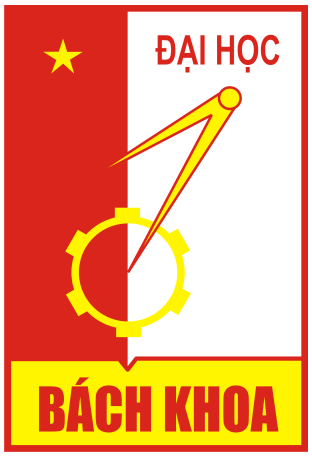
ĐẠI HỌC BÁCH KHOA HÀ NỘI

TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG



MÔN HỌC: PROJECT 03 - IT3940

TÊN ĐỀ TÀI:
HỆ THỐNG THU THẬP, LƯU TRỮ VÀ XỬ LÝ DỮ LIỆU CHỨNG KHOÁN THỜI GIAN THỰC

Giáo viên hướng dẫn: **TS. Nguyễn Đức Anh**

Sinh viên thực hiện **Nguyễn Xuân Bình** **20224930**

Hà Nội, ngày 10 tháng 01 năm 2026

Mục lục

Lời nói đầu **3**

1 Giới thiệu chung **4**

1.1 Lý do chọn đề tài 4

1.2 Mục tiêu của hệ thống 4

1.3 Đối tượng và Phạm vi 5

2 Kiến trúc hệ thống **6**

2.1 Tổng quan 7

2.2 Các thành phần chính 7

2.2.1 Tầng Thu Thập Dữ Liệu (Data Ingestion) 7

2.2.2 Tầng Lưu Trữ Dữ Liệu (Temporary Storage) 7

2.2.3 Tầng Điều Phối (Orchestration) 8

2.2.4 Tầng Kho Dữ Liệu (Data Warehouse - BigQuery) 8

2.2.5 Tầng Chuyển Đổi Dữ Liệu (dbt - Data Build Tool) 8

2.2.6 Tầng Trực Quan Hóa (Visualization) 8

3 Thiết kế và Triển khai chi tiết **9**

3.1 Tầng Thu Thập Dữ Liệu (Data Ingestion) 9

3.1.1 Thành Phần 1: Kafka Producer 9

3.1.2 Thành Phần 2: Apache Kafka Broker 9

3.1.3 Thành Phần 3: Kafka Consumer 10

3.1.4 Biến Đổi Định Dạng Dữ Liệu (Data Format Transformation) 10

3.2 Tầng lưu trữ dữ liệu 11

3.2.1 Tổng Quan 11

3.2.2 Thành Phần: MinIO Object Storage 11

3.2.3 Cấu Trúc Lưu Trữ 11

3.2.4 Quy Trình Ghi Dữ Liệu (Write Path) 12

3.2.5 Quy Trình Đọc Dữ Liệu (Read Path) 13

3.2.6 Đặc Điểm Kiến Trúc (Buffer Pattern) 13

3.3 Tầng điều phối 13

3.3.1 Tổng Quan 13

3.3.2 Vai Trò và Chức Năng 14

3.3.3 DAG: minio_to_bigquery_multi 14

3.3.4 Chi tiết các Task 15

3.3.5 Metadata Tracking và Incremental Loading 15

3.3.6 Đặc điểm Kiến trúc 16

3.3.7 Đặc tính Hiệu suất (Performance Characteristics) 16

3.4 Tầng Kho Dữ Liệu (Data Warehouse) 17

3.4.1 Tổng Quan . . . 17
3.4.2 Thành Phần: Google BigQuery . . . 17
3.4.3 Cấu Trúc Dữ Liệu . . . 18
3.4.4 Bronze Layer (Raw Data) . . . 18
3.4.5 Silver Layer (Cleaned Data) . . . 19
3.4.6 Gold Layer (Curated Analytics) . . . 20
3.5 Tầng chuyển đổi dữ liệu . . . 21
3.5.1 Tổng Quan . . . 21
3.5.2 Kiến Trúc Tổng Thể . . . 21
3.5.3 Mô hình Kiến trúc Medallion (Medallion Architecture) . . . 21
3.5.4 Bronze Layer . . . 22
3.5.5 Silver Layer . . . 22
3.5.6 Gold Layer . . . 22
3.6 Tầng trực quan hóa dữ liệu . . . 23
3.6.1 Tổng Quan . . . 23
3.6.2 Thành Phần: Grafana . . . 24
3.6.3 Các loại biểu đồ và Dashboard . . . 24

4 Phân tích chi tiết các chỉ báo kỹ thuật 25
4.1 Biểu đồ chỉ số Relative Strength Index (RSI) . . . 25
4.2 Biểu đồ chỉ số Average True Range (ATR) . . . 25
4.3 Biểu đồ Độ biến động tương đối (Relative Volatility) . . . 26
4.4 Biểu đồ Giá trung bình mới nhất theo ngày . . . 27
4.5 Biểu đồ chỉ số Simple Moving Average (SMA) . . . 27
4.6 Biểu đồ nến (Candlestick Chart) . . . 28
4.7 Biểu đồ giá giao dịch thời gian thực . . . 28

5 Kết quả và Đánh giá 29
5.1 Kết quả triển khai . . . 29
5.2 Đánh giá hệ thống . . . 31
5.2.1 Tổng quan triển khai . . . 31
5.2.2 Nhận xét và Đánh giá . . . 32

Lời nói đầu

Trong kỷ nguyên số hóa hiện nay, dữ liệu được ví như "nguồn nhiên liệu mới" thúc đẩy sự phát triển của nền kinh tế toàn cầu, đặc biệt là trong lĩnh vực tài chính và thị trường chứng khoán. Với sự biến động không ngừng từng giây, từng phút của giá cổ phiếu, việc tiếp cận thông tin nhanh chóng và chính xác đã trở thành yếu tố sống còn, quyết định lợi thế cạnh tranh của các nhà đầu tư cũng như các tổ chức tài chính.

Tuy nhiên, thách thức lớn nhất hiện nay không còn nằm ở việc thiếu hụt thông tin mà nằm ở khả năng xử lý khối lượng dữ liệu khổng lồ (*Big Data*) phát sinh liên tục theo thời gian thực. Các phương pháp quản trị dữ liệu truyền thống đang dần bộc lộ những hạn chế về độ trễ, khả năng mở rộng và tính linh hoạt trước sự đa dạng của các nguồn dữ liệu từ các sàn giao dịch quốc tế. Thực tế này đòi hỏi một giải pháp công nghệ mang tính đột phá, có khả năng tự động hóa hoàn toàn quy trình từ khâu thu thập đến phân tích chuyên sâu.

Xuất phát từ nhu cầu cấp thiết đó, nhóm chúng em đã nghiên cứu và thực hiện đề tài: **"Xây dựng hệ thống Thu thập, Lưu trữ và Xử lý dữ liệu chứng khoán thời gian thực"**. Hệ thống được thiết kế dựa trên các tiêu chuẩn hiện đại nhất của kỹ thuật dữ liệu (*Data Engineering*), kết hợp sức mạnh của kiến trúc hướng sự kiện (*Event-driven Architecture*) với mô hình **Modern Data Stack**.

Đề tài tập trung giải quyết bài toán cốt lõi thông qua việc ứng dụng **Apache Kafka** để truyền tải thông điệp với độ trễ thấp, kết hợp cùng hệ thống lưu trữ đối tượng **MinIO** (*Data Lake*) và kho dữ liệu đám mây **Google BigQuery** (*Data Warehouse*). Đặc biệt, hệ thống áp dụng mô hình kiến trúc **Medallion (Bronze – Silver – Gold)** và quy trình biến đổi dữ liệu bằng **dbt**, giúp chuyển hóa các bản ghi thô thành những chỉ số phân tích kỹ thuật có giá trị cao như RSI, ATR và SMA. Toàn bộ quy trình này được điều phối tự động bởi **Apache Airflow**, đảm bảo tính toàn vẹn, khả năng tự phục hồi và tính sẵn sàng cao của dữ liệu.

Hệ thống không chỉ dừng lại ở một công cụ kỹ thuật mà còn hướng tới mục tiêu cung cấp một nền tảng tri thức đáng tin cậy cho các công cụ trực quan hóa như **Grafana**, giúp người dùng nắm bắt nhịp đập thị trường một cách trực quan và khoa học nhất. Thông qua đề tài này, chúng em mong muốn đóng góp một mô hình tham chiếu hiệu quả trong việc xử lý luồng dữ liệu tài chính, đồng thời củng cố kiến thức về các công nghệ tiên tiến trong lĩnh vực khoa học dữ liệu.

Chúng em xin chân thành cảm ơn sự hướng dẫn của thầy **TS. Nguyễn Đức Anh** và sự hỗ trợ từ các nguồn tài liệu chuyên môn đã giúp chúng em hoàn thành đề tài này.

Sinh viên thực hiện đề tài

1 Giới thiệu chung

1.1 Lý do chọn đề tài

Trong kỷ nguyên tài chính số, dữ liệu được coi như một phần đặc biệt quan trọng của thị trường chứng khoán. Khả năng tiếp cận và phân tích dữ liệu nhanh chóng chính là yếu tố cốt lõi tạo nên lợi thế cạnh tranh của các nhà đầu tư. Tuy nhiên, việc theo dõi thủ công các chỉ số trên bảng điện tử không chỉ gây tốn thời gian mà còn dễ dẫn đến sai sót do yếu tố tâm lý và sự chậm trễ trong việc cập nhật thông tin.

Bên cạnh nhu cầu về thông tin, thách thức lớn nhất trong việc xây dựng hệ thống tài chính hiện nay chính là khả năng xử lý khối lượng dữ liệu khổng lồ (Big Data) một cách linh hoạt. Các kiến trúc truyền thống thường gặp khó khăn trong việc mở rộng và duy trì tính ổn định khi tần suất dữ liệu thay đổi liên tục. Do đó, việc nghiên cứu và áp dụng mô hình **Modern Data Stack** — kết hợp giữa khả năng truyền tải thông điệp của **Apache Kafka**, lưu trữ của **Data Lake** và sức mạnh xử lý của **Cloud Data Warehouse** — không chỉ giúp tối ưu hóa hiệu suất mà còn đảm bảo tính sẵn sàng cao của dữ liệu. Việc thực hiện đề tài này là cơ hội để tiếp cận với những công nghệ xử lý dữ liệu tiên tiến nhất, đáp ứng các tiêu chuẩn khắt khe về độ trễ và độ tin cậy trong ngành phân tích chứng khoán hiện đại.

Sự ra đời của **Hệ thống thu thập, lưu trữ và xử lý dữ liệu chứng khoán thời gian thực** giải quyết bài toán cấp thiết về việc tự động hóa hoàn toàn quy trình thu thập và xử lý dữ liệu. Hệ thống giúp loại bỏ các thao tác thủ công, đảm bảo dữ liệu được cập nhật liên tục với độ trễ thấp nhất. Điều này không chỉ cung cấp cái nhìn tức thời về biến động thị trường mà còn tạo nền tảng vững chắc để thực hiện các phân tích lịch sử chuyên sâu, từ đó giúp nhà đầu tư đưa ra các quyết định chính xác và kịp thời hơn.

1.2 Mục tiêu của hệ thống

Hệ thống được thiết kế với các mục tiêu trọng tâm sau:

- **Thu thập dữ liệu thời gian thực (Real-time Ingestion):** Tự động kết nối và lấy dữ liệu liên tục từ nguồn API tài chính bên ngoài (Finnhub API) với độ trễ thấp nhất có thể.
- **Lưu trữ dữ liệu (Storage):** Xây dựng cơ chế lưu trữ dữ liệu thô (Raw data) để đảm bảo khả năng truy vết (traceability) và phục hồi khi có sự cố, đồng thời tổ chức dữ liệu theo kiến trúc nhiều lớp (Medallion Architecture) để tối ưu hóa truy vấn.
- **Xử lý và biến đổi dữ liệu (Process & Transform):** Chuyển đổi các bản tin dữ liệu từ định dạng thô sang các định dạng đã được chuẩn hóa, phù hợp cho việc tính toán các chỉ số kỹ thuật và phân tích xu hướng.
- **Tự động hóa luồng công việc (Orchestration):** Ứng dụng Apache Airflow để lập lịch, quản lý và vận hành tự động toàn bộ chuỗi cung ứng dữ liệu (Data Pipeline). Đảm bảo các tiến trình từ thu thập, chuyển đổi (dbt) đến kiểm tra chất lượng dữ liệu được thực hiện đúng trình tự và có khả năng tự phục hồi khi gặp lỗi.
- **Triển khai (Containerization):** Sử dụng công nghệ Docker để đóng gói toàn bộ các thành phần

của hệ thống (Kafka, Airflow, MinIO...). Mục tiêu là tạo ra một môi trường triển khai đồng nhất, dễ dàng quản lý tài nguyên và có khả năng mở rộng nhanh chóng trên các hạ tầng khác nhau.

