**Open Data Lakehouse**

**Description**

ETL pipelin flow:

* clean object storage (xóa data của lần chạy trước đó)
* upload local csv raw data to MinIO object storage (S3)
* Trino SQL tạo bảng staging table trong Hive schema từ file csv
  + lúc này toàn bộ dữ liệu được parse đều là varchar
* Trino SQL tạo bảng trong schema Iceberg (đẩy đủ data type int, varchar)
* Transform dữ liệu bảng Hive để insert vào Iceberg table
* Toàn bộ Flow được quản lý = Apache Airflow

**Tech Stack**

* orchestration: Airflow
* Transform: Trino
* Catalog: Hive metastore
* Storage: Minio
* Table format: Apache Iceberg

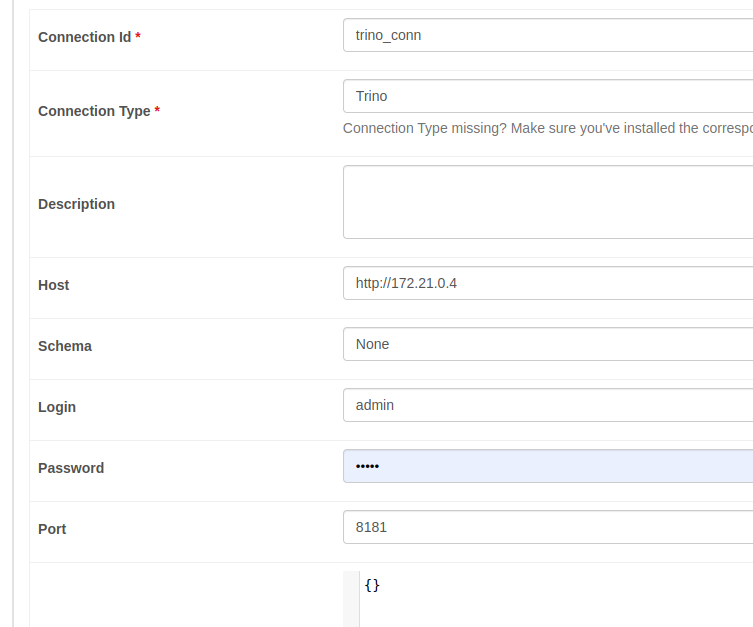
**Cách chạy project**

python run.py (build custom docker image for airflow and run docker compose up)

docker inspect network lakehouse-trino-mino-iceberg-airflow\_open-lakehouse-network

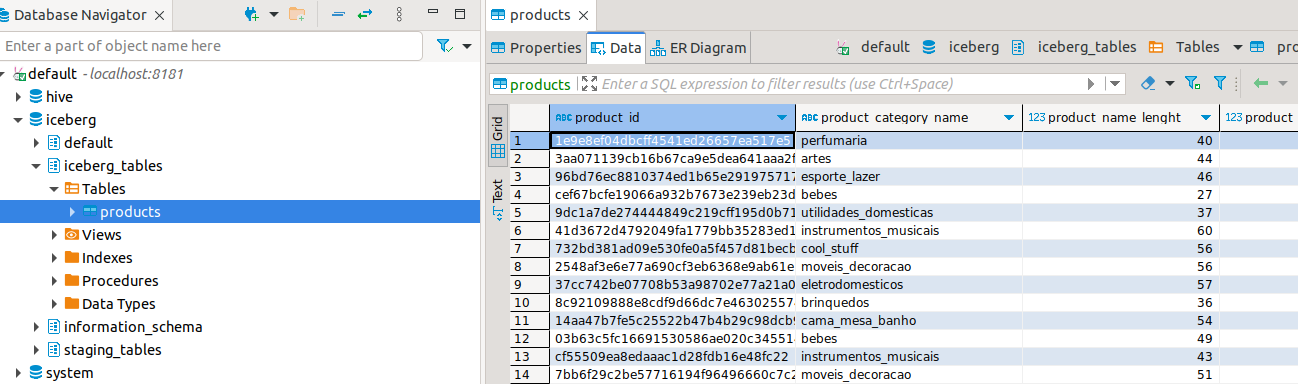
→ find IPv4Address của “trino-coordinator" (ví dụ 172.24.0.2) → host = http://172.24.0.2)

→ define Trino connection on Airflow UI như hình dưới (chỉ thay đổi host), password để trống



