



DONE BY: IKRAM EL-HAJRI

16/10/2023

I. Introduction

1. What is Apache Airflow?

Apache Airflow is a platform to **programmatically author, schedule, and monitor workflows**. It allows us to define a Directed Acyclic Graph (DAG) to represent our workflow, where each node in the graph represents a task. It provides a powerful and flexible way to manage and execute complex workflows.



2. What is Apache Airflow Used For?

Apache Airflow is primarily used for **orchestrating and automating workflows**. Data engineers and data scientists often use it for tasks such as ETL (Extract, Transform, Load) processes, data pipeline orchestration, and job scheduling.

3. Why Should Data Engineers/Scientists Learn Apache Airflow?

1. **DAG (Directed Acyclic Graph):** Apache Airflow uses the concept of DAGs to represent workflows, making it easier to visualize and manage complex data pipelines.

2. **Pipeline Orchestration:** Airflow provides a centralized platform to orchestrate data pipelines, ensuring data flows seamlessly and reliably.

3. **Benefits of Airflow:**

- **Dynamic Workflow Changes:** You can easily modify workflows, add new tasks, or change task dependencies without rewriting the entire pipeline.
- **Extensible:** Airflow supports custom operators, making it adaptable to different use cases.
- **Flexible Scheduling:** You can set up complex schedules based on time, events, or external triggers.

4. Misunderstandings I had at first about Apache Airflow:

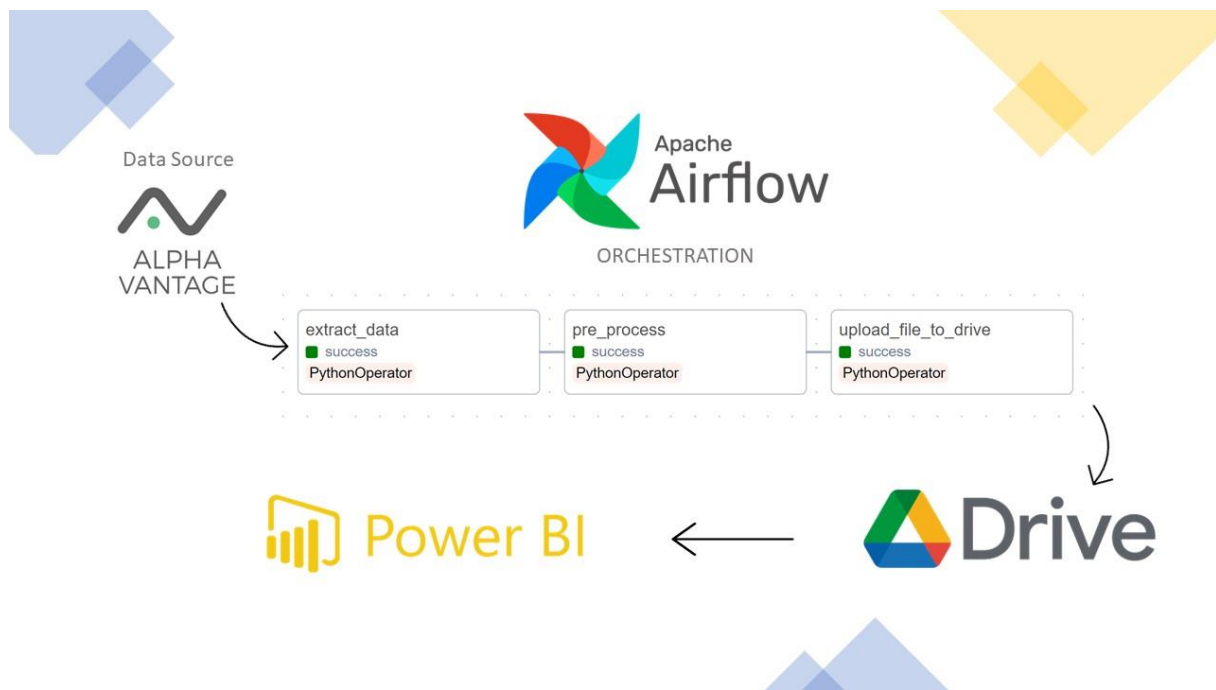
- **Not an ETL Tool:** While it's often used in ETL (Extract, Transform, Load) processes, Airflow isn't an ETL tool itself. It's a workflow orchestrator that can manage ETL tasks.
- **Not a Data Storage System:** Apache Airflow doesn't store data. It manages the execution and scheduling of tasks in our data pipeline but doesn't store the data itself.
- **Not Only for Big Data:** Although it's beneficial for Big Data, Airflow is not exclusive to large-scale projects. It's valuable for any workflow automation.
- **Not Limited to Python:** While many use cases involve Python, Apache Airflow supports various languages and can integrate with non-Python tools.
- **Not Only for Scheduled Jobs:** Airflow isn't just for running tasks on schedules. It supports event-driven and manual triggering too.

