**Documentation technique**  
[BTC\_ETH\_streaming\_data](https://github.com/gaetancorin/btc_eth_streaming_data)

Réalisé par Gaëtan Corin

Introduction au projet:

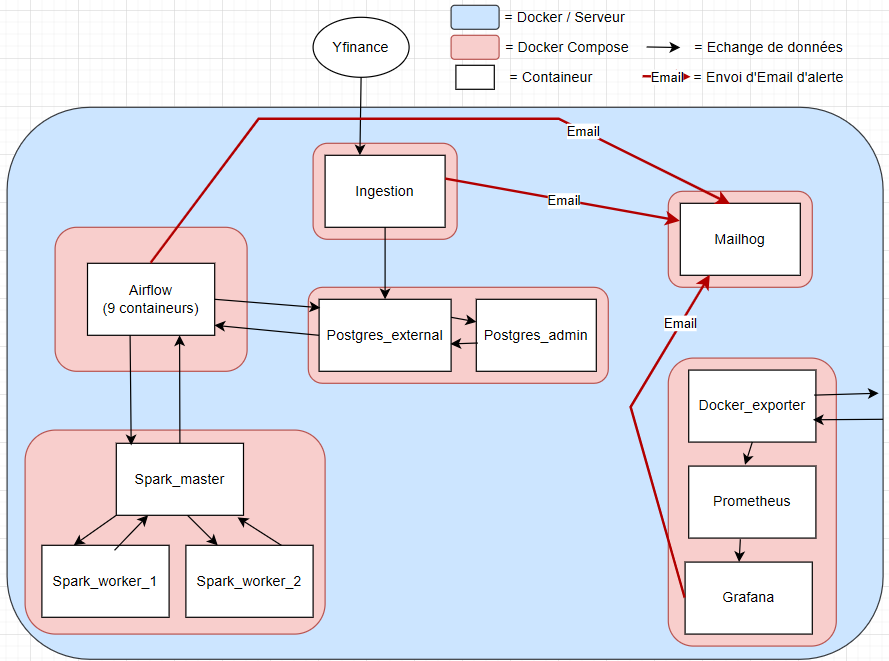
Le projet BTC\_ETH\_streaming\_data consiste en la récupération de données de crypto-monnaies BTC et ETH en données temps réels, puis en la création d’indicateurs technique demandant plus ou moins de puissance de calculs.

**Le document ci-dessous permet la montée en compétence technique et fonctionnelle de cette application, à destination de data-engineer.**

Prérequis:

Avoir Python3.11 sur son espace de travail, ainsi que docker.

Schéma architectural de l’application:



L’application est constituée de 6 services déployables en Docker de manière séparée, contenant en totalité 19 conteneurs.

Les services sont les suivants:

* Postgres

- **Postgres\_externa**l

- **Postgres\_admin**

* Mailhog

- **Mailhog**

* Ingestion

- **Ingestion**

* Airflow

- **Postgres**

- **Redis**

- **Airflow-apiserver**

- **Airflow-scheduler**

- **Airflow-dag-processor**

- **Airflow-worker**

- **Airflow-trigerrer**

- **Airflow-init**

- **Airflow-cli**

* Spark

- **Spark-master**

- **Spark-worker-1**

- **Spark-worker-2**

* Monitoring

- **Docker\_exporter**

(code python servant à récupérer les données de Docker sur un WSL)

- **Prometheus**

- **Grafana**

**Postgres\_externa**l:

Il s’agit du conteneur PostgreSQL contenant les données BTC, ETH, et le résultat des indicateurs qui ont été calculés.

C’est ce conteneur qui va créer le network “postgres\_streaming\_default”, sur lequel l’ensemble des autres conteneurs vont se connecter pour communiquer entre eux.

**Postgres\_admin**:

Il s’agit de l’interface graphique officiel de PostgreSQL. Cette interface permet de réaliser des requêtes ou bien d’analyser les contenues des différentes tables.

Ce conteneur est disponible à l'adresse:

<http://localhost:8081/>

**Mailhog**:

Il s'agit d’un outil de réception d’emails, adapté pour les tests en local ou pour le Continuous Integration de la CI/CD. Il permet de pouvoir facilement analyser les emails reçus, sans polluer une vrai boite email.

Ce conteneur est disponible à l’adresse:

<http://localhost:8025/>

**Ingestion**:

Il s’agit d’un programme Python permettant de récupérer les données boursières en temps réel avec la librairie Yfinance qui utilise un Websocket.

