Docker Challenge - Midterm

Student Name: Mourya Shah
Student ID: 000891568

Challenge 1: Simple Web Server for Static Web Pages

Description

This challenge involves creating a simple web server to serve static web pages using Docker and Nginx.

Pre-Requisites

1. A windows computer
2. Visual Studio Code
3. Docker Desktop Installed on your computer

Steps

1. **Set Up Directory Structure**

   Create a directory for your project:

   ```
   mkdir docker-challenge-1
   cd docker-challenge-1
   ```

2. **Create `index.html`**

   In the `docker-challenge-1` directory, create a file named `index.html`:

   ```
   <!DOCTYPE html>
   <html>
   <head>
       <title>Docker Challenge</title>
   </head>
   <body>
       <h1>ID=123456789</h1>
       <h2>NAME=John Doe</h2>
   </body>
   </html>
   ```

3. **Create a Dockerfile**

***Side note:** Make sure computer has docker desktop installed and running*

In the `docker-challenge-1` directory, create a file named `Dockerfile`:

```
FROM nginx:alpine

COPY index.html /usr/share/nginx/html/index.html

EXPOSE 8080

CMD ["nginx", "-g", "daemon off;"]
```

4. **Build the Docker Image**

   Open a terminal and navigate to the `docker-challenge-1` directory. Run the following command to build the Docker image:

   ```
   docker build -t my-static-web-server .
   ```

5. **Run the Docker Container**

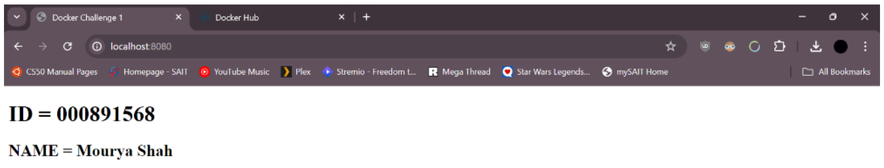
   Run the Docker container using the following command:

   ```
   docker run -d -p 8080:80 my-static-web-server
   ```

6. **Verify the Application**

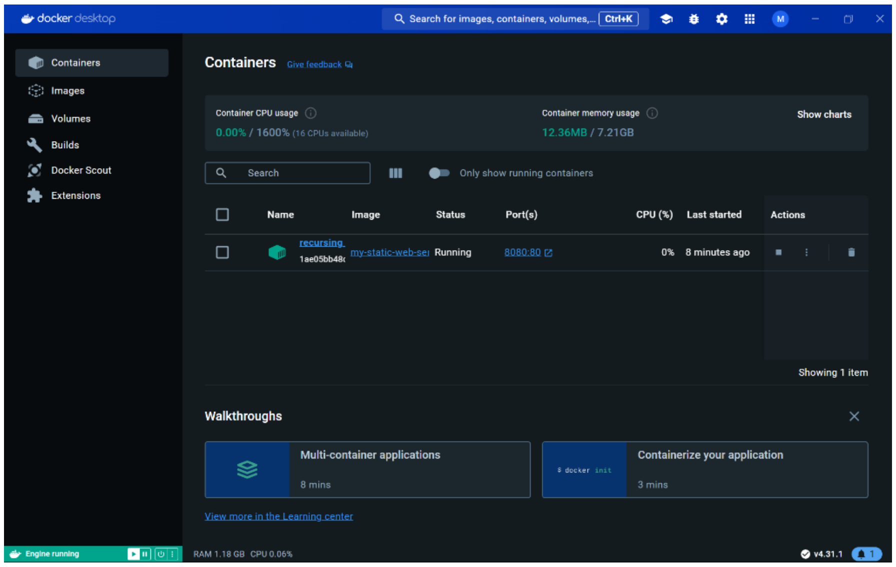
   Open your browser and go to `http://localhost:8080/`. You should see a web page displaying your name and student ID.

   Screenshot of the running application:



ID = 000891568

NAME = Mourya Shah


docker desktop
Search for images, containers, volumes,... Ctrl+K
Containers
Images
Volumes
Builds
Docker Scout
Extensions
Containers
Give feedback
Show charts
Search
Only show running containers
Walkthroughs
Multi-container applications
8 mins
Containerize your application
3 mins
View more in the Learning center
Engine running
v4.31.1

Container CPU usage
0.00% / 1600% (16 CPUs available)

Container memory usage
12.36MB / 7.21GB

Name	Image	Status	Port(s)	CPU (%)	Last started	Actions
recursing_ 1ae05bb48c	my-static-web-se	Running	8080:80	0%	8 minutes ago	

Showing 1 item

RAM 1.18 GB CPU 0.06%

Expected Outcome

When you make a request to `http://localhost:8080/`, you will get a home page with your name and ID number.

