CA6C1 – DevOps - National Institute of Technology, Trichy

Assignment 2 – Setting Up DOCKER – Workshop

Name : Sudhanshu Kumar Tiwary

Roll No. :205224024

Q1 (A): Create a Container with PostgresDB or mySQL database installed.

Objective:

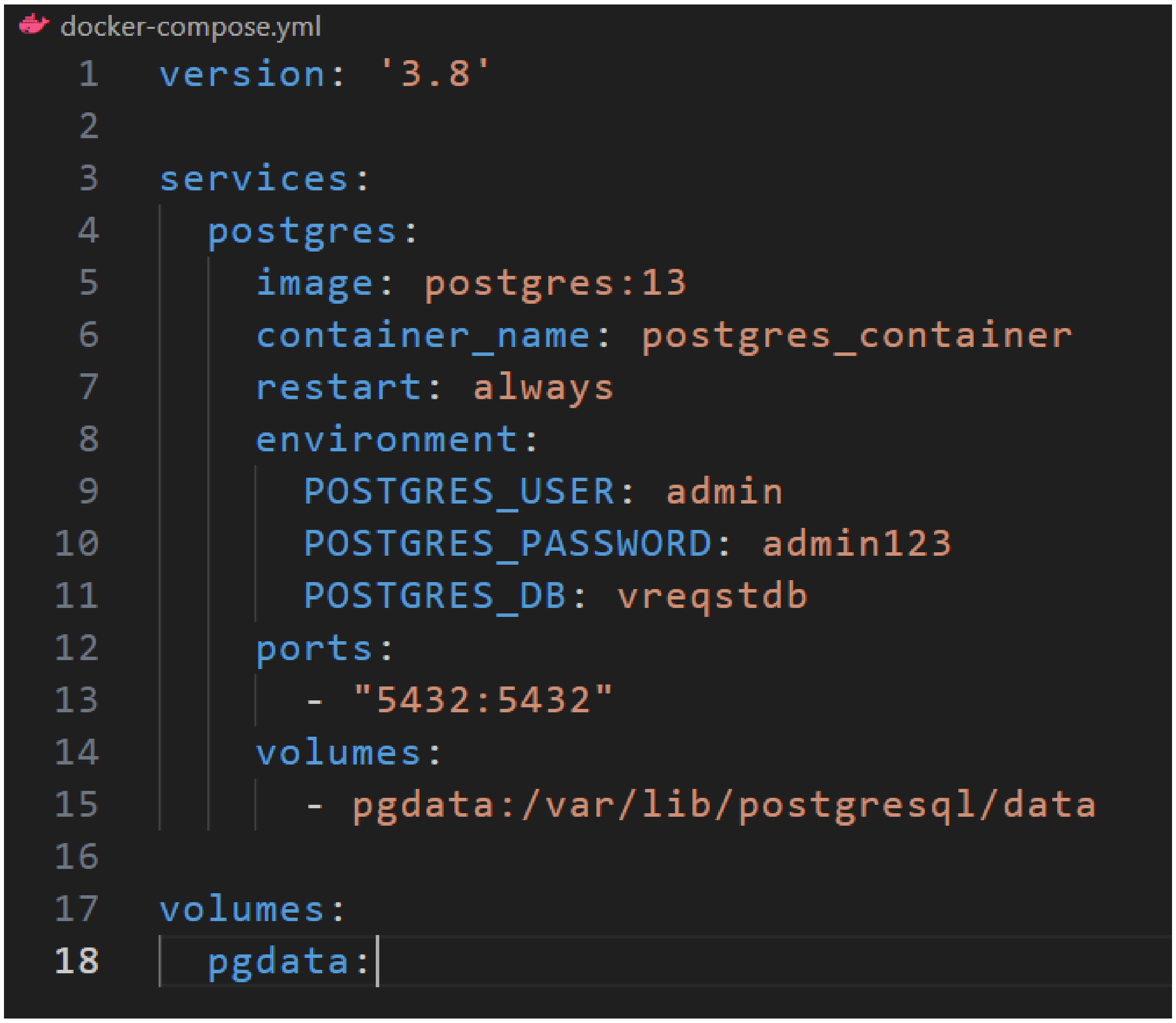
To create a Docker container with PostgreSQL database installed, perform basic database operations, and document the process.

Environment Details:

- Docker Version: 3.8
- Operating System: Windows 10
- PostgreSQL Version: 13

Docker Compose Configuration:

The following **docker-compose.yml** file was created to set up the PostgreSQL container:

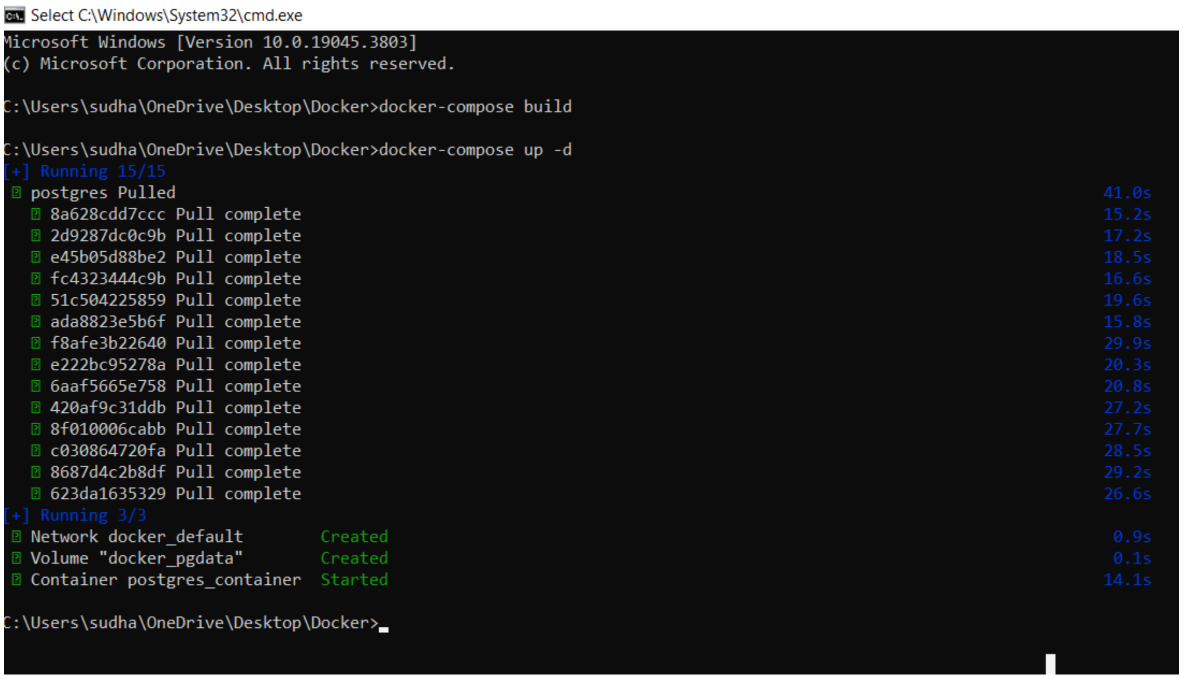


docker-compose.yml

```yaml
version: '3.8'

services:
  postgres:
    image: postgres:13
    container_name: postgres_container
    restart: always
    environment:
      POSTGRES_USER: admin
      POSTGRES_PASSWORD: admin123
      POSTGRES_DB: vreqstdb
    ports:
      - "5432:5432"
    volumes:
      - pgdata:/var/lib/postgresql/data

volumes:
  pgdata:
```