- **Trực quan hóa (Visualization):** Cung cấp giao diện biểu đồ trực quan, giúp người dùng nắm bắt nhanh chóng các biến động của thị trường thông qua các chỉ số đã được xử lý.

1.3 Đối tượng và Phạm vi

Đối tượng nghiên cứu:

Hệ thống tập trung xử lý dữ liệu của 5 mã chứng khoán công nghệ hàng đầu thế giới (thường được gọi là nhóm Big Tech) để đảm bảo tính đại diện và khối lượng dữ liệu lớn:

- AAPL (Apple Inc.)
- MSFT (Microsoft Corporation)
- TSLA (Tesla, Inc.)
- GOOGL (Alphabet Inc. - Google)
- AMZN (Amazon.com, Inc.)

Phạm vi và Cấu trúc dữ liệu:

Dữ liệu được thu thập từ API Finnhub dưới dạng cấu trúc JSON. Mỗi bản ghi bao gồm các trường thông tin kỹ thuật quan trọng:

- **Các chỉ số giá:** Giá hiện tại (c), Giá mở cửa (o), Giá cao nhất (h), Giá thấp nhất (l), Giá đóng cửa phiên trước (pc).
- **Biến động:** Mức thay đổi tuyệt đối (d) và Phần trăm thay đổi (dp).
- **Định danh thời gian:** Timestamp thị trường (t) và thời điểm hệ thống thu thập (fetched_at).

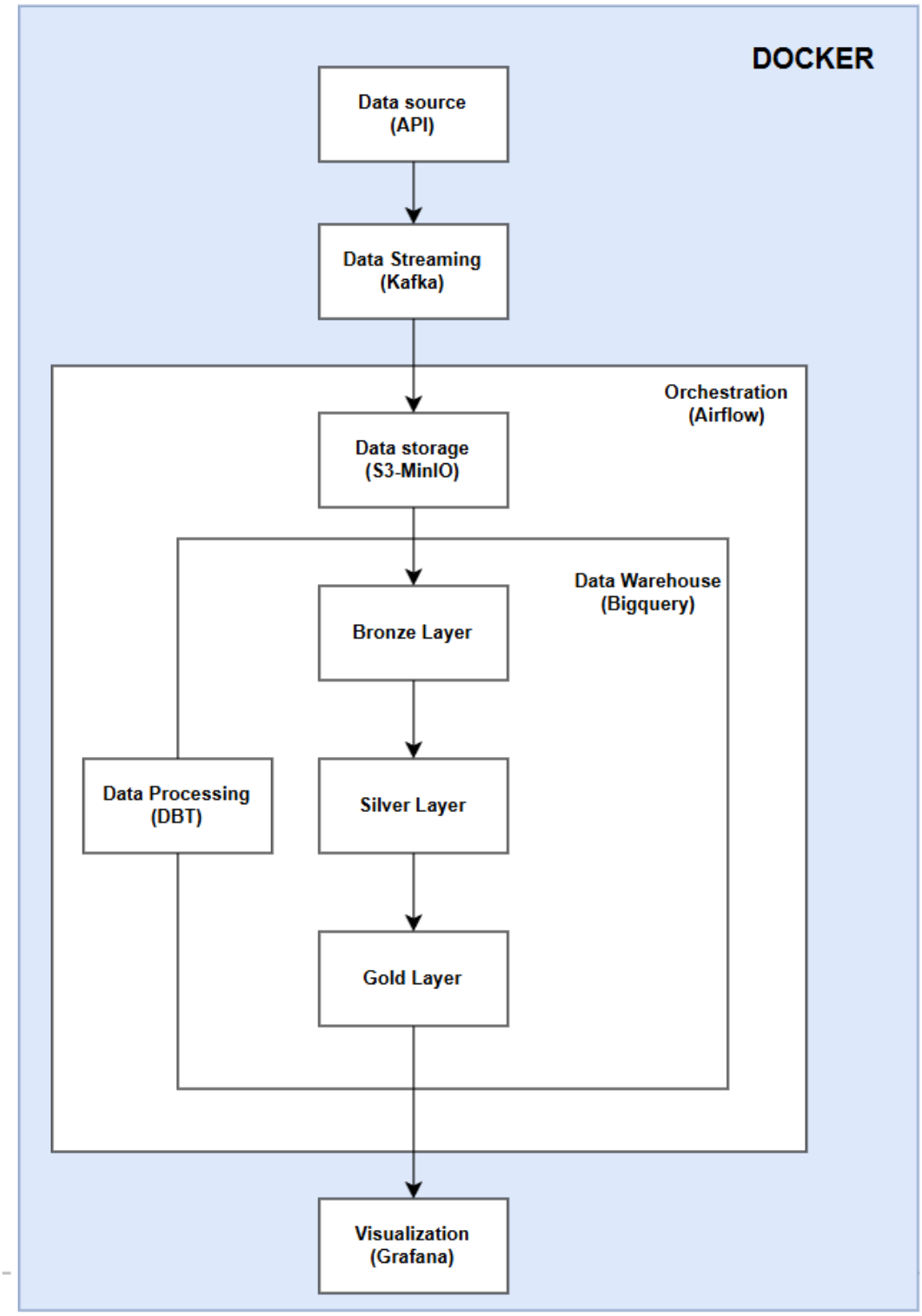
Ví dụ về một bản ghi dữ liệu thô:

```
{
  'symbol': 'GOOGL', 'c': 337.36, 'd': 5.5, 'dp': 1.6573,
  'h': 340.49, 'l': 333.33, 'o': 333.33, 'pc': 331.86,
  't': 1768320685, 'fetched_at': 1768320716
}
```

Phạm vi công nghệ:

Hệ thống triển khai dựa trên mô hình Modern Data Stack bao gồm: Kafka (Streaming), MinIO (Data Lake), BigQuery (Data Warehouse), dbt (Transform), và Airflow (Orchestration), tất cả vận hành trong môi trường Docker.

2 Kiến trúc hệ thống



Hình 1: Kiến trúc tổng quan

2.1 Tổng quan

Hệ thống Real-Time Stock Analytics được xây dựng dựa trên sự kết hợp giữa kiến trúc hướng sự kiện (*Event-driven Architecture*) và mô hình xử lý dữ liệu hiện đại (*Modern Data Stack*). Toàn bộ hệ thống vận hành trong môi trường Docker, giúp đảm bảo tính đóng gói, khả năng mở rộng và nhất quán trong quá trình triển khai.

Kiến trúc này được thiết kế theo luồng xử lý dữ liệu khép kín, tích hợp cả hai phương thức xử lý luồng (*Streaming*) và xử lý theo lô (*Batch Processing*). Trung tâm của hệ thống là nền tảng Apache Kafka, đóng vai trò là "xương sống" để truyền tải dữ liệu từ các nguồn API bên ngoài với độ trễ thấp. Sau đó, dữ liệu được chuyển tiếp vào hệ thống lưu trữ đối tượng MinIO (*Data Lake*) và kho dữ liệu Google BigQuery (*Data Warehouse*).

Điểm cốt lõi của hệ thống là việc áp dụng kiến trúc Medallion (*Bronze - Silver - Gold*). Quy trình chuyển đổi dữ liệu qua các tầng này được điều phối bởi Apache Airflow và thực thi thông qua công cụ dbt (*data build tool*). Cách tiếp cận này giúp phân tách rõ ràng giữa dữ liệu thô ban đầu và dữ liệu đã qua tinh chế, từ đó cung cấp các chỉ số tài chính có độ tin cậy cao cho tầng trực quan hóa trên Grafana.

Việc sử dụng Airflow làm lớp quản lý (*Orchestration Layer*) bao bọc các tiến trình xử lý trong *Data Warehouse* cho phép hệ thống tự động hóa hoàn toàn chuỗi cung ứng dữ liệu, từ khâu thu thập đến khâu trình diễn báo cáo.

2.2 Các thành phần chính

2.2.1 Tầng Thu Thập Dữ Liệu (Data Ingestion)

Đây là điểm khởi đầu của luồng dữ liệu, vận hành theo cơ chế hướng sự kiện (*Event-driven*):

- **Producer:** Một *Python script* thực hiện nhiệm vụ truy vấn *Finnhub API*. Cứ mỗi 6 giây, *Producer* sẽ lấy dữ liệu của 5 mã chứng khoán trọng điểm (AAPL, MSFT, TSLA, GOOGL, AMZN) và đẩy (*publish*) vào *Kafka topic* có tên `stock-quotes`.
- **Kafka Broker:** Đóng vai trò là hệ thống vận chuyển tin nhắn trung gian. Với cơ chế *Consumer Group* (`bronze-Consumer`), Kafka đảm bảo tính toàn vẹn của dữ liệu thông qua chính sách giao hàng ít nhất một lần (*at-least-once delivery*), tránh việc mất mát dữ liệu khi hệ thống gặp sự cố.
- **Consumer:** *Script Python* lắng nghe các bản tin từ Kafka, sau đó thực hiện ghi dữ liệu xuống MinIO. Dữ liệu được tổ chức dưới dạng file JSON với cấu trúc cây thư mục tối ưu cho việc truy xuất: `{symbol}/{timestamp}.json`.

2.2.2 Tầng Lưu Trữ Dữ Liệu (Temporary Storage)

MinIO: Sử dụng hệ thống lưu trữ đối tượng tương thích S3 để khởi tạo *bucket* `bronze-transactions`. Tầng này đóng vai trò là một *Data Buffer* (vùng đệm), giúp tách biệt hoàn toàn tốc độ ghi dữ liệu cực nhanh của quá trình *Ingestion* và tốc độ xử lý định kỳ của quá trình *Batch Processing* phía sau.

2.2.3 Tầng Điều Phối (Orchestration)

Apache Airflow: Giữ vai trò "nhạc trưởng" điều phối toàn bộ luồng dữ liệu. *DAG* `minio_to_bigquery_multi` được thiết lập chạy định kỳ mỗi 1 phút.

Cơ chế xử lý: Airflow xử lý song song 5 mã cổ phiếu. Với mỗi mã, hệ thống thực hiện 2 tác vụ (*tasks*) chính: theo dõi siêu dữ liệu (*metadata tracking*) để xác định file mới và thực hiện nạp dữ liệu tăng trưởng (*Incremental Load*) từ MinIO vào BigQuery.

2.2.4 Tầng Kho Dữ Liệu (Data Warehouse - BigQuery)

Dữ liệu bắt đầu bước vào quy trình quản lý chất lượng tại BigQuery:

- **Bronze Layer:** Bảng `bronze_stock_quotes_raw` chứa toàn bộ dữ liệu thô được đổ vào từ MinIO.
- **Metadata Tracking:** Bảng `metadata_last_ts` ghi lại dấu thời gian (*timestamp*) cuối cùng đã xử lý, đảm bảo luồng Airflow không nạp trùng lặp dữ liệu trong các chu kỳ sau.

2.2.5 Tầng Chuyển Đổi Dữ Liệu (dbt - Data Build Tool)

Tại đây, mô hình *Medallion* được hiện thực hóa thông qua các câu lệnh SQL mã hóa trong dbt:

- **Bronze (Views):** Thực hiện ánh xạ dữ liệu, chuyển đổi tên các cột từ định dạng viết tắt của API (`c, d, dp, o, h, l`) sang tên định dạng chuẩn có ý nghĩa nghiệp vụ (`current_price, change_amount, change_percent...`).
- **Silver (Incremental Tables):** Thực hiện chuẩn hóa kiểu dữ liệu (*Type casting*), kiểm tra tính hợp lệ (*Validation*: giá phải dương, giá cao nhất phải lớn hơn giá thấp nhất), loại bỏ dữ liệu trùng (*Deduplication*) và đồng bộ múi giờ quốc tế (UTC) sang giờ địa phương (US timezone).
- **Gold (Analytics Tables):** Các bảng tổng hợp cấp cao gồm `gold_kpi` (chỉ số mới nhất), `gold_kpi_history` (lịch sử chỉ số), `gold_candlestick` (dữ liệu biểu đồ nến theo ngày) và `gold_treechart` phục vụ phân tích biến động.

