

MIT WORLD PEACE UNIVERSITY

Cloud Infrastructure and Security
Third Year B. Tech, Semester 6

INSTALLING AND RUNNING DOCKER ON WINDOWS

ASSIGNMENT 5

Prepared By

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A1, PA 10

March 28, 2024

Contents

1 Aim	1
2 Objectives	1
3 Theory	1
3.1 What is Docker?	1
3.2 Why Docker?	1
4 Setting up Docker on Windows	2
5 Platform	8
6 FAQs	8
7 Conclusion	8
References	9

1 Aim

To install and Run Docker on Windows or Ubuntu

2 Objectives

1. To learn how to install Docker on Windows
2. To understand the importance of Docker

3 Theory

3.1 What is Docker?

Docker is a platform for developing, shipping, and running applications in containers. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications. By taking advantage of Docker's methodologies for shipping, testing, and deploying code quickly, you can significantly reduce the delay between writing code and running it in production.

3.2 Why Docker?

1. **Rapid Deployment:** Docker containers can be built, deployed, and scaled quickly.
2. **Version Control:** Docker images are version-controlled and can be pushed to a remote repository.
3. **Isolation:** Docker containers are isolated from each other and from the host system.
4. **Portability:** Docker containers can run on any system that supports Docker.
5. **Resource Efficiency:** Docker containers share the host system's kernel and require fewer resources than virtual machines.
6. **Security:** Docker containers are secure by default and can be further secured using Docker security features.
7. **Scalability:** Docker containers can be scaled horizontally and vertically to meet demand.
8. **Microservices:** Docker containers are ideal for building microservices-based applications.
9. **DevOps:** Docker containers are a key enabler of DevOps practices.
10. **Continuous Integration/Continuous Deployment (CI/CD):** Docker containers are used in CI/CD pipelines to automate the build, test, and deployment process.
11. **Cloud-Native Applications:** Docker containers are the foundation of cloud-native applications.
12. **Open Source:** Docker is open source and has a large community of contributors.

4 Setting up Docker on Windows

Download and run the docker installer from the official website: <https://www.docker.com/products/docker-desktop>

