Shreya Kamath / DevOps Lab / T-13 / Roll no. 53

Lab Assignment 7

AIM: To understand Docker architecture and container life cycle, install docker , deploy container in docker.

LAB OUTCOME:
LO1, LO5 Mapped.

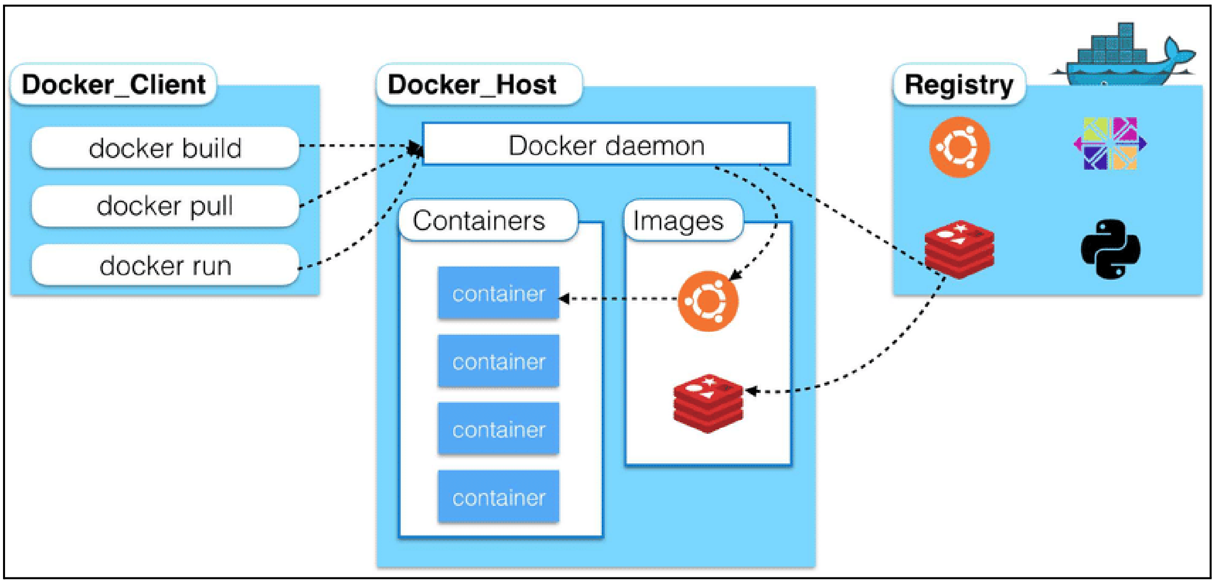
THEORY:
Docker is a technology that allows you to package and run applications and their dependencies in a consistent and isolated environment called a container. Think of it like a shipping container for your software – it contains everything your application needs to run, such as code, libraries, and settings, all bundled together.

Docker Architecture:
1. Docker Engine: This is like the core of Docker. It's a program that runs on your computer or server and manages containers. It consists of the Docker daemon (a background service) and the Docker command-line interface (CLI).

2. Images: Containers start from images. An image is like a blueprint or template for a container. It includes all the files and instructions needed to create a container. Images can be shared and used to create multiple containers.

3. Containers: These are the instances of images. When you run an image, it becomes a container. Containers are isolated environments that contain your application and its dependencies, making sure it runs consistently across different systems.

4. Registry: A registry is like a library of Docker images. Docker Hub is a popular public registry, but you can also set up private registries. You can push (upload) and pull (download) images to/from registries.



Life Cycle of a Container:

1. Create: You start by creating a container from an image using the `docker run` command. This creates an isolated instance of your application.

2. Run: Once created, you can start the container with `docker start`. Your application runs within the container as if it's on its own little computer.

