5.1P: Containerisation of a simple web application using Docker

Introduction

Containerization has revolutionized software development by enabling applications to run consistently across various environments. This document demonstrates how to containerize a Node.js web application using Docker and Docker Compose. Docker Compose simplifies the management of multi-container applications, making it an ideal tool for both the application and its dependencies.

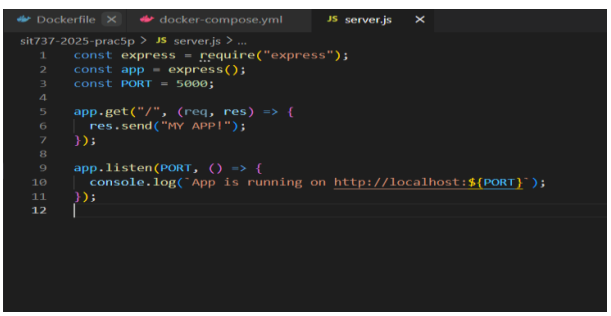
Prerequisites

1. GIT
2. Visual Studio Code
3. Node.js
4. Docker

Part I: Containerizing the Application

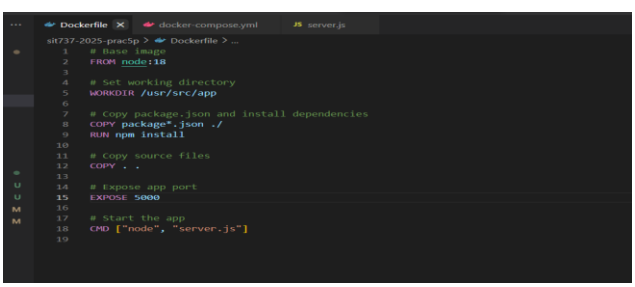
Step-1 Creating base folder and Node.js Environment

The steps include creating a new directory and initializing the project using npm.



Step-2 Creating Docker file

The Dockerfile defines the application's environment and the instructions needed to build the Docker image. This includes specifying the base image, copying the necessary files, installing dependencies, and setting up the necessary configuration for the application.



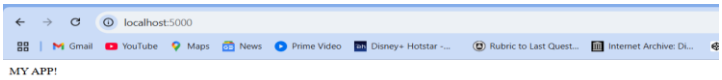
Step-3 Build Docker image

The Docker image is built based on the Dockerfile created in the previous step. The image will contain all the necessary dependencies and configurations required to run the application in a containerized environment.

Step-4 Create Compose file

The Docker Compose file is used to define and run multi-container Docker applications. In this step, we create a docker-compose.yml file.