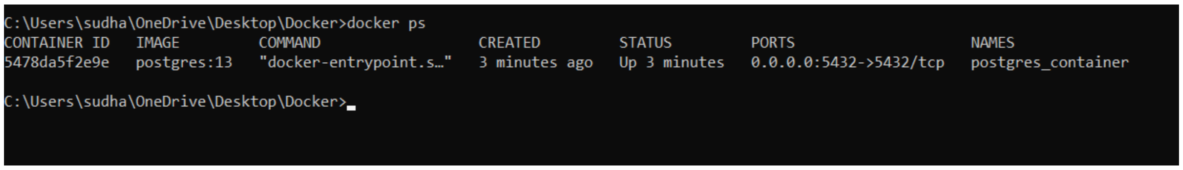
Commands Executed:

- Starting the container: **docker-compose up -d**

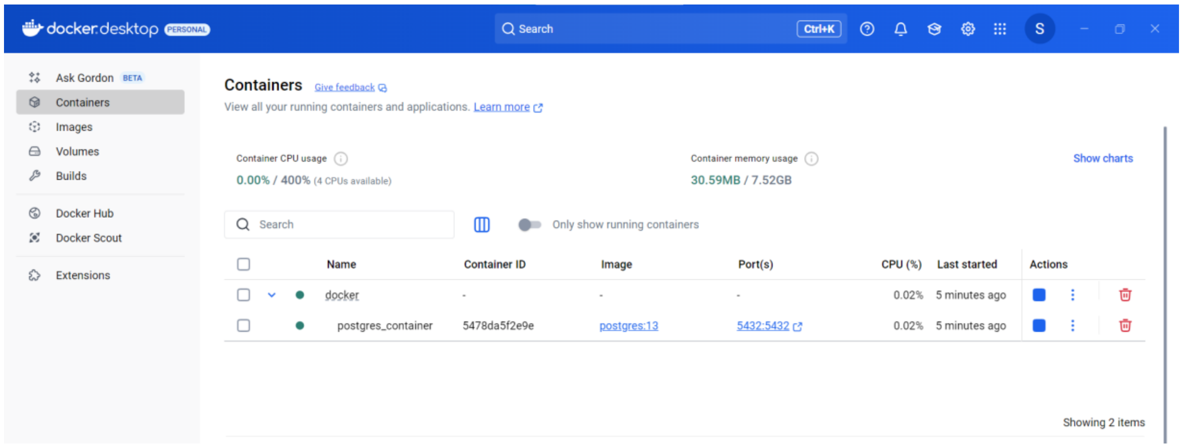


```
Select C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.3803]
(c) Microsoft Corporation. All rights reserved.

C:\Users\sudha\OneDrive\Desktop\Docker>docker-compose build

C:\Users\sudha\OneDrive\Desktop\Docker>docker-compose up -d
[+] Running 15/15
 ⍰ postgres Pulled                                                                    41.0s
   ⍰ 8a628cdd7ccc Pull complete                                                       15.2s
   ⍰ 2d9287dc0c9b Pull complete                                                       17.2s
   ⍰ e45b05d88be2 Pull complete                                                       18.5s
   ⍰ fc4323444c9b Pull complete                                                       16.6s
   ⍰ 51c504225859 Pull complete                                                       19.6s
   ⍰ ada8823e5b6f Pull complete                                                       15.8s
   ⍰ f8afe3b22640 Pull complete                                                       29.9s
   ⍰ e222bc95278a Pull complete                                                       20.3s
   ⍰ 6aaf5665e758 Pull complete                                                       20.8s
   ⍰ 420af9c31ddb Pull complete                                                       27.2s
   ⍰ 8f010006cabb Pull complete                                                       27.7s
   ⍰ c030864720fa Pull complete                                                       28.5s
   ⍰ 8687d4c2b8df Pull complete                                                       29.2s
   ⍰ 623da1635329 Pull complete                                                       26.6s
[+] Running 3/3
 ⍰ Network docker_default        Created                                               0.9s
 ⍰ Volume "docker_pgdata"        Created                                               0.1s
 ⍰ Container postgres_container  Started                                              14.1s

C:\Users\sudha\OneDrive\Desktop\Docker>
```

- To check running containers: **docker ps**



```
C:\Users\sudha\OneDrive\Desktop\Docker>docker ps
CONTAINER ID   IMAGE         COMMAND                  CREATED         STATUS         PORTS                    NAMES
5478da5f2e9e   postgres:13   "docker-entrypoint.s…"   3 minutes ago   Up 3 minutes   0.0.0.0:5432->5432/tcp   postgres_container

C:\Users\sudha\OneDrive\Desktop\Docker>
```



Containers

View all your running containers and applications.

Container CPU usage: 0.00% / 400% (4 CPUs available)

Container memory usage: 30.59MB / 7.52GB

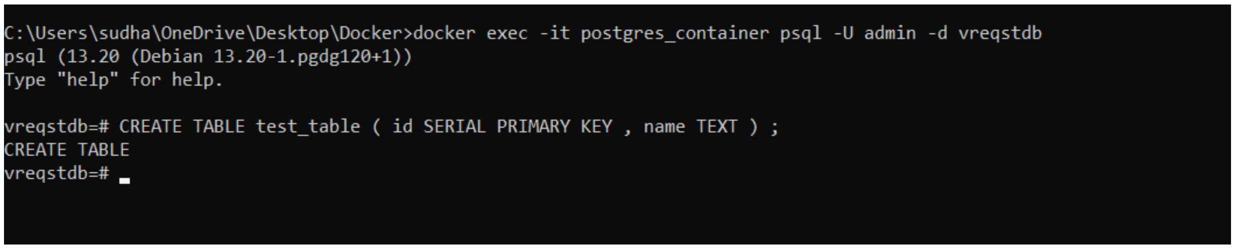
Name	Container ID	Image	Port(s)	CPU (%)	Last started
docker	-	-	-	0.02%	5 minutes ago
postgres_container	5478da5f2e9e	postgres:13	5432:5432	0.02%	5 minutes ago

