

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 19/10/23

Experiment No 8

Aim : To deploy static web application on docker.

Lab Outcome :

LO1: To understand the fundamentals of DevOps engineering and be fully proficient with DevOps terminologies, concepts, benefits, and deployment options to meet your business requirements.

LO5 : To understand the concept of containerization and analyze the containerization of OS images and deployment of applications over Docker.

Theory :

In the ever-evolving world of web development and deployment, containers have gained immense popularity due to their efficiency, scalability, and portability. Docker, a leading containerization platform, has become the de facto choice for packaging and deploying applications. While Docker is frequently associated with complex microservices and multi-container applications, it can also be used effectively for deploying simple static web applications. This article will delve into the theory and steps involved in deploying a static web application on Docker.

Docker is an open-source platform designed to automate the deployment, scaling, and management of applications inside lightweight, portable containers. These containers encapsulate the application code, runtime, system tools, and libraries, ensuring that the application runs consistently in any environment.

Before we begin deploying a static web application on Docker, there are a few prerequisites to consider:

1. Docker Installation: Ensure that Docker is installed on your system. Docker provides a user-friendly installation process for various operating systems.
2. Static Web Application: You should have a static web application ready for deployment. A static web application typically consists of HTML, CSS, JavaScript, and other static assets.

Steps to Deploy a Static Web Application on Docker

1. Create a Dockerfile

A Dockerfile is a script containing a set of instructions that Docker will use to build a Docker image. For a static web application, the Dockerfile is straightforward. Here's a sample Dockerfile:

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 19/10/23

In this Dockerfile, we start with a base image that contains Node.js, set the working directory, copy the application files, expose port 80, define an environment variable, and finally, run the static web application.

2. Build a Docker Image

To build a Docker image from the Dockerfile, navigate to the directory containing the Dockerfile and the web application files and execute the following command:

```
```bash
docker build -t my-static-web-app .
```
```

This command will create a Docker image tagged as "my-static-web-app." Make sure to replace it with your preferred image name.

3. Run a Docker Container

Now that we have a Docker image, we can run a Docker container based on that image. Use the following command:

```
```bash
docker run -p 8080:80 my-static-web-app
```
```

This command will start a Docker container from the "my-static-web-app" image, mapping port 8080 on your host machine to port 80 inside the container. You can access your static web application by opening a web browser and navigating to <http://localhost:8080>.

Benefits of Docker for Deploying Static Web Applications

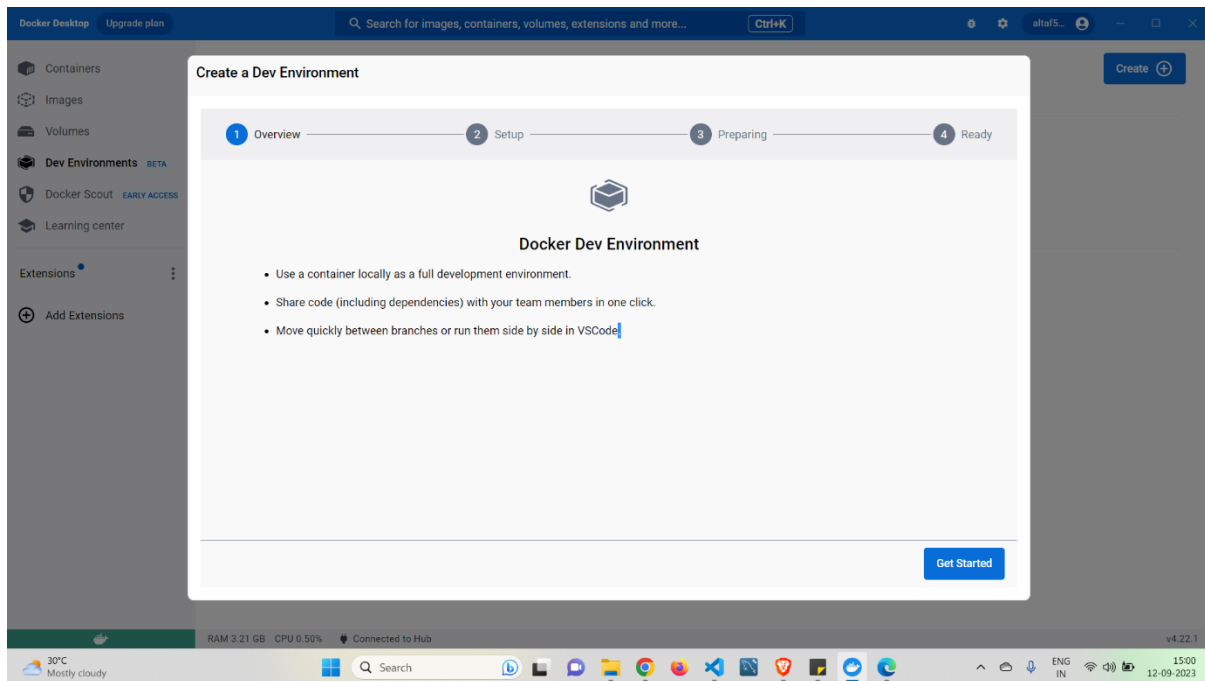
1. Portability: Docker containers encapsulate the application and its dependencies, ensuring consistent behavior across different environments.
2. Isolation: Each container operates independently, avoiding conflicts and ensuring a clean environment.
3. Scalability: Docker simplifies scaling by allowing the deployment of multiple containers and load balancing between them.

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 19/10/23

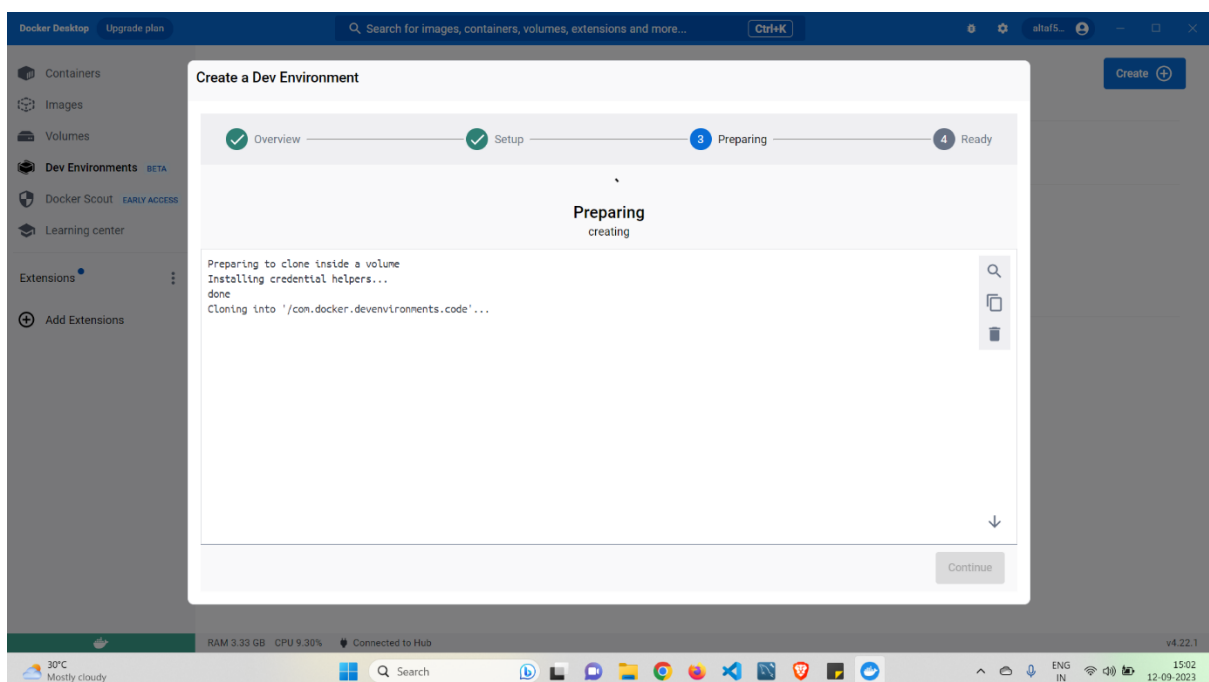
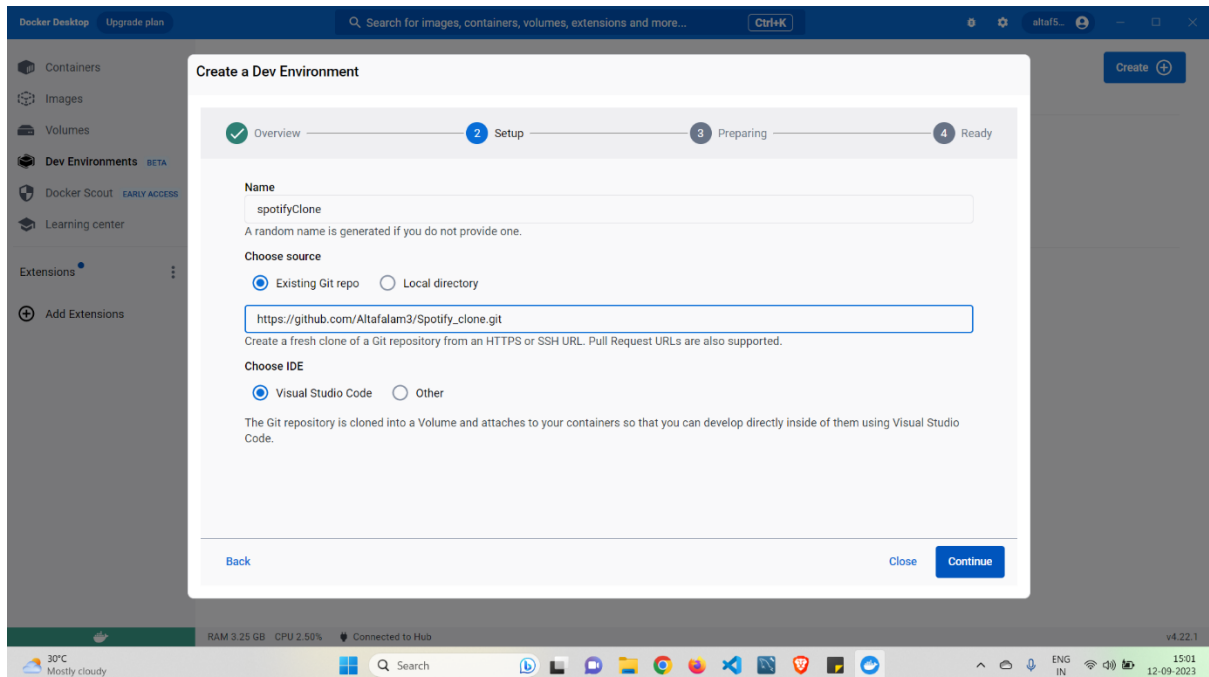
4. Resource Efficiency: Containers are lightweight and share the host OS kernel, leading to efficient resource utilization.
5. Easy Version Control: Images and containers can be versioned, making it easy to roll back to previous states if needed.
6. Security: Docker provides security features to isolate containers and control access.
7. Community and Ecosystem: Docker has a vast community and ecosystem, with a wealth of pre-built images and tools.

