DevOps Assignment 2

Setting Up DOCKER – Workshop

Name : Patel Het Kushalkumar

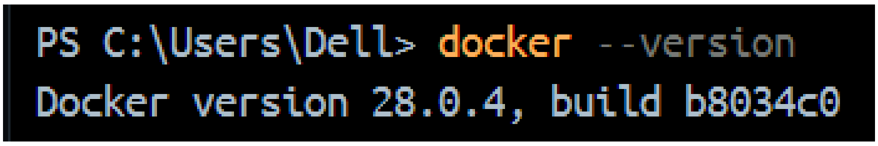
Roll No : 205224014

M.tech Data Analytics

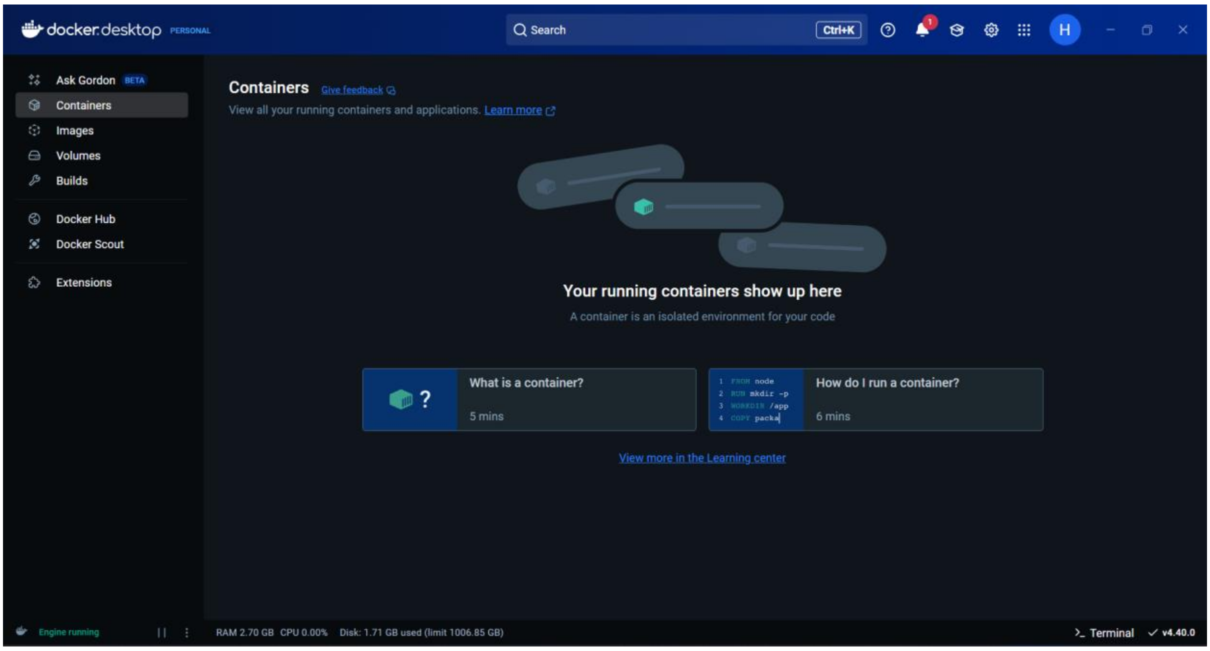
a) Create a Container with PostgresDB or mySQL database installed

1. Install Docker

- Download and install Docker Desktop.
- After installation, verify Docker is running by executing the following command in the terminal:



```
PS C:\Users\Dell> docker --version
Docker version 28.0.4, build b8034c0
```



Containers

View all your running containers and applications.

