**Aim:** Explore Docker commands for content management. develop a single containerized application using docker

**Objective:**

 Understand and utilize fundamental Docker commands for managing containers and images.

 Build and deploy a containerized application to demonstrate Docker's efficiency in content management.

 Gain hands-on experience in using Docker for application packaging, testing, and running in isolated environments.

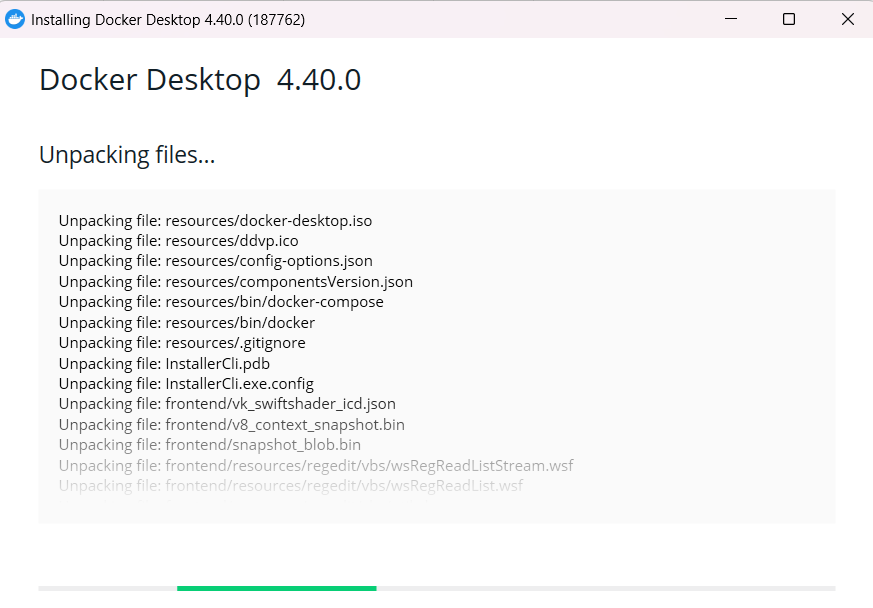
**Description:**Docker is a platform designed to make it easier to create, deploy, and run applications using containers. Containers allow developers to package applications with all parts they need, such as libraries and other dependencies, and ship it all out as one package. This project involves building a simple web application and running it inside a Docker container while exploring key Docker commands that help manage container content and configuration.

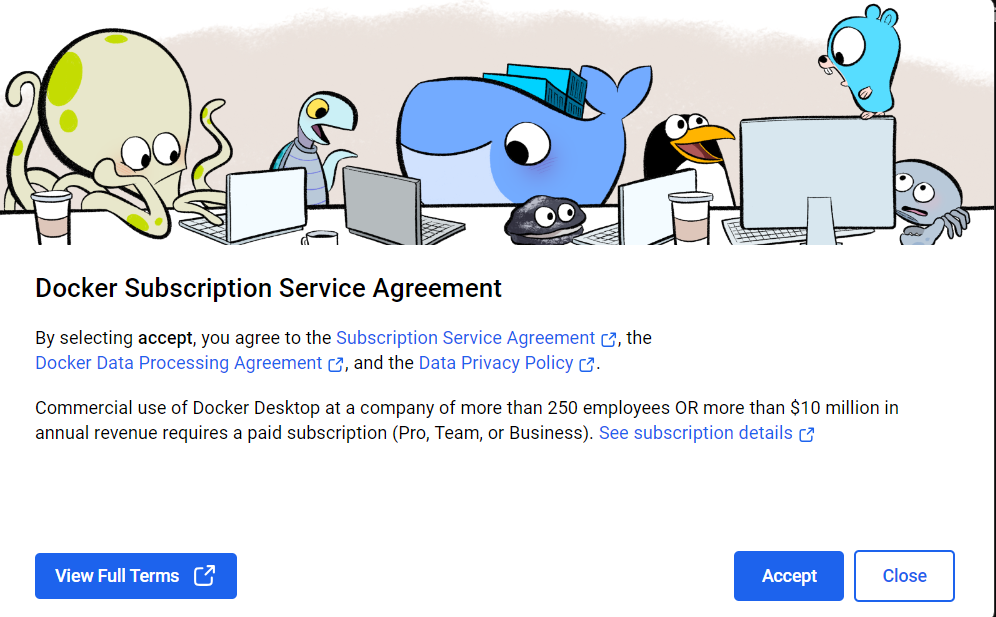
**Steps Required:**

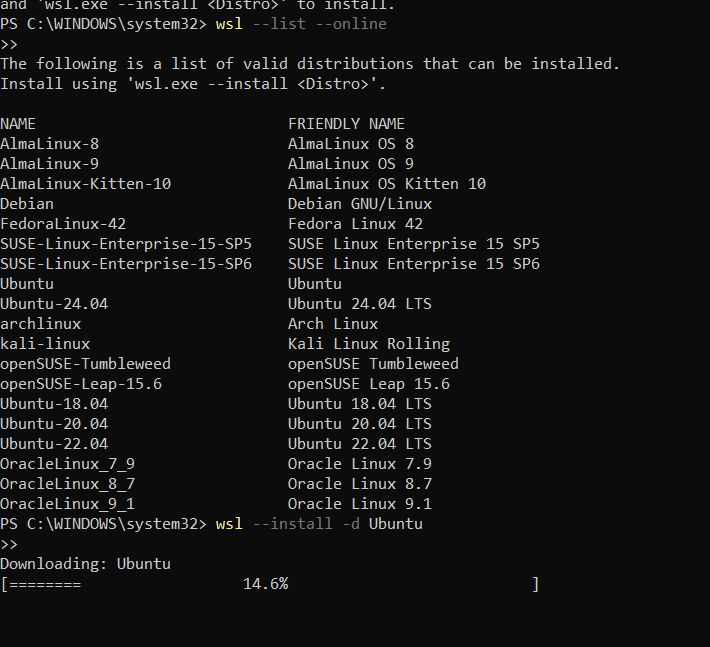
**Setup and Requirements:**

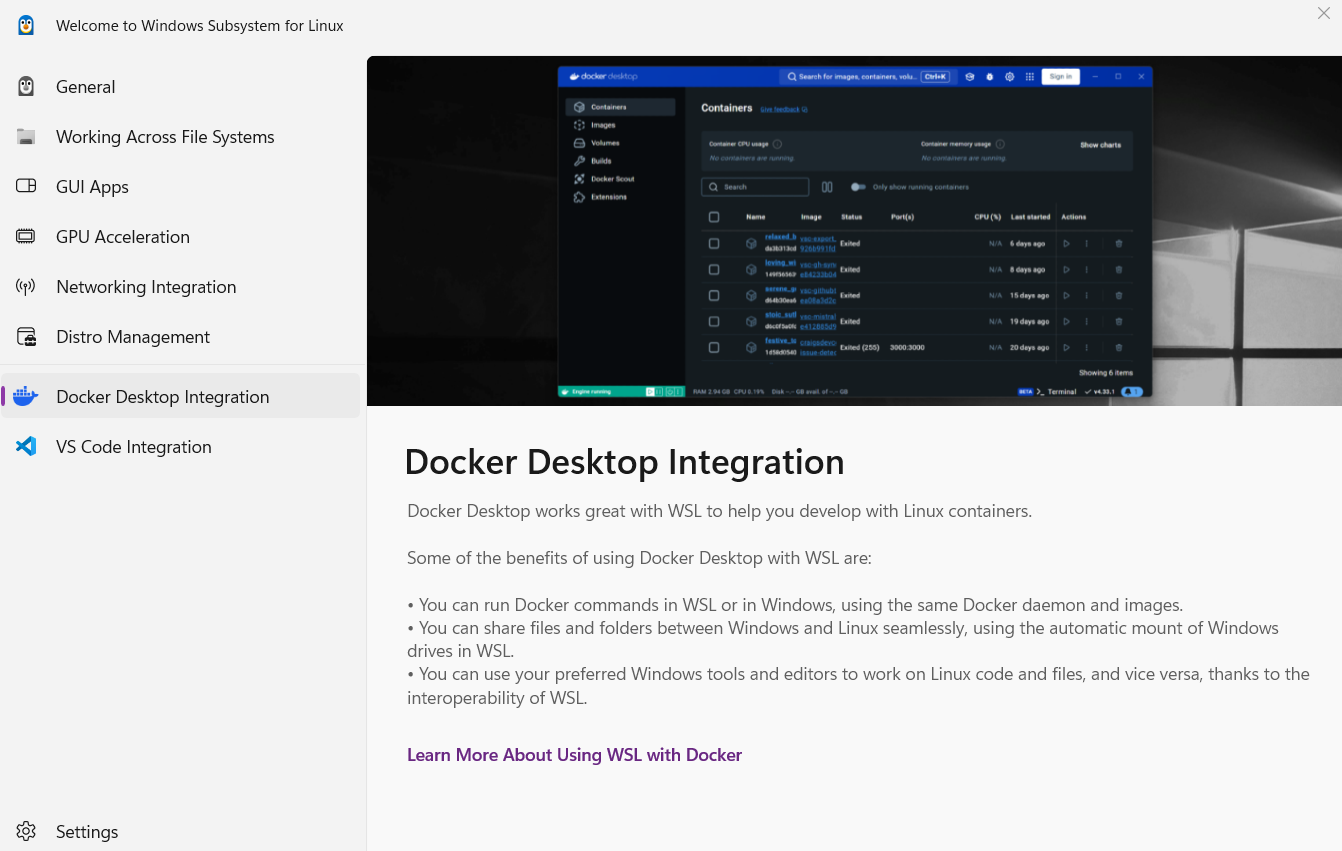
* Docker installed on the system - <https://docs.docker.com/desktop/setup/install/windows-install/>
* Basic knowledge of Dockerfile and image building
* A simple web application (e.g., a static HTML page or a Node.js app)

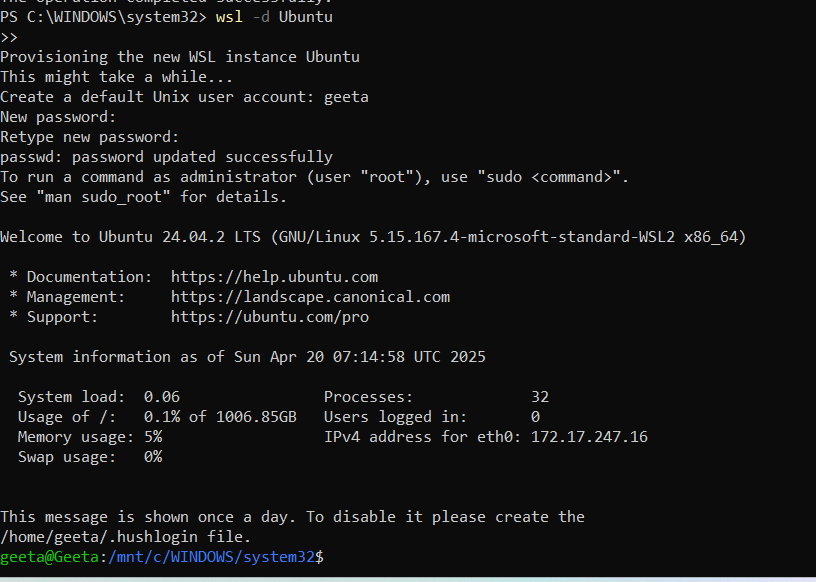
**Implementation:**

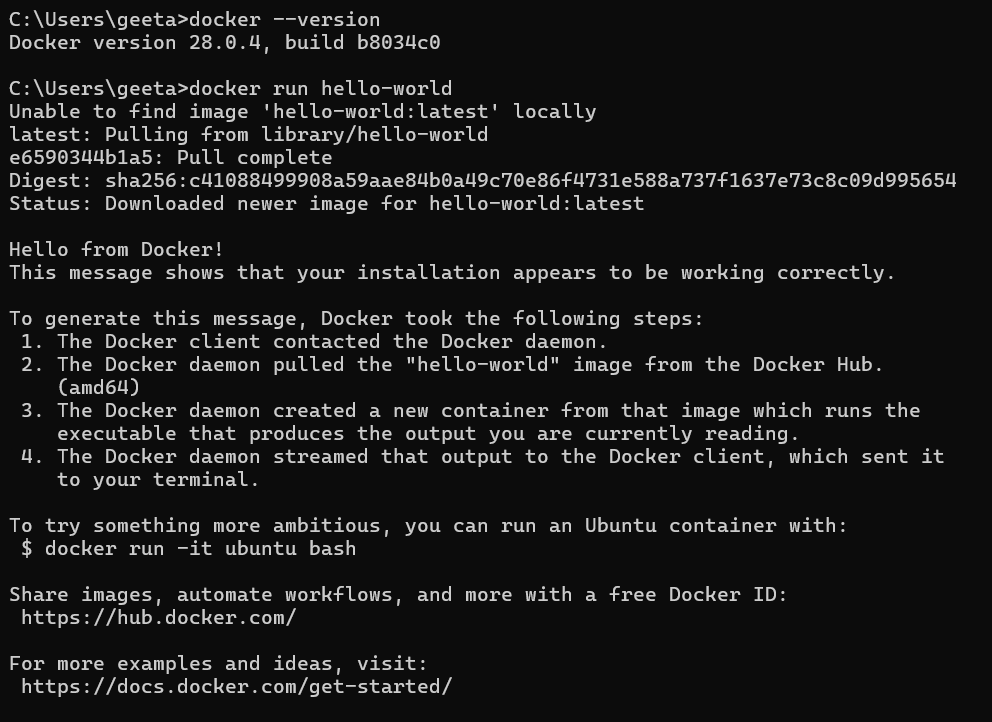


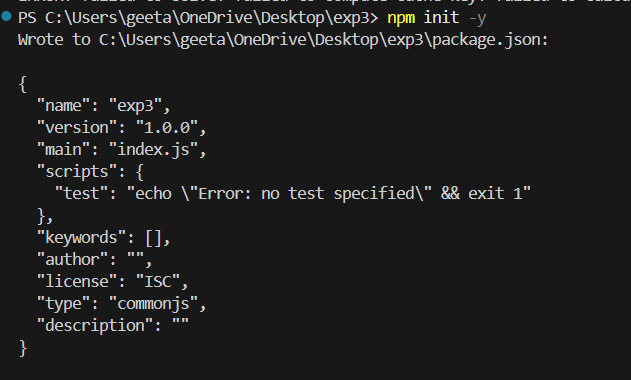


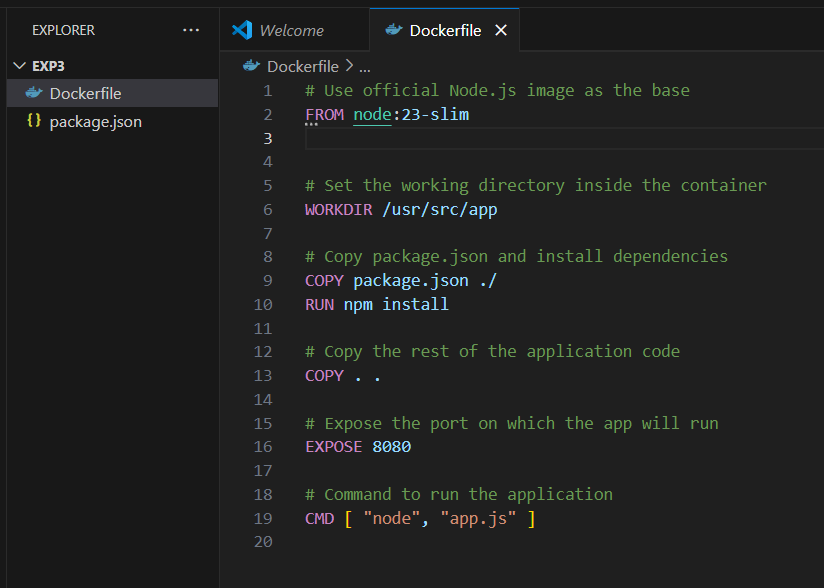








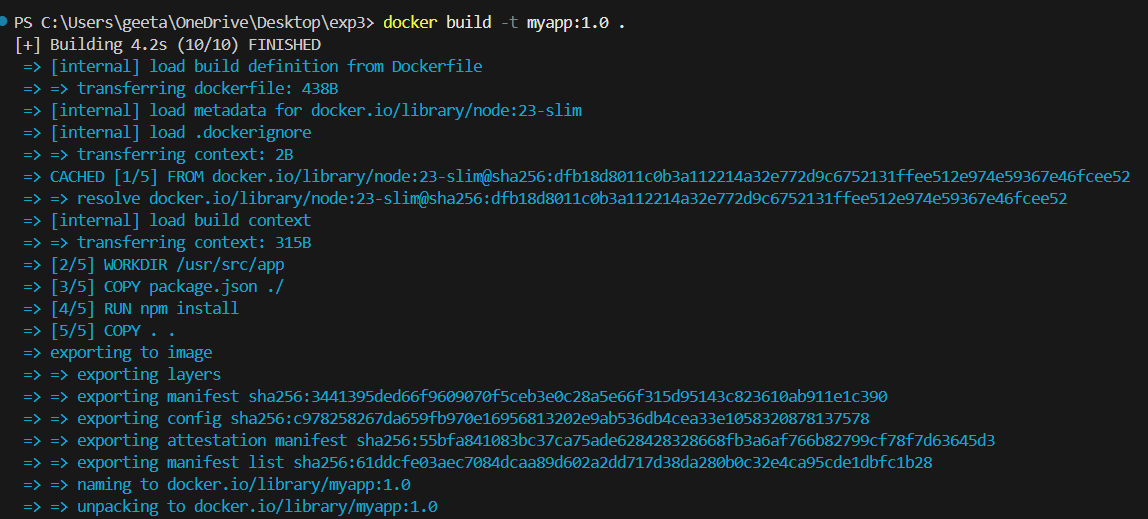




**Part 1: Docker Commands for Content Management**

Docker provides several useful commands for managing containers and images related to content management. Below are some essential Docker commands:

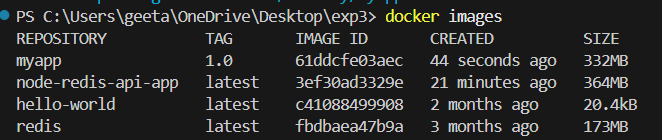
1. **Build an Image from a Dockerfile**
   * **Command:** *docker build -t <image\_name>:<tag> <path\_to\_dockerfile>*
   * **Example:** *docker build -t myapp:1.0.*



This command builds an image from the Dockerfile located in the current directory (.)

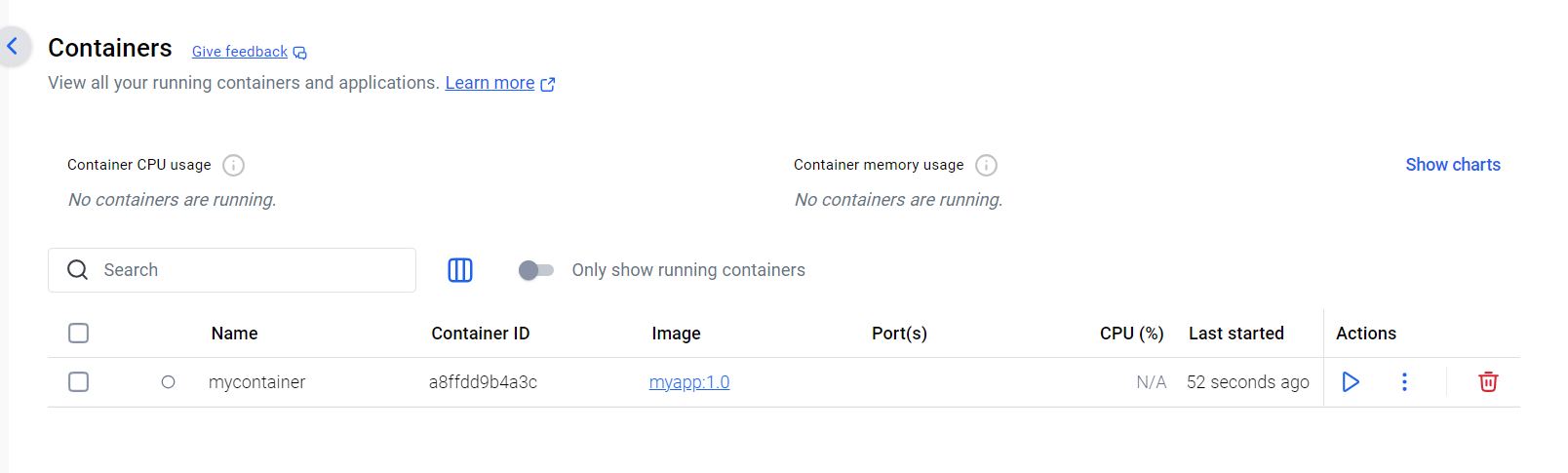
**List Docker Images**

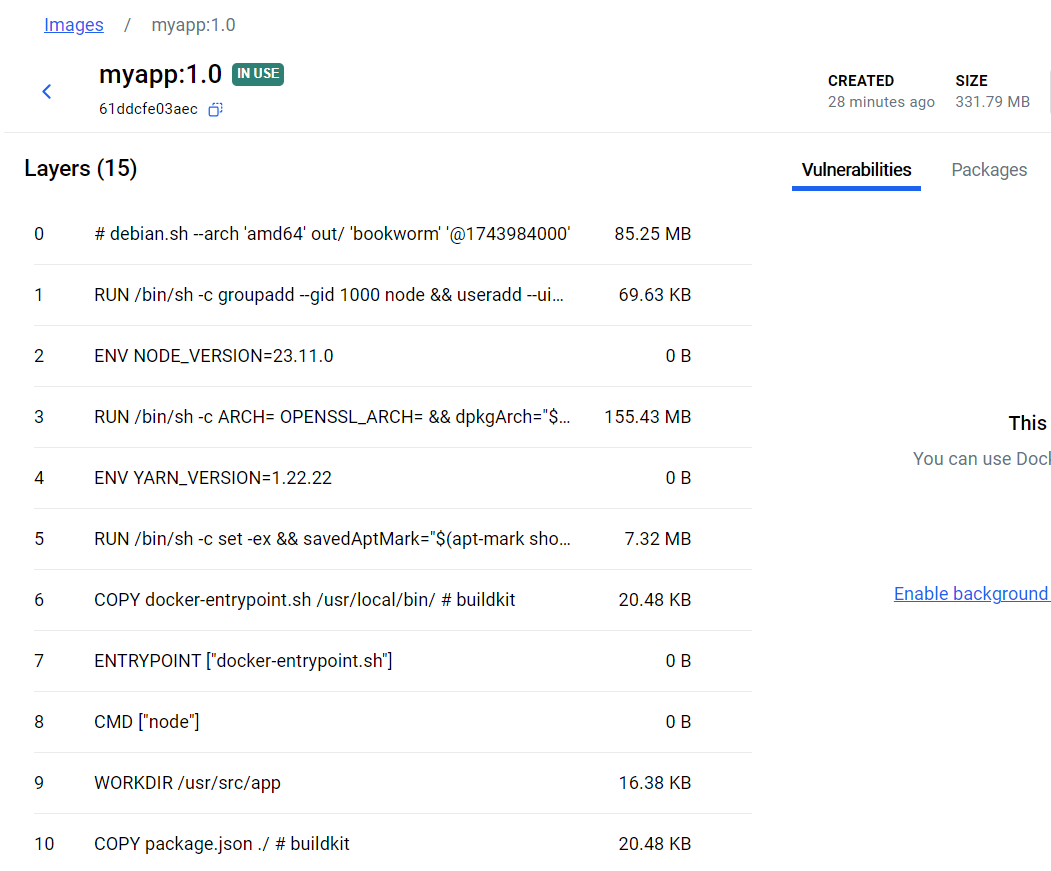
* **Command:** *docker images*
* **Example:** *docker images*

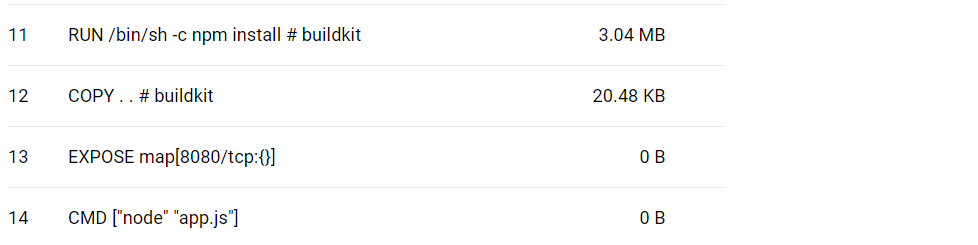


**Run a Container from an Image**

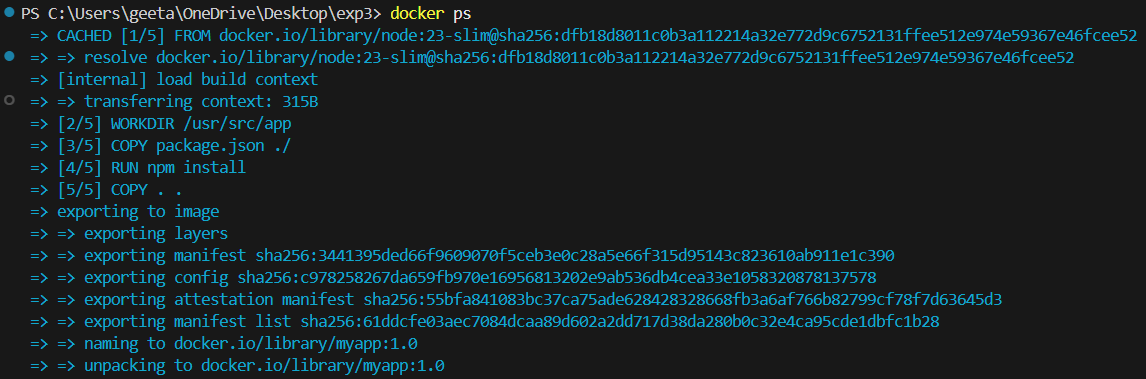
* **Command:** *docker run -d --name <container\_name> <image\_name>:<tag>*
* **Example:** *docker run -d --name mycontainer myapp:1.0*****This runs the container in detached mode (-d) from the specified image.