Challenge 2: Creating a Dynamic Application

Description

This challenge involves creating a dynamic web application using Python (Flask) and Docker, with Nginx as a reverse proxy.

Pre-Requisites

1. A windows computer
2. Visual Studio Code
3. Docker Desktop Installed on your computer

Steps

1. **Set Up Directory Structure**

   Create a directory for your project:

```
mkdir my-dynamic-app
cd my-dynamic-app
```

2. **Set Up an Isolated Python Environment**

   Create a virtual environment:

   ```
   python -m venv venv
   ```

   Activate the virtual environment:

   ```
   venv\Scripts\activate
   ```

3. **Install Flask**

   Install Flask within the virtual environment:

   ```
   pip install Flask
   ```

4. **Create `app.py`**

   In the `my-dynamic-app` directory, create a file named `app.py`:

   ```
   from flask import Flask, jsonify

   app = Flask(__name__)

   # A list of books to be returned by the /api/books endpoint
   books = [
       {'id': 1, 'title': 'Dune', 'author': 'Frank Herbert'},
       {'id': 2, 'title': 'Neuromancer', 'author': 'William Gibson'},
       {'id': 3, 'title': 'Ender\'s Game', 'author': 'Orson Scott Card'}
   ]

   @app.route('/api/books', methods=['GET'])
   def get_books():
       return jsonify(books)

   @app.route('/api/books/<int:book_id>', methods=['GET'])
   def get_book(book_id):
       book = next((book for book in books if book['id'] == book_id),
   None)
       if book:
           return jsonify(book)
       else:
           return jsonify({'error': 'Book not found'}), 404

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

5. **Create a `requirements.txt` File**

In the `my-dynamic-app` directory, create a file named `requirements.txt`:

```
Flask
```

6. **Create a Dockerfile**

   Side note: *Make sure computer has docker desktop installed and running*

   In the `my-dynamic-app` directory, create a file named `Dockerfile`:

```
FROM python:3.9-alpine

WORKDIR /app

COPY requirements.txt .

RUN pip install -r requirements.txt

COPY . .

EXPOSE 5000

CMD ["python", "app.py"]
```

7. **Create a Docker Compose File**

   In the `my-dynamic-app` directory, create a file named `docker-compose.yml`:

```
version: '3'
services:
  web:
    image: nginx:alpine
    ports:
      - "8080:80"
    volumes:
      - ./nginx.conf:/etc/nginx/nginx.conf
  api:
    build: .
    ports:
      - "5000:5000"
```

8. **Create an Nginx Configuration File**

   In the `my-dynamic-app` directory, create a file named `nginx.conf`:

```
events {}
http {
  server {
    listen 80;
    location / {
            proxy_pass http://api:5000;
        }
    }
}
```

9. **Build and Run the Application**

   Build and run the application using Docker Compose:

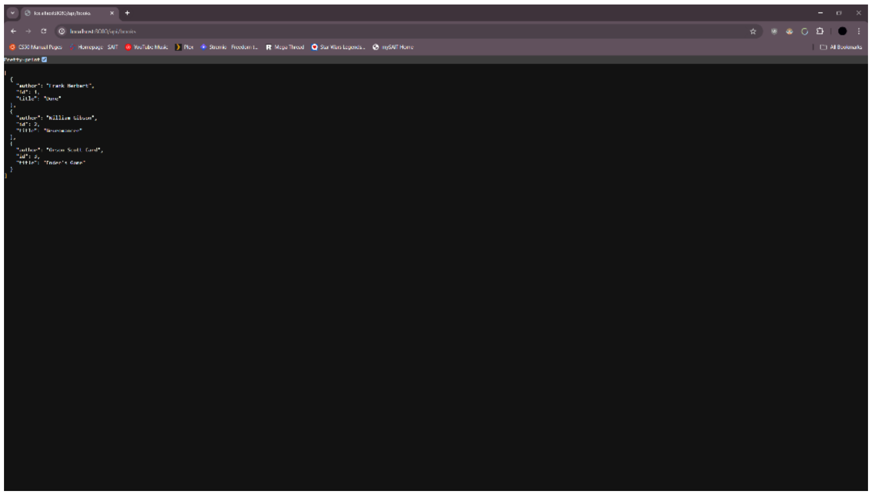
   ```
   docker-compose up --build
   ```

10. **Verify the Application**

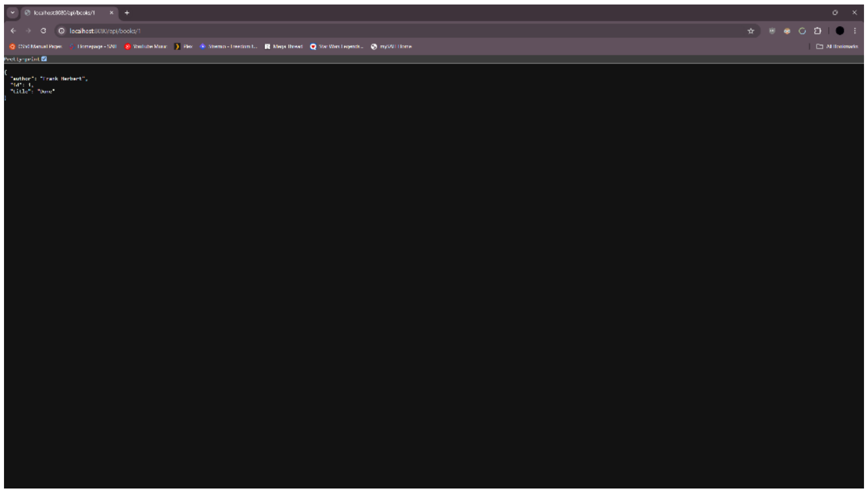
    Open your browser and go to `http://localhost:8080/api/books`. You should see a JSON message with a list of books.

    Open your browser and go to `http://localhost:8080/api/books/1`. You should see a JSON message with the details of the book with ID 1.

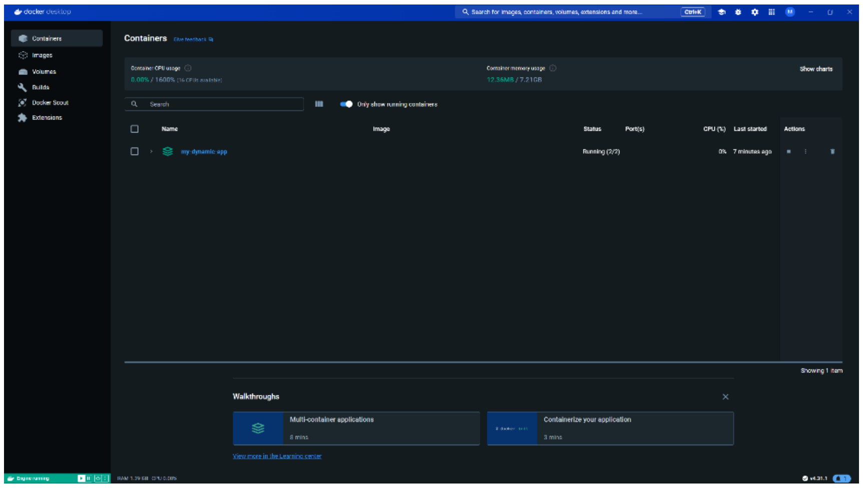
    Screenshots of the running application:



```
[
  {
    "author": "Frank Herbert",
    "id": 1,
    "title": "Dune"
  },
  {
    "author": "William Gibson",
    "id": 2,
    "title": "Neuromancer"
  },
  {
    "author": "Orson Scott Card",
    "id": 3,
    "title": "Ender's Game"
  }
]
```



```
{
  "author": "Frank Herbert",
  "id": 1,
  "title": "Dune"
}
```



Expected Outcome

When you make a request to `http://localhost:8080/api/books`, you will get a JSON message with all books. When you make a request to `http://localhost:8080/api/books/1`, you will get a JSON message with just one book.

References

- Docker Documentation: https://docs.docker.com/
- Flask Documentation: https://flask.palletsprojects.com/
- Nginx Documentation: https://nginx.org/en/docs/
- Official Python Docker Image: https://hub.docker.com/_/python
- Official Nginx Docker Image: https://hub.docker.com/_/nginx
- FreeCodeCamp Docker Handbook: https://www.freecodecamp.org/news/the-docker-handbook/

- YouTube - Docker Tutorial for Beginners: https://www.youtube.com/watch?v=pTFZFxd4hOI