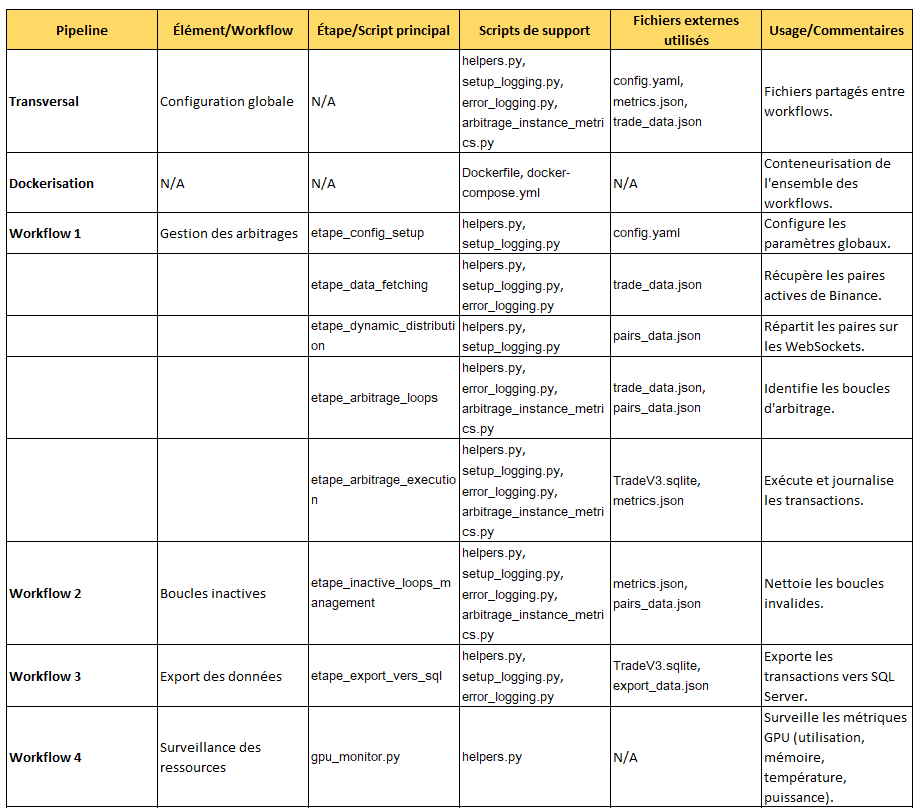
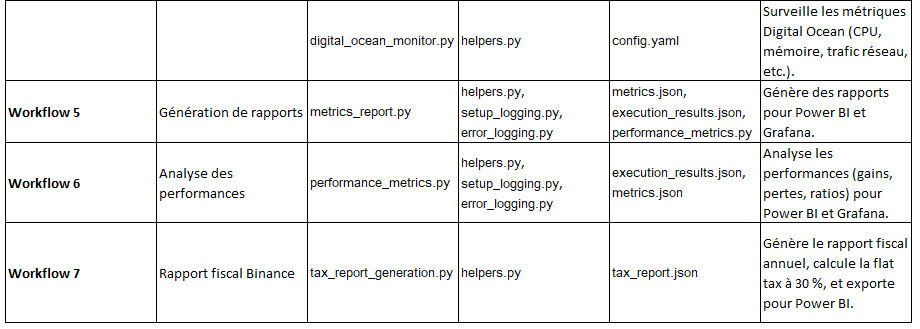
Moi :

Voici le pipeline.





