**Counter App with Python and Flask**

This repo is contain files which needed to create an app using python, in order to count webpage hits. The application is bundled into a container and deployed on docker using docker compose. The backend database is MySQL.

**API functionality:**

Description: Retrives number of hits for the current deployment.

Request: GET /

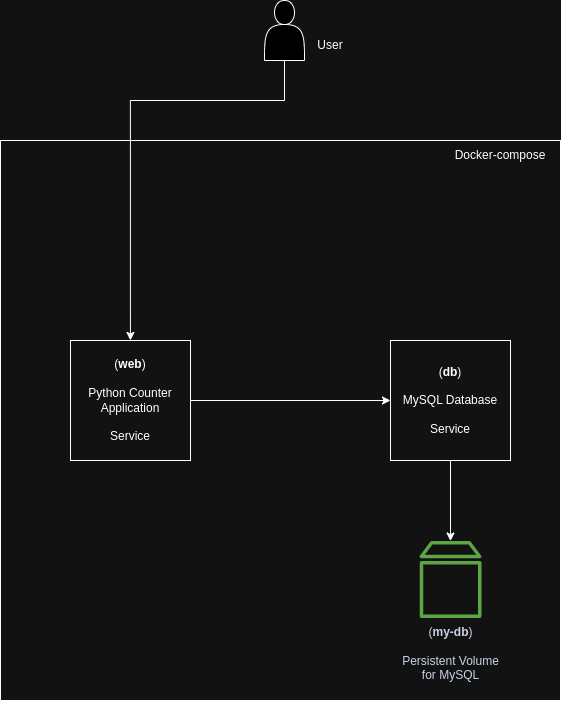
Response: This page has been visited **1** times.

**Details:**

The app is written in Python, using Flask framework

* **app.py** is the actual app code.
* **requirements.txt** are the dependencies required to run the app.
* **Init\_database.py** file is used to establish connection with mysql database and created tables and entries.
* **Dockerfile** is used to build docker container.
* **entrypoint.sh** is used to configure database entries and run application by required commands when container will start.
* **wsgi.py** file is Web Server Gateway Interface (WSGI) is a mediator responsible for conveying communication between a web server and a Python web application.

