Vishal S
1BM21IS206

Assignment 1

Dockerized Multi-Container Web Application with Networking and Container Management

Objective/Problem Statement:

To build a Python-based web application using Flask with CRUD functionality, containerize the application and SQLite database, create a custom Docker network, and programmatically manage the containers.

Step-by-Step Process:

1. Web Application Development

Objective:
Create a Python Flask application with basic CRUD operations using SQLite as the database.

- Created a Python `app.py` file that includes:
  - Flask routes to handle basic CRUD (Create, Read, Update, Delete) functionality.
  - SQLite database to store and perform operations on the data.
  - This file helps in User Creation,Updating User Information and Deleting Users altogether.

```python
from flask import Flask, render_template, request, jsonify
import sqlite3



app = Flask(__name__)
DATABASE = 'data.db'


def get_db_connection():
    conn = sqlite3.connect(DATABASE)
    conn.row_factory = sqlite3.Row
    return conn


@app.route('/')
def index():
    return render_template('index.html')
```

```python
@app.route('/create', methods=['POST'])
def create():
    data = request.json
    name = data['name']
    conn = get_db_connection()
    conn.execute('INSERT INTO users (name) VALUES (?)', (name,))
    conn.commit()
    conn.close()
    return jsonify({"message": "User created!"})


@app.route('/read', methods=['GET'])
def read():
    conn = get_db_connection()
    users = conn.execute('SELECT * FROM users').fetchall()
    conn.close()
    return jsonify([dict(row) for row in users])

@app.route('/update/<int:id>', methods=['PUT'])
def update(id):
    data = request.json
    name = data['name']
    conn = get_db_connection()
    conn.execute('UPDATE users SET name = ? WHERE id = ?', (name, id))
    conn.commit()
    conn.close()
    return jsonify({"message": "User updated!"})
```

```python
@app.route('/delete/<int:id>', methods=['DELETE'])
def delete(id):
    conn = get_db_connection()
    conn.execute('DELETE FROM users WHERE id = ?', (id,))
    conn.commit()
    conn.close()
    return jsonify({"message": "User deleted!"})


if __name__ == '__main__':
    conn = get_db_connection()
    conn.execute('CREATE TABLE IF NOT EXISTS users (id INTEGER PRIMARY KEY AUTOINCREMENT, name TEXT)')
    conn.close()
    app.run(host='0.0.0.0', port=5000)
```

Docker-compose file :

```yaml
version: '3.8'


services:
  web:
    build:
      context: .
      dockerfile: Dockerfile
    ports:
      - "5000:5000"
    networks:
      - flask-network
    depends_on:
      - db

  db:
    image: nouchka/sqlite3
    volumes:
      - db_data:/data
    networks:
      - flask-network


networks:
  flask-network:
    driver: bridge


volumes:
  db_data:
```

Docker file :

```
FROM python:3.12-slim

# Set the working directory in the container
WORKDIR /app

# Copy the current directory contents into the container at /app
COPY . .

# Install any needed packages specified in requirements.txt
RUN pip install --no-cache-dir -r requirements.txt

# Make port 5000 available to the world outside this container
EXPOSE 5000

# Define environment variable
ENV NAME World

# Run app.py when the container launches
CMD ["python", "app.py"]
```

2. Containerization of Flask App and Database

Objective:
Containerize the web application and the SQLite database using Docker.

- Created a `Dockerfile` to containerize the Flask application.
  Dockerfile Contents:
  - Used Python base image.
  - Installed dependencies from `requirements.txt`.
  - Exposed port 5000.
  - Set the entry point to run `app.py`.

