# Placement Empowerment Program

***Cloud Computing and DevOps Centre***

Build and Deploy a Multi-Tier Application Using Docker Compose: Use Docker Compose to build a web application and connect it to a database.

Name: Yasvanth Rajan E Department:AML



# Introduction

In modern application development, **containerization** plays a crucial role in simplifying deployment and scaling. **Docker Compose** allows developers to define and manage multi-container applications using a simple YAML file. This Proof of Concept (PoC) demonstrates how to build and deploy a **multi-tier application** using **Docker Compose**, connecting a **Flask web application** to a **PostgreSQL database** in a containerized environment.

# Overview

This PoC involves:

1. **A Flask Web Application** – A Python-based web app that interacts with a database.
2. **A PostgreSQL Database** – A relational database running as a separate service in a Docker container.
3. **Docker & Docker Compose** – The web and database services are containerized and orchestrated using docker-compose.yml.

Using **Docker Compose**, we can build, run, and manage both services seamlessly, ensuring smooth communication between the application and the database.

* 1. **Containerize a web application and a database** using Docker.
  2. **Use Docker Compose** to define multi-container applications.
  3. **Deploy and test the application** locally with containerized services.
  4. **Ensure communication** between the web app and database inside Docker containers.
  5. **Simplify deployment** by automating multi-container setups with a single command.

# Importance

* + 1. **Real-world Use Case**: Multi-tier applications are commonly used in production environments (e.g., web apps with databases).
    2. **Portability**: With containerization, the app runs identically on any machine with Docker.
    3. **Scalability**: Services can be easily scaled by modifying

docker-compose.yml.

* + 1. **Efficient Deployment**: Eliminates dependency issues by packaging everything in containers.
    2. **DevOps Readiness**: This PoC aligns with DevOps practices by automating environment setup and deployment.

## Step 1:

Check if Docker is installed

Open Command Prompt (CMD) and run:

### docker --version

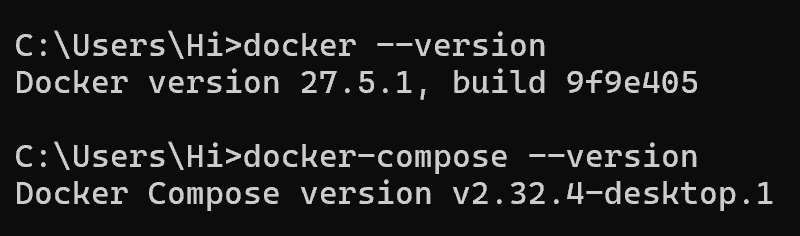
Now, check Docker Compose:

### docker-compose --version

If you don’t have Docker installed, download and install it from:

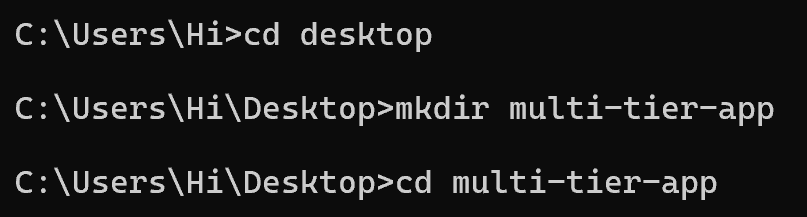
◻ Docker Desktop for Windows

Once installed, restart your computer and ensure Docker Desktop is running.



Navigate to your preferred location (e.g., Desktop) Create a new folder named multi-tier-app:

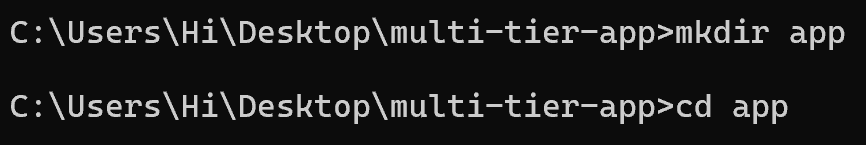
**mkdir multi-tier-app cd multi-tier-app**

