**DOCKERS PROJECTS WITH STEP-BY-STEP SOLUTION**

Here are **three hands-on Docker projects** with step-by-step solutions:

**Project 1: Deploy a Python Flask App using Docker**

**Goal: Containerize a simple Flask web application and run it using Docker.**

**Prerequisites: Install Docker and Python**

**Step 1: Create a Flask Application**

Create a project directory and inside it, create a file named app.py:

python

CopyEdit

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)

**Step 2: Create a Requirements File**

Create a requirements.txt file in the same directory:

nginx

CopyEdit

flask

**Step 3: Write a Dockerfile**

Create a Dockerfile in the project directory:

dockerfile

CopyEdit

# Use an official Python runtime as base image

FROM python:3.9

# Set the working directory

WORKDIR /app

# Copy files to the container

COPY requirements.txt requirements.txt

COPY app.py app.py

# Install dependencies

RUN pip install -r requirements.txt

# Expose the application port

EXPOSE 5000

# Start the application

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

**Step 4: Build and Run the Docker Container**

Run the following commands in your terminal:

1. **Build the Docker image**:

bash

CopyEdit

docker build -t flask-app .

1. **Run the container**:

bash

CopyEdit

docker run -p 5000:5000 flask-app

1. Open your browser and visit http://localhost:5000. You should see:

CopyEdit

Hello, Dockerized Flask App!

**Project 2: Deploy an Nginx Web Server with Docker**

**Goal: Run an Nginx server using Docker.**

**Prerequisites: Install Docker**

**Step 1: Pull the Nginx Image**

Run the following command to pull the official Nginx image:

bash

CopyEdit

docker pull nginx

**Step 2: Run an Nginx Container**

Start a container and map port 80 to access it from your browser:

bash

CopyEdit

docker run -d -p 8080:80 --name my-nginx nginx

Now, open http://localhost:8080 in your browser, and you should see the Nginx welcome page.

**Step 3: Customize Nginx with Your Own HTML Page**

1. Create a directory for your Nginx files:

bash

CopyEdit

mkdir nginx\_project && cd nginx\_project

1. Create an index.html file inside this directory:

html

CopyEdit

<!DOCTYPE html>

<html>

<head>

<title>My Dockerized Nginx</title>

</head>

<body>

<h1>Hello, this is a custom Nginx page inside Docker!</h1>

</body>

</html>

1. Run a new Nginx container with your custom HTML page:

bash

CopyEdit

docker run -d -p 8080:80 --name custom-nginx -v $(pwd)/index.html:/usr/share/nginx/html/index.html nginx

1. Refresh http://localhost:8080, and you should see your custom page.

**Project 3: Set Up a Multi-Container Application with Docker Compose**

**Goal: Deploy a Flask app with a PostgreSQL database using Docker Compose.**

**Prerequisites: Install Docker & Docker Compose**

**Step 1: Create a Flask App with PostgreSQL Integration**

Create a directory flask-postgres-app, and inside it, create app.py:

python

CopyEdit

from flask import Flask

import psycopg2

app = Flask(\_\_name\_\_)

def connect\_db():

try:

conn = psycopg2.connect(

dbname="mydb",

user="myuser",

password="mypassword",

host="db"

)

return conn

except Exception as e:

return str(e)

@app.route('/')

def home():

return "Flask App with PostgreSQL!"

@app.route('/db')

def test\_db():

conn = connect\_db()

if isinstance(conn, str):

return f"Database connection failed: {conn}"

return "Connected to the database!"

if \_\_name\_\_ == "\_\_main\_\_":

app.run(host='0.0.0.0', port=5000)

**Step 2: Create a Requirements File**

Create requirements.txt:

nginx

CopyEdit

flask

psycopg2

**Step 3: Write the Dockerfile**

Create a Dockerfile:

dockerfile

CopyEdit

FROM python:3.9

WORKDIR /app

COPY requirements.txt requirements.txt

RUN pip install -r requirements.txt

COPY . .

EXPOSE 5000

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

**Step 4: Write a Docker Compose File**

Create docker-compose.yml:

yaml

CopyEdit

version: '3.8'

services:

web:

build: .

ports:

- "5000:5000"

depends\_on:

- db

db:

image: postgres

environment:

POSTGRES\_USER: myuser

POSTGRES\_PASSWORD: mypassword

POSTGRES\_DB: mydb

ports:

- "5432:5432"

**Step 5: Run the Multi-Container App**

Run the following command:

bash

CopyEdit

docker-compose up -d

Now, open http://localhost:5000/db, and it should confirm the PostgreSQL connection.

**Final Thoughts**

These projects give you hands-on experience with **Docker, Nginx, Flask, PostgreSQL, and Docker Compose**. Let me know if you need help with any step!