**Cahier de recettes et de tests**  
[BTC\_ETH\_streaming\_data](https://github.com/gaetancorin/btc_eth_streaming_data)

Réalisé par Gaëtan Corin

Introduction de l’application :

L’objectif de l’application BTC\_ETH\_streaming\_data est de pouvoir récupérer les données de cryptomonnaies en temps réel, les stocker en base de données, réaliser des calculs simples et des calculs complexes nécessitant du calcul distribué, afin de créer des indicateurs techniques qui seront sauvegardés en base de données.

Un système de monitoring doit être présent afin de contrôler l’état du serveur et des différents services/conteneurs, et un système d’alerte sur l’ensemble du système doit permettre de remonter les problématiques rencontrées.

En cas de problématique sur le système d’ingestion des données vers la base de données, les données doivent être conservées pour être réinjecté plus tard lorsque le système retrouve sa normalité.

Cahier de recettes et de tests :

Le Cahier de recettes et de tests sert à garantir que l’application est solide, robuste, et répond aux attentes du client. Il permet de tester l’ensemble des points de l’application afin de s’assurer de sa fiabilité.

| **Test effectué** | **Résultat attendu des tests** | **Commentaire (test validé ou invalide)** |
| --- | --- | --- |
| Lancer le service Postgres | Les conteneurs PostgreSQL\_external et PG\_admin doivent pouvoir communiquer entre eux. Il doit pouvoir être possible de se connecter a PG\_admin, et que PG\_admin se connecte automatiquement à PostgreSQL. Les tables de données vide doivent être générées automatiquement, et visible sur le dashboard | test validé |
| Lancer le service Ingestion | Le service doit afficher les logs sans message d’erreur signifiant la bon lancement du container. | test validé |
| Vérifier les tests unitaires du service Ingestion | Les tests doivent tous être succès | test validé |
| Lancer le service Mailhog | Le service doit afficher les logs sans message d’erreur signifiant la bon lancement du container. | test validé |
| Lancer le service Spark | Le service ne doit pas avoir de log d’erreur, le Spark master doit pouvoir se connecter au 2 Spark worker | test validé |
| Lancer le Service Airflow | Le service doit pouvoir se lancer correctement, signifiant que les 9 conteneurs du service communiquent entre eux. Il doit pouvoir être possible de se connecter sur le dashboard, et que le dashboard signifie que l’ensemble des conteneurs communique entre eux. | test validé |
| Lancer le Service Monitoring | Le service doit pouvoir se lancer correctement.  Le conteneur docker\_exporter doit transmettre l’ensemble des information sur le port 8000/metrics Le conteneur prometheus doit pouvoir avoir un dashboard accessible sur le port 9090/targets, et connecter à docker\_exporter. Le conteneur Grafana doit pouvoir avoir un dashboard accessible sur le port 3000/login, et connecter à Prometheus. | test validé |
| Vérifier l’ingestion de données de Ingestion à PostgreSQL | Les logs du conteneur Ingestion doivent contenir *“Data insert successfully on crypto\_usd”*, signifiant la bonne ingestion d’une donnée vers PostgreSQL. La donnée doit être visible dans le dashboard pg\_admin. | test validé |
| Vérifier la sauvegarde des données en CSV lorsque PostgreSQL est indisponible | Les logs du conteneur Ingestion doivent contenir *“DATA SAVE into waiting\_data\_store/crypto\_usd.csv”*. Le fichier CSV contenant les données doit être présent dans le conteneur, avec le format adapté | test validé |
| Vérifier l’ingestion des données sauvegardé en CSV lorsque PostgreSQL est de nouveau opérationnel | Les logs du conteneur Ingestion doivent contenir *“crypto\_USD WAITING DATA LOADING SUCESSFULLY INTO POSTGRES”*. Le fichier CSV contenant les données doit être supprimé de manière automatique du conteneur après ingestion. | test validé |
| Vérifier le fonctionnement des dags de Airflow | Les dags “btc\_avg\_3m\_5m\_indicator”, “eth\_avg\_3m\_5m\_indicator”, “spark\_btc\_eth\_gap\_avg\_5m\_indicator” doivent être présent sur le dashboard et fonctionner au lancement. Il doivent tous les trois être en succès | test validé |
| Vérifier le fonctionnement de Spark durant un calcul distribué | Les logs du Spark Master doivent envoyer des requêtes au Spark Worker 1 et 2. Les Spark Workers 1 et 2 doivent effectuer les calculs en distribués, puis demander de terminer la tâche au Spark Master. Sur le dashboard, la tâche doit être en état “Finished” | test validé |
| Vérifier la pertinence des données généré sur le service Postgres | Vérifier que les données générées par les dags Airflow et insérés dans le service Postgres soit cohérentes avec les données d’origines et les transformations demandés. | test validé |
| Vérifier que le monitoring de Monitoring est effectif | Sur le dashboard Grafana, l’état de chaque conteneur doit être disponible sur le moment passé et présent. Le dashboard doit présenter le CPU et mémoire de chaque conteneur et de l’ensemble des conteneurs sur le moment passé et présent. Le dashboard doit présenter le CPU et mémoire disponible sur la machine sur le moment passé et présent. | test validé |
| Vérifier le fonctionnement de Mailhog | Utiliser le dag “send\_email” de Airflow pour envoyer un email vers Mailhog. Mailhog doit bien recevoir l’email | test validé. |
| Vérifier le système d’alerte de Ingestion | Activer le service Ingestion mais éteindre le service Postgres. Au bout de 2 minutes, un email d’alerte sera envoyé à Mailhog. | test validé. |
| Vérifier le système d’alerte de Airflow | Activer le service Airflow sans activé le service Postgres ou Spark. Un email d’alerte sera envoyé à Mailhog. | test validé |
| Vérifier le système d’alerte de Monitoring | Une fois l’ensemble des services allumés avec l’ensemble de leurs conteneurs, éteindre un conteneur. Au bout de 2 minutes, un email d’alerte sera envoyé à Mailhog. | test validé |
| Vérifier le fonctionnement de la CI | Réaliser un commit et push sur github pour lancer la CI.  Celle-ci doit fonctionner sans message d’erreur. | test validé |
| Vérifier le fonctionnement de la CD | En ayant le serveur ou windows hébergeant la CD d’allumé, après avoir fait fonctionner la CI, vérifier le fonctionnement de la CD. Celle-ci doit pouvoir générer l’ensemble des services sans message d’erreur | test validé |
| Vérifier le système d’alerte de la CD par email Gmail au lieu de Mailhog | Une fois la CD généré, essayer d’envoyer un email d’alerte par le service Ingestion, par Airflow et par Grafana. Les 3 services doivent envoyer un email sur le Gmail destiné à la production.  Le service Mailhog ne doit pas être existant sur la CD. | test validé |
| Vérifier la non divulgation des variables d’environnements sensibles | Vérifier qu' aucune variable d’environnements considérée comme sensible ne soit écrite en clair sur le répertoire Github. | test validé |
|  |  |  |