5. Operations in Apache Airflow

- **Task Execution:** Each task within a DAG is executed by an operator. The execution is managed by the Airflow scheduler.
- **Resource Management:** Airflow allows you to allocate resources and specify resource requirements for tasks, ensuring efficient utilization of resources.
- **Job Monitoring:** You can monitor the progress of your workflows using the Airflow UI, which provides insights into the status of tasks and historical runs.
- **Logs:** Airflow logs provide detailed information on task execution, making it easier to troubleshoot issues.
- **Apache Airflow Scheduler:** The scheduler is responsible for queuing and executing tasks according to the defined schedule.

II. Project Description:

In this project, I set up a data pipeline using Apache Airflow to automate the extraction, preprocessing, loading and then analysis of daily stock data for IBM.



1. Data Extraction:

I used the Alpha Vantage API to fetch daily stock data for IBM.

ALPHA VANTAGEMenu

Stock Market API, *Reimagined*

- ▶ Realtime & historical stock market data APIs
- ▶ Forex, commodity & crypto data feeds
- ▶ 60+ technical & economic indicators
- ▶ Market news API & sentiments
- ▶ Global coverage

[STOCK MARKET API](#)[GLOBAL NEWS API](#)


[GET FREE API KEY](#)

2. Data Preprocessing:








- To make the data suitable for analysis, I performed several preprocessing steps:
 - o Converted the timestamp column to a datetime format.
 - o Calculated a 10-period moving average for better trend analysis.
- The preprocessed data was saved as "processed_stock_data.csv."


Uploading to Google Drive:


To authenticate the application and enable it to access and manipulate the Google Drive Folder, a service account key file was created.

 Start your free trial with \$300 in credit. Don't worry – you won't be charged if you run out of credit. [Learn more](#)

DISMISS [START FREE](#)

  Google Drive API     

 Product details



Google Drive API

[Google Enterprise API](#)

Create and manage resources in Google Drive.

[ENABLE](#) [TRY THIS API](#)

[OVERVIEW](#) [DOCUMENTATION](#) [SUPPORT](#) [RELATED PRODUCTS](#)

Overview

With the Google Drive API, you can access resources from Google Drive to

Additional details

Google Cloud

IAM and admin

my app

DETAILS

Private key saved to your computer

optical-sight-402123-ba62f90c9f2e.json allows access to your cloud resources, so store it securely. [Learn more best practices](#)

CLOSE

ADD KEY ▾

Type	Status	Key	Key creation date
	Active	1db506cec1ad418eb100185875e4ebe77e1bf6f6	15 Oct 2023
	Active	ba62f90c9f2e77678704eefd42490862caf4ba42	16 Oct 2023