Output:



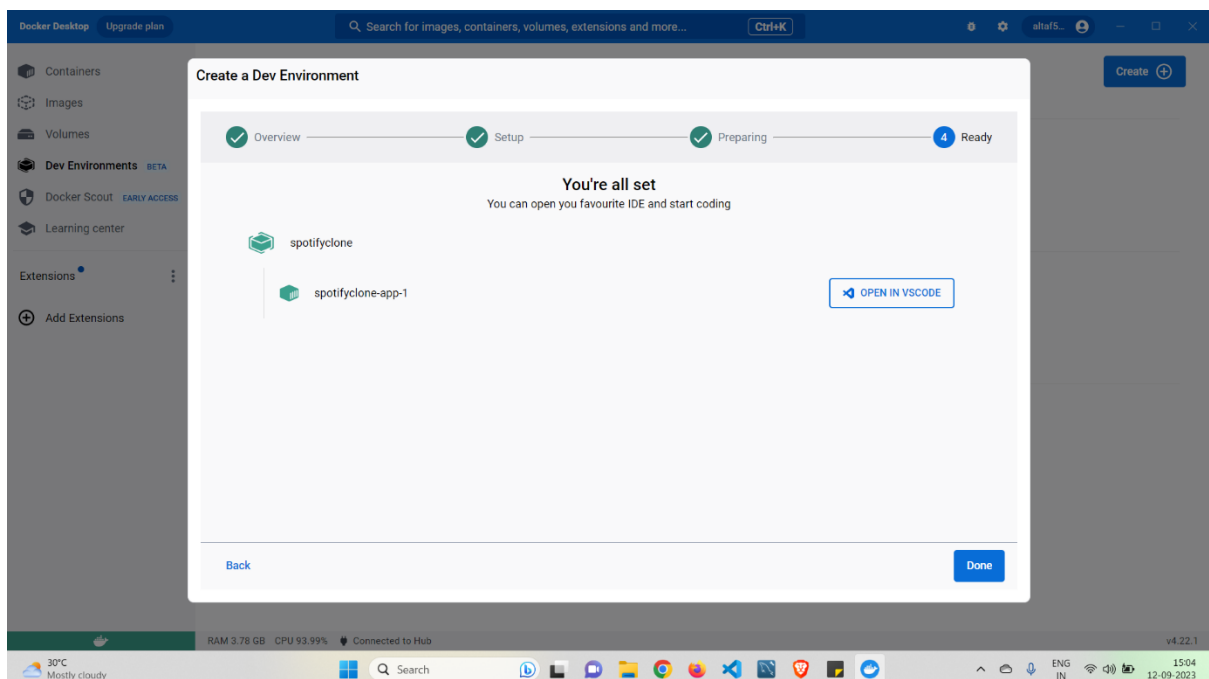
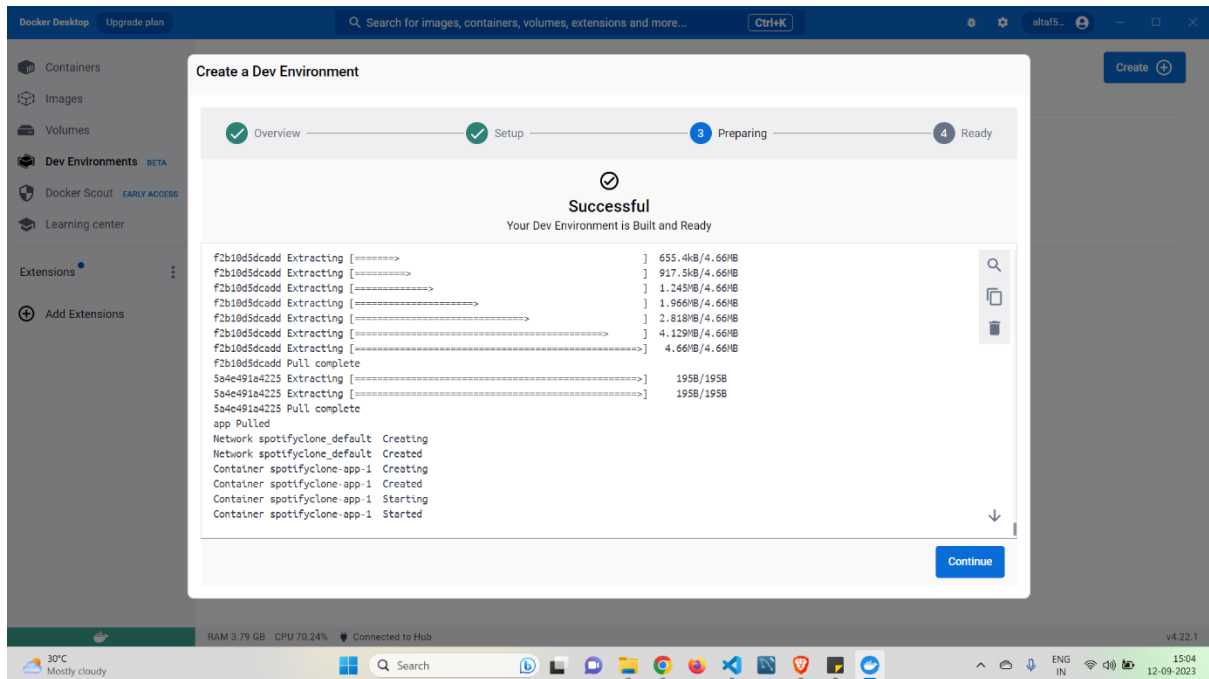
Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 19/10/23



