API CONTAINERS DOCKER

DATE: 17/06/2024

Rapport d'Intervention : Développement et Containerisation d'une Application FastAPI

Objectif

Créer, configurer, et containeriser une application FastAPI pour gérer une liste d'éléments avec une base de données MySQL

Étapes Suivies et Commandes Utilisées

1. Création d'un Nouveau Projet FastAPI

Endpoints

- **GET /items/** : Récupérer une liste d'éléments.
- **POST /items/** : Créer un nouvel élément (détails stockés dans la base de données MySQL).
- **GET /items/{item_id}** : Récupérer les détails d'un élément spécifique par son ID.

Étapes:

1. **Initialiser le Projet FastAPI** :
   - **Commande** :

```
mkdir fastapi_project
cd fastapi_project
python -m venv env
source env/bin/activate
pip install fastapi uvicorn sqlalchemy pymysql
```

2. Créer les Fichiers Nécessaires :

: Fichier principal de l'application.

: Définir les modèles SQLAlchemy pour les éléments.

: Définir les schémas Pydantic pour la validation.

: Configurer la connexion à la base de données.

Écrire le Code FastAPI :

main.py :

```
from fastapi import FastAPI, HTTPException
from sqlalchemy.orm import Session
from . import models, schemas, database
app = FastAPI()

# Création de la base de données
models.Base.metadata.create_all(bind=database.engine)

@app.get("/items/", response_model=List[schemas.Item])
def read_items(skip: int = 0, limit: int = 10, db: Session = Depends(database.get_db)):
    items = db.query(models.Item).offset(skip).limit(limit).all()
    return items

@app.post("/items/", response_model=schemas.Item)
def create_item(item: schemas.ItemCreate, db: Session = Depends(database.get_db)):
    db_item = models.Item(**item.dict())
    db.add(db_item)
    db.commit()
    db.refresh(db_item)
    return db_item

@app.get("/items/{item_id}", response_model=schemas.Item)
def read_item(item_id: int, db: Session = Depends(database.get_db)):
    item = db.query(models.Item).filter(models.Item.id == item_id).first()
    if item is None:
        raise HTTPException(status_code=404, detail="Item not found")
    return item
```

models.py :

```
from sqlalchemy import Column, Integer, String
from .database import Base
```

```
class Item(Base):
    __tablename__ = "items"
    id = Column(Integer, primary_key=True, index=True)
    name = Column(String, index=True)
    description = Column(String, index=True)
```

schemas.py

```
from pydantic import BaseModel

class ItemBase(BaseModel):
    name: str
    description: str

class ItemCreate(ItemBase):
    pass

class Item(ItemBase):
    id: int

    class Config:
        orm_mode = True
```

database.py

```
from sqlalchemy import create_engine
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker

SQLALCHEMY_DATABASE_URL = "mysql+pymysql://user:password@localhost/dbname"

engine = create_engine(SQLALCHEMY_DATABASE_URL)
SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)
Base = declarative_base()

def get_db():
    db = SessionLocal()
    try:
        yield db
    finally:
        db.close()
```

2. Configuration de la Base de Données MySQL

Étapes :

1. **Installer MySQL** :
    - **Commande** :

```
sudo apt-get update

sudo apt-get install mysql-server
```

Configurer la Base de Données :

- **Commande** :

```
sudo mysql

CREATE DATABASE fastapi_db;

CREATE USER 'user'@'localhost' IDENTIFIED BY 'password';

GRANT ALL PRIVILEGES ON fastapi_db.* TO 'user'@'localhost';

FLUSH PRIVILEGES
```

3. Containerisation de l'Application

Écriture du Dockerfile

Dockerfile :

```
FROM python:3.9

WORKDIR /app

COPY requirements.txt .

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

COPY . .

CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8000"]
```

3. Containerisation de l'Application

Écriture du Dockerfile

Dockerfile :

```
FROM python:3.9
```

```
WORKDIR /app

COPY requirements.txt .

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

COPY . .

CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8000"]
```

Écriture du Fichier docker-compose.yml

docker-compose.yml :

```
version: '3.8'

services:
  db:
    image: mysql:5.7
    environment:
      MYSQL_DATABASE: fastapi_db
      MYSQL_USER: user
      MYSQL_PASSWORD: password
      MYSQL_ROOT_PASSWORD: password
    ports:
      - "3306:3306"

  web:
    build: .
    ports:
      - "8000:8000"
    depends_on:
      - db
    environment:
      SQLALCHEMY_DATABASE_URL: mysql+pymysql://user:password@db/fastapi_db
```

Commandes Utilisées :

Build et Start des Services:

- **Commande**:

```
docker-compose up --build
```

Start des Services:

- **Commande:**

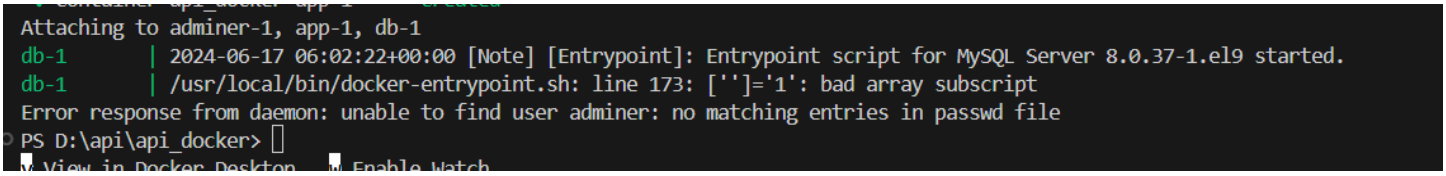
docker-compose start

stop Services:

- **Commande**:

docker-compose down

4. Problèmes Rencontrés et Solutions



Problème 1 : "unable to find user adminer: no matching entries in passwd file"

- **Cause** : Utilisateur adminer non existant dans l'image.
- **Solution** : Créer l'utilisateur dans le Dockerfile.

Problème 2 : Connexion à la Base de Données MySQL

- **Cause** : Mauvaise configuration de l'URL de connexion.
- **Solution** : Vérifier et corriger SQLALCHEMY_DATABASE_URL dans database.py et docker-compose.yml.