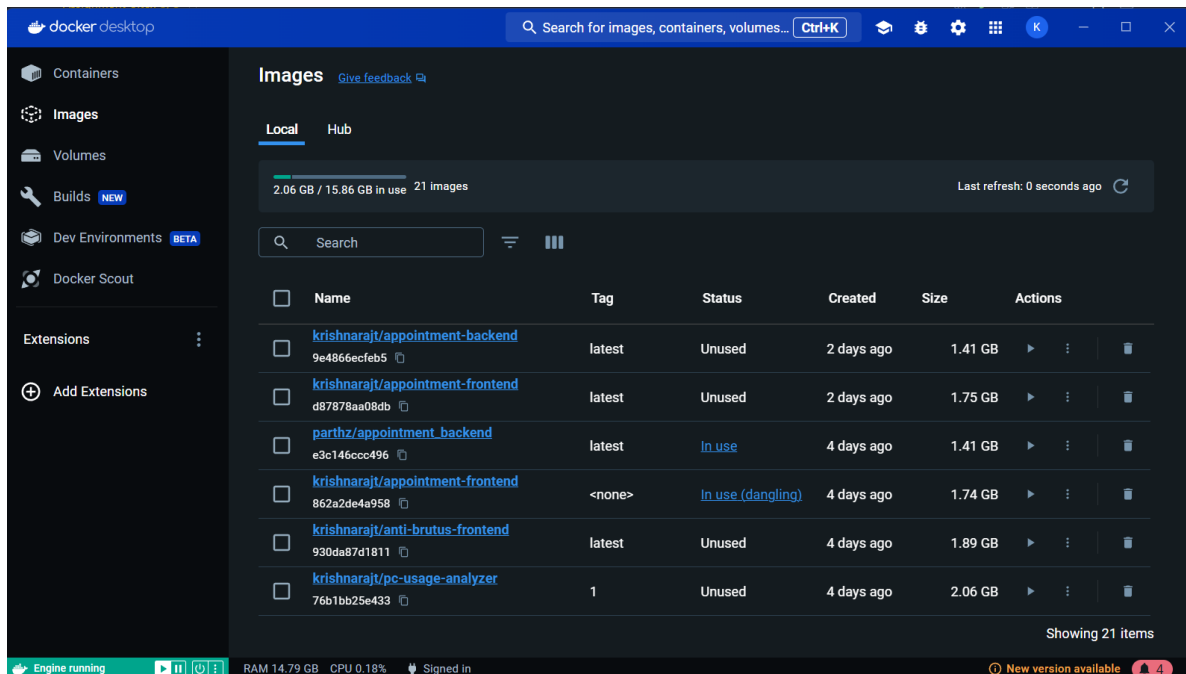


Figure 1: The Home page with images in the Docker GUI

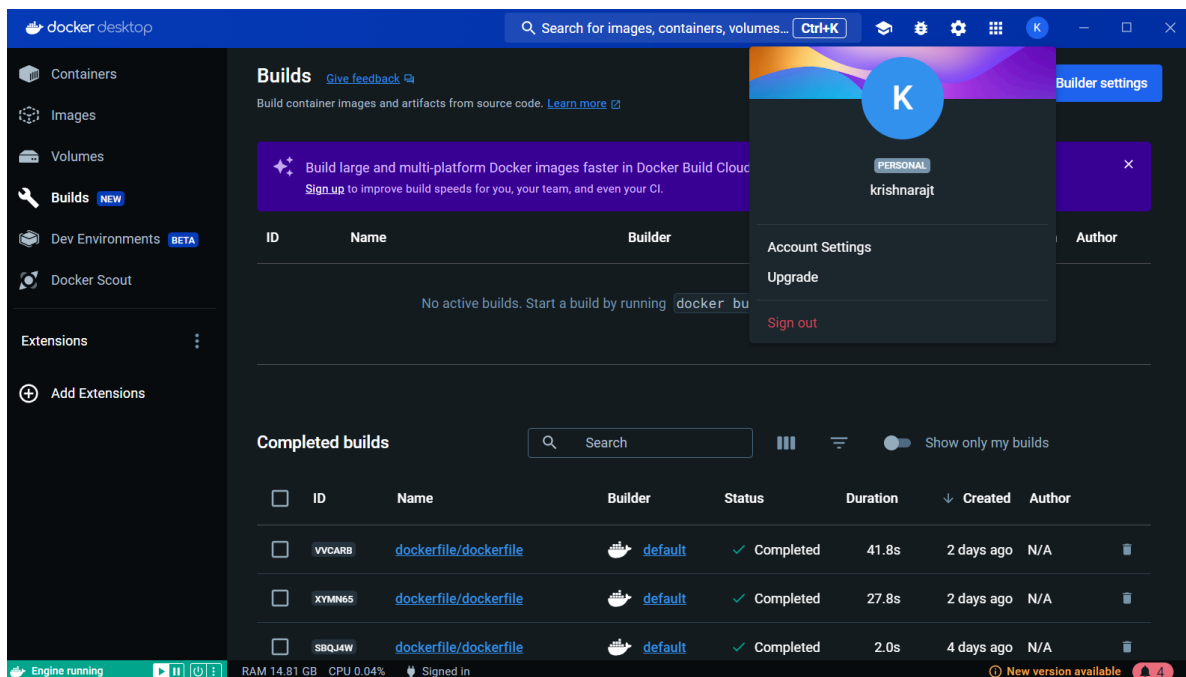


Figure 2: You can then log in to your docker account in the GUI.

