Compte Rendu TP ATL-Datamart

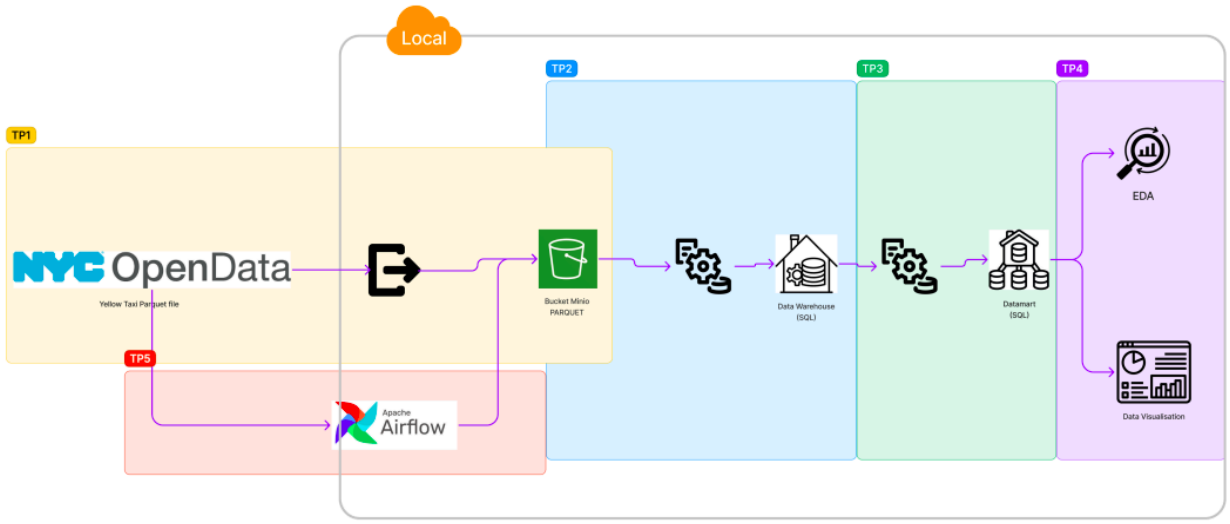
Déploiement d'une architecture décisionnel

Hamza OUJJA

Sommaire

I - Introduction

Le but de ce TP est de pratiquer le déploiement d'une architecture décisionnelle. Il s'agit d'aider une entreprise de VTC basée à New York à utiliser une partie de ses données pour générer une connaissance suivant l'architecture suivante



Le TP est composé de 5 parties, le TP1 a pour but d'automatiser la récupération des données du site de l'Etat de New York vers un Data Lake. Le TP2 consiste à appliquer un processus « ETL » pour avoir des données propres et les stocker dans un Data Warehouse. Ensuite le TP3 consiste à appliquer un deuxième processus ETL pour stocker les données dans un Datamart, pour les utiliser en visualisation de dashboard. Le TP4 consiste à utiliser un outil de Dataviz pour développer un dashboard pour visualiser les données. Et finalement le TP5 consiste à automatiser ce processus de récupération de données depuis la source vers le dashboard.

II - TP1

Cette partie de l'architecture décisionnelle consiste à alimenter un Data Lake avec des données de l'entreprise sur le site

https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page

Pour cela on utilise MinIO en tant que Data Lake. On utilisera docker et le fichier de configuration `docker-compose.yml` pour pouvoir gérer l'intégralité de l'infrastructure de l'architecture décisionnelle. On utilise la commande `docker compose up -d` pour lancer l'infrastructure.

```
(venv) hamzaoujja@MacBook-Air-de-Hamza ATL-Datamart % docker compose up -d
WARN[0000] The "AIRFLOW_UID" variable is not set. Defaulting to a blank string.
WARN[0000] The "AIRFLOW_UID" variable is not set. Defaulting to a blank string.
[+] Running 11/11
 ✓ Network spark_network                          Created
 ✓ Container atl-datamart-data-mart-1             Started
 ✓ Container atl-datamart-data-warehouse-1        Started
 ✓ Container atl-datamart-postgres-airflow-1      Healthy
 ✓ Container atl-datamart-redis-1                 Healthy
 ✓ Container minio                                Started
 ✓ Container atl-datamart-airflow-init-1          Exited
 ✓ Container atl-datamart-airflow-triggerer-1     Started
 ✓ Container atl-datamart-airflow-worker-1        Started
 ✓ Container atl-datamart-airflow-scheduler-1     Started
 ✓ Container atl-datamart-airflow-webserver-1     Started
(venv) hamzaoujja@MacBook-Air-de-Hamza ATL-Datamart %
```

