Devops Program 3

Tools required(make sure below tools are installed to your local computer)

- Docker desktop
- Vs code
- Docker account
- Python

PART 1: Dockerfile, Build, Run, and Push to Docker Hub (Python App)

```
docker-app/
├── app1/
  ├──
app.py
  ├── requirements.txt
  └── Dockerfile
```

Create one folder on desktop with name docker-app then open this folder in vscode

Steps →

1. create app1/app.py

   ```
   from flask import Flask
   app = Flask(__name__)

   @app.route('/')
   def hello():
       return "Hello from App 1!"

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

2. Create `requirements.txt`

```
flask==3.0.0
```

3. Create `Dockerfile`

```
FROM python:3.12-slim

WORKDIR /app

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

COPY app.py .

EXPOSE 5000
CMD ["python", "app.py"]



-----------------------------------------------------------


FROM python:3.12-slim

# Set the working directory
WORKDIR /app

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

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

# Copy the application code into the container
COPY app.py .

# Expose the port the app runs on
EXPOSE 5000
```

```
# Command to run the application
CMD ["python", "app.py"]
```

4. login to docker from vs code(connect docker to vscode)

   docker login -u <username>

   type password

5. Build Image & Change into the app1 directory before running the build command:

   cd app1

   docker build -t docker-app-name .

6. Rub Container

   docker run -d -p 5000:5000 python-docker-app:v1.0

   now visit http://localhost:5000 →u will get output

7. Push to docker hub

   a. Tag Image

      docker tag docker-app-name <docker_username>/docker-app-name

   b. Push

      docker push rahulkrchaudhary/docker-app-name

8. stop running docker image

   docker ps // list all running image(container ID)

   docker stop <containerID>

PART 2: Multi-Container App with Docker Compose

```
docker-app/
├── app1/
│   ├──
app.py
│   ├── requirements.txt
```

│   └── Dockerfile
├── app2/
│   ├──
app.py
│   ├── requirements.txt
│   └── Dockerfile
└── docker-compose.yml

step

1. we all ready created app1 folder all classes now we will focus on app2 folder
2. App 2: `app2/app.py`

    ```
    import requests
    response = requests.get("http://app1:5000/")
    print("Response from App 1:", response.text)
    ```

3. app2/requirements.txt

    ```
    requests==2.31.0
    ```

4. app2/Dockerfile

    ```
    FROM python:3.12-slim
    WORKDIR /app
    COPY requirements.txt .
    RUN pip install --no-cache-dir -r requirements.txt
    COPY app.py .
    CMD ["python", "app.py"]
    ```

5. docker-compose.yml

    ```
    version: '3.9'
    services:
      app1:
        build: ./app1
    ```

```
    networks:
      - app-network
    ports:
      - "5000:5000"
  app2:
    build: ./app2
    networks:
      - app-network
    depends_on:
      - app1

networks:
  app-network:
    driver: bridge
```

6. Shut down any existing containers (clean start)

   docker compose down

7. Build and start containers fresh

   docker compose up --build

   optional - docker compose logs app2

   This will:

   - Build images for `app1` and `app2`
   - Start both containers
   - Map `app1`'s port 5000 → your local port 5000

Open in Browser

Go to: http://localhost:5000

You should see: output

You can also check the terminal log to see: Response from App 1: Hello from App 1!