Showing 2 items

- To connect to PostgreSQL container:
  docker exec -it postgres_container psql -U admin -d vreqstdb

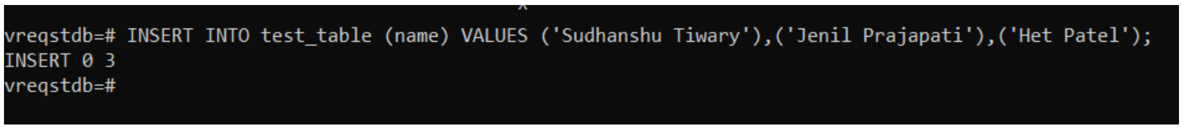
PostgreSQL CLI:

- Table creation



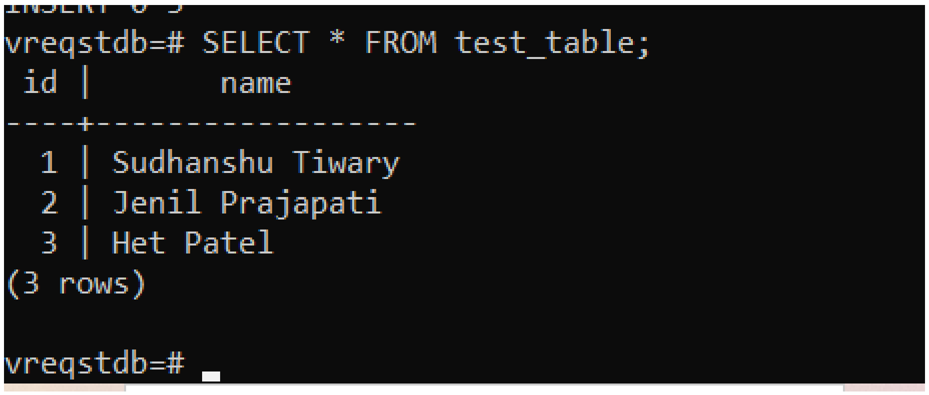
```
C:\Users\sudha\OneDrive\Desktop\Docker>docker exec -it postgres_container psql -U admin -d vreqstdb
psql (13.20 (Debian 13.20-1.pgdg120+1))
Type "help" for help.

vreqstdb=# CREATE TABLE test_table ( id SERIAL PRIMARY KEY , name TEXT ) ;
CREATE TABLE
vreqstdb=#
```

- Data Insertion



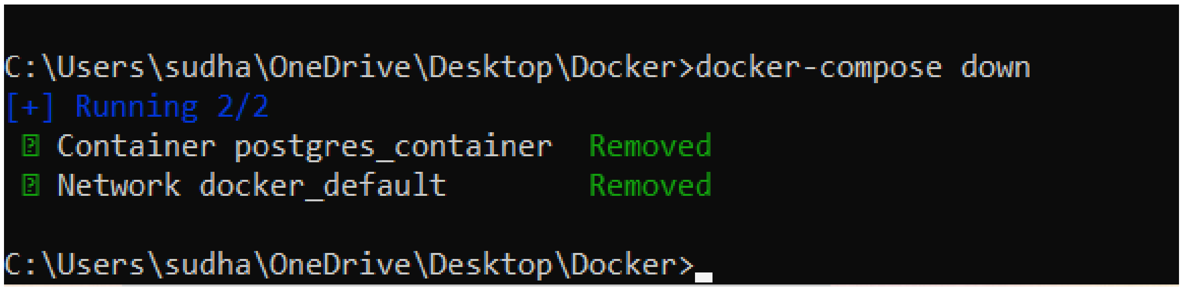
```
vreqstdb=# INSERT INTO test_table (name) VALUES ('Sudhanshu Tiwary'),('Jenil Prajapati'),('Het Patel');
INSERT 0 3
vreqstdb=#
```

- Select query showing inserted rows



```
vreqstdb=# SELECT * FROM test_table;
 id |       name
----+------------------
  1 | Sudhanshu Tiwary
  2 | Jenil Prajapati
  3 | Het Patel
(3 rows)

vreqstdb=#
```

- Stopping the container: **docker-compose down**



```
C:\Users\sudha\OneDrive\Desktop\Docker>docker-compose down
[+] Running 2/2
 ✔ Container postgres_container  Removed
 ✔ Network docker_default        Removed

C:\Users\sudha\OneDrive\Desktop\Docker>
```

Summary:

- A PostgreSQL container was created and configured successfully utilizing Docker Compose.
- Fundamental database activities such as creating tables, inserting data, and querying data were done.
- Environment variables were utilized in order to set up the PostgreSQL username, password, and database.
- A Docker volume was setup to retain data.
- Screenshots were taken in order to capture the running container and SQL command execution.

Conclusion: This task demonstrated the setup and basic use of a PostgreSQL database within a Docker container environment, providing hands-on experience with containerized database deployment and management.

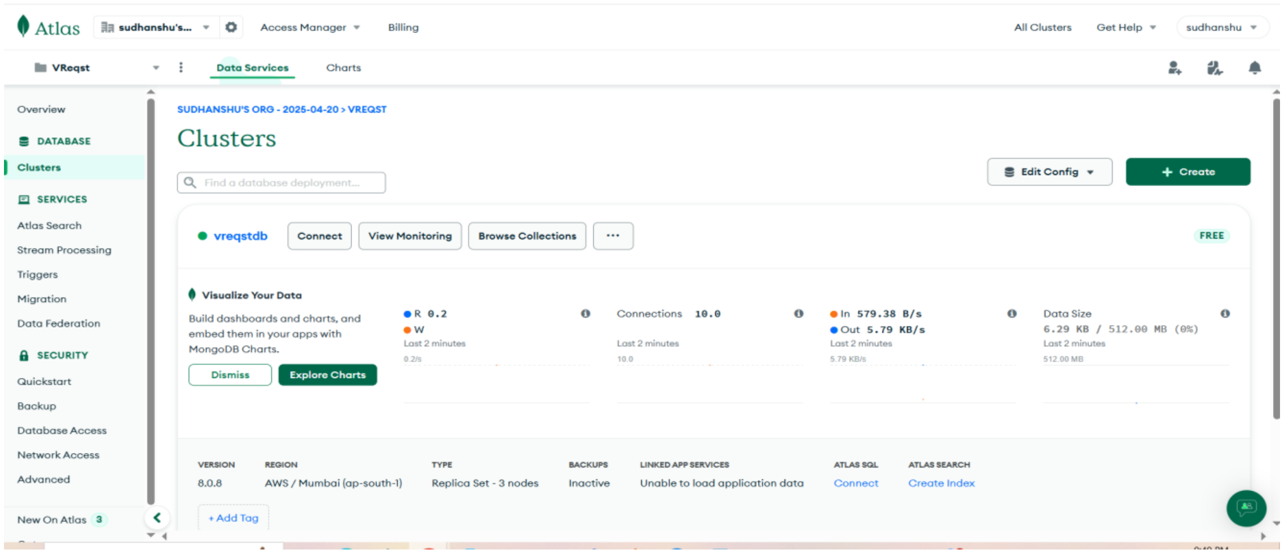
Q1 (B): Deploy VReqST – A requirement specification tool in a container.