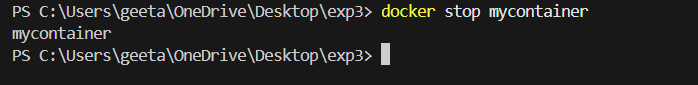




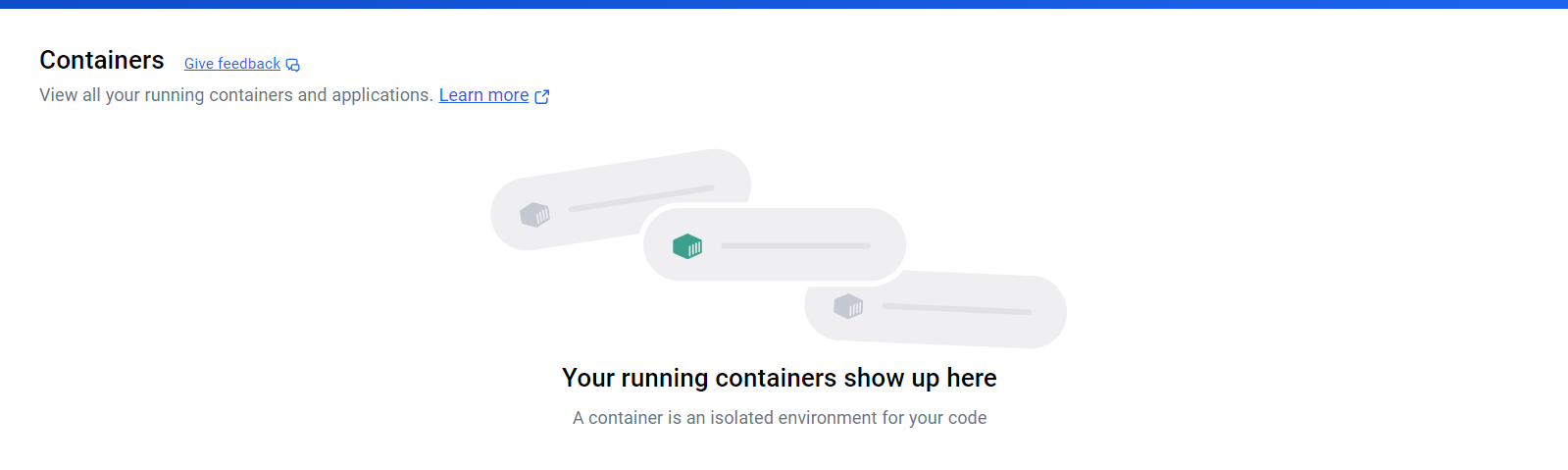
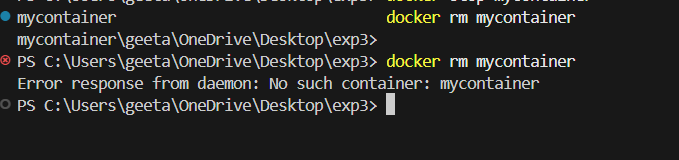
**List Running Containers**

* **Command:** *docker ps*
* **Example:** *docker ps******This shows a list of all running containers.*

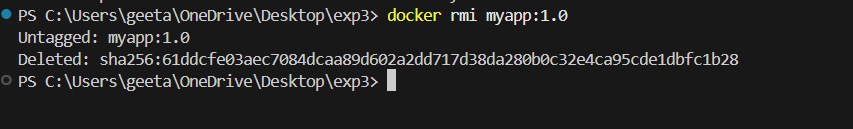
**Stop a Running Container**

* **Command:** docker stop <container\_name>
* **Example:** *docker stop mycontainer*****

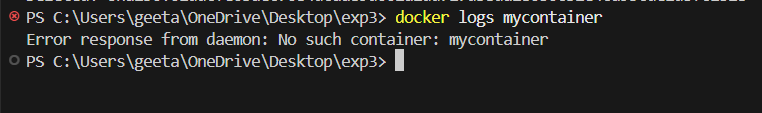
**Remove a Container**

* **Command:** docker rm <container\_name>
* **Example:** *docker rm mycontainer*****

**Remove an Image**

* **Command:** docker rmi <image\_name>:<tag>
* **Example:** *docker rmi myapp:1.0*****

**View Logs of a Running Container**

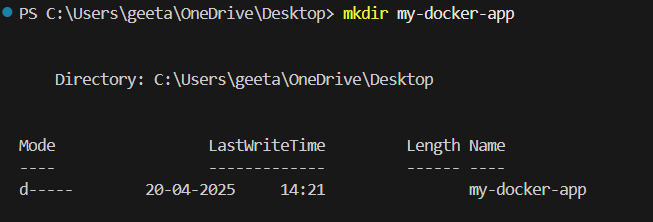
* **Command:** docker logs <container\_name>
* **Example:** *docker logs mycontainer*****

**Part 2: Developing a Simple Containerized Application**

Let’s create a simple Node.js web application that serves a "Hello, Docker!" message. We will containerize this application using Docker.

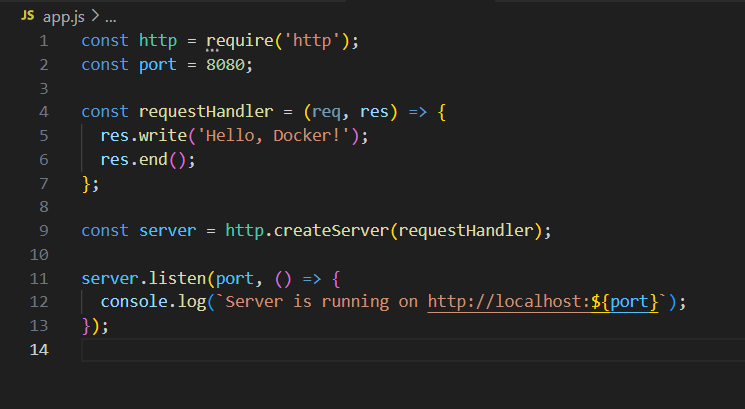
**Step 1: Create the Application**

1. **Create a new directory for your application.**



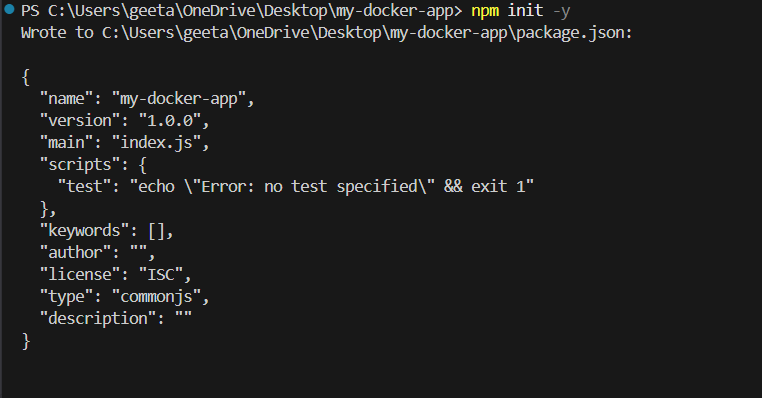
**Create the Node.js application:**

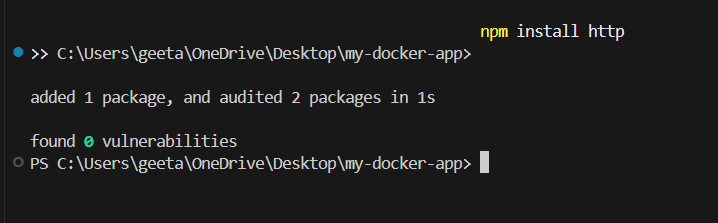
* Create a file called app.js with the following content:



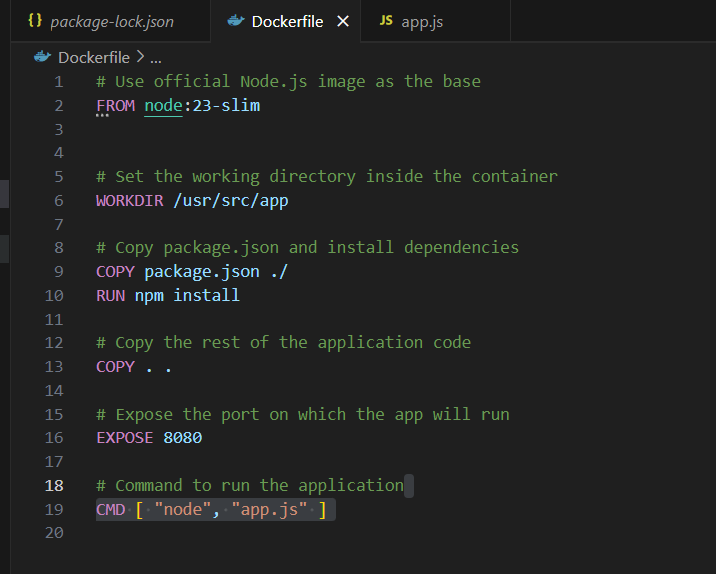
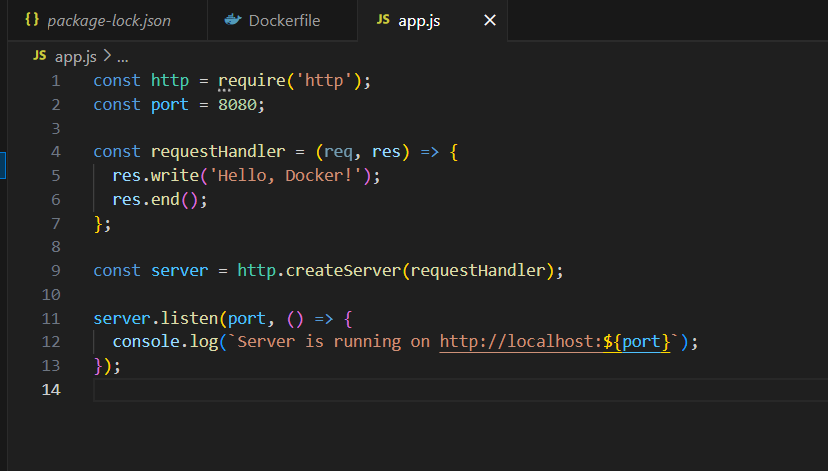
**Create a package.json file:**

* Run npm init -y to generate a basic package.json file, and then run: npm install http



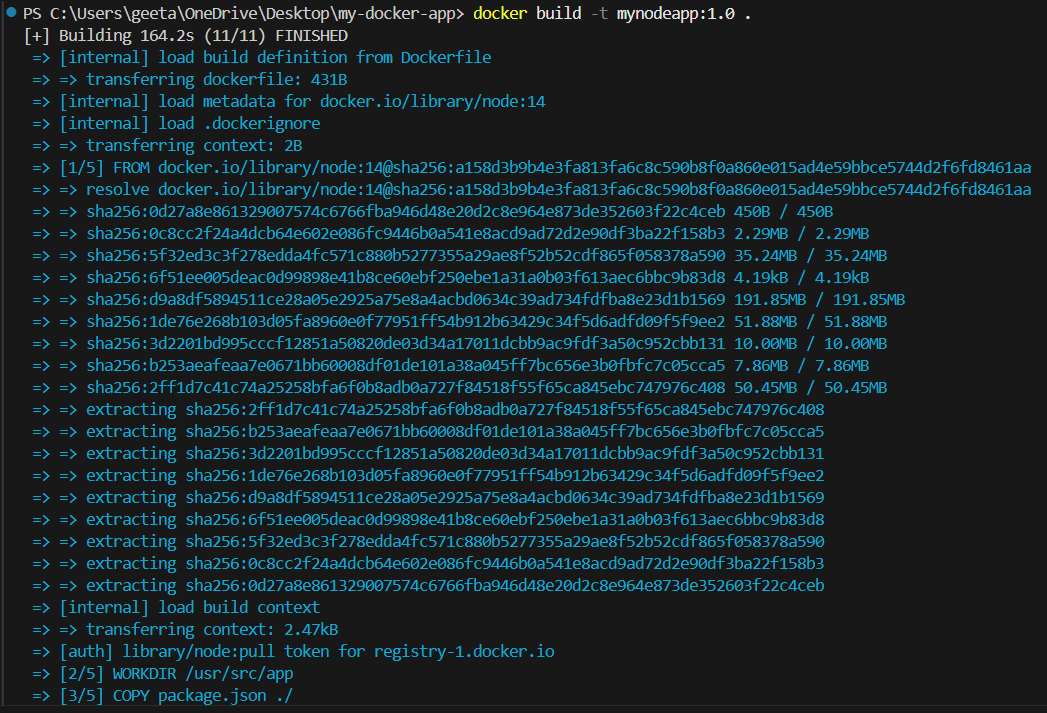


**Step 2: Create the Dockerfile**

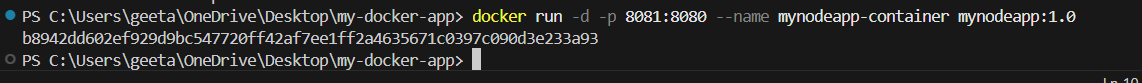
1. **Create a Dockerfile in the same directory with the following content:** ****
2. App.js

