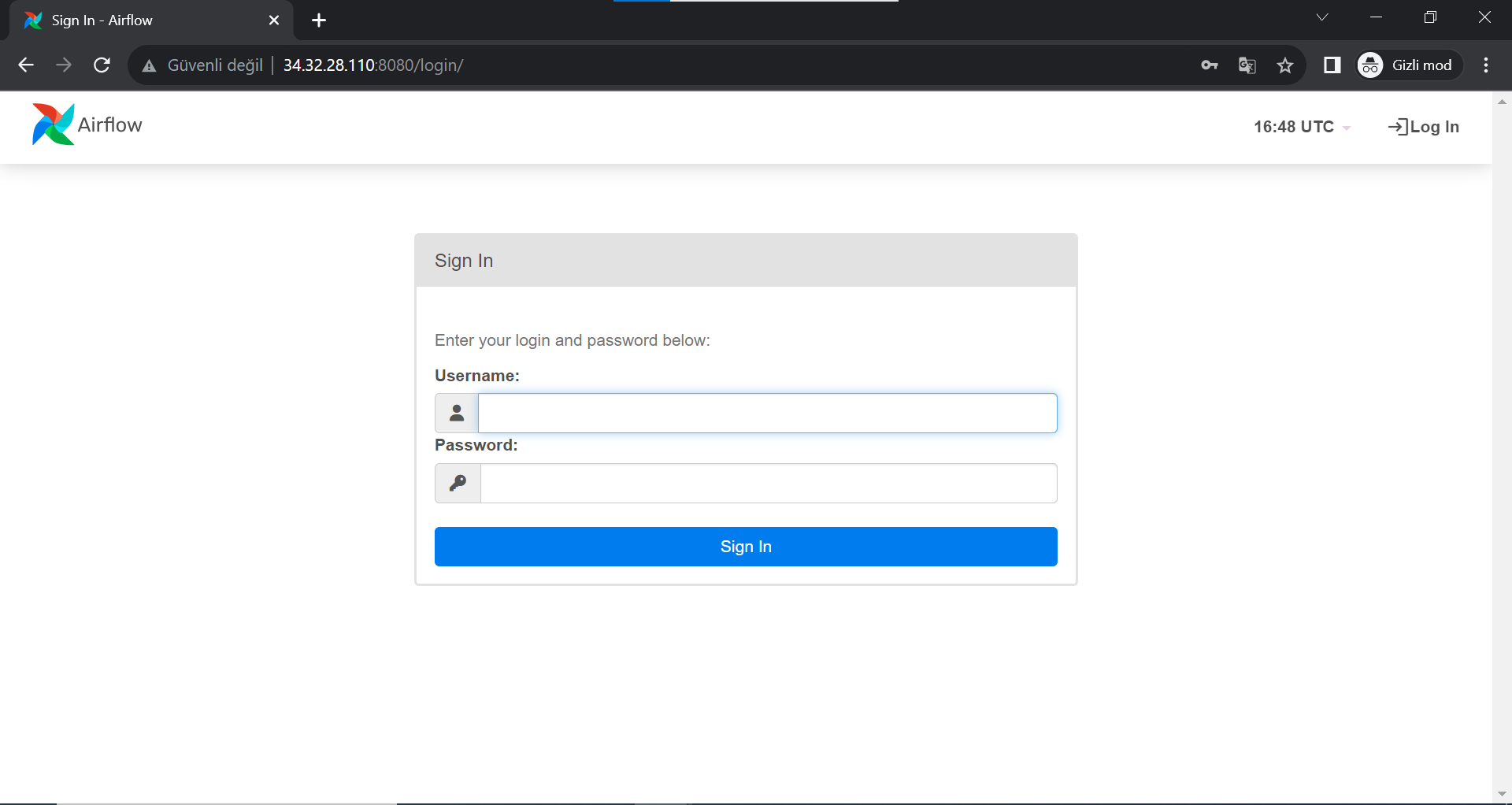
**STEPS FOR PIPELINE EXECUTION:**

The Data Pipeline developed by the team can be executed by anyone because the Airflow was deployed using Google Cloud Virtual Machine and Network Configurations were done accordingly. Therefore, anyone desiring to see the pipeline and flow can open the Airflow, then monitor and examine the pipeline. The following steps include the process of how to trigger the pipeline from outside:

