École Nationale d'Ingénieurs de Tunis
Département d'Informatique

Rapport de Projet

Titre du Projet :

Big-Data-Crypto-Real-Time-Dashboard

Préparé par :
Sami Masmoudi
Amine Ziadi
Classe : 3AINFO2

Table des matières

1 Contexte du projet **2**

2 Objectifs du projet **2**

3 Architecture et technologies **2**

3.1 Réseau : netw 2

3.2 Services déployés 3

3.2.1 Zookeeper 3

3.2.2 Kafka 3

3.2.3 Cassandra 3

3.2.4 Spark Master et Spark Worker 3

3.2.5 Hadoop (Namenode et Datanode) 3

3.2.6 Dashboard 4

4 Technologies utilisées **4**

1 Contexte du projet

L'application consiste à concevoir et mettre en œuvre un tableau de bord en temps réel pour visualiser les données liées aux cryptomonnaies. Il intègre plusieurs composants :

— Collecte de données
— Traitement en temps réel
— Affichage interactif des métriques.

Les objectifs sont d'analyser des données massives en temps réel, d'agréger des informations clés (prix moyen, volume) et de les transmettre efficacement à un tableau de bord utilisateur.

2 Objectifs du projet

Le projet visait plusieurs objectifs spécifiques :

1. **Collecte de données :** Intégrer des flux externes de données en continu depuis des API cryptographiques.
2. **Traitement des données :** Calculer en temps réel des métriques clés comme le prix moyen et le volume agrégé.
3. **Stockage des données :** Implémenter une base de données NoSQL capable de gérer les écritures rapides et massives (Cassandra).
4. **Visualisation des données :** Fournir un tableau de bord interactif et dynamique qui reçoit des mises à jour en temps réel via des WebSockets.

3 Architecture et technologies

L'architecture réseau et les services mis en œuvre dans ce projet se basent sur une infrastructure containerisée orchestrée à l'aide de Docker Compose. Chaque composant est conçu pour assurer une fonction spécifique, et tous les services communiquent via un réseau privé dédié nommé netw, configuré avec un sous-réseau spécifique pour garantir l'isolation et une communication efficace entre les conteneurs.

3.1 Réseau : netw

Un réseau personnalisé nommé netw est défini pour établir une connexion fiable entre tous les conteneurs du projet. Ce réseau utilise : Un driver bridge pour une communication locale entre les conteneurs. Un sous-réseau spécifique configuré avec 172.23.0.0/24 pour une gestion simplifiée des adresses IP.

3.2 Services déployés

3.2.1 Zookeeper

Rôle : Coordination distribuée et gestion des métadonnées pour Kafka.
Configuration :
— Expose le port 2182 pour permettre à Kafka de se connecter.
— Fonctionne dans le réseau netw pour une communication avec Kafka sans besoin d'exposer d'autres ports externes.

3.2.2 Kafka

Rôle : Middleware de messagerie pour gérer les flux de données en temps réel.
Configuration : Dépend de Zookeeper pour la gestion des brokers et des partitions. Offre deux points d'accès pour les clients :
— **9092 :** Pour les communications internes avec d'autres conteneurs ou services.
— **29092** : Pour les connexions externes depuis localhost.
Configuré avec des listeners pour garantir la compatibilité interne et externe.

3.2.3 Cassandra

Rôle : Base de données distribuée pour stocker les données des capteurs et les résultats calculés.
Configuration :
— Utilise une adresse IP statique dans le sous-réseau netw (172.23.0.6) pour simplifier les connexions.
— Expose le port 9042 pour les requêtes CQL (Cassandra Query Language).
La gestion mémoire est optimisée avec des variables d'environnement définissant la taille maximale du heap.

3.2.4 Spark Master et Spark Worker

Rôle : Traitement distribué des données avec Apache Spark.
Configuration Spark Master : Fournit une interface accessible via le port 8080. Gère les tâches de traitement distribuées soumises par les travailleurs Spark.
Configuration Spark Worker : Connecté au Spark Master via le réseau netw.
— Permet le traitement distribué en utilisant le cluster Spark.
— Expose le port 8081 pour le monitoring.

3.2.5 Hadoop (Namenode et Datanode)

Rôle : Stockage distribué des données en utilisant HDFS (Hadoop Distributed File System).
Configuration Namenode :

Gère les métadonnées et coordonne les opérations de fichiers.
Ports exposés : 9870 pour l'interface web et 8020 pour les opérations HDFS.
Configuration Datanode :

Stocke les blocs de données et communique avec le Namenode.
— **Ports exposés :** 50075 pour le monitoring et 50010 pour les transferts de données.

3.2.6 Dashboard

— **Rôle :** Interface utilisateur du projet, permettant de visualiser les données traitées et stockées.
— **Configuration :** avec une application Java, le fichier .jar est monté dans le conteneur.
— Accessible via le port 3000 pour interagir avec les utilisateurs finaux.

4 Technologies utilisées

— **Apache Spark :** Traitement distribué pour les calculs en temps réel.
— **Apache Cassandra :** Base de données NoSQL pour le stockage rapide des données cryptographiques.
— **Spring Boot :** Framework Java utilisé pour la création des services backend.
— **WebSocket :** Communication bidirectionnelle en temps réel entre le serveur et le tableau de bord.
— **Docker :** Conteneurisation des services pour assurer leur portabilité.
— **Git :** Gestion du code source et du versioning.