****

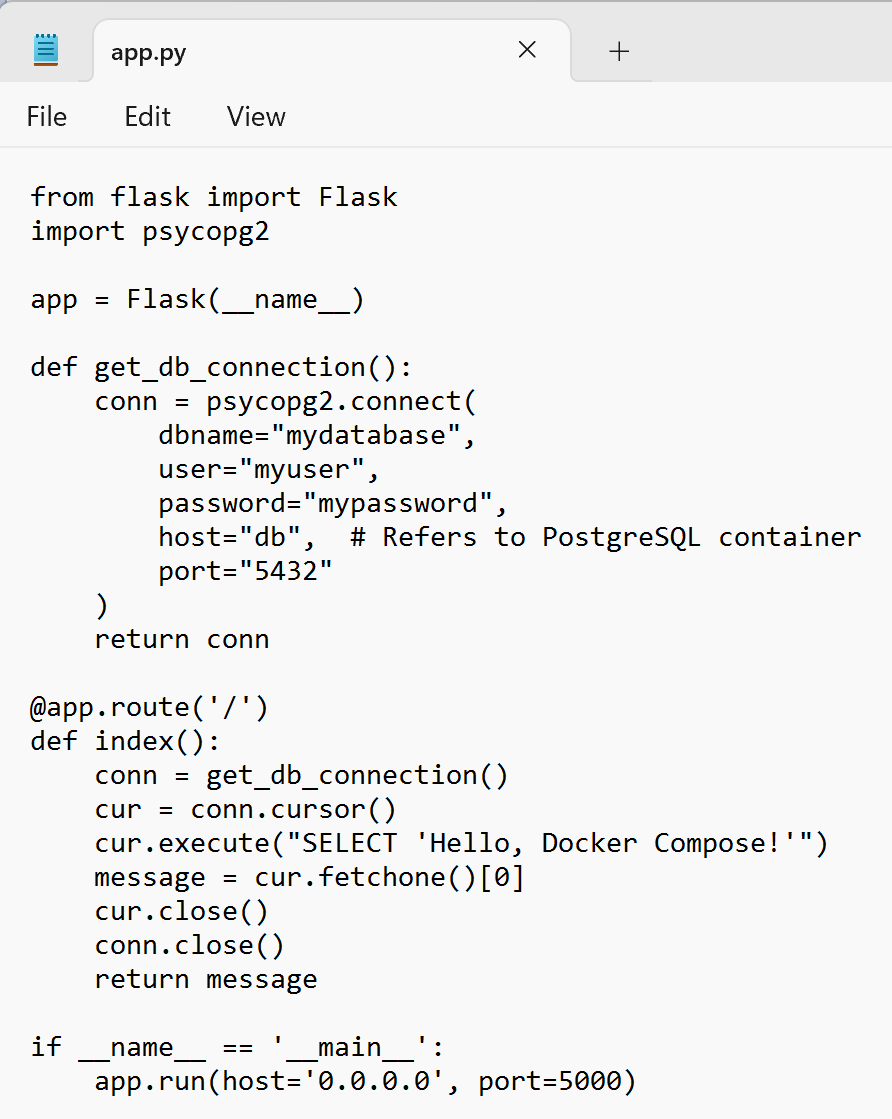
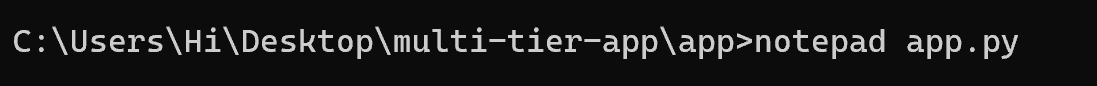
## Step 3:

Inside multi-tier-app, create another folder named app:

### mkdir app cd app

****

Inside the app folder, create a file named **app.py**. **notepad app.py**



We need a list of dependencies for our Flask app.

