**Assignment 06**  
  
HPCSA 0824 Cloud &amp; Security Lab Assignment 6 1.

**Q1**

Create a python program that takes 2 numbers from the user and displays addition and multiplication result. Now create an image of this program. Run the container from this image and display the results.

**Mkdir app  
  
cd app**  
 **Product.py**

**sudo nano product.py**  
# Python Program to add and multiply two numbers

def main():

num1 = float(input("Enter the first number: "))

num2 = float(input("Enter the second number: "))

addition\_result = num1 + num2

multiplication\_result = num1 \* num2

print(f"The sum of {num1} and {num2} is: {addition\_result}")

print(f"The product of {num1} and {num2} is: {multiplication\_result}")

if \_\_name\_\_ == "\_\_main\_\_":

main()

**DockerFile  
sudo Dockerfile** GNU nano 6.2 Dockerfile

# Use an official Python runtime as a base image

FROM python:3.9-slim

# Set the working directory inside the container

WORKDIR /app

# Copy the Python script into the container

COPY Product.py /app/Product.py

# Install any dependencies (if needed)

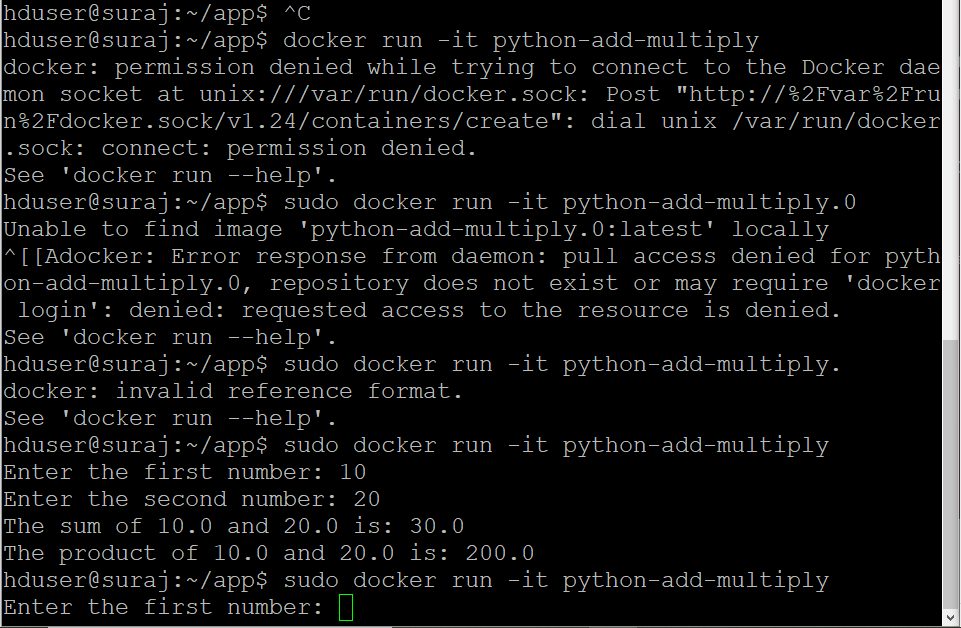
# RUN pip install --no-cache-dir <package-name>

# Set the command to run the Python program

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

Step 3: Build the Docker Image  
 **docker build -t python-add-multiply .  
  
  
check**

**Sudo hduser@suraj:~/app$ sudo docker images ls**

Step 4: Run the Docker Container  
  
sudo **docker run -it python-add-multiply**

**Q2**  
  
Cteate a directory. Create an index.html file. Create a image using httpd image. Copy index.html

inside the container. Run the container and map port 8200 using your image and display the webpage

is displayed.  
  
