Dockerized Python & PostgreSQL Project Complete Documentation

This document provides a complete, step-by-step guide to installing Docker, setting up a Python web application, running PostgreSQL using Docker Compose, and connecting the application to the database. It is intended for beginners learning Docker and DevOps fundamentals.

1. Prerequisites

• Ubuntu 20.04 or 22.04 (Local machine or EC2 instance)
• sudo privileges
• Internet connectivity

2. Installing Docker

Update the system:
sudo apt update

Install required packages:
sudo apt install -y ca-certificates curl gnupg lsb-release

Add Docker GPG key:
sudo mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

Add Docker repository:
echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list

Install Docker Engine and Docker Compose plugin:
sudo apt update
sudo apt install -y docker-ce docker-ce-cli containerd.io docker-compose-plugin

3. Docker Permissions

Add current user to Docker group to avoid using sudo:
sudo usermod -aG docker ubuntu
newgrp docker

4. Project Structure

project_2/
■■■ Dockerfile
■■■ app.py
■■■ docker-compose.yml
■■■ requirements.txt

5. Dockerfile Explanation

The Dockerfile defines how the Python web application image is built. It installs dependencies and runs the application inside a container.

Example Dockerfile:

```
FROM python:3.10-slim
WORKDIR /app
COPY requirements.txt .
RUN pip install -r requirements.txt
COPY app.py .
CMD ["python", "app.py"]
```

6. requirements.txt

This file contains all Python dependencies required by the application.

```
flask
psycopg2-binary
```

7. Application Code (app.py)

The application uses Flask and connects to PostgreSQL using psycopg2. The database hostname is the Docker Compose service name (db).

```
psycopg2.connect(
  host="db",
  database="mydb",
  user="user",
  password="password"
)
```

8. Docker Compose Configuration

Docker Compose manages multiple containers and networking. It starts both the web application and PostgreSQL database.

```
services:
  web:
    build: .
    ports:
    - "5000:5000"
    depends_on:
    - db

  db:
    image: postgres
    environment:
    POSTGRES_USER: user
    POSTGRES_PASSWORD: password
    POSTGRES_DB: mydb
```

9. Running the Project

Run the following command from the project directory:

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

Access the web application using:
http://<EC2-PUBLIC-IP>:5000

10. Database Connection Methods

Recommended way to connect to PostgreSQL (inside container):
docker compose exec db psql -U user -d mydb

Optional host connection (requires port 5432 exposure):
psql -h localhost -p 5432 -U user -d mydb

11. Ports Summary

• Flask Web App → Port 5000
• PostgreSQL Database → Port 5432

12. Common Errors and Fixes

• Connection refused → Docker not running
• Cannot find requirements.txt → File missing
• localhost DB connection fails → Use 'db' hostname
• Wrong port → PostgreSQL uses 5432

13. Conclusion

This project demonstrates Docker fundamentals, container networking, multi-service applications, and database integration using Docker Compose. It is suitable for DevOps beginners and cloud deployment practice.