CRYPTOBIZ

RYVOUN Filipp

Introduction

Le projet "CryptoBiz" est un système de collecte, de traitement et de visualisation d'informations sur les cryptomonnaies provenant de différentes sources. L'objectif du projet est de fournir à l'utilisateur un outil pratique pour surveiller et analyser la dynamique des marchés de cryptomonnaies en utilisant des technologies Big Data modernes.

Le projet est conçu pour les objectifs principaux suivants :

- **Collecte de données :**
  - Utilisation de Selenium et BeautifulSoup pour le web scraping des données sur les cryptomonnaies depuis le site crypto.com.
  - Intégration avec API.coingecko.com et un script Python personnalisé pour demander des données supplémentaires.
- **Traitement et stockage des données :**
  - Utilisation de la technologie Kafka comme intermédiaire pour le transfert de données entre les composants du système. Kafka offre une plateforme robuste et évolutive pour le traitement des flux de données.
  - Envoi des données collectées à Kafka, où elles sont traitées et préparées pour le stockage dans InfluxDB.
- **Stockage des données :**
  - Utilisation d'InfluxDB pour créer une base de données stockant des données structurées en fonction de métriques telles que "crypto_scraping" et "crypto_api".
- **Visualisation des données :**
  - Intégration de Grafana avec InfluxDB pour créer un tableau de bord.
  - Organisation de visualisations de données sur les cryptomonnaies provenant du web scraping et de l'API.

Technologies utilisées

Le projet fait usage des technologies clés suivantes :

- **Selenium et BeautifulSoup :**
  - Pour automatiser le web scraping des données sur les cryptomonnaies depuis le site crypto.com. Selenium est utilisé pour l'interaction avec le navigateur, tandis que BeautifulSoup facilite le parsing et l'extraction de données structurées à partir du HTML.
- **Kafka** :
  - En tant qu'intermédiaire pour le transfert de données entre les composants du système. Kafka fournit une plateforme stable et évolutive pour le traitement des flux de données.
- **Kafka-ui** :
  - Ce service déploie une interface utilisateur Kafka (Kafka-UI) à des fins de gestion et de surveillance du cluster Kafka. Le choix de Kafka-UI est motivé par le besoin de gestion et de surveillance facile du cluster Kafka. Cette interface utilisateur fournit des fonctionnalités essentielles pour superviser les topics, les partitions et d'autres métriques du cluster Kafka de manière visuelle et intuitive.
- **InfluxDB** :
  - Pour stocker et structurer les séries chronologiques de données. Les données peuvent être organisées efficacement en métriques pour un accès et une analyse faciles. InfluxDB offre une gestion simple des métriques, facilitant ainsi l'accès et l'analyse des données temporelles, ce qui est essentiel pour le suivi des fluctuations dans le temps.

- **Grafana :**
  - Pour créer un tableau de bord et visualiser les données. Grafana offre un environnement flexible et interactif pour la surveillance des données en temps réel. Grafana permet une représentation visuelle des données provenant de différentes sources, offrant une expérience utilisateur riche en matière de surveillance et d'analyse.
- **Python** :
  - Utilisé pour écrire les scripts de web scraping, pour interagir avec l'API, ainsi que pour l'intégration avec Kafka et InfluxDB.

Le projet vise à créer un outil pratique pour surveiller et analyser les données sur les cryptomonnaies, permettant ainsi aux utilisateurs de prendre des décisions plus éclairées dans le domaine des investissements en cryptomonnaie.

Apache Kafka

Dans le cadre de ce projet, nous utilisons l'image Docker **bitnami/kafka:latest** pour déployer trois brokers Apache Kafka, formant ainsi un cluster Kafka distribué. Voici une brève description des services spécifiques dans le fichier **docker-compose.yml** :

Services kafka0, kafka1, kafka2 :

- Ces services représentent trois instances de brokers dans le cluster Kafka. Chacun est configuré pour jouer le rôle de controller et de broker.
- Des volumes sont montés pour partager des clés entre les conteneurs, et des ports sont ouverts pour assurer la communication entre les différentes composantes du cluster Kafka.
- Les services Kafka sont configurés avec des identifiants uniques (KAFKA_CFG_NODE_ID=0, KAFKA_CFG_NODE_ID=1, KAFKA_CFG_NODE_ID=2) et sont connectés à un réseau (nw-crypto).
- Kafka Kraft (KAFKA_KRAFT_CLUSTER_ID) est utilisé pour identifier le cluster.