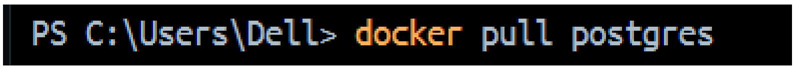
Your running containers show up here

A container is an isolated environment for your code

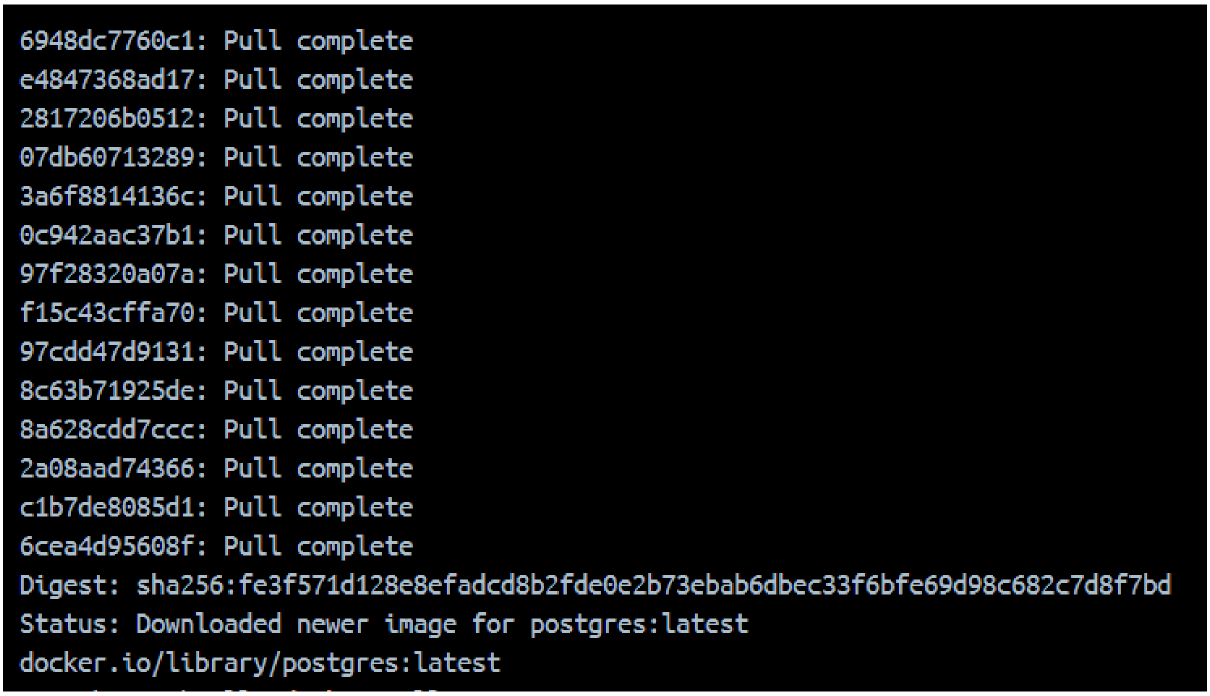
RAM 2.70 GB CPU 0.00% Disk: 1.71 GB used (limit 1006.85 GB)

2. Pull the Docker Image for PostgreSQL or MySQL

- **For PostgreSQL**: Run the following command to pull the official PostgreSQL Docker image:



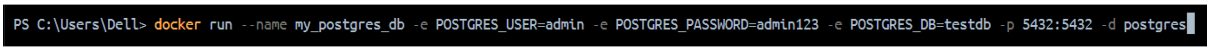
```
PS C:\Users\Dell> docker pull postgres
```



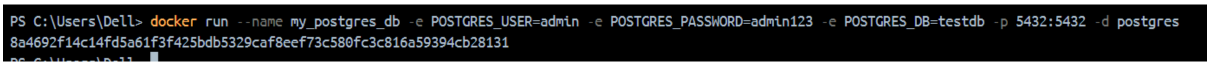
```
6948dc7760c1: Pull complete
e4847368ad17: Pull complete
2817206b0512: Pull complete
07db60713289: Pull complete
3a6f8814136c: Pull complete
0c942aac37b1: Pull complete
97f28320a07a: Pull complete
f15c43cffa70: Pull complete
97cdd47d9131: Pull complete
8c63b71925de: Pull complete
8a628cdd7ccc: Pull complete
2a08aad74366: Pull complete
c1b7de8085d1: Pull complete
6cea4d95608f: Pull complete
Digest: sha256:fe3f571d128e8efadcd8b2fde0e2b73ebab6dbec33f6bfe69d98c682c7d8f7bd
Status: Downloaded newer image for postgres:latest
docker.io/library/postgres:latest
```