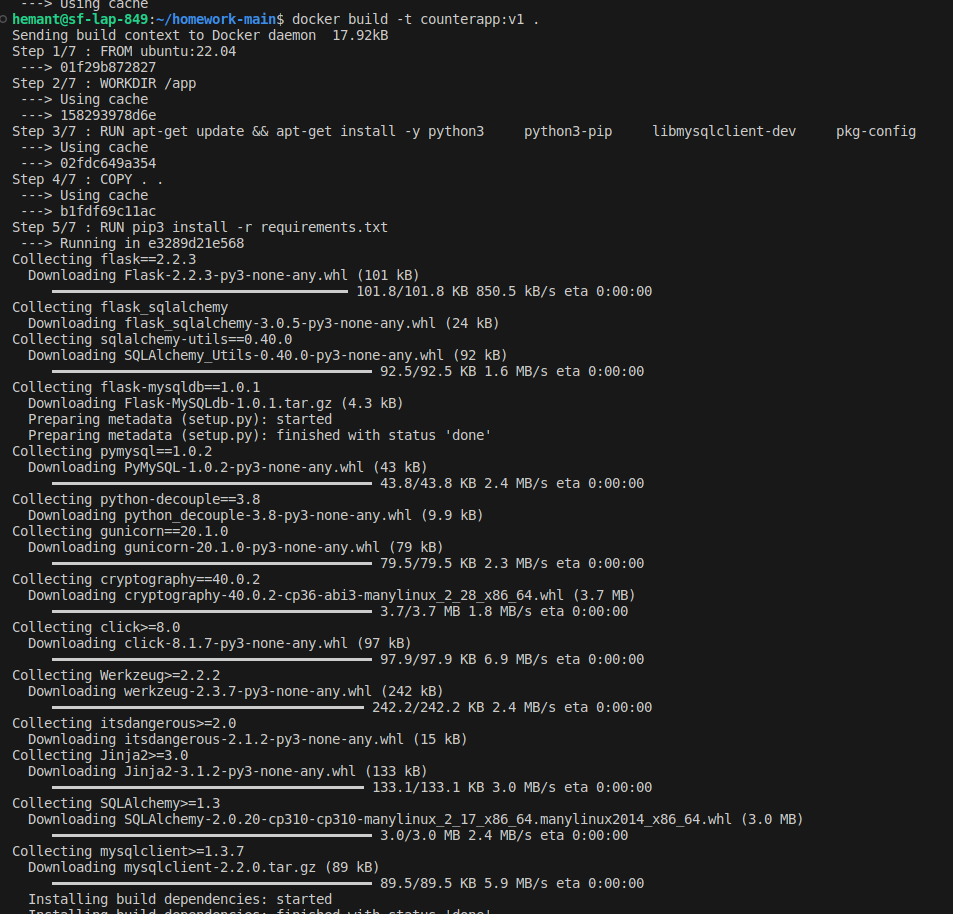
**Architecture diagram:**

****

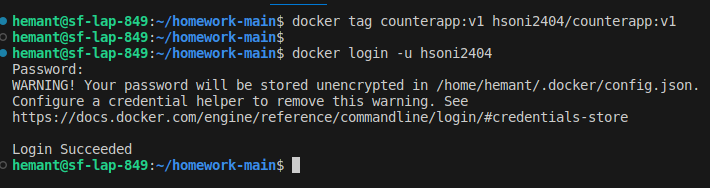
**Building/testing steps:**

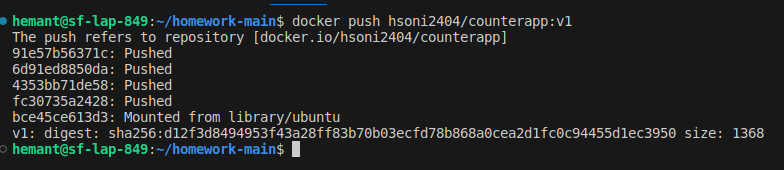
Build and tag docker image:

docker build -t counterapp .



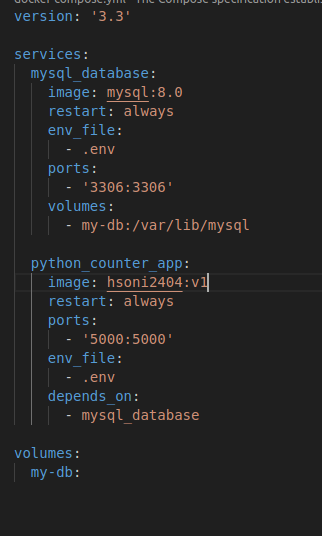
push the image to docker hub:





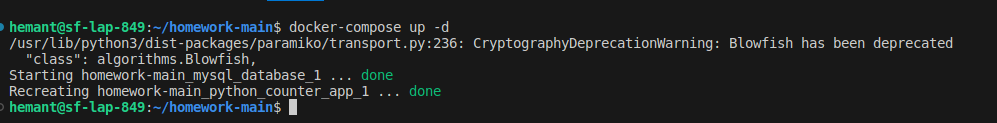
Create a docker-compose file to run this application with it’s dependency on server within same network.

docker-compose.yaml



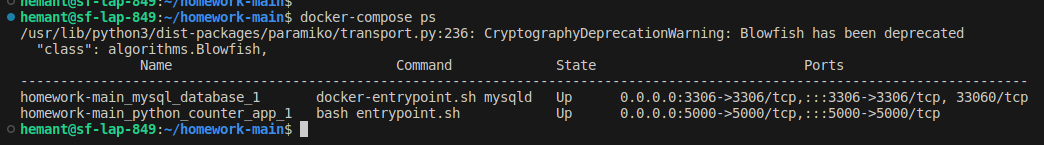
**Deployment steps:**

Run application and its dependency (MySQL) using docker-compose.

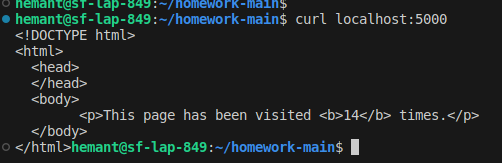


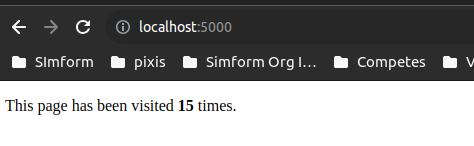
Check Application status my using below command:

Docker-compose ps



Access application using curl command in CLI and also you can use browser to check hits count.





**Deliverables**

I have Choosen Docker to deploy this application instead fo virtual machine. I am sharing with you reason behind it:

* Docker containers start up quickly, often in seconds. VMs typically take longer to boot because they need to load and initialize a full operating system.
* Fast start-up times are essential for auto-scaling and dynamic workloads.
* Docker containers are highly portable across different environments.
* Docker images contain all dependencies and configurations, ensuring consistent behavior regardless of the host environment.
* Docker images are versioned, allowing you to track changes and roll back to previous versions if needed.
* Dockerfiles enable you to define the application's environment and dependencies as code.
* Docker provides a user-friendly CLI and graphical interfaces, making it easier to manage containers, networks, and volumes compared to managing VMs.
* Docker Images are easy to share with others. We can created tarball of images and easily share.

**Improvement suggestions:**

* We can easily monitor containerize applications.
* We can deploy docker containers with multiple container orchestration platform like docker-swarm, AWS ECS, Kubernetes etc.
* We can create multi-stage docker build to reduce image size.
* We can create multiple networks in docker for different environments.
* We can easily manage volume mounting in docker for data persistency.