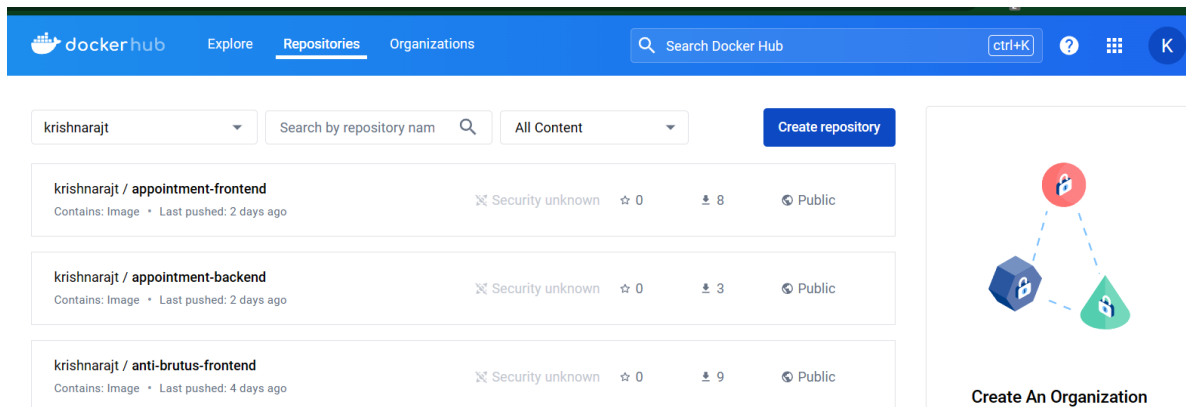


Figure 3: Docker Hub in the GUI

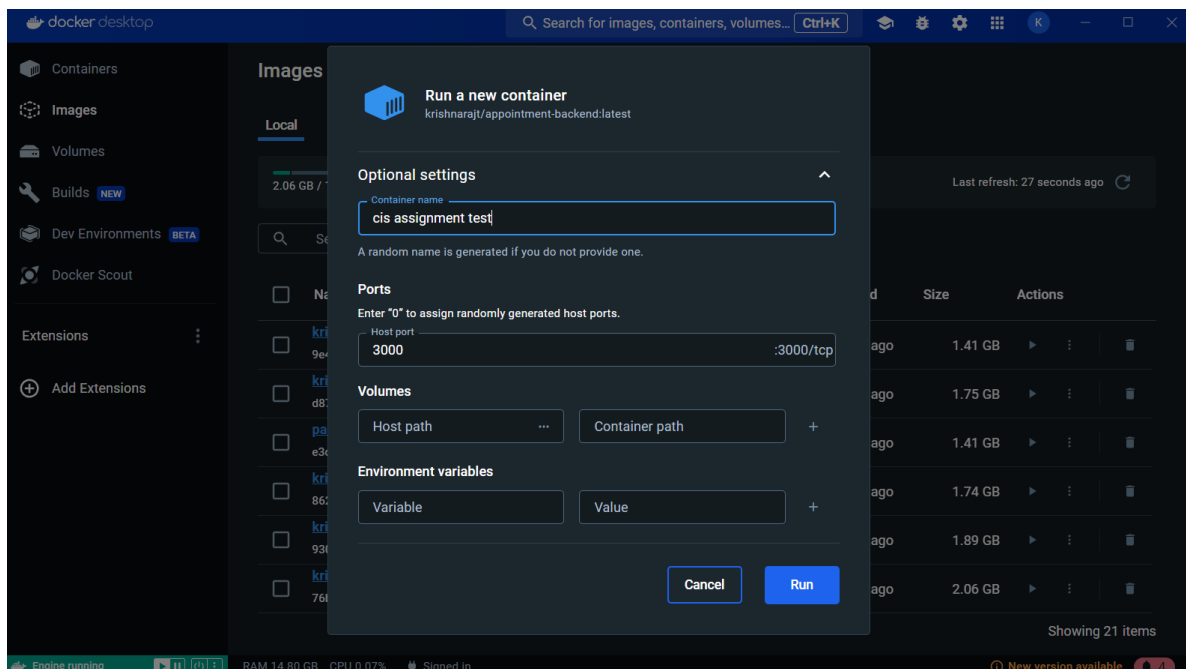


Figure 4: Setting up a new Container in the GUI.