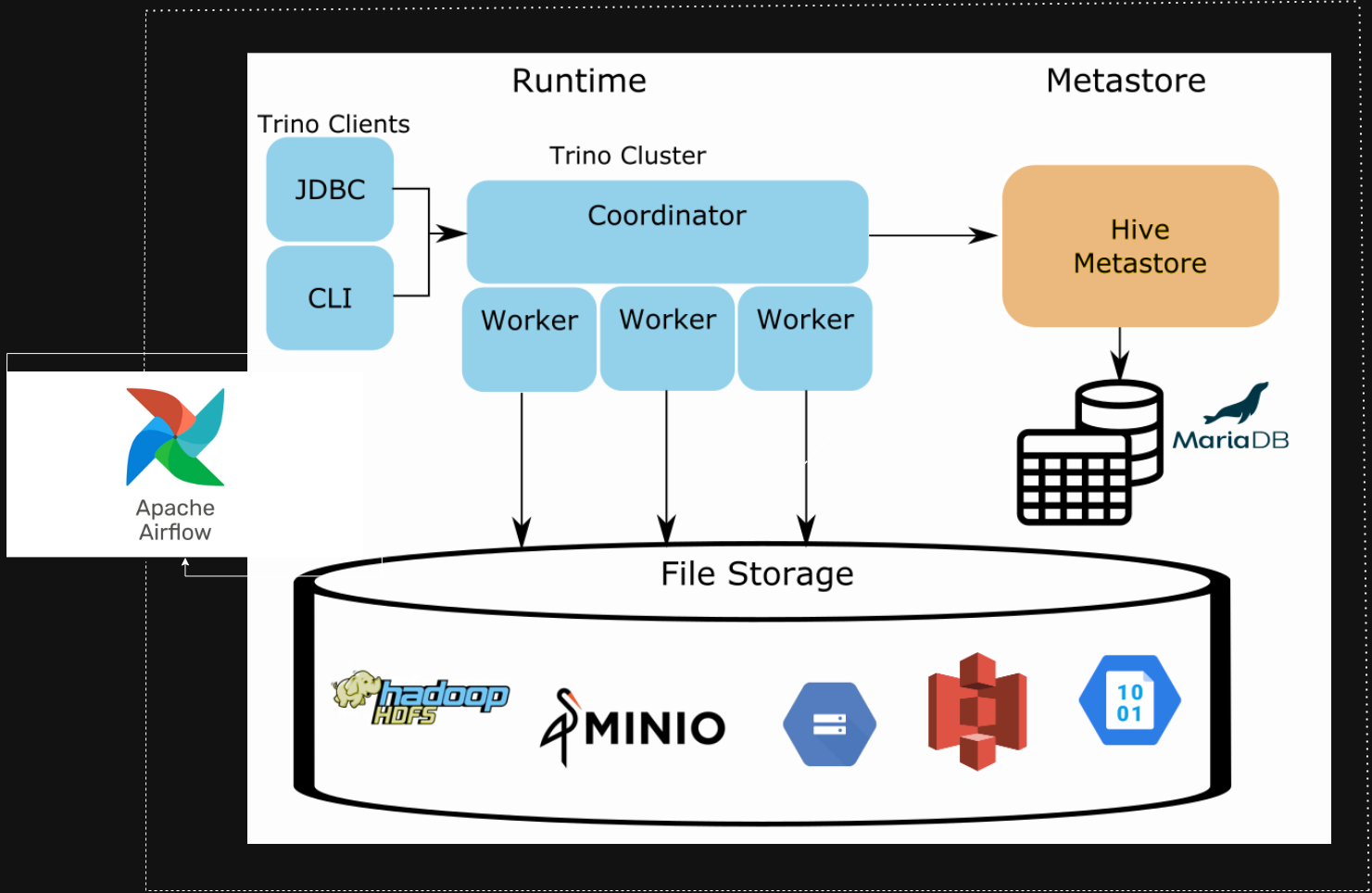
Trigger Airflow DAG trên UI

* kiểm tra dữ liệu trên minio
  + localhost:9001 → minio / minio123
* query dữ liệu trên Iceberg table bảng Products với Dbeaver
  + catalog iceberg/ iceberg\_tables



docker compose down -v

**Overview**



**Nhận xét và khám phá**

**Storage: Minio**

* minio là open source object storage engine và có API tương tự với Amazon S3
* có thể tương tác với MinIO dùng API của minio hoặc boto của aws
* Các service tương tự: amazon S3, google cloud storage, hdfs

**Open Table format Apache Iceberg**

nguồn tham khảo chính

<https://github.com/tabular-io/docker-spark-iceberg>

* là data format với cốt lõi là Parquet và có thêm nhiều file metadata giúp cho việc quản lý phiên bản, update schema, update dữ liệu trở nên dễ dàng và hiệu quả hơn
* giúp tương tác với các file này khi sử dụng các query engine phổ biến như SparkSQL hay Trino sẽ giống như database thông thường
* open source tương tự: delta lake, apache hudi, apache paimon (mạnh cho streaming - tên cũ là flink table store)

Cấu trúc thư mục của 1 bảng Iceberg table gồm

* thư mục data —> nơi chứa file parquet
* thư mục metadata —> chứa metadata về bảng parquet này như schema, change, partition, …

warehouse 
Created on: Fri, May 03 2024 (GMT+7) 
4 Objects 
warehouse I nyc I taxis 
A Name 
data 
metadata 

Với một số thao tác không thay đổi dữ liệu ví dụ như update schema (thay đổi data type từ int → big int) thì thư mục data ko thay đổi, chỉ thay đổi (thêm) nhiều file metadata để tracking

warehouse 
Created on: Fri, May 03 2024 (GMT+7) Access: PRIVATE 32.4 MB -4 Objects 
warehouse 1 nyc 1 taxis 1 metadata 1 00000-495e6042-7d25-4340-833b-4aoee45d3b08.metadata.json 
A Name 
00000-495e6042-7d25-4340-833b-4aOee45d3b08.metadata.json 
00001-35b2b4c4-a159-449f-abdd-b1ad55722dee.metadata.json 
00002-a6d27400-97aa-46bO-adOb-79b83d30de6e.metadata.json 
00003-d8375886-8ee1-4cb1-aafO-1fa8d8b66ab8.metadata.json 
00004-07d8c5bO-7f58-4194-a74d-27f36d7fe04b.metadata.json 
1462290c-10c5-4649-8fc1-c3dOdf63fdad-mO.avro 
snap-5528844671794013515-1-1462290c-10c5-4649-8fc1-c3dOdf63fdad.... 
Last Modified 
Today, 
Today, 
Today, 
Today, 
Today, 
Today, 
Today, 
11:02 
11:28 
11:28 
11:28 
11:29 
11:02 
11:02 

Với một số thao tác như xóa row, thêm cột thì sẽ sinh thêm cả data (parquet) và metadata

* nhưng hình như sẽ không cần viết lại toàn bộ data mà chỉ một số partition hoặc cột thêm mới
  + metadata có trách nhiệm connect các mẩu này lại với nhau khi Query Engine gọi đến Iceberg Table

warehouse 
Created on: Fri, May 03 2024 (GMT+7) 
warehouse I nyc I taxis I data 
A Name 
Access: PRIVATE 
68.8 MiB - 13 C 
00000-13-c32b7218-599b-4e11-9f77-984b3069a8db-O-00001.parquet 
00000-8-1d6814c9-e286-482e-b92b-86d974536ce4-O-00001.parquet 
00001-2-fe639fdd-bdef-4ade-a7f6-fbb4072aOdd6-O-00001.parquet 

**Hive metastore**

* thư mục config: metastore\_conf/
* là một service độc lập tách ra từ hệ sinh thái của Hive
* trong project này Hive metastore gồm 2 component:
  + hive metastore service
  + maria db để lưu trữ data của Hive
* Hive metastore có thể được triển khai như là 1 standalone service không cần mariadb
  + <https://github.com/recap-build/hive-metastore-standalone>
  + tuy nhiên về mặt config cho file metastore-site.xml sẽ có sự khác biệt
* Mỗi khi query engine gọi đến table thì sẽ chọc vào Hive metastore service trước để lấy thông tin catalog, schema, table từ Hive metatore
* Có nhiềm service tương tự trong hệ sinh thái Lakehouse có thể dùng thay thế cho Hive metastore
  + iceberg catalog
  + nessie catalog
  + aws glue catalog

**Trino**

* thư mục config: trino/
* distributed query engine có thể query gần mọi thứ: file trong HDFS, data trong database, dữ liệu trong kafka topic, ….
* phù hợp trong kiến trúc lakehouse khi có thể dễ dàng query dữ liệu dưới dạng table format trong data lake —> với người dùng thì không khác thì một database thông thường
* Trino nổi tiếng trong việc chạy các query cho interactive analytics nhưng cũng có thể được dùng cho ETL dữ liệu (long running batch query) trong data lake/ lakehouse
* Trong project này thì Trino dùng Hive metastore và được config có 2 loại catalog và Hive và Iceberg
* khi submit query lên cluster thì Trino sẽ chọc vào metastore trước để lấy thông tin

connector.name=iceberg

hive.metastore.uri=thrift://hive-metastore:9083

hive.s3.path-style-access=true

hive.s3.endpoint=http://minio:9000

hive.s3.aws-access-key=minio

hive.s3.aws-secret-key=minio123

→ config catalog iceberg gồm nơi lấy metadata của các bảng parquet/ iceberg là hive metastore server, địa chỉ và quyển access để vào Object Store phục read và write data

connector.name=hive

hive.metastore.uri=thrift://hive-metastore:9083

hive.s3.path-style-access=true

hive.s3.endpoint=http://minio:9000

hive.s3.aws-access-key=minio

hive.s3.aws-secret-key=minio123

hive.non-managed-table-writes-enabled=true

# hive.s3select-pushdown.enabled=true # this config cause error

hive.storage-format=PARQUET

# hive.storage-format=ORC

hive.s3.ssl.enabled=false

hive.allow-drop-table=true

→ config Hive catalog tương tự nhưng có phần config storage format có thể là Parquet hoặc ORC