2.2.6 Tầng Trực Quan Hóa (Visualization)

Grafana: Kết nối trực tiếp với BigQuery để truy vấn dữ liệu từ các bảng *Gold Layer*.

Hệ thống Dashboard: Trình diễn các chỉ số tài chính theo thời gian thực (*Real-time metrics*). Người dùng có thể theo dõi biểu đồ nến (*Candlestick*), phân tích độ biến động (*Volatility Analysis*) thông qua *Tree chart*, cùng các chỉ báo kỹ thuật quan trọng như RSI, ATR và các đường trung bình động (*Moving Averages*).

3 Thiết kế và Triển khai chi tiết

3.1 Tầng Thu Thập Dữ Liệu (Data Ingestion)

Đây là điểm khởi đầu của luồng dữ liệu, vận hành theo cơ chế hướng sự kiện (*Event-driven*). Tầng này chịu trách nhiệm đưa dữ liệu từ nguồn phát sinh bên ngoài vào hệ thống nội bộ một cách an toàn và liên tục.

3.1.1 Thành Phần 1: Kafka Producer

Vai Trò và Chức Năng:

Producer là thành phần chịu trách nhiệm thu thập dữ liệu từ nguồn bên ngoài (Finnhub API) và *publish* vào Kafka topic. Producer hoạt động như một *data source adapter*, chuyển đổi dữ liệu từ định dạng HTTP API sang định dạng *message queue*.

Quy Trình Hoạt Động:

1. **Khởi Tạo và Cấu Hình:**
   - Producer kết nối với Kafka broker tại `localhost:29092`.
   - Sử dụng *JSON serializer* để chuyển đổi Python dictionary sang UTF-8 bytes.
   - Topic `stock-quotes` được tạo tự động khi producer gửi message đầu tiên.
2. **Thu Thập Dữ Liệu:**
   - Producer thực hiện vòng lặp vô hạn, duyệt qua 5 mã chứng khoán: AAPL, MSFT, TSLA, GOOGL, AMZN.
   - Gọi API Finnhub endpoint `/api/v1/quote` để lấy dữ liệu cho từng symbol.
   - API trả về JSON với các trường: `c` (giá hiện tại), `d` (biến động), `dp` (% biến động), `h` (cao nhất), `l` (thấp nhất), `o` (mở cửa), `pc` (đóng cửa phiên trước), `t` (timestamp thị trường).
3. **Làm giàu dữ liệu (Data Enrichment):**
   - Producer bổ sung hai trường metadata: `symbol` (mã chứng khoán) và `fetched_at` (thời điểm thu thập).
   - Việc này hỗ trợ truy vết (*tracking*) và kiểm thử (*debugging*) trong các tầng sau.
4. **Kiểm soát lưu lượng (Rate Limiting):**
   - Producer nghỉ 6 giây giữa mỗi chu kỳ. Với 5 mã chứng khoán, hệ thống thực hiện khoảng 50 calls/phút, đảm bảo nằm trong giới hạn miễn phí của Finnhub (60 calls/phút).
5. **Xử lý lỗi và Phát hành:**
   - Sử dụng khối `try-catch` để xử lý lỗi API; nếu một mã bị lỗi, hệ thống sẽ tiếp tục mã tiếp theo.
   - Message được gửi vào Kafka theo cơ chế *fire-and-forget* để tối ưu hiệu suất (*throughput*).

3.1.2 Thành Phần 2: Apache Kafka Broker

Vai Trò và Chức Năng:

Kafka đóng vai trò là *message broker* và *event store*, tạo ra một lớp trừu tượng giữa producer và consumer.

Kafka lưu trữ messages trong các *log files* với khả năng lưu trữ bền vững (*persistence*).

Cấu Hình Cơ Sở Hạ Tầng:

- **Zookeeper Dependency:** Kafka phụ thuộc vào Zookeeper để quản lý metadata và điều phối cụm (*broker coordination*).
- **Network Configuration:** Cấu hình hai listeners: `9092` cho nội bộ Docker và `29092` cho kết nối từ máy chủ bên ngoài.
- **Topic Configuration:** Topic `stock-quotes` được thiết lập với 1 *partition* và *replication factor* bằng 1 (phù hợp với môi trường phát triển).

Đặc Điểm Hoạt Động:

- **Message Persistence:** Lưu dữ liệu xuống đĩa cứng, cho phép consumer đọc lại dữ liệu từ bất kỳ thời điểm nào (*offset*).
- **Consumer Group Management:** Đảm bảo mỗi thông điệp chỉ được xử lý một lần bởi mỗi nhóm consumer.
- **Monitoring:** Sử dụng *Kafdrop* (Web UI) để theo dõi các topic, thông điệp và độ trễ (*lag*) của consumer.

3.1.3 Thành Phần 3: Kafka Consumer

Vai Trò và Chức Năng:

Consumer đọc dữ liệu từ Kafka topic và ghi vào hệ thống lưu trữ MinIO. Đây là thành phần chuyển đổi định dạng từ *message queue* sang *object storage*.

Quy Trình Hoạt Động:

- **Cấu Hình:** Sử dụng group ID `bronze-Consumer`. Thiết lập `auto_offset_reset="earliest"` để đảm bảo không bỏ lỡ dữ liệu cũ nếu consumer mới khởi động lần đầu.
- **Xử Lý Dữ Liệu:** Trích xuất `symbol` và `fetched_at` để tạo khóa lưu trữ (*storage key*) theo định dạng: `{symbol}/{timestamp}.json`.
- **Lưu Trữ MinIO:** Sử dụng thư viện `Boto3` để kết nối với S3 API của MinIO, ghi mỗi thông điệp thành một tệp JSON riêng biệt trong *bucket* `bronze-transactions`.

Đặc Điểm Thiết Kế:

- **At-Least-Once Delivery:** Đảm bảo dữ liệu không bị mất. Nếu consumer gặp sự cố trước khi xác nhận (*commit offset*), thông điệp sẽ được xử lý lại khi hệ thống phục hồi.
- **Idempotency:** Việc ghi đè tệp có cùng tên (nếu trùng lặp) giúp đảm bảo tính nhất quán của dữ liệu tại tầng Bronze.
- **Scalability:** Khả năng mở rộng bằng cách thêm nhiều *instances* vào cùng một nhóm consumer để chia sẻ tải xử lý.

3.1.4 Biến Đổi Định Dạng Dữ Liệu (Data Format Transformation)

Dữ liệu trải qua các giai đoạn chuyển đổi logic như sau:

1. **Nguồn (Finnhub):** Định dạng JSON thô với các trường viết tắt (`c, d, dp...`).
2. **Producer:** Làm giàu dữ liệu bằng cách thêm `symbol` và `fetched_at`.

3. **Kafka:** Chuyển đổi sang dạng nhị phân (*UTF-8 bytes*) để truyền tải.
4. **Consumer:** Chuyển đổi ngược từ bytes sang Python dictionary để xử lý.
5. **Lưu trữ (MinIO):** Lưu thành tệp vật lý với cấu trúc phân cấp thư mục để tối ưu hóa việc truy xuất theo mã chứng khoán và thời gian.

3.2 Tầng lưu trữ dữ liệu

3.2.1 Tổng Quan

Tầng 2 là tầng lưu trữ tạm thời, sử dụng MinIO làm *object storage* để lưu trữ dữ liệu thô trước khi được xử lý và nạp vào *data warehouse*. Tầng này đóng vai trò là *buffer layer* (tầng đệm) và *decoupling point* (điểm tách rời) giữa quá trình thu thập dữ liệu thời gian thực (từ Kafka Consumer) và quá trình xử lý theo lô (bởi Airflow DAG). MinIO cung cấp khả năng lưu trữ dữ liệu dưới dạng tệp JSON riêng biệt, cho phép hệ thống thực hiện nạp dữ liệu tăng trưởng (*incremental loading*) và xử lý lại dữ liệu khi cần thiết.

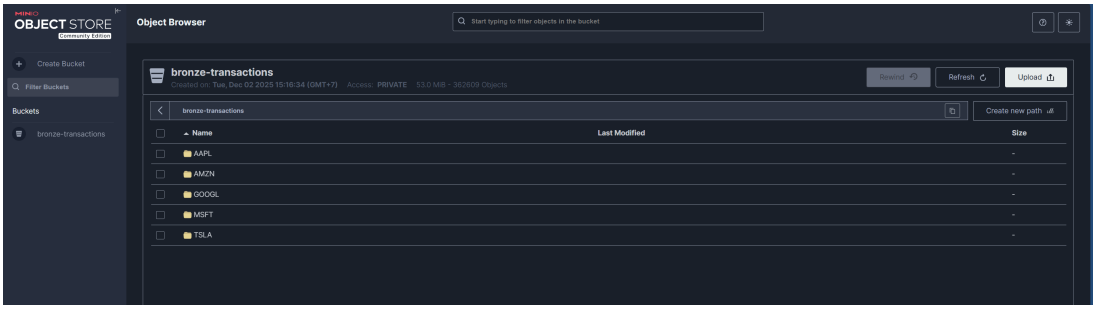
3.2.2 Thành Phần: MinIO Object Storage

Vai Trò và Chức Năng:

MinIO là một *object storage server* tương thích hoàn toàn với Amazon S3 API. Trong hệ thống này, MinIO thực hiện các chức năng cốt lõi sau:

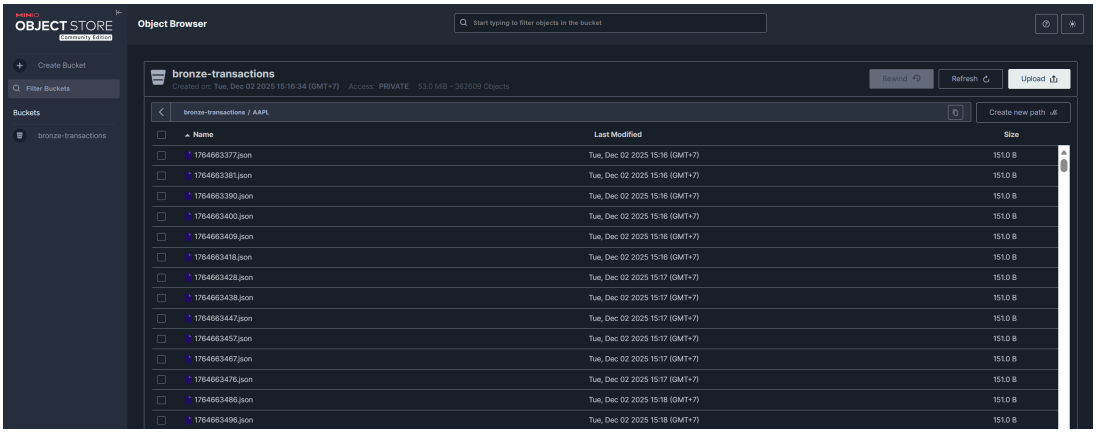
1. **Buffer Layer:** Lưu trữ dữ liệu từ Consumer trước khi Airflow xử lý, cho phép hai quá trình hoạt động với tốc độ khác nhau. Consumer có thể ghi nhanh (50 messages/phút) mà không cần đợi các tiến trình phía sau.
2. **Decoupling Point:** Tách biệt hoàn toàn quá trình *ingestion* và *processing*. Consumer không cần nhận biết về các hệ thống hạ nguồn như BigQuery hay Airflow.
3. **Data Persistence:** Lưu trữ dữ liệu thô dưới dạng tệp JSON bền vững, cho phép tái lập (*replay*) dữ liệu để xử lý lại hoặc phục hồi sau sự cố.
4. **Incremental Processing Support:** Cấu trúc khóa (*key*) cho phép lọc và sắp xếp dữ liệu dễ dàng, hỗ trợ Airflow chỉ đọc các tệp mới dựa trên dấu thời gian.

3.2.3 Cấu Trúc Lưu Trữ



Hình 2: Cấu trúc lưu trữ trong MinIO

Tổ chức Bucket: `bronze-transactions` được tạo tự động bởi Consumer trong lần kết nối đầu tiên (*idempotent operation*), chứa toàn bộ dữ liệu thô từ Kafka.



Hình 3: File json trong thư mục AAPL

Cấu trúc Khóa (Key Structure):

- **Định dạng:** `{symbol}/{timestamp}.json`
- **Lợi ích:**
  - *Partitioning by Symbol:* Truy vấn và lọc theo mã chứng khoán bằng tiền tố (*prefix*).
  - *Time-based Sorting:* Dấu thời gian trong tên tệp cho phép sắp xếp tự nhiên.
  - *Unique Keys:* Tránh xung đột khi xử lý trùng lặp.

