LAB04 – Flask API avec SQLite et Docker

Objectif

Continuer le LAB03 en ajoutant :

1. L'utilisation d'une base de données SQLite dans un conteneur Docker.
2. Un fichier `docker-compose.yaml` pour lancer tous les conteneurs de l'application.
3. Un script Python `db-test.py` pour tester l'accès de l'API à la base SQLite.

Structure du projet

```
docker-lab04/
│
├─ app.py              # API Flask avec CRUD et SQLite
├─ requirements.txt    # Dépendances Python (Flask)
├─ Dockerfile          # Image Docker pour l'API
├─ docker-compose.yaml # Lancement du conteneur avec volume pour SQLite
├─ db-test.py          # Script Python pour tester l'API
└─ data/               # Dossier créé par Docker pour la base SQLite
```

---

1. API Flask avec SQLite

- `app.py` expose les routes suivantes :

Méthode	URL	Description
GET	/books	Liste tous les livres
POST	/books	Ajoute un nouveau livre
PUT	/books/<id>	Met à jour un livre existant
DELETE	/books/<id>	Supprime un livre

- SQLite stocke les données dans `/app/data/books.db`.
- La base est initialisée automatiquement au démarrage du conteneur si elle n'existe pas.
- Trois livres sont insérés par défaut si la table est vide.

---

2. Dockerfile

- Image basée sur `python:3.11-slim`.
- Copie `requirements.txt` et installe Flask.
- Copie `app.py`.

- Expose le port 5000 et lance l'application avec `python app.py`.

---

3. Docker Compose

- `docker-compose.yaml` :

```
services:
  api:
    build: .
    container_name: flask-api-sqlite
    ports:
      - "5000:5000"
    volumes:
      - sqlite-data:/app/data

volumes:
  sqlite-data:
```

- Volume Docker `sqlite-data` pour persistance de la base SQLite.
- Permet de conserver les données même si le conteneur est supprimé.

---

4. Script de test db-test.py

```
import requests

API = "http://localhost:5000/books"

print("Lecture initiale :")
print(requests.get(API).json())

print("\nAjout d'un livre :")
new_book = {"title": "Test Book", "author": "Tester", "year": 2030}
print(requests.post(API, json=new_book).json())

print("\nNouvelle liste :")
print(requests.get(API).json())
```

- Vérifie l'accès à l'API et la persistance des données dans SQLite.

---

5. Commandes utilisées

```
# 1. Se placer dans le projet
cd /mnt/c/Users/dorsa/docker-lab04
```

```
# 2. Supprimer ancien conteneur
docker rm -f flask-api-sqlite

# 3. Construire et lancer le conteneur
docker-compose up --build -d

# 4. Vérifier que l'API fonctionne
curl http://localhost:5000/books

# 5. Tester avec le script Python
pip install requests
python3 db-test.py

# 6. Vérifier la base SQLite dans le volume
docker run --rm -v docker-lab04_sqlite-data:/data busybox ls /data
```

---

6. Résultat attendu

- L'API retourne la liste des livres initiaux.
- Le CRUD fonctionne pour ajouter, mettre à jour et supprimer des livres.
- Le volume Docker contient le fichier `books.db` pour persistance des données.
- Le script `db-test.py` montre que l'API fonctionne correctement et peut accéder à la base SQLite.