Objective:

To deploy the VReqST (Virtual Reality Requirement Specification Tool) application using Docker, containerizing both the application and the associated MongoDB database service, ensuring proper database configuration and connectivity.

Setting up MongoDB Database:

- Create a new Project within MongoDB Atlas.
- Inside the project, create a Database Cluster (free-tier is sufficient for this assignment).



R 0.2
W
Last 2 minutes

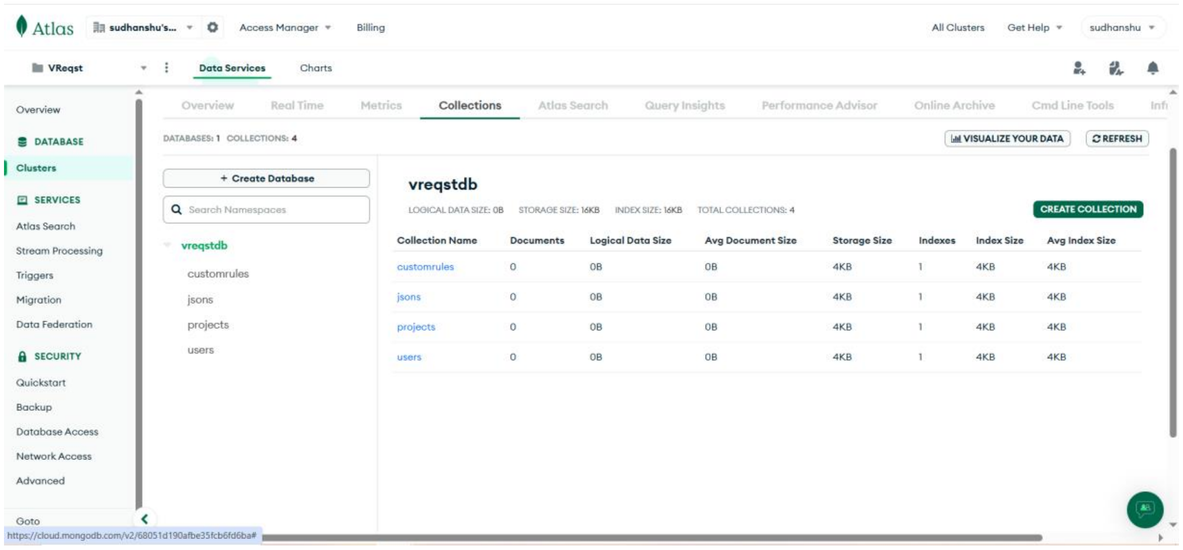
Connections 10.0
Last 2 minutes

In 579.38 B/s
Out 5.79 KB/s
Last 2 minutes

Data Size
6.29 KB / 512.00 MB (0%)
Last 2 minutes

VERSION	REGION	TYPE	BACKUPS	LINKED APP SERVICES	ATLAS SQL	ATLAS SEARCH
8.0.8	AWS / Mumbai (ap-south-1)	Replica Set - 3 nodes	Inactive	Unable to load application data	Connect	Create Index

- Once the cluster is created, define four collections within a new database using the following names:
    - customrules
    - jsons
    - projects
    - users



DATABASES: 1 COLLECTIONS: 4

vreqstdb

LOGICAL DATA SIZE: 0B STORAGE SIZE: 16KB INDEX SIZE: 16KB TOTAL COLLECTIONS: 4

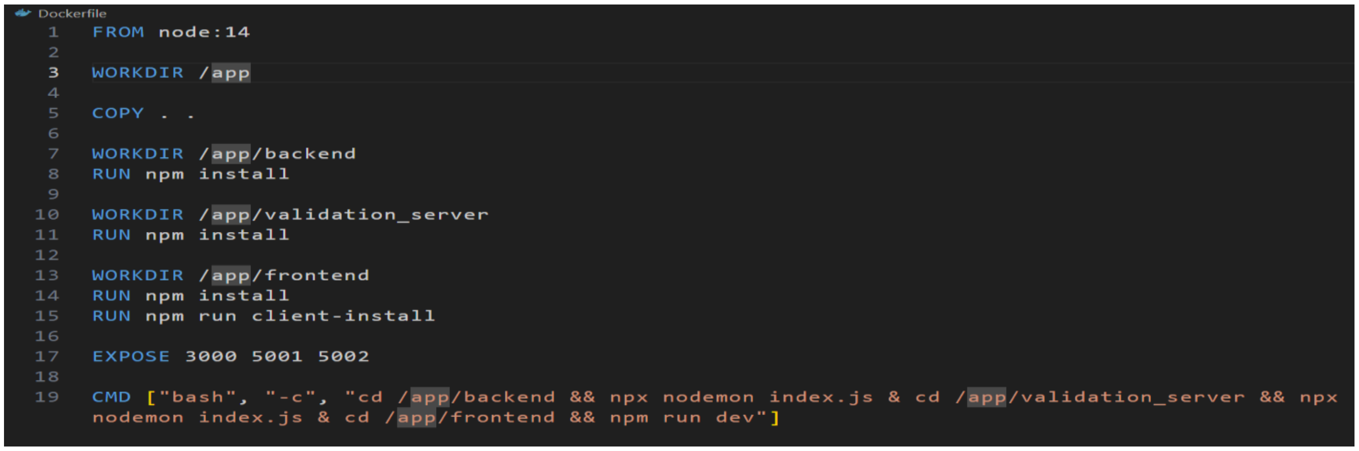
Collection Name	Documents	Logical Data Size	Avg Document Size	Storage Size	Indexes	Index Size	Avg Index Size
customrules	0	0B	0B	4KB	1	4KB	4KB
jsons	0	0B	0B	4KB	1	4KB	4KB
projects	0	0B	0B	4KB	1	4KB	4KB
users	0	0B	0B	4KB	1	4KB	4KB

https://cloud.mongodb.com/v2/68051d190afbe35fcb6fd6ba#

- Update the application’s server code to replace any local MongoDB connection string (e.g. mongodb://localhost:27017/vreqst) with the above cloud-hosted MongoDB Atlas connection string.
  Typically, this connection string is found in either:
  - backend/server.js
  - backend/app.js
  - or a configuration file such as backend/config.js or backend/config/db.js

