# Placement Empowerment Program

***Cloud Computing and DevOps Centre***

Containerize a Multi-Tier Application: Create Docker containers for your web application and database.

Configure them to communicate using a Docker network.

Name: Sharon Hanna A Department:Ads



# Introduction

In modern software development, applications often follow a **multi- tier architecture**, where different components such as the web application and database run separately to enhance scalability and maintainability. **Containerization** with Docker enables efficient deployment and management of such applications by packaging them with all their dependencies.

This PoC demonstrates how to **containerize a multi-tier application** using **Flask (web application) and MySQL (database)** in Docker on Windows. The goal is to ensure both containers communicate within a Docker network.

# Overview

A **multi-tier application** consists of multiple layers:

1. **Web Application (Flask)** – Handles user interactions and sends queries to the database.
2. **Database (MySQL)** – Stores and manages data.
3. **Docker Network** – Enables communication between containers.

**Key steps in this PoC:**

1. Create a **Docker network** for communication.
2. Build and run a **MySQL database container** with environment variables.
3. Build and run a **Flask web application container** that connects to MySQL.
4. Test communication between the containers.

# Objectives

✔Learn to containerize a web application and database separately.

✔Configure a **Docker network** to enable container communication.

✔ Use **environment variables** to manage database credentials securely.

✔Deploy and test a working **multi-tier application** using Docker.

# Importance

1. **Isolation:** Keeps the web app and database separate for better scalability.
2. **Portability:** Containers can run anywhere, making deployment easy.
3. **Efficiency:** Avoids conflicts between dependencies, ensuring a smooth development workflow.
4. **Scalability:** Supports future extensions like load balancing or additional services.

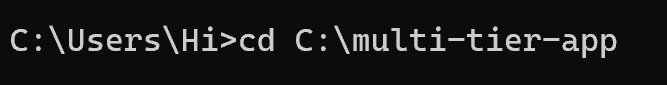
# Step-by-Step Overview

## Step 1:

**Create a Project Folder**

Open **Command Prompt** , then run:

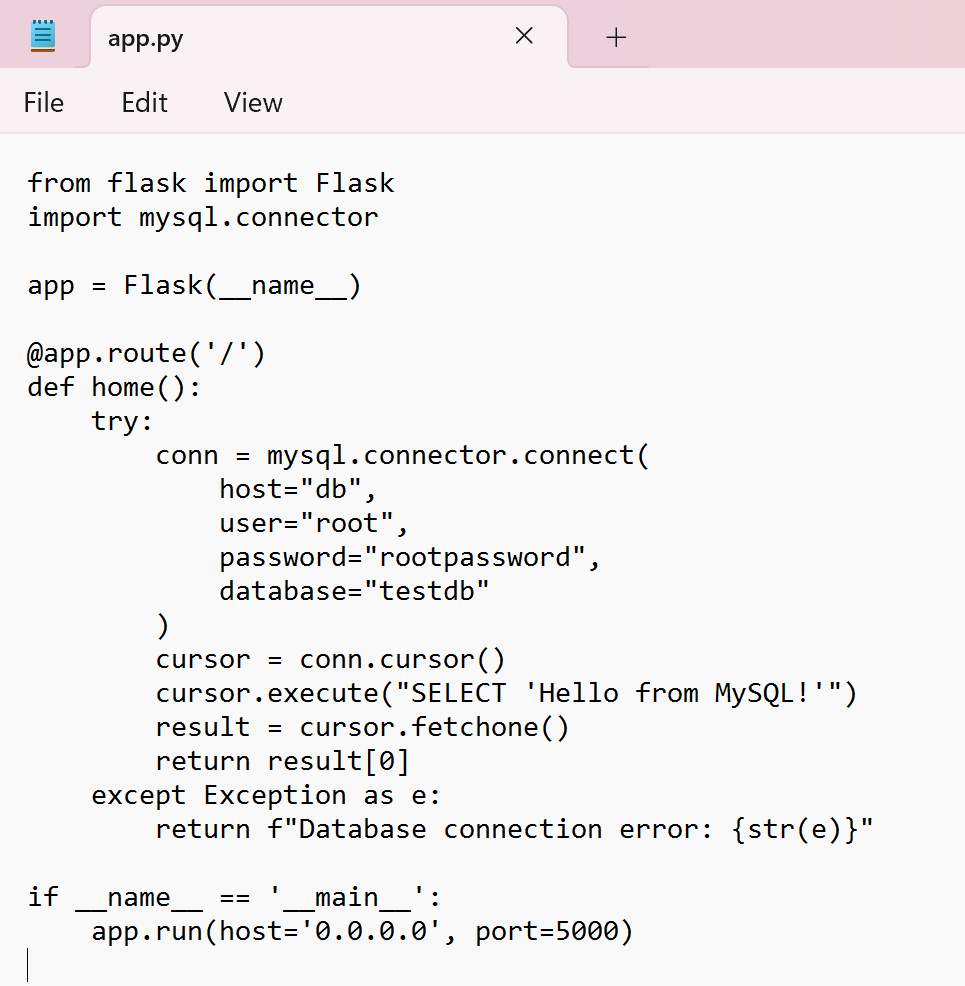
**mkdir C:\multi-tier-app cd C:\multi-tier-app**