Ensuite on crée un environnement virtuel pour gérer les dépendances de notre projet à l'aide des commandes `python -m venv venv` puis `source venv/bin/activate` cela permet d'utiliser la commande pip pour installer des dépendances dans le dossier "`venv`".

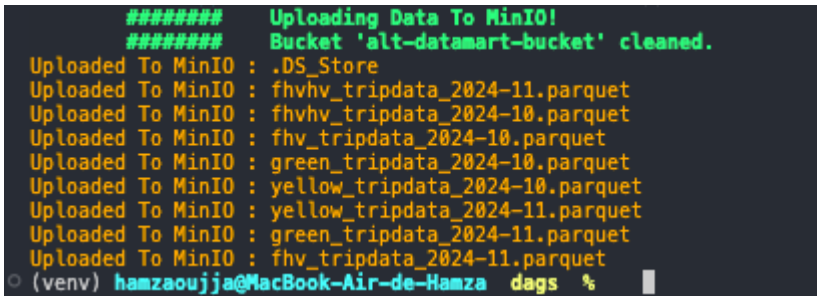
Nous allons ensuite diviser ce 1er TP en deux parties, la première consiste à télécharger la donnée depuis de site de l'entreprise, et la deuxième consiste à charger la donnée sur MinIO. Nous utilisons pour cela le dossier local "`data/row`" pour stocker la donnée téléchargée avant de la charger sur le Data Lake.

Nous créons alors dans le répertoir "src/data" un premier fichier python `grab_Data_From_Sourc(years, months, trip_types)` cette fonction (code source dans l'annexe) permet de télécharger avec un suivie la donnée de chaque type et de chaque mois, et année. Sur le terminal pendant son exécution, on peut visualiser l'avancement du téléchargement.

```
(venv) hamzaoujja@MacBook-Air-de-Hamza dags % python minio_workflow.py
       ########    Downloading Data From The Source!
Downloading From Internet : yellow_tripdata_2024-11.parquet : 100.00%
Downloading From Internet : green_tripdata_2024-11.parquet : 100.00%
Downloading From Internet : fhv_tripdata_2024-11.parquet : 100.00%
Downloading From Internet : fhvhv_tripdata_2024-11.parquet : 100.00%
(venv) hamzaoujja@MacBook-Air-de-Hamza dags %
```

Une deuxième fonction `grab_Last_Month()` permet de télécharger la donnée du dernier mois disponible sur le site source. Chaque fichier téléchargé est stocké dans le dossier local `data/row`

La deuxième partie consiste à charger les données dans le dossier local vers le Data Lake (MinIO). Pour cela on crée dans le même répertoire `src/data` la fonction `write_Data_To_MinIO()` qui permet de suivre d'alimentation du Data Lake.

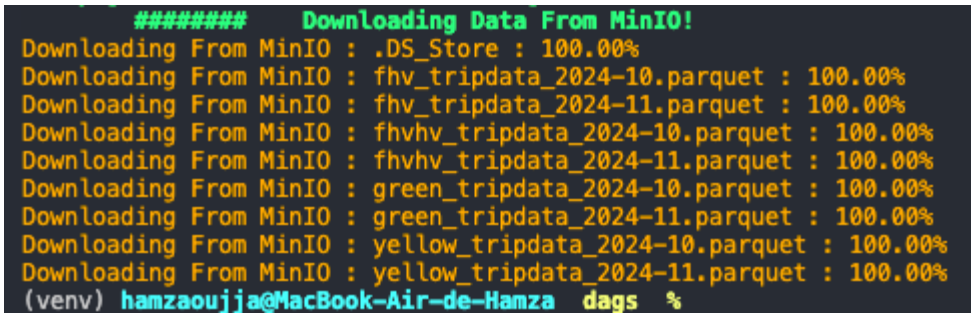


Grâce à ces trois fonctions nous pouvons automatiser la récupération de la donnée depuis la source vers le Data Lake.