Une attention particulière est mise sur la résilience du programme afin d’obtenir un maximum de données lors du fonctionnement du conteneur.  
Les données sont ensuite transmises à la base de données PostgreSQL.  
En cas d’échec de la transmission de données, elles sont stockées dans le conteneurs dans des fichiers CSV, pour être de nouveau transmises de manière automatique lorsque la situation est rétablie.

De plus, un système d'email permet d’alerter lors d’échec de la transmission de données.

**Airflow**:

Il s’agit de la version Airflow 3, contenant de manière native les conteneurs “Postgres”, “Redis”, “Airflow-apiserver”, Airflow-scheduler”, “Airflow-dag-processor”, Airflow-trigerrer”, Airflow-init”, Airflow-cli”.

Airflow se connecte avec PostgreSQL pour récupérer les données, ainsi qu' avec Spark lorsque la puissance de calcul nécessite un calcul distribué.  
Sur ce Airflow, les dags les plus importantes permettant le calcul des indicateurs sont:

“btc\_avg\_3m\_5m\_indicator”, “eth\_avg\_3m\_5m\_indicator”, “spark\_btc\_eth\_gap\_avg\_5m\_indicator”.

Lors d’échec d’une de ces dags, un système d’email permet d’alerter afin de rétablir la situation le plus vite possible.

Le dashboard est disponible à l’adresse:

<http://localhost:8080/dags>

**Spark**:

Il s’agit du réseau de calculs distribué, constitué d’un conteneur Master (Spark-master) et de deux conteneurs Worker (Spark-worker-1 et Spark-worker-2)

Ce réseau de calculs distribué permet de réaliser le traitement des indicateurs nécessitant un map and reduce, comme c’est le cas pour la dag de Airflow “spark\_btc\_eth\_gap\_avg\_5m\_indicator”.

Le dashboard permettant la visualisation des calculs distribués est disponible à l’adresse:

<http://localhost:8082/>

**Docker\_exporter**:

Il s’agit d’un programme Python permettant de récupérer les données Docker sous WSL Windows.

Les données récupérées sont les états des conteneurs Docker, le CPU et mémoire utilisé par docker et pour l’ensemble des conteneurs, ainsi que le CPU et mémoire disponible pour l’ensemble des conteneurs.

Les données sont ensuite formaté pour Prometheus, puis disponible à l'adresse:

<http://localhost:8000/>

Il existe des outils ayant la même fonctionnalité comme “Cadvisor” ou “node\_exporter” qui permettent de récupérer ce type de données, mais ces outils sont adaptés pour fonctionner sous Linux, et l’application nécessite un monitoring sous WSL Windows. N’ayant pas d’outil existant pour ce genre de cas d’usage, il a fallu en créer un.

**Prometheus**:

Il s’agit du programme natif Prometheus permettant de se connecter à des données temps réel adapté Prometheus, puis à les stocker pour les rendre disponible à la visualisation.

L’état des connections de Prometheus sont disponible à l’adresse:

<http://localhost:9090/targets>

**Grafana**:

Il s’agit du programme natif Grafana permettant de créer des dashboard de données de monitoring permettant une supervision du système.

Sur ce cas d’usage, ce sont les données de Prometheus liées au données du conteneur “Docker-exporter” qui sont utilisées.

Une visualisation est ensuite créée, affichant l’état des conteneurs, l’utilisation CPU et mémoire par conteneur ainsi que pour l’ensemble des conteneurs, et la capacité CPU et mémoire de l’hôte.  
Un système d’alerte envoie un email lorsqu’un conteneur nécessitant d’être actif est éteint, ou bien lorsque le CPU ou la Mémoire de l’ensemble des conteneurs atteint la limite de l’hôte.

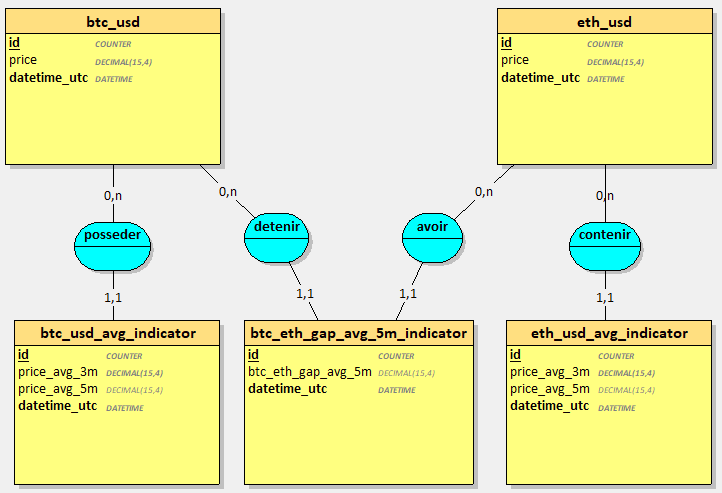
Le dashboard Grafana est disponible à l’adresse:

<http://localhost:3000/>

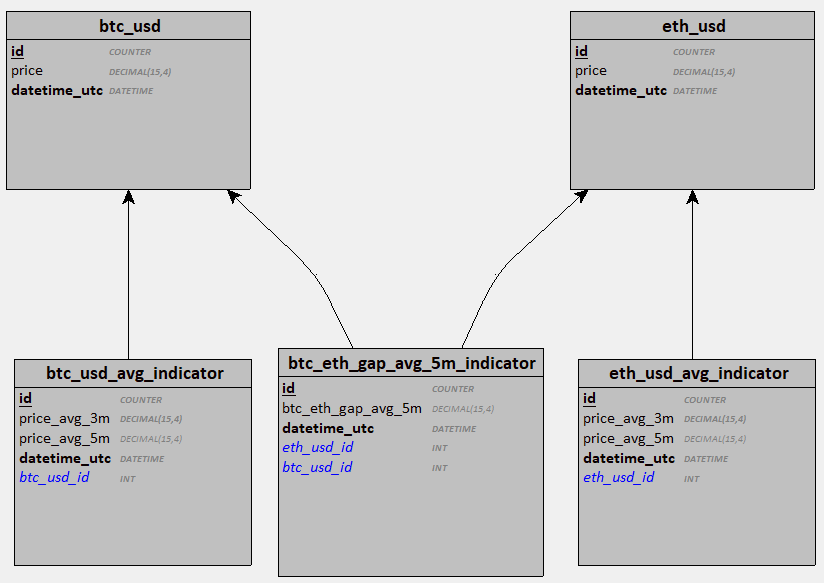
Schéma de données PostgreSQL:

Voici le schéma des tables de données PostgreSQL

**Schéma Modèle Conceptuel de Données (MCD):**



**Schéma Modèle Logique de Données (MLD):**



Le **modèle de données** pour cette application est constitué de 5 tables de données.

La table “**btc\_usd**”: Il s’agit du prix de la cryptomonnaie BTC en dollars et par minute(heure utc +0).

La table “**eth\_usd**”: Il s’agit du prix de la cryptomonnaie ETH en dollars et par minute(heure utc +0).

La table “**btc\_usd\_avg\_indicator**”: Il s’agit du résultat de l’indicateur technique calculé en réalisant la moyenne des 3 et 5 dernières minutes de la table btc\_usd. La datetime\_utc affiché correspond à la dernière minute des données de la table btc\_usd servant a calculé la moyenne

La table “**eth\_usd\_avg\_indicator**”: Il s’agit du résultat de l’indicateur technique calculé en réalisant la moyenne des 3 et 5 dernières minutes de la table eth\_usd. La datetime\_utc affiché correspond à la dernière minute des données de la table eth\_usd servant a calculé la moyenne

La table “**btc\_eth\_avg\_5m\_indicator**”: Il s’agit du résultat de la différence entre la valeur de btc\_usd et eth\_usd sur 5 valeurs. Ces 5 résultats qui représentent ces diférences de valeurs sont ensuite transformés en une moyenne, représentant la moyenne des 5 minutes des différences entre btc\_usd et eth\_usd. Le datetime représente la dernière minute de btc\_usd et eth\_usd utilisé sur le calcul.

Une relation One-To-Many entre les tables btc\_usd et btc\_usd\_avg\_indicator permet de réaliser une suppression en cascade de la table btc\_usd\_avg\_indicator lors de la suppression de la table btc\_usd. (en effet, les besoins métiers nécessite que ces deux données soient affichés sur le même graphique)

Une autre relation One-To-Many entre les tables eth\_usd et eth\_usd\_avg\_indicator permet de réaliser une suppression en cascade de la table eth\_usd\_avg\_indicator lors de la suppression de la table eth\_usd. (en effet, les besoins métiers nécessite que ces deux données soient affichés sur le même graphique)

En revanche, il n’y a pas de suppression en cascade sur les relations One-To-Many avec la table btc\_eth\_avg\_5m\_indicator. En effet, la logique métier de cette table est plus cohérente lorsque le graphique représentant ces données est analysé séparément des données de btc\_usd ou de eth\_usd. Il est donc normal dans la logique métier qu' il n’y ai pas besoin de suppression en cascade.

