Amazon Stock Price Prediction for 7 days

Shao-Yu Huang
San Jose State University
shao-yu.huang@sjsu.edu

Lakshmi Bharathy Kumar
San Jose State University
lakshmibharathy.kumar@sjsu.edu

***Abstract*—This project aims to predict Amazon's future stock price using weekly historical data from the Alpha Vantage API and leveraging machine learning algorithms in Snowflake. It follows the ETL process, extracts data through the Alpha Vantage API, transforms it into a suitable format and sequence, and loads it into the Snowflake database for analysis. It uses time-series analysis in machine learning to forecast stock price trends, providing insights for investors in making decisions.**

I. Introduction

Stock Market investors usually consider various factors before investing their money. The investors need reliable data driven ways to make informed decisions. This stock price forecasting model acts as an auxiliary tool to offer guidance and insights for the investors by exploring trends, patterns and anomalies in the stock price. This statistical Time series analysis uses historical data to uncover seasonality, cyclic patterns, and future price movements, allowing investors to make informed decisions about when to buy or sell stocks. Stock market is very unpredictable, it fluctuates with companies performance and future expected earnings. Hence the purpose of building this time series forecasting system to mitigate the risk and uncertainty in the stock market investment.

This time series analysis has been done by leveraging the Snowpark which includes python libraries and Machine learning infrastructure, enabling popular ML frameworks such as Scikit-learn and XGBoost for handling complex model selection, feature engineering, and training processes. The ETL and ML forecasting pipeline has become easier with the use of Apache Airflow: an open-source tool used for scheduling and monitoring complex pipelines and workflows. This renders efficient data transformation, and automation, especially when working with large datasets.

The main goal of this forecasting is to predict Amazon's (AMZN) stock price for next 7 days using the last 180 days of daily stock price data from Alpha Vantage API. Predicting the next 7 days of Amazon's price data helps day traders/swing traders to make short-term buying or selling decisions. Amazon's stock is sensitive to macroeconomic news and sentiment, hence predicting next 7 days stock price helps assess short term market reactions.

II. Data Pipeline and Forecasting Techniques: A Review

Apache Airflow is the batch workflow orchestration platform. Airflow uses python to define workflows such as DAGs (Directed Acyclic Graph). We can create a number of Dags based on our project goals. Each DAGs represents a unique workflow and allows us to modularize the solution. Backfilling in Apache Airflow allows us to re-run the pipeline everytime we make changes to our logic. For this time series analysis, creating two DAGS, one for ETL and one for Prediction help us manage the workflows individually and enable tracking tasks run using logs, where we can view the error in the python code and resolve it easily by analysing the logs of each task.

To perform seamless pipeline Automation using Airflow, it is necessary to use Docker- an open source platform which provides an isolated environment to Airflow and manages dependencies in the Snowflake pipeline.

Docker simplifies the deployment and scaling by packaging all necessary components within the container. We defined all necessary software versions and configurations in a Docker compose file for reproducible deployment.

```
-f  Docker compose docker-compose.yaml up air flow-init
Docker compose up
Docker compose down
```

To automate the pipeline using Airflow and Snowflake together, first we need to establish a connection between them.

```
From airflow.providers.snowflake.hooks.snowflake import SnowFlake
```

In Apache Airflow, Snowflake Hook is used to establish a connection between airflow and snowflake. This allows us to execute queries within Dags and allow airflow tasks to interact with the snowflake. Before using snowflakehook, setup a config in Airflow Web UI(localhost:8081) { "account": "your_snowflake_account", "warehouse": "your_warehouse", "database": "your_database", "schema": "your_schema", } .

In this project, there are three tasks created for ETL(Extract,transform,load) using python. Each performs its own function such as Task1: extract data using SnowPark API, Task2:Transforming the data structure based on our need, in this case, we added a column “symbol: AMZN” , used (”symbol”,”Date”) as a composite primary key for the table “stock_price_table”. Task 3: Loading data to the snow_price_table created in snowflake using the previous task. This table is created under Database: DEV, Schema: RAW in snowflake which holds the raw data of Amazon’s stock price.

