DOCKERIZED WEB APPLICATION

OVERVIEW

This project demonstrates how to containerize a Flask-based web application using Docker. The objective is to create a lightweight, portable, and easily deployable web application. The project covers key DevOps practices such as writing a Dockerfile, building an image, pushing it to Docker Hub, and managing containers.

TECHNOLOGIES USED

- Python 3.13 (Flask framework)
- Docker (for containerization)
- Docker Hub (for image hosting)
- Git Bash / CMD / PowerShell (for command-line operations)

WORKFLOW

1. Develop a simple Flask web application.
2. Create a Dockerfile to containerize the application.
3. Build a Docker image and run a container locally.
4. Push the image to Docker Hub.
5. Pull and manage the containerized application.

PREREQUISITES

Ensure you have the following installed on your system:

- Docker
- Python 3.13
- Git Bash / CMD / PowerShell
- A Docker Hub account

SETTING UP THE PROJECT

1. CREATE A FLASK APPLICATION

Create a app.py file with the following content:

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

@app.route('/')
def home():
    return "Hello, Dockerized Flask App!"

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

2. CREATE A REQUIREMENTS FILE

Create a requirements.txt file to specify dependencies:

```
Flask
```

3. WRITE THE DOCKERFILE

Create a Dockerfile in the project root:

```
# Use an official Python image as base
FROM python:3.13-slim

# Set the working directory
WORKDIR /app

# Copy files to the container
COPY . .

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

# Expose port 5000
EXPOSE 5000

# Run the Flask application
CMD ["python", "app.py"]
```

BUILDING AND RUNNING THE DOCKER CONTAINER

1. Build the Docker Image

```
docker build -t flask-docker-app .
```

2. Run the Container

```
docker run -d -p 5000:5000 flask-docker-app
```

3. Verify the Application

OPEN A BROWSER AND GO TO:

```
http://localhost:5000
```

You should see "Hello, Dockerized Flask App!"

PUSHING THE IMAGE TO DOCKER HUB

1. Log in to Docker Hub

```
docker login
```

(If you signed up using Google, generate an access token from Docker Hub settings and use it instead of a password.)

2. Tag the Image

```
docker tag flask-docker-app <your-dockerhub-username>/flask-docker-app
```

3. Push the Image

```
docker push <your-dockerhub-username>/flask-docker-app
```

MANAGING CONTAINERS

List Running Containers

```
docker ps
```

Stop a Container

```
docker stop <container_id>
```

Remove a Container

```
docker rm <container_id>
```

Run a Container from Docker Hub

```
docker run -d -p 5000:5000 <your-dockerhub-username>/flask-docker-app
```

CONCLUSION

This project successfully containerized a simple Flask application and demonstrated Docker workflows, including image creation, container management, and pushing to Docker Hub. This setup provides a foundation for deploying the application to cloud services like AWS or Kubernetes in the future.