Les droits de ces tables sont en lecture / écriture pour le conteneur Ingestion ainsi que les conteneurs Airflow.

Lorsque ces donées seront utilisées afin de réaliser des graphiques visuels, seul le droit le lecture sera fourni.

Processus de traitement de données:

Voici le cycle de la pipeline de traitement de données pour la valeur BTC ainsi que pour ces indicateurs. (le cycle de la pipeline de ETH est similaire).

* Le conteneur Ingestion s’enregistre en Websocket sur la librairie Yfinance afin de recevoir les données BTC en prix USD chaque minute, dès que celle- ci est disponible.
* A réception de la donnée en temps réel, le prix est extrait, la daten heure et minute est créée, puis le tout est inséré en base de données sur la table “btc\_usd” dans le conteneur Postgres-external.
* Les conteneurs Airflow ont un planificateur qui va vérifier toutes les minutes la table “btc\_usd”. Si une nouvelle donnée une insérée, un dag “btc\_avg\_3m\_5m\_indicator” va récupéré les 5 dernières données représentant les 5 dernières minutes, calculé la moyenne des 3 dernières ainsi que des 5 dernières minutes, récupéré le datetime de la dernière minutes servant au calcul, puis créer un nouvelle enregistrement avec ces résultats dans la table de données “btc\_usd\_avg\_indicator”.
* Dans les conteneurs Airflow, une autre dag appelé “spark\_btc\_eth\_gap\_avg\_5m\_indicator” va récupéré les données des tables “ btc\_usd” et “eth\_usd” sur les 5 dernières minutes, puis va réaliser un map reduce avec les conteneurs Spark en calcul distribué.  
  Le MAP va servir a calculer la différence de prix entre BTC et ETH pour chacune des minutes, afin d’en faire 5 résultats.

Le REDUCE va servir à calculer la moyenne de ces 5 résultats, pour donner une moyenne des différences entre la valeur BTC et ETH sur 5 minutes.

Un enregistrement est réalisé avec cette valeur, ainsi que le datetime de la dernière minute des valeurs d'origine. Ce nouvel enregistrement est enregistré sur la table “btc\_eth\_avg\_5m\_indicator” du conteneur Postgres\_external.

Enfin, s' il y a un problème sur l’ingestion, la transformation dans les dags Airflow, ou bien sur la consommation de CPU, de mémoire, ou de conteneur qui s'éteint, un email est automatiquement envoyé pour alerter.

Processus de pipeline CI/CD:

Le Continuous Integration and Continuous Development (CI/CD) est une pipeline orchestrer avec Github Action, en lien avec le repository Github ou est situé le projet

<https://github.com/gaetancorin/btc_eth_streaming_data>  
L’objectif de cette pipeline est de tester le code lors de la phase CI, puis de la déployer lors de la phase CD

Voici le déroulement des événements lors de la phase CI:

* Création des Conteneurs du service Postgres
* Création des Conteneurs du service Mailhog
* Création des Conteneurs du service Monitoring
* Création des Conteneurs du service Spark
* Création des Conteneurs du service Airflow
* Création des Conteneurs du service Ingestion
* Réalisation des tests unitaires du service Ingestion

L’ensemble des actions de la CI se déroule sur les serveurs Github Action, et utilise les variables d’environnements “dev” paramétrées sur Github.  
En cas de problème nécessitant l’envoi d’un Email, les services sont paramétrés pour envoyer l’alerte sur Mailhog.

Voici le déroulement des événements lors de la phase CD:

* Création des Conteneurs du service Postgres
* Création des Conteneurs du service Monitoring
* Création des Conteneurs du service Spark
* Création des Conteneurs du service Airflow
* Création des Conteneurs du service Ingestion

L’ensemble des actions de la CD se déroule sur un ordinateur Windows sous WSL (principalement en local pour les tests de CD). La CD utilise les variables d’environnements “prod” paramétrées sur Github.  
En cas de problème nécessitant l’envoi d’un Email, les services sont paramétrés pour envoyer l’alerte sur une adresse Gmail, permettant à ceux qui travaillent sur le projet d’en être notifié.