Dorénavant je souhaite

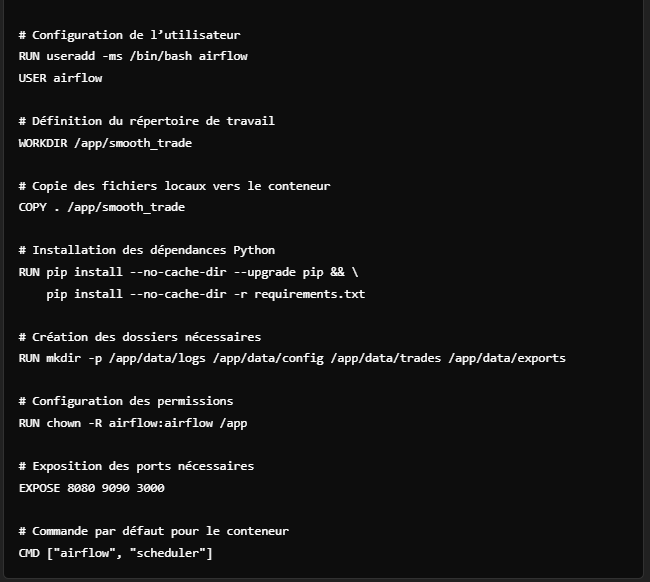
En 1/ constituer le répertoire smooth\_trade sous C/utilisateur avec les fichiers sous D/ actuellement (voir image 1). Voici le répertoire 2 actuel (voir image 2). Mais mon but ultime est de finir le développement du pipeline en travaillant de digital ocean, avec apache airflow, le tout conteneurisé et stocké sur mon repository github. Dans un premier temps, propose le dockerfile.paper-trading.custom à jour

ChatGPT :

Pour répondre à votre demande de création d’un Dockerfile.paper-trading.custom à jour et aligné avec votre pipeline, voici une proposition intégrale. Ce fichier configure un conteneur Docker capable de supporter le pipeline mentionné, y compris l'intégration de Digital Ocean, Apache Airflow, et d'autres dépendances nécessaires.

Dockerfile.paper-trading.custom



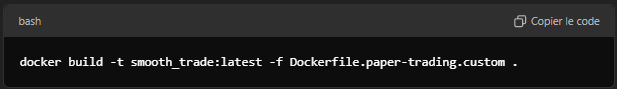


**Explications des sections :**

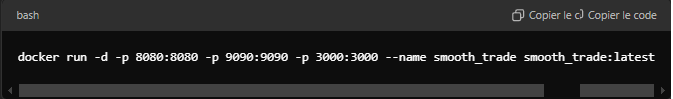
1. **Base Image :**
   * Utilisation de python:3.10-slim comme base légère et compatible avec vos scripts Python.
2. **Installation des Dépendances Système :**
   * Inclut des outils nécessaires tels que build-essential, libpq-dev, et Docker CLI pour les interactions avec Digital Ocean.
3. **Apache Airflow :**
   * Installation de la version spécifique d’Apache Airflow compatible avec votre pipeline.
4. **Utilisateur Airflow :**
   * Création d’un utilisateur spécifique airflow pour sécuriser l’exécution des tâches.
5. **Structure du Projet :**
   * Configuration de /app/smooth\_trade comme répertoire de travail.
   * Copie des fichiers locaux (smooth\_trade) vers le conteneur.
6. **Installation des Packages Python :**
   * Lecture du fichier requirements.txt (qui doit contenir toutes les bibliothèques nécessaires à votre pipeline).
7. **Création de Dossiers :**
   * Création de répertoires pour les journaux, les configurations, les échanges de données (trades), et les exports.
8. **Ports Exposés :**
   * Ports exposés pour Airflow (8080), Prometheus (9090), et Grafana (3000).
9. **Commande par Défaut :**
   * Configuration pour lancer le scheduler d’Airflow par défaut dans le conteneur.

**Étapes Suivantes :**

1. **Ajout de requirements.txt :**
   * Incluez les bibliothèques Python nécessaires à votre pipeline (exemple : pandas, requests, prometheus-client, etc.).
2. **Création d’une Image Docker :**



1. Lancement du Conteneur :



1. **Configuration d’Apache Airflow :**
   * Configurez vos DAGs pour orchestrer les tâches de votre pipeline, comme la gestion des arbitrages, l’export des rapports, et la surveillance des ressources.
2. **Déploiement vers Digital Ocean :**
   * Configurez une instance sur Digital Ocean pour héberger et exécuter votre pipeline.
3. **Stockage sur GitHub :**
   * Ajoutez votre projet à un dépôt GitHub pour le versionnement et l'accès collaboratif.