- The necessary libraries had to be installed.

```
root@Ikram:~/airflow# pip install --upgrade google-api-python-client google-auth-http lib2 google-auth-oauthlib
Collecting google-api-python-client
  Downloading google_api_python_client-2.103.0-py2.py3-none-any.whl (12.6 MB)
    12.6/12.6 MB 145.1 kB/s eta 0:00:00Collecting google-auth-http lib2
  Downloading google_auth_httplib2-0.1.1-py2.py3-none-any.whl (9.3 kB)
Collecting google-auth-oauthlib
  Downloading google_auth_oauthlib-1.1.0-py2.py3-none-any.whl (19 kB)
Requirement already satisfied: http lib2<1.dev0, >=0.15.0 in /usr/lib/python3/dist-packages (from google-api-python-client) (0.20.2)
Collecting google-api-core!=2.0.*, !=2.1.*, !=2.2.*, !=2.3.0, <3.0.0.dev0, >=1.31.5
  Downloading google_api_core-2.12.0-py3-none-any.whl (121 kB)
    121.4/121.4 KB 2.1 MB/s eta 0:00:00Collecting uritemplate<5, >=3.0.1
  Downloading uritemplate-4.1.1-py2.py3-none-any.whl (10 kB)
Collecting google-auth<3.0.0.dev0, >=1.19.0
  Downloading google_auth-2.23.3-py2.py3-none-any.whl (182 kB)
    182.3/182.3 KB 37.9 kB/s eta 0:00:00Collecting requests-oauthlib>=0.7.0
  Downloading requests_oauthlib-1.3.1-py2.py3-none-any.whl (23 kB)
Requirement already satisfied: googleapis-common-protos<2.0.dev0, >=1.56.2 in /usr/local/lib/python3.10/dist-packages (from google-api-core!=2.0.*, !=2.1.*, !=2.2.*, !=2.3.0, <3.0.0.dev0, >=1.31.5->google-api-python-client) (1.61.0)
Requirement already satisfied: requests<3.0.0.dev0, >=2.18.0 in /usr/local/lib/python3.10/dist-packages (from google-api-core!=2.0.*, !=2.1.*, !=2.2.*, !=2.3.0, <3.0.0.dev0, >=1.31.5->google-api-python-client) (2.31.0)
Requirement already satisfied: protobuf!=3.20.0, !=3.20.1, !=4.21.0, !=4.21.1, !=4.21.2, !=4.21.3, !=4.21.4, !=4.21.5, <5.0.0.dev0, >=3.19.5 in /usr/local/lib/python3.10/dist-packages (from google-api-core!=2.0.*, !=2.1.*, !=2.2.*, !=2.3.0, <3.0.0.dev0, >=1.31.5->google-api-python-client) (4.24.4)
Collecting rsa<5, >=3.1.4
  Downloading rsa-4.9-py3-none-any.whl (34 kB)
Collecting pyasn1-modules>=0.2.1
  Downloading pyasn1_modules-0.3.0-py2.py3-none-any.whl (181 kB)
    181.3/181.3 KB 1.2 MB/s eta 0:00:00Collecting cachetools<6.0, >=2.0.0
  Downloading cachetools-5.3.1-py3-none-any.whl (9.3 kB)
```

3. Apache Airflow DAG:

- I created an Apache Airflow Directed Acyclic Graph (DAG) with the following tasks:
 - **fetch_stock_data_task**: Fetches stock data from the Alpha Vantage API.

- **pre_process_stock_data_task:** Preprocesses the downloaded data, including calculating the moving average.
- **upload_file_to_drive_task:** Uploads the preprocessed data to Google Drive.