After the successful execution of ETL Dag, created another Snowpark file for time series forecasting which is a separate DAG, to perform forecasting upon the stored data. As Snowpark renders us with necessary in-built libraries and modules, the process of deploying time series ML forecasting becomes easier. Here are the few imported modules,

```
from airflow import DAG
from airflow.models import Variable
from airflow.decorators import task
from airflow.providers.snowflake.hooks.snowflake import SnowflakeHook
from airflow.utils.dates import days_ago
```

To predict the next 7 days stock price of Amazon using snowflake ML modelling, Created two different tasks, train and predict. Train task taking care of creating a “View”: which is dynamically updated and ensures that forecasting model always gets the latest stock price without the need of modifying the query or logic. And feeding “View” as input to the Snowflake ML forecast model created using,

```
CREATE_MODEL_SQL = f"""CREATE OR REPLACE SNOWFLAKE.ML.FORECAST
{FORECAST_FUNCTION_NAME} (
    INPUT_DATA => SYSTEM$REFERENCE('VIEW', '{TRAIN_VIEW}'),
    SERIES_COLNAME => 'SYMBOL',
    TIMESTAMP_COLNAME => 'DATE',
    TARGET_COLNAME => 'CLOSE',
    CONFIG_OBJECT => {{ 'ON_ERROR': 'SKIP' }}
);"""
```

The Prediction task is to call the model and do necessary prediction based on the input data.

```
FORECASTING_PERIODS => 7,
```

Setting the forecasting period to “7” in the prediction task is to tell the model to predict future 7 days stock_price and storing the forecasted result into the table. To offer a comprehensive view for analysis, combining the actual data with the forecasted data, with which we can compare actual stock price with the forecasted stock price and also assess the prediction accuracy. Once the forecast is done, the final combined data can be used in dashboards to visualize the data and compare actual vs predicted stock price data side by side. By creating the final table having both historical and forecasted data helps us analyse the performance of our model over time.

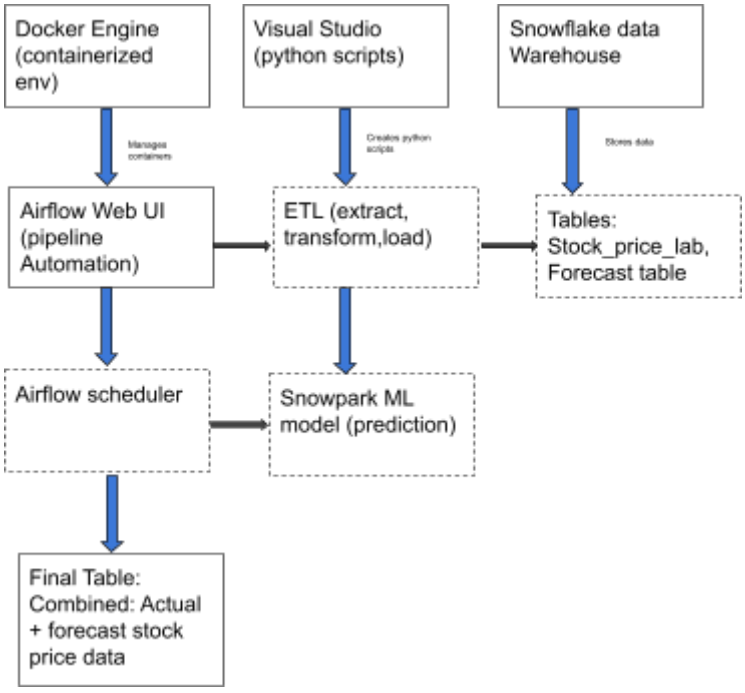


Fig.1. System Architecture Diagram

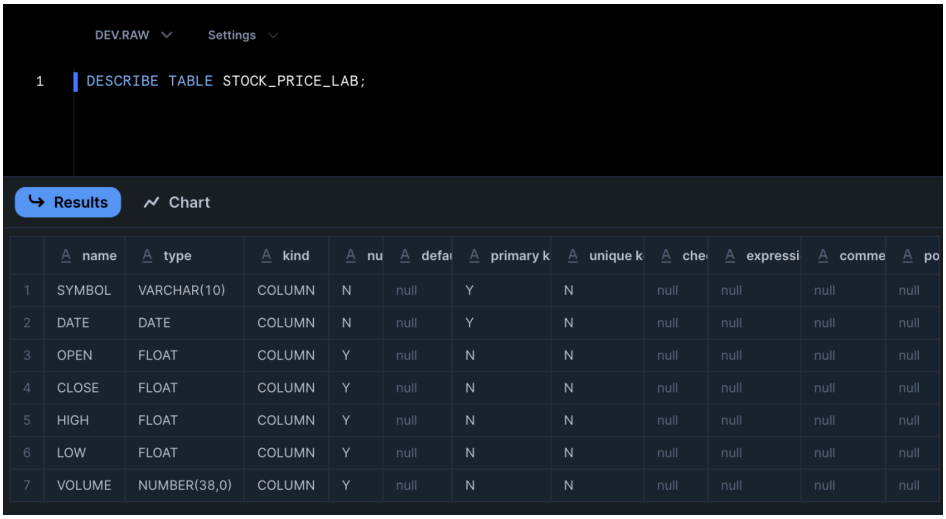
III. Database Table Structures and Execution Results and Output