Then we test the application

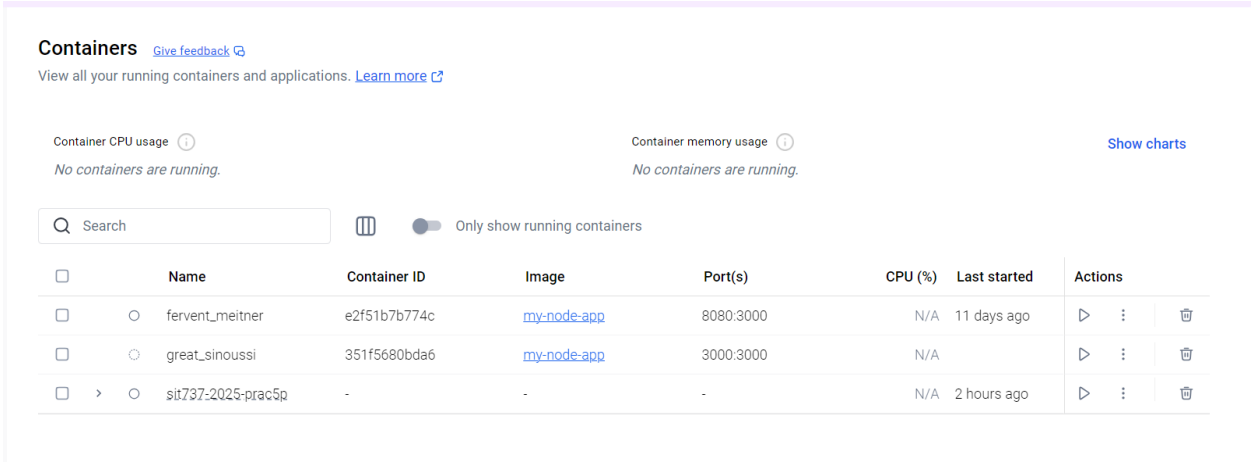


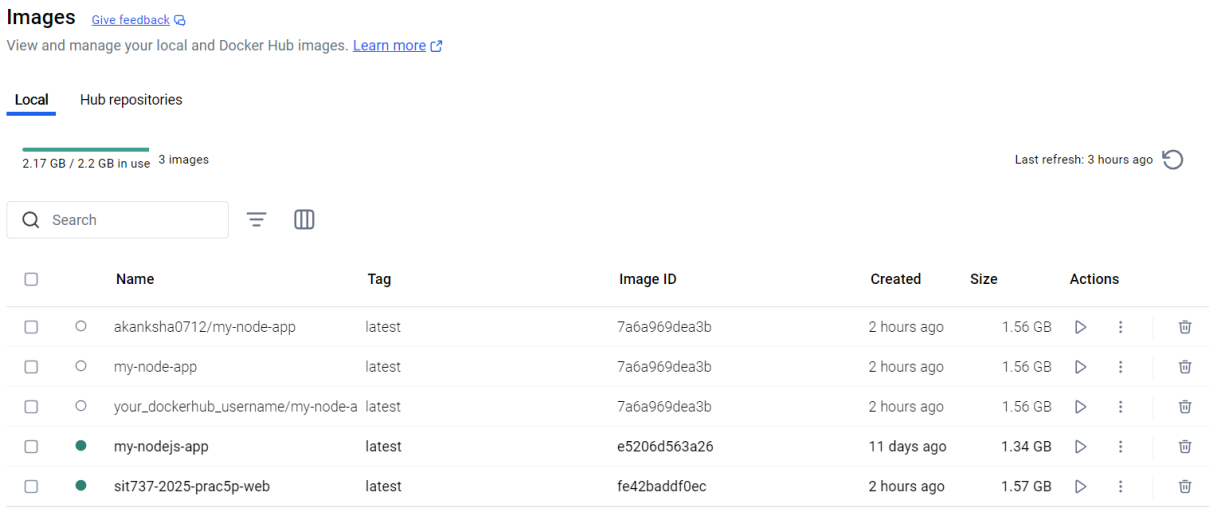
Step-5 Pushing file

After building the Docker image and creating the Compose file, the next step is to push the Docker image to a container

Commands used :

1. docker tag my-node-app akanksha0712/my-node-app
2. docker push akanksha0712/my-node-app





Part II: Implementing Container Health Checks

Health checks monitor the status of your application, ensuring it is functioning correctly.

In docker-compose.yml file, under the app service, we add the healthcheck configuration

```
PS C:\Users\WELCOME> cd C:\sit737\sit737-2025-prac5p
PS C:\sit737\sit737-2025-prac5p> docker ps
CONTAINER ID   IMAGE                    COMMAND                  CREATED          STATUS                             POR
TS                    NAMES
d6dac4a63885   sit737-2025-prac5p-web   "docker-entrypoint.s…"   20 seconds ago   Up 19 seconds (health: starting)   500
0/tcp, 0.0.0.0:3001->3000/tcp   sit737-2025-prac5p-web-1
PS C:\sit737\sit737-2025-prac5p>
```

Conclusion

Using docker and Docker compose we have successfully containerized Node.js application. Putting health checks in place guarantees that your application will continue to function dependably and be able to automatically fix any problems. This configuration offers a consistent and portable environment, enhancing the reliability and scalability of your application across various platforms.