3. Create a Docker Container with PostgreSQL or MySQL

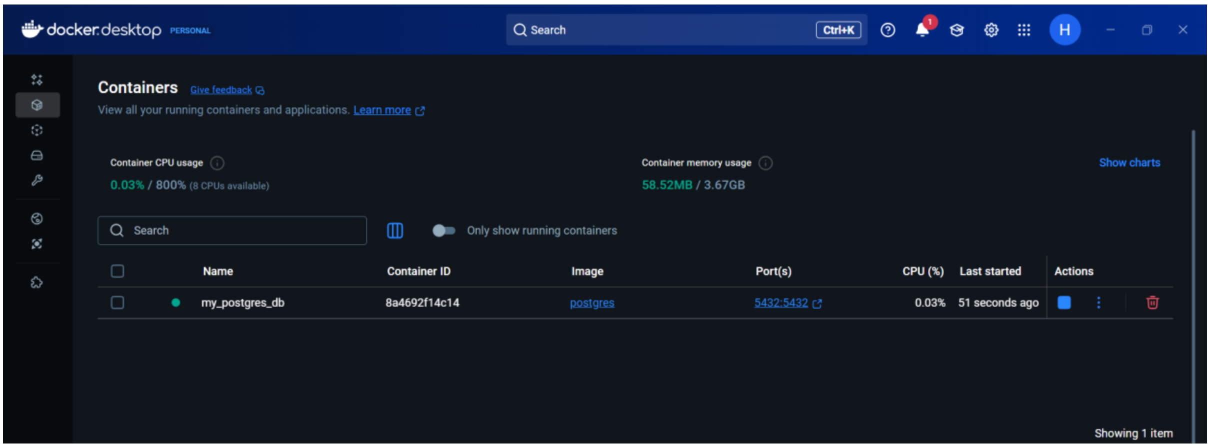
- **For PostgreSQL**: Run the following command to create a PostgreSQL container:



```
PS C:\Users\Dell> docker run --name my_postgres_db -e POSTGRES_USER=admin -e POSTGRES_PASSWORD=admin123 -e POSTGRES_DB=testdb -p 5432:5432 -d postgres
```



```
PS C:\Users\Dell> docker run --name my_postgres_db -e POSTGRES_USER=admin -e POSTGRES_PASSWORD=admin123 -e POSTGRES_DB=testdb -p 5432:5432 -d postgres
8a4692f14c14fd5a61f3f425bdb5329caf8eef73c580fc3c816a59394cb28131
```



Containers Give feedback

View all your running containers and applications. Learn more

Container CPU usage: 0.03% / 800% (8 CPUs available)

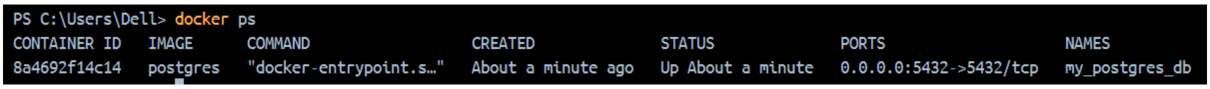
Container memory usage: 58.52MB / 3.67GB

□		Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
□	●	my_postgres_db	8a4692f14c14	postgres	5432:5432	0.03%	51 seconds ago	

Showing 1 item

4. Verify the Container is Running

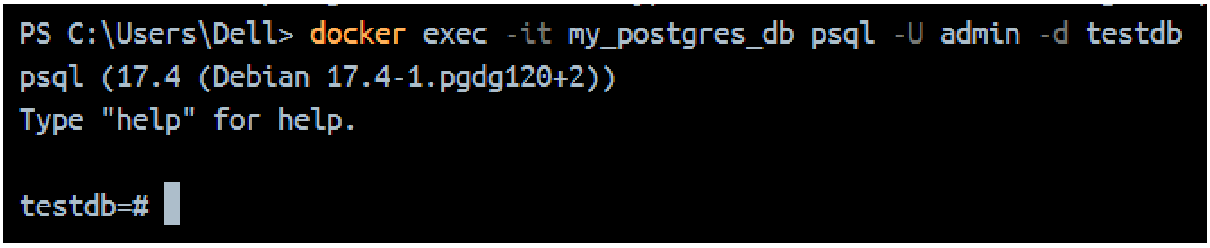
- To check if the container is running, use the following command:



```
PS C:\Users\Dell> docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED              STATUS              PORTS                    NAMES
8a4692f14c14   postgres   "docker-entrypoint.s…"   About a minute ago   Up About a minute   0.0.0.0:5432->5432/tcp   my_postgres_db
```

