**Containerize and Deploy an Employee Feedback Portal**

**Using Python Flask, PostgreSQL, Git, and Docker**

**Kartiki Gaikwad**

**1. Project Overview**

This project demonstrates the deployment of a **Flask web application** integrated with **PostgreSQL** and containerized using **Docker**.

Employees can enter their **name**, **department**, **rating**, and **feedback**, which are stored in the PostgreSQL database and displayed dynamically on the web interface.

**Project Structure**

Employee\_Feedback\_Portal/

│

├── main.py # Flask application (backend logic)

├── Dockerfile # Container configuration

├── requirements.txt # Python dependencies

├── templates/

│ └── index.html # Frontend template

└── README.md # Project documentation

**Employees can**:

* Submit feedback with name, department, email, category and rating
* Attach files (optional)
* View all previous feedback entries dynamically

**2. Objectives**

* Develop a **Flask-based web application** for feedback submission.
* Store data securely using **PostgreSQL**.
* Implement **Git-based version control** for source code.
* Use **Docker** to containerize both the Flask app and PostgreSQL database.
* Demonstrate **inter-container networking** and **persistent storage**.

**3. Application Architecture**

Flask Web App <---> PostgreSQL Database

(Container 1) (Container 2)

- main.py - Stores feedback

- index.html - Persistent volume

|

| Docker Network

|

Docker Engine / Host Machine

**Flow:**

1. User submits feedback via Flask UI.
2. Flask routes data to PostgreSQL using SQL queries.
3. Data is stored and displayed back dynamically on the same page.

**Implementation Steps**

**Step 1: Flask Application (main.py)**

Handles routes / and /submit, connecting to PostgreSQL using psycopg2.

**Step 2: HTML Template (index.html)**

Simple Bootstrap form for submitting and displaying feedback.

**Step 3: requirements.txt**

flask

psycopg2-binary

**Step 4: Dockerfile**

Builds a Python Flask image:

FROM python:3.9

WORKDIR /app

COPY . /app

RUN pip install -r requirements.txt

EXPOSE 5000

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

**Database Setup**

**Create a database**

Start PostgreSQL (either locally or using Docker) and create a database:

CREATE DATABASE kartikidb;

# **Create the Table**

Run the following SQL commands to create the feedback table:

DROP TABLE IF EXISTS feedback;

CREATE TABLE feedback (

id SERIAL PRIMARY KEY,

name VARCHAR(100) NOT NULL,

email VARCHAR(120) NOT NULL,

department VARCHAR(100) NOT NULL,

category VARCHAR(100) NOT NULL,

rating INTEGER CHECK (rating BETWEEN 1 AND 5) NOT NULL,

feedback TEXT NOT NULL,

attachment VARCHAR(255),

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

**## Python Dependencies**

List all dependencies in requirements.txt:

flask

psycopg2-binary

**## Docker Setup**

1️⃣ **Build the Flask App Image**

docker build -t python-app .

2️⃣ **Run PostgreSQL Container**

docker run -d \

--name postgres-db \

-e POSTGRES\_DB=kartikidb \

-e POSTGRES\_USER=kartiki \

-e POSTGRES\_PASSWORD=kartiki \

postgres:13

3️⃣ **Create Feedback Table in PostgreSQL**

docker exec -it postgres-db psql -U kartiki -d kartikidb

table is as follows:

DROP TABLE IF EXISTS feedback;

CREATE TABLE feedback (

id SERIAL PRIMARY KEY,

name VARCHAR(100) NOT NULL,

email VARCHAR(120) NOT NULL,

department VARCHAR(100) NOT NULL,

category VARCHAR(100) NOT NULL,

rating INTEGER CHECK (rating BETWEEN 1 AND 5) NOT NULL,

feedback TEXT NOT NULL,

attachment VARCHAR(255),

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

Now, enter command **\dt**

It shows a list of all tables in the current database and schema.

you might see something like:

List of relations

Schema | Name | Type | Owner

--------+------------+-------+---------

public | feedback | table | kartiki

(1 row)

4️⃣ **Run Flask Container**

docker container run \

--name python-app \

-p 5000:5000 \

--link postgres-db:db \

-e POSTGRES\_HOST=db \

-e POSTGRES\_DB=kartikidb \

-e POSTGRES\_USER=kartiki \

-e POSTGRES\_PASSWORD=kartiki \

python-app

**## Docker Container Management Commands**

**Stop All Running Containers**

docker stop $(docker ps -a -q)

**Remove All Stopped Containers**

docker rm $(docker ps -a -q)

**Stop a Specific Container**

docker stop python-app

**Remove a Specific Container**

docker rm python-app

**Access the Application**

Once both containers are running, open:

**http://18.16.121.223:5000**

You’ll see the Employee Feedback Portal where you can:

**Submit feedback through form**

**Data stored in PostgreSQL**

**All entries displayed dynamically**

**Git Workflow:**

# Clone repository

git clone (github url)

cd EmployeeFeedback

# Stage and commit changes

git add .

git commit -m "Initial commit - Flask Feedback Portal"

# Push to GitHub

git push origin main

**The screenshots below are attached to illustrate the steps followed during the development of the application.**