**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Skill-10: Containeraize backed application (Springboot Project) using Docker Composer with volume**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Skill-9 Continuation:**

**What is a Docker Volume?**

**----------------------------**

**A volume is persistent storage managed by Docker.**

**Normally, when you remove a container, its internal data (like your MySQL tables) gets deleted.**

**With a volume, data is stored outside the container lifecycle, so even if the container is removed/rebuilt, your database contents remain safe.**

**2) Types of mounts**

**-------------------**

**Named volume : mysql\_data:/var/lib/mysql**

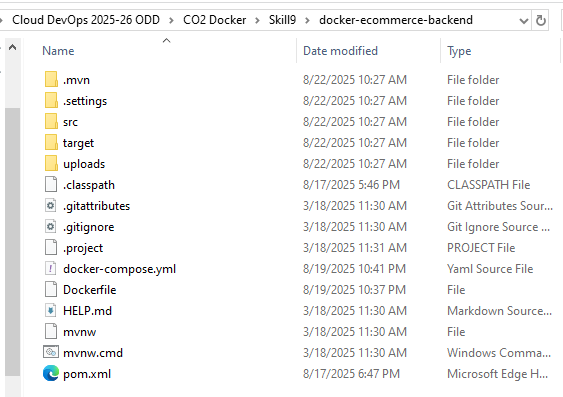
**Managed by Docker, portable, fewer permission issues.**

**Bind mount: C:\path\on\host:/container/path**

**Directly maps a host folder. Good for dev files, but DBs can hit permission/perf issues on Windows.**

**Anonymous volume: created implicitly.**

**Open the skill-9 springboot project from the VS Code using (code .) command from the following dir.**

****

**Delete the existing docker-compose.yml file and create the same file again.**

**Copy and paste the following code in the docker-compose.yml file.**

services:

db:

image: mysql:8.0

container\_name: ecommerce-db

environment:

MYSQL\_ROOT\_PASSWORD: root

MYSQL\_DATABASE: ecommerce

volumes:

- mysql\_data:/var/lib/mysql # ← persistent MySQL data

healthcheck:

test: ["CMD", "mysqladmin", "ping", "-h", "127.0.0.1", "-uroot", "-proot"]

interval: 5s

timeout: 3s

retries: 20

# (optional) expose DB to host tools like Workbench/DBeaver:

# ports:

# - "3307:3306" # host:container

backend:

build: .

container\_name: ecommerce-backend

depends\_on:

db:

condition: service\_healthy

environment:

SERVER\_PORT: 8081

SPRING\_DATASOURCE\_URL: jdbc:mysql://db:3306/ecommerce?useSSL=false&allowPublicKeyRetrieval=true&serverTimezone=UTC

SPRING\_DATASOURCE\_USERNAME: root

SPRING\_DATASOURCE\_PASSWORD: root

ports:

- "8081:8081"

restart: unless-stopped

volumes:

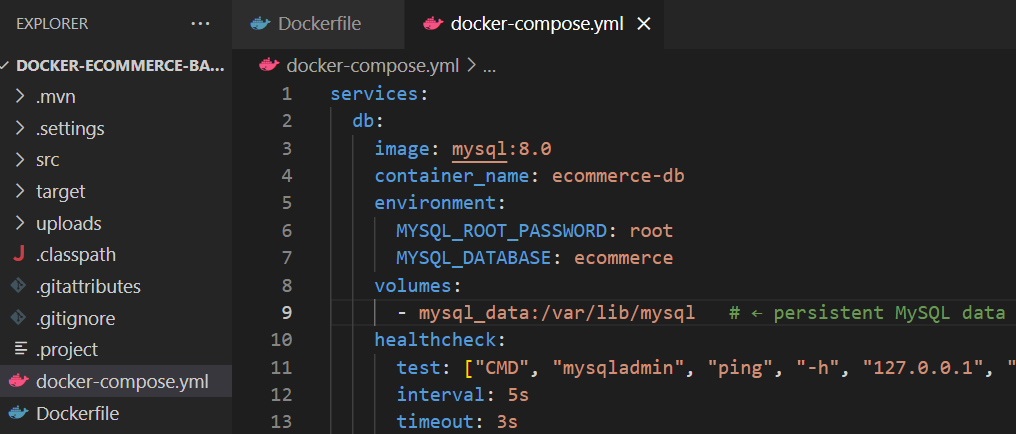
mysql\_data: {}

**modify mysql username and password, if you have other than “root”**

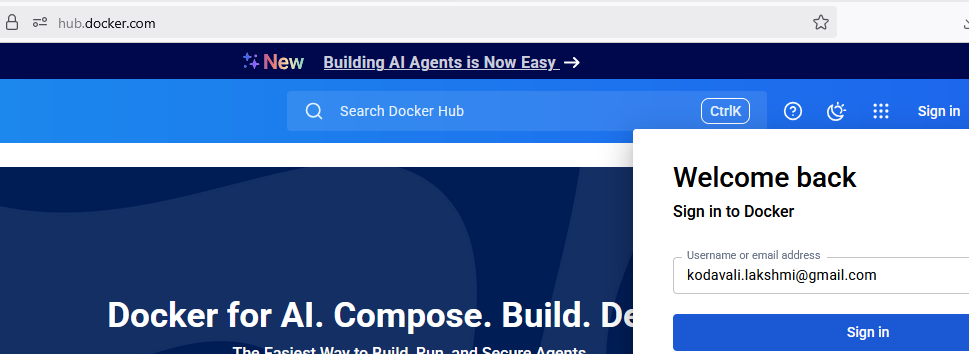
**Save this file.**

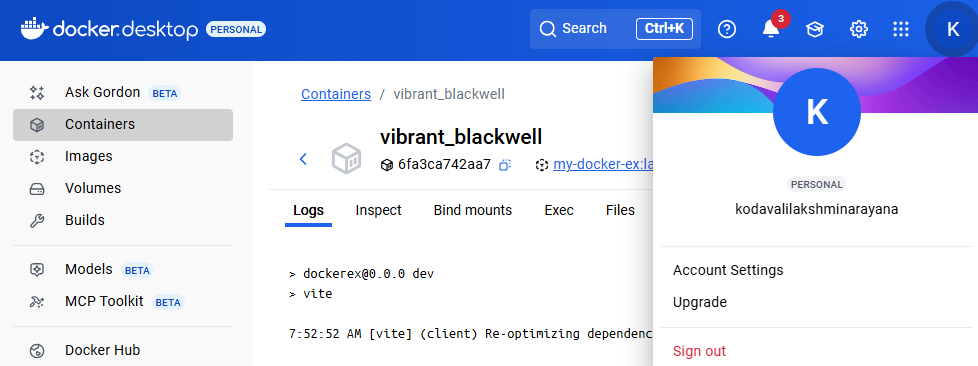
**Verify and understand the code present in the docker-compose.yml file as follows:**

**It must be present in the folder where pom.xml file is located, because it is having only backend one project.**

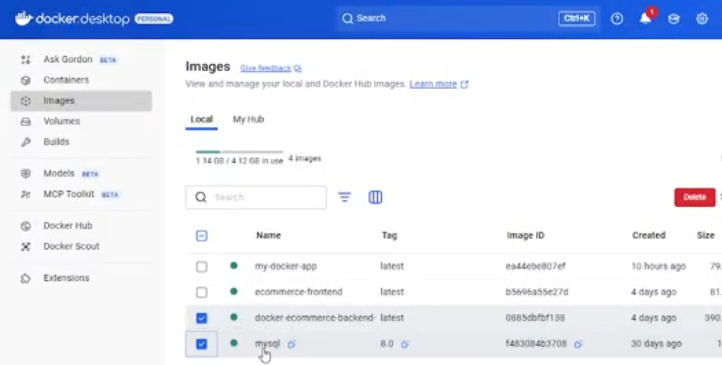
****

**Open the Docker Desktop and Docker Hub and Login into both.**

****

****

**Delete the image and container which was created in skill-9 from the docker desktop.**

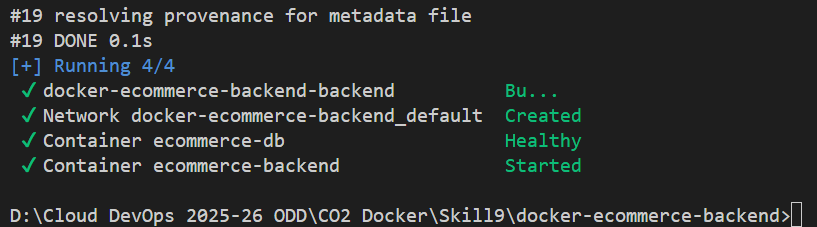
****

**Delete the above two selected images which were created in skill-9.**

Open command prompt from VS Code, and execute the following commands:

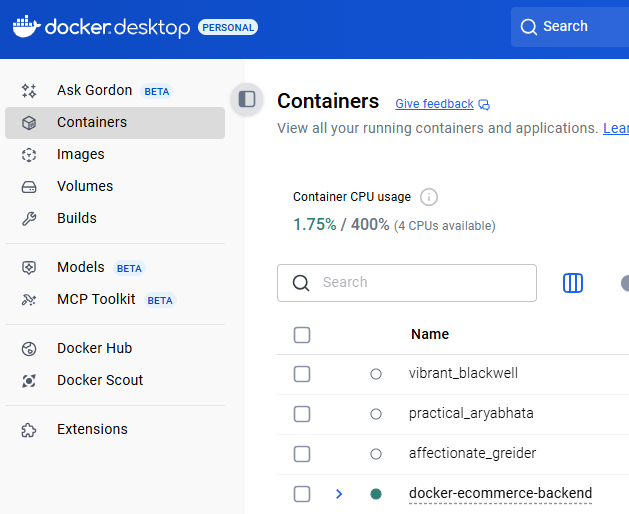
$docker login docker.io //you must get login succeeded message.

$ docker compose up -d --build

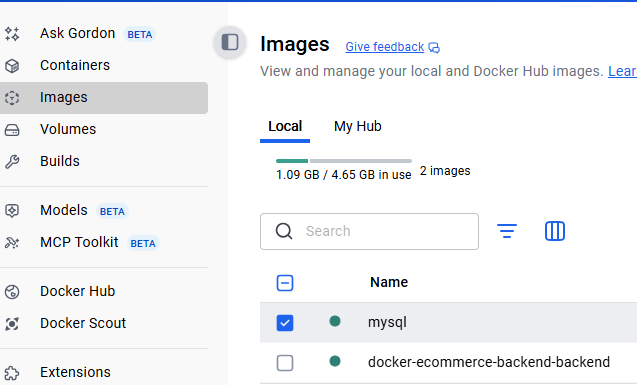


Build successful.

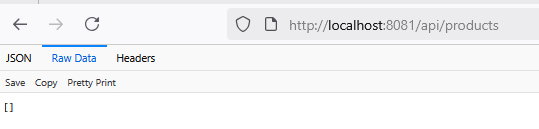
Verify the created container in the Containers tab (docker-ecommerce-backend)



Verify the Two images were created as follows:



**To verify the output, go to the browser and type:** [**http://localhost:8081/api/products**](http://localhost:8081/api/products)

****

**To see the output, we need to execute the database queries to insert records in the mysql container. (Note: Not in the local system mysql)**

**To open the database container, execute the following command in VS Code:**

**$docker exec -it ecommerce-db mysql -uroot -proot**

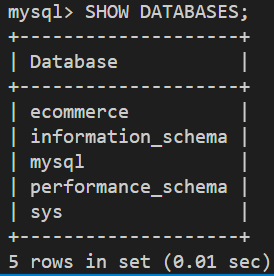
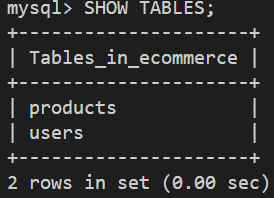
**Now mysql> container terminal is opened.**

**Now execute the following commands in the mysql container terminal.**

mysql>SHOW DATABASES;

mysql>USE ecommerce;

mysql>SHOW TABLES;

**** ****

-- Insert Computers

mysql>INSERT INTO products (name, category, price, image\_path) VALUES

('Gaming PC', 'computers', 1200.00, 'gaming\_pc.jpeg'),

('Office Desktop', 'computers', 800.00, 'office\_desktop.jpeg'),

('Mini PC', 'computers', 500.00, 'mini\_pc.jpeg'),

('Workstation', 'computers', 2500.00, 'workstation.jpeg');