5. Access the Database

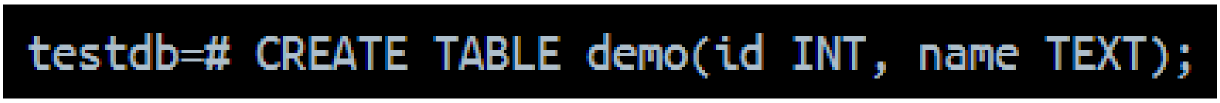
- **For PostgreSQL**: Access the PostgreSQL database by running the following command:



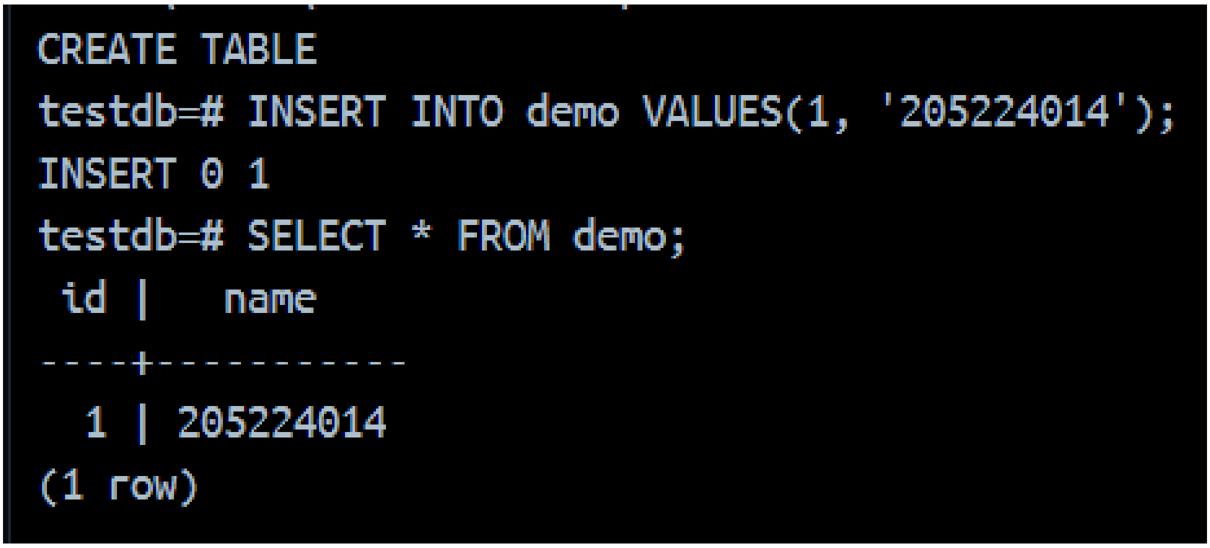
```
PS C:\Users\Dell> docker exec -it my_postgres_db psql -U admin -d testdb
psql (17.4 (Debian 17.4-1.pgdg120+2))
Type "help" for help.

testdb=#
```

6. Creating and Managing Databases

- **For PostgreSQL**: To create a new database in PostgreSQL, you can run:



```
testdb=# CREATE TABLE demo(id INT, name TEXT);
```

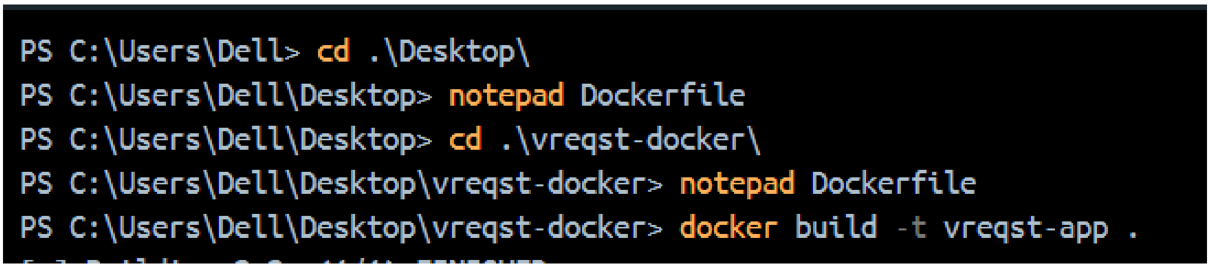


```
CREATE TABLE
testdb=# INSERT INTO demo VALUES(1, '205224014');
INSERT 0 1
testdb=# SELECT * FROM demo;
 id |   name
----+-----------
  1 | 205224014
(1 row)
```

b) Deploy VReqST – A requirement specification tool in a container.

