Lab 2: Dockerized Flask Web Application

You can find the complete project here :- >>>>>> My Repo <<<<<<

1. Project Overview

This project demonstrates containerization of a Python/Flask web application with a PostgreSQL database using Docker and Docker Compose. It includes:

- Flask (Python) web server
- PostgreSQL database
- Docker containerization
- Persistent storage for database using Docker volumes
- Docker Compose orchestration

2. Prerequisites

1. Docker installed
2. Docker Compose installed
3. Text editor (VS Code, Sublime, etc.)

3. Project Structure

```
├── Dockerfile
├── docker-compose.yml
├── app.py
└── postgres_data/ (auto-created volume)
```

4. Setup Steps

Step 1: Dockerfile Configuration

```
# Base image
FROM python:3.9-slim

# Environment setup
ENV PYTHONDONTWRITEBYTECODE 1
ENV PYTHONUNBUFFERED 1

# Install dependencies
RUN apt-get update && \
    apt-get install -y --no-install-recommends gcc python3-dev && \
    pip install --no-cache-dir flask psycopg2-binary

# Application setup
WORKDIR /app
COPY app.py .
EXPOSE 5000

# Runtime command
CMD ["flask", "run", "--host", "0.0.0.0"]
```

Key Features:

- Uses official Python 3.9 slim image
- Installs required system dependencies
- Copies application code (app.py)
- Exposes port 5000 for Flask
- Configures Flask to listen on all interfaces

Step 2: Docker Compose Configuration

File: docker-compose.yml

```
version: '3.8'

services:
  web:
    build: .
    ports:
      - "5000:5000"
    environment:
      - DATABASE_URL=postgresql://appuser:apppass@db:5432/appdb
    depends_on:
      - db

  db:
    image: postgres:13
    volumes:
      - postgres_data:/var/lib/postgresql/data
    environment:
      - POSTGRES_USER=appuser
      - POSTGRES_PASSWORD=apppass
      - POSTGRES_DB=appdb

volumes:
  postgres_data:
```

Key Components:

1. Web Service:
    - Builds from local Dockerfile
    - Maps host port 5000 → container port 5000
    - Connects to database using environment variables
    - Depends on database service
2. Database Service:
    - Uses PostgreSQL 13 image
    - Persistent volume for data storage
    - Pre-configured credentials:
        - User: appuser
        - Password: apppass
        - Database: appdb

Step 3: Flask Application

File: app.py

```
from flask import Flask
import os
import psycopg2

app = Flask(__name__)

@app.route('/')
def hello():
    try:
        conn = psycopg2.connect(os.getenv('DATABASE_URL'))
        conn.close()
        return "Hello! Database connection successful!"
    except Exception as e:
        return f"Hello! Database connection failed: {str(e)}"

if __name__ == '__main__':
    app.run()
```

Functionality:

- Simple endpoint (/) that tests database connectivity
- Uses psycopg2 for PostgreSQL connection
- Returns connection status message

5. Deployment

Command Line Execution

```
# Build and start containers
$ docker-compose up --build

# Stop containers (CTRL+C to stop in foreground)
$ docker-compose down
```

6. Verification

1. Access application:

   http://localhost:5000

2. Successful response: