Docker Project

Objective: To develop a two-tier flask application that adds value to the database and shows the value.

Files used in the application:

- Application backend code (taken from opensource)
- Application frontend code (taken from opensource)
- Back-end Dockerfile
- Database Dockerfile
- Volume for the Database
- Docker compose file for the 2 containers

GitHub repo used in this project: https://github.com/BineethSharma/Docker-two-tier.git

Step 1: Creating the app.py (backend) file, here the app run on the port 5000:

```
import os

from flask import Flask, render_template, request, redirect, url_for

from flask_mysqldb import MySQL


app = Flask(__name__)


# Configure MySQL from environment variables

app.config['MYSQL_HOST'] = os.environ.get('MYSQL_HOST', 'localhost')

app.config['MYSQL_USER'] = os.environ.get('MYSQL_USER', 'default_user')

app.config['MYSQL_PASSWORD'] = os.environ.get('MYSQL_PASSWORD',
'default_password')

app.config['MYSQL_DB'] = os.environ.get('MYSQL_DB', 'default_db')


# Initialize MySQL

mysql = MySQL(app)
```

```
@app.route('/')
def hello():
    cur = mysql.connection.cursor()
    cur.execute('SELECT message FROM messages')
    messages = cur.fetchall()
    cur.close()
    return render_template('index.html', messages=messages)

@app.route('/submit', methods=['POST'])
def submit():
    new_message = request.form.get('new_message')
    cur = mysql.connection.cursor()
    cur.execute('INSERT INTO messages (message) VALUES (%s)', [new_message])
    mysql.connection.commit()
    cur.close()
    return redirect(url_for('hello'))

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000, debug=True)
```

STEP 2: Creating the index.html file which is stored inside the templates directory

```
<!DOCTYPE html>
<html>
<head>
    <title>Flask App</title>
    <style>
```

```
        /* ... (your CSS styles) */
    </style>
</head>
<body>
    <div class="container">
        <h1>Hello Dosto, Let's make a 2 Tier App with Docker Compose!</h1>
        {% for message in messages %}
            <p>{{ message[0] }}</p>
        {% endfor %}

        <form action="/submit" method="post">
            <input type="text" name="new_message" placeholder="Enter a new message">
            <input type="submit" value="Submit">
        </form>
    </div>
</body>
</html>
```

STEP 3: Creating the requirements.txt file:

```
Flask==2.0.1
Flask-MySQLdb==0.2.0
requests==2.26.0
```

STEP 4: Creating the Dockerfile:

```
# Use an official Python runtime as the base image
FROM python:3.9-slim

# Set the working directory in the container
WORKDIR /app

# install required packages for system
RUN apt-get update \
    && apt-get upgrade -y \
    && apt-get install -y gcc default-libmysqlclient-dev pkg-config \
    && rm -rf /var/lib/apt/lists/*

# Copy the requirements file into the container
COPY requirements.txt .

# Install app dependencies
RUN pip install mysqlclient
RUN pip install --no-cache-dir -r requirements.txt

# Copy the rest of the application code
```

```
COPY . .

# Specify the command to run your application
CMD ["python", "app.py"]
```

STEP 5: Creating the message.sql table to store the value in the two_tier database

```
CREATE TABLE messages (
    id INT AUTO_INCREMENT PRIMARY KEY,
    message TEXT
);
```

STEP 6: Adding the execution steps in the README.md file:

```
# Flask App with MySQL Docker Setup

This is a simple Flask app that interacts with a MySQL database. The app allows users to submit messages, which are then stored in the database and displayed on the frontend.

## Prerequisites

Before you begin, make sure you have the following installed:

- Docker
- Git (optional, for cloning the repository)
```

## Setup

1. Clone this repository (if you haven't already):

   ```bash
   git clone https://github.com/your-username/your-repo-name.git
   ```

2. Navigate to the project directory:

   ~~~bash
   cd your-repo-name

3. Create a `.env` file in the project directory to store your MySQL environment variables:

   ```bash
   touch .env
   ~~~

4. Open the `.env` file and add your MySQL configuration:

   ```
   MYSQL_HOST=mysql
   MYSQL_USER=your_username
   MYSQL_PASSWORD=your_password
   MYSQL_DB=your_database
   ```

## Usage

1. Start the containers using Docker Compose:

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

2. Access the Flask app in your web browser:

   - Frontend: http://localhost
   - Backend: http://localhost:5000

3. Create the `messages` table in your MySQL database:

   - Use a MySQL client or tool (e.g., phpMyAdmin) to execute the following SQL commands:

     ```sql
     CREATE TABLE messages (
         id INT AUTO_INCREMENT PRIMARY KEY,
         message TEXT
     );
     ```

4. Interact with the app:

   - Visit http://localhost to see the frontend. You can submit new messages using the form.
   - Visit http://localhost:5000/insert_sql to insert a message directly into the `messages` table via an SQL query.

## Cleaning Up

To stop and remove the Docker containers, press `Ctrl+C` in the terminal where the containers are running, or use the following command:

```bash
docker-compose down
```

## Notes

- Make sure to replace placeholders (e.g., `your_username`, `your_password`, `your_database`) with your actual MySQL configuration.

- This is a basic setup for demonstration purposes. In a production environment, you should follow best practices for security and performance.

- Be cautious when executing SQL queries directly. Validate and sanitize user inputs to prevent vulnerabilities like SQL injection.

- If you encounter issues, check Docker logs and error messages for troubleshooting.

```
```

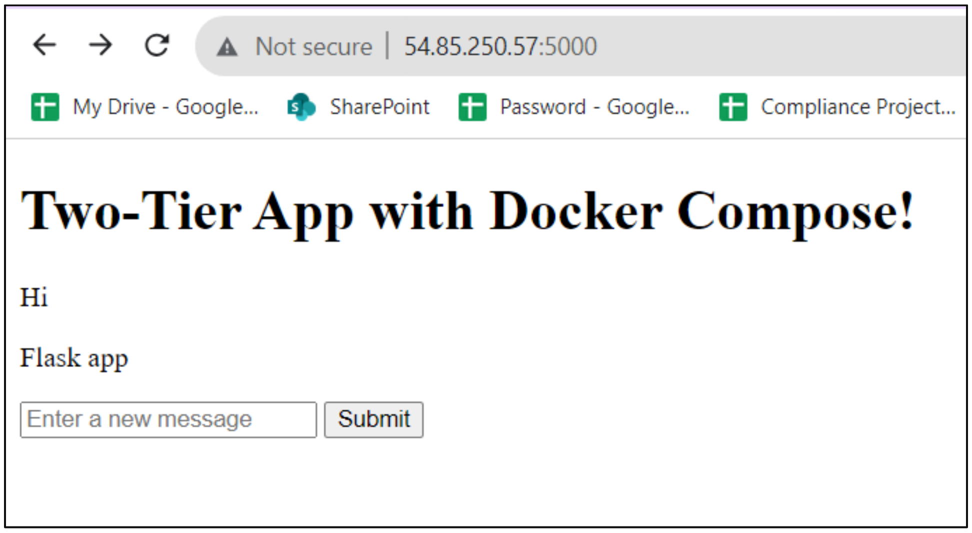
STEP 7: Creating the docker-compose file which runs both the backend and database containers:

```
version: '3'
services:

  backend:
    build:
      context: .
    ports:
      - "5000:5000"
    environment:
      MYSQL_HOST: mysql
      MYSQL_USER: root
      MYSQL_PASSWORD: test@123
      MYSQL_DB: two_tier
    depends_on:
      - mysql

  mysql:
    image: mysql:5.7
    environment:
      MYSQL_ROOT_PASSWORD: test@123
      MYSQL_USER: devops
      MYSQL_DATABASE: two_tier
      MYSQL_PASSWORD: devops
    volumes:
      - my-datavolume:/var/lib/mysql  # Mount the volume for MySQL data storage
volumes:
  my-datavolume:
```

OUTPUT running on port 5000:



Two-Tier App with Docker Compose!

Hi

Flask app