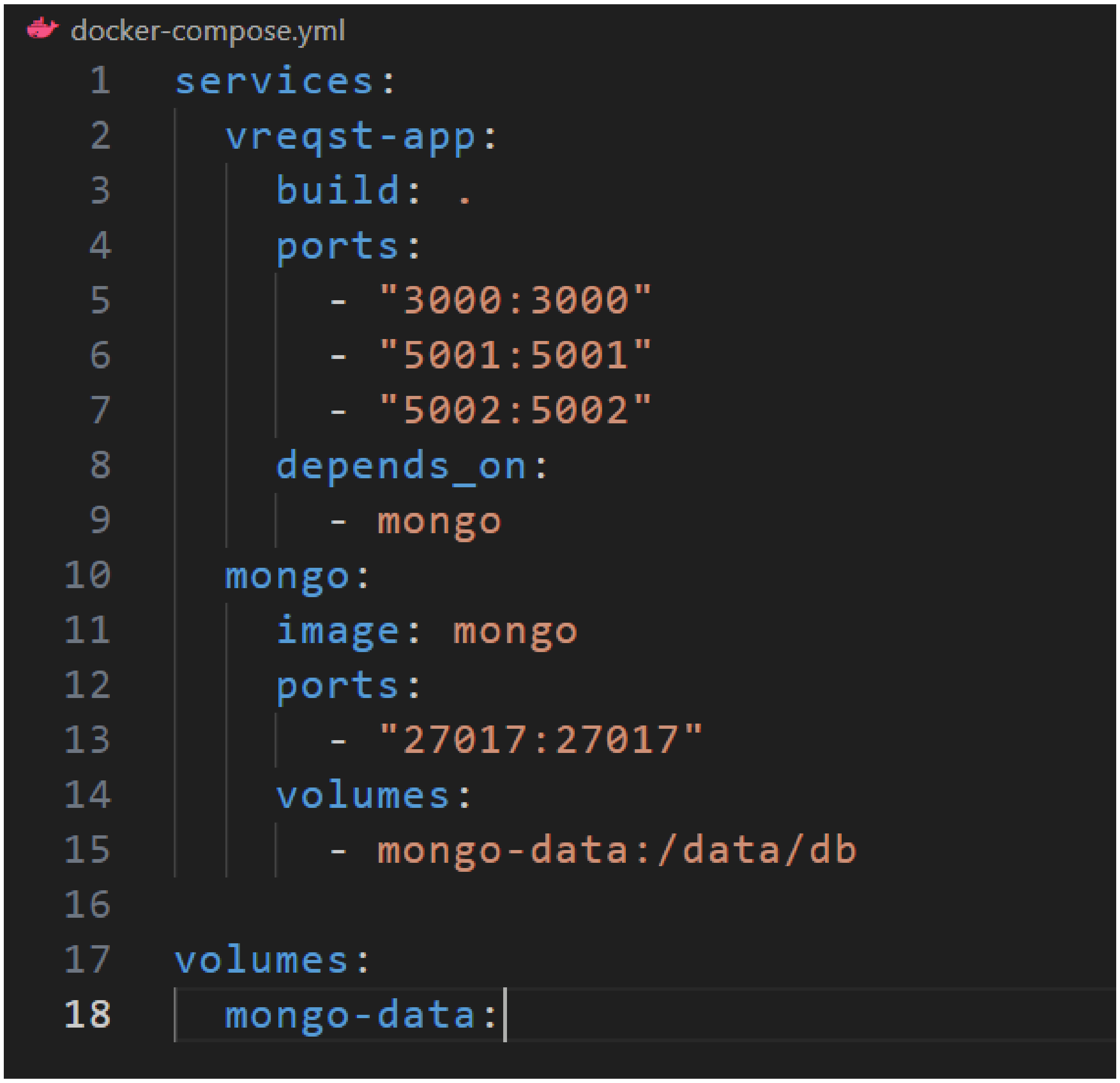
Writing the DockerFile and docker-compose.yml File:

Create a **Dockerfile** file in the project root directory with the following code:



```
Dockerfile
FROM node:14

WORKDIR /app

COPY . .

WORKDIR /app/backend
RUN npm install

WORKDIR /app/validation_server
RUN npm install

WORKDIR /app/frontend
RUN npm install
RUN npm run client-install

EXPOSE 3000 5001 5002

CMD ["bash", "-c", "cd /app/backend && npx nodemon index.js & cd /app/validation_server && npx nodemon index.js & cd /app/frontend && npm run dev"]
```

Create a **docker-compose.yml** file in the project root directory with the following configuration:



docker-compose.yml

```yaml
services:
  vreqst-app:
    build: .
    ports:
      - "3000:3000"
      - "5001:5001"
      - "5002:5002"
    depends_on:
      - mongo
  mongo:
    image: mongo
    ports:
      - "27017:27017"
    volumes:
      - mongo-data:/data/db

volumes:
  mongo-data:
```

Explanation:

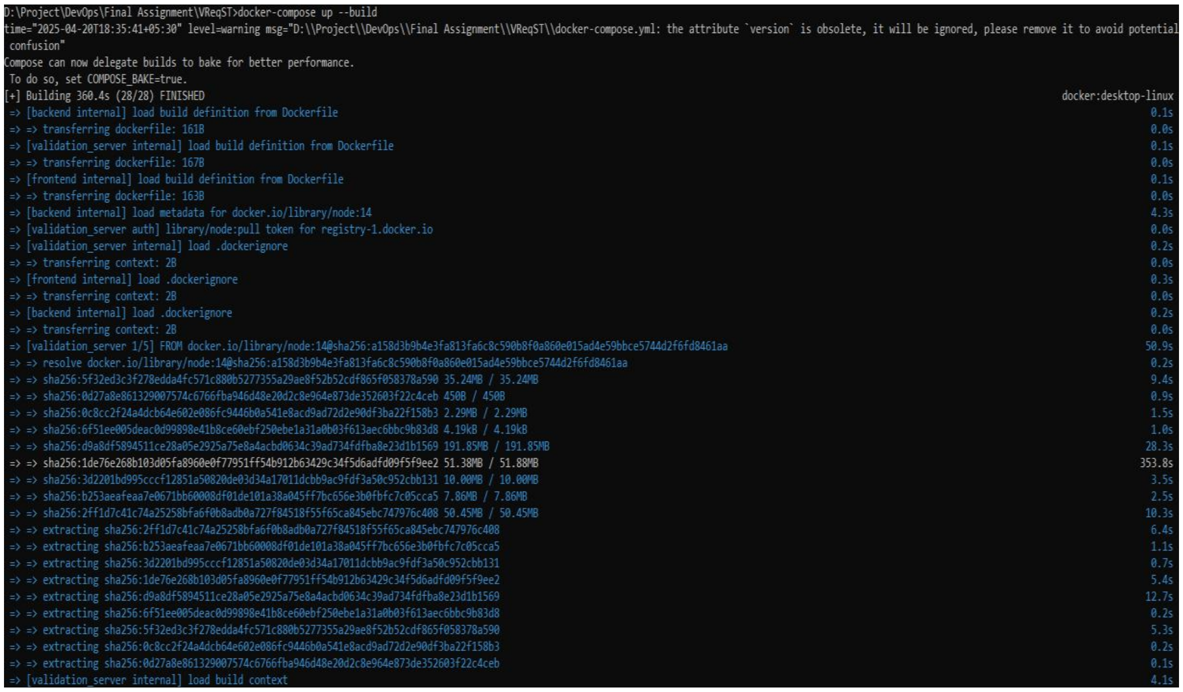
- The vreqst-app service builds the VReqST application from the Dockerfile in the current directory and maps necessary ports.
- The mongo service pulls the official MongoDB image and binds port 27017.
- mongo-data volume ensures data persistence for MongoDB.