Nội dung tệp (File Content): Mỗi tệp JSON chứa một bản ghi *quote* cho một mã chứng khoán tại một thời điểm, bao gồm các trường từ API và siêu dữ liệu từ Producer với định dạng `application/json`.

3.2.4 Quy Trình Ghi Dữ Liệu (Write Path)

Consumer nhận thông điệp từ Kafka topic `stock-quotes`, trích xuất `symbol` và `fetched_at` để tạo khóa `{symbol}/{timestamp}.json`. Sau đó, dữ liệu được tuần tự hóa thành JSON và tải lên MinIO thông qua *Boto3 S3 client*.

- **Bất đồng bộ (Asynchronous):** Consumer không đợi Airflow xử lý, chỉ tập trung vào việc lưu tệp thành công.
- **Hiệu suất cao:** MinIO xử lý 50 tệp/phút một cách dễ dàng, không gây nghẽn cổ chai.
- **Bền vững (Durability):** Tệp được ghi trực tiếp xuống đĩa cứng, đảm bảo an toàn ngay cả khi container khởi động lại.

Đặc điểm của quy trình ghi dữ liệu:

- **Bất đồng bộ (Asynchronous):** *Consumer* hoạt động độc lập và không đợi *Airflow* xử lý; nhiệm vụ duy nhất là đảm bảo tệp tin được lưu trữ thành công vào *Object Storage*.
- **Thông lượng cao (High Throughput):** MinIO có khả năng xử lý 50 tệp/phút một cách dễ dàng, đảm bảo không trở thành điểm nghẽn (*bottleneck*) trong toàn bộ luồng dữ liệu.
- **Tính bền vững (Durability):** Các tệp tin được ghi trực tiếp vào đĩa cứng ngay lập tức, đảm bảo tính duy trì (*persistence*) ngay cả khi các *container* hệ thống khởi động lại.

- **Phục hồi lỗi (Error Recovery):** Nếu hệ thống MinIO tạm thời ngưng hoạt động, các thông điệp dữ liệu vẫn được lưu giữ an toàn trong *Kafka* để thực hiện cơ chế thử lại (*retry*) sau khi dịch vụ phục hồi.

3.2.5 Quy Trình Đọc Dữ Liệu (Read Path)

Tiến trình này được điều phối bởi Airflow DAG thông qua các bước:

1. **Truy vấn Metadata:** Task `download_{symbol}` truy vấn bảng `metadata_last_ts` trong BigQuery để lấy dấu thời gian xử lý cuối cùng.
2. **Lọc dữ liệu:** Liệt kê các đối tượng trong bucket với tiền tố `{symbol}/`, thực hiện phân trang (*pagination*) và chỉ lấy các tệp có `timestamp > last_ts`.
3. **Tải về và Xử lý:** Tải các tệp về thư mục tạm `/tmp/minio_downloads`. Sau khi nạp vào BigQuery thành công, các tệp tạm thời sẽ bị xóa để tối ưu bộ nhớ local.

Đặc điểm của quy trình đọc dữ liệu:

- **Xử lý tăng trưởng (Incremental Processing):** Hệ thống chỉ thực hiện xử lý các tệp mới dựa trên cơ chế *metadata tracking*. Điều này giúp giảm thiểu đáng kể lưu lượng băng thông mạng (*network traffic*) và tối ưu hóa thời gian xử lý của hệ thống.
- **Xử lý phân trang (Pagination Handling):** Tự động quản lý việc phân trang thông qua `ContinuationToken` trong trường hợp danh mục lưu trữ có chứa hơn 1.000 đối tượng (*objects*), đảm bảo không bỏ sót dữ liệu.
- **Đảm bảo thứ tự (Ordering Guarantee):** Các tệp dữ liệu được sắp xếp (*sort*) theo dấu thời gian (*timestamp*) trước khi thực hiện bộ lọc (*filter*), đảm bảo dữ liệu luôn được xử lý đúng theo trình tự thời gian thực tế phát sinh.
- **Phục hồi lỗi (Error Recovery):** Trong trường hợp tác vụ tải dữ liệu (*download task*) thất bại, hệ thống sẽ thực hiện cơ chế thử lại (*retry*). Do dữ liệu vẫn được lưu trữ bền vững trên MinIO, các tệp này có thể được tải lại một cách an toàn mà không gây mất mát thông tin.

3.2.6 Đặc Điểm Kiến Trúc (Buffer Pattern)

Cơ chế đệm của MinIO giải quyết hai vấn đề tách rời quan trọng:

- **Time Decoupling:** Consumer ghi dữ liệu mỗi 6 giây (thời gian thực) trong khi Airflow đọc và xử lý mỗi 1 phút (theo lô).
- **Rate Decoupling:** Consumer có thể ghi với tốc độ cao mà không bị ảnh hưởng bởi tốc độ xử lý của Airflow. Nếu Airflow tạm dừng, MinIO sẽ lưu trữ dữ liệu chờ để Airflow xử lý bù (*catch up*) sau đó.

3.3 Tầng điều phối

3.3.1 Tổng Quan

Tầng 3 là tầng điều phối, sử dụng Apache Airflow để quản lý và thực thi các workflow ETL (*Extract, Transform, Load*) một cách có lịch trình và đáng tin cậy. Tầng này chịu trách nhiệm điều phối quá

trình *batch processing*, thực hiện *incremental load* từ MinIO vào BigQuery, và quản lý *metadata* để đảm bảo tính nhất quán và không trùng lặp dữ liệu. Airflow cung cấp khả năng *monitoring, retry*, và *error handling* cho toàn bộ pipeline.

3.3.2 Vai Trò và Chức Năng

Trong hệ thống này, Airflow đóng vai trò:

1. **Workflow Orchestration:** Định nghĩa và quản lý các DAG (*Directed Acyclic Graph*), thực thi các tác vụ theo phụ thuộc (*dependency*) và lịch trình.
2. **Scheduling và Automation:** Chạy DAG định kỳ mỗi 1 phút, tự động kích hoạt workflow và quản lý lịch sử thực thi.
3. **Error Handling và Retry:** Tự động thử lại các tác vụ khi thất bại, cung cấp log chi tiết để phục vụ *debugging*.
4. **Monitoring và Observability:** Web UI cho phép giám sát trạng thái các lượt chạy, giá trị XCom và hiệu suất hệ thống.

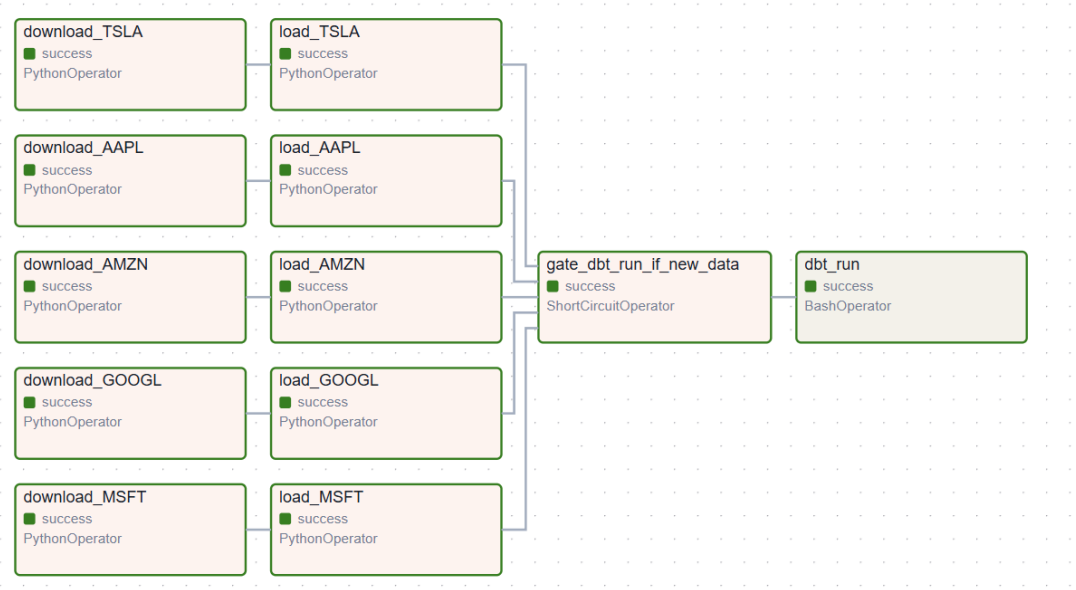
3.3.3 DAG: minio_to_bigquery_multi

Cấu Hình DAG:

- **Schedule Interval:** `* * * * *` (mỗi 1 phút).
- **Catchup:** `False` (không chạy lại các tiến trình đã lỡ).
- **Max Active Runs:** `1` (đảm bảo tính tuần tự khi thực hiện dbt run).
- **Default Arguments:** Khởi tạo từ 01/01/2025, thử lại 1 lần sau mỗi 2 phút nếu lỗi.

Cấu Trúc Tasks: DAG xử lý song song 5 mã chứng khoán (AAPL, MSFT, TSLA, GOOGL, AMZN).

- **Parallel Execution:** 5 biểu tượng chứng khoán được xử lý không phụ thuộc nhau.
- **Sequential per Symbol:** Task `download_{symbol}` phải hoàn thành trước `load_{symbol}`. Mối quan hệ được định nghĩa: `t1 >> t2`.



Hình 4: Cấu trúc của DAG

3.3.4 Chi tiết các Task

Task 1: download_{symbol}

- **Chức năng:** Tải tăng trưởng tệp từ MinIO về bộ nhớ cục bộ của Airflow server.
- **Quy trình:** Truy vấn BigQuery lấy `last_ts`, kết nối MinIO qua Boto3, lọc các tệp mới (`timestamp > last_ts`) và đẩy danh sách tệp vào XCom.

Task 2: load_{symbol}

- **Chức năng:** Nạp các tệp JSON vào bảng BigQuery Bronze.
- **Quy trình:** Đọc danh sách tệp từ XCom, thực hiện `insert_rows_json()` vào bảng `bronze_stock_quotes_raw`, cập nhật bảng `metadata_last_ts` bằng lệnh `MERGE` và xóa tệp tạm.

Task 3: gate_dbt_run_if_new_data

- **Chức năng:** `ShortCircuitOperator` kiểm tra sự hiện diện của dữ liệu mới.
- **Quy trình:** Tổng hợp số lượng tệp từ XCom của cả 5 mã. Trả về `True` để chạy dbt hoặc `False` để bỏ qua các bước sau.

Task 4: dbt_run

- **Chức năng:** `BashOperator` thực thi các biến đổi dữ liệu (Bronze → Silver → Gold).
- **Quy trình:** Di chuyển đến thư mục dự án dbt, chạy `dbt deps` và `dbt run -target prod`.

3.3.5 Metadata Tracking và Incremental Loading

Hệ thống sử dụng bảng `metadata_last_ts` trong BigQuery để theo dõi:

- **Schema:** Gồm hai trường `symbol` (STRING) và `last_ts` (INT64).
- **Cơ chế:**
  - *Get Last Timestamp:* Truy vấn BigQuery để lấy giá trị `last_ts` mới nhất của từng mã chứng khoán. Giá trị này được dùng làm bộ lọc thời gian khi quét tệp trên MinIO.

  - *Update Last Timestamp:* Sau khi nạp dữ liệu thành công, hệ thống sử dụng lệnh `MERGE` để cập nhật dấu thời gian lớn nhất vừa xử lý vào bảng metadata. Điều này đảm bảo tính *idempotent* (không nạp trùng nếu tác vụ chạy lại).
- **Lợi ích:** Tối ưu hiệu suất xử lý bằng cách tránh quét toàn bộ dữ liệu (*full scan*), đảm bảo không bỏ sót tệp tin và ngăn chặn việc trùng lặp dữ liệu trong kho.

3.3.6 Đặc điểm Kiến trúc

1. Xử lý Song song (Parallel Processing)

- **Symbol-level Parallelism:** 5 mã chứng khoán được xử lý song song. Mỗi mã có 2 tác vụ riêng biệt, hoàn toàn không phụ thuộc vào các mã khác. *Airflow Scheduler* sẽ tự động thực thi song song các tác vụ độc lập này.
- **Lợi ích:**
  - *Performance:* Tốc độ xử lý nhanh hơn đáng kể so với xử lý tuần tự.
  - *Resource Utilization:* Tận dụng tối đa tài nguyên hệ thống của Airflow.
  - *Fault Isolation:* Nếu việc xử lý một mã bị lỗi, các mã chứng khoán khác vẫn tiếp tục mà không bị ảnh hưởng.

