Docker Mini Project: Flask + PostgreSQL To-Do App

Project Overview:
This mini project demonstrates how to containerize a Python Flask application that connects to a PostgreSQL database using Docker and Docker Compose. It helps understand how multiple containers (backend and database) communicate in a networked environment.

Goal: Build and deploy a simple To-Do App using Flask + PostgreSQL with Docker Compose.

Folder Structure:
docker-todoapp/
app.py
requirements.txt
Dockerfile
docker-compose.yml
.gitignore (optional)

Technology	Purpose
Python 3.11 (Flask)	Backend REST API
PostgreSQL 15	Database for storing tasks
Docker	Containerization of application and DB
Docker Compose	Multi-container orchestration

Step-by-Step Implementation:

1 Create a project folder: mkdir docker-todoapp && cd docker-todoapp
2 Create the required files: app.py, requirements.txt, Dockerfile, docker-compose.yml
3 Write Flask API (CRUD for tasks) connecting to PostgreSQL via environment variables.
4 Build and start containers: docker compose up --build
5 Initialize the database: curl http://localhost:5000/init-db
6 Test the APIs:
• GET /tasks → List tasks
• POST /tasks → Add a new task
• POST /tasks/<id>/toggle → Mark task done/undone

7 Stop and clean up: docker compose down -v

Key Docker Concepts:

• **Dockerfile** — Instructions to build a Docker image.
• **Image** — Blueprint to create containers.
• **Container** — Running instance of an image.
• **Docker Compose** — Tool to manage multiple containers together.
• **Volumes** — Store persistent data (PostgreSQL data survives restarts).
• **Ports** — Maps internal container ports to the host (5000:5000).

Expected Output:

After running docker compose up --build, logs show Flask and PostgreSQL containers running.
Test API with: curl http://localhost:5000/init-db → “DB
initialized” curl http://localhost:5000/tasks → Returns task
list in JSON.

Screenshot Reference:
Docker Compose building images and creating containers successfully on AWS EC2.

Conclusion:
This mini project demonstrates the complete workflow of Docker containerization — from creating Dockerfiles to orchestrating services with Docker Compose. It’s a solid foundation for understanding how real-world applications are deployed in isolated, reproducible environments.