In this lab, we constructed three tables and one view to manage and analyze stock prices. The tables are named: stock_price_lab, stock_price_lab_forecast, amazon_stock, and the view is referred to as stock_price_lab_view. The amazon_stock_prediction serves as a forecasting model which utilizes stock_price_lab_view as input and uses the DATE column as the time series and the CLOSE column as the target value to predict and train the forecasting model.

- stock_price_lab: stored the original data from alpha vantage API.
- stock_price_lab_view: stored date and close price data from stock_price_lab.
- stock_price_lab_forecast: stores the forecasted stock price data for the next 7 days, generated by the Snowflake ML forecasting function.
- amazon_stock: is the final combination of historical data and forecasting prediction.

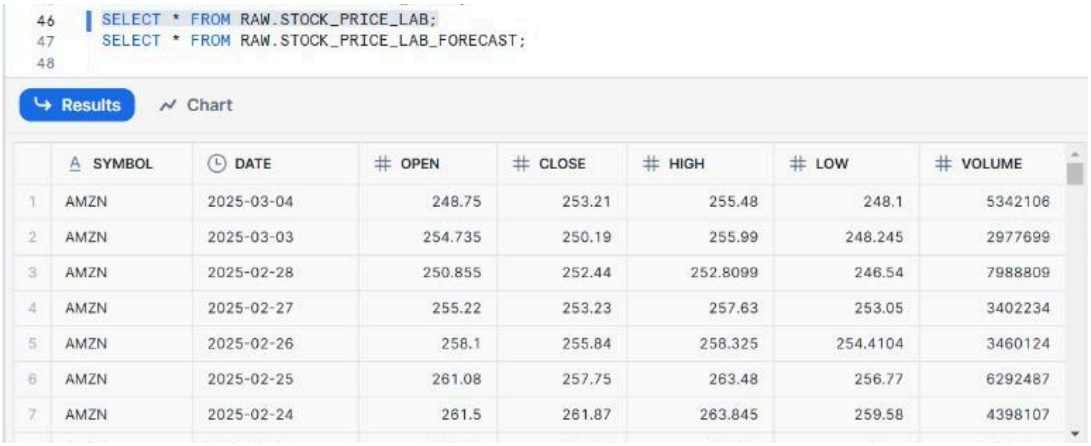
The structure of each table, including fields, attributes, data types, and constraints are shown by Snowflake's command "DESCRIBE", Figures 1 through 5 show the respective table structures and results in the Snowflake UI.

- stock_price_lab: Column names are SYMBOL, DATE, OPEN, CLOSE, HIGH, LOW, VOLUME and SYMBOL and DATE are the primary key.



	name	type	kind	nu	defa	primary k	unique k	che	expressi	comme	po
1	SYMBOL	VARCHAR(10)	COLUMN	N	null	Y	N	null	null	null	null
2	DATE	DATE	COLUMN	N	null	Y	N	null	null	null	null
3	OPEN	FLOAT	COLUMN	Y	null	N	N	null	null	null	null
4	CLOSE	FLOAT	COLUMN	Y	null	N	N	null	null	null	null
5	HIGH	FLOAT	COLUMN	Y	null	N	N	null	null	null	null
6	LOW	FLOAT	COLUMN	Y	null	N	N	null	null	null	null
7	VOLUME	NUMBER(38,0)	COLUMN	Y	null	N	N	null	null	null	null

Fig. 2. Structure of the stock_price_lab table.



