**Task-1:** Create docker containers for all these components manually using **docker run** command and make the stack working into your **local workstation/laptop/VM**

**Solution:**

**Note:** The Task 1 is done in my local workstation/laptop.

* 1. **Deploying the code base for the python flask application in the local:**

**Installing the dependency module:**

> psycopg2 & redis

A screen shot of a computer program

Description automatically generated

Pull the required PostgreSQL Database:

> docker pull postgres:13 A computer screen with text on it

Description automatically generated

> docker pull redis A black screen with white text

Description automatically generated

> docker images – Docker images for the redis and postgres is created.A screen shot of a computer

Description automatically generated

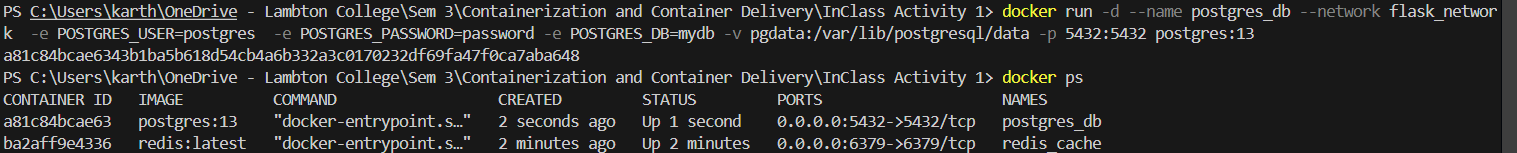
**1.2) Creating a network for the python flask application stack:**

> docker network create flask\_network A screen shot of a computer

Description automatically generated

> docker run -d --name redis --network=flask\_network redis:latest A black screen with white text

Description automatically generated

> docker run -d --name db --network flask\_network -e POSTGRES\_USER=postgres -e POSTGRES\_PASSWORD=password -e POSTGRES\_DB=mydb -v pgdata:/var/lib/postgresql/data -p 5432:5432 postgres:13 

**a) Creating the databae and tables in Postgres:**

> docker exec -it postgresCont bash

**b) Connect to a PostgreSQL Database Server:**

> psql -h localhost -U postgres

**c) Creating table in DB**

> CREATE TABLE users (ID INT PRIMARY KEY NOT NULL, NAME TEXT NOT NULL);

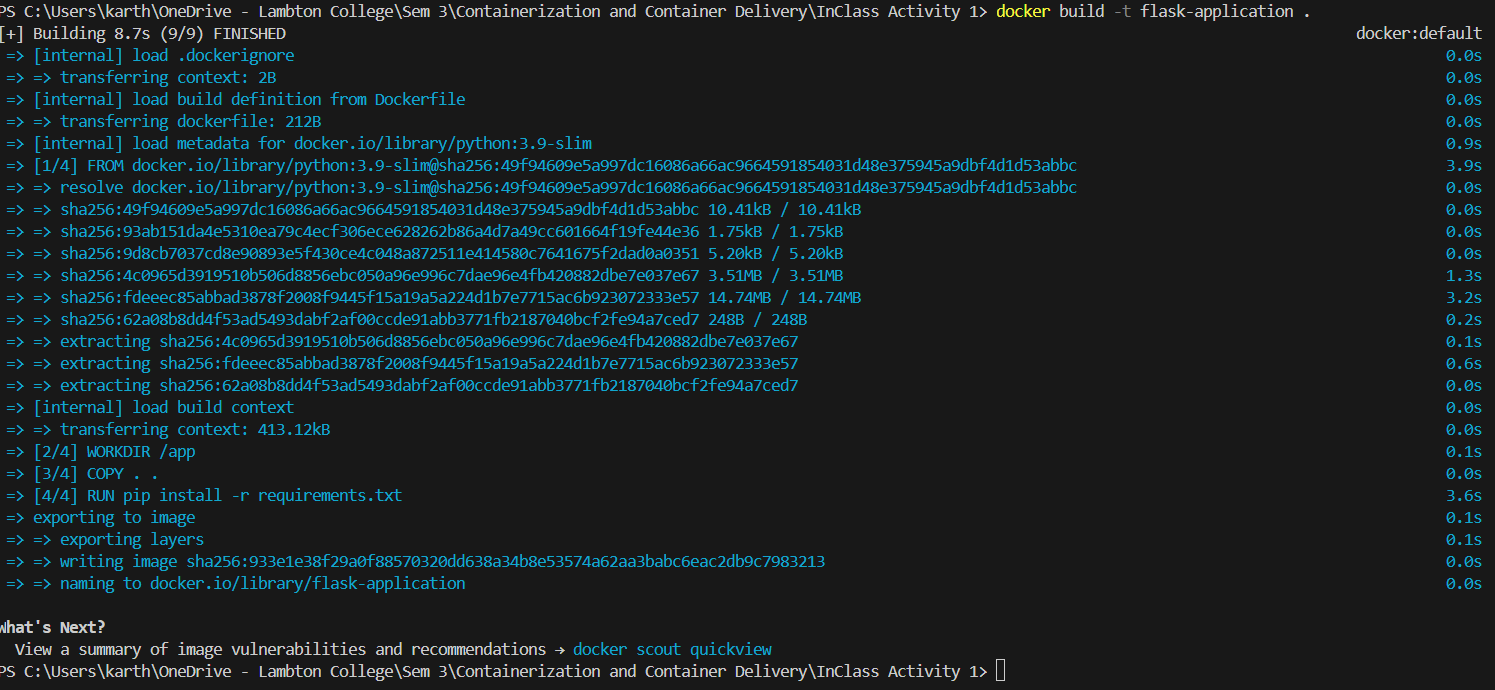
**d) Insert Records into a Postgres Table**