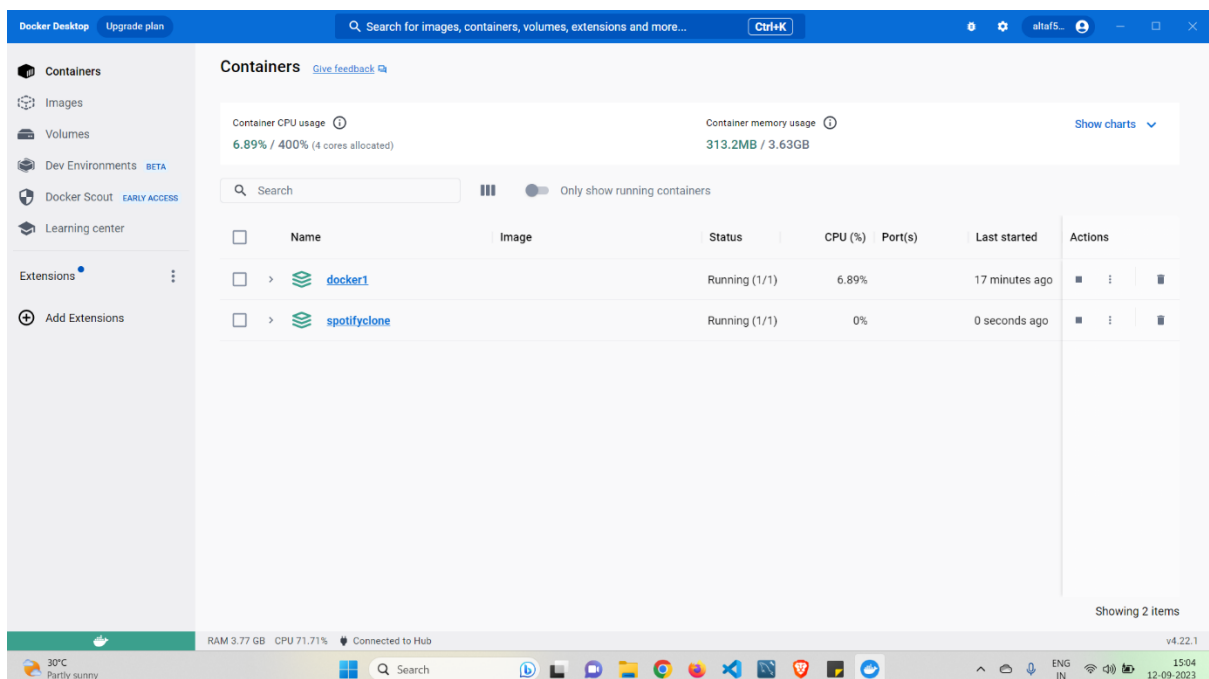
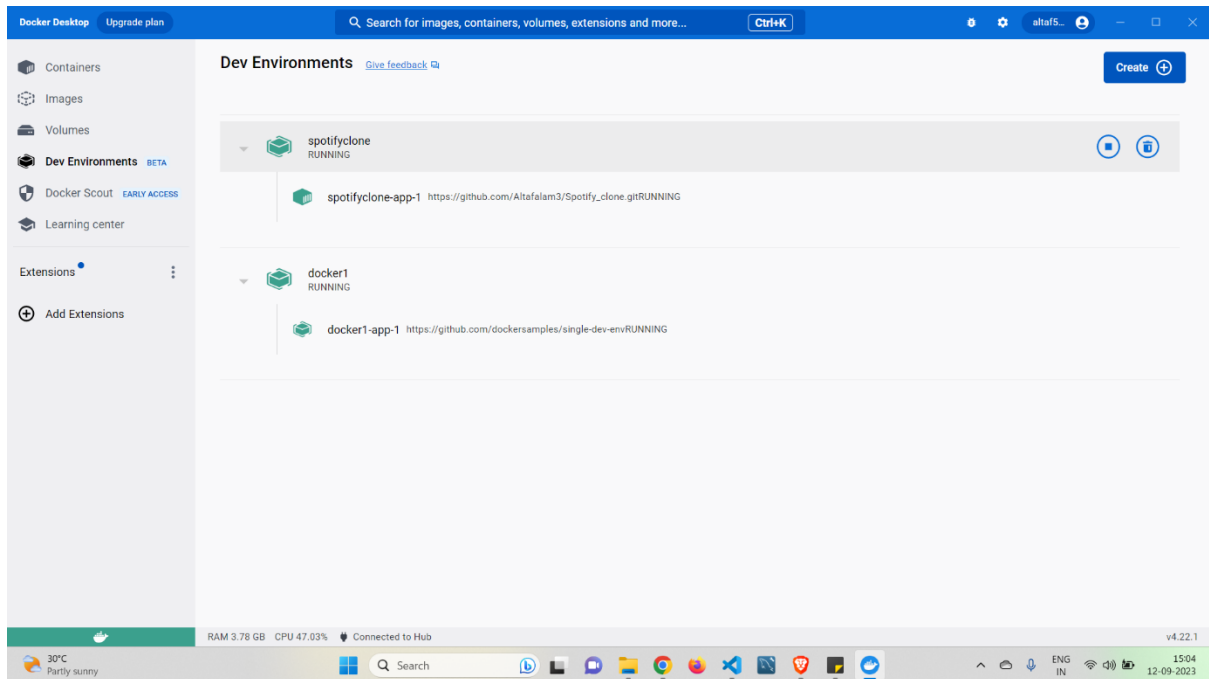
Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 19/10/23