	SYMBOL	DATE	OPEN	CLOSE	HIGH	LOW	VOLUME
1	AMZN	2025-03-04	248.75	253.21	255.48	248.1	5342106
2	AMZN	2025-03-03	254.735	250.19	255.99	248.245	2977699
3	AMZN	2025-02-28	250.855	252.44	252.8099	246.54	7988809
4	AMZN	2025-02-27	255.22	253.23	257.63	253.05	3402234
5	AMZN	2025-02-26	258.1	255.84	258.325	254.4104	3460124
6	AMZN	2025-02-25	261.08	257.75	263.48	256.77	6292487
7	AMZN	2025-02-24	261.5	261.87	263.845	259.58	4398107

Fig. 3. Output result of the stock_price_lab table.

- stock_price_lab_view: Column names are DATE and CLOSE with no constraints.

```
12
13  DESCRIBE TABLE dev.raw.stock_price_lab_view;
14
```

Results | Chart

	name	type	kind	nu	defa	primary k	unique k	che	expressi	comme	policy nar
1	DATE	DATE	COLUMN	N	null	N	N	null	null	null	null
2	CLOSE	FLOAT	COLUMN	Y	null	N	N	null	null	null	null

Fig. 4. Structure of the stock_price_lab_view table.

```
48  SELECT * FROM RAW.STOCK_PRICE_LAB_VIEW;
49
```

Results | Chart

	DATE	CLOSE	SYMBOL
1	2025-03-04	253.21	AMZN
2	2025-03-03	250.19	AMZN
3	2025-02-28	252.44	AMZN
4	2025-02-27	253.23	AMZN
5	2025-02-26	255.84	AMZN
6	2025-02-25	257.75	AMZN
7	2025-02-24	261.87	AMZN

Fig. 5. Output result of the stock_price_lab_view table.

- stock_price_lab_forecast: Column names are SERIES, TS, FORECAST, LOWER_BOUND, UPPER_BOWND with no constraints.

```
14
15
16
17  DESCRIBE TABLE dev.raw.stock_price_lab_forecast;
18
```

Results | Chart

	name	type	kind	nu	defa	primary k	unique k	che	expressi	com
1	SERIES	VARIANT	COLUMN	Y	null	N	N	null	null	null
2	TS	TIMESTAMP_NTZ(9)	COLUMN	Y	null	N	N	null	null	null
3	FORECAST	FLOAT	COLUMN	Y	null	N	N	null	null	null
4	LOWER_BOUND	FLOAT	COLUMN	Y	null	N	N	null	null	null
5	UPPER_BOUND	FLOAT	COLUMN	Y	null	N	N	null	null	null

Fig. 6. Structure of the stock_price_lab_forecast table.

```
47  SELECT * FROM RAW.STOCK_PRICE_LAB_FORECAST;
48
```

Results | Chart

	SERIES	TS	FORECAST	LOWER_BOUND	UPPER_BOUND
1	null	2025-03-05 00:00:00.000	253.719436522	245.91736889	262.608400396
2	null	2025-03-06 00:00:00.000	253.808281625	241.160414506	265.939143513
3	null	2025-03-07 00:00:00.000	253.757476474	238.085016769	267.881149174
4	null	2025-03-10 00:00:00.000	253.751129798	235.966432533	272.5039073
5	null	2025-03-11 00:00:00.000	253.978397127	234.582554804	272.822748782
6	null	2025-03-12 00:00:00.000	254.185154247	234.609376509	272.98655378
7	null	2025-03-13 00:00:00.000	254.389647322	231.542644541	276.944073626

Fig. 7. Output result of the stock_price_lab_forecast table.

- amazon_stock: Column names are SYMBOL, DATE, ACTUAL, FORECAST, LOWER_BOUND, UPPER_BOUNDs with no constraints.

```
24  DESCRIBE TABLE dev.analytics.amazon_stock;
25
26
27
```

Results | Chart

	name	type	kind	nu	defa	primary k	unique k	che	expressi	cor
1	SYMBOL	VARCHAR(16777216)	COLUMN	Y	null	N	N	null	null	null
2	DATE	TIMESTAMP_NTZ(9)	COLUMN	Y	null	N	N	null	null	null
3	ACTUAL	FLOAT	COLUMN	Y	null	N	N	null	null	null
4	FORECAST	FLOAT	COLUMN	Y	null	N	N	null	null	null
5	LOWER_BOUND	FLOAT	COLUMN	Y	null	N	N	null	null	null
6	UPPER_BOUND	FLOAT	COLUMN	Y	null	N	N	null	null	null

