Mise en place du Docker pour le projet HBnB

Pour mettre votre projet dans un conteneur Docker, suivez ces étapes pour créer une image Docker et exécuter votre projet dans un environnement isolé. Voici les étapes détaillées :

1. Préparer votre projet pour Docker

Assurez-vous que :

- Tous les fichiers nécessaires au projet (code, configurations, etc.) sont dans un répertoire dédié.
- Les dépendances (comme Python, les bibliothèques, etc.) sont listées dans un fichier (par exemple, requirements.txt pour Python).

2. Créer un fichier Dockerfile

Un Dockerfile est un script contenant les instructions pour créer une image Docker. Placez-le dans le répertoire principal de votre projet.

Exemple pour un projet Python avec SQLAlchemy :

```dockerfile

# Utiliser une image de base Python

FROM python:3.10-slim


# Définir le répertoire de travail dans le conteneur

WORKDIR /app


# Copier les fichiers du projet dans le conteneur

COPY . /app


# Installer les dépendances

RUN pip install --no-cache-dir -r requirements.txt


# Exposer le port (par exemple, si votre application utilise Flask ou FastAPI)

EXPOSE 5000


# Définir la commande pour lancer l'application

CMD ["python", "main.py"]  # Remplacez "main.py" par le fichier d'entrée de votre projet

```

3. Créer un fichier requirements.txt

Listez toutes les dépendances Python de votre projet :

```txt
flask
sqlalchemy
mysql-connector-python
# Ajoutez les autres dépendances ici
```

Ici pour le le projethbnb

```txt
Flask==2.1.3
flask-restx==0.5.1
marshmallow==3.18.0
requests==2.31.0
flask-bcrypt==1.0.1
flask-jwt-extended==4.4.4
pytest==7.3.1
sqlalchemy==1.4.47
flask-sqlalchemy==2.5.1
Werkzeug==2.1.2
pytest==7.3.1
pytest-flask
pytest-cov
```

4. Créer une image Docker

1. Ouvrez un terminal dans le répertoire contenant le Dockerfile.
2. Construisez l'image avec la commande suivante :

```bash
```

docker build -t mon-projet .

```

- -t mon-projet : Nom que vous voulez donner à l'image.
- . : Indique que le contexte de construction est le répertoire actuel.

---

**## 5. Lancer le conteneur**

Une fois l'image créée, exécutez un conteneur basé sur cette image :

```bash

docker run -d -p 5000:5000 --name mon-projet-container mon-projet

```

- -d : Exécute le conteneur en arrière-plan.
- -p 5000:5000 : Lie le port 5000 du conteneur au port 5000 de votre machine hôte.
- --name mon-projet-container : Donne un nom au conteneur.
- mon-projet : Nom de l'image que vous avez créée.

6. Tester l'application

Accédez à votre application via http://localhost:5000 (ou le port que vous avez configuré).

7. Ajouter un fichier .dockerignore (optionnel)

Pour éviter de copier des fichiers inutiles dans l'image Docker, créez un fichier .dockerignore dans le répertoire principal :

```txt

__pycache__/

*.pyc

*.pyo

.env

```

8. Utiliser Docker Compose (facultatif, pour des services multiples)

Si votre projet nécessite plusieurs services (par exemple, une base de données MySQL), créez un fichier docker-compose.yml pour gérer plusieurs conteneurs.

Exemple avec une base de données MySQL :

```yaml

version: '3.8'

services:

  web:

    build:

      context: .

    ports:

      - "5000:5000"

    environment:

      - DATABASE_URL=mysql+pymysql://root:password@db:3306/mon_projet

    depends_on:

      - db

  db:

    image: mysql:8.0

    environment:

      MYSQL_ROOT_PASSWORD: password

      MYSQL_DATABASE: mon_projet

    ports:

      - "3306:3306"

Lancez les services avec :

```bash

docker-compose up -d

```

9. Vérifier et déboguer

- Liste des conteneurs actifs :

```bash

docker ps

```

- Afficher les journaux du conteneur :

```bash
```

docker logs mon-projet-container

```

- Entrer dans un conteneur pour le déboguer :

```bash

docker exec -it mon-projet-container /bin/bash

```

Avec ces étapes, votre projet sera fonctionnel dans Docker, ce qui facilite le déploiement et l'exécution sur différents environnements.
```