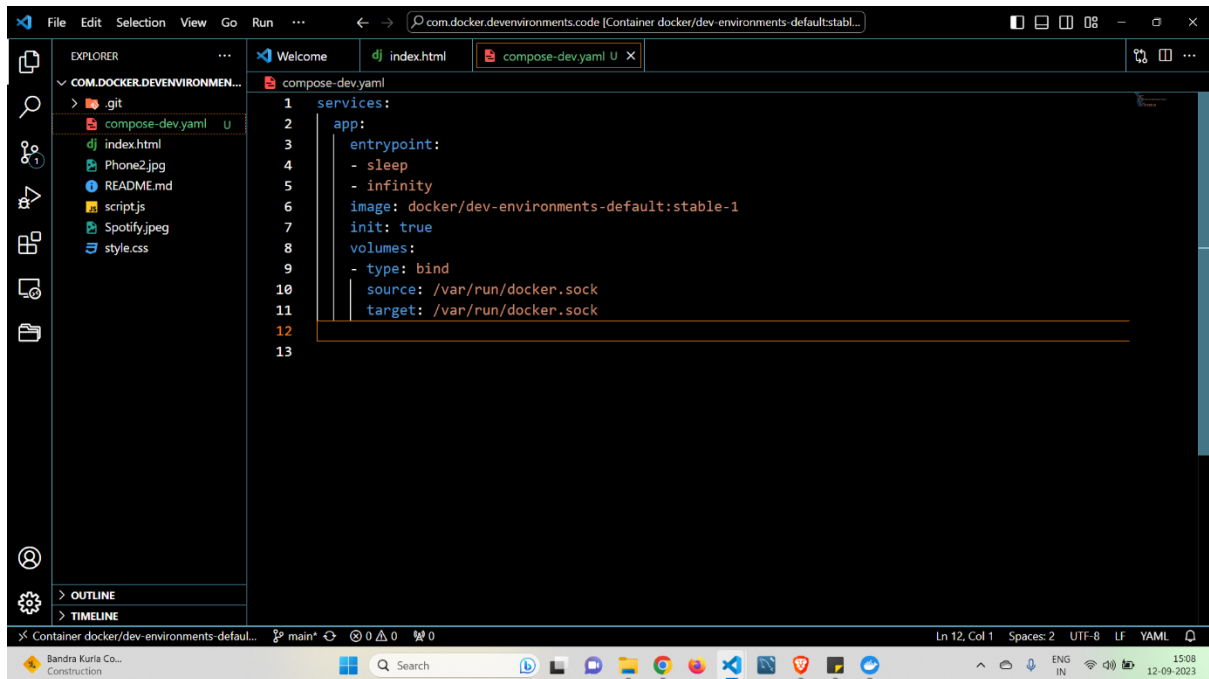
Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 19/10/23