Fig. 8. Structure of the amazon_stock table.

```
44
45  SELECT * FROM ANALYTICS.AMAZON_STOCK;
46
```

Results | Chart

	SYMBOL	DATE	ACTUAL	FORECAST	LOWER_BOUND	UPPER_BOUND
101	null	2025-03-05 00:00:00.000	null	253.732003094	245.201971112	263.028714636
102	null	2025-03-06 00:00:00.000	null	254.201856326	242.252509993	265.784645644
103	null	2025-03-07 00:00:00.000	null	254.431330224	240.346537855	267.41837919
104	null	2025-03-10 00:00:00.000	null	254.658713667	239.080932064	270.909679652
105	null	2025-03-11 00:00:00.000	null	254.622730412	237.903998054	270.935550344
106	null	2025-03-12 00:00:00.000	null	254.799866663	237.950334853	271.080331619
107	null	2025-03-13 00:00:00.000	null	255.033361252	235.873406294	273.958847139

Fig. 9. Output result of the amazon_stock table.

IV. Airflow Pipeline Configuration and Execution

This project consists of two pipelines: Lab1_ETL_stockprice.py for performing ETL from the API source and Lab1_train_predict.py for building an ML model and forecasting. These two pipelines have five tasks, and the execution order is as follows: extract, transform, load, train, and predict. The screenshots of each task are in the following:

- Airflow homepage of all the DAGs:

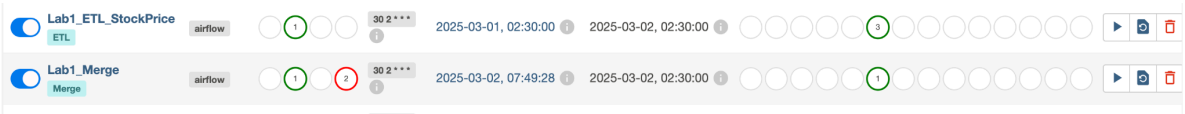


Fig. 10. Screenshot of Airflow homepage.

- DAG - Lab1_StockPrice - extract:

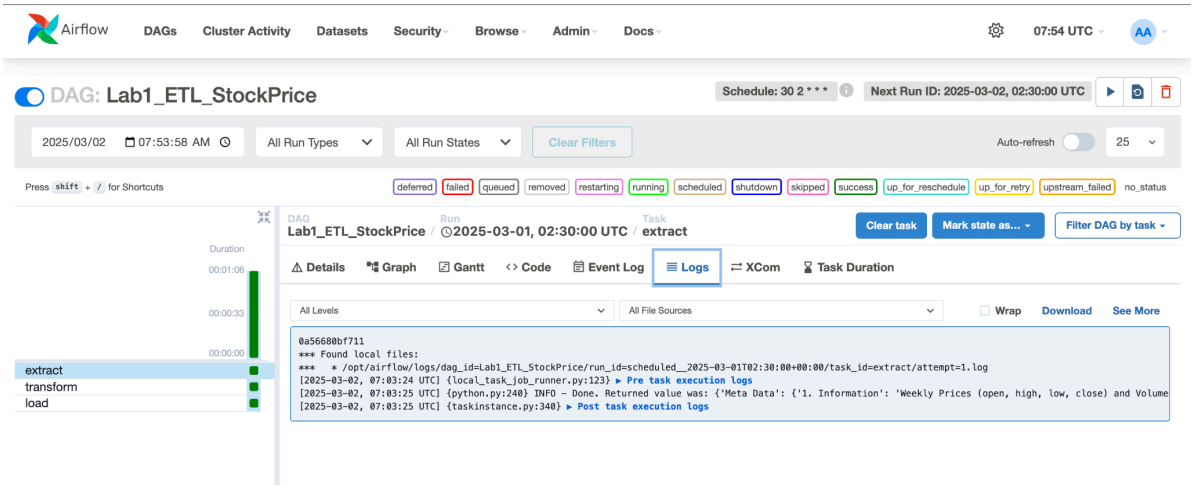


Fig. 11. Screenshot of task extract in Lab1_StockPrice.