* ETL flow
  + để từ file raw CSV tạo iceberg table cần các bước sau:
    - ban đầu dữ liệu được mount từ local vào airflow
    - upload file raw lên MINIO
    - tạo bảng Hive Table để parse dữ liệu csv sang bảng Hive (do ko thể tạo bảng Iceberg trực tiếp từ CSV file)
      * thời điểm này toàn bộ các column đều là varchar
    - tạo bảng Iceberg với đầy đủ data type (int, varchar ,...)
    - Select đồng thời filter và transform (cast type) dữ liệu trong Hive table để insert vào Iceberg table
* việc sử Trino cho ETL đã phổ biến ở một số công ty lớn nhưng có lẽ dùng spark (cụ thể là spark SQL) cho việc tạo bảng iceberg, read dữ liệu và save dữ liệu thành iceberg table phổ biến hơn
  + <https://medium.com/@ongxuanhong/dataops-03-trino-dbt-spark-everything-everywhere-all-at-once-241932d27a6>

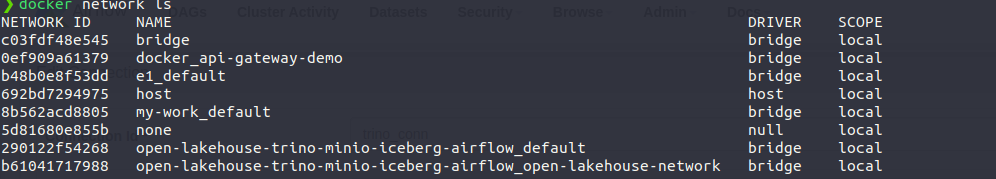
**Apache Airflow**

Do có dùng các thêm các thư viện nên cần build custom image cho apache airflow

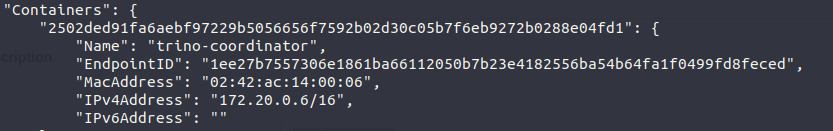
* Dockerfile trong thư mục Dockerfiles/Airflow

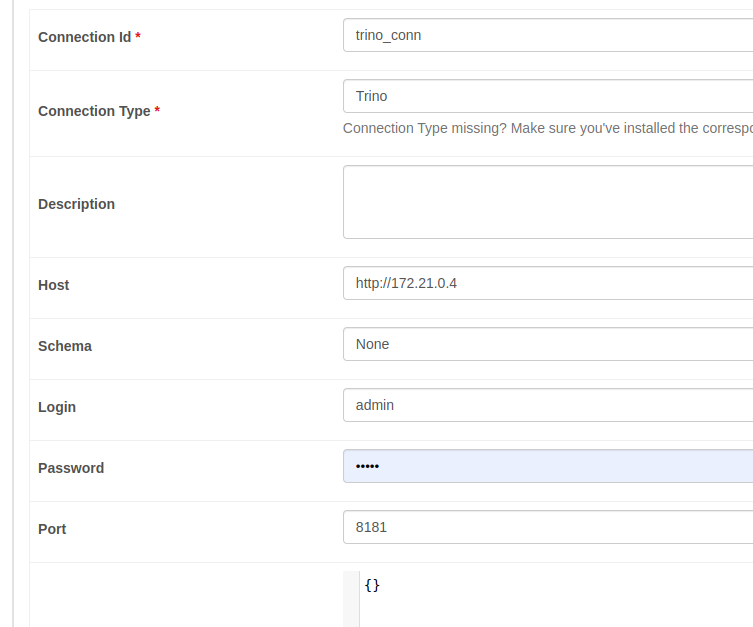
Tạo connect tới Trino

* schema default của Trino là None (ko phải default - default là catalog)
  + khám phá bằng cách dùng python trino client (<https://github.com/trinodb/trino-python-client>) connect và test SQL command —> sau đó print các attribute của connection
* host của Trino được define trong hình là <http://172.21.0.4> quá trình tìm ra bao gồm:
  + dựa vào network define trong docker compose file —> chạy docker network ls để tìm network
    - ở đây là open-lakehouse-trino-minio-iceberg-airflow\_open-lakehouse-network
  + các service được define trong docker compose file này liên lạc với thông qua địa chỉ trong network này
    - docker inspect open-lakehouse-trino-minio-iceberg-airflow\_open-lakehouse-network
      * —> tìm ra địa chỉ của trino-coordinator service → tạo connection tới Trino



kết quả docker inspect network command:





DAG