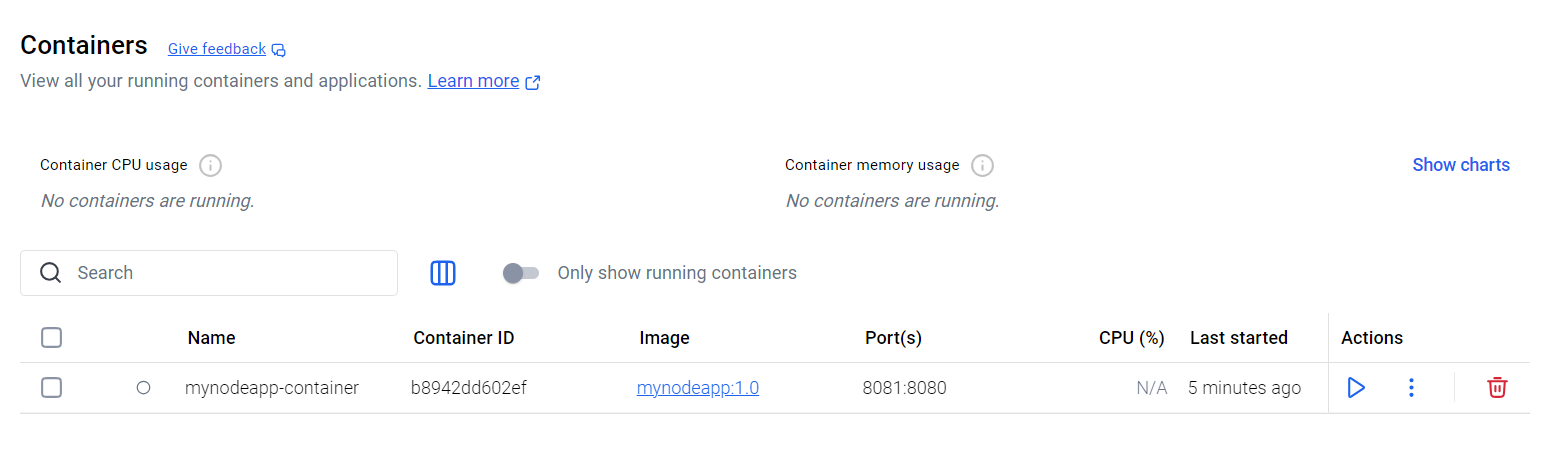
**Step 3: Build the Docker Image**

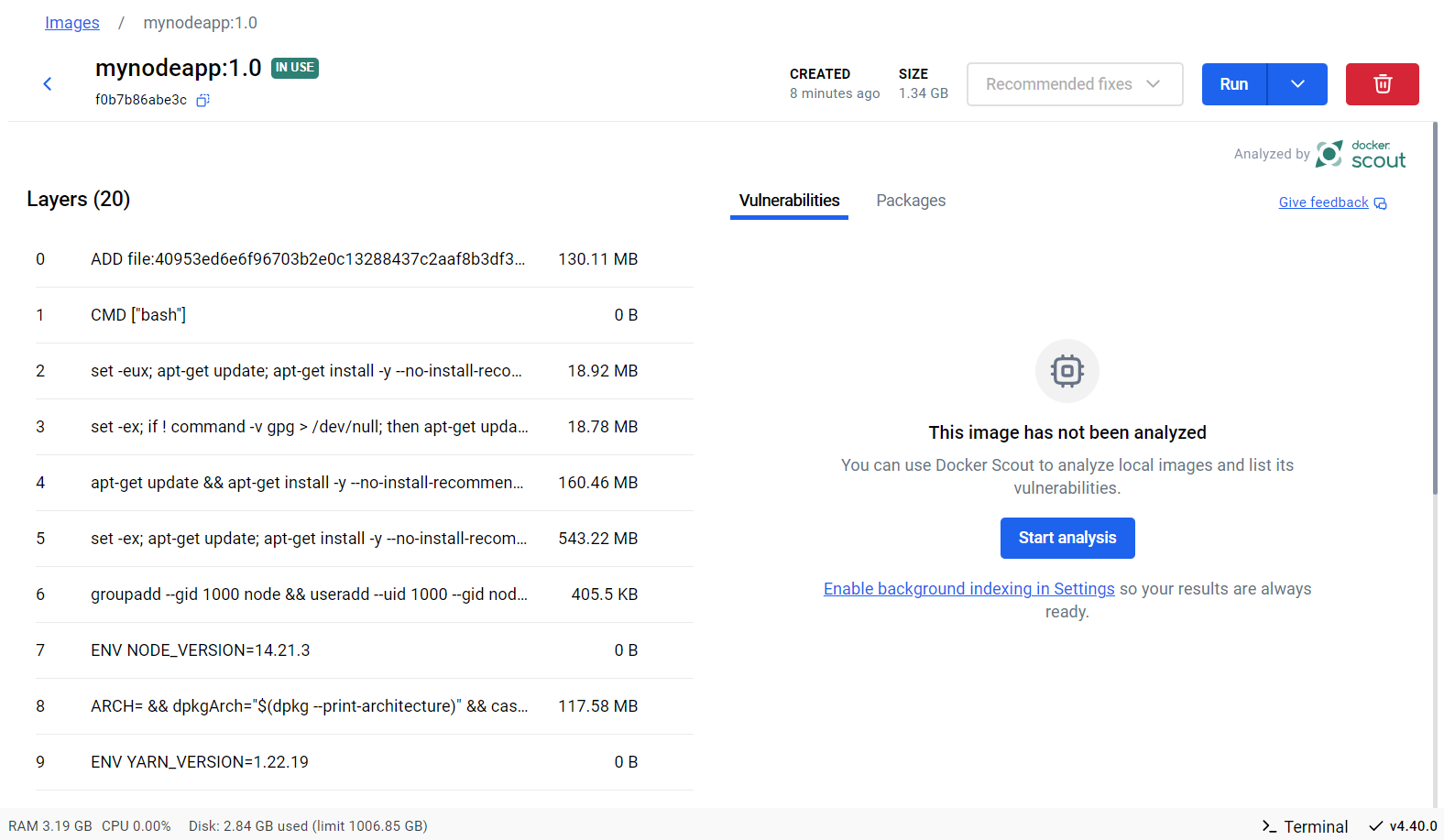
1. **Build the Docker image using the command: docker build -t mynodeapp:1.0 .**