-- Insert Mobiles

mysql>INSERT INTO products (name, category, price, image\_path) VALUES

('iPhone 14', 'mobiles', 999.00, 'iphone\_14.jpeg'),

('Samsung Galaxy S23', 'mobiles', 899.00, 'samsung\_galaxy\_s23.jpeg'),

('Google Pixel 7', 'mobiles', 799.00, 'google\_pixel\_7.jpeg'),

('OnePlus 11', 'mobiles', 749.00, 'oneplus\_11.jpeg');

-- Insert Laptops

mysql>INSERT INTO products (name, category, price, image\_path) VALUES

('MacBook Air', 'laptops', 1099.00, 'macbook\_air.jpeg'),

('Dell XPS 15', 'laptops', 1299.00, 'dell\_xps\_15.jpeg'),

('Lenovo ThinkPad', 'laptops', 999.00, 'lenovo\_thinkpad.jpeg'),

('HP Spectre x360', 'laptops', 1199.00, 'hp\_spectre\_x360.jpeg');

-- Insert Pendrives

mysql>INSERT INTO products (name, category, price, image\_path) VALUES

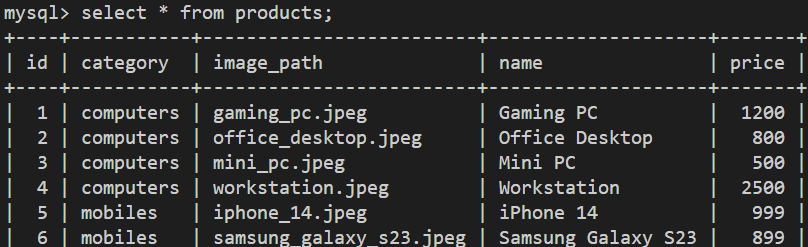
('SanDisk 64GB', 'pendrives', 15.00, 'sandisk\_64gb.jpeg'),

('Kingston 128GB', 'pendrives', 25.00, 'kingston\_128gb.jpeg'),

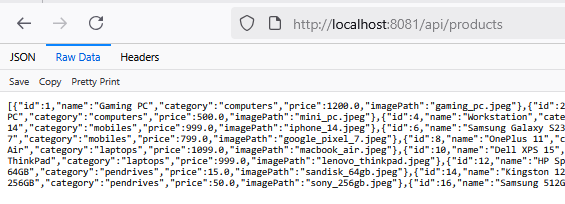
('Sony 256GB', 'pendrives', 50.00, 'sony\_256gb.jpeg'),

('Samsung 512GB', 'pendrives', 80.00, 'samsung\_512gb.jpeg');

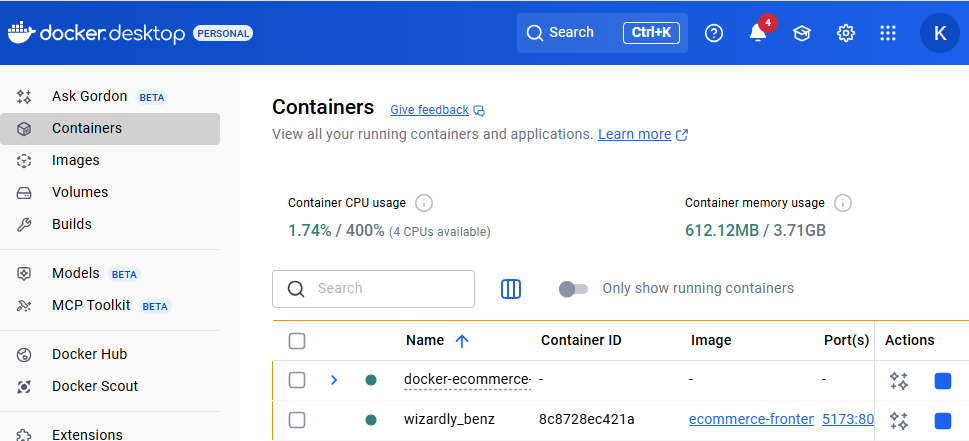
mysql>select \* from products;