Souhaitez-vous un complément ou une adaptation ?

Moi :

je souhaite un complément d'information sur la mise en place intégrale

ChatGPT :

**Mise en place intégrale de votre pipeline sur Digital Ocean avec Apache Airflow et Docker**

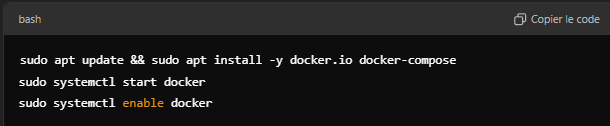
Pour que votre pipeline soit opérationnel sur Digital Ocean, avec une conteneurisation et une orchestration via Apache Airflow, voici un guide complet.

**Étape 1 : Préparer votre environnement local**

1. **Structure du projet** (répertoire smooth\_trade) :
2. **Vérifiez votre requirements.txt** : Incluez les bibliothèques nécessaires :
3. **Créer un docker-compose.yml** : Ce fichier gère les conteneurs (Airflow, Prometheus, Grafana).
4. **Lancer votre environnement local** :

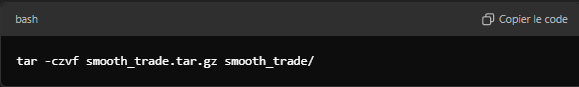
**Étape 2 : Déployer sur Digital Ocean**

1. **Créer une Droplet sur Digital Ocean :**
   * Choisissez une image Ubuntu 22.04 x64.
   * Allouez 4 vCPUs et 8 Go de RAM pour une performance optimale.
2. **Installer Docker et Docker Compose :**

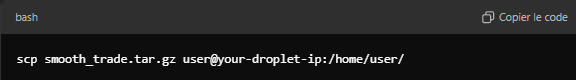


1. **Uploader votre projet sur la Droplet :**

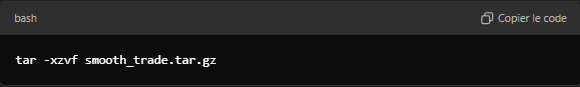
* Archivez le projet local :



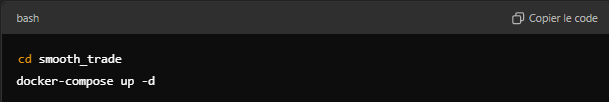
Transférez-le sur la Droplet via SCP :



Décompressez sur la Droplet :

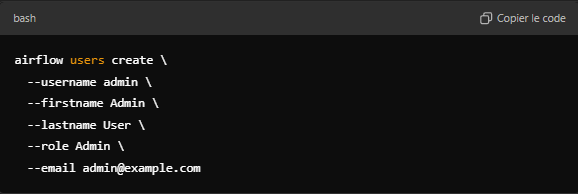


1. Démarrer les conteneurs sur Digital Ocean :



**Étape 3 : Configurer Airflow**

1. **Accédez à l’interface Web :**
   * Rendez-vous sur http://<your-droplet-ip>:8080.
   * Créez un utilisateur administrateur si nécessaire :



1. **Ajoutez vos DAGs :**

* Copiez les fichiers DAG dans dags/.
* Redémarrez le conteneur Airflow pour les charger :



1. **Configurer les connexions nécessaires :**

* Dans Airflow, configurez :
  + Une connexion Binance API pour gérer les clés API.
  + Une connexion Prometheus pour collecter les métriques.

**Étape 4 : Configurer Prometheus et Grafana**

1. **Prometheus** :
   * Accédez à http://<your-droplet-ip>:9090.
   * Vérifiez que les métriques sont disponibles via les endpoints configurés.
2. **Grafana** :
   * Accédez à http://<your-droplet-ip>:3000.
   * Connectez Prometheus comme source de données.
   * Importez le tableau de bord grafana\_dashboard.json.

