CA6C1 – DevOps - National Institute of Technology, Trichy

Assignment 2 – Setting Up DOCKER - Workshop

Roll No. 205224005

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**

```
D:\Project\DevOps\Final Assignment\docker-postgres-db>docker-compose up -d
time="2025-04-20T12:09:55+05:30" level=warning msg="D:\\Project\\DevOps\\Final Assignment\\docker-postgres-db\\docker-compose.yml:
id potential confusion"
[+] Running 14/14
 postgres Pulled                                                                        17.7s
   f8afe3b22640 Pull complete                                                           10.4s
   420af9c31ddb Pull complete                                                           10.4s
   e45b05d88be2 Pull complete                                                            2.8s
   ada8823e5b6f Pull complete                                                            1.2s
   2d9287dc0c9b Pull complete                                                            2.3s
   623da1635329 Pull complete                                                           10.3s
   8f010006cabb Pull complete                                                            1.0s
   fc4323444c9b Pull complete                                                            2.2s
   51c504225859 Pull complete                                                            3.0s
   c030864720fa Pull complete                                                            1.1s
   e222bc95278a Pull complete                                                            1.2s
   6aaf5665e758 Pull complete                                                            1.2s
   8687d4c2b8df Pull complete                                                            1.2s
[+] Running 3/3
 Network docker-postgres-db_default  Created                                             0.0s
 Volume "docker-postgres-db_pgdata"  Created                                             0.0s
 Container postgres_container        Started                                             1.6s

D:\Project\DevOps\Final Assignment\docker-postgres-db>
```

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

```
D:\Project\DevOps\Final Assignment\docker-postgres-db>docker ps
CONTAINER ID   IMAGE         COMMAND                  CREATED         STATUS         PORTS                    NAMES
6a1936b0d9e3   postgres:13   "docker-entrypoint.s…"   8 minutes ago   Up 8 minutes   0.0.0.0:5432->5432/tcp   postgres_container
```

Containers Give feedback

View all your running containers and applications. Learn more

Container CPU usage: 0.02% / 800% (8 CPUs available)

Container memory usage: 20.28MB / 7.5GB

Show charts

Search | Only show running containers

Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
my_postgres_db	9853700b5cf8	postgres	5432:5432	0%	13 hours ago	
sleepy_perlman	449f9789ee6d	hello-world		0%	13 hours ago	
jovial_hermann	d204bd6e5d65	hello-world		0%	13 hours ago	
vreqst	-	-	-	0%	12 hours ago	
docker-postgres-db	-	-	-	0.03%	2 minutes ago	

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

PostgreSQL CLI:

- Table creation

```
D:\Project\DevOps\Final Assignment\docker-postgres-db>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 (
vreqstdb(#     id SERIAL PRIMARY KEY,
vreqstdb(#     name TEXT
vreqstdb(# );
CREATE TABLE
```

- Data Insertion

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

- Select query showing inserted rows

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

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

```
D:\Project\DevOps\Final Assignment\docker-postgres-db>docker-compose down
[+] Running 2/2
 ⠿ Container postgres_container        Removed
 ⠿ Network docker-postgres-db_default  Removed
```

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.