Name : Altaf Alam

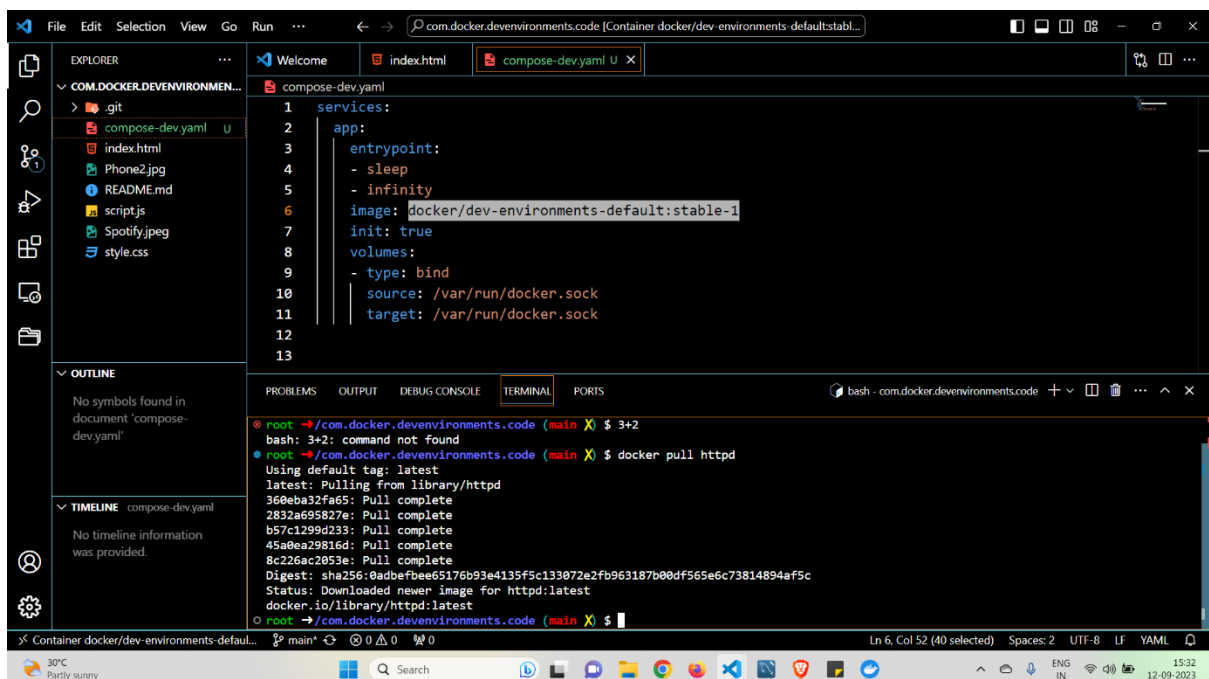
Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 19/10/23



This screenshot shows the Visual Studio Code editor interface. The Explorer panel on the left displays the file structure of a project named 'COM.DOCKER.DEVENVIRONMEN...', including files like '.git', 'compose-dev.yaml', 'index.html', 'Phone2.jpg', 'README.md', 'script.js', 'Spotify.jpeg', and 'style.css'. The main editor area is open to 'compose-dev.yaml', which contains the following YAML configuration:

```
1 services:
2   app:
3     entrypoint:
4       - sleep
5       - infinity
6     image: docker/dev-environments-default:stable-1
7     init: true
8     volumes:
9       - type: bind
10         source: /var/run/docker.sock
11         target: /var/run/docker.sock
```

The status bar at the bottom indicates the active container is 'Container docker/dev-environments-default...' and the current file is 'main*'.