****

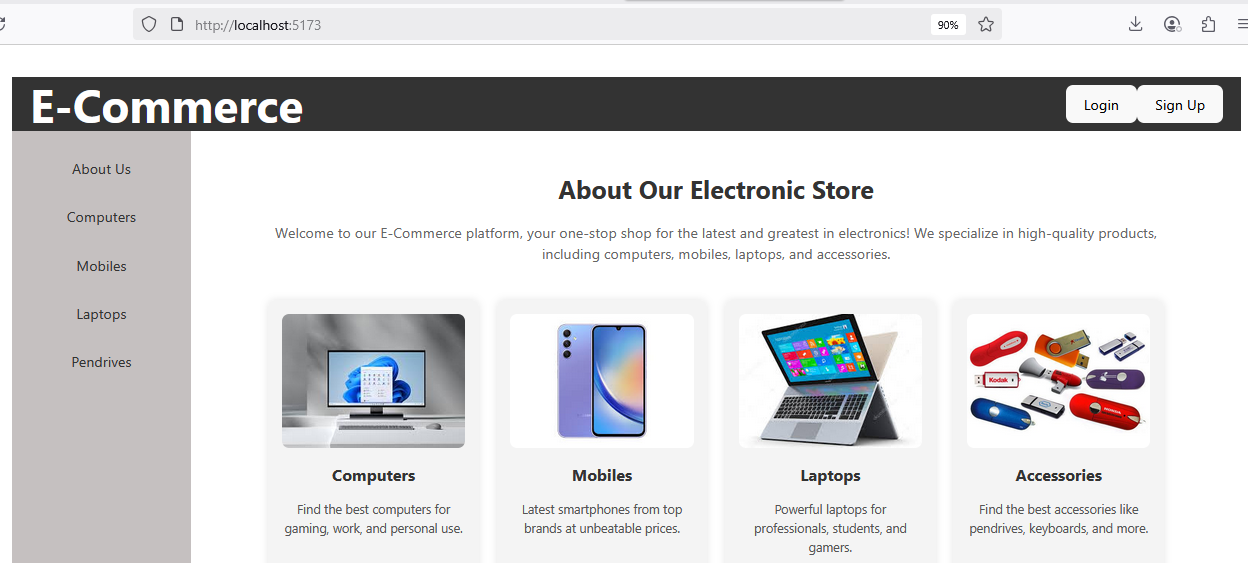
**Now come to the browser and do refresh to verify output**

****

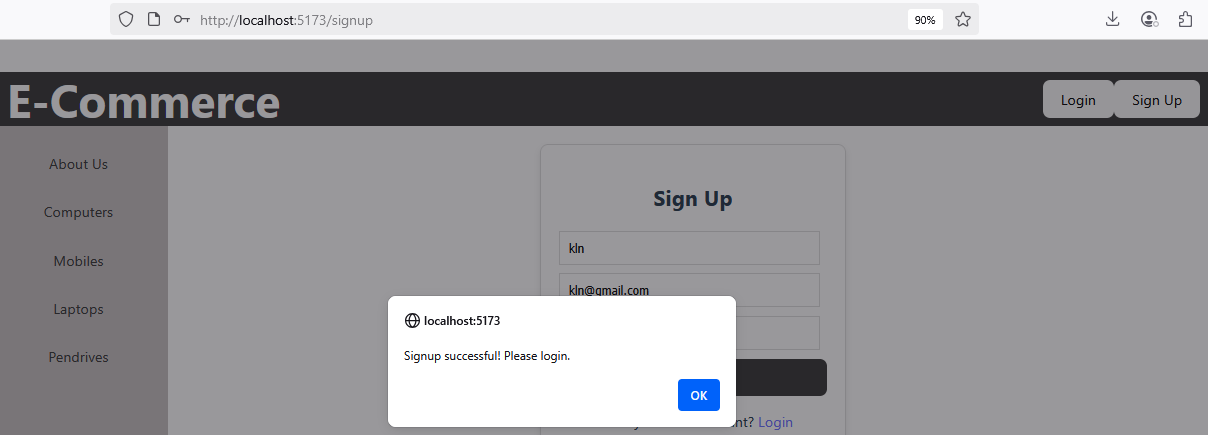
**Now to test with frontend, Run the frontend-container from docker desktop ( which was created with Practical-4 experiment)**