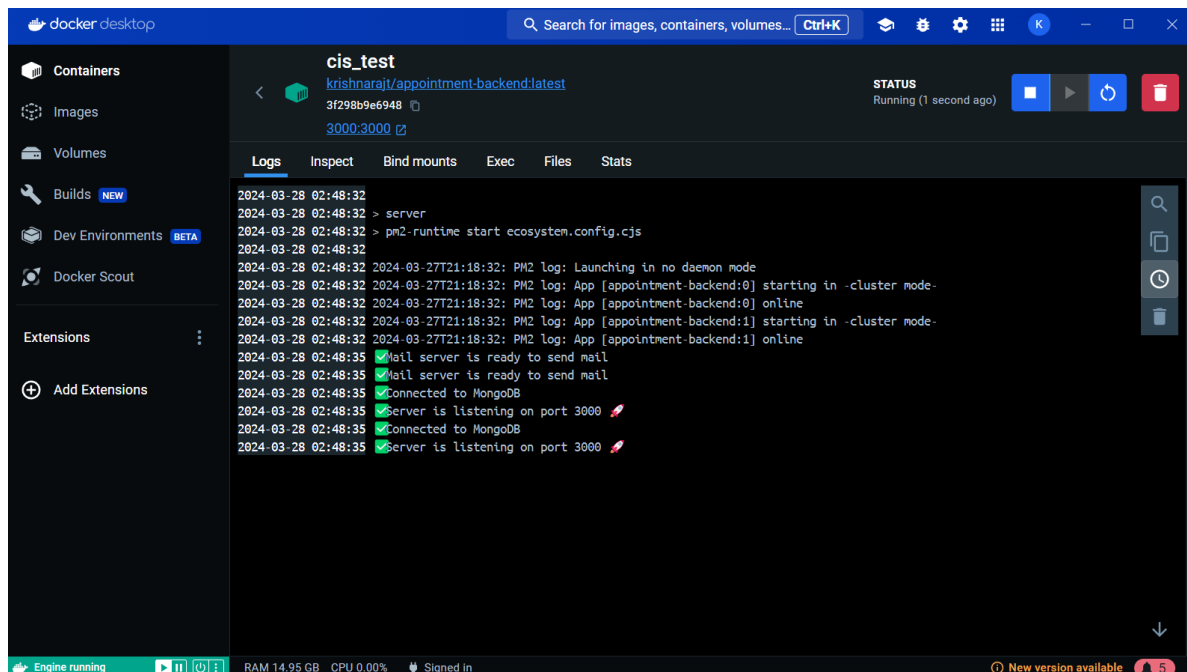


Figure 5: Running the Container

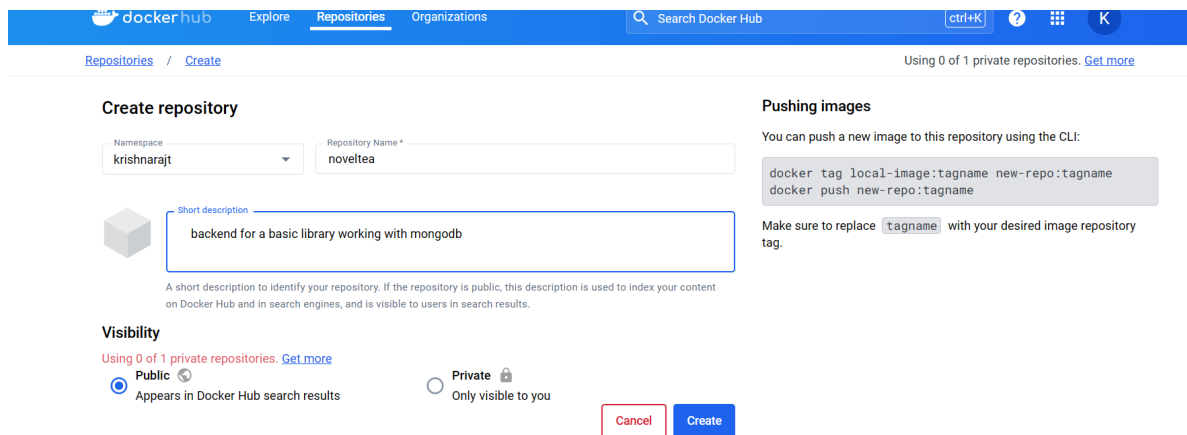
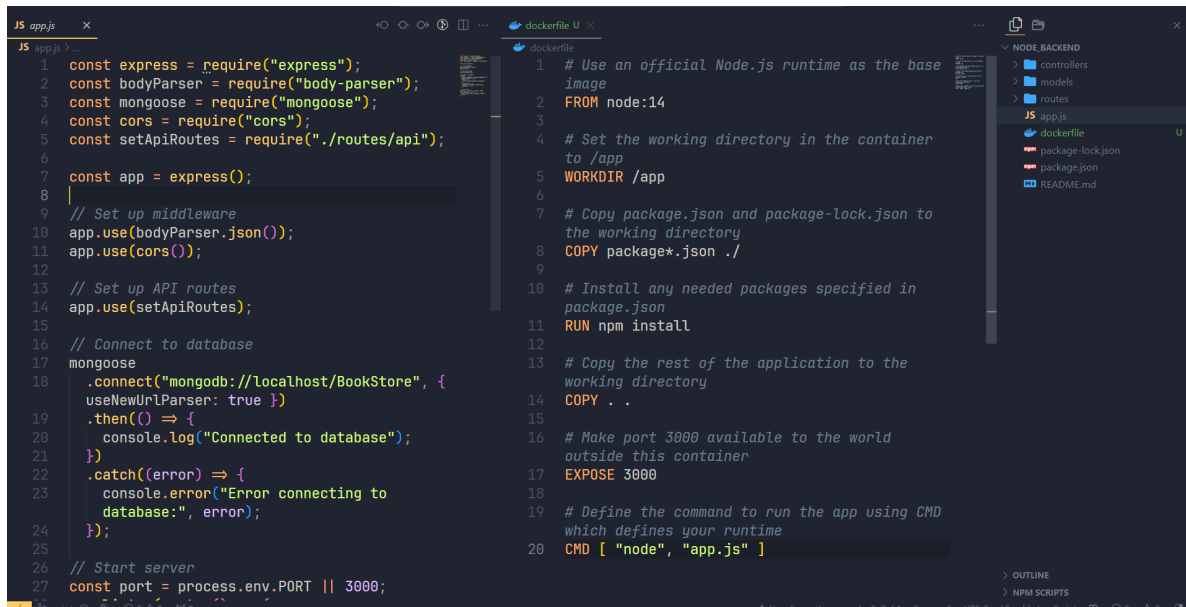


