Projektbeschrieb Stock Data Pipeline

Studierendenprojekt

Modul: Data Engineering and Wrangling
Institution: Fachhochschule Nordwestschweiz
Vivien Aregger WIVZ 2.51

Einleitung

Im Rahmen des Moduls *Data Engineering and Wrangling* entwickelte ich eine vollständige Datenpipeline zur Erfassung, Speicherung, Verarbeitung und Visualisierung von Finanzzeitreihendaten mit modernen Tools und Technologien. Der Fokus des Projekts lag auf der automatisierten täglichen Abfrage von Börsendaten (konkret Apple Inc. – AAPL), deren strukturierter Speicherung in einer relationalen Datenbank sowie einer einfachen Zeitreihenprognose mit anschließender Visualisierung. Das gesamte System wurde mithilfe von Docker containerisiert und mit Apache Airflow automatisiert.

Was ich gemacht habe

- **Automatisierte Datenpipeline:** Umsetzung einer ETL-Pipeline (Extract, Transform, Load) mit Apache Airflow zur automatisierten Abfrage von Börsendaten über Yahoo Finance.
- **Datenbereinigung und -speicherung:** Bereinigung der Daten mit Python (pandas) und Speicherung in einer PostgreSQL-Datenbank.
- **Prognose:** Entwicklung eines einfachen ARIMA-Modells zur Vorhersage von Aktienkursen für die nächsten fünf Tage.
- **Visualisierung:** Erstellung eines Python-Skripts zur grafischen Darstellung historischer und prognostizierter Aktienkurse.
- **Deployment:** Nutzung von Docker Compose zur Containerisierung der gesamten Umgebung (PostgreSQL, Airflow und Python-Skripte), um eine einfache Bereitstellung und Verwaltung zu ermöglichen.