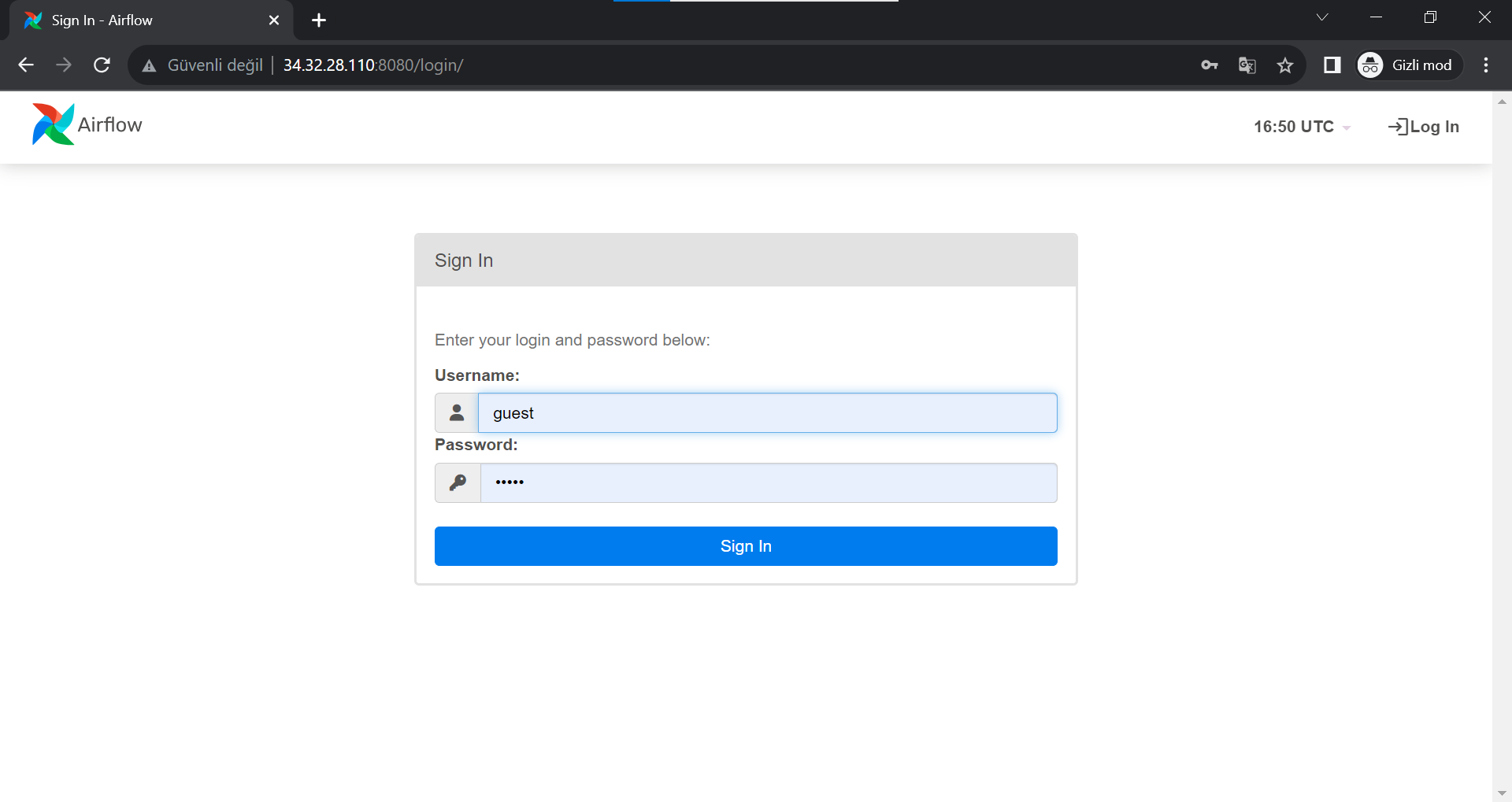
**Step 1**: Please enter the following URL to your browser for entering the Airflow Web User Interface: <http://34.32.28.110:8080/login/>