**Étape 5 : Stockage et versionnement**

1. **Stockage GitHub :**
   * Initialisez un dépôt Git dans votre projet :



1. **Automatisation CI/CD :**
   * Configurez un workflow GitHub Actions pour automatiser les tests et les déploiements.

**Étape 6 : Surveillance et maintenance**

1. **Airflow :**
   * Surveillez l'exécution des DAGs via l'interface Web.
2. **Prometheus :**
   * Ajoutez des alertes pour les anomalies (par exemple, limites de clés API atteintes, erreurs GPU).
3. **Grafana :**
   * Affinez les panneaux pour une meilleure visualisation des performances et des métriques fiscales.

**Conclusion**

Vous disposez maintenant d’un pipeline conteneurisé, déployé sur Digital Ocean, et orchestré via Apache Airflow. Prometheus et Grafana assurent la collecte et la visualisation des métriques, tandis que GitHub gère le versionnement et la CI/CD.

Souhaitez-vous que je détaille une étape particulière ou que je vous assiste dans l’automatisation CI/CD ?

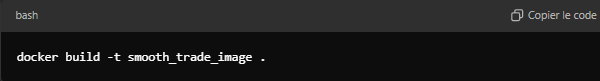
Moi :

Dockerfile.paper-trading.custom

CHATGPT :

**Prochaines étapes :**

1. Placez ce fichier dans la racine de votre projet sous le nom Dockerfile.
2. Construisez l’image Docker :



1. Lancez le conteneur avec Docker ou Docker Compose (selon votre configuration).
2. Configurez docker-compose.yml pour orchestrer plusieurs services si nécessaire.

Moi :

"dresse le pipeline intégral. En 2/ dresse le répertoire "smooth\_trade" sous C://utilisateur. En 3/ dresse le repository github" et il manque data\_fetching.py. En 1/ dresse le pipeline intégral. En 2/ dresse le répertoire "smooth\_trade" sous C://utilisateur. En 3/ dresse le repository github

1. Pipeline Intégral

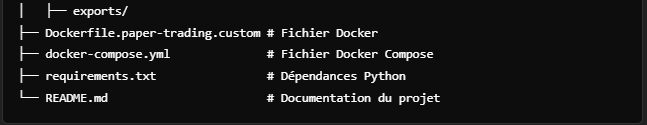




**2. Répertoire "smooth\_trade" sous C://utilisateur**

Voici le répertoire complet :

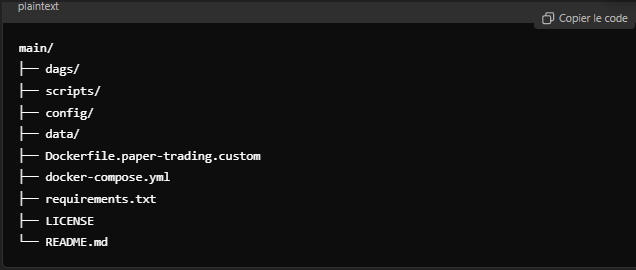




**3. Structure du Repository GitHub**

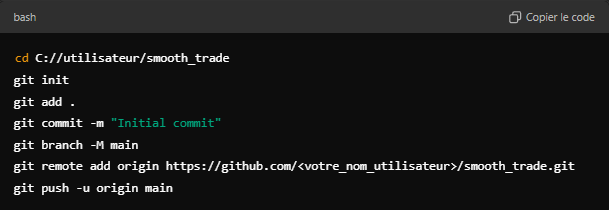
Votre repository GitHub pourrait suivre cette structure et inclure des branches pour différentes fonctionnalités :

**Repository smooth\_trade**



**Étapes pour uploader sur GitHub**

1. **Initialiser le dépôt :**



1. **vConfigurer les branches :**
   1. Branche main : Fonctionnalités validées.
   2. Branche feature/arbitrage : Développement du workflow d'arbitrage.
   3. Branche feature/monitoring : Surveillance des métriques.
   4. Branche feature/tax-report : Rapport fiscal.
2. **Configurer CI/CD :**
   1. Créez un fichier .github/workflows/deploy.yml pour automatiser les tests et déploiements.