Figure 6: Making a New Repository in Docker Hub



```
JS app.js X
1 const express = require("express");
2 const bodyParser = require("body-parser");
3 const mongoose = require("mongoose");
4 const cors = require("cors");
5 const setApiRoutes = require("./routes/api");
6
7 const app = express();
8
9 // Set up middleware
10 app.use(bodyParser.json());
11 app.use(cors());
12
13 // Set up API routes
14 app.use(setApiRoutes);
15
16 // Connect to database
17 mongoose
18   .connect("mongodb://localhost/BookStore", {
19     useNewUrlParser: true })
20   .then(() => {
21     console.log("Connected to database");
22   })
23   .catch((error) => {
24     console.error("Error connecting to database:", error);
25   });
26
27 // Start server
28 const port = process.env.PORT || 3000;
```

```
dockerfile U X
1 # Use an official Node.js runtime as the base
  image
2 FROM node:14
3
4 # Set the working directory in the container
  to /app
5 WORKDIR /app
6
7 # Copy package.json and package-lock.json to
  the working directory
8 COPY package*.json ./
9
10 # Install any needed packages specified in
  package.json
11 RUN npm install
12
13 # Copy the rest of the application to the
  working directory
14 COPY . .
15
16 # Make port 3000 available to the world
  outside this container
17 EXPOSE 3000
18
19 # Define the command to run the app using CMD
  which defines your runtime
20 CMD [ "node", "app.js" ]
```

