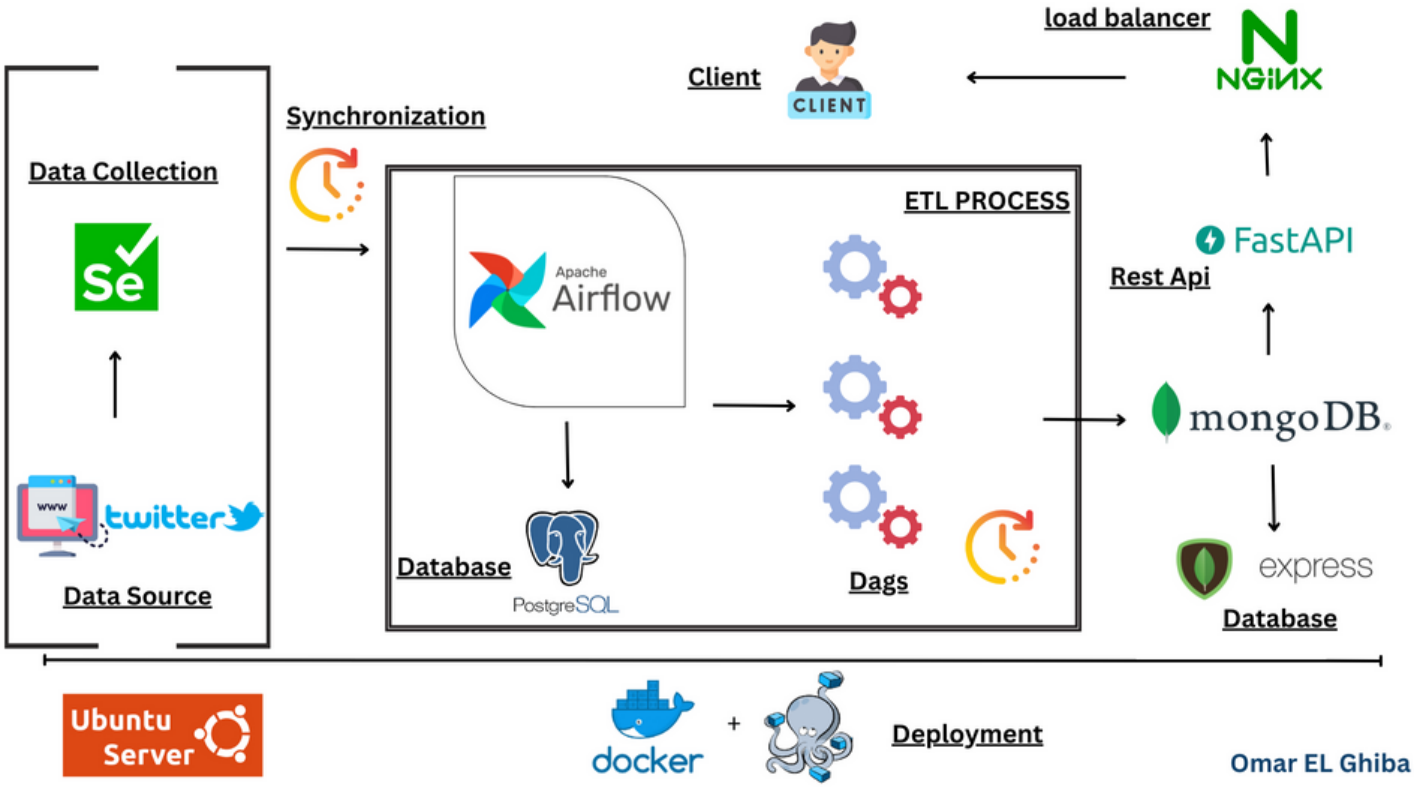
Web Scraping avec Airflow et Selenium



Dans le cadre de ce projet, j'extrais des données de Twitter et d'autres sites web en utilisant une combinaison de différentes technologies pour créer un système robuste, performant et facilement maintenable.

Tout d'abord, j'utilise Selenium, un outil puissant pour le web scraping, pour récupérer les données de Twitter et des autres sites. Le script Selenium est encapsulé dans un conteneur Docker, ce qui le rend portable et facile à déployer sur n'importe quelle machine qui prend en charge Docker.

Ensuite, ces données sont stockées dans une base de données MongoDB, qui est également contenue dans un conteneur Docker. MongoDB est une excellente option pour stocker les tweets et les autres données non structurées que je récupère. MongoDB Express, une interface web pour MongoDB, est utilisée pour visualiser les données et suivre l'évolution des métriques.

Pour automatiser le processus d'extraction des données, j'utilise Airflow, une plateforme utilisée pour programmer et surveiller les workflows. Airflow est configuré pour exécuter le script Selenium à intervalles réguliers, ce qui permet de suivre l'évolution des métriques de Twitter et d'autres sites web au fil du temps.