Service kafka-ui :

- Ce service déploie une interface utilisateur Kafka (Kafka-UI) pour la gestion et la surveillance du cluster Kafka.
- Il est configuré pour se connecter au cluster Kafka via le serveur d'amorçage avec kafka0:9092.
- Le port 8081 est ouvert pour permettre l'accès à l'interface utilisateur Kafka-UI depuis l'extérieur.

Pourquoi utiliser trois brokers ?

- L'utilisation de trois brokers Kafka garantit une haute disponibilité, une résilience aux pannes et une distribution efficace de la charge.

- Trois brokers permettent la réplication des données, assurant la préservation du service même en cas de défaillance d'un curateur. De plus, cela garantit une répartition uniforme de la charge et une mise à l'échelle horizontale efficace du système Kafka.

En conclusion, ce Docker Compose configure un environnement Kafka avec trois brokers, fournissant une plateforme distribuée et fiable pour la gestion des flux de données en temps réel.

InfluxDB & Grafana

Service InfluxDB :

- **Image :** Nous utilisons l'image **influxdb:latest**.
- **Volumes :** Ils sont montés pour sauvegarder les données d'InfluxDB.
- **Dépendances :** Ce service dépend des conteneurs Kafka (kafka0, kafka1, kafka2).
- **Variables d'environnement :** Configurées pour initialiser InfluxDB avec des paramètres prédéfinis tels que le mode d'initialisation, l'hôte, l'organisation, le nom d'utilisateur, le mot de passe, le token administrateur.
- **Port :** Ouvert pour permettre l'accès à InfluxDB sur le port 8086.
- **Réseau :** Connecté au réseau **nw-crypto**.

Service Grafana :

- **Image :** Nous utilisons l'image **grafana/grafana-enterprise:latest**.
- **Volumes :** Ils sont montés pour sauvegarder les tableaux de bord Grafana et les configurations.
- **Réseau :** Lié au réseau nw-crypto.
- **Port :** Ouvert pour permettre l'accès à Grafana sur le port 3001.
- **Variables d'environnement :** Nous définissons les noms d'utilisateur et de mot de passe de l'administrateur Grafana.

Dans le contexte du projet, InfluxDB est utilisé pour stocker et structurer les données temporelles, tandis que Grafana est utilisé pour créer des tableaux de bord et visualiser ces données. Ces deux services sont intégrés à l'environnement du projet, assurant la gestion des données et la surveillance en temps réel.

Scraping

- **Configuration des paramètres Kafka :**
  - Une liste de courtiers Kafka et un sujet vers lequel les données seront envoyées sont spécifiés.
- **Configuration de Selenium :**
  - Selenium est utilisé avec le navigateur Chrome en mode headless.
  - Des options sont définies pour assurer un fonctionnement correct du navigateur.
- **Boucle principale :**
  - Boucle infinie où le scraping des données est effectué sur plusieurs pages du site crypto.com.
- **Analyse des données :**
  - À l'aide de BeautifulSoup, les données sur les cryptomonnaies sont extraites de la page Web.
  - Les données de chaque cryptomonnaie (id, name, price, change, volume, cap) sont analysées et traitées par la fonction parse_data.
- **Envoi vers Kafka :**
  - Les données traitées sont envoyées à Kafka au format JSON.
- **Gestion des erreurs :**
  - Une gestion des erreurs est prévue en cas d'échec de l'envoi des données à Kafka.
- **Pause entre les requêtes :**
  - Une pause de 60 secondes est observée après chaque requête.