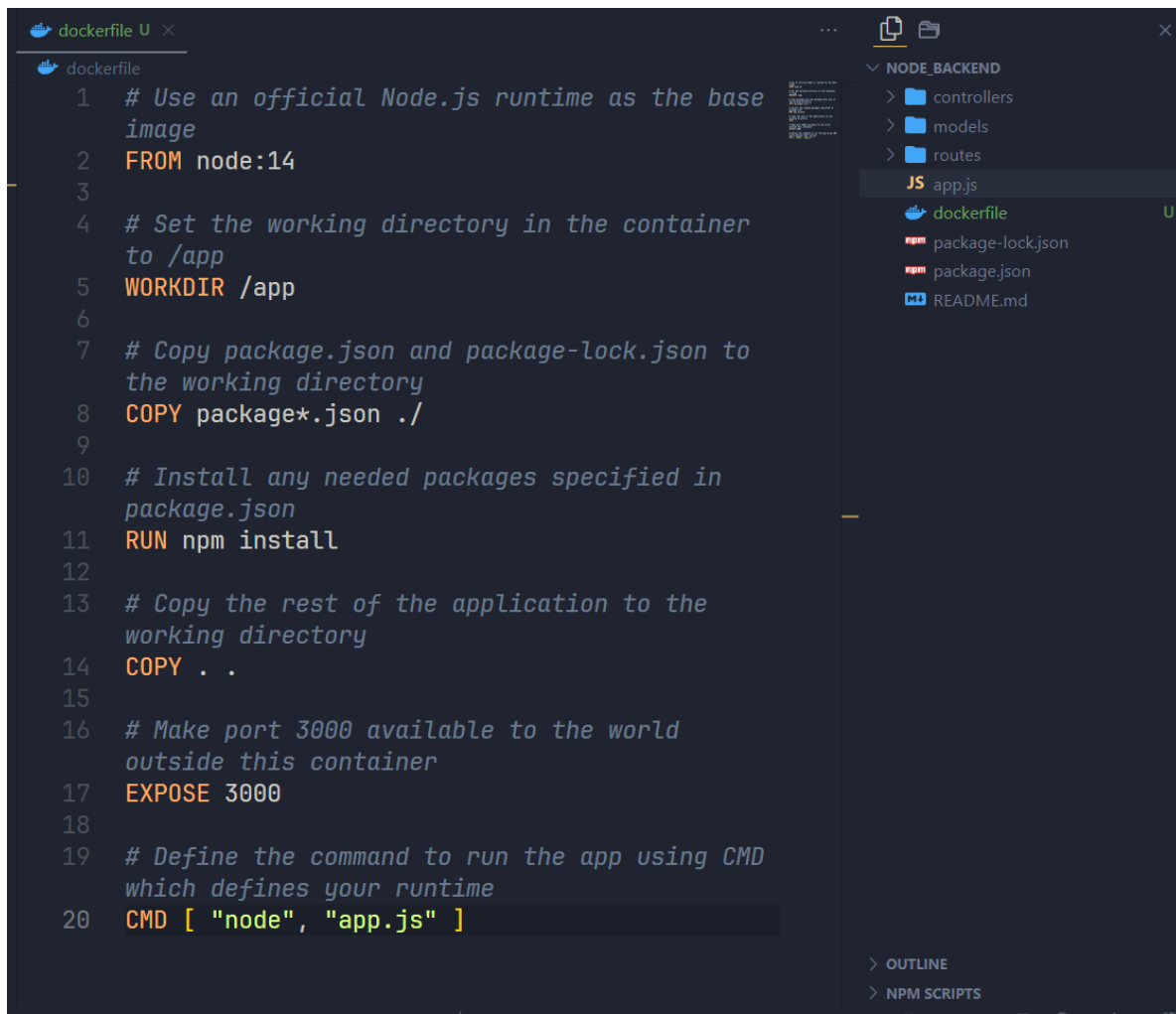
FILE EXPLORER

- node-backend
 - controllers
 - models
 - routes
 - JS
 - app.js
 - dockerfile
 - package-lock.json
 - package.json
 - README.md

OUTLINE

NPM SCRIPTS

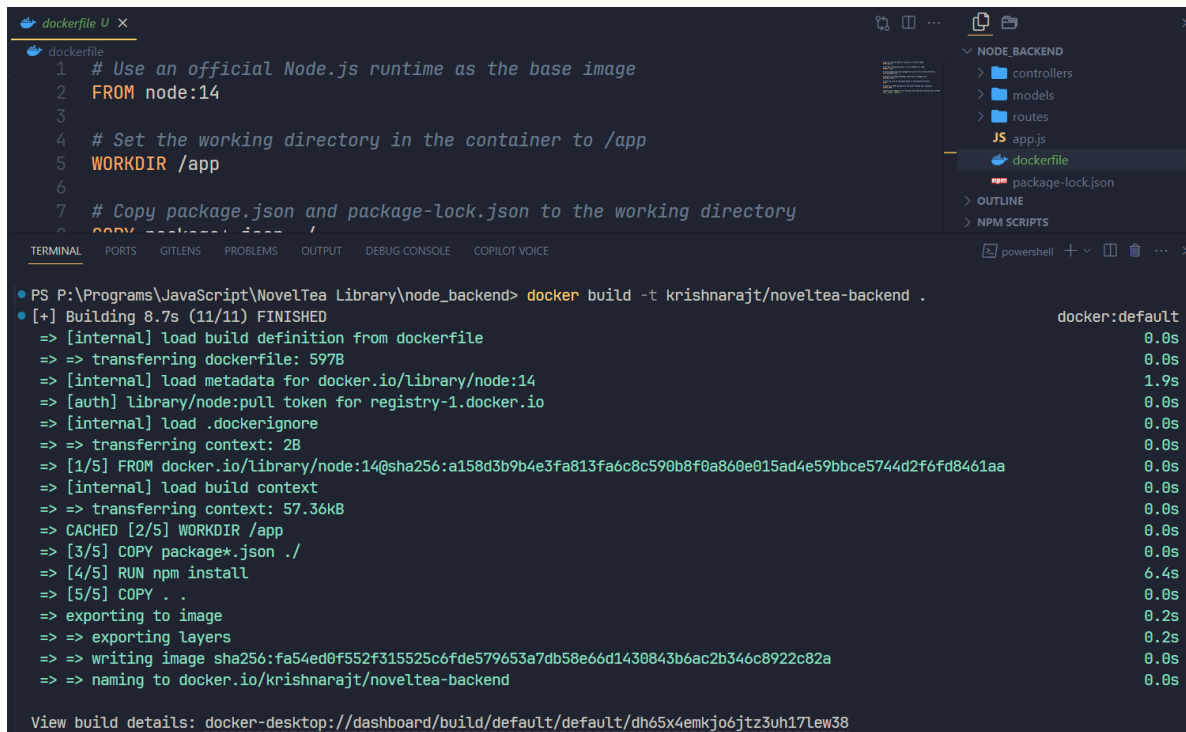
Figure 7: You have to write a dockerfile for your code.



```
1 # Use an official Node.js runtime as the base image
2 FROM node:14
3
4 # Set the working directory in the container to /app
5 WORKDIR /app
6
7 # Copy package.json and package-lock.json to the working directory
8 COPY package*.json ./
9
10 # Install any needed packages specified in package.json
11 RUN npm install
12
13 # Copy the rest of the application to the working directory
14 COPY . .
15
16 # Make port 3000 available to the world outside this container
17 EXPOSE 3000
18
19 # Define the command to run the app using CMD which defines your runtime
20 CMD [ "node", "app.js" ]
```

The image shows a code editor window titled 'dockerfile U'. The main editor area contains a Dockerfile with 20 lines of code. The code starts with a comment and 'FROM node:14', followed by 'WORKDIR /app', 'COPY package*.json ./', 'RUN npm install', 'COPY . .', 'EXPOSE 3000', and 'CMD ["node", "app.js"]'. On the right side, there is a file explorer panel titled 'NODE_BACKEND' showing a directory structure with 'controllers', 'models', and 'routes' folders, and 'app.js', 'dockerfile', 'package-lock.json', 'package.json', and 'README.md' files. The 'dockerfile' file is highlighted with a green icon and a 'U' next to it. Below the file explorer, there are sections for 'OUTLINE' and 'NPM SCRIPTS'.