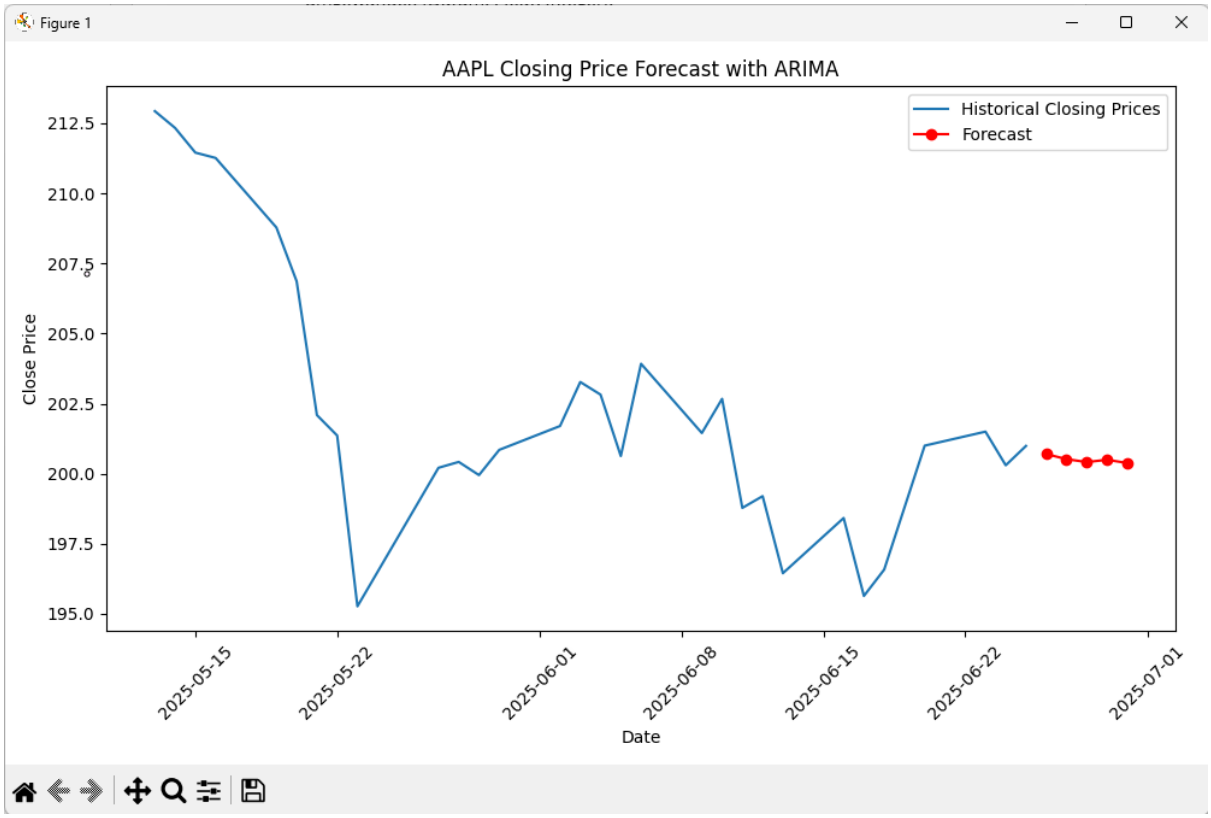
Eingesetzte Tools und Technologien

- **Apache Airflow:** Für die Orchestrierung der Datenpipeline (tägliche Ausführung).
- **Python:** Für Datenverarbeitung, Prognose (ARIMA mit statsmodels) und Visualisierung (Matplotlib).
- **PostgreSQL:** Als relationale Datenbank zur Speicherung der Börsendaten.
- **Docker + Docker Compose:** Zur Containerisierung und Sicherstellung der Reproduzierbarkeit.
- **Yahoo Finance (über yfinance-Bibliothek):** Als Datenquelle für aktuelle Börsenkurse.

Funktionsweise

1. **Datenextraktion:** Ein DAG in Airflow startet ein Python-Skript, das die letzten 30 Tage Börsendaten von AAPL über die yfinance-API abruft.
2. **Datenaufbereitung:** Die Daten werden bereinigt und in passende Datentypen umgewandelt (Datum, Float, Integer).
3. **Datenbank-Speicherung:** Die bereinigten Daten werden in eine PostgreSQL-Tabelle namens stock_prices eingefügt. Doppelte Datensätze werden mithilfe eines Konflikt-Checks übersprungen.
4. **Prognose & Visualisierung:** Ein separates Skript liest die Daten aus der Datenbank, erstellt ein ARIMA-Modell und visualisiert sowohl historische als auch prognostizierte Kurse.

Beispiel



Was ich gelernt habe

- **Praxisnahe Nutzung von Airflow:** Vertieftes Verständnis für den Aufbau und die Steuerung von Airflow-DAGs, Aufgabenstrukturierung und automatisierte Ausführungen.
- **Datenintegration und -aufbereitung:** Die Daten von **Yahoo Finance** (über yfinance) sind bereits gut strukturiert und keine offensichtlichen Fehler oder Inkonsistenzen. Es war also **keine umfangreiche Datenbereinigung notwendig**. Das, was gemacht wurde (z. B. Umwandlung von Datumsformaten, Typkonvertierung), war **Wrangling der Daten** für die Datenbank-Kompatibilität.
- **Zeitreihenanalyse:** Grundlagen des ARIMA-Modells und dessen Anwendung auf Finanzzeitreihen kennengelernt.
- **Containerisierung mit Docker:** Erlernt, wie man komplexe Multi-Service-Umgebungen mit Docker Compose aufbaut und verwaltet.

- **Datenbankanbindung mit Python:** Praktische Erfahrung im Umgang mit PostgreSQL aus Python heraus über psycopg2.