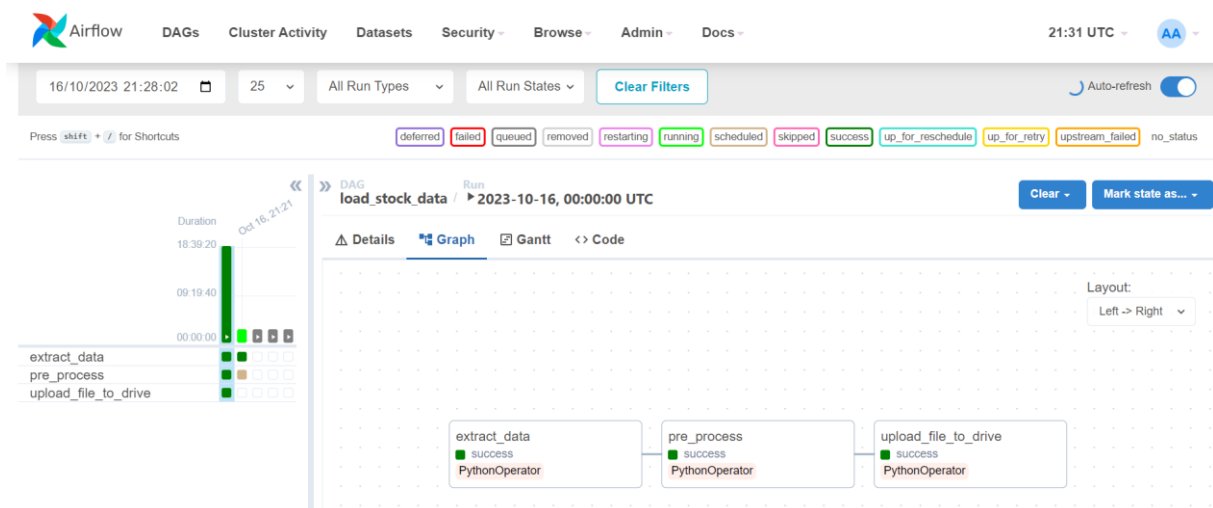
Here, we can see all the dags executed successfully:

```

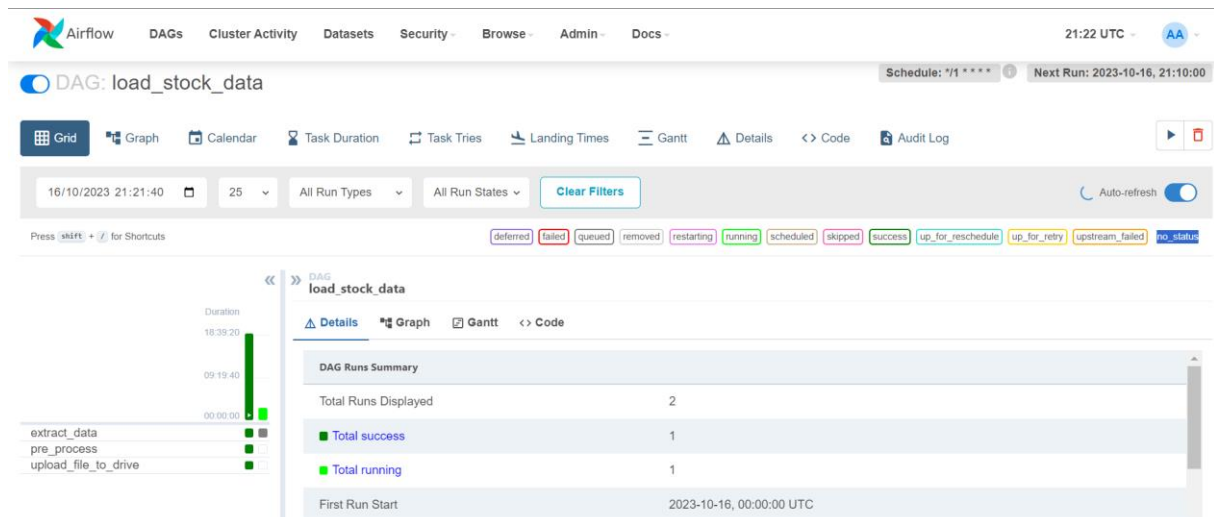
root@ikram: ~/airflow
[2023-10-16 19:39:15,004] {taskinstance.py:1400} INFO - Marking task as SUCCESS. dag_id=load_stock_data, task_id=pre_process, execution_date=20231016T000000, start_date=, end_date=20231016T183915
[2023-10-16T19:39:15.004+0100] {taskinstance.py:1400} INFO - Marking task as SUCCESS. dag_id=load_stock_data, task_id=pre_process, execution_date=20231016T000000, start_date=, end_date=20231016T183915
[2023-10-16T19:39:15.042+0100] {dag.py:3938} INFO - pre_process ran successfully!
[2023-10-16T19:39:15.043+0100] {dag.py:3941} INFO - *****
[2023-10-16T19:39:15.054+0100] {dag.py:3938} INFO - *****
[2023-10-16T19:39:15.056+0100] {dag.py:3934} INFO - Running task upload_file_to_drive
[2023-10-16 19:39:15,146] {taskinstance.py:1662} INFO - Exporting env vars: AIRFLOW_CTX_DAG_OWNER='airflow' AIRFLOW_CTX_DAG_ID='load_stock_data' AIRFLOW_CTX_TASK_ID='upload_file_to_drive' AIRFLOW_CTX_EXECUTION_DATE='2023-10-16T00:00:00+00:00' AIRFLOW_CTX_TRY_NUMBER='1' AIRFLOW_CTX_DAG_RUN_ID='manual__2023-10-16T00:00:00+00:00'
[2023-10-16T19:39:15.146+0100] {taskinstance.py:1662} INFO - Exporting env vars: AIRFLOW_CTX_DAG_OWNER='airflow' AIRFLOW_CTX_DAG_ID='load_stock_data' AIRFLOW_CTX_TASK_ID='upload_file_to_drive' AIRFLOW_CTX_EXECUTION_DATE='2023-10-16T00:00:00+00:00' AIRFLOW_CTX_TRY_NUMBER='1' AIRFLOW_CTX_DAG_RUN_ID='manual__2023-10-16T00:00:00+00:00'
Uploading 'processed_stock_data.csv' to Google Drive...
[2023-10-16T19:39:15.294+0100] {__init__.py:49} INFO - file_cache is only supported with oauth2client<4.0.0
File 'processed_stock_data.csv' uploaded to Google Drive with ID: 1gHCfaANvKgc8rp2k18BNP8aaoBdzF4Mc
[2023-10-16 19:39:20,518] {python.py:194} INFO - Done. Returned value was: None
[2023-10-16T19:39:20.518+0100] {python.py:194} INFO - Done. Returned value was: None
[2023-10-16 19:39:20,528] {taskinstance.py:1400} INFO - Marking task as SUCCESS. dag_id=load_stock_data, task_id=upload_file_to_drive, execution_date=20231016T000000, start_date=, end_date=20231016T183920
[2023-10-16T19:39:20.528+0100] {taskinstance.py:1400} INFO - Marking task as SUCCESS. dag_id=load_stock_data, task_id=upload_file_to_drive, execution_date=20231016T000000, start_date=, end_date=20231016T183920
[2023-10-16T19:39:20.547+0100] {dag.py:3938} INFO - upload_file_to_drive ran successfully!
[2023-10-16T19:39:20.547+0100] {dag.py:3941} INFO - *****
[2023-10-16T19:39:20.552+0100] {dagrun.py:653} INFO - Marking run <DagRun load_stock_data @ 2023-10-16T00:00:00+00:00: manual__2023-10-16T00:00:00+00:00, state:running, queued_at: None. externally triggered: False> successful
[2023-10-16T19:39:20.553+0100] {dagrun.py:704} INFO - DagRun Finished: dag_id=load_stock_data, execution_date=2023-10-16T00:00:00+00:00, run_id=manual__2023-10-16T00:00:00+00:00, run_start_date=2023-10-16T00:00:00+00:00, run_end_date=2023-10-16 18:39:20.553356+00:00, run_duration=67160.553356, state=success, external_trigger=False, run_type=manual, data_interval_start=2023-10-15T23:59:00+00:00, data_interval_end=2023-10-16T00:00:00+00:00, dag_hash=None
root@ikram:~/airflow#