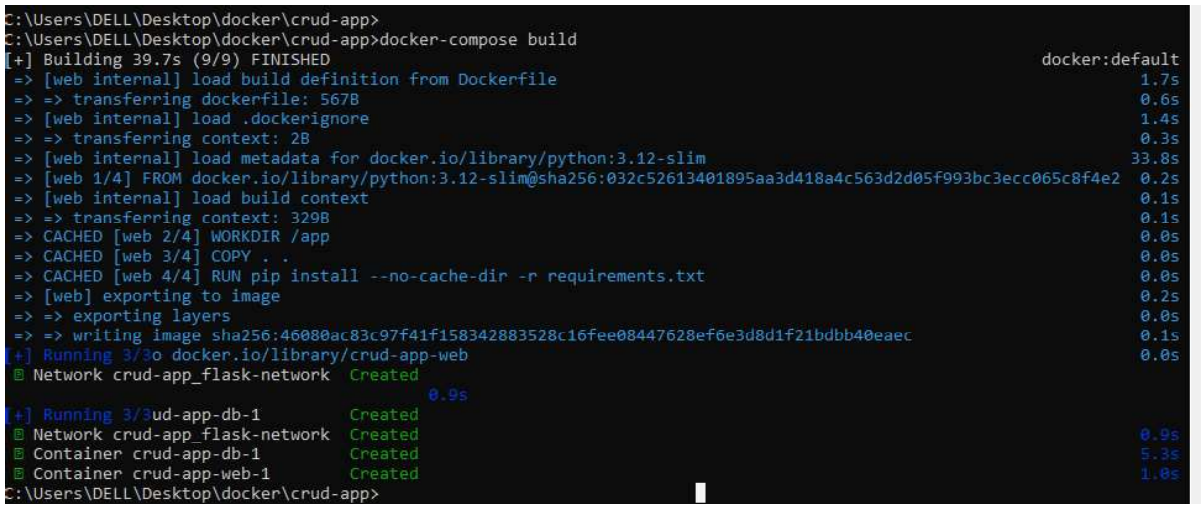
- Created a `docker-compose.yml` file to define and manage multiple containers:
    - **Web Service**: Flask application.
    - **Database Service**: Used SQLite with persistent storage using Docker volumes.

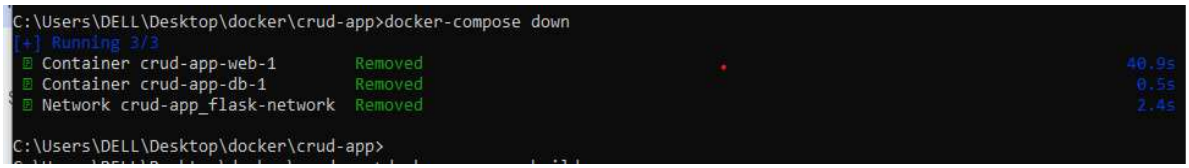
Files Created:

- `Dockerfile`: Defines the Flask application container.
- `docker-compose.yml`: Defines the services and network for the web app and database.

Build and Run Containers:

```
docker-compose build    # Build the containers
docker-compose up -d
```





Adding HTML and CSS Templates:

Create:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Create Item</title>
</head>
<body>
    <h1>Create a new item</h1>
    <form method="POST">
        <label for="name">Name:</label><br>
        <input type="text" name="name" required><br><br>
        <label for="description">Description:</label><br>
        <textarea name="description" required></textarea><br><br>
        <button type="submit">Submit</button>
    </form>
</body>
</html>
```

Index:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CRUD App</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
</head>
<body>
    <h1>Item List</h1>
    <a href="{{ url_for('create') }}">Add New Item</a>
    <ul>
        {% for item in items %}
        <li>
            <strong>{{ item.name }}</strong>: {{ item.description }}
            <a href="{{ url_for('update', id=item.id) }}">Edit</a>
            <a href="{{ url_for('delete', id=item.id) }}" onclick="return confirm('Are you sure?')">Delete</a>
        </li>
        {% endfor %}
    </ul>
</body>
</html>
```

Update:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Update Item</title>
</head>
<body>
    <h1>Update item</h1>
    <form method="POST">
        <label for="name">Name:</label><br>
        <input type="text" name="name" value="{{ item.name }}" required><br><br>
        <label for="description">Description:</label><br>
        <textarea name="description" required>{{ item.description }}</textarea><br><br>
        <button type="submit">Submit</button>
    </form>
</body>
</html>
```

Flask SQLite CRUD Application

Create User

Name:

Create

Update item

Name

Description:

Submit

Delete User

User ID:

10

Delete

Users

Refresh Users List

ID: 6, Name: Vishal