En outre, j'ai créé une API utilisant FastAPI, un framework web moderne et performant pour la construction d'API en Python. Cette API expose les données extraites, permettant ainsi à d'autres applications ou services d'interagir avec elles.

Toutes ces technologies sont orchestrées à l'aide de Docker Compose, ce qui permet de démarrer, d'arrêter et de gérer tous les services de manière cohérente. Docker Compose rend également le système facile à déployer sur n'importe quelle machine qui prend en charge Docker.

Enfin, tout cela est hébergé sur un serveur Ubuntu, un système d'exploitation robuste et largement utilisé qui offre de nombreuses fonctionnalités de sécurité et de gestion. Pour la mise en réseau et la gestion des requêtes entrantes, j'utilise Nginx, un serveur web haute performance.

En combinant ces technologies, je peux extraire efficacement des données de Twitter et d'autres sites web, stocker ces données de manière structurée et rendre ces données disponibles via une API. Le tout est automatisé et facilement gérable grâce à l'utilisation de conteneurs Docker et d'Airflow.

Section 1 : Extraction de données de comptes Twitter sans utiliser l'API de Twitter

la stratégie d’attaque et la détailler étape par étape :

last_week_date(): Cette fonction génère deux dates, la date actuelle et la date il y a une semaine. Ces deux dates sont ensuite renvoyées sous forme de liste.

twitter_login(user_name, pass_word): Cette fonction se connecte à un compte Twitter donné. Elle utilise le pilote initialisé par get_driver(), remplit les champs de nom d'utilisateur et de mot de passe avec les paramètres user_name et pass_word, puis clique sur le bouton de connexion.

search_accounts(account): Cette fonction est le cœur du programme. Elle récupère des informations sur un compte Twitter spécifié par le paramètre account. Voici les étapes de son fonctionnement :

Connectez-vous à Twitter à l'aide de la fonction twitter_login(user_name, pass_word).

Pour chaque tweet de la semaine dernière, elle extrait le texte, les hashtags, les métriques (vues, likes, retweets et réponses) et si le tweet contient une vidéo ou une image.

Les informations sur chaque tweet sont stockées dans un dictionnaire article_data, qui est ensuite ajouté à la liste account_data_list.

Enfin, elle stocke toutes les informations du compte dans le dictionnaire all_accounts_data sous la clé account et renvoie ce dictionnaire.

1. les outils/bibliothèques utilisées

Selenium : Selenium est une bibliothèque Python pour automatiser les navigateurs web. Elle permet de simuler l'interaction d'un utilisateur avec un site web, ce qui est utile pour le web scraping, les tests automatisés de sites web, et d'autres tâches. Vous avez importé plusieurs modules de Selenium dans votre code :

webdriver : Il fournit un moyen d'interagir avec les navigateurs web, par exemple Chrome dans votre cas. Il permet de charger une page web, de trouver des éléments HTML, d'interagir avec ces éléments (par exemple, en cliquant sur un bouton ou en remplissant un formulaire), etc.

By : Il fournit les méthodes pour localiser les éléments dans une page. Vous pouvez localiser les éléments par leur nom de balise, leur attribut d'id, leur nom de classe, etc.

WebDriverWait et expected_conditions (EC) : Ces modules sont utilisés pour implémenter des attentes explicites, qui permettent à votre script de se suspendre jusqu'à ce qu'une certaine condition soit remplie. Par exemple, vous pouvez attendre qu'un élément soit visible, que la page soit complètement chargée, etc.

Keys : Il fournit les codes clés pour les touches spéciales du clavier, comme ENTER, ESC, etc. Dans votre code, vous l'avez utilisé pour simuler la pression de la touche RETOUR afin de soumettre une recherche.

datetime : Le module datetime de Python fournit des classes pour manipuler les dates et les heures. Vous avez utilisé les classes datetime et timedelta pour calculer les dates de "maintenant" et "il y a une semaine".

time : Le module time de Python fournit diverses fonctions liées au temps. Vous ne semblez pas l'utiliser directement dans le code que vous avez montré, mais généralement il est utilisé pour les pauses (time.sleep()), pour obtenir le temps actuel en secondes depuis l'époque (time.time()), etc.

re : Le module re de Python fournit des fonctions pour travailler avec des expressions régulières. Les expressions régulières sont des séquences de caractères qui forment un motif de recherche. Vous avez utilisé la fonction findall() de ce module pour extraire les hashtags d'un texte de tweet.

Docker et Docker Compose : Docker est une plate-forme qui permet d'encapsuler des applications et leurs dépendances dans des conteneurs, qui peuvent être exécutés de manière uniforme sur n'importe quel système qui prend en charge Docker. Docker Compose est un outil qui permet de définir et de gérer des applications multi-conteneurs avec Docker.

FastAPI : Fastapi est un framework web moderne, rapide (hautes performances), basé sur des normes pour la construction d'API avec Python 3.6+ basé sur les astuces de type standard pour Python. Il est idéal pour créer des services web et des API REST.