2. Xử lý Tăng trưởng (Incremental Processing)

- **Timestamp-based Filtering:** Sử dụng *metadata* để theo dõi tiến độ. Hệ thống chỉ xử lý các tệp mới hơn dấu thời gian cuối cùng ghi nhận, giúp tránh việc quét lại toàn bộ *bucket* mỗi lần chạy.
- **Lợi ích:** Giảm thời gian xử lý và lưu lượng mạng, đồng thời tối ưu hóa chi phí truy vấn trên BigQuery.

3. Tính Idempotency (Khả năng thực hiện lại)

- **Metadata-based Idempotency:** Việc theo dõi *metadata* đảm bảo mỗi tệp chỉ được xử lý đúng một lần. Nếu một lượt chạy DAG bị lỗi và phải thử lại, các tệp đã xử lý thành công sẽ không bị nạp lại. Câu lệnh `MERGE` khi cập nhật *metadata* đảm bảo tính nhất quán này.
- **Lợi ích:** Tăng độ tin cậy và đảm bảo dữ liệu trong BigQuery không bị trùng lặp khi có sự cố.

4. Xử lý Lỗi và Thử lại (Error Handling & Retry)

- **Cấu hình Thử lại:** Mỗi tác vụ được thiết lập `retries=1` và `retry_delay` là 2 phút. Nếu tác vụ thất bại, Airflow tự động thực thi lại sau khoảng thời gian chờ.
- **Khả năng Phục hồi:** Nếu bước tải dữ liệu (*load*) thất bại, *metadata* sẽ không được cập nhật. Điều này đảm bảo trong lần chạy tiếp theo, hệ thống sẽ nhận diện lại các tệp này để xử lý lại từ đầu.

3.3.7 Đặc tính Hiệu suất (Performance Characteristics)

Thời gian Thực thi (Execution Time)

- **Thông số lượt chạy điển hình:**
  - *Download tasks:* ∼5–15 giây (tùy số lượng tệp mới).
  - *Load tasks:* ∼10–30 giây (tùy số lượng bản ghi).
  - *Gate task:* <1 giây.

- *dbt run:* ∼30–60 giây.
- **Tổng cộng:** ∼1–2 phút cho một chu kỳ DAG đầy đủ.

- **Hiệu quả Song song:** Nhờ xử lý song song 5 mã chứng khoán, tổng thời gian thực thi xấp xỉ (≈) thời gian của mã chậm nhất cộng với thời gian chạy dbt. Phương pháp này hiệu quả gấp khoảng 5 lần so với xử lý tuần tự truyền thống.

3.4 Tầng Kho Dữ Liệu (Data Warehouse)

3.4.1 Tổng Quan

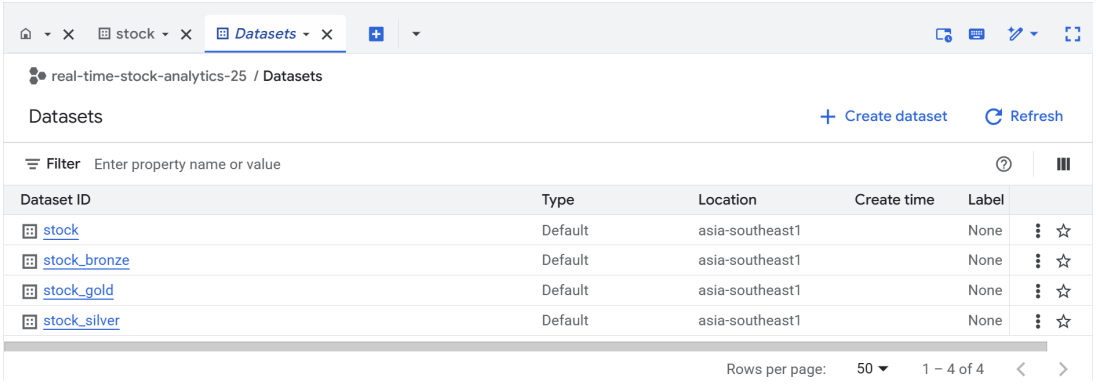
Tầng 4 là tầng data warehouse, sử dụng Google BigQuery làm nền tảng lưu trữ và xử lý dữ liệu chính của hệ thống. BigQuery cung cấp khả năng lưu trữ dữ liệu dưới dạng *columnar storage*, cho phép truy vấn và phân tích hiệu quả với khả năng *scale* tự động. Tầng này lưu trữ dữ liệu thô từ MinIO (*Bronze layer*), quản lý *metadata* để hỗ trợ *incremental loading*, và phục vụ như nền tảng cho các *transformations* (dbt) và *analytics*. BigQuery là một *managed service*, không yêu cầu quản lý hạ tầng cơ sở dữ liệu.

3.4.2 Thành Phần: Google BigQuery

Vai Trò và Chức Năng:

Google BigQuery là một *serverless, highly-scalable data warehouse* được thiết kế để xử lý và phân tích dữ liệu lớn. Trong hệ thống này, BigQuery đóng vai trò:

1. **Data Storage:**
   - Lưu trữ dữ liệu thô (*Bronze layer*) từ MinIO - table `bronze_stock_quotes_raw`.
   - Lưu trữ dữ liệu đã được làm sạch (*Silver layer*) - table `silver.silver_stock_quotes`.
   - Lưu trữ dữ liệu đã được tổng hợp (*Gold layer*) - các tables trong schema `gold`.
   - Định dạng *columnar storage* cho phép truy vấn hiệu quả cho tất cả các lớp.



Hình 5: Cấu trúc thư mục trong Big Query

2. **Query Engine:**
   - Xử lý SQL queries từ dbt transformations.
   - Analytics queries từ dashboards và reporting tools.
   - Xử lý truy vấn phân tán (*distributed query processing*) với khả năng tự động mở rộng.

3. **Metadata Management:**
   - Lưu trữ bảng *metadata* để theo dõi tiến trình nạp dữ liệu tăng trưởng (*incremental loading progress*).
   - Hỗ trợ các truy vấn *metadata* để điều phối quy trình ETL.
4. **Analytics Platform:**
   - Phục vụ như nền tảng cho *business intelligence* và *analytics*.
   - Kết nối với visualization tools như Grafana.
   - Hỗ trợ phân tích cả thời gian thực (*real-time*) và theo lô (*batch*).

3.4.3 Cấu Trúc Dữ Liệu

Dataset Organization:

Dataset `stock` được tổ chức theo mô hình *Medallion Architecture* với các schemas:

- **Bronze Schema:** Chứa *raw data tables* và *metadata*.
- **Silver Schema:** Chứa dữ liệu đã được làm sạch và xác thực (*cleaned và validated data*), được tạo bởi dbt.
- **Gold Schema:** Chứa dữ liệu đã tổng hợp và sẵn sàng cho kinh doanh (*aggregated và business-ready data*), được tạo bởi dbt.

3.4.4 Bronze Layer (Raw Data)

Bronze layer lưu trữ dữ liệu thô được load trực tiếp từ MinIO, không có transformation. Layer này bao gồm:

Table 1: `bronze_stock_quotes_raw`

Mục đích: Lưu trữ dữ liệu thô được nạp trực tiếp từ MinIO, giữ nguyên định dạng và cấu trúc từ API response.

Schema:

- `c` (FLOAT64): Current price
- `d` (FLOAT64): Change amount
- `dp` (FLOAT64): Change percent
- `h` (FLOAT64): Day high
- `l` (FLOAT64): Day low
- `o` (FLOAT64): Day open
- `pc` (FLOAT64): Previous close
- `t` (INT64): Market timestamp (Unix epoch seconds)
- `symbol` (STRING): Stock symbol (AAPL, MSFT, TSLA, GOOGL, AMZN)
- `fetched_at` (INT64): Fetch timestamp (Unix epoch seconds)

Đặc điểm:

- **Raw Data:** Giữ nguyên format từ API, không có transformation.
- **Append-only:** Dữ liệu được ghi thêm, không cập nhật hoặc xóa.
- **Columnar Storage:** Lưu trữ theo định dạng cột của BigQuery.

- **Compression:** Tự động nén để tối ưu chi phí lưu trữ.
- **No Partitioning:** Bảng hiện tại không được phân vùng (có thể tối ưu sau).

Data Loading: Dữ liệu được nạp từ Airflow DAG task `load_{symbol}`, sử dụng phương thức `insert_rows_json()` để nạp theo lô (*batch insert*). Mỗi lần chạy (1 phút) sẽ nạp các tệp mới từ MinIO, đảm bảo tính tăng trưởng và không trùng lặp.

Table 2: `metadata_last_ts`

Mục đích: Lưu trữ *metadata* để theo dõi timestamp cuối cùng đã được xử lý cho mỗi symbol, hỗ trợ cơ chế nạp tăng trưởng (*incremental loading mechanism*).

Schema:

- `symbol` (STRING): Stock symbol (primary key)
- `last_ts` (INT64): Timestamp cuối cùng đã được xử lý

Đặc điểm: Chỉ lưu thông tin theo dõi (*tracking information*), sử dụng logic `MERGE` để *upsert* (update hoặc insert) đảm bảo tính *idempotent*.

Cách dùng: Airflow task `download_{symbol}` truy vấn bảng này để lấy `last_ts` và task `load_{symbol}` sẽ cập nhật bảng sau khi nạp thành công.

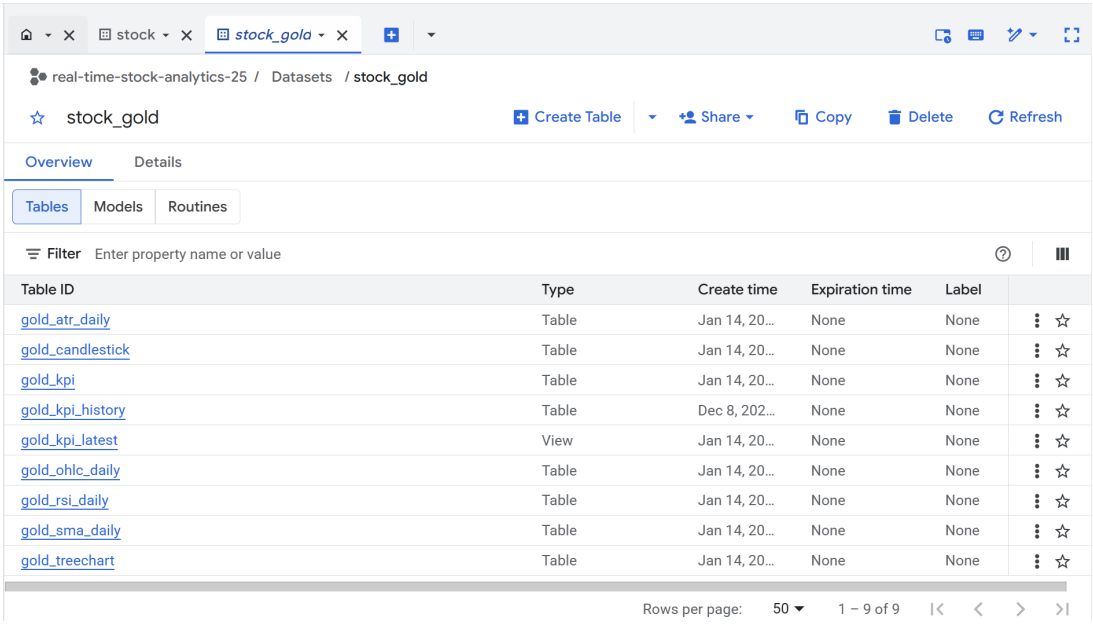
3.4.5 Silver Layer (Cleaned Data)

Silver layer chứa dữ liệu đã được làm sạch, xác thực và chuẩn hóa. Layer này được tạo bởi dbt và lưu trữ trong BigQuery schema `silver`.

Table: `silver_stock_quotes`

- **Materialization:** Incremental table với chiến lược *merge*.
- **Data Source:** Đọc từ Bronze view `bronze.bronze_stock_quotes`.
- **Transformations:** Ép kiểu (*Type casting*), chuyển đổi timestamp (UTC và US Eastern), xác thực (*validation*), và loại bỏ trùng lặp (*deduplication*).
- **Đặc điểm:** Dữ liệu sạch, không trùng lặp, đúng kiểu dữ liệu và nhận diện được múi giờ.
- **Unique key:** `[symbol, market_timestamp_raw]`.

3.4.6 Gold Layer (Curated Analytics)



Hình 6: Cấu trúc thư mục trong *gold layer*

Gold layer chứa các bảng phân tích sẵn sàng cho kinh doanh với các tổng hợp, KPIs và các thông số đo lường (*metrics*).