```

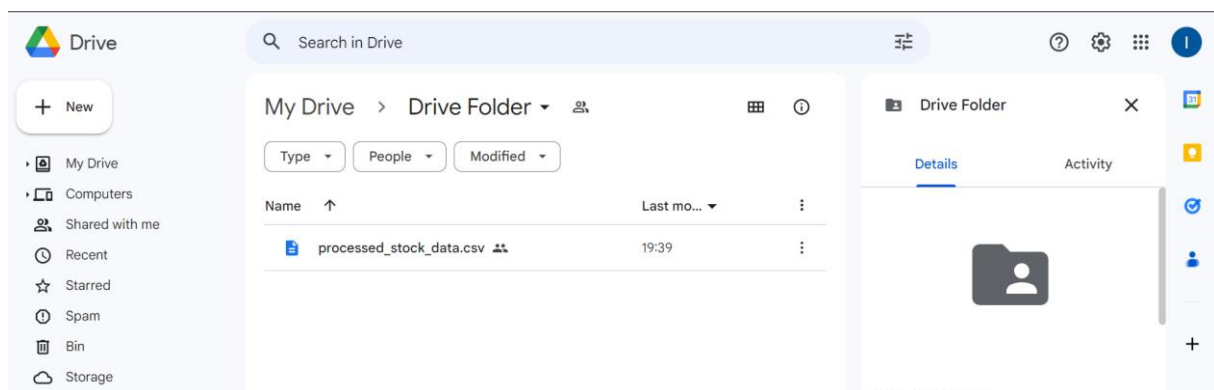
Which we can check in the airflow UI as well:



Here's the Workflow of our schedules ETL:



And the final processed stock csv file is successfully uploaded to my drive folder:

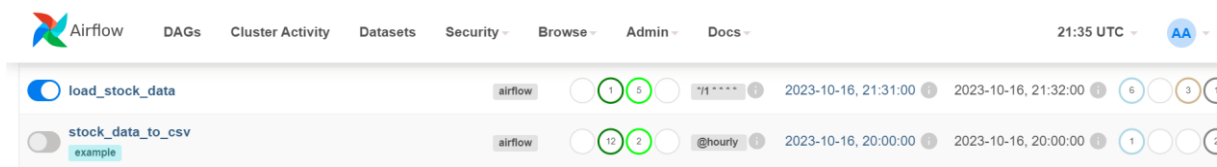


4. Scheduling:

- The Airflow DAG is scheduled to run every minute (**schedule="*/1 * * * *"**).
- The project is set to "catch up," meaning it will run for all the past dates up to the current date.

