Atelier 6 : Conteneurisation avec Docker

Prérequis : Application FastAPI fonctionnelle et MLflow configuré.

1) Objectifs :

a. Créer une image Docker pour l'application.

b. Utiliser les artefacts MLflow dans le conteneur.

2) Contenu :

a. Création d'un Dockerfile

b. Construction et push de l'image sur Docker Hub.

3) Livrables : Une image Docker déployable.

Atelier 6 : Conteneurisation avec Docker

Introduction

Cet atelier vous guide dans la création d'une image Docker pour conteneuriser votre application FastAPI tout en intégrant les artefacts de MLflow. L'objectif est de faciliter le déploiement et la portabilité de l'application.

I. Étape 01 : Installer Docker

Suivre la documentation au niveau du site officiel de Docker pour l'installation de Docker sur votre machine ubuntu.

Assurer que Docker est installé et en cours d'exécution en exécutant cette commande « docker –version »

II. Étape 02 : Préparer le Dockerfile

1) Créer le fichier Dockerfile :

Dans le répertoire principal du projet, créer un fichier nommé Dockerfile :

```
(venv) sirine@DESKTOP-4U94PKC:~/ml_project$ nano Dockerfile
```

a. Compléter le contenu du Dockerfile qui permet de :

- Copier le projet dans l'image docker
- Installer les dépendances du requirements.txt
- Lance le web service grâce à FastAPI (ou équivalent) pour tester la prédiction.

III. Étape 03 : Construire l'image Docker

Construire l'image :

Utiliser la commande suivante pour construire l'image :

```
docker build -t fastapi-mlflow-app .
```

Remplacer **obligatoirement** le nom de l'image fastapi-mlflow-app par un nom d'image composé comme suit votre_prénom_votre_nom_vote_classe_mlops

Vérifier l'image :

Lister les images Docker disponibles en tapant la commande suivante :

« docker images »

```
(venv) sirine@DESKTOP-4U94PKC:~/ml_project$ docker images
REPOSITORY          TAG       IMAGE ID       CREATED              SIZE
fastapi-mlflow-app  latest    9de41c17bb84   About a minute ago   2.84GB
```

IV. Étape 04 : Tester l'image localement

1) Exécuter le conteneur Docker

2) Tester l'API

V. Étape 05 : Pousser l'image sur Docker Hub

Docker Hub est une plateforme en ligne qui permet de stocker, partager et distribuer des images Docker, facilitant leur réutilisation et leur déploiement. Pour publier une image sur Docker Hub, vous devez d'abord vous connecter à votre compte, associer un nom spécifique à l'image pour qu'elle soit identifiable, puis l'envoyer sur la plateforme. Une fois l'image publiée, elle devient accessible depuis n'importe quelle machine connectée, simplifiant son utilisation et son partage.

1) Se connecter à Docker Hub :

Connecter avec à Docker Hub en tapant la commande « docker login »

```
(venv) sirine@DESKTOP-4U94PKC:~/ml_project$ docker login

USING WEB-BASED LOGIN
To sign in with credentials on the command line, use 'docker login -u <username>'

Your one-time device confirmation code is: MTJP-FQTX
Press ENTER to open your browser or submit your device code here: https://login.docker.com/activate

Waiting for authentication in the browser…
```

2) Taguer l'image :

Donner un tag à l'image pour la pousser sur Docker Hub :

3) Push l'image sur DockerHub

4) Vérifier sur Docker Hub l'existence de l'image

VI. Étape 06 : Automatisation avec le Makefile

Ajouter les commandes au Makefile :

Modifier le fichier « Makefile » pour inclure des tâches Docker pour la création de l'image, le push de l'image ainsi que le lancement du conteneur

Livrables

1) **Dockerfile** : Un fichier fonctionnel pour créer une image Docker.
2) **Image Docker** : Disponible localement et sur Docker Hub.
3) **Makefile** : Automatisation des tâches de construction, exécution et publication de l'image.