> INSERT INTO tech\_authors VALUES (1, 'Laiba');

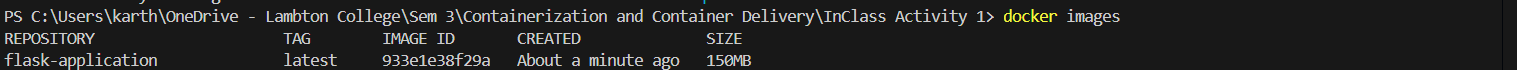
**1.3) Dockerizing the Python Flask Application:**

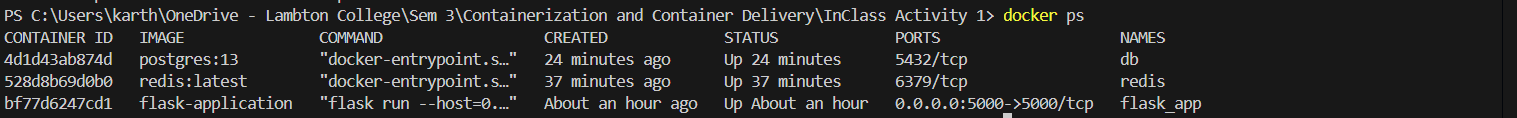
**Dockerfile:** **A screen shot of a computer

Description automatically generated**

> docker build -t flask-application . 

> docker run --name flask\_application --network flask\_network -d -p 5000:5000 flask-application

> docker images – Flask application image created.

> docker ps

Flask Application up and Running: A screenshot of a computer

Description automatically generated

Redis Cache is accessible: A screenshot of a computer

Description automatically generated

Database is accessible: A screenshot of a computer

Description automatically generated

**Task-2:** On a **different host**(VM), deploy the stack using **docker compose**. You should reuse the images you will be creating in task-1