III - TP2

Cette partie consiste à construire un outil ETL pour récupérer les données stockées dans le Data Lake vers une base de données considérée comme un Data Warehouse. Le fichier de configuration de docker nous permet d'avoir une SGBD PostgresQL.

Pour réaliser cette tâche, on fait la même chose que pour le premier TP, une fonction pour récupérer la data depuis MinIO, et une autre fonction pour insérer la donnée dans le Data Warehouse. Dans le répertoire `src/data` on crée la fonction `grab_Data_From_MinIO()`, cette fonction récupère la data depuis le Data Lake puis stocké sur le dossier local `data/row`. Elle permet aussi de suivre l'avancement du téléchargement.



L'étape suivante de ce TP consiste à insérer la donnée dans le Data Warehouse. Pour cela il fallait comprendre que sur le site source de l'entreprise la donnée est stockées sous forme de fichiers .parquet. Il s'agit d'un format de tableau qui permet de réduire le volume de stockage. Dans la data source il y a 4 types de tableaux : fhv, fhvhv, green et yellow. Il s'agit du type de taxi, et le but dans cette partie est d'unifier toute la data dans un seul tableau. On crée donc dans le répertoir source le fichier `write_Data_To_Warehouse.py` Ce script permet dans un premier temps de se connecter au serveur de la base de donnée Data Warehouse avec la fonction suivante

```
warehouse_conn = psycopg2.connect(
    dbname="nyc_warehouse",
    user="postgres",
    password="admin",
    host="localhost",
    port="15432"
)
```

La configuration de la connexion au Data Warehouse s'effectue à l'aide du script `connection_config.py` . Une fois la connexion établie, on exécute une requête sql qui permet de créer le tableau `warehouse_data` ce tableau contiendra toute la donnée insérée dans le Data Warehouse. Une fois créés les fichiers parquet récupérés depuis MinIO sont chargés dans le Data Warehouse. Le script permet de charger le fichier par chunks de 10000 lignes, puisque les fichiers parquets sont très volumineux. Un suivi d'insertion de chaque fichier est affiché sur le terminal. Une fois le tableau est chargé dans le Warehouse, une requête sql est exécuté pour l'insérer dans le tableau `warehouse_data` Une fois inséré, il est supprimé.

```
(venv) hamzaoujja@MacBook-Air-de-Hamza  dags  %    python minio_workflow.py
       ########    Uploading Data To WareHouse!
Uploading fhvhv_tripdata_2024-10.parquet: 100.000%
Inserting Table : fhvhv_tripdata_2024-10.parquet
Uploading fhv_tripdata_2024-10.parquet: 100.000%
Inserting Table : fhv_tripdata_2024-10.parquet
Uploading green_tripdata_2024-10.parquet: 100.000%
Inserting Table : green_tripdata_2024-10.parquet
Uploading yellow_tripdata_2024-10.parquet: 100.000%
Inserting Table : yellow_tripdata_2024-10.parquet
(venv) hamzaoujja@MacBook-Air-de-Hamza  dags  %
```