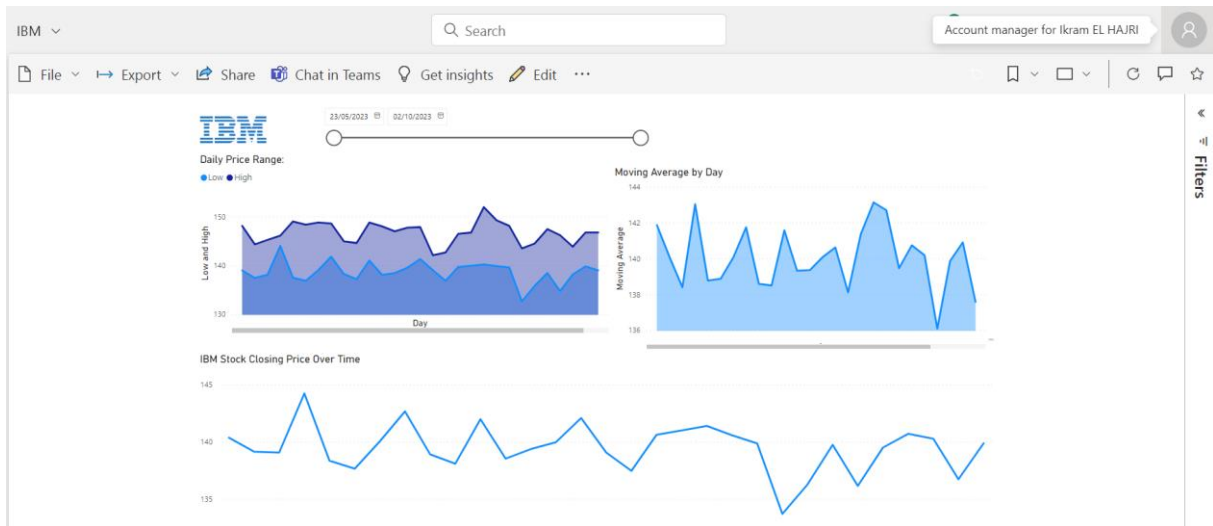
```
root@ikram:~/airflow# airflow scheduler

[2023-10-15T20:25:09.070+0100] {executor_loader.py:117} INFO - Loaded executor: SequentialExecutor
[2023-10-15 20:25:09 +0100] [1900] [INFO] Starting gunicorn 21.2.0
[2023-10-15 20:25:09 +0100] [1900] [INFO] Listening at: http://[::]:8793 (1900)
[2023-10-15 20:25:09 +0100] [1900] [INFO] Using worker: sync
[2023-10-15 20:25:09 +0100] [1901] [INFO] Booting worker with pid: 1901
[2023-10-15 20:25:09 +0100] [1902] [INFO] Booting worker with pid: 1902
[2023-10-15T20:25:09.328+0100] {scheduler_job_runner.py:798} INFO - Starting the scheduler
[2023-10-15T20:25:09.332+0100] {scheduler_job_runner.py:805} INFO - Processing each file at most -1 times
[2023-10-15T20:25:09.345+0100] {manager.py:166} INFO - Launched DagFileProcessorManager with pid: 1903
[2023-10-15T20:25:09.372+0100] {scheduler_job_runner.py:1598} INFO - Adopting or resetting orphaned tasks for active dag runs
[2023-10-15T20:25:09.415+0100] {settings.py:59} INFO - Configured default timezone Timezone('UTC')
airflow webserverairflow webserver
[2023-10-15T20:27:03.856+0100] {manager.py:410} WARNING - Because we cannot use more than 1 thread (parsing_processes = 2) when using
```

5. Integration with Power BI:

- I connected the "processed_stock_data.csv" file in Google Drive to Power BI.
- In Power BI, I created various visualizations to analyze the stock data, gaining insights into IBM stock trends.



6. Potential for Improvement:

While this project stands as a testament to the power of automation and data analysis, it also represents a steppingstone toward even greater possibilities. Among these possibilities is the integration of machine learning models to predict stock trends. This would elevate the project from a data analysis tool to a predictive analytics powerhouse, offering critical insights for investment decisions.

III. Conclusion

In my journey to learn and use Apache Airflow, I discovered its power in simplifying the management of complex data workflows. It enables data engineers and data scientists to create, automate, and monitor data pipelines efficiently. The flexibility, dynamic workflow changes, and extensibility make Apache Airflow a valuable skill for anyone working in the field of Big Data. My experience with Apache Airflow has opened new possibilities for orchestrating data tasks and has significantly improved my workflow management capabilities.