**Solution:**

**Note:** Task 2 is done in Vagrant VM

**2.1) Tagging and Pushing Image into my personal docker repository:**

**1) Flask Application**

> docker image tag flask-app karthik8898/flaskapplication

> docker push karthik8898/flaskapplication

**2) Redis Container**

> docker image tag redis karthik8898/redis

> docker push karthik8898/redis

**3) Modified Postgres\_DB**

> docker image tag redis karthik8898/db

> docker push karthik8898/db

A black screen with white text

Description automatically generated

**2.2) Creating Docker-compose.yml file to run the application stack in the VM:**

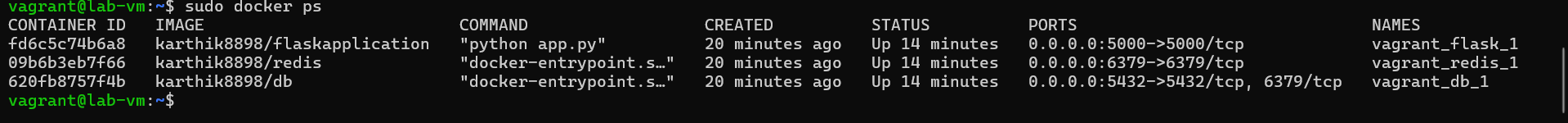
**Docker-compose.yml file**

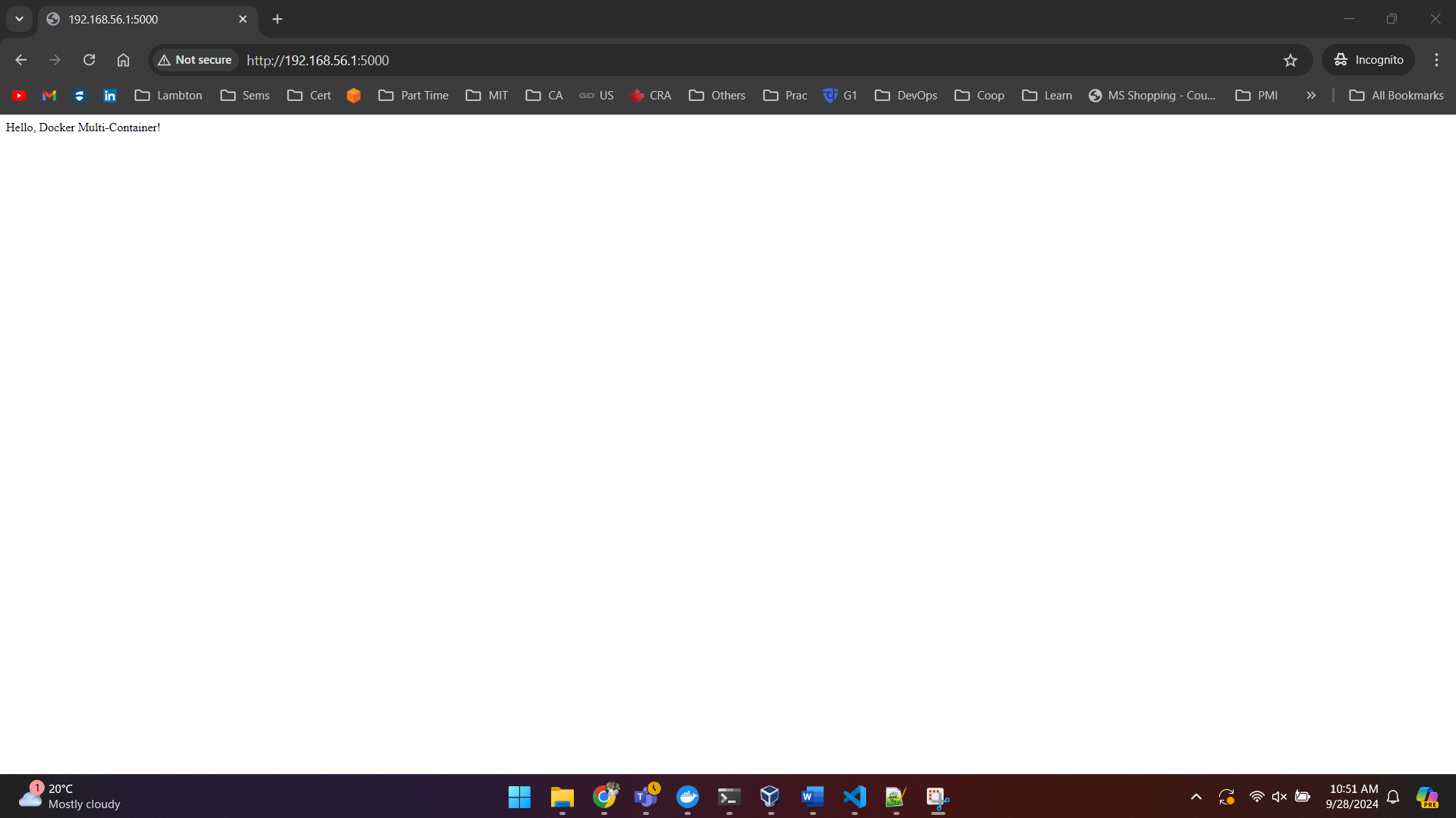


> docker-compose up -d

A screenshot of a computer

Description automatically generated

> docker ps 

Flask application accessible from the VM using docker-compose: 

Redis Cache accessible from the VM using docker-compose: A screenshot of a computer

Description automatically generated

Postgres DB accessible from the VM using docker-compose:

A screenshot of a computer

Description automatically generated