

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.

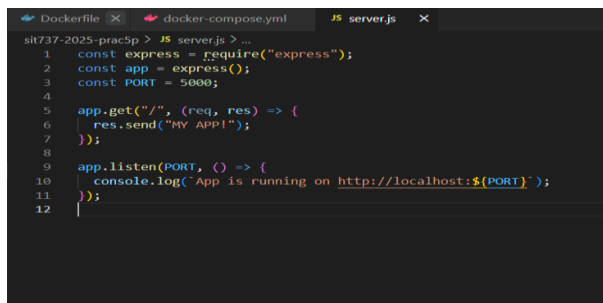
Prerequisites

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

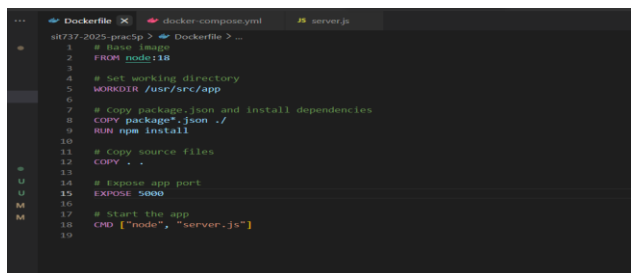
The steps include creating a new directory and initializing the project using npm.

A screenshot of a code editor with three tabs: 'Dockerfile', 'docker-compose.yml', and 'server.js'. The 'server.js' tab is active, showing a Node.js Express server. The code starts with 'const express = require("express");', followed by 'const app = express();' and 'const PORT = 5000;'. It then defines a route for '/' that sends 'MY APP!' and listens on the specified port, logging the URL. The code is as follows:

```
1 const express = require("express");
2 const app = express();
3 const PORT = 5000;
4
5 app.get("/", (req, res) => {
6   res.send("MY APP!");
7 });
8
9 app.listen(PORT, () => {
10   console.log(`App is running on http://localhost:${PORT}`);
11 });
12
```

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.

A screenshot of a code editor with three tabs: 'Dockerfile', 'docker-compose.yml', and 'server.js'. The 'Dockerfile' tab is active, showing instructions for building a Docker image. The instructions include setting the base image to 'node:18', setting the working directory to '/usr/src/app', copying package.json and installing dependencies with 'npm install', copying source files, exposing port 5000, and starting the application with 'CMD ["node", "server.js"]'. The code is as follows:

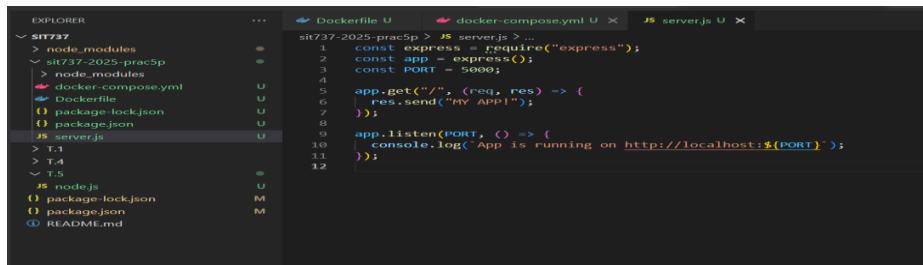
```
1 # Base image
2 FROM node:18
3
4 # Set working directory
5 WORKDIR /usr/src/app
6
7 # Copy package.json and install dependencies
8 COPY package.json ./
9 RUN npm install
10
11 # Copy source files
12 COPY . .
13
14 # Expose app port
15 EXPOSE 5000
16
17 # Start the app
18 CMD ["node", "server.js"]
19
```

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

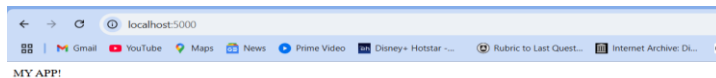
The Docker Compose file is used to define and run multi-container Docker applications. In this step, we create a docker-compose.yml file.



The screenshot shows a code editor with three tabs: 'Dockerfile U', 'docker-compose.yml U', and 'JS server.js U'. The 'Dockerfile U' tab is active, displaying the following content:

```
1  FROM node:14
2  WORKDIR /usr/src/app
3  COPY package.json package-lock.json ./
4  RUN npm install
5  COPY . .
6  EXPOSE 5000
7  CMD ["npm", "start"]
```

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`

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.

[Show charts](#)

Search



Only show running containers

<input type="checkbox"/>	Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
<input type="checkbox"/>	<input type="radio"/> fervent_meitner	e2f51b7b774c	my-node-app	8080:3000	N/A	11 days ago	
<input type="checkbox"/>	<input type="radio"/> great_sinoussi	351f5680bda6	my-node-app	3000:3000	N/A		
<input type="checkbox"/>	<input checked="" type="radio"/> sit737-2025-prac5p	-	-	-	N/A	2 hours ago	

Images [Give feedback](#)

View and manage your local and Docker Hub images. [Learn more](#)

Local Hub repositories

2.17 GB / 2.2 GB in use 3 Images

Last refresh: 3 hours ago

Search



<input type="checkbox"/>	Name	Tag	Image ID	Created	Size	Actions
<input type="checkbox"/>	<input type="radio"/> akanksha0712/my-node-app	latest	7a6a969dea3b	2 hours ago	1.56 GB	
<input type="checkbox"/>	<input type="radio"/> my-node-app	latest	7a6a969dea3b	2 hours ago	1.56 GB	
<input type="checkbox"/>	<input type="radio"/> your_dockerhub_username/my-node-a	latest	7a6a969dea3b	2 hours ago	1.56 GB	
<input type="checkbox"/>	<input checked="" type="radio"/> my-nodejs-app	latest	e5206d563a26	11 days ago	1.34 GB	
<input type="checkbox"/>	<input checked="" type="radio"/> sit737-2025-prac5p-web	latest	fe42badff0ec	2 hours ago	1.57 GB	

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                    PORTS
16dac4a63885   sit737-2025-prac5p-web             "docker-entrypoint.s...  20 seconds ago Up 19 seconds (health: starting)  5000/tcp, 0.0.0.0:3001->3000/tcp
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.