IV - TP3

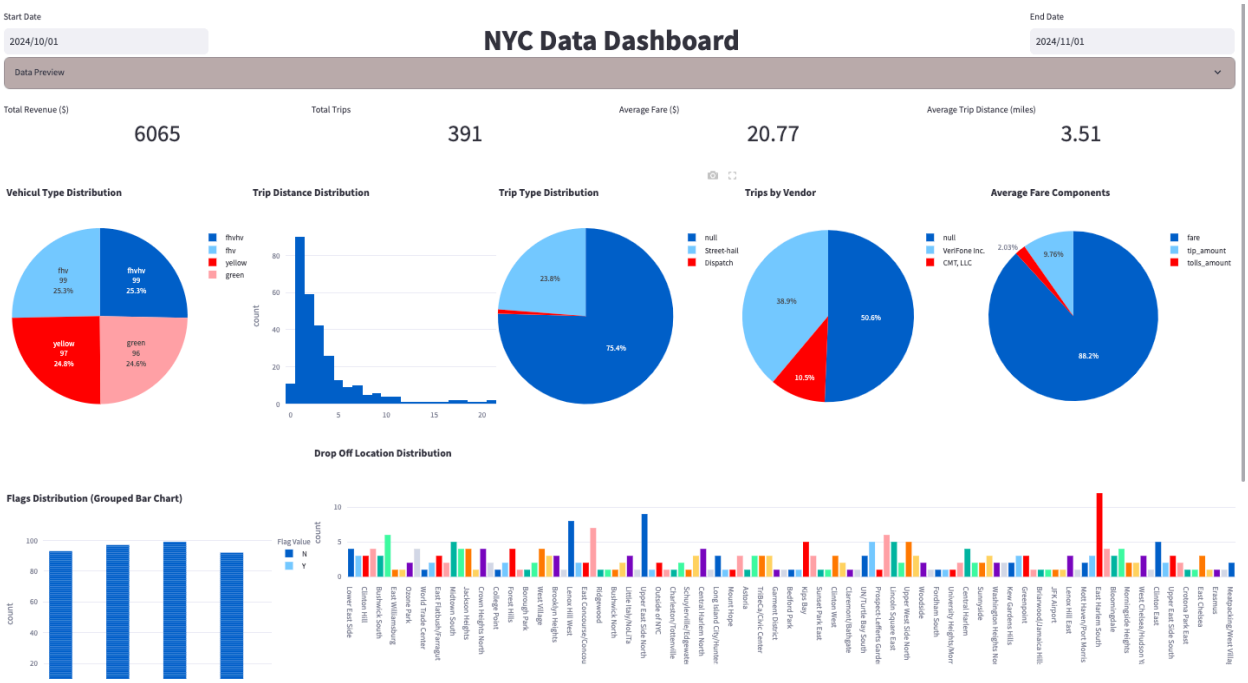
Cette partie consiste à transformer les données dans le serveur de Data Warehouse, pour produire des données utilisables pour une visualisation sur dashboard. Ces données utiles seront stockées dans des Data Marts dans un autre serveur SGBD. Pour effectuer cette tâche on créer un script python `create_Marts.py` ce script permet de se connecter au serveur Datamart avec la fonction

```
mart_conn = psycopg2.connect(
    dbname="nyc_datamart",
    user="postgres",
    password="admin",
    host="localhost",
    port="15435"
)
```

Une fois la connexion établie, le script exécute deux fichiers sql dans le Datamart `creation.sql` et `insertion.sql` . Ces deux scripts se situent dans le répertoire `/sql` Le premier permet de créer des tableaux des Marts suivant un modèle en étoile, le deuxième permet d'insérer les données depuis le Warehouse.

V - TP4

Ce TP est consacré à la conception et le développement d'un dashboard de visualisation de données du Datamart. Pour cela plusieurs outils sont possibles notamment PowerBI, Tableaux et Streamlit. On choisit ce dernier pour sa simplicité. Sur le répertoire `src/visualization` et après avoir installé Streamlit. On développe le script `visualize.py` qui permet de visualiser la donnée sur un dashboard web. Le script permet d'intégrer un fichier CSS pour le style. Le script est exécuté à l'aide de commande `streamlit run visualize.py` . Cela nous donne le dashboard suivant



Le permet de sélectionner la donnée ainsi que les différentes courbes pour une période de temps choisi par l'utilisateur. Le dashboard permet d'effectuer des analyses sur la performance de l'entreprise sur n'importe quelle période de temps.

VI - TP5

Ce TP est réservé à l'automatisation des TPs 1, 2 et 3. Pour cela on utilise Airflow qui va permettre de définir des dags qui vont s'exécuter de façon périodique pour rendre la récupération de la data automatique. Cela nécessite l'utilisation de la fonction `grab_Last_Month()` pour récupérer chaque 1er du mois la donnée du dernier mois.

VII - Annexes

Code Source : https://github.com/HamzaOUJJA/ATL-Datamart