Souhaitez-vous une assistance pour écrire le fichier deploy.yml pour CI/CD avec GitHub Actions ?

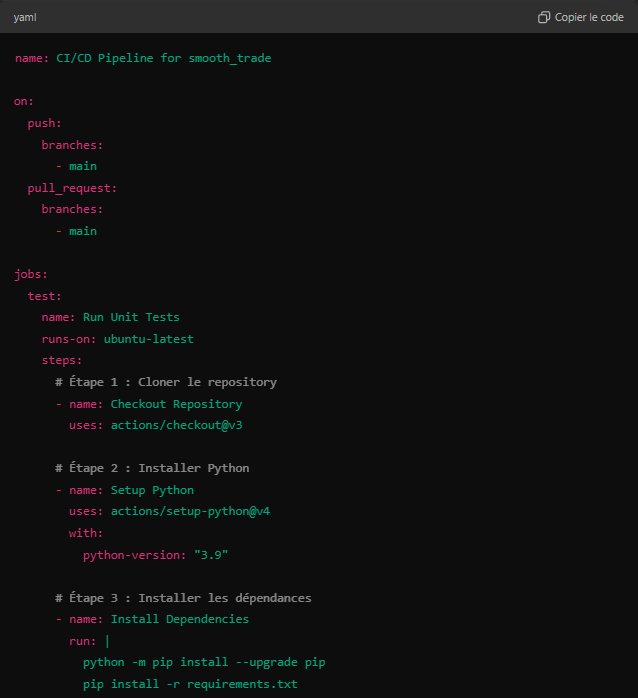
Moi :

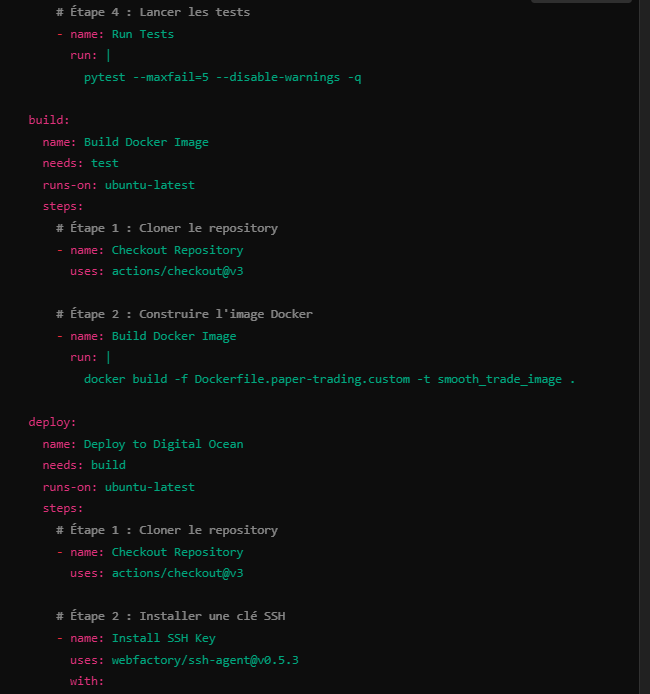
"Souhaitez-vous une assistance pour écrire le fichier deploy.yml pour CI/CD avec GitHub Actions ?" oui