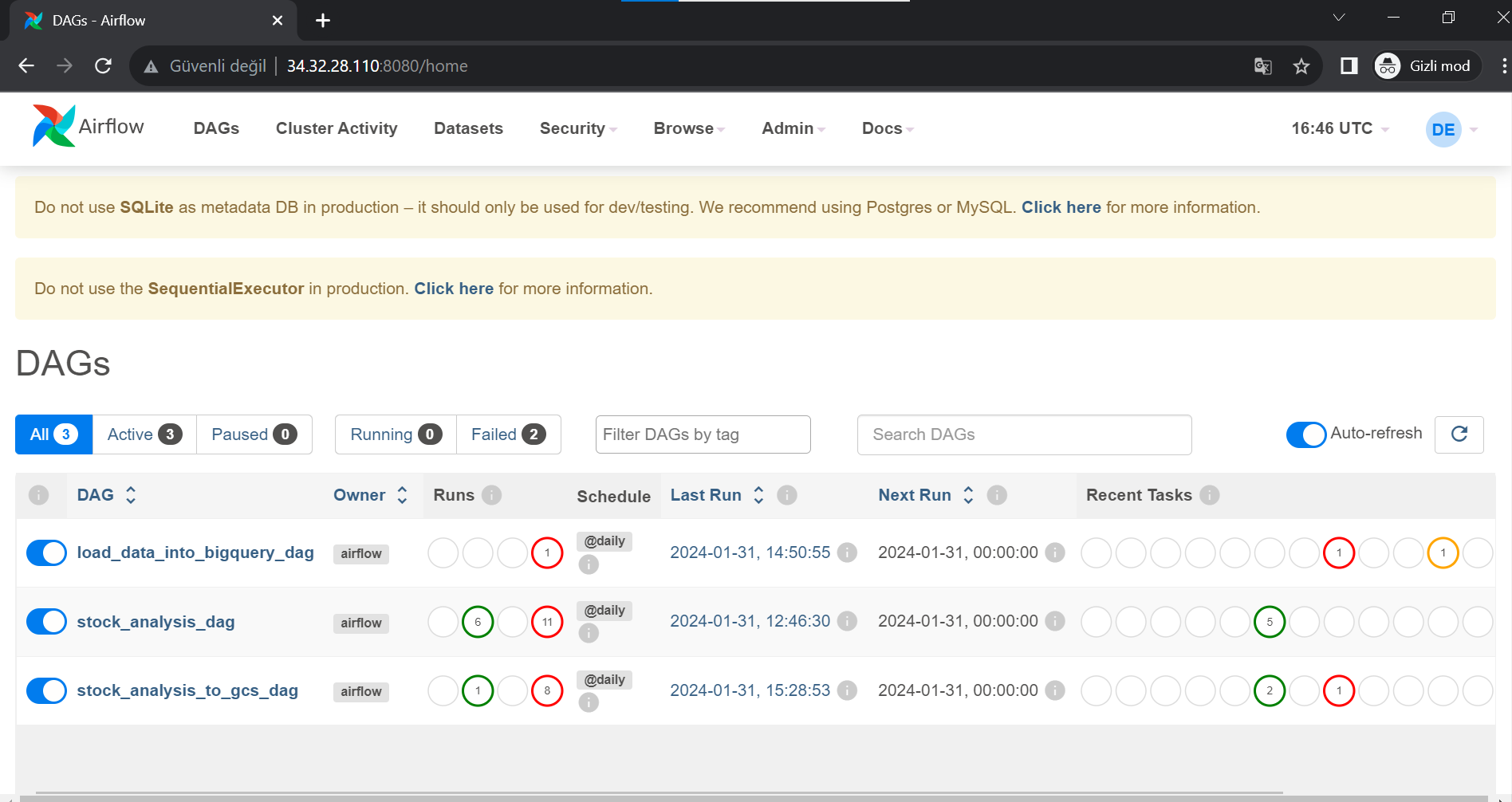
**PLEASE USE THE ABOVE LINK FOR OPENING THE WEB-UI OF APACHE AIRFLOW!!**

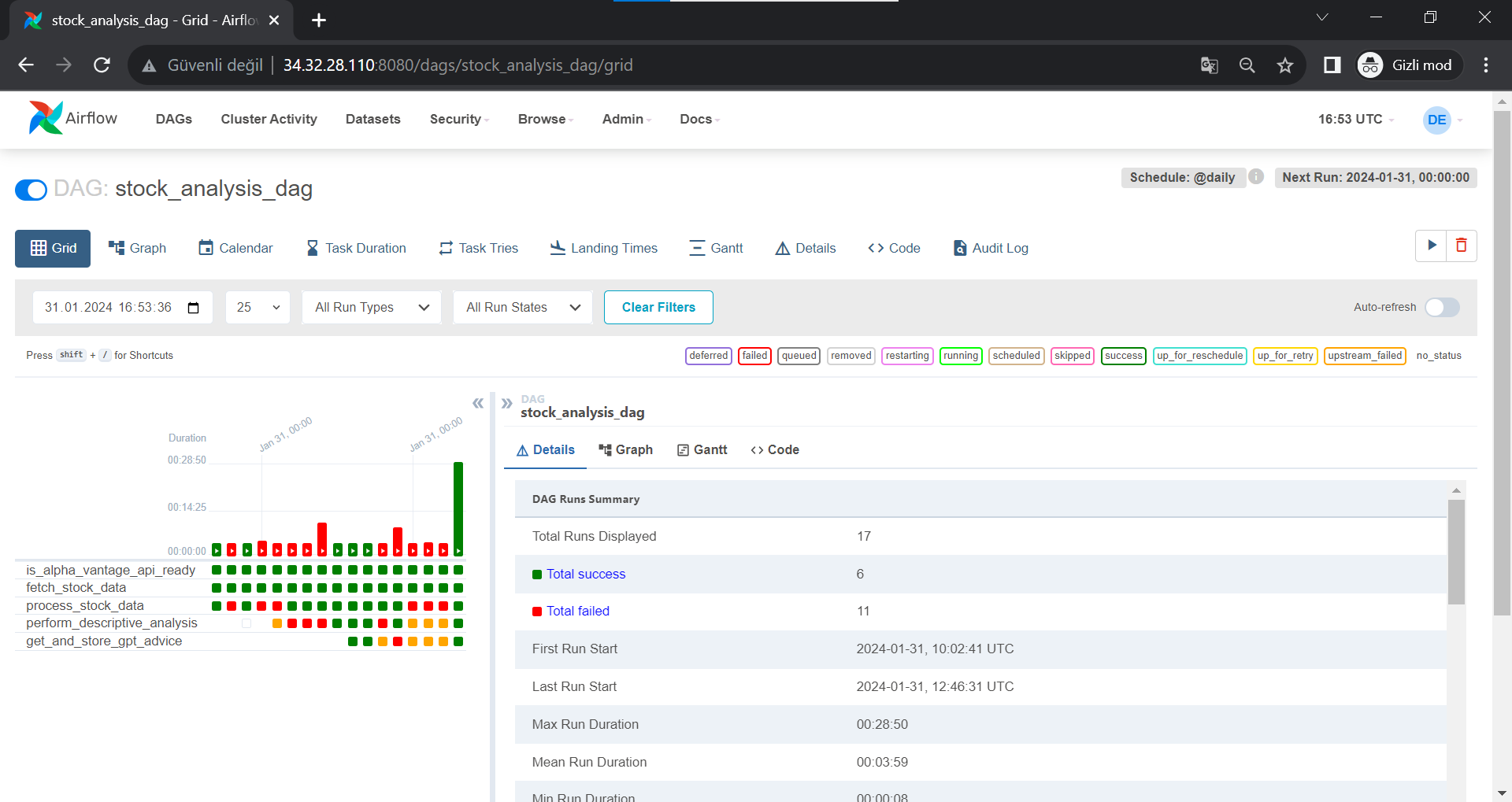
**Step 2**: The Login page of Apache Airflow will welcome you after entering the above URL, as following figure:



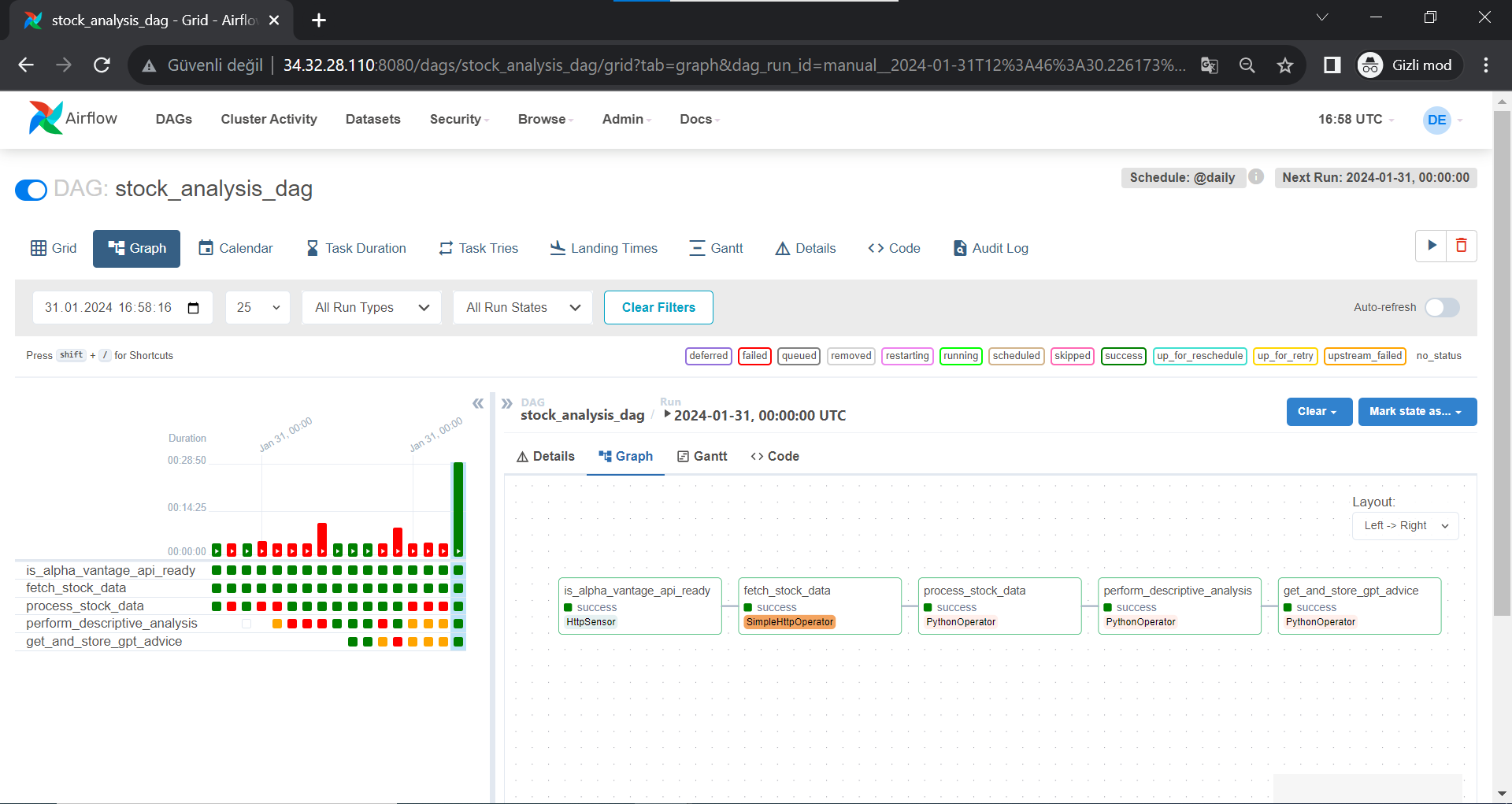
**Step 3**: Use the following credentials to authenticate yourself for visualizing the DAGS: ***username: guest*** *&* ***password: guest*** as following figure:



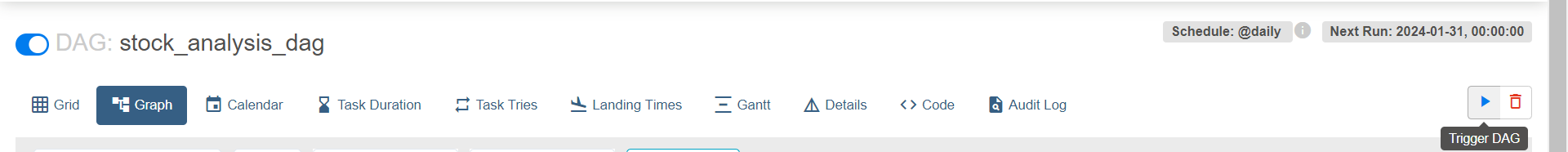
**Step 4**: After authenticating yourself, you will see 3 different DAGs as demonstrated in the following figure:

**Step 5:** The stock\_analysis\_dag is the Data Pipeline DAG representation of this project. Therefore, click on it, and the following page will appear after clicking on it:  
  


**Step 6:** You can also visualize it as a graph using the above Graph section as in the following figure:

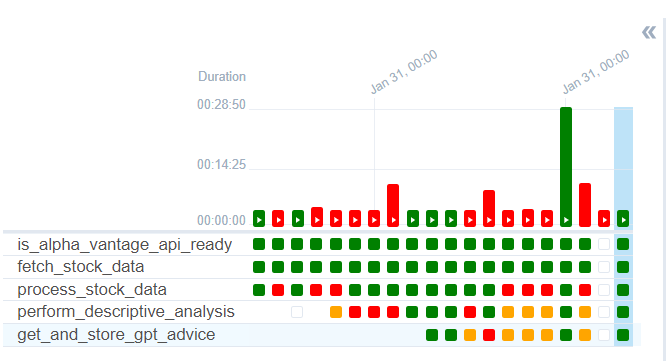


**Step 7:** As can be seen, there is a “Trigger DAG” button at the top right of the Airflow Web UI, and you can hit that button to trigger the data pipeline to execute:



After executing the DAG, you can monitor the left side details pane, and also using the graph view, you can view the Graph tasks and observe their execution status as well. After the pipeline execution, you can investigate the logs from the logs pane that is available at the top panel.