- DAG - Lab1_StockPrice - transform:

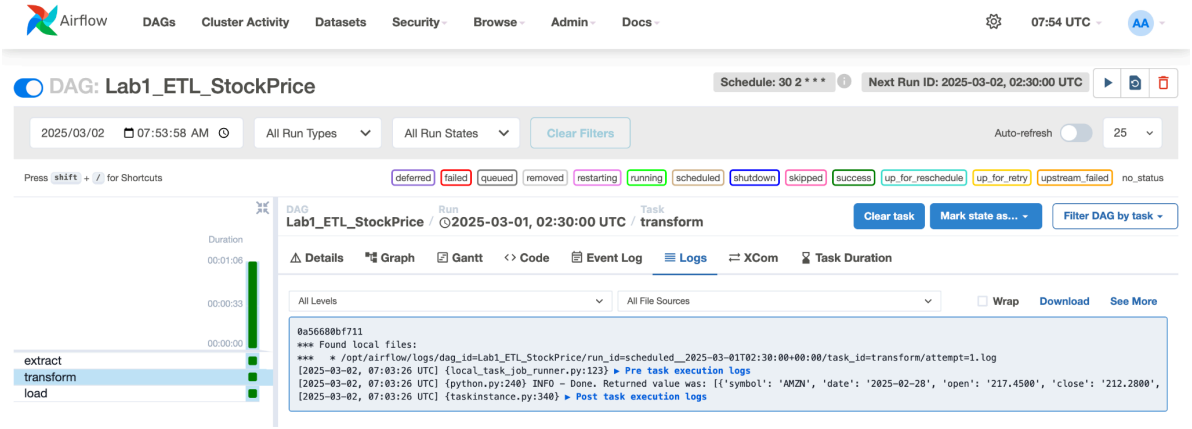


Fig. 12. Screenshot of task transform in Lab1_StockPrice.

- DAG - Lab1_StockPrice - load:

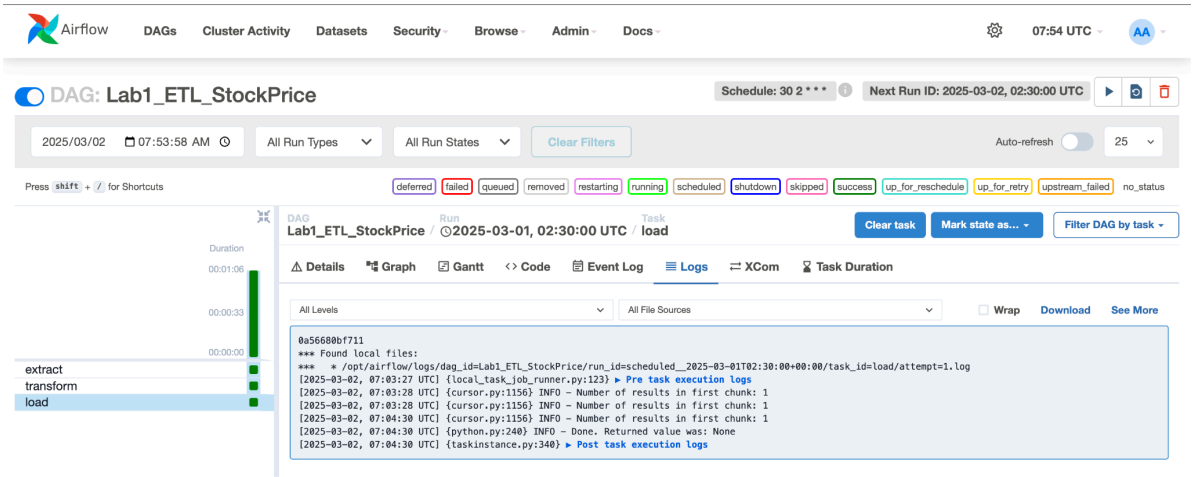


Fig. 13. Screenshot of task load in Lab1_StockPrice.