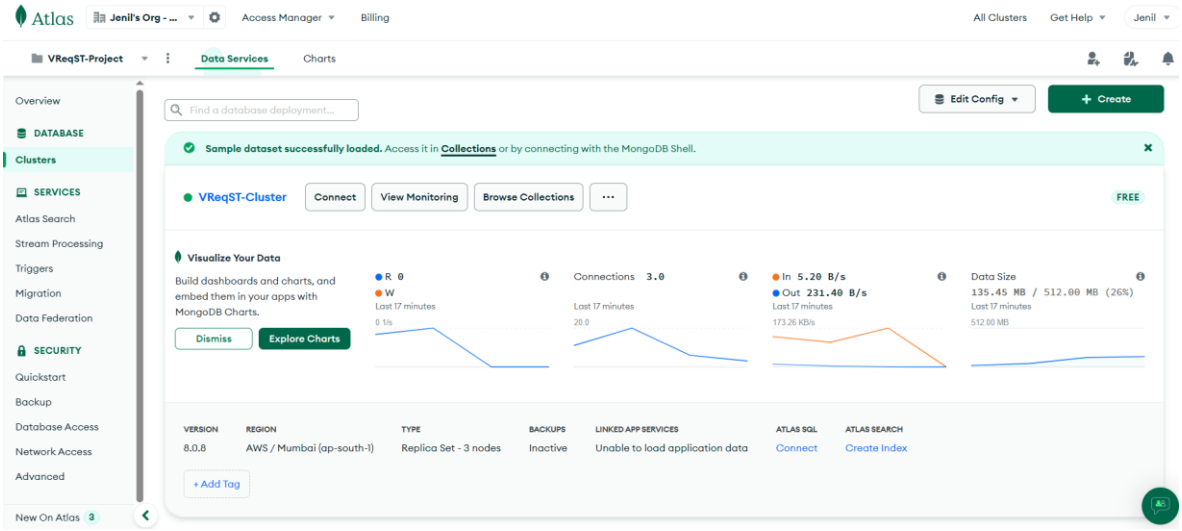
Q1 (B): Deploy VReqST – A requirement specification tool in a container.

Objective:

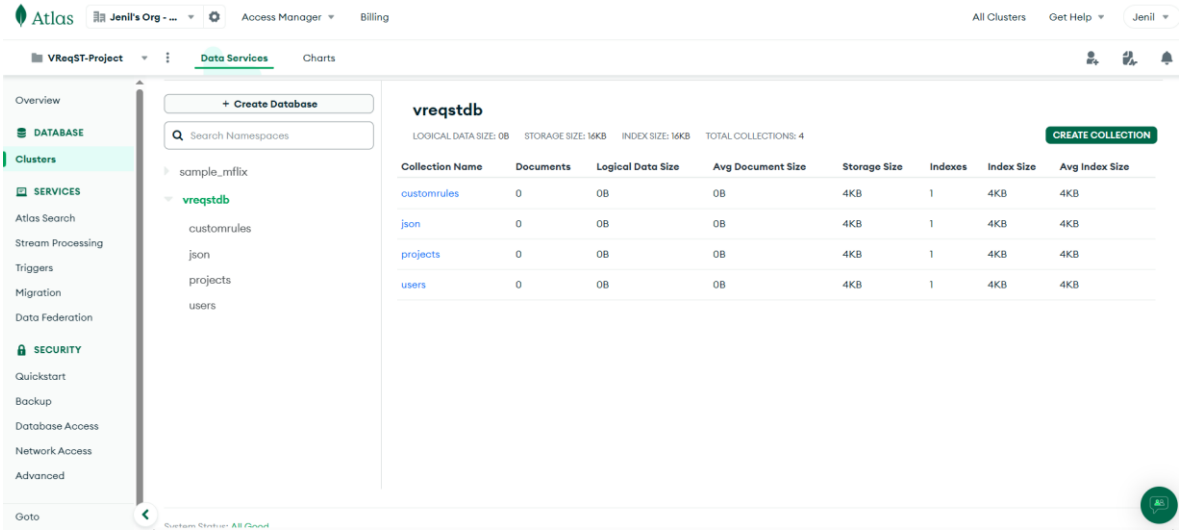
To deploy the VReqST (Virtual Reality Requirement Specification Tool) application using Docker, containerizing both the application and the associated MongoDB database service, ensuring proper database configuration and connectivity.

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).