Step 1: Create Dockerfile

Inside vreqst-docker/, create a file named **Dockerfile**:



```
PS C:\Users\Dell> cd .\Desktop\
PS C:\Users\Dell\Desktop> notepad Dockerfile
PS C:\Users\Dell\Desktop> cd .\vreqst-docker\
PS C:\Users\Dell\Desktop\vreqst-docker> notepad Dockerfile
PS C:\Users\Dell\Desktop\vreqst-docker> docker build -t vreqst-app .
```

Dockerfile

# Use Node.js base image

FROM node:14

# Set working directory

WORKDIR /app

# Copy local files into the container

COPY . .

# Set environment variables (replace with actual MongoDB URIs if needed)

ENV validation_server=http://localhost:5001

ENV backend=http://localhost:5002

# Install backend dependencies

WORKDIR /app/VReqST-main/VReqST/backend

RUN npm install

# Install validation server dependencies

```
WORKDIR /app/VReqST-main/VReqST/validation_server

RUN npm install


# Install frontend dependencies

WORKDIR /app/VReqST-main/VReqST/frontend

RUN npm install

RUN npm run client-install || echo "client-install script not found, skipping..."


# Expose necessary ports

EXPOSE 3000 5001 5002


# Start all services

CMD sh -c "cd /app/VReqST-main/VReqST/backend && nodemon index.js & \

        cd /app/VReqST-main/VReqST/validation_server && nodemon index.js & \

        cd /app/VReqST-main/VReqST/frontend && npm run dev"
```

Step 2: Create docker-compose.yml

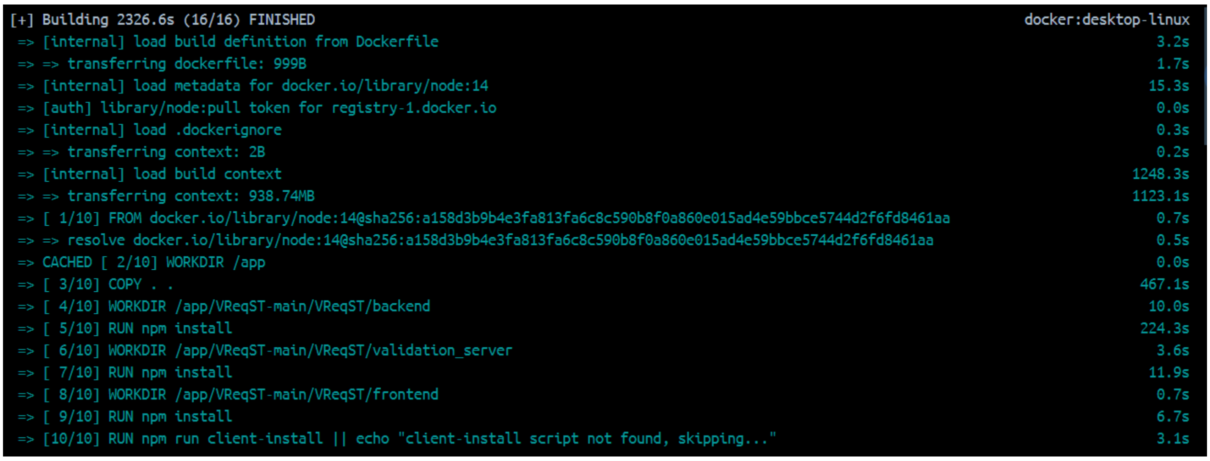
In the **same directory** (vreqst-docker/), create a file named docker-compose.yml:

docker-compose.yml

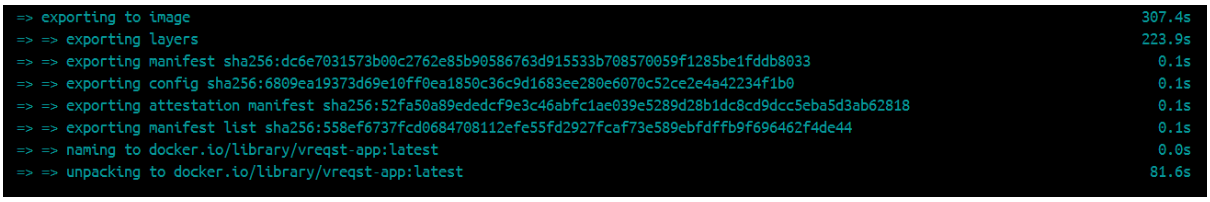
```
version: '3'


services:

  vreqst-app:

    build: .

    ports:

      - "3000:3000"   # Frontend

      - "5001:5001"   # Validation Server

      - "5002:5002"   # Backend

    volumes:
      - .:/app
    working_dir: /app
    command: sh -c "cd VReqST-2-main/VReqST-main/VReqST/backend && nodemon index.js & \
             cd VReqST-2-main/VReqST-main/VReqST/validation_server && nodemon index.js & \
             cd VReqST-2-main/VReqST-main/VReqST/frontend && npm run dev"
```

Step 3: Build the Image

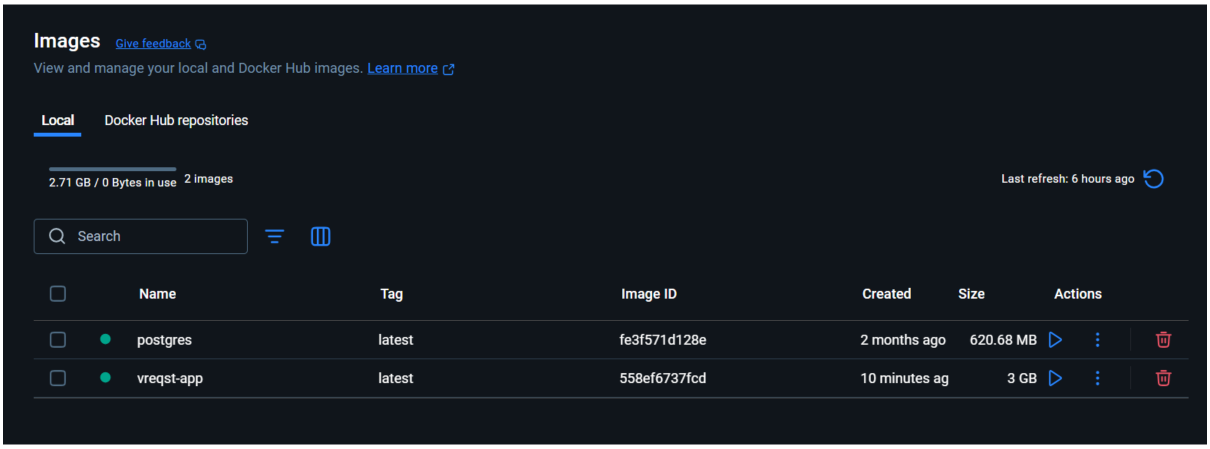
In terminal (Command Prompt or PowerShell), navigate to the vreqst-docker folder and run:



```
[+] Building 2326.6s (16/16) FINISHED                                                        docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                         3.2s
 => => transferring dockerfile: 999B                                                                         1.7s
 => [internal] load metadata for docker.io/library/node:14                                                  15.3s
 => [auth] library/node:pull token for registry-1.docker.io                                                  0.0s
 => [internal] load .dockerignore                                                                            0.3s
 => => transferring context: 2B                                                                              0.2s
 => [internal] load build context                                                                         1248.3s
 => => transferring context: 938.74MB                                                                     1123.1s
 => [ 1/10] FROM docker.io/library/node:14@sha256:a158d3b9b4e3fa813fa6c8c590b8f0a860e015ad4e59bbce5744d2f6fd8461aa  0.7s
 => => resolve docker.io/library/node:14@sha256:a158d3b9b4e3fa813fa6c8c590b8f0a860e015ad4e59bbce5744d2f6fd8461aa  0.5s
 => CACHED [ 2/10] WORKDIR /app                                                                              0.0s
 => [ 3/10] COPY . .                                                                                       467.1s
 => [ 4/10] WORKDIR /app/VReqST-main/VReqST/backend                                                         10.0s
 => [ 5/10] RUN npm install                                                                                224.3s
 => [ 6/10] WORKDIR /app/VReqST-main/VReqST/validation_server                                                3.6s
 => [ 7/10] RUN npm install                                                                                 11.9s
 => [ 8/10] WORKDIR /app/VReqST-main/VReqST/frontend                                                         0.7s
 => [ 9/10] RUN npm install                                                                                  6.7s
 => [10/10] RUN npm run client-install || echo "client-install script not found, skipping..."                3.1s
```



