

Rapport TP2 Ely SENE : Configuration d'un workflow GitHub Action pour publier une API météo sur Docker Hub

Introduction

Ce rapport détaille les étapes et les choix techniques effectués pour configurer un workflow GitHub Action permettant de transformer un wrapper en API météo, de le publier automatiquement sur Docker Hub et de le rendre disponible pour une utilisation via Docker. L'objectif est de créer un pipeline automatisé qui garantit la disponibilité de l'API météo à chaque mise à jour du code.

Sommaire

1. Configuration du workflow GitHub Action
2. Transformation du wrapper en API météo
3. Publication de l'image Docker sur Docker Hub
4. Bonus : Intégration de Hadolint dans le workflow

1. Configuration du workflow GitHub Action

Le workflow GitHub Action est déclenché à chaque push sur la branche principale (main) du repository. Il utilise une image Ubuntu comme environnement d'exécution et comporte trois étapes principales :

- Checkout repository : Cette étape récupère les fichiers du repository.
- Login to Docker Hub : Cette étape utilise l'action docker/login-action pour se connecter à Docker Hub en utilisant les informations d'identification stockées dans les secrets GitHub.
- Build and push Docker image : Cette étape construit l'image Docker à partir du Dockerfile fourni et la pousse vers Docker Hub avec le tag "latest".

2. Transformation du wrapper en API météo

Le fichier weather_wrapper.py fourni est transformé en une API météo, qui peut être exécutée dans un conteneur Docker. Les variables d'environnement pour l'API key et les coordonnées géographiques sont récupérées à partir des variables d'environnement. L'image Docker est configurée pour exécuter cette API météo par défaut.

3. Publication de l'image Docker sur Docker Hub

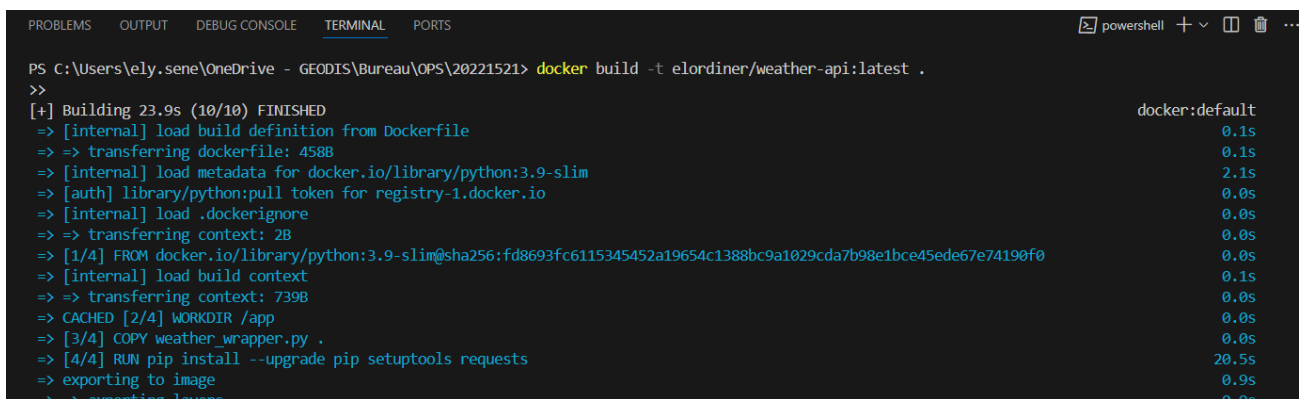
Une fois l'image Docker construite avec succès, elle est automatiquement poussée vers Docker Hub. Cela permet à quiconque d'accéder à l'API météo en tirant l'image Docker depuis Docker Hub et en l'exécutant localement ou dans un environnement cloud.

4. Test de l'API localement

Avant de publier l'image Docker sur Docker Hub, il est essentiel de tester l'API localement pour s'assurer qu'elle fonctionne comme prévu.

Étapes pour tester l'API localement :

1. Cloner le repository météo depuis GitHub.
2. Installer Docker localement
3. Créez un fichier .env à la racine du projet pour définir les variables d'environnement nécessaires, comme l'API key et les coordonnées géographiques. Par exemple :
4. Construire l'image Docker



```
PS C:\Users\ely.sene\OneDrive - GEODIS\Bureau\OPS\20221521> docker build -t elordiner/weather-api:latest .
>>
[+] Building 23.9s (10/10) FINISHED                                docker:default
=> [internal] load build definition from Dockerfile                0.1s
=> => transferring dockerfile: 458B                                0.1s
=> [internal] load metadata for docker.io/library/python:3.9-slim 2.1s
=> [auth] library/python:pull token for registry-1.docker.io      0.0s
=> [internal] load .dockerignore                                   0.0s
=> => transferring context: 2B                                       0.0s
=> [1/4] FROM docker.io/library/python:3.9-slim@sha256:fd8693fc6115345452a19654c1388bc9a1029cda7b98e1bce45ede67e74190f0 0.0s
=> [internal] load build context                                  0.1s
=> => transferring context: 739B                                       0.0s
=> CACHED [2/4] WORKDIR /app                                       0.0s
=> [3/4] COPY weather_wrapper.py .                                0.0s
=> [4/4] RUN pip install --upgrade pip setuptools requests        20.5s
=> exporting to image                                              0.9s
=> => exporting layers                                              0.9s
```

5. Lancer un conteneur Docker en spécifiant les variables d'environnement à partir du fichier .env.

```
docker run --env-file .env -p 8081:8081 weather-api:local
```

6. Tester l'API : Une fois le conteneur en cours d'exécution, on test l'API en utilisant curl ou le navigateur.

```
curl "http://localhost:8081/?lat=5.902785&lon=102.754175"
```

Bonus : Intégration de Hadolint dans le workflow

En cas d'erreurs détectées par Hadolint, le workflow échouera, empêchant ainsi la construction et la publication d'une image Docker contenant des erreurs potentielles.