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



- 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

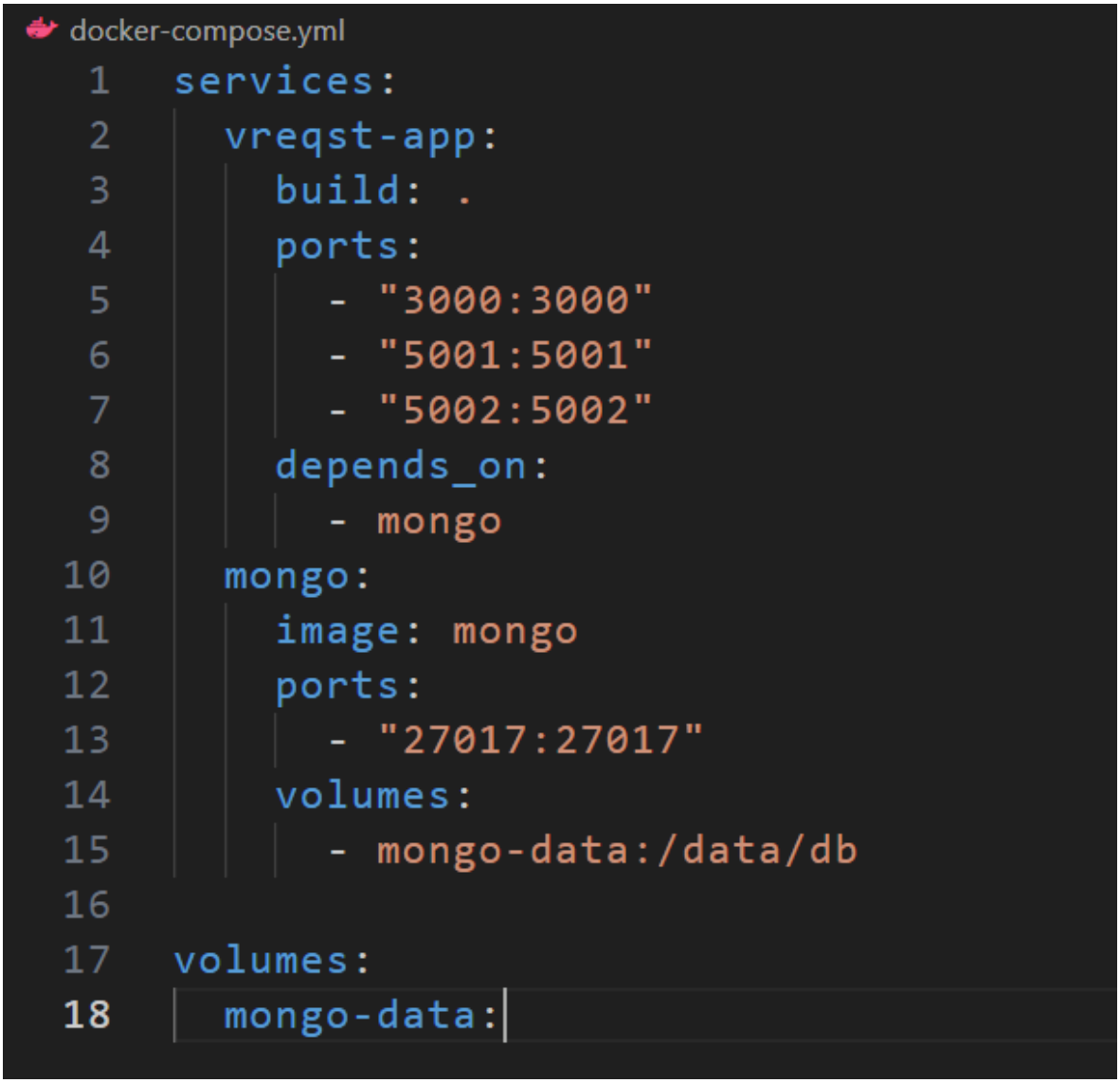
Writing the DockerFile and docker-compose.yml File:

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:



Explanation:

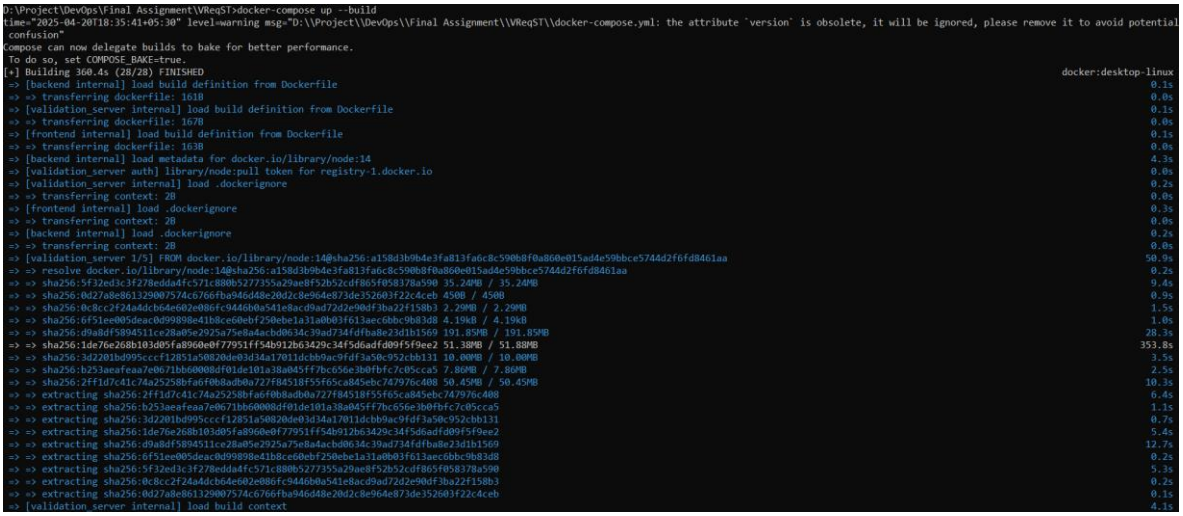
- The vreqst-app service builds the VReqST application from the Dockerfile in the current directory and maps necessary ports.
- The mongo service pulls the official MongoDB image and binds port 27017.
- mongo-data volume ensures data persistence for MongoDB.

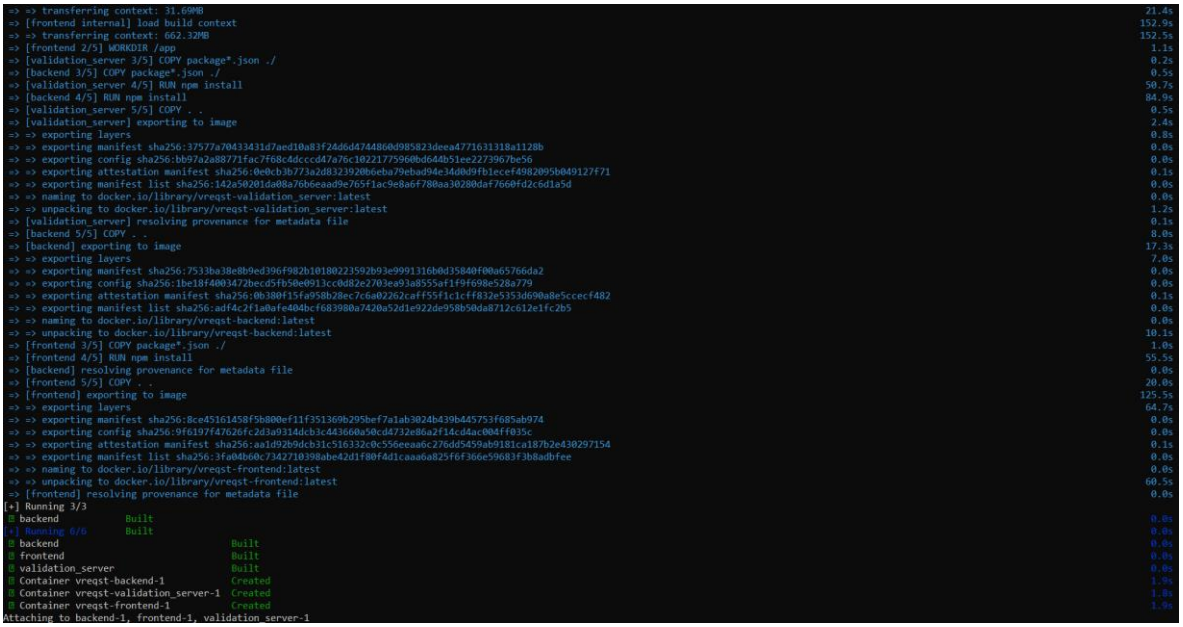
Building and Running the Docker Containers:

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

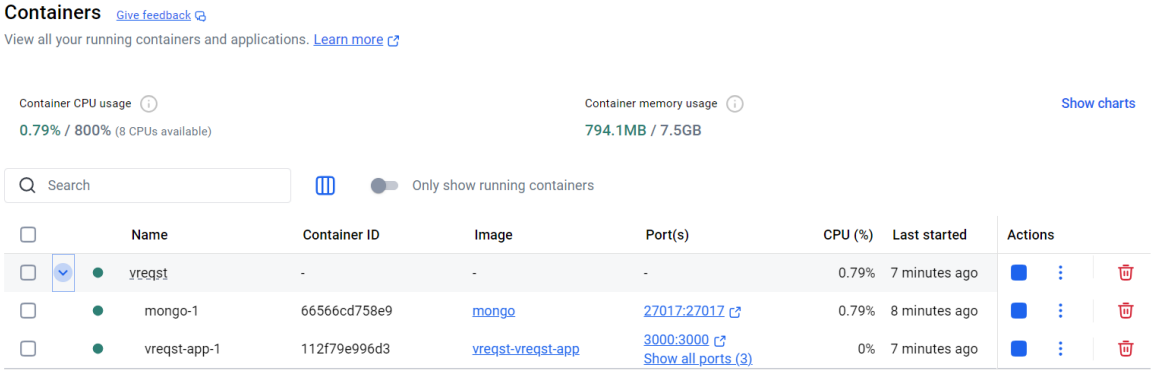
This command will:

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





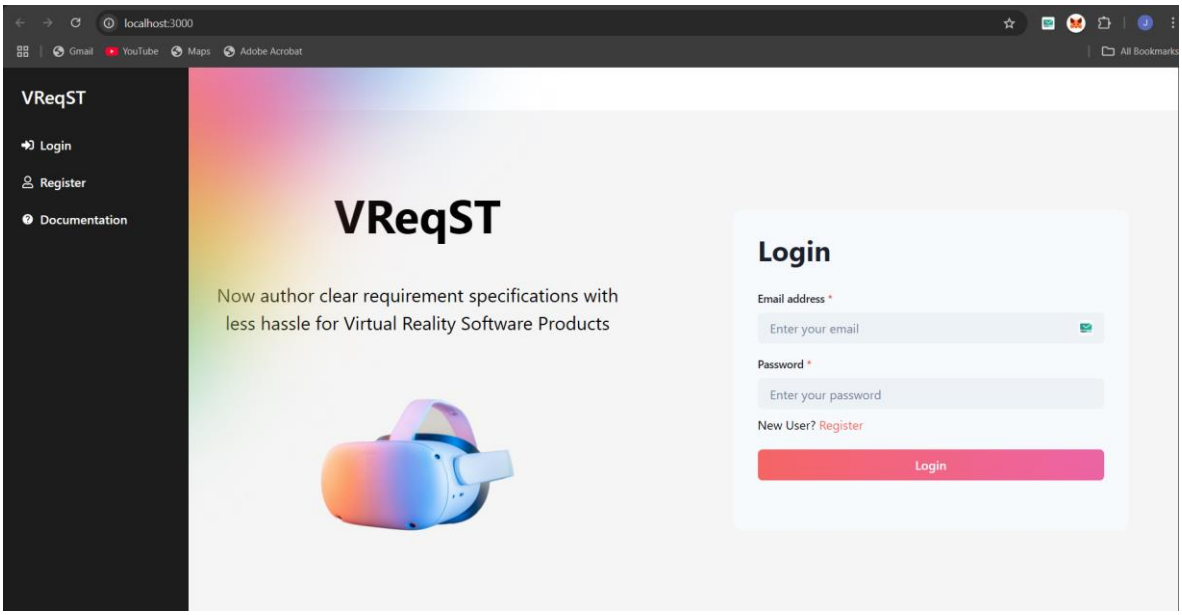
Running Container



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.