****

## Step 2:

Create the Flask Web Application

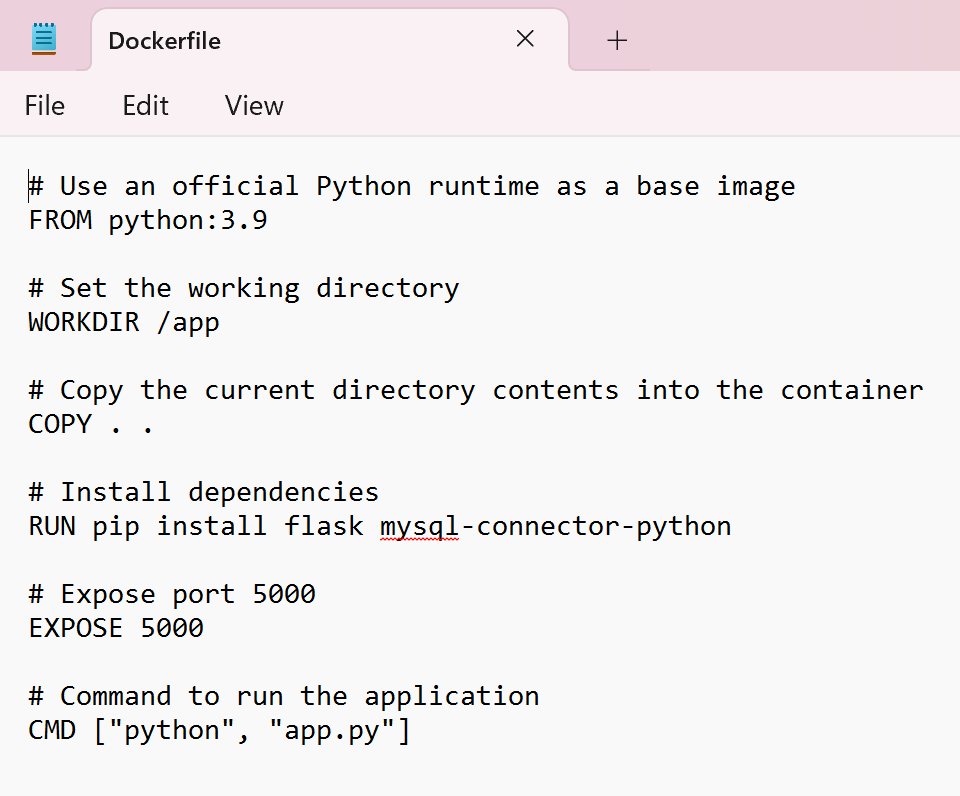
Inside C:\multi-tier-app, create a new Python file: Create **app.py**



## Step 3:

**Create a Dockerfile**

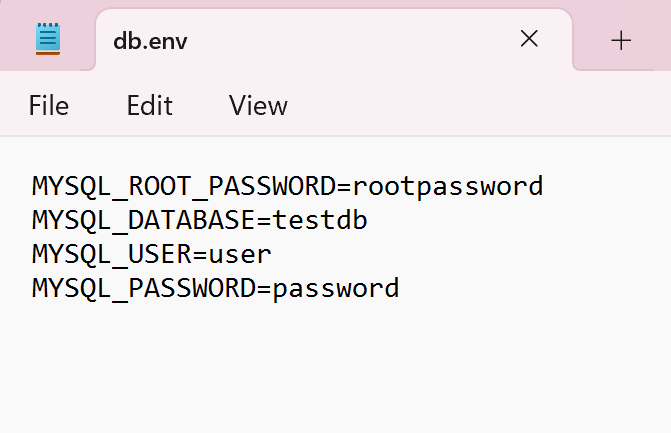
In the **same folder**, create a file named Dockerfile (without an extension):



## Step 4:

**Create the MySQL Database Environment File**

Create a new file named db.env in the same folder:



## Step 5:

Create a Docker Network

Open Command Prompt (cmd) and create a network:

**docker network create app-network**

## Step 6:

Run the MySQL Database Container

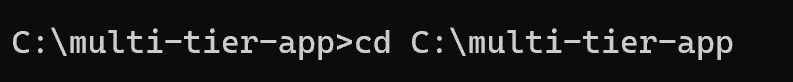
Start the MySQL container using the network:

**docker run -d --name db --network app-network --env-file C:\multi-tier-app\db.env mysql:5.7**

****

## Step 7:

Build and Run the Flask Web App Container Navigate to your project folder:

**cd C:\multi-tier-app**

## Step 8:

Build the Docker image:

**docker build -t web-app .**

## Step 9:

Run the container and connect it to the network:

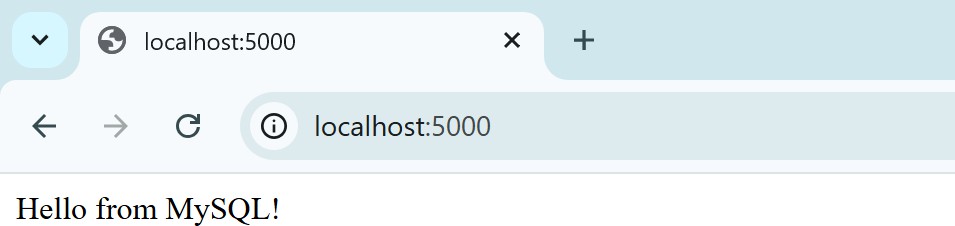
**docker run -d --name web --network app-network -p 5000:5000 web-app**

## Step 10:

Test the Web App . Open a browser and go to:

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

You should see: Hello from MySQL!



# Outcomes

By completing this PoC, you will:

1. **Master Multi-Tier Containerization** – Gain hands-on experience in containerizing both a web application and a database separately, ensuring modular and scalable deployments.
2. **Set Up Docker Networking** – Learn how to create and configure a **Docker network** to enable secure communication between application containers.
3. **Work with Environment Variables** – Understand how to securely manage database credentials and configurations using an **env file** in Docker.
4. **Enhance Docker Command Proficiency** – Improve skills in using essential Docker commands like docker network create, docker run, docker build, and docker exec for efficient container management.
5. **Test and Debug Containerized Applications** – Learn how to verify container communication using tools like docker logs and docker exec for debugging and troubleshooting.