Building and Running the Docker Containers:

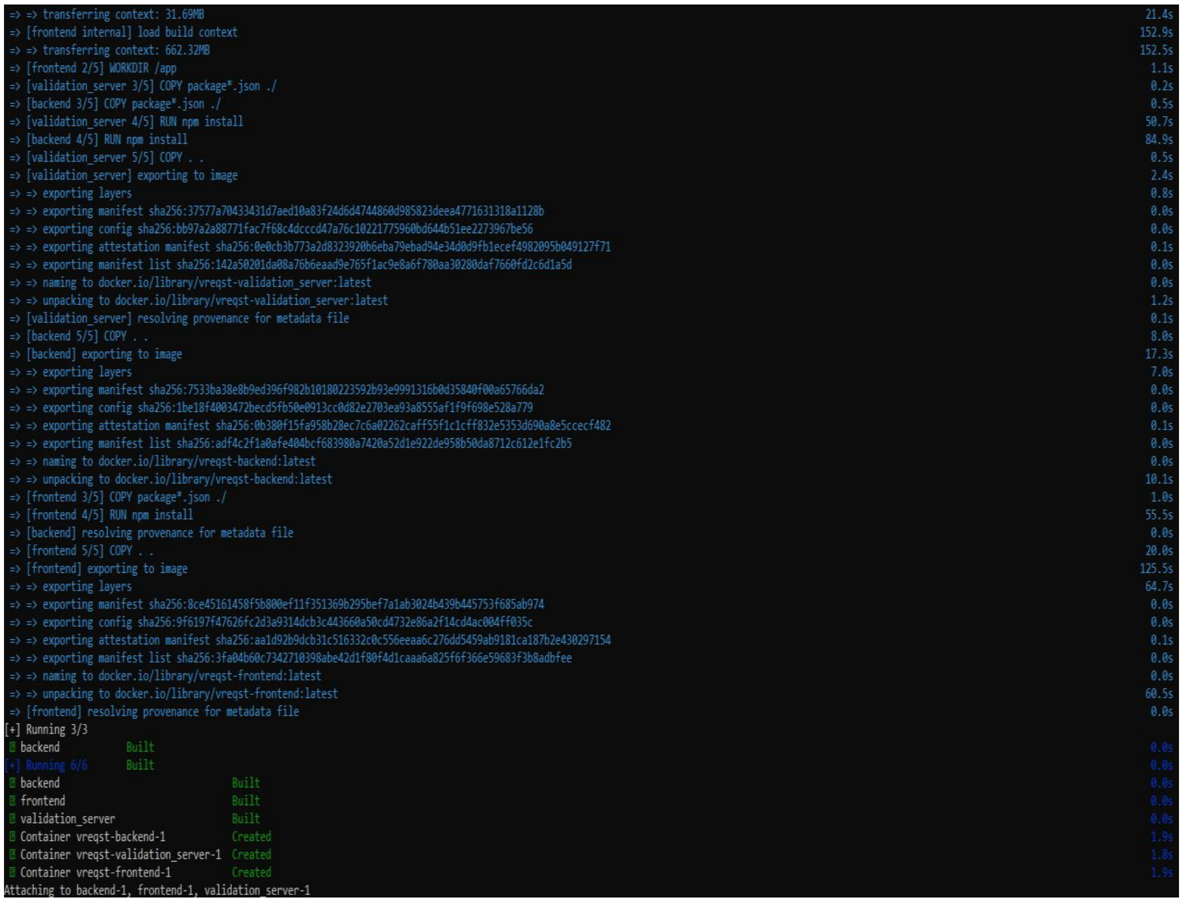
Building and running: **docker-compose up –build**

This command will:

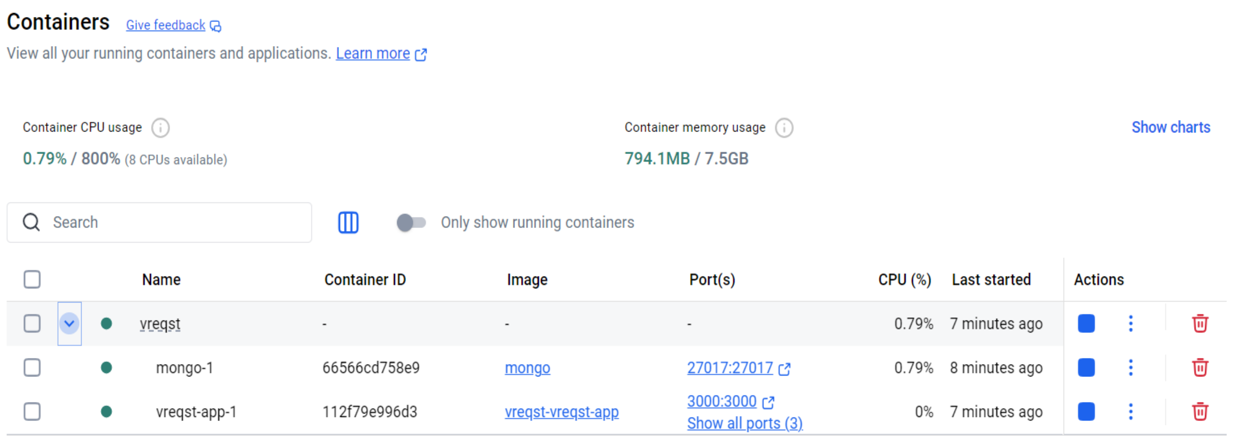
- Build the Docker images.
- Start both vreqst-app and mongo services.
- Map ports as defined in docker-compose.yml



```
D:\Project\DevOps\Final Assignment\VReqST>docker-compose up --build
time="2025-04-20T18:35:41+05:30" level=warning msg="D:\\Project\\DevOps\\Final Assignment\\VReqST\\docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential
 confusion"
Compose can now delegate builds to bake for better performance.
 To do so, set COMPOSE_BAKE=true.
[+] Building 360.4s (28/28) FINISHED                                                                                                   docker:desktop-linux
 => [backend internal] load build definition from Dockerfile                                                                                          0.1s
 => => transferring dockerfile: 161B                                                                                                                 0.0s
 => [validation_server internal] load build definition from Dockerfile                                                                                0.1s
 => => transferring dockerfile: 167B                                                                                                                 0.0s
 => [frontend internal] load build definition from Dockerfile                                                                                         0.1s
 => => transferring dockerfile: 163B                                                                                                                 0.0s
 => [backend internal] load metadata for docker.io/library/node:14                                                                                    4.3s
 => [validation_server auth] library/node:pull token for registry-1.docker.io                                                                         0.0s
 => [validation_server internal] load .dockerignore                                                                                                   0.2s
 => => transferring context: 2B                                                                                                                      0.0s
 => [frontend internal] load .dockerignore                                                                                                            0.3s
 => => transferring context: 2B                                                                                                                      0.0s
 => [backend internal] load .dockerignore                                                                                                             0.2s
 => => transferring context: 2B                                                                                                                      0.0s
 => [validation_server 1/5] FROM docker.io/library/node:14@sha256:a158d3b9b4e3fa813fa6c8c590b8f0a860e015ad4e59bbce5744d2f6fd8461aa                    50.9s
 => => resolve docker.io/library/node:14@sha256:a158d3b9b4e3fa813fa6c8c590b8f0a860e015ad4e59bbce5744d2f6fd8461aa                                      0.2s
 => => sha256:5f32ed3c3f278edda4fc571c880b5277355a29ae8f52b52cdf865f058378a590 35.24MB / 35.24MB                                                     9.4s
 => => sha256:0d27a8e861329007574c6766fba946d48e20d2c8e964e873de352603f22c4ceb 450B / 450B                                                           0.9s
 => => sha256:0c8cc2f24a4dcb64e602e086fc9446b0a541e8acd9ad72d2e90df3ba22f158b3 2.29MB / 2.29MB                                                       1.5s
 => => sha256:6f51ee005deac0d99898e41b8ce60ebf250ebe1a31a0b03f613aec6bbc9b83d8 4.19kB / 4.19kB                                                       1.0s
 => => sha256:d9a8df5894511ce28a05e2925a75e8a4acbd0634c39ad734fdfba8e23d1b1569 191.85MB / 191.85MB                                                  28.3s
 => => sha256:1de76e268b103d05fa8960e0f77951ff54b912b63429c34f5d6adfd09f5f9ee2 51.38MB / 51.88MB                                                   353.8s
 => => sha256:3d2201bd995cccf12851a50820de03d34a17011dcbb9ac9fdf3a50c952cbb131 10.00MB / 10.00MB                                                     3.5s
 => => sha256:b253aeafeaa7e0671bb60008df01de101a38a045ff7bc656e3b0fbfc7c05cca5 7.86MB / 7.86MB                                                       2.5s
 => => sha256:2ff1d7c41c74a25258bfa6f0b8adb0a727f84518f55f65ca845ebc747976c408 50.45MB / 50.45MB                                                    10.3s
 => => extracting sha256:2ff1d7c41c74a25258bfa6f0b8adb0a727f84518f55f65ca845ebc747976c408                                                            6.4s
 => => extracting sha256:b253aeafeaa7e0671bb60008df01de101a38a045ff7bc656e3b0fbfc7c05cca5                                                            1.1s
 => => extracting sha256:3d2201bd995cccf12851a50820de03d34a17011dcbb9ac9fdf3a50c952cbb131                                                            0.7s
 => => extracting sha256:1de76e268b103d05fa8960e0f77951ff54b912b63429c34f5d6adfd09f5f9ee2                                                            5.4s
 => => extracting sha256:d9a8df5894511ce28a05e2925a75e8a4acbd0634c39ad734fdfba8e23d1b1569                                                           12.7s
 => => extracting sha256:6f51ee005deac0d99898e41b8ce60ebf250ebe1a31a0b03f613aec6bbc9b83d8                                                            0.2s
 => => extracting sha256:5f32ed3c3f278edda4fc571c880b5277355a29ae8f52b52cdf865f058378a590                                                            5.3s
 => => extracting sha256:0c8cc2f24a4dcb64e602e086fc9446b0a541e8acd9ad72d2e90df3ba22f158b3                                                            0.2s
 => => extracting sha256:0d27a8e861329007574c6766fba946d48e20d2c8e964e873de352603f22c4ceb                                                            0.1s
 => [validation_server internal] load build context                                                                                                  4.1s
```



```
 => => transferring context: 31.69MB                                                                                                                21.4s
 => [frontend internal] load build context                                                                                                         152.9s
 => => transferring context: 662.32MB                                                                                                              152.5s
 => [frontend 2/5] WORKDIR /app                                                                                                                      1.1s
 => [validation_server 3/5] COPY package*.json ./                                                                                                    0.2s
 => [backend 3/5] COPY package*.json ./                                                                                                              0.5s
 => [validation_server 4/5] RUN npm install                                                                                                         50.7s
 => [backend 4/5] RUN npm install                                                                                                                   84.9s
 => [validation_server 5/5] COPY . .                                                                                                                 0.5s
 => [validation_server] exporting to image                                                                                                           2.4s
 => => exporting layers                                                                                                                              0.8s
 => => exporting manifest sha256:37577a70433431d7aed10a83f24d6d4744860d985823deea4771631318a1128b                                                    0.0s
 => => exporting config sha256:bb97a2a88771fac7f68c4dcccd47a76c10221775960bd644b51ee2273967be56                                                      0.0s
 => => exporting attestation manifest sha256:0e0cb3b773a2d8323920b6eba79ebad94e34d0d9fb1ecef4982095b049127f71                                        0.1s
 => => exporting manifest list sha256:142a50201da08a76b6eaad9e765f1ac9e8a6f780aa30280daf7660fd2c6d1a5d                                               0.0s
 => => naming to docker.io/library/vreqst-validation_server:latest                                                                                   0.0s
 => => unpacking to docker.io/library/vreqst-validation_server:latest                                                                                1.2s
 => [validation_server] resolving provenance for metadata file                                                                                       0.1s
 => [backend 5/5] COPY . .                                                                                                                           8.0s
 => [backend] exporting to image                                                                                                                    17.3s
 => => exporting layers                                                                                                                              7.0s
 => => exporting manifest sha256:7533ba38e8b9ed396f982b10180223592b93e9991316b0d35840f00a65766da2                                                    0.0s
 => => exporting config sha256:1be18f4003472becd5fb50e0913cc0d82e2703ea93a8555af1f9f698e528a779                                                      0.0s
 => => exporting attestation manifest sha256:0b380f15fa958b28ec7c6a02262caff55f1c1cff832e5353d690a8e5ccecf482                                        0.1s
 => => exporting manifest list sha256:adf4c2f1a0afe404bcf683980a7420a52d1e922de958b50da8712c612e1fc2b5                                               0.0s
 => => naming to docker.io/library/vreqst-backend:latest                                                                                             0.0s
 => => unpacking to docker.io/library/vreqst-backend:latest                                                                                         10.1s
 => [frontend 3/5] COPY package*.json ./                                                                                                             1.0s
 => [frontend 4/5] RUN npm install                                                                                                                  55.5s
 => [backend] resolving provenance for metadata file                                                                                                 0.0s
 => [frontend 5/5] COPY . .                                                                                                                         20.0s
 => [frontend] exporting to image                                                                                                                  125.5s
 => => exporting layers                                                                                                                             64.7s
 => => exporting manifest sha256:8ce45161458f5b800ef11f351369b295bef7a1ab3024b439b445753f685ab974                                                    0.0s
 => => exporting config sha256:9f6197f47626fc2d3a9314dcb3c443660a50cd4732e86a2f14cd4ac004ff035c                                                      0.0s
 => => exporting attestation manifest sha256:aa1d92b9dcb31c516332c0c556eeaa6c276dd5459ab9181ca187b2e430297154                                        0.1s
 => => exporting manifest list sha256:3fa04b60c7342710398abe42d1f80f4d1caaa6a825f6f366e59683f3b8adbfee                                               0.0s
 => => naming to docker.io/library/vreqst-frontend:latest                                                                                            0.0s
 => => unpacking to docker.io/library/vreqst-frontend:latest                                                                                        60.5s
 => [frontend] resolving provenance for metadata file                                                                                                0.0s
[+] Running 3/3
 □ backend           Built                                                                                                                           0.0s
[+] Running 6/6      Built                                                                                                                           0.0s
 □ backend                          Built                                                                                                            0.0s
 □ frontend                         Built                                                                                                            0.0s
 □ validation_server                Built                                                                                                            0.0s
 □ Container vreqst-backend-1           Created                                                                                                      1.9s
 □ Container vreqst-validation_server-1 Created                                                                                                      1.8s
 □ Container vreqst-frontend-1          Created                                                                                                      1.9s
Attaching to backend-1, frontend-1, validation_server-1
```