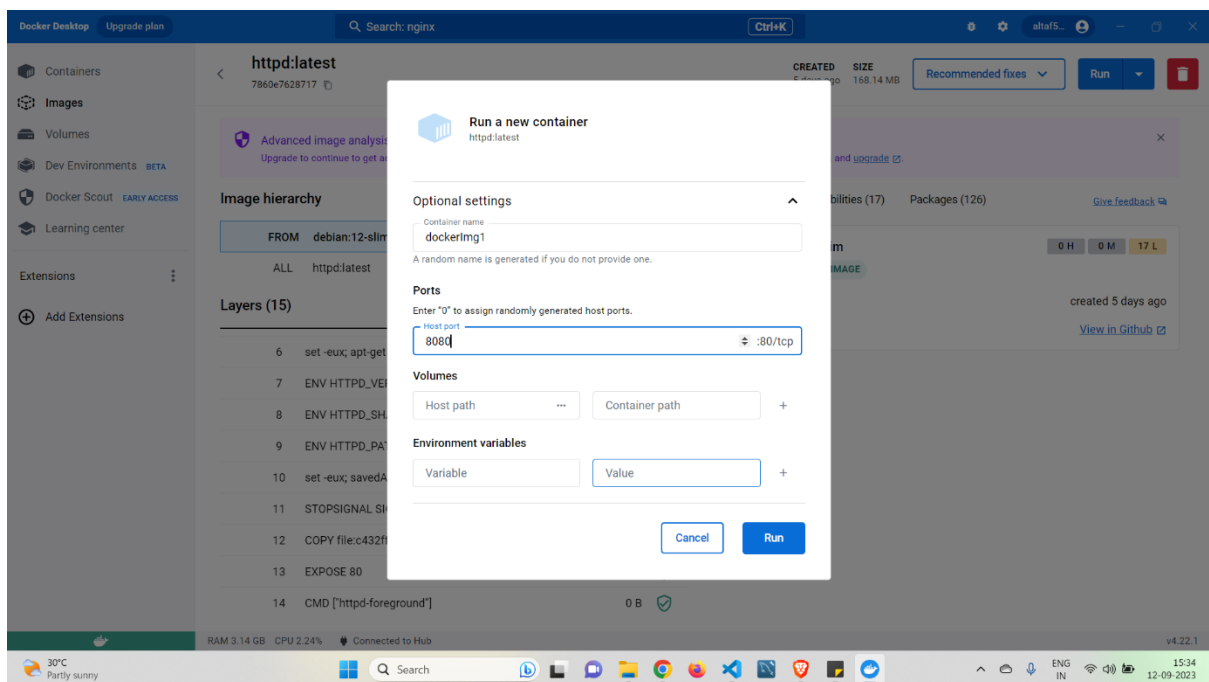
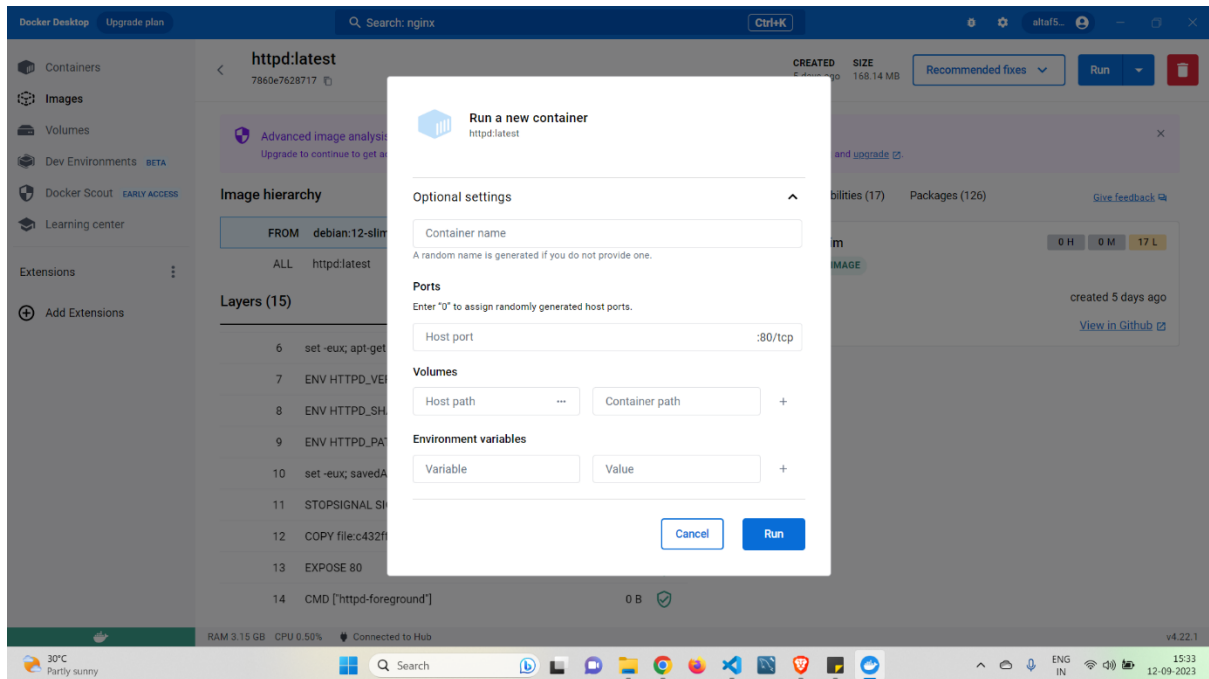
This screenshot shows the same VS Code editor interface, but with the 'TERMINAL' panel open at the bottom. The terminal shows the following commands and output:

```
root → /com.docker.devenvironments.code (main) $ 3+2
bash: 3+2: command not found
root → /com.docker.devenvironments.code (main) $ docker pull httpd
Using default tag: latest
latest: Pulling from library/httpd
360eba32fa65: Pull complete
2832a695827e: Pull complete
b57c1299d233: Pull complete
45a0ea29816d: Pull complete
8c226ac2053e: Pull complete
Digest: sha256:0adbfbee65176b93e4135f5c133072e2fb963187b00df565e6c73814894af5c
Status: Downloaded newer image for httpd:latest
docker.io/library/httpd:latest
root → /com.docker.devenvironments.code (main) $
```

The Explorer panel on the left now shows the 'OUTLINE' and 'TIMELINE' sections, both indicating 'No symbols found in document' and 'No timeline information was provided' for the 'compose-dev.yaml' file.

Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 19/10/23



Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 19/10/23

The screenshot displays the Docker Desktop interface. On the left sidebar, the 'Containers' section is active. The main panel shows a container named 'dockerimg1' (image: httpd:latest) with status 'Running (1 second ago)'. The 'Logs' tab is selected, showing the following log entries:

```
2023-09-12 15:34:25 AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 172.17.0.2. Set the 'ServerName' directive globally to suppress this message
2023-09-12 15:34:25 AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 172.17.0.2. Set the 'ServerName' directive globally to suppress this message
2023-09-12 15:34:25 [Tue Sep 12 10:04:25.270190 2023] [mpm_event:notice] [pid 1:tid 139736767059840] AH00489: Apache/2.4.57 (Unix) configured -- resuming normal operations
2023-09-12 15:34:25 [Tue Sep 12 10:04:25.320816 2023] [core:notice] [pid 1:tid 139736767059840] AH00694: Command line: 'httpd -D FOREGROUND'
```

Below the logs, system metrics are shown: RAM 3.16 GB, CPU 2.25%, and 'Connected to Hub'. The bottom status bar indicates 'v4.22.1'.

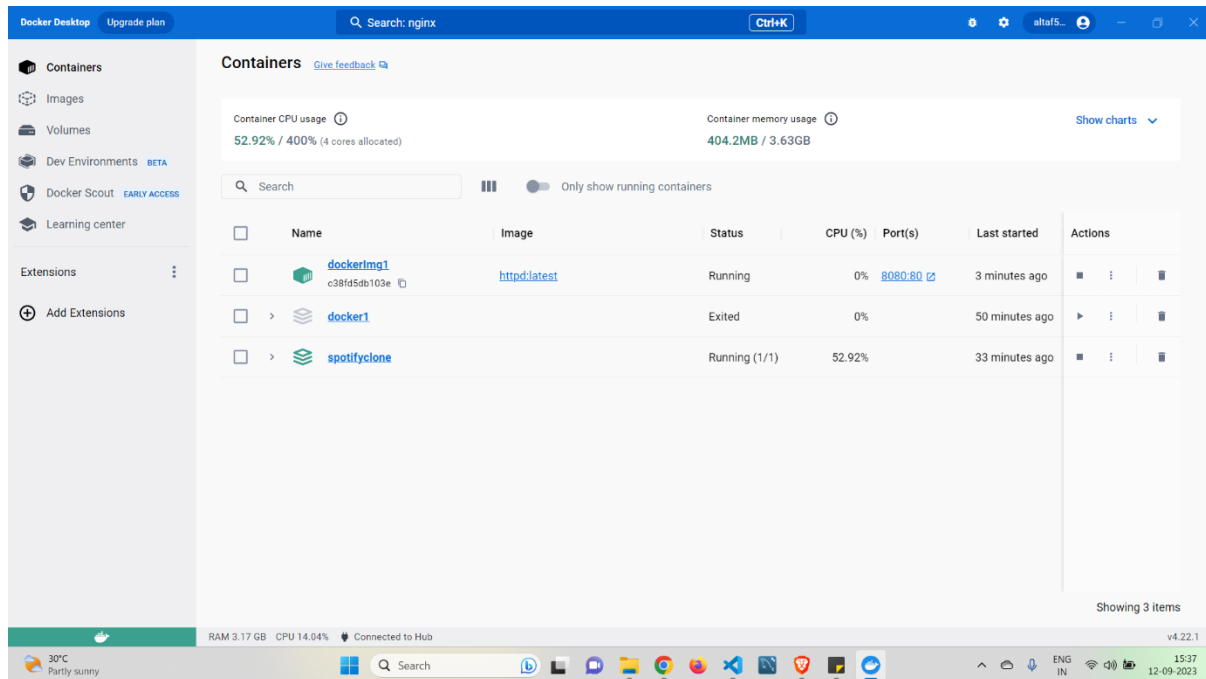
Below the Docker Desktop window, a web browser window is open to 'localhost:8080'. The address bar shows 'localhost:8080' and the page content is not visible.

It works!



Name : Altaf Alam

Roll no : 02 , Batch : T11 , Subject : DevOps Lab , Date : 19/10/23



Conclusion :

In conclusion, deploying a static web application on Docker is a straightforward yet powerful way to leverage containerization for web development. Docker simplifies the process of packaging and deploying your application, providing the benefits of portability, isolation, scalability, and resource efficiency. With Docker, you can streamline the deployment of static web applications and ensure a consistent experience across different environments.