ADVANCE PYTHON PROGRAMING LANGUAGE:

PROJECT: Project Title: Containerizing a Python Web Application using Docker

APP USING DOCKER:

NAME:VIBHU AADHAV.M

REGNO:22MID0141

Objective

The objective of this project is to demonstrate how **Docker** can be used to **containerize** a Python application, ensuring consistent performance across different environments.

🧱 Technologies Used

- **Python 3.10**
- **Flask (Web Framework)**
- **Docker**

Project Structure

```
docker-python-project/
|
├── app.py
├── requirements.txt
└── Dockerfile
```

CODE FOR DOCKER:

```
from flask import Flask, jsonify, request


app = Flask(__name__)


@app.route('/')
def home():
    return "<h2>Welcome to My Dockerized Python App 🚀 </h2>"


@app.route('/info')
def info():
    return jsonify({
        "project": "Dockerized Flask App",
        "developer": "Vibhu",
        "status": "Running Successfully inside Docker Container"
    })


@app.route('/greet', methods=['POST'])
def greet():
    data = request.get_json()
    name = data.get('name', 'User')
    return jsonify({"message": f"Hello, {name}! Welcome to Docker World 🐳 "})
```

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

DOCKER FILE:

```
# Step 1: Use an official Python image
FROM python:3.10-slim

# Step 2: Set working directory
WORKDIR /app

# Step 3: Copy dependency file and install packages
COPY requirements.txt .
RUN pip install -r requirements.txt

# Step 4: Copy the application code
COPY . .

# Step 5: Expose the port Flask runs on
EXPOSE 5000

# Step 6: Command to run the application
CMD ["python", "app.py"]
```

Steps to Run

1️⃣ Build the Docker image

docker build -t python-docker-app .

2️⃣ Run the Docker container

docker run -p 5000:5000 python-docker-app

3️⃣ Access the App

Open your browser → http://localhost:5000

You will see:

Welcome to My Dockerized Python App 🚀

4️⃣ Try API Endpoints

Get Info:

http://localhost:5000/info

Send POST Request:

curl -X POST -H "Content-Type: application/json" \

-d '{"name": "Vibhu"}' http://localhost:5000/greet

Response:

{

"message": "Hello, Vibhu! Welcome to Docker World 🐬 "

}

📊 Output Screenshot (when running)

* Running on http://0.0.0.0:5000

* Press CTRL+C to quit

Browser Output:

“Welcome to My Dockerized Python App 🚀”

Explanation

- Docker provides a **lightweight container environment** that packages code, dependencies, and configurations together.
- This ensures the same behavior across **development, testing, and production**.
- The Flask app serves as an example of a **microservice** that can easily be deployed anywhere Docker is supported.

How Docker Works (Conceptual Overview)

Docker Component	Description
Dockerfile	Set of instructions that define how to build a Docker image.
Docker Image	Blueprint of the application (code + dependencies + OS libraries).
Docker Container	A running instance of an image (isolated environment).
Docker Hub	Public registry to store and share images.
Docker Compose	Tool to run multiple containers (for advanced setups).

Project Workflow Diagram

Source Code (Flask App)

↓

Dockerfile created

↓

docker build → Image

↓

docker run → Container

↓

App runs on localhost:5000

Advantages of Using Docker

1. **Consistency Across Environments:** Eliminates “works on my machine” issues.
2. **Portability:** Containers can run on any OS or cloud platform.
3. **Scalability:** Easy to deploy multiple containers for load balancing.
4. **Isolation:** Each container runs independently.
5. **Efficiency:** Lightweight compared to full virtual machines.

⚙️ Possible Extensions

You can make the project more advanced by:

- Adding a **database container** (e.g., MySQL/PostgreSQL).
- Using **Docker Compose** to run multiple containers together.
- Integrating a **machine learning model** served through Flask API.
- Deploying the Docker container to **AWS EC2** or **Azure Container Instances**.

📜 Conclusion

This project successfully demonstrates:

- How to containerize a Python web app using Docker.
- How containers provide portability, reliability, and scalability.
- How modern deployment pipelines use Docker to simplify DevOps workflows.

Thus, Docker acts as a bridge between **development and production**, ensuring smooth and consistent application performance everywhere.

References

- Docker Official Documentation
- Flask Framework Documentation
- Docker Hub
- Python.org

NAME:VIBHU AADHAV.M

REGNO:22MID0141