MongoDB et MongoDB Express : MongoDB est une base de données orientée documents qui est excellente pour stocker des données non structurées comme les tweets. MongoDB Express est une interface web pour MongoDB, qui vous permet de visualiser et d'interagir facilement avec vos données.

Airflow : Airflow est une plate-forme utilisée pour programmer et surveiller des workflows. Vous pouvez l'utiliser pour automatiser l'exécution de votre script Python. Par exemple, vous pouvez configurer Airflow pour exécuter le script une fois par jour, afin de suivre l'évolution des métriques sur 7 jours. Airflow peut également être utilisé pour surveiller l'exécution du script et vous alerter en cas de problème.

Selenium Grid : Selenium Grid est un outil qui est utilisé avec Selenium pour exécuter des tests en parallèle sur différents navigateurs et systèmes d'exploitation. Il est particulièrement utile lorsqu'une grande quantité de données doit être traitée, car il permet d'exploiter plusieurs machines pour exécuter des tâches simultanément, ce qui peut accélérer le processus d'extraction des données.

Ubuntu Server : Ubuntu Server est un système d'exploitation qui peut être utilisé pour héberger tous les services mentionnés ci-dessus.

Pour intégrer toutes ces technologies, vous pouvez procéder comme suit :

Étape 1 : Installer Docker et Docker Compose sur Ubuntu Server.
Étape 2 : Créer un fichier Dockerfile pour votre script Python, qui définit comment construire l'image Docker pour votre script.
Étape 3 : Créer un fichier docker-compose.yml qui définit vos services (Python script, MongoDB, MongoDB Express, Airflow, Selenium Gird).

Étape 4 : Exécuter docker-compose up pour démarrer tous vos services.
Étape 5 : Configurer un flux de travail Airflow pour exécuter votre script Python à intervalles réguliers (par exemple, une fois par jour).
Étape 6 : Utiliser MongoDB Express pour visualiser vos données et suivre l'évolution des métriques.

Section 2 : Web Scraping et traitement de données :

Dans cette tâche, nous avons pour objectif d'extraire des données de Site web, de les stocker dans une base de données NoSQL MongoDB et de les servir via une API.

Voici un résumé des étapes de mise en œuvre :

Étape 1 : Extraction des données :

Nous avons utilisé Selenium et Python pour extraire les données . Le script Python recherche des comptes spécifiques, extrait les informations . Nous utilisons également Selenium Grid pour extraire des données de plusieurs site web en parallèle.

Étape 2 : Stockage des données dans MongoDB

Les données extraites sont stockées dans une base de données MongoDB. Nous avons créé un schéma flexible pour stocker les données.

Étape 3 : Serveur de données via une API

Nous avons créé une API pour servir les données stockées dans MongoDB. Cette API permet à d'autres services ou applications d'accéder aux données.

Étape 4 : Automatisation et déploiement

Nous avons utilisé Docker et Docker Compose pour encapsuler les services, y compris le script Python, MongoDB, MongoDB Express pour la visualisation des données et Airflow pour l'automatisation. Chaque service est exécuté dans un conteneur Docker, ce qui facilite le déploiement et la mise à l'échelle. Nous avons utilisé Airflow pour exécuter le script Python à intervalles réguliers pour suivre l'évolution des métriques au fil du temps.

Tout cela est hébergé sur un serveur Ubuntu, un système d'exploitation largement utilisé et robuste. Grâce à cette approche, nous avons créé un système complet pour l'extraction, le stockage, l'analyse et la visualisation des données .

Dans le cadre de ce projet, j'utilise une base de données NoSQL, MongoDB, pour stocker les données extraites des sites web, en me concentrant notamment sur l'exemple du titre.

Voici comment je proposerais le schéma pour la base de données NoSQL :

Collection : my_collection_1

_id : Identifiant unique généré automatiquement par MongoDB
title : Le titre extrait du site web
Pour importer les données dans cette base de données, je peux utiliser la bibliothèque MongoDB pour Python. Après avoir extrait le titre d'un site web, je peux créer un nouveau document JSON avec les champs correspondants (id, title) et l'insérer dans la collection WebsiteData.

Maintenant, pour servir les données via l'API que j'ai créée avec FastAPI, je peux ajouter un point de terminaison qui interagit avec la base de données MongoDB et renvoie les données demandées. Par exemple, je peux avoir un point de terminaison /items qui récupère tous les documents de la collection WebsiteData et les renvoie en tant que réponse.

Ce point de terminaison récupérera tous les documents de la collection WebsiteData et les renverra sous forme de liste JSON.

Ainsi, en utilisant ce schéma de base de données NoSQL avec MongoDB, j'importe les données extraites des sites web dans la collection correspondante et les sers via mon API FastAPI. Cela me permet de stocker et de fournir les données extraites de manière efficace et flexible.