Figure 8: This is what the dockerfile looks like.



```

1 # Use an official Node.js runtime as the base image
2 FROM node:14
3
4 # Set the working directory in the container to /app
5 WORKDIR /app
6
7 # Copy package.json and package-lock.json to the working directory
8 COPY package.json package-lock.json /
9
10 # Install dependencies
11 RUN npm install
12
13 # Build the application
14 RUN npm run build
15
16 # Expose the port
17 EXPOSE 3000
18
19 # Start the application
20 CMD ["npm", "start"]

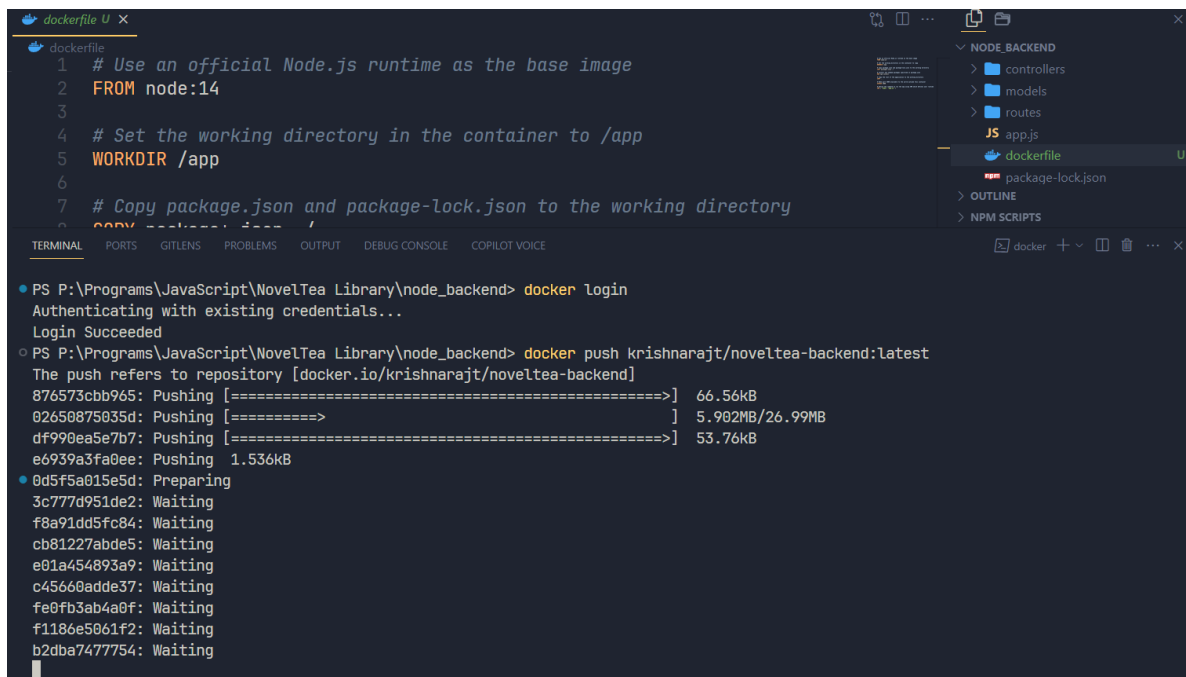
```

```

PS P:\Programs\JavaScript\NovelTea Library\node_backend> docker build -t krishnarajt/noveltea-backend .
[+] Building 8.7s (11/11) FINISHED
=> [internal] load build definition from dockerfile
=> => transferring dockerfile: 597B
=> [internal] load metadata for docker.io/library/node:14
=> [auth] library/node:pull token for registry-1.docker.io
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/5] FROM docker.io/library/node:14@sha256:a158d3b9b4e3fa813fa6c8c590b8f0a860e015ad4e59bbce5744d2f6fd8461aa
=> [internal] load build context
=> => transferring context: 57.36kB
=> CACHED [2/5] WORKDIR /app
=> [3/5] COPY package*.json ./
=> [4/5] RUN npm install
=> [5/5] COPY . .
=> exporting to image
=> => exporting layers
=> => writing image sha256:fa54ed0f552f315525c6fde579653a7db58e66d1430843b6ac2b346c8922c82a
=> => naming to docker.io/krishnarajt/noveltea-backend
View build details: docker-desktop://dashboard/build/default/default/dh65x4emkjo6jtz3uh17lew38