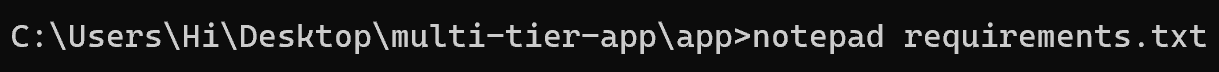
1. Run:

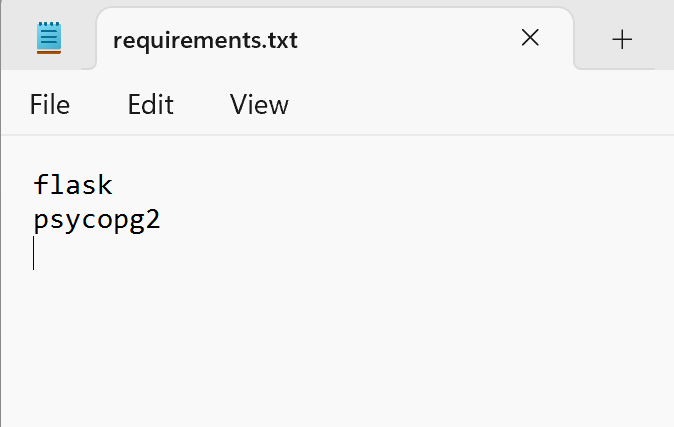
### notepad requirements.txt

1. Paste this:

### flask psycopg2

1. Save and close.



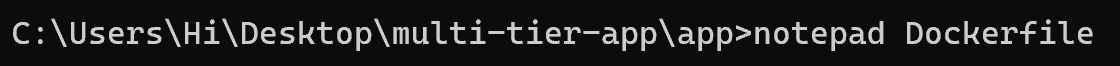


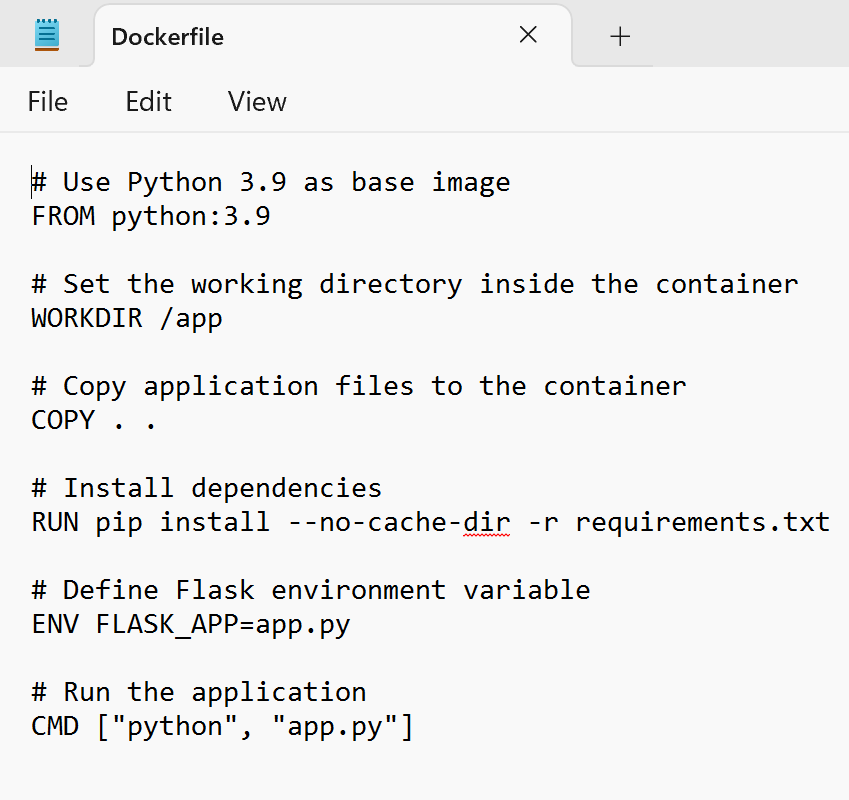
Now, create a Dockerfile inside the app folder.