Các Tables trong Gold Schema:

1. `gold_kpi`**:** Lưu trữ KPIs mới nhất cho mỗi symbol, phục vụ real-time dashboards. (Materialization: Table).
2. `gold_kpi_history`**:** Toàn bộ lịch sử KPIs theo thời gian, hỗ trợ phân tích chuỗi thời gian. (Materialization: Incremental table).
3. `gold_candlestick`**:** Dữ liệu OHLC theo ngày cho 12 ngày gần nhất. (Materialization: Table).
4. `gold_treechart`**:** Phân tích biến động và giá trung bình cho biểu đồ *tree chart*. (Materialization: Table).
5. `gold_ohlc_daily`**:** Dữ liệu OHLC được tổng hợp theo ngày. Là bảng cơ sở để tính toán các chỉ số kỹ thuật (RSI, ATR, SMA).
6. `gold_rsi_daily`**:** Chỉ số RSI 14 ngày cho mỗi symbol. Phân vùng (*partition*) theo `trade_date` và cụm (*cluster*) theo `symbol`.
7. `gold_atr_daily`**:** Khoảng dao động thực tế trung bình (ATR) 14 ngày phục vụ quản trị rủi ro. Phân vùng theo `trade_date`.
8. `gold_sma_daily`**:** Đường trung bình động đơn giản (SMA) 20, 50, và 200 ngày để phân tích xu hướng và các tín hiệu *golden/death cross*.

Đặc điểm Gold Layer: Các bảng phân tích đã sẵn sàng cho kinh doanh, các chỉ số được tính toán trước (*pre-computed*) và tối ưu hóa cho việc tiêu thụ dữ liệu qua Dashboards.

3.5 Tầng chuyển đổi dữ liệu

3.5.1 Tổng Quan

Tầng 5 là tầng chuyển đổi dữ liệu, sử dụng **dbt (Data Build Tool)** để thực hiện các phép biến đổi dữ liệu từ định dạng thô (*Bronze*) sang định dạng đã làm sạch (*Silver*) và cuối cùng là các bảng phân tích sẵn sàng cho kinh doanh (*Gold*). Tầng này áp dụng mô hình **Medallion Architecture** với ba tầng rõ ràng, mỗi tầng có mục đích và logic biến đổi riêng biệt. dbt cho phép quản lý các phép biến đổi dữ liệu dưới dạng mã nguồn (SQL), kiểm soát phiên bản (*version control*), và tự động hóa quá trình xây dựng các mô hình dữ liệu. Tất cả các phép biến đổi được thực thi trực tiếp trong BigQuery, tận dụng sức mạnh xử lý truy vấn phân tán của nền tảng này.

3.5.2 Kiến Trúc Tổng Thể

Thành Phần: dbt (Data Build Tool)

Vai Trò và Chức Năng: dbt là một công cụ biến đổi dữ liệu cho phép các kỹ sư phân tích viết các phép biến đổi bằng ngôn ngữ SQL và quản lý chúng như mã nguồn phần mềm. Trong hệ thống này, dbt đóng vai trò:

- **Data Transformation:** Định nghĩa các SQL transformations để chuyển đổi dữ liệu qua các tầng (Bronze, Silver, Gold) và quản lý sự phụ thuộc (*dependencies*) giữa các mô hình.
- **Data Quality:** Thực thi các logic xác thực, làm sạch dữ liệu, ép kiểu (*type casting*) và loại bỏ trùng lặp (*deduplication*).
- **Business Logic:** Tính toán các chỉ số KPIs, các hàm cửa sổ (*window functions*) và triển khai các quy tắc nghiệp vụ.
- **Code Management:** Kiểm soát phiên bản với Git, quản lý mã nguồn theo mô-đun và hỗ trợ khả năng kiểm thử (*testing*).

3.5.3 Mô hình Kiến trúc Medallion (Medallion Architecture)

Tổng Quan về Medallion Pattern: Medallion Architecture là một mô hình thiết kế dữ liệu chia luồng dữ liệu thành ba lớp với mục đích rõ ràng:

- **Bronze (Raw):** Lưu trữ dữ liệu thô không thay đổi, giữ nguyên định dạng từ nguồn gốc. Tầng này chỉ thực hiện các thay đổi tối thiểu như đổi tên cột (*rename*) để dễ đọc, không làm sạch hay xác thực dữ liệu. Đây là nguồn sự thật (*source of truth*) đảm bảo có thể truy vết và tái thiết lập dữ liệu gốc nếu cần.
- **Silver (Cleaned):** Chứa dữ liệu đã được làm sạch, xác thực và chuẩn hóa. Tầng này thực hiện ép kiểu dữ liệu, loại bỏ các bản ghi trùng lặp và lọc bỏ dữ liệu không hợp lệ. Dữ liệu tại đây đã sẵn sàng cho phân tích chi tiết.
- **Gold (Curated):** Cung cấp các bảng phân tích sẵn sàng cho nghiệp vụ với dữ liệu đã được tổng hợp (*aggregated*) và tính toán KPIs. Các bảng ở đây được tối ưu hóa cho việc tiêu thụ dữ liệu bởi các công cụ trực quan hóa như Grafana.

Lợi Ích của Medallion Architecture:

- **Separation of Concerns:** Phân tách rõ ràng trách nhiệm giữa các lớp, giúp dễ dàng bảo trì, gỡ lỗi và truy xuất nguồn gốc dữ liệu (*data lineage*).
- **Flexibility:** Cung cấp tính linh hoạt cao, cho phép xây dựng lại dữ liệu từ bất kỳ lớp nào. Việc thay đổi ở một lớp không làm ảnh hưởng đến các lớp khác.
- **Data Quality:** Đảm bảo chất lượng dữ liệu được cải thiện dần dần qua từng lớp thông qua các bước xác thực và chuẩn hóa nghiêm ngặt.

3.5.4 Bronze Layer

Model: `bronze_stock_quotes`

Materialization: View

Chức năng: View này đọc từ bảng thô `bronze_stock_quotes_raw` và thực hiện đổi tên cột để tăng tính dễ hiểu.

- **Transformation:** `c` → `current_price`, `d` → `change_amount`, `dp` → `change_percent`, `h` → `day_high`, `l` → `day_low`, `o` → `day_open`, `pc` → `prev_close`, `t` → `market_timestamp`.
- **Đặc điểm:** Không thay đổi nội dung dữ liệu, truy vấn thời gian thực từ bảng gốc, không tốn chi phí lưu trữ (*View*).

3.5.5 Silver Layer

Model: `silver_stock_quotes`

Materialization: Incremental Table (Merge strategy)

Chức năng: Làm sạch, xác thực và loại bỏ trùng lặp dữ liệu.

- **Các bước biến đổi:**
  1. *Type Casting:* Ép kiểu các trường số sang `FLOAT64` và chuyển đổi dấu thời gian sang định dạng `TIMESTAMP` (hỗ trợ cả múi giờ UTC và US Eastern).
  2. *Incremental Filter:* Chỉ xử lý các bản ghi mới chưa tồn tại trong bảng Silver thông qua `LEFT JOIN`.
  3. *Deduplication:* Sử dụng hàm `ROW_NUMBER()` theo cụm `[symbol, market_timestamp_raw]` và lấy bản ghi mới nhất dựa trên `fetched_at`.
  4. *Data Validation:* Kiểm tra logic (ví dụ: giá phải dương, `day_high` $\geq$ `day_low`).
- **Lợi ích:** Tối ưu hiệu suất xử lý tăng trưởng và đảm bảo tính toàn vẹn của dữ liệu (*Data Integrity*).

3.5.6 Gold Layer

Lớp này bao gồm nhiều mô hình phục vụ các mục đích phân tích và trực quan hóa khác nhau:

1. Model: `gold_kpi` (*Table*)

Lưu trữ chỉ số KPI mới nhất cho mỗi mã chứng khoán (Latest per symbol) phục vụ Dashboard thời gian thực. Sử dụng `ROW_NUMBER()` để lấy bản ghi có `fetched_at` lớn nhất.

2. Model: `gold_kpi_history` (*Incremental Table*)

Lưu trữ toàn bộ lịch sử biến động theo thời gian, hỗ trợ phân tích chuỗi thời gian (*time-series analysis*). Khóa duy nhất là `[symbol, market_time_utc]`.

3. Model: `gold_candlestick` (*Table*)

Tạo dữ liệu biểu đồ nến (OHLC) cho 12 ngày gần nhất. Sử dụng các hàm cửa sổ như `FIRST_VALUE` và `LAST_VALUE` để xác định giá mở/đóng cửa trong ngày.

4. Model: `gold_treechart` (*Table*)

Tính toán độ biến động (*volatility*) dựa trên độ lệch chuẩn (`STDDEV_POP`) và giá trung bình để phục vụ biểu đồ Tree Chart (kích thước ô dựa trên độ biến động).

5. Model: `gold_ohlc_daily` (*Table*)

Bảng tổng hợp OHLC theo ngày (trade_date). Đây là bảng nền tảng (*foundation*) để tính toán các chỉ số kỹ thuật nâng cao.

6. Model: `gold_rsi_daily` (*Table - Partitioned & Clustered*)

Tính toán chỉ số **RSI 14 ngày** (Relative Strength Index) để đo lường quán tính và xác định trạng thái quá mua/quá bán.

- *Công thức:* $100 - (100/(1 + \text{Avg Gain}/\text{Avg Loss}))$.
- *Tối ưu:* Phân vùng theo `trade_date` và cụm theo `symbol` để tăng tốc độ truy vấn.

7. Model: `gold_atr_daily` (*Table - Partitioned & Clustered*)

Tính toán **ATR 14 ngày** (Average True Range) để đo lường độ biến động thực tế, phục vụ quản trị rủi ro và xác định điểm dừng lỗ (*stop-loss*).

8. Model: `gold_sma_daily` (*Table*)

Tính toán đường trung bình động đơn giản (**SMA**) cho các chu kỳ 20, 50 và 200 ngày. Đây là các chỉ báo quan trọng để xác định xu hướng dài hạn và các tín hiệu giao cắt (*Golden/Death Cross*).

Đặc điểm chung của Gold Layer: Tất cả các bảng đều là dữ liệu đã qua tính toán trước (*pre-computed*), sẵn sàng cho báo cáo và được tối ưu hóa tối đa cho việc tiêu thụ dữ liệu phía người dùng cuối.

3.6 Tầng trực quan hóa dữ liệu

3.6.1 Tổng Quan

Tầng 6 là tầng trực quan hóa dữ liệu, sử dụng **Grafana** làm nền tảng *dashboard* chính để hiển thị *metrics*, KPIs, và *analytics* từ lớp *Gold* của BigQuery. Tầng này đóng vai trò là điểm cuối trong *data pipeline*, chuyển đổi dữ liệu đã được xử lý và tổng hợp thành các hình ảnh trực quan (*visualizations*) có ý nghĩa cho người dùng cuối.

Luồng Dữ Liệu (Data Flow):

- Các bảng trong *BigQuery Gold layer* chứa dữ liệu đã được biến đổi và tổng hợp.
- Grafana kết nối với BigQuery thông qua *BigQuery data source plugin*.
- Người dùng truy cập giao diện Grafana UI để xem các *dashboards* và hình ảnh trực quan.
- Các *dashboards* tự động làm mới (*refresh*) để hiển thị dữ liệu mới nhất.

3.6.2 Thành Phần: Grafana

Vai Trò và Chức Năng:

Grafana là một nền tảng mã nguồn mở chuyên dụng cho phân tích và giám sát, cho phép tạo các *dashboard* tương tác và trực quan hóa phức tạp. Trong hệ thống này, Grafana đóng vai trò:

1. **Data Visualization:** Hiển thị dữ liệu chứng khoán dưới dạng biểu đồ (*charts, graphs*) và bảng biểu với các *dashboard* tương tác để giám sát thời gian thực.
2. **Real-Time Monitoring:** Tự động làm mới dữ liệu, hỗ trợ lựa chọn khoảng thời gian (*time-range selection*) và cung cấp khả năng cảnh báo (*alerting*).
3. **Analytics Interface:** Cung cấp giao diện thân thiện cho các nhà phân tích, cho phép đi sâu vào chi tiết (*drill-down*) và xuất dữ liệu phục vụ báo cáo.