**Step 8:** After triggering the DAG, the left panel demonstrates the status for each task and if the left side seems green for each task, then the pipeline is executed without errors. The status seems as follows:

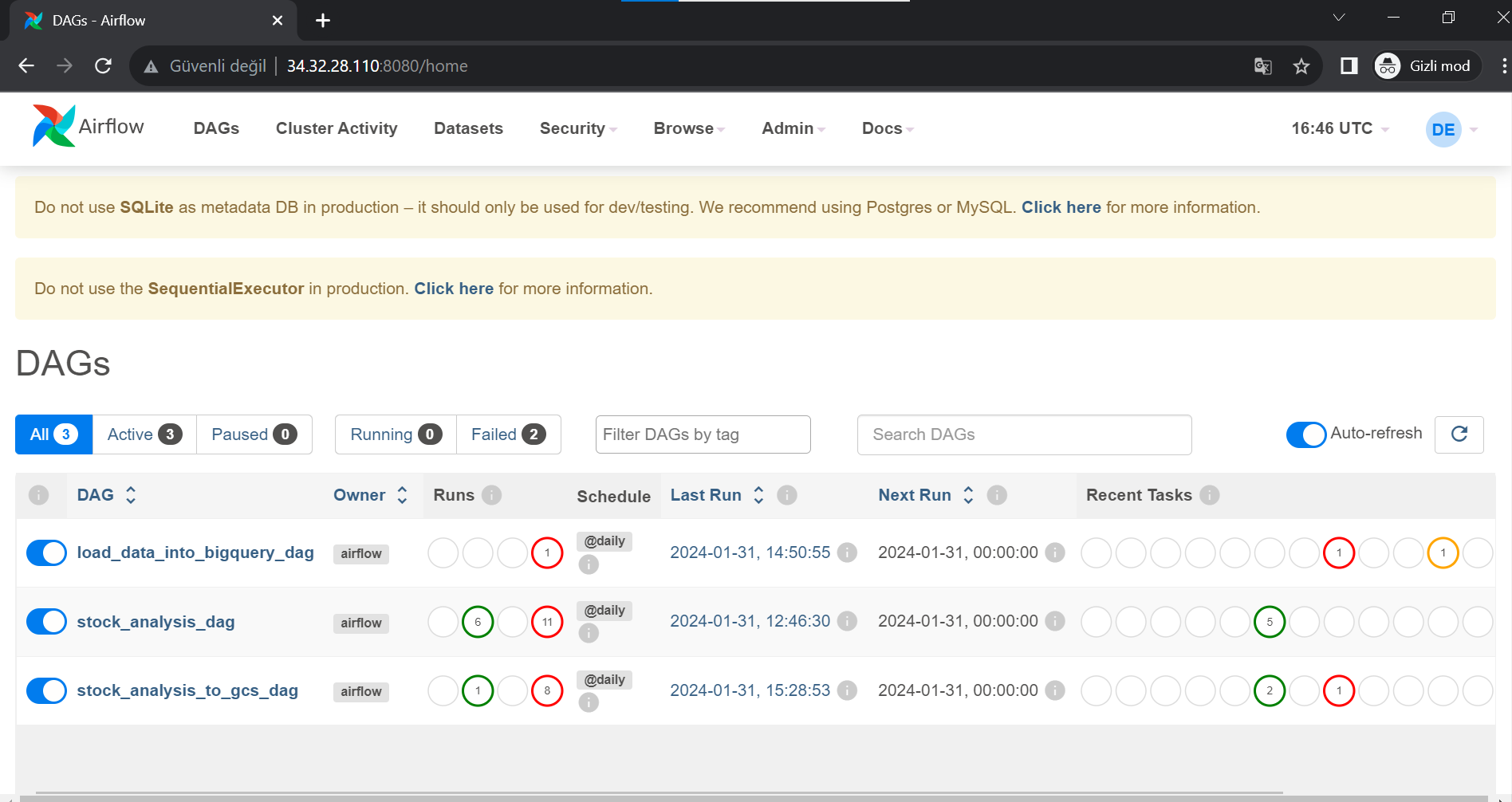


## 

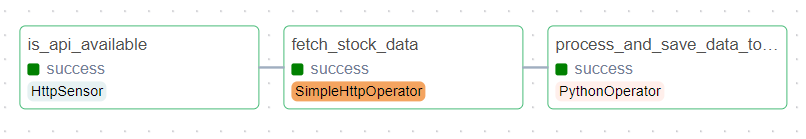
## 

## **2.2 Second Pipeline**

Apart from the first provided pipeline, we have developed a second pipeline that contains the processes of Data Ingestion to a Data Lake by applying transformations to provide a CSV file and then process the CSV file to store it in BigQuery. Therefore, there are 2 different DAGs for ensuring this functionality. In Apache Airflow’s Web Interface, there are 3 different DAGs as demonstrated in the following Figure:



The DAG with the name: **stock\_analysis\_to\_gcs\_dag** first fetches the data from the mentioned Data Source, and applies transformations to the data and stores the data as a structured form of CSV into the Google Cloud Storage Bucket. The following Figure demonstrates the Graph Representation of the DAG:



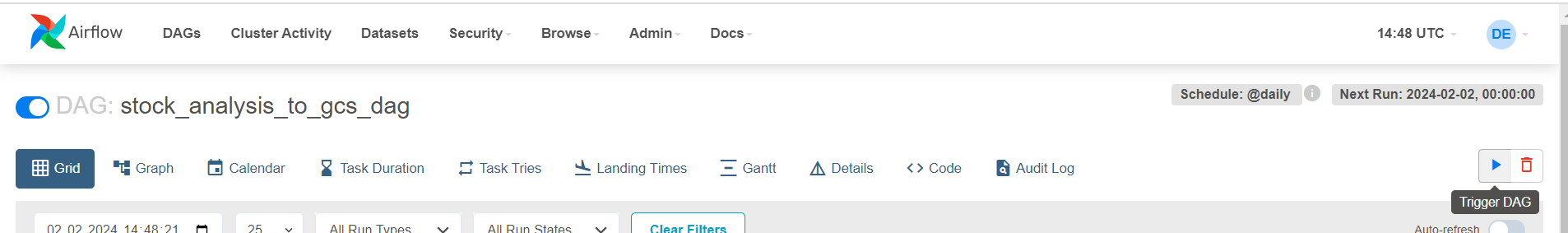
**is\_api\_available: HttpSensor**

A HttpSensor task that controls whether the given endpoint is ready to be applied to a GET function. This initial task is essential for the pipeline to ensure the flow and data consistency because when the data cannot be fetched correctly, then the remaining tasks would become meaningless.

**fetch\_stock\_data: SimpleHttpOperator**

A SimpleHttpSensor task that ensures the data fetch from the specified endpoint. Hereby, the fetched data can be utilized for the upcoming tasks. The prior task ensured us that the endpoint is ready to be fetched, and using this task, the data is fetched to be processed for the next task

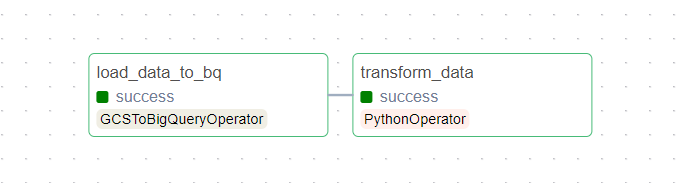
**Please first trigger this DAG to ingest the transformed data to the data lake as in the following Figure:**



**process\_and\_save\_data\_to\_gcs: PythonOperator**

A PythonOperator task that processes data with applying the transformations to make it more structured. Then, it saves the data as CSV file to the Google Cloud Storage Bucket.

**The DAG with the name: load\_data\_into\_bigquery\_dag** uses the data in the Google Cloud Storage Bucket and makes it a table in the BigQuery. The following is the Graph view of the DAG:

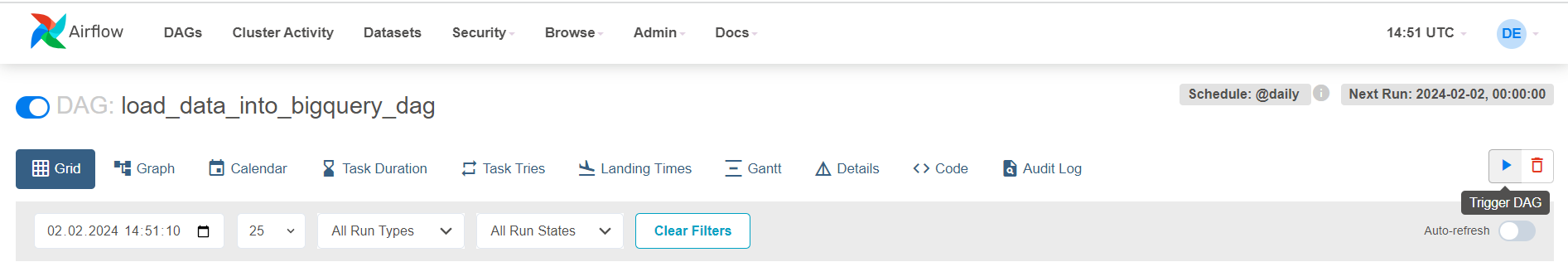


**load\_data\_to\_bq: GCSToBigQueryOperator**

This task loads the data in the Google Cloud Storage Bucket to the BigQuery with a structured way.

**transform\_data: PythonOperator**

This task applies the structuring Table Schema to the data coming from the Google Cloud Storage Bucket and then stores the data into the BigQuery Table.

**After triggering the ABOVE DAG, then please trigger the following DAG to save the structured data in data lake to the data warehouse which is BigQuery in our case:**