1. Run:

### notepad Dockerfile

1. Paste this
2. Save and close.





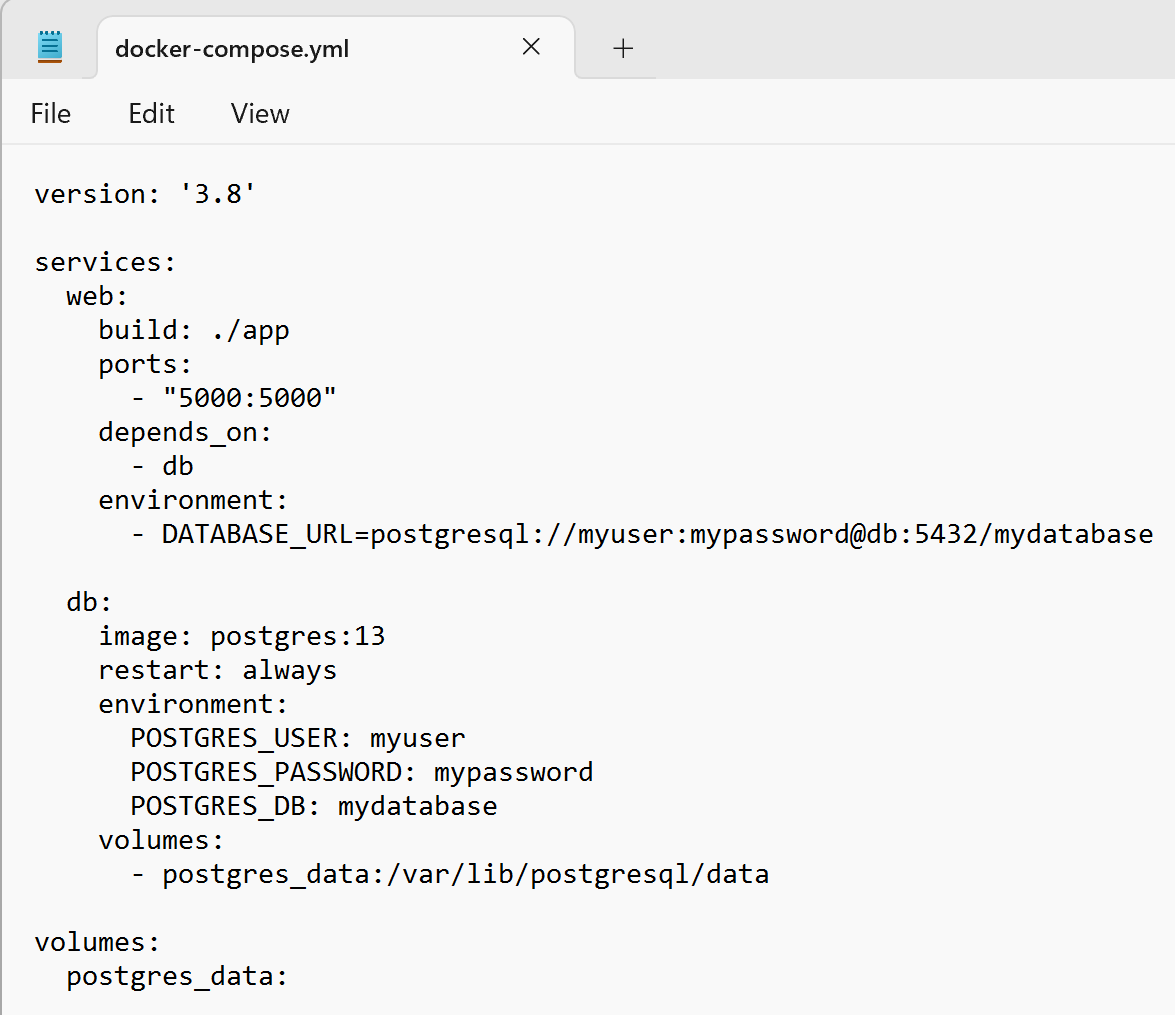
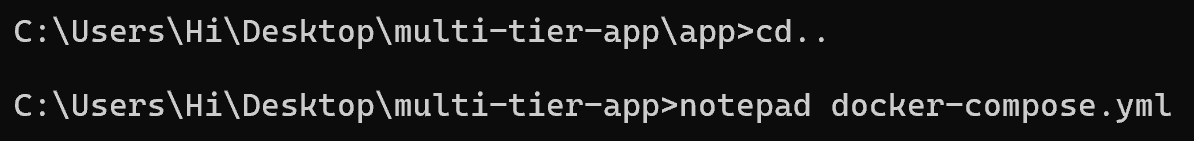
Now, navigate back to multi-tier-app:

### cd ..

1. Run:

### notepad docker-compose.yml

1. Paste this content:
2. Save and close.



Now, **ensure Docker Desktop is running**. Then, from the multi-tier- app folder, run:

### docker-compose up --build

* + **What Happens?**
    - The **Flask app** is built and runs in a container.
    - The **PostgreSQL database** starts as another container.
    - Both services are connected.

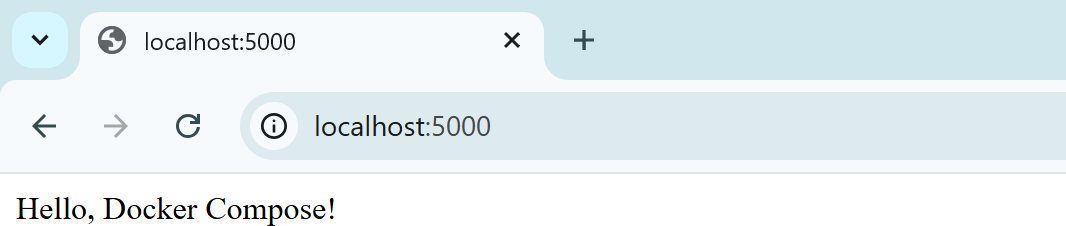
Wait until you see logs indicating that both containers are running.



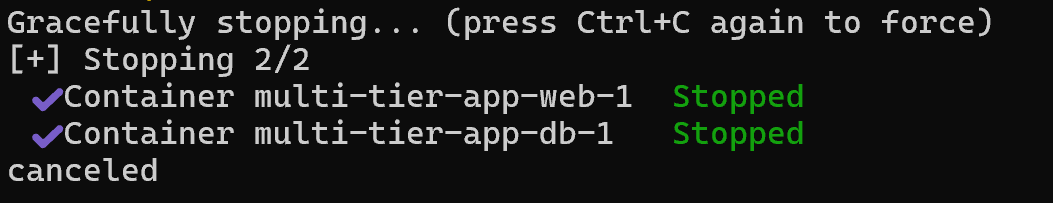
## Step 9:

Once the containers are running, **open a browser** and visit:

[**http://localhost:5000**](http://localhost:5000/)

****

Press Ctrl+C two times to cancel the process.



## Step 11:

To **stop the application**, run:

**docker-compose down**

****

## Summary

1. We **installed** Docker and verified the setup.
2. We **created** a Flask web application.
3. We **defined** a PostgreSQL database.
4. We **built** and **deployed** the application using Docker Compose.
5. We **tested** the setup successfully.

**Outcomes**

By completing this PoC, you will:

* 1. **Understand Multi-Tier Application Architecture** – Gain hands-on experience in designing and deploying a **web application with a separate database layer** using containerization.
  2. **Use Docker Compose for Multi-Container Deployment** – Learn how to define and manage multiple services (**web app and database**) in a docker-compose.yml file and orchestrate them with a single command.
  3. **Deploy and Run a Flask Web Application in Docker** – Build a **Flask-based web application**, containerize it with a Dockerfile, and run it as a service inside a Docker container.
  4. **Set Up and Configure a PostgreSQL Database in a Container** – Deploy a **PostgreSQL database** in a separate container, expose necessary ports, and configure authentication and database settings using environment variables.
  5. **Establish Communication Between Containers** – Ensure seamless interaction between the **Flask web app** and the **PostgreSQL database** by leveraging Docker’s networking capabilities.