Containers Give feedback

View all your running containers and applications. Learn more

Container CPU usage
0.79% / 800% (8 CPUs available)

Container memory usage
794.1MB / 7.5GB

Show charts

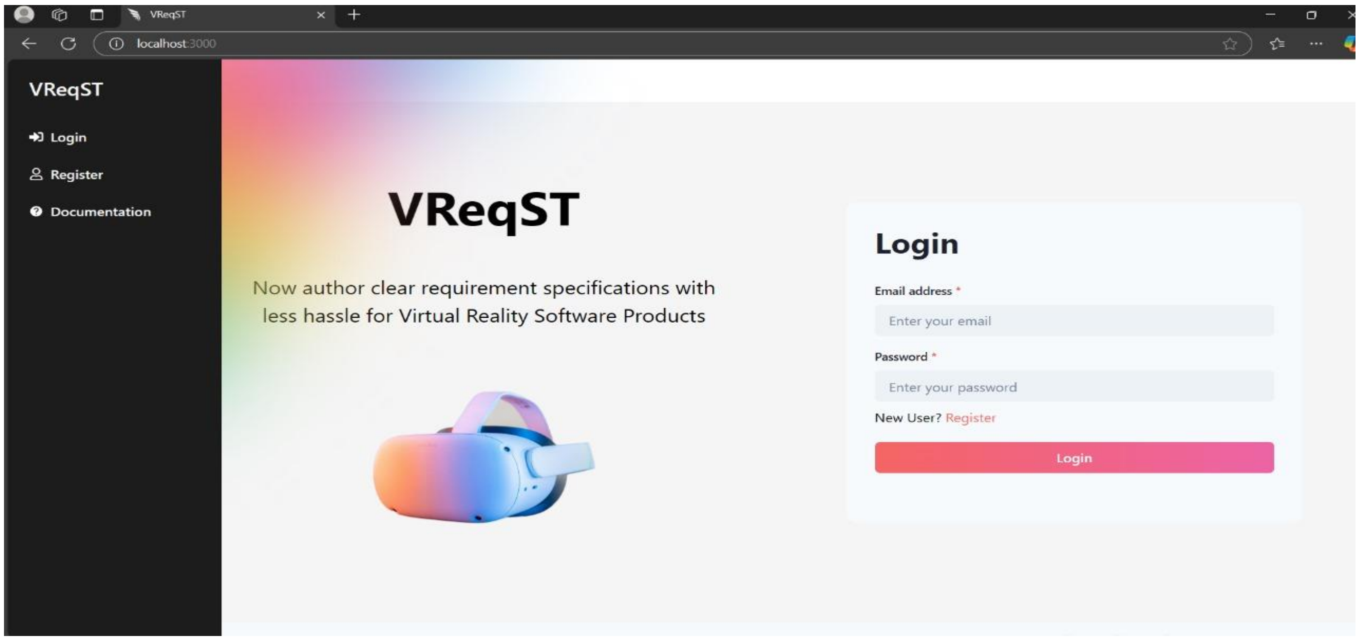
Only show running containers

Name	Container ID	Image	Port(s)	CPU (%)	Last started
vreqst	-	-	-	0.79%	7 minutes ago
mongo-1	66566cd758e9	mongo	27017:27017	0.79%	8 minutes ago
vreqst-app-1	112f79e996d3	vreqst-vreqst-app	3000:3000 Show all ports (3)	0%	7 minutes ago

Verifying Application and Database

Application Access:
Once the containers are running, access the application through:

- http://localhost:3000



MongoDB Access assumption :
If connecting to a locally running Mongo container, MongoDB would be accessible on mongodb://localhost:27017.
However, as per our configuration, we are using **MongoDB Atlas**, so no local connection is necessary after linking the Atlas connection string in the server code.

Outcome:

At the end of this task:

- The VReqST application runs inside a Docker container.
- It is connected to a cloud-hosted MongoDB Atlas database instance.
- Application services are accessible via defined ports.