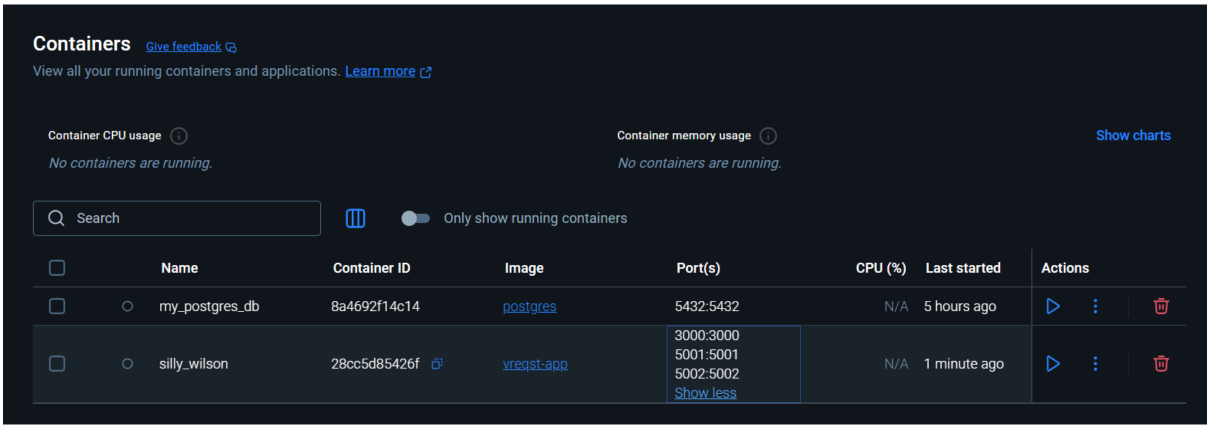
```
 => exporting to image                                                                                     307.4s
 => => exporting layers                                                                                    223.9s
 => => exporting manifest sha256:dc6e7031573b00c2762e85b90586763d915533b708570059f1285be1fddb8033             0.1s
 => => exporting config sha256:6809ea19373d69e10ff0ea1850c36c9d1683ee280e6070c52ce2e4a42234f1b0               0.1s
 => => exporting attestation manifest sha256:52fa50a89ededcf9e3c46abfc1ae039e5289d28b1dc8cd9dcc5eba5d3ab62818  0.1s
 => => exporting manifest list sha256:558ef6737fcd0684708112efe55fd2927fcaf73e589ebfdffb9f696462f4de44        0.1s
 => => naming to docker.io/library/vreqst-app:latest                                                         0.0s
 => => unpacking to docker.io/library/vreqst-app:latest                                                     81.6s
```



Images Give feedback

View and manage your local and Docker Hub images. Learn more

□		Name	Tag	Image ID	Created	Size	Actions
□	●	postgres	latest	fe3f571d128e	2 months ago	620.68 MB	▷ ⋮ 🗑
□	●	vreqst-app	latest	558ef6737fcd	10 minutes ag	3 GB	▷ ⋮ 🗑



Containers Give feedback

View all your running containers and applications. Learn more

Container CPU usage

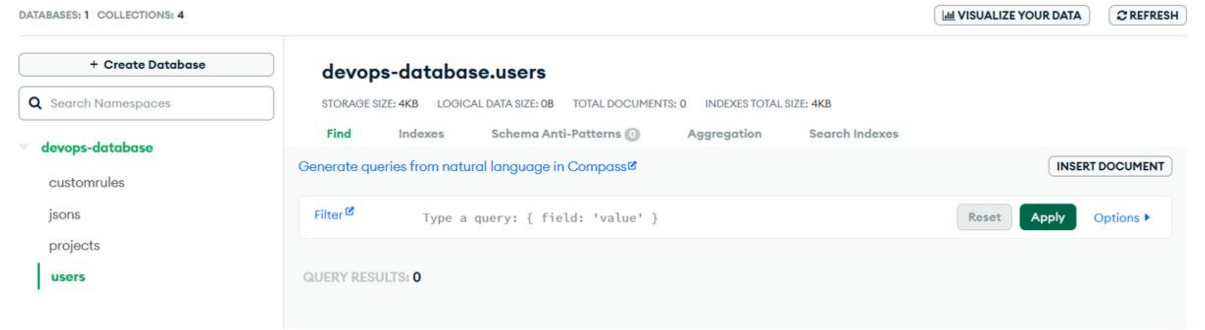
No containers are running.