- DAG - TrainPredict - train:

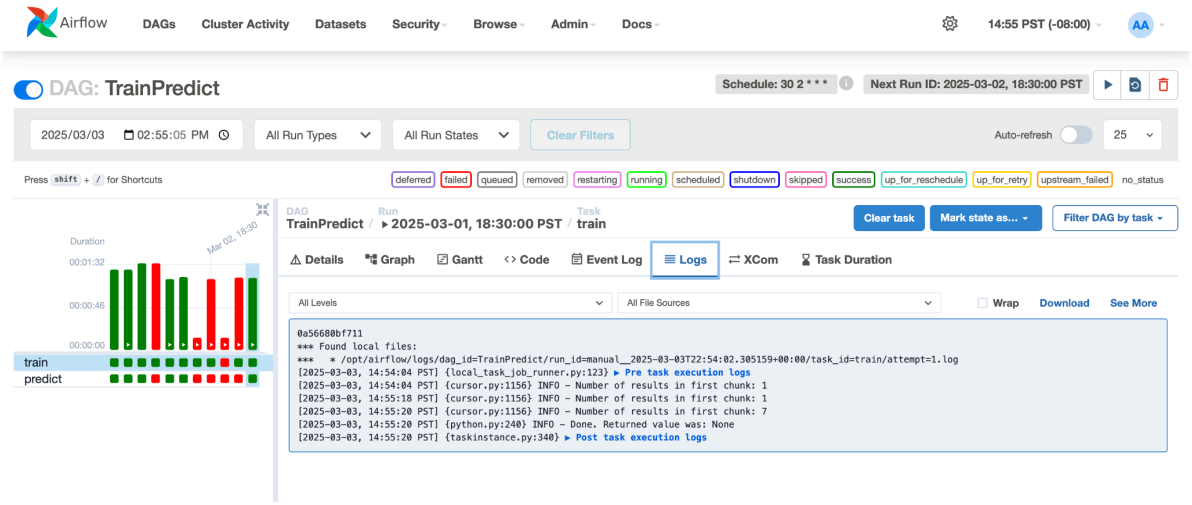


Fig. 14. Screenshot of task train in TrainPredict.

- DAG - TrainPredict - predict:

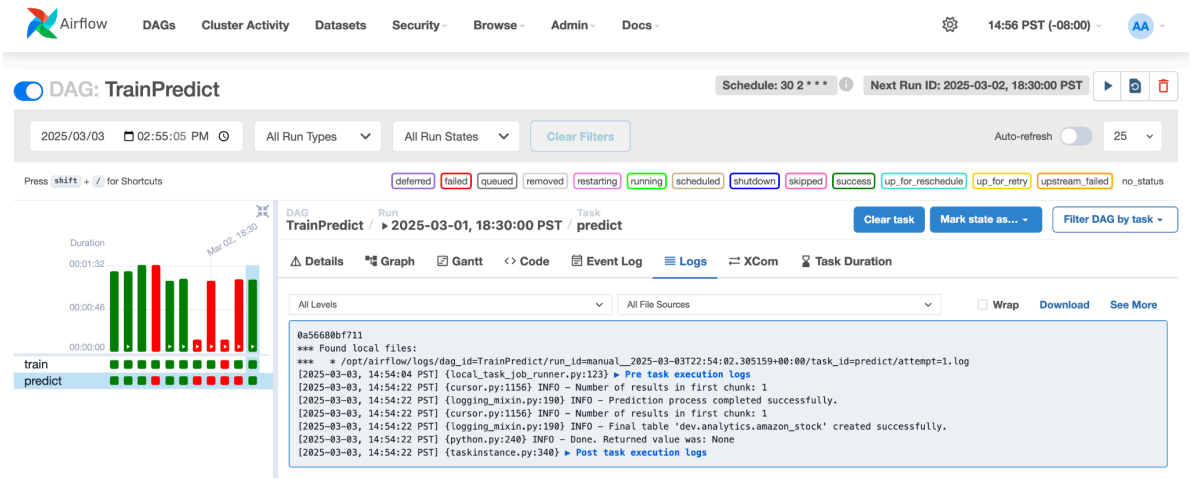


Fig. 15. Screenshot of task predicted in TrainPredict.

V. References

Github Link for Lab1_ETL_stockprice.py:
https://github.com/Sandyhsy/DATA226-Data_Warehouse/blob/96efa6b2fe3992ed5ddb5e6ff828c8e51a6b5587/Lab1/Lab1_ETL_stockprice.py
Github Link for Lab1_trainpredict.py:
https://github.com/Sandyhsy/DATA226-Data_Warehouse/blob/2802a5aa77d2803afdc72c6646bfa368fe9bff73/Lab1/Lab1_trainpredict.py