3.6.3 Các loại biểu đồ và Dashboard

1. Dashboard Giám Sát Thời Gian Thực

- **Mục đích:** Hiển thị dữ liệu thời gian thực để giám sát biến động thị trường tức thời.
- **Thành phần:** Các bảng điều khiển (*panels*) hiển thị giá hiện tại, chỉ báo phần trăm thay đổi, bảng KPI mới nhất và biểu đồ chuỗi thời gian cho các biến động gần đây.
- **Tần suất làm mới:** Tự động làm mới mỗi 1–5 phút, hỗ trợ tùy chọn làm mới thủ công.
- **Trường hợp sử dụng:** Giám sát trực tiếp, cảnh báo nhanh, hỗ trợ ra quyết định cho nhà giao dịch.

2. Dashboard Phân Tích Kỹ Thuật

- **Mục đích:** Cung cấp cái nhìn toàn diện về kỹ thuật thị trường với nhiều chỉ báo chuyên sâu.
- **Thành phần:** Biểu đồ nến (*Candlestick*) với dữ liệu OHLC, các chỉ báo kỹ thuật như RSI, ATR và các đường trung bình động SMA.
- **Tính năng:** Chọn khoảng thời gian tương tác, so sánh giữa nhiều mã chứng khoán, hỗ trợ các thao tác phóng to (*zoom*) và di chuyển biểu đồ (*pan*).
- **Trường hợp sử dụng:** Phân tích kỹ thuật, phát triển chiến lược giao dịch, nhận diện mô hình giá và xác định điểm vào/ra lệnh.

3. Dashboard Phân Tích Rủi Ro

- **Mục đích:** Đánh giá mức độ rủi ro và phân tích độ biến động của danh mục.
- **Thành phần:** Bản đồ cây (*Tree map*) về độ biến động, xu hướng ATR, so sánh các chỉ số rủi ro và phân tích biến động tương đối.
- **Tính năng:** Trực quan hóa điểm rủi ro, xếp hạng các mã chứng khoán theo độ biến động, tính toán lợi nhuận điều chỉnh theo rủi ro.
- **Trường hợp sử dụng:** Quản lý rủi ro danh mục, tối ưu hóa tỷ trọng đầu tư và giám sát các điều kiện thị trường bất thường.

4 Phân tích chi tiết các chỉ báo kỹ thuật

4.1 Biểu đồ chỉ số Relative Strength Index (RSI)

RSI (Relative Strength Index), hay Chỉ số Sức mạnh Tương đối, là một trong những chỉ báo kỹ thuật phổ biến nhất trong chứng khoán. Nó được dùng để đo lường mức độ biến động của giá cổ phiếu nhằm xác định xem cổ phiếu đó đang ở trạng thái "quá mua" hay "quá bán".

Công thức tính RSI

Chỉ số RSI được tính toán dựa trên mức tăng và giảm giá trung bình trong một khoảng thời gian nhất định (mặc định thường là 14 phiên). Công thức tổng quát như sau:

$$RSI = 100 - \left[\frac{100}{1 + RS}\right]$$

Trong đó, RS (*Relative Strength*) là Sức mạnh tương đối, được tính bằng:

$$RS = \frac{\text{Trung bình giá tăng (Average Gain)}}{\text{Trung bình giá giảm (Average Loss)}}$$

Các bước tính chi tiết (Ví dụ chu kỳ 14 ngày):

1. **Tính biến động giá:** Lấy giá đóng cửa hôm nay trừ giá đóng cửa hôm trước. Nếu kết quả dương thì đó là "tăng", nếu âm thì đó là "giảm" (lấy giá trị tuyệt đối).
2. **Tính Trung bình tăng (AG):** Tổng các mức tăng trong 14 ngày chia cho 14.
3. **Tính Trung bình giảm (AL):** Tổng các mức giảm trong 14 ngày chia cho 14.
4. **Tính RS:** Chia AG cho AL.
5. **Tính RSI:** Áp dụng vào công thức cuối cùng để đưa giá trị về thang điểm từ 0 đến 100.

Ý nghĩa của các mốc RSI

RSI luôn dao động trong khoảng từ 0 đến 100. Nhà đầu tư thường nhìn vào các ngưỡng sau để ra quyết định:

Chỉ số RSI	Trạng thái thị trường	Ý nghĩa / Hành động
Trên 70	Quá mua (Overbought)	Giá tăng quá cao, có nguy cơ đảo chiều. Cân nhắc chốt lời.
Dưới 30	Quá bán (Oversold)	Giá giảm quá sâu, có khả năng phục hồi. Cân nhắc mua vào.
Mức 50	Trung tính	Ranh giới giữa xu hướng tăng và giảm. RSI $>$ 50 là ưu thế mua.

4.2 Biểu đồ chỉ số Average True Range (ATR)

Average True Range (ATR), hay Khoảng dao động thực tế trung bình, là một chỉ báo kỹ thuật dùng để đo lường mức độ biến động (*volatility*) của thị trường.

Cách tính ATR

Để tính được ATR, trước hết chúng ta phải tính True Range (TR - Khoảng dao động thực tế) của từng phiên.

Bước 1: Tính True Range (TR)

TR là giá trị lớn nhất trong 3 phép tính sau:

- $H - L$: Giá cao nhất phiên hiện tại trừ giá thấp nhất phiên hiện tại.
- $|H - C_p|$: Giá trị tuyệt đối của (Giá cao nhất hiện tại trừ giá đóng cửa phiên trước).
- $|L - C_p|$: Giá trị tuyệt đối của (Giá thấp nhất hiện tại trừ giá đóng cửa phiên trước).

Giải thích: Việc tính thêm giá đóng cửa phiên trước giúp bao quát toàn bộ biến động thực sự của giá khi có hiện tượng nhảy khoảng trống giá (Gap).

Bước 2: Tính ATR (Trung bình của TR)

Chỉ số ATR thường được lấy mặc định là chu kỳ 14 phiên.

- **ATR đầu tiên:** Là trung bình cộng đơn giản của 14 giá trị TR đầu tiên.
- **Các ATR tiếp theo:** Được làm mượt theo công thức:

$$\text{ATR}_{\text{hiện tại}} = \frac{(\text{ATR}_{\text{trước đó}} \times 13) + \text{TR}_{\text{hiện tại}}}{14}$$

Ý nghĩa của ATR

1. Thước đo khách quan về sự "ồn ào" của thị trường:

- **ATR cao:** Thị trường biến động dữ dội, biên độ lớn. Rủi ro cao đi kèm cơ hội lớn.
- **ATR thấp:** Thị trường yên lặng, đi ngang (*sideway*) hoặc tích lũy. Báo hiệu một đợt bùng nổ sắp diễn ra.

2. Công cụ đặt Cắt lỗ (Stop-loss) linh hoạt:

- **Tránh bị "quét" lệnh oan:** Cổ phiếu biến động mạnh yêu cầu mức dừng lỗ rộng hơn.
- **Tối ưu hóa lợi nhuận:** Dùng làm *Trailing Stop*. Khi giá tăng, nâng mức dừng lỗ lên một khoảng bằng $2 \times$ ATR hoặc $3 \times$ ATR dưới giá hiện tại.

3. Xác định quy mô vị thế (Position Sizing):

- **Biến động mạnh (ATR tăng):** Nên giảm khối lượng cổ phiếu nắm giữ.
- **Biến động thấp (ATR giảm):** Có thể mua với khối lượng lớn hơn.

4. Cảnh báo sớm về sự đảo chiều: Nếu giá đang tăng mạnh nhưng ATR bắt đầu sụt giảm, đó có thể là dấu hiệu cho thấy lực mua đang yếu dần.

4.3 Biểu đồ Độ biến động tương đối (Relative Volatility)

Công thức tính:

```
relative_volatility = STDDEV_POP(price) / AVG(price)
```

- `STDDEV_POP(price)`: độ lệch chuẩn của giá (toàn bộ lịch sử).
- `AVG(price)`: giá trung bình.
- Kết quả là hệ số biến thiên (*coefficient of variation*).

Ý nghĩa:

- Đo mức biến động giá so với giá trung bình.
- Đã chuẩn hóa, nên có thể so sánh giữa các mã có mức giá khác nhau.
- Giá trị cao = biến động lớn (rủi ro cao); Giá trị thấp = ổn định hơn.

Mục đích: Đánh giá rủi ro, so sánh các mã, phân tích danh mục thông qua *Tree map, Bubble chart.*

4.4 Biểu đồ Giá trung bình mới nhất theo ngày

Biểu đồ này hiển thị `avg_price` của các mã cổ phiếu, sắp xếp từ cao xuống thấp.

- **Cách tính:** Là trung bình cộng (AVG) của tất cả các điểm giá trong ngày giao dịch mới nhất (theo giờ Mỹ), không chỉ tính giá đóng cửa.
- **Ý nghĩa:** Phản ánh mức giá trung bình trong phiên, giúp phân loại và so sánh giá giữa các mã.

4.5 Biểu đồ chỉ số Simple Moving Average (SMA)

SMA (Simple Moving Average), hay Đường trung bình động đơn giản, giúp làm mượt dữ liệu giá để nhìn rõ xu hướng mà không bị nhiễu bởi biến động ngắn hạn.

Công thức tính SMA

$$SMA = \frac{P_1 + P_2 + \cdots + P_n}{n}$$

Trong đó: P_n là giá đóng cửa phiên thứ n; n là chu kỳ chọn (Ví dụ SMA(10) là trung bình 10 phiên gần nhất).

Các mốc SMA phổ biến

- **Ngắn hạn:** SMA(10), SMA(20) – Phản ứng nhanh.
- **Trung hạn:** SMA(50) – Xu hướng vài tháng.
- **Dài hạn:** SMA(100), SMA(200) – Xu hướng năm.

Ý nghĩa và Cách sử dụng

- **Xác định xu hướng:** Tăng khi giá nằm trên SMA đang hướng lên; giảm khi giá nằm dưới SMA đang hướng xuống.
- **Tín hiệu giao cắt (Crossover):** *Golden Cross* (SMA ngắn cắt lên SMA dài) là điểm mua; *Death Cross* (ngược lại) là điểm bán.
- **Hỗ trợ và Kháng cự động:** Giá thường có xu hướng bật lại khi chạm vào đường SMA trong một xu hướng rõ ràng.

4.6 Biểu đồ nến (Candlestick Chart)

Cấu tạo của một cây nến

- **Thân nến (Body):** Hình chữ nhật giữa giá mở và đóng cửa.
- **Bóng nến / Râu nến (Shadow/Wick):** Giá cao nhất và thấp nhất phiên.

Hai loại nến cơ bản

- **Nến xanh (Tăng):** Giá đóng cửa $>$ giá mở cửa.
- **Nến đỏ (Giảm):** Giá đóng cửa $<$ giá mở cửa.

Ý nghĩa

Thân nến dài thể hiện áp lực mua/bán mạnh. Bóng nến trên dài thể hiện áp lực bán tại vùng giá cao; bóng nến dưới dài thể hiện lực cầu bắt đáy mạnh.

[Image of candlestick chart patterns explanation]

4.7 Biểu đồ giá giao dịch thời gian thực

- **Loại biểu đồ:** Biểu đồ đường (*Line Chart*) nối các mức giá liên tiếp.
- **Trục ngang:** Thời gian thực (phút/giờ). Các đoạn nối dài thể hiện giai đoạn nghỉ giao dịch (cuối tuần).
- **Trục đứng:** Giá trị thị trường của cổ phiếu.
- **Ý nghĩa chuyển động:** Độ dốc thể hiện tốc độ tăng/giảm. Răng cưa thể hiện sự tranh chấp phe mua/bán. Khoảng trống (Gap) thể hiện giá nhảy vọt khi mở phiên.