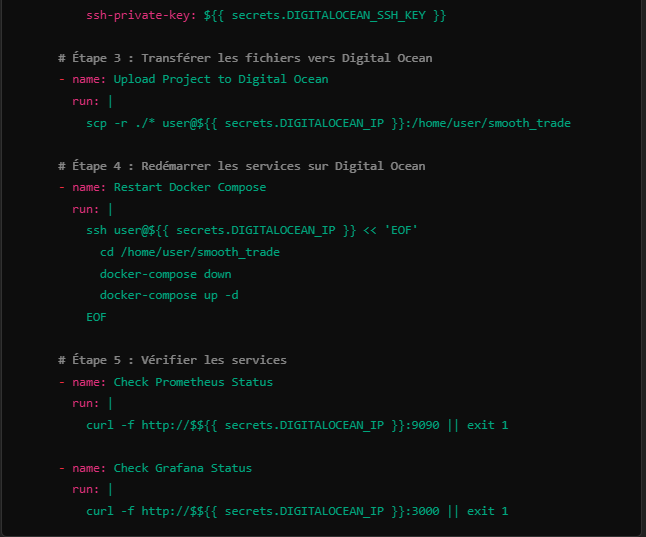
ChatGPT :

Voici une version détaillée et personnalisée du fichier deploy.yml pour mettre en place un pipeline CI/CD avec **GitHub Actions**. Ce fichier inclut les étapes pour tester votre code, construire vos conteneurs Docker, et déployer automatiquement sur votre **Droplet Digital Ocean**.

Fichier deploy.yml







**Explications des Étapes**

1. **test Job** :
   * Clone le repository.
   * Installe Python et les dépendances listées dans requirements.txt.
   * Exécute les tests unitaires avec pytest.
2. **build Job** :
   * Clone le repository.
   * Construit l'image Docker à l'aide du fichier Dockerfile.paper-trading.custom.
3. **deploy Job** :
   * Clone le repository.
   * Installe une clé SSH pour accéder à votre Droplet Digital Ocean.
   * Transfère tous les fichiers du projet sur votre Droplet.
   * Relance les services Docker via docker-compose.
   * Vérifie que Prometheus et Grafana fonctionnent correctement.

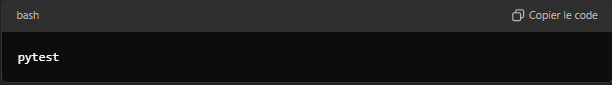
**Configuration des Secrets GitHub**

1. Accédez à **Settings** > **Secrets and variables** > **Actions**.
2. Ajoutez les secrets nécessaires :
   * DIGITALOCEAN\_IP: Adresse IP publique de votre Droplet.
   * DIGITALOCEAN\_SSH\_KEY: Clé SSH privée utilisée pour se connecter à votre Droplet.

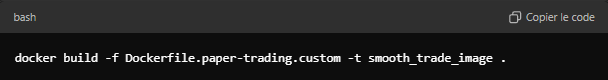
**Commandes Locales pour Vérification**

Avant de pousser sur GitHub, testez manuellement :

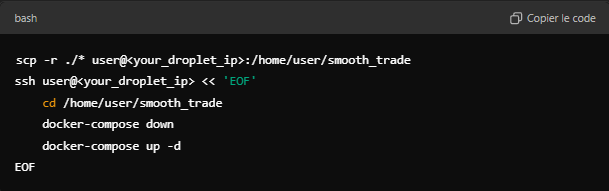
1. **Tests unitaires** :



1. **Construction Docker** :

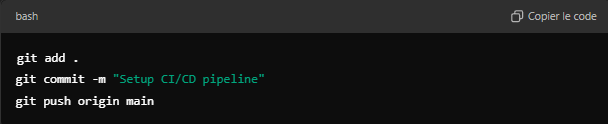


1. **Déploiement manuel** :



**Étapes Suivantes**

1. **Pousser le projet sur GitHub** :



1. **Surveiller le Workflow GitHub Actions** :
   1. Rendez-vous dans l'onglet **Actions** de votre repository GitHub.
   2. Surveillez l'exécution des jobs test, build, et deploy.
2. **Optimisations CI/CD** :
   1. Ajoutez des alertes pour les erreurs dans Grafana.
   2. Intégrez des notifications Slack ou par email en cas d'échec du pipeline.

Souhaitez-vous personnaliser ce fichier davantage ou approfondir une étape spécifique ?