```

Figure 9: Building the image on the terminal.



```

PS P:\Programs\JavaScript\NovelTea Library\node_backend> docker login
Authenticating with existing credentials...
Login Succeeded

PS P:\Programs\JavaScript\NovelTea Library\node_backend> docker push krishnarajt/noveltea-backend:latest
The push refers to repository [docker.io/krishnarajt/noveltea-backend]
876573cbb965: Pushing [=====] 66.56kB
02650875035d: Pushing [=====] 5.982MB/26.99MB
df990ea5e7b7: Pushing [=====] 53.76kB
e6939a3fa0ee: Pushing 1.536kB
0d5f5a015e5d: Preparing
3c777d951de2: Waiting
f8a91dd5fc84: Waiting
cb81227abde5: Waiting
e01a454893a9: Waiting
c45660adde37: Waiting
fe0fb3ab4a0f: Waiting
f1186e5061f2: Waiting
b2dba7477754: Waiting

```

Figure 10: Pushing the new image on the hub.

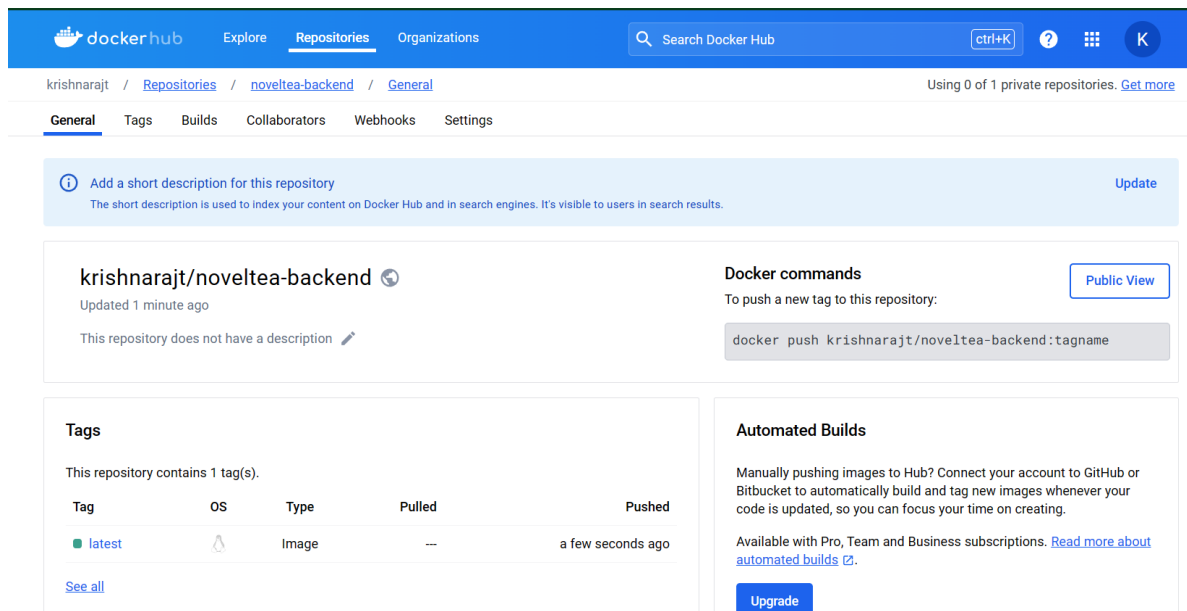


Figure 11: The updated repo with the new image.

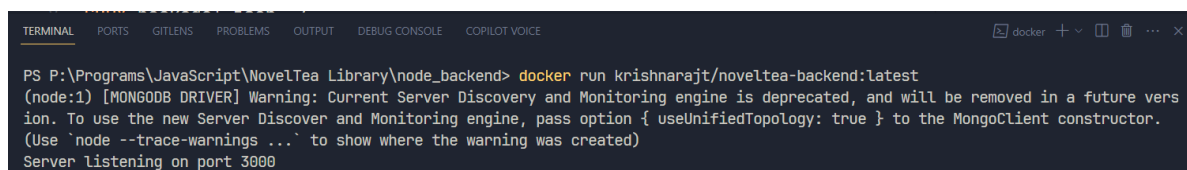


Figure 12: Running the newly created container on the terminal.

5 Platform

Operating System: Windows 11

IDEs or Text Editors Used: Visual Studio Code

Compilers or Interpreters: Python 3.10.1

6 FAQs

7 Conclusion

In this assignment, we have learnt how to install Docker on Windows and run it. We have also understood the importance of Docker in the field of Cloud Computing and DevOps.

References