****

**Now click on the frontend container port number to open output as follows:**

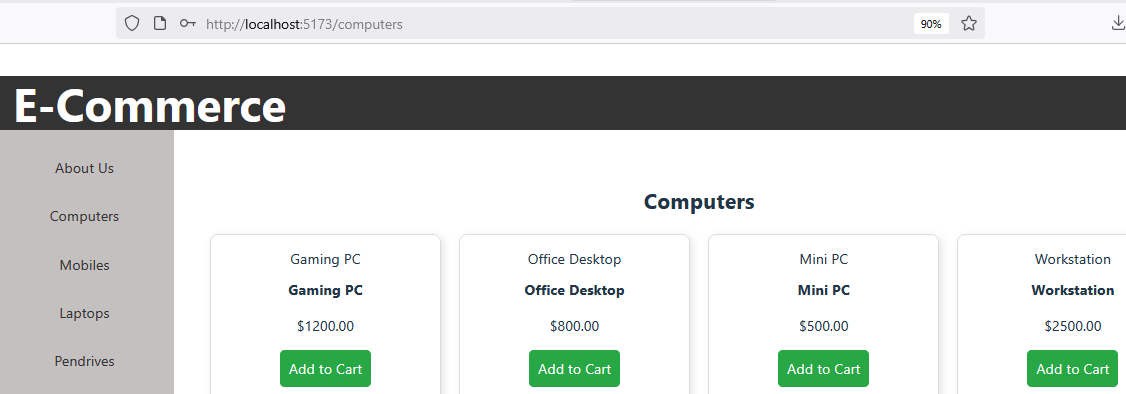
****

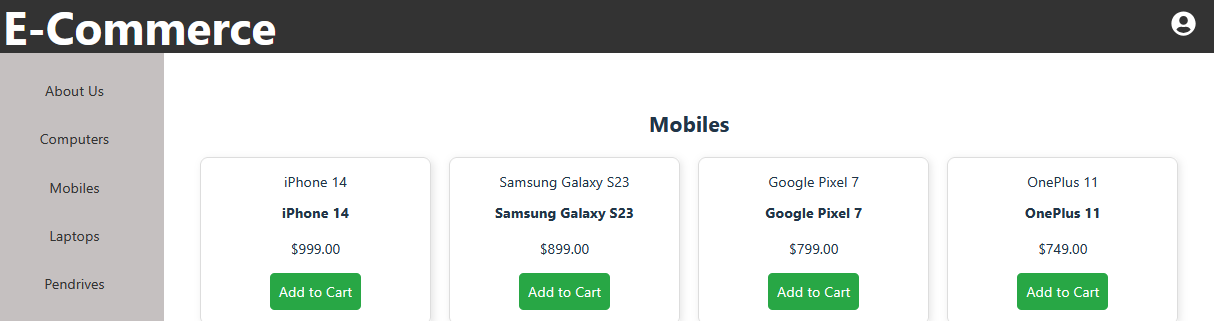
**Now click on Signup Button, and create a account by providing username (kln), email, password**

****

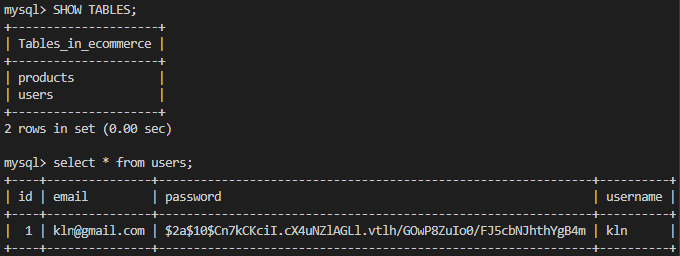
**Now click on Login Button to login with above credentials,**

**Now Click on all the tabs and verify output.**

****

****

**Now verify the “users” table content from the container mysql, where you user name will be there as follows:**

****

**That means both frontend and backend containers are communicating with each other.**

**You can exit from the mysql container using the command: exit**

**Mysql>exit**