3. Software Requirement: (Include details about any software or tools used)

Windows 10

container tool Docker Desktop

Visual Studio Code

Windows Terminal

MongoDB Compass

4. Lab Tasks:

1. Download the container tool Docker in the host machine and check the architecture ARM/ADM and then install it according to the requirement.

2. To verify container tool Docker is running, run these commands

docker --version

docker-compose --version

and test using the command.

docker run hello-world

This command downloads the hello-world image and runs a test container to confirm that container tool Docker is installed and functioning properly.

3. We are tasked with setting up a multi-server environment using container tool Docker, which includes:

Two app servers

One database server

A separate container for logging activity

A simple login application will be deployed across the two app servers, utilizing the database for authentication. Activity logs will be stored in a separate container, and access will be demonstrated from both the host system and an external system.

4. To create a new project folder to store the container tool Docker setup and application files.

Changing the directory to where I want to create the project folder.

Create a new directory for the docker project using the mkdir command; name of it is multi-server-app.

Change the directory to the multi-server-app.

Create the subdirectories for the application code and logs.

Go to directory app-server1>>>Create a new container tool Docker file >>> Add the container tool Docker code there.

Same steps for the app-server2 but change the EXPOSE to 3002

Same steps for creating dockerfile in db-server but the code to the file will be different.

Go to the project root directory and create the docker-compose.yml

Add the code

Create the server.js in app-server1 and install the necessary extensions.

Install MonoDB and create a new database.

Add the path of the MongoDB in the System Environment.

MongoDB instance has successfully started.

In the logging initialize a new Node.js project.

In logging create a file named logger.js and and add the code.

Create a server.js file in directory logging and add the following code.

Create a file named container tool Dockerfile in the logging directory to containerize this logging service and add this code.

Once done, run the command ”docker-compose up –build ”, then the docker will ask the permission to run at the background and it will start.

Access the app servers from the web browser.

App Server 1: Access it at http://localhost:3001

App Server 2: Access it at http://localhost:3002

Logging Service: Access it at http://localhost:4000

4 docker servers running successfully.

5. To create a register and loggin applications, import necessary libraries. And connect them to Monogodb database

Change the server.js code and add the register.html and login.html files to spp-server1 and app-server2

Access the registration page at:

http://localhost:3001/register

Access the login page at:

http://localhost:3001/login

5.Observations:  
1. I have faced this error which was resolved by adding the new code and restarting the vs code again.

6. Results and Analysis:

The deployment of the multi-server application was completed successfully. All containers for the app servers, database server, and logging service were initiated without errors. The app servers responded correctly to HTTP requests, and logs confirmed successful database connections.

app-server1 and app-server2 are running on ports 3001 and 3002, respectively.

db-server is running on port 27017, and the logging service is active on port 4000.

7. Conclusion:

The lab successfully demonstrated the deployment and operation of a multi-server architecture using container tool Docker. Key takeaways include:

The use of containers facilitated easy management of application services, promoting scalability and isolation.

Understanding the communication between services in a networked environment is crucial for application performance.

The hands-on experience gained during this lab enhances my understanding of container tool Docker's capabilities and its role in modern application development.

8. References:

container tool Docker Documentation. (n.d.). container tool Docker overview. Retrieved from https://docs.docker.com/

MongoDB Documentation. (n.d.). MongoDB Atlas. Retrieved from

Chatgpt