Container memory usage

No containers are running.

□		Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
□	○	my_postgres_db	8a4692f14c14	postgres	5432:5432	N/A	5 hours ago	
□	○	silly_wilson	28cc5d85426f	vreqst-app	3000:3000 5001:5001 5002:5002 Show less	N/A	1 minute ago	

Step 4 : Setting up MongoDB Database



DATABASES: 1 COLLECTIONS: 4

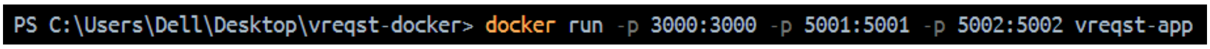
devops-database.users

STORAGE SIZE: 4KB LOGICAL DATA SIZE: 0B TOTAL DOCUMENTS: 0 INDEXES TOTAL SIZE: 4KB

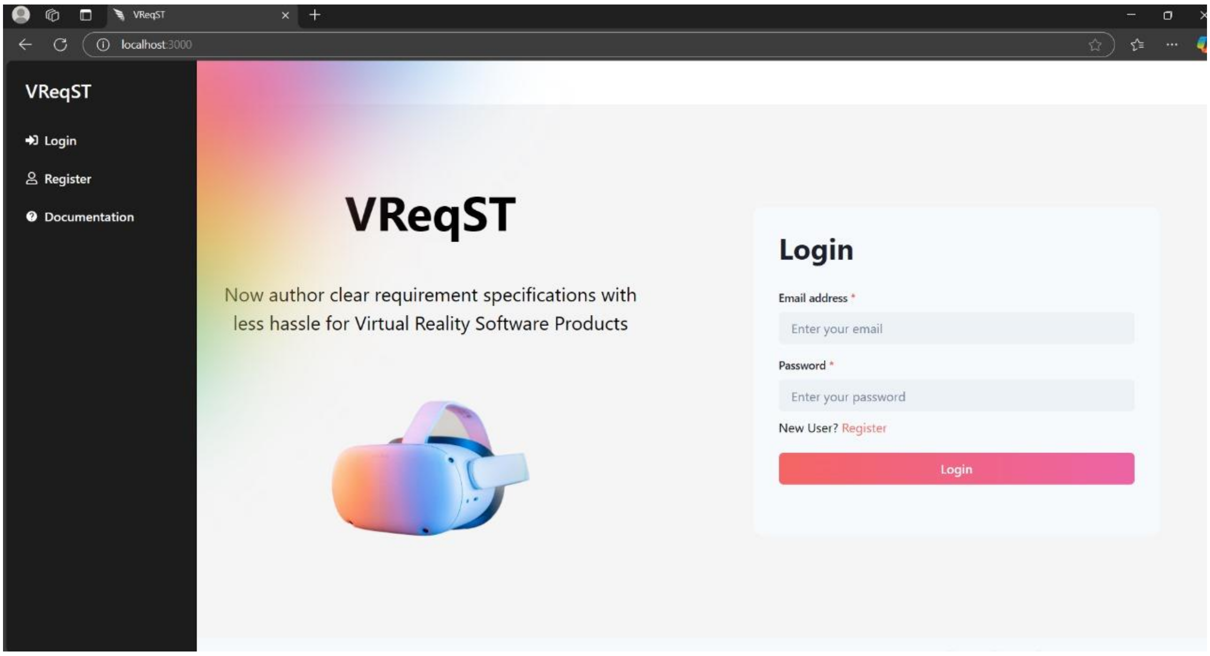
QUERY RESULTS: 0

Step 5: Run the Container

Start the app:



```
PS C:\Users\Dell\Desktop\vreqst-docker> docker run -p 3000:3000 -p 5001:5001 -p 5002:5002 vreqst-app
```



VReqST

Now author clear requirement specifications with less hassle for Virtual Reality Software Products

Login

Email address *

Enter your email

Password *

Enter your password

New User? Register