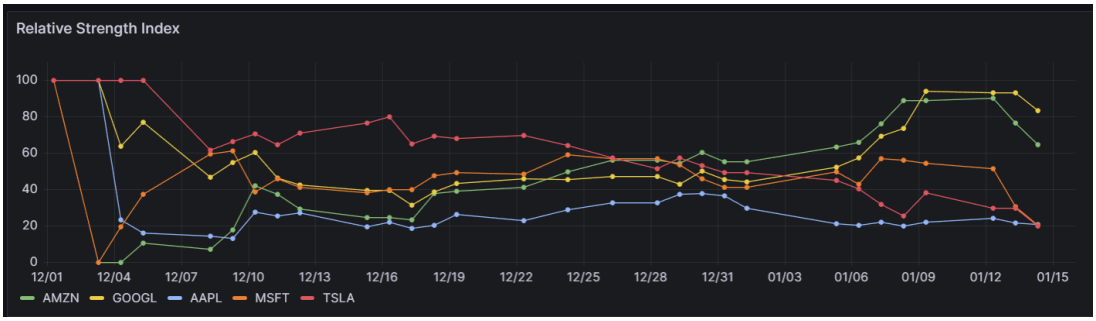
5 Kết quả và Đánh giá

5.1 Kết quả triển khai

Mục tiêu cốt lõi của hệ thống Real-Time Stock Analytics là cung cấp một cái nhìn toàn diện và tức thời về thị trường. Dashboard được thiết kế như một hệ sinh thái thông tin hoàn chỉnh, phân tách rõ ràng giữa các nhóm nhu cầu: từ giám sát giá nhảy số thời gian thực, phân tích kỹ thuật dựa trên các chỉ báo động (RSI, SMA) đến đánh giá rủi ro thông qua độ biến động ATR. Thông qua các giao diện đồ họa tương tác, người dùng có thể nhanh chóng nhận diện các xu hướng đảo chiều, các vùng quá mua/quá bán và sức mạnh của dòng tiền, từ đó tối ưu hóa quy trình ra quyết định đầu tư.

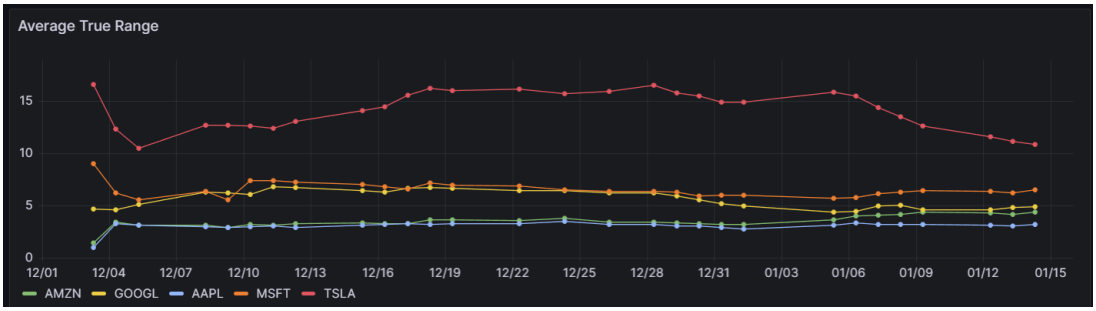
Sau đây sẽ thể hiện các dashboard của hệ thống:

- Biểu đồ chỉ số Relative Strength Index (RSI)



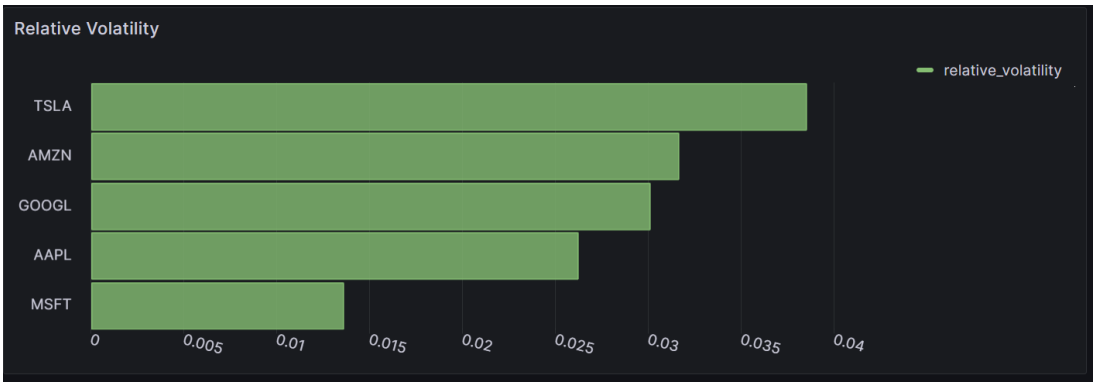
Hình 7: Chỉ số RSI

- Biểu đồ chỉ số Average True Range (ATR)



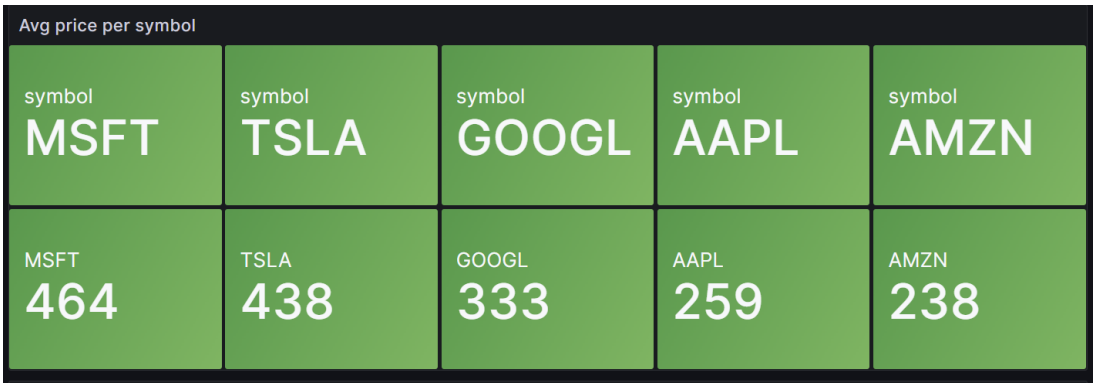
Hình 8: Chỉ số ATR

- Biểu đồ Độ biến động tương đối (Relative Volatility)



Hình 9: Chỉ số Relative Volatility

- Biểu đồ Giá trung bình mới nhất theo ngày

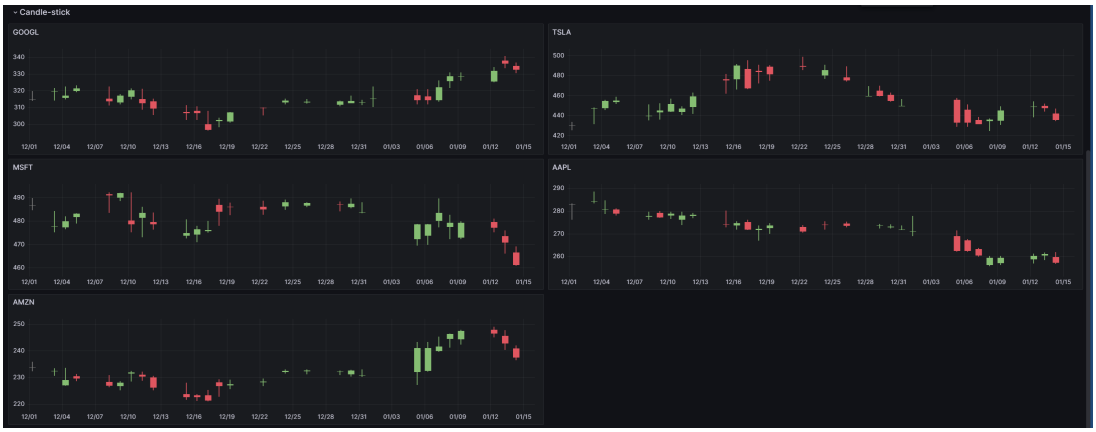


Hình 10: Giá trung bình mới nhất theo ngày

- Biểu đồ chỉ số Simple Moving Average (SMA)

Hình 11: Chỉ số SMA

- Biểu đồ nến (Candlestick Chart)



Hình 12: Candlestick Chart

- Biểu đồ giá giao dịch thời gian thực

Hình 13: Biểu đồ giá giao dịch thời gian thực

5.2 Đánh giá hệ thống

5.2.1 Tổng quan triển khai

Hệ thống đã được triển khai hoàn chỉnh bằng *Docker Compose* với các dịch vụ cốt lõi: **Apache Kafka** và **Zookeeper** (phục vụ *real-time ingestion*), **MinIO** (*object storage*), **Apache Airflow 2.9.3** (*orchestration*) cùng cơ sở dữ liệu *PostgreSQL*, **Google BigQuery** (*data warehouse*), và **Grafana** (*visualization*). Kiến trúc được xây dựng dựa trên mô hình *Medallion Architecture* (Bronze–Silver–Gold) để xử lý dữ liệu của 5 mã chứng khoán Big Tech: AAPL, MSFT, TSLA, GOOGL, AMZN.

Luồng dữ liệu và hiệu năng:

Producer thực hiện thu thập dữ liệu từ *Finnhub API* với chu kỳ 6 giây và phát hành (*publish*) vào *Kafka topic* `stock-quotes`. *Consumer* tiếp nhận và lưu trữ vào MinIO dưới dạng JSON. *Airflow DAG* `minio_to_bigquery_multi` vận hành mỗi 1 phút, thực hiện xử lý song song 5 mã chứng khoán với cơ chế nạp tăng trưởng (*incremental loading*) dựa trên *metadata tracking*.

- **Độ trễ toàn quy trình (End-to-end latency):** Từ API đến lớp Gold đạt khoảng 1–2 phút.

- **Thông lượng (Throughput):** Đạt xấp xỉ 50 thông điệp/phút (tương đương ~72,000 thông điệp/ngày).

Thành phần chuyển đổi dữ liệu:

Công cụ **dbt** thực thi các phép biến đổi trực tiếp trong BigQuery qua 3 tầng logic:

- **Bronze:** Thực hiện đổi tên cột (*column renaming*).
- **Silver:** Thực hiện xác thực (*validation*), loại bỏ trùng lặp (*deduplication*), ép kiểu (*type casting*) và chuyển đổi múi giờ.
- **Gold:** Tính toán các chỉ số KPI, nến OHLC, và các chỉ báo kỹ thuật (RSI, ATR, SMA).

Hệ thống tối ưu hóa việc sử dụng tài nguyên thông qua *gate task* (chỉ chạy dbt khi có dữ liệu mới) và sử dụng *XCom* để truyền tải trạng thái giữa các tác vụ.

Trạng thái và kết quả:

Các dịch vụ hiện đang hoạt động ổn định, các tiến trình lập lịch (*DAGs*) tự động thực thi đúng chu kỳ. Giao diện *Grafana* trực quan hóa các số liệu thực tế từ các bảng thuộc lớp Gold. Hệ thống tích hợp sẵn logic thử lại (*retry logic*), xử lý lỗi (*error handling*) và khả năng cập nhật nóng (*hot reload*) cho các tệp định nghĩa DAG.

5.2.2 Nhận xét và Đánh giá

Điểm mạnh

- **Kiến trúc hiện đại:** Kết hợp hài hòa giữa kiến trúc hướng sự kiện (*event-driven*) và xử lý theo lô (*batch*).
- **Quản lý dữ liệu chặt chẽ:** Áp dụng mô hình Medallion giúp phân tầng dữ liệu rõ ràng, đảm bảo khả năng truy vết (*traceability*) và tính toàn vẹn thông qua lớp Silver.
- **Hiệu suất và Mở rộng:** Tận dụng khả năng xử lý song song của Airflow và tính năng tự động mở rộng (*auto-scale*) của Google BigQuery.
- **Tính an toàn:** Cơ chế *idempotency* cho phép hệ thống thử lại các tác vụ lỗi mà không gây sai lệch hoặc trùng lặp dữ liệu.

Hạn chế và Hướng khắc phục Mặc dù đạt được các mục tiêu đề ra, hệ thống vẫn tồn tại một số hạn chế cần cải thiện:

- **Khả năng chịu lỗi (Fault Tolerance):** Các thành phần như Kafka, Airflow và MinIO hiện chỉ chạy đơn nút (*single node*), tạo ra các điểm yếu đơn nhất (*single points of failure*).
- **Giám sát (Monitoring):** Thiếu vắng các công cụ như Prometheus để theo dõi hạ tầng và hệ thống cảnh báo (*alerting*) chủ động khi có sự cố.
- **Kiểm thử (Testing):** Chưa triển khai khung kiểm thử dữ liệu (*dbt tests*) và đơn vị (*unit tests*) cho các thành phần Producer/Consumer.
- **Tối ưu hóa hiệu năng:** *SequentialExecutor* trong Airflow chưa tận dụng tốt đa nhân (*multi-core*); các bảng BigQuery cần được phân vùng (*partition*) và cụm hóa (*cluster*) sâu hơn.
- **Bảo mật (Security):** Các thông tin đăng nhập (*credentials*) còn ở dạng thô, thiếu hệ thống quản lý bí mật (*Secrets Management*) và cô lập mạng (*Network Isolation*).

- **Quản lý chi phí:** Cần thiết lập chính sách lưu giữ dữ liệu (*Retention Policy*) và dọn dẹp nhật ký (*logs*) để tối ưu chi phí lưu trữ dài hạn.