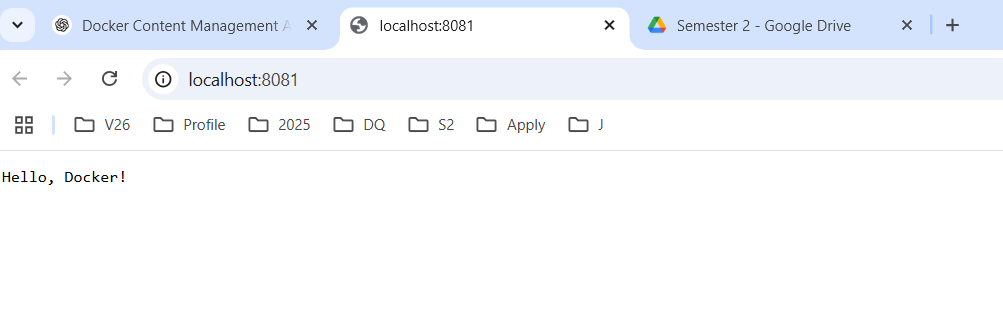


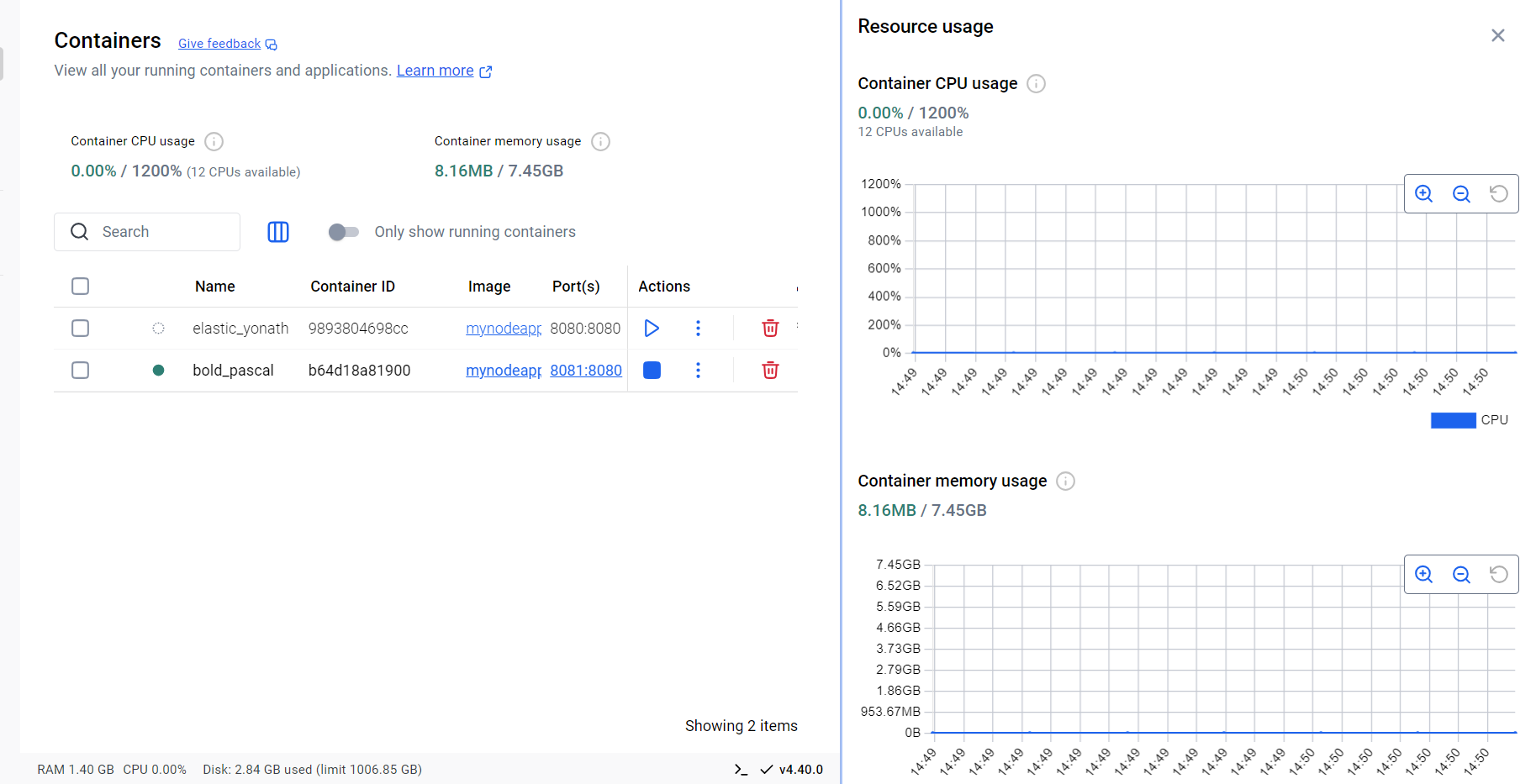
**Step 4: Run the Docker Container**

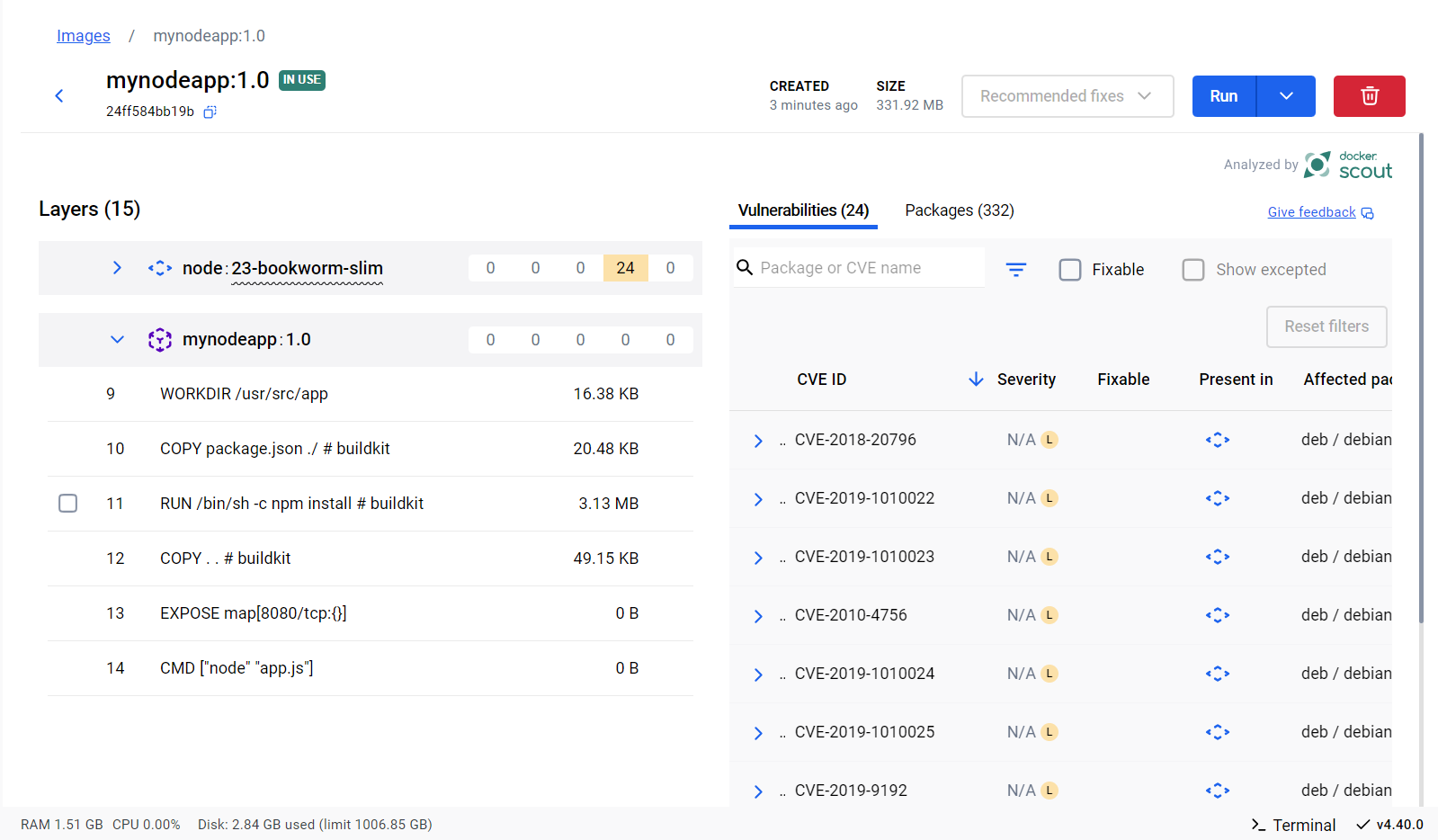
1. **Run the container: docker run -d -p 8081:8080 --name mynodeapp-container mynodeapp:1.0**











**Outcome:**

* A functional containerized application that runs independently of the host system’s environment
* Practical experience with Docker commands and concepts such as images, containers, volumes, and networking
* Understanding of how Docker facilitates efficient content and configuration management

**Conclusion:**

Through this project, a strong foundation in Docker's core commands and concepts was developed. The hands-on experience of containerizing a simple application demonstrated the power of Docker in managing application content, dependencies, and deployment with ease. This forms a stepping stone toward container orchestration and more complex DevOps implementations.