Fazit

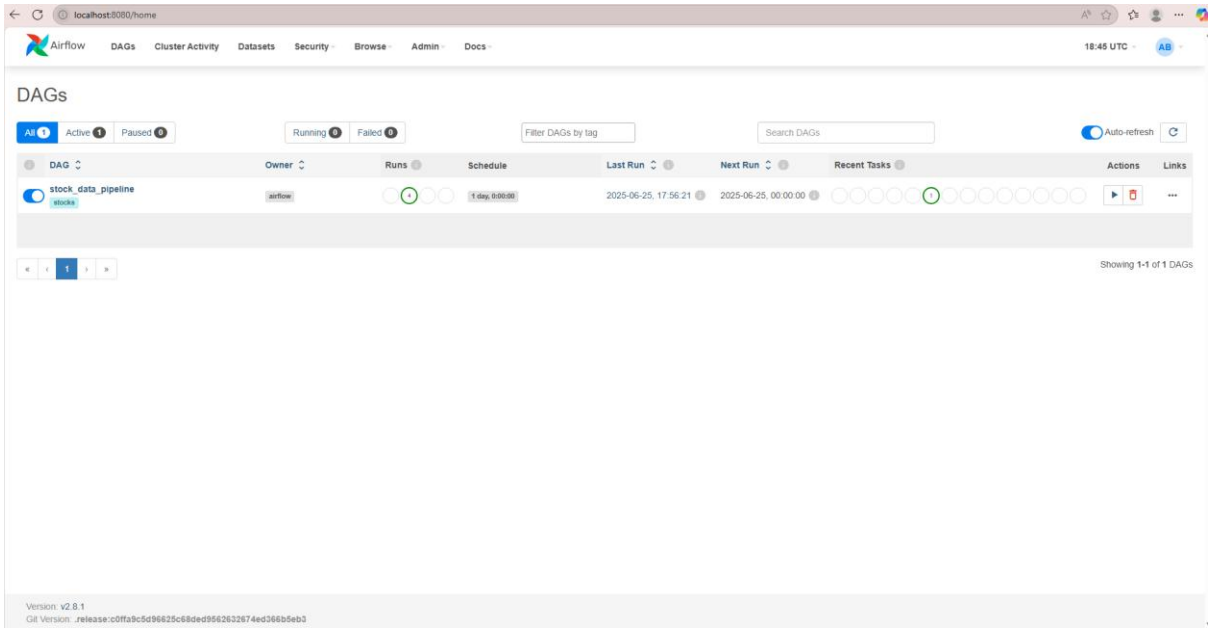
Dieses Projekt bot mir wertvolle Einblicke in den Aufbau von End-to-End-Datenprozessen unter realistischen Bedingungen. Es kombinierte zahlreiche Kernkompetenzen wie Datenbeschaffung, -aufbereitung, -speicherung, Orchestrierung, Modellierung und Visualisierung – alles essenzielle Bausteine im modernen Data Engineering.

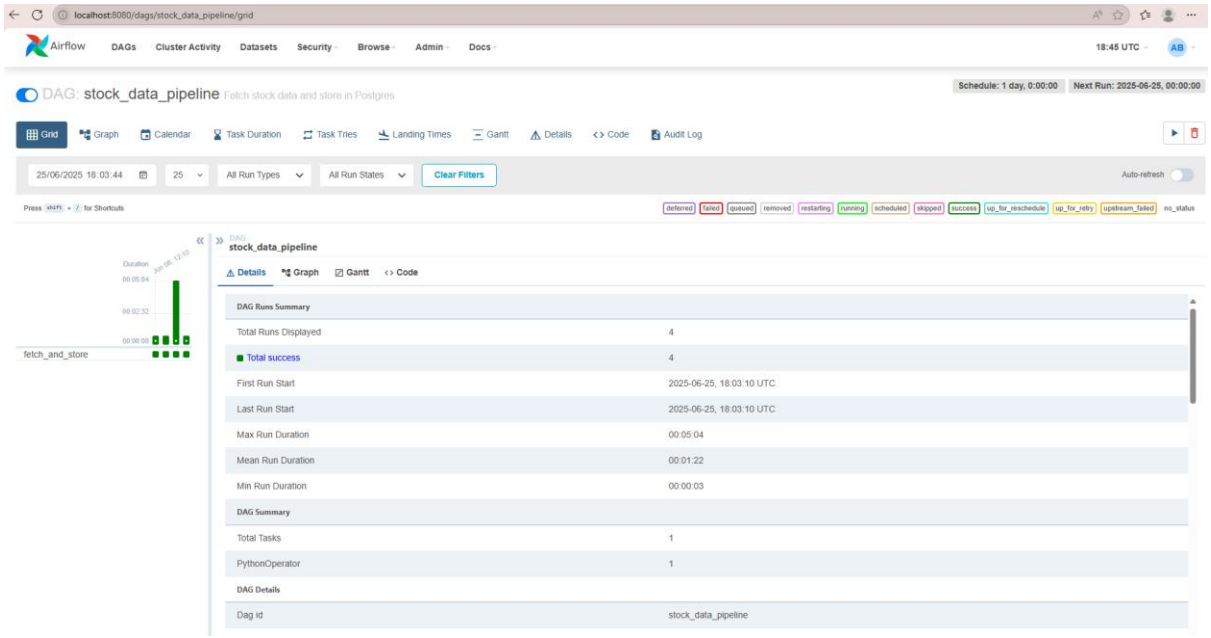
Ausblick

Dieses Projekt bildet eine solide Grundlage für weitere Entwicklungen und Erweiterungen. In zukünftigen Versionen könnten ich folgende Verbesserungen und Ergänzungen machen:

- **Mehrere Aktien-Ticker:** Erweiterung der Pipeline zur dynamischen Verarbeitung mehrerer Aktienkennzahlen anstelle eines fest kodierten Tickers (AAPL).
- **Fortgeschrittene Prognosemodelle:** Einsatz robusterer Vorhersagemethoden wie Prophet oder LSTM-Neuronale Netzwerke für präzisere Langzeitprognosen.
- **Echtzeit-Streaming:** Integration von Echtzeit-Datenströmen z. B. über Kafka oder WebSocket-APIs für zeitnahe Analysen.
- **Dashboard-Integration:** Einbindung der Visualisierungen in ein interaktives Dashboard (z. B. mit Streamlit, Dash oder Tableau) für eine benutzerfreundliche Präsentation.
- **Datenqualitäts-Monitoring:** Implementierung von Validierungs- und Qualitätssicherungsmechanismen innerhalb der Airflow-DAGs zur Früherkennung von Anomalien.

Anhang


localhost:8080/home
Airflow
DAGs
Cluster Activity
Datasets
Security
Browse
Admin
Docs
18:45 UTC
AB
DAGs
All 1
Active 1
Paused 0
Running 0
Failed 0
Filter DAGs by tag
Search DAGs
Auto-refresh
DAG
Owner
Runs
Schedule
Last Run
Next Run
Recent Tasks
Actions
Links
stock_data_pipeline
stocks
airflow
1 day, 0:00:00
2025-06-25, 17:56:21
2025-06-25, 00:00:00
Showing 1-1 of 1 DAGs
Version: v2.8.1
Git Version: .release:c0ffa9c5d96625c68ded9562632674ed366b5eb3


localhost:8080/dags/stock_data_pipeline/grid
Airflow
DAGs
Cluster Activity
Datasets
Security
Browse
Admin
Docs
18:45 UTC
AB
DAG: stock_data_pipeline
Fetch stock data and store in Postgres
Schedule: 1 day, 0:00:00
Next Run: 2025-06-25, 00:00:00
Grid
Graph
Calendar
Task Duration
Task Tries
Landing Times
Gantt
Details
Code
Audit Log
25/06/2025 18:03:44
25
All Run Types
All Run States
Clear Filters
Auto-refresh
Press shift + / for Shortcuts
deferred
failed
queued
removed
restarting
running
scheduled
skipped
success
up_for_reschedule
up_for_retry
upstream_failed
no_status
Duration
fetch_and_store
DAG
stock_data_pipeline
Details
Graph
Gantt
Code
DAG Runs Summary
Total Runs Displayed
4
Total success
4
First Run Start
2025-06-25, 18:03:10 UTC
Last Run Start
2025-06-25, 18:03:10 UTC
Max Run Duration
00:05:04
Mean Run Duration
00:01:22
Min Run Duration
00:00:03
DAG Summary
Total Tasks
1
PythonOperator
1
DAG Details
Dag id
stock_data_pipeline

```
PS C:\Users\vivie> docker exec -it stock-pipeline-main-postgres-1 psql -U airflow -d stockdb
psql (13.21 (Debian 13.21-1.pgdg120+1))
Type "help" for help.

stockdb=# \d stock_prices
                Table "public.stock_prices"
 Column |       Type       | Collation | Nullable | Default
--------+------------------+-----------+----------+---------
 date   | date             |           | not null |
 open   | double precision |           |          |
 high   | double precision |           |          |
 low    | double precision |           |          |
 close  | double precision |           |          |
 volume | bigint           |           |          |
Indexes:
    "stock_prices_pkey" PRIMARY KEY, btree (date)

stockdb=# SELECT COUNT(*) FROM stock_prices;
 count
-------
    30
(1 row)

stockdb=# SELECT * FROM stock_prices ORDER BY date DESC LIMIT 5;
    date    |        open        |        high        |        low         |       close        |  volume
------------+--------------------+--------------------+--------------------+--------------------+----------
 2025-06-25 |  201.4199981689453 | 203.66000366210938 | 200.62010192871094 | 200.99000549316406 | 25383186
 2025-06-24 | 202.58999633789062 | 203.44000244140625 |  200.1999969482422 |  200.3000030517578 | 53972500
 2025-06-23 |  201.6300048828125 |  202.3000030517578 |  198.9600067138672 |              201.5 | 55814300
 2025-06-20 | 198.24000549316406 |  201.6999969482422 | 196.86000061035156 |                201 | 96813500
 2025-06-18 | 195.94000244140625 | 197.57000732421875 | 195.07000732421875 |  196.5800018310547 | 45394700
(5 rows)

stockdb=#
```