  
**mkdir my-webpage**

# Create a directory called "my-webpage"

**cd my-webpage**

# Navigate into the "my-webpage" directory

**nano index.html**

# Open the "index.html" file in nano editor to create/edit it

**sudo docker run -d -p 8200:80 --name my-webpage-container httpd**

# Run a Docker container with Apache (httpd), map port 8200 to 80, and name it "my-webpage-container"

**sudo docker ps**

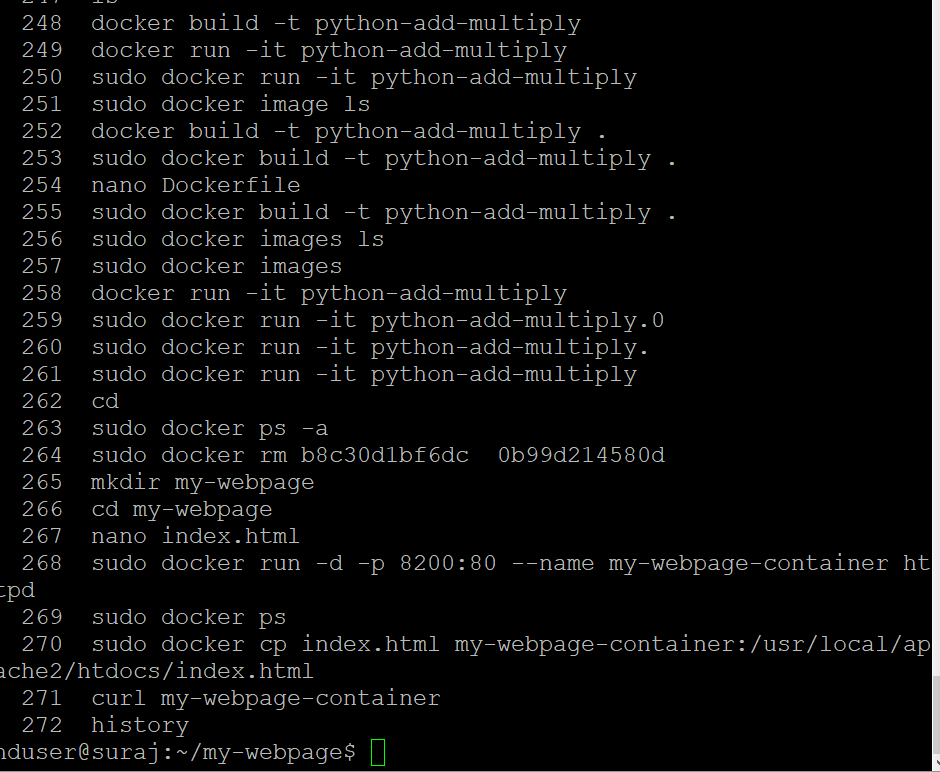
# List running Docker containers to confirm the container is running

**sudo docker cp index.html my-webpage-container:/usr/local/apache2/htdocs/index.html**

# Copy "index.html" to the container's web directory

**curl my-webpage-container**

# Try to use curl to access the container





**Q3**

Create a image of the test.py file. Create a network by name app2. Create a MYSQL container and

connect it to this network. Now create a container of test.py application and connect it to the same

network and display that the database is created.  
  
# Create a custom network

Sudo docker network create app2  
  
**# Run a MySQL container connected to the app2 network**

Sudo docker run --name mysql-container --network app2 -e MYSQL\_ROOT\_PASSWORD=root -e MYSQL\_DATABASE=testdb -d mysql:latest  
  
**Dockerfile**  
FROM python:3.9-slim

WORKDIR /app

COPY . /app

RUN pip install mysql-connector-python

CMD ["python", "test.py"]  
  
  
  
t**est.py**import mysql.connector

import os

MYSQL\_HOST = os.getenv("MYSQL\_HOST", "mysql-container")

# Get the MySQL host from environment variables or use the default "mysql-container"

MYSQL\_USER = os.getenv("MYSQL\_USER", "root")

# Get the MySQL user from environment variables or use the default "root"

MYSQL\_PASSWORD = os.getenv("MYSQL\_PASSWORD", "root")

# Get the MySQL password from environment variables or use the default "root"

MYSQL\_DB = os.getenv("MYSQL\_DB", "testdb")

# Get the MySQL database name from environment variables or use the default "testdb"

connection = mysql.connector.connect(

host=MYSQL\_HOST,

user=MYSQL\_USER,

password=MYSQL\_PASSWORD,

database=MYSQL\_DB

)

# Connect to the MySQL server using the provided credentials and database

cursor = connection.cursor()

# Create a cursor to execute SQL queries

cursor.execute("SHOW DATABASES LIKE %s", (MYSQL\_DB,))

# Query to check if the specified database exists

db = cursor.fetchone()

# Fetch the result of the query, will return None if no match is found

if db:

print(f"Database '{MYSQL\_DB}' exists!")

# If the database exists, print a success message

else:

print(f"Database '{MYSQL\_DB}' does not exist!")