# 

# 

# **3 Visualizing the Pipeline(s) Through Looker Studio**

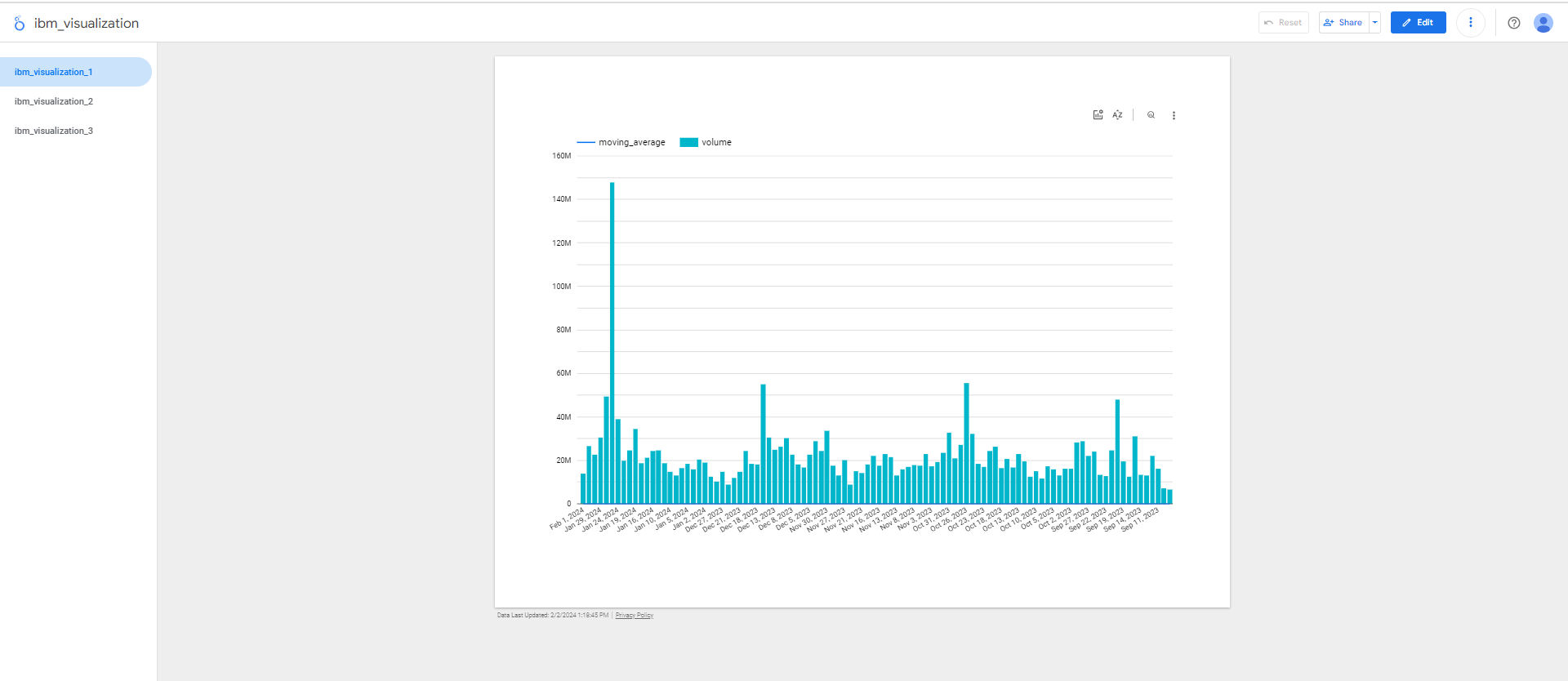
After executing the pipelines, the data can be visualized through Looker Studio. There are two different Looker Studio visualizations for two different Pipelines provided in this report.

## **3.1 First Pipeline**

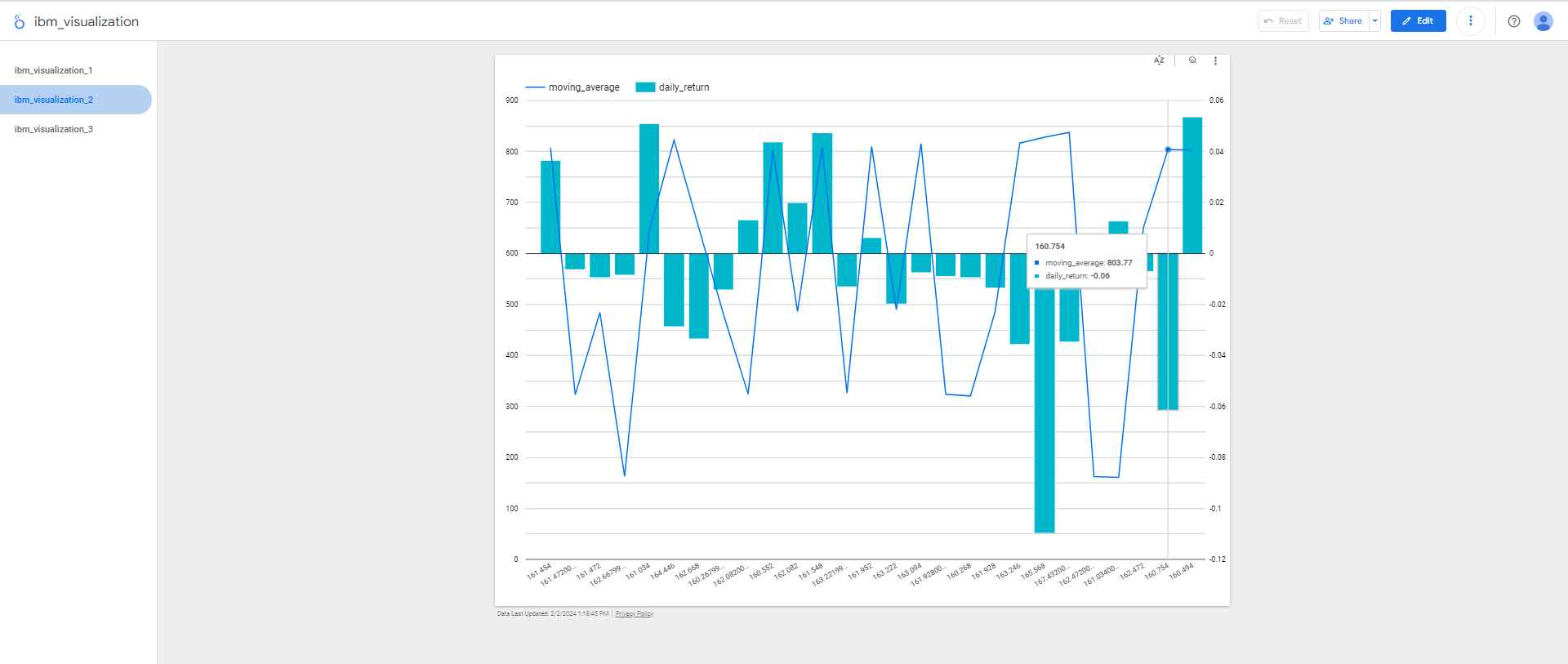
As mentioned earlier, there are two different pipelines, and for the first pipeline, the following link is the visualization in the Google Looker Studio:

<https://lookerstudio.google.com/reporting/b3573932-ba41-4923-8484-7f9e03abc06c>

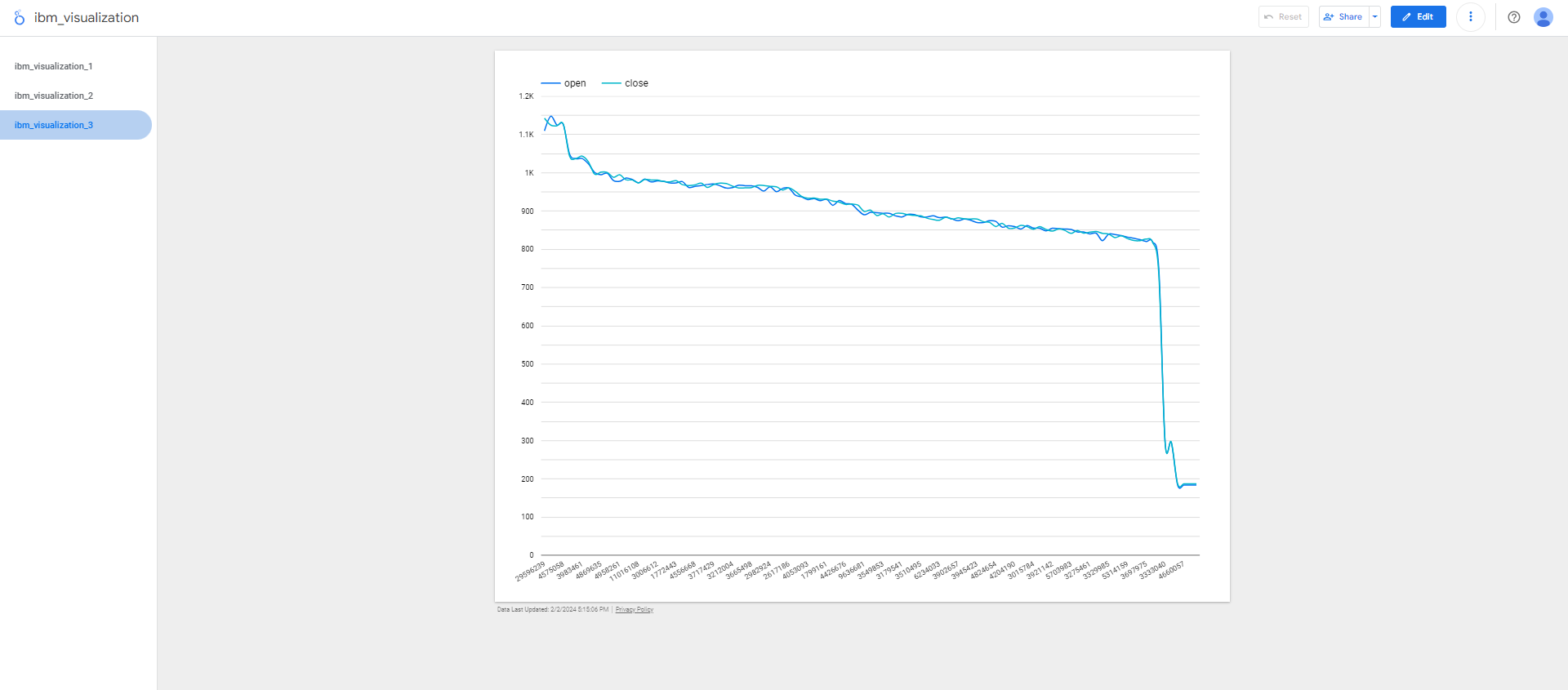
After clicking on this URL, the Looker Studio will be visible and there are three different visualizations for the data gathered and structured through the pipeline as follows:



As demonstrated the first visualization demonstrates the volume and the calculated moving average of the stock.

The following is the second visualization:

As demonstrated, this visualization provides the relation of the calculated moving averages and calculated daily returns for each day. Lastly, there is a third visualization for the data as follows:



## 

## **3.2 Second Pipeline**

The above figure demonstrates the general structured data along with visualizations of graphs of volatility, volume, and record count along with close and calculated moving averages graph. With each pipeline trigger and execution, the graphs will be updated accordingly.

# **4 Teamwork and Conclusion**

As a team, this project was developed with the following jıb divisions; however, each division team:

|  |  |
| --- | --- |
| **Division** | **Team** |
| Pipeline Development | Ömer Onat Postacı, Hameed Rahkooy |
| Infrastructure Arrangement | Ömer Onat Postacı, Hameed Rahkooy, Savas Basar Trak, Tolga Ozalevli, Jihye Kim, Khaing Zin Thant, Ceren Berzenc |
| Report & Search | Ömer Onat Postacı, Jihye Kim, Khaing Zin Thant, Ceren Berzenc, Ahmet Karakus, Mehmet Taskesen, Kazım Ilhan |
| Visualization | Hameed Rahkooy, Tolga Ozalevli, Ahmet Karakus |

In conclusion, this project was developed to provide different pipelines to analyze IBM Stock for it’s daily values and volumes. Using the power of Gen-AI and data analysis, this project provided data insights using the Data Engineering technologies of Google Cloud Services like Storage Bucket, BigQuery, and Virtual Machine, also Apache Airflow, and Python. In this documentation, it can be found how to trigger a DAG in the deployed Airflow WEB-UI and also the code was provided along with the ZIP file. Lastly, this report also includes the GitHub Repository of the code as follows:

**GITHUB REPOSITORY:** [**https://github.com/onatpostaci/DataEngProjectUE**](https://github.com/onatpostaci/DataEngProjectUE)