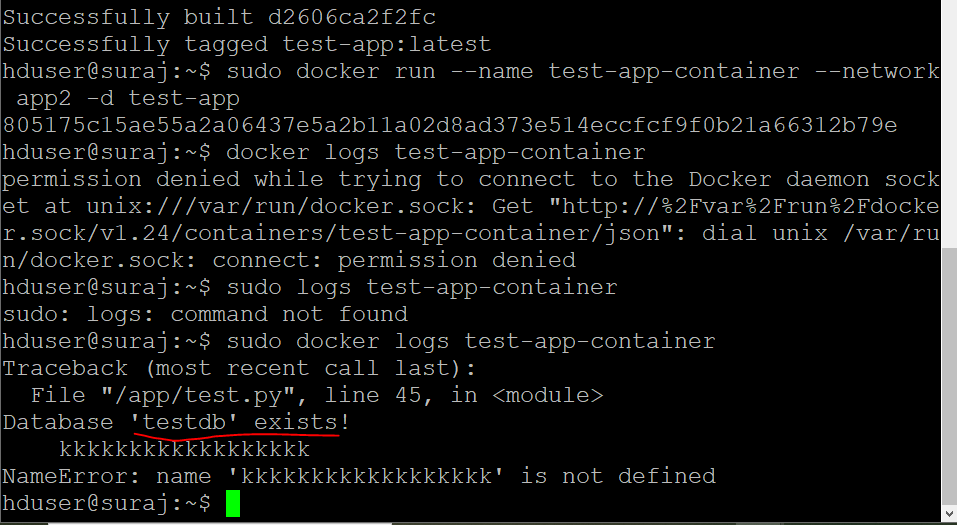
# If the database doesn't exist, print an error message

cursor.close()

# Close the cursor to release database resources

connection.close()

# Close the database connection to clean up

3. **Build and Run the Python App Container**# Build the Python app Docker image  
 sudo docker build -t test-app .   
  
# Run the Python app container connected to the same network   
  
sudo docker run --name test-app-container --network app2 -d test-app  
  
  
4. **Checking the Logs  
  
sudo** docker logs test-app-container  
  
  
  
  
  
  
**Q 4**.   
Create a docker compose file to start 4 containers using httpd image. Map ports 5000, 6000, 7000 and 8000. Port 8000 website should display “Welcome to ACTS”. Port 5000 should display “Welcome to HPCSA”. Port 6000 should display “Welcome to DITISS” and port 7000 should display “Welcome to DAI”. Use docker-compose command.  
  
 mkdir myproject

213 sudo nano compose.yml

214 mkdir web\_content

215 mv compose.yml myproject

216 cd web\_content

217 mkdir hpcsa

218 cd hpcsa

219 mkdir index.html

220 nano index.html

221 ls

222 sudo rm -rm index.html

223 sudo rm -rf index.html

224 nano index.html

225 cd .

226 mkdir ditiss

227 cd ditiss

228 index.html

229 nano index.html

230 cd .

231 clear

232 ls

233 cd

234 ls

235 sudo rm -rf myproject

236 ls

237 clear

238 mkdir myproject/

239 cd myproject/

240 sudo nano docker-compose.yml

241 mkdir web\_content

242 cd web\_content/

243 mkdir hpcsa

244 cd hpcsa

245 sudo nano index.html

246 cd ..

247 mkdir ditiss

248 cd ditiss/

249 sudo nano index.html

250 cd ..

251 mkdir dai

252 cd dai

253 sudo index.html

254 sudo nano index.html

255 cd ..

256 mkdir acts

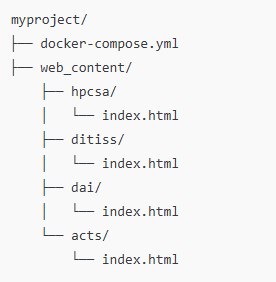
257 cd acts

258 sudo nano index.html

259 docker-compose up -d

260 snap install docker

261 sud snap install docker

262 sudo apt install docker-compose  
 **docker-compose up -d**  
  


**Yaml file**  
version: '3.8'

services:

hpcsa:

image: httpd:latest

container\_name: hpcsa

ports:

- "5000:80"

volumes:

- ./web\_content/hpcsa:/usr/local/apache2/htdocs/

ditiss:

image: httpd:latest

container\_name: ditiss

ports:

- "6000:80"

volumes:

- ./web\_content/ditiss:/usr/local/apache2/htdocs/

dai:

image: httpd:latest

container\_name: dai

ports:

- "7000:80"

volumes:

- ./web\_content/dai:/usr/local/apache2/htdocs/

acts:

image: httpd:latest

container\_name: acts

ports:

- "8000:80"

volumes:

- ./web\_content/acts:/usr/local/apache2/htdocs/