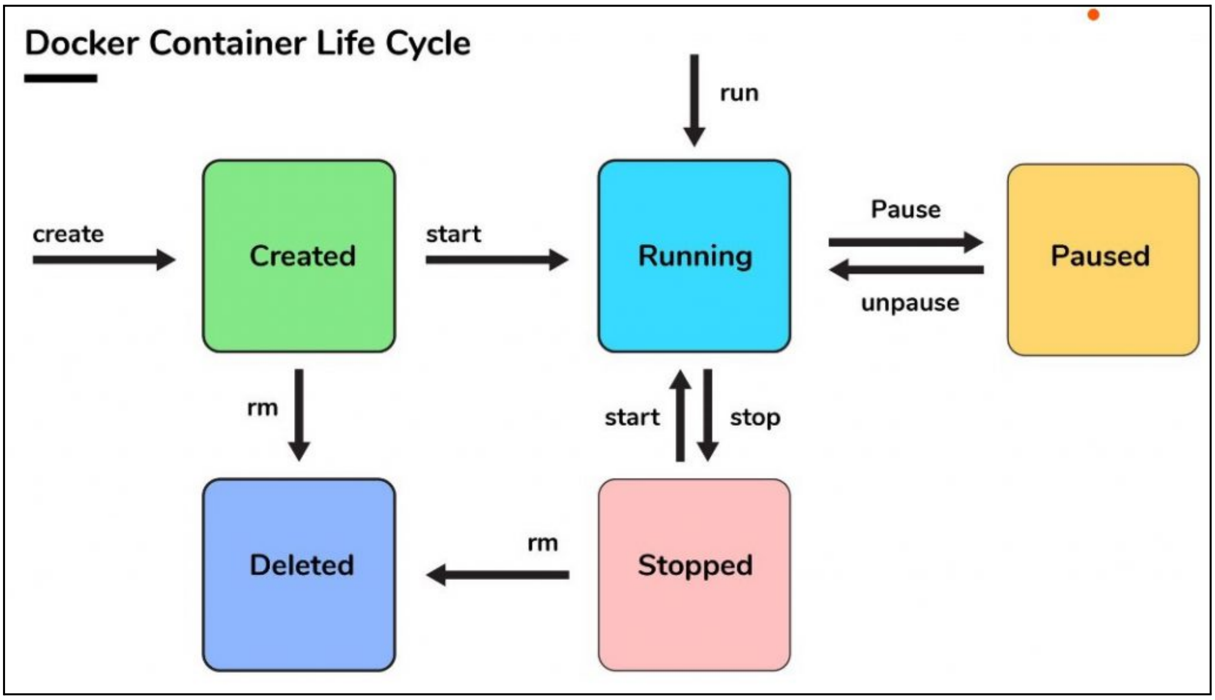
3. Pause and Resume: You can pause a running container with `docker pause` and then resume it with `docker unpause`. This can be handy for saving resources when a container isn't actively in use.

4. Stop: When you're done with a container, you can stop it with `docker stop`. This gracefully shuts down your application.

5. Start: You can later start the container again with `docker start`, and it will resume from where it left off.

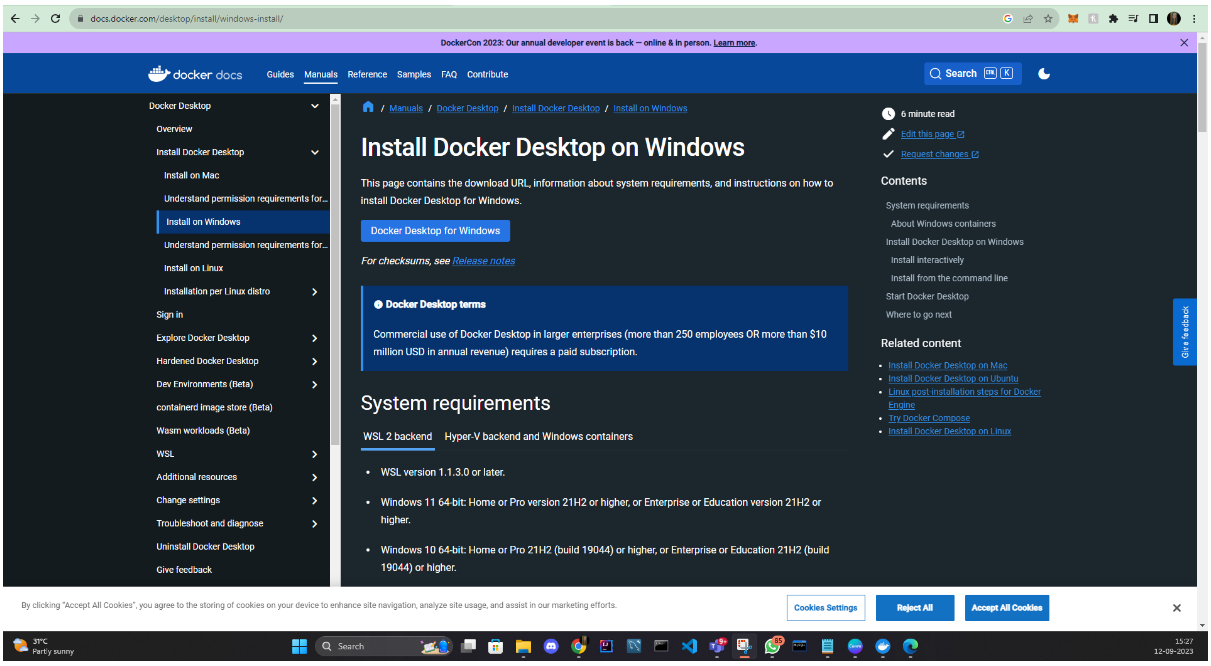
6. Remove: If you no longer need a container, you can remove it with `docker rm`. This deletes the container, but not the image it was created from.

7. Cleanup: You can also clean up unused images with `docker image prune` to free up storage space.



Installation steps of docker with screenshot.

1. Download Docker Desktop for Windows:
- Visit https://www.docker.com/products/docker-desktop and download the installer.



Install Docker Desktop on Windows

This page contains the download URL, information about system requirements, and instructions on how to install Docker Desktop for Windows.

Docker Desktop for Windows

For checksums, see Release notes

Docker Desktop terms

Commercial use of Docker Desktop in larger enterprises (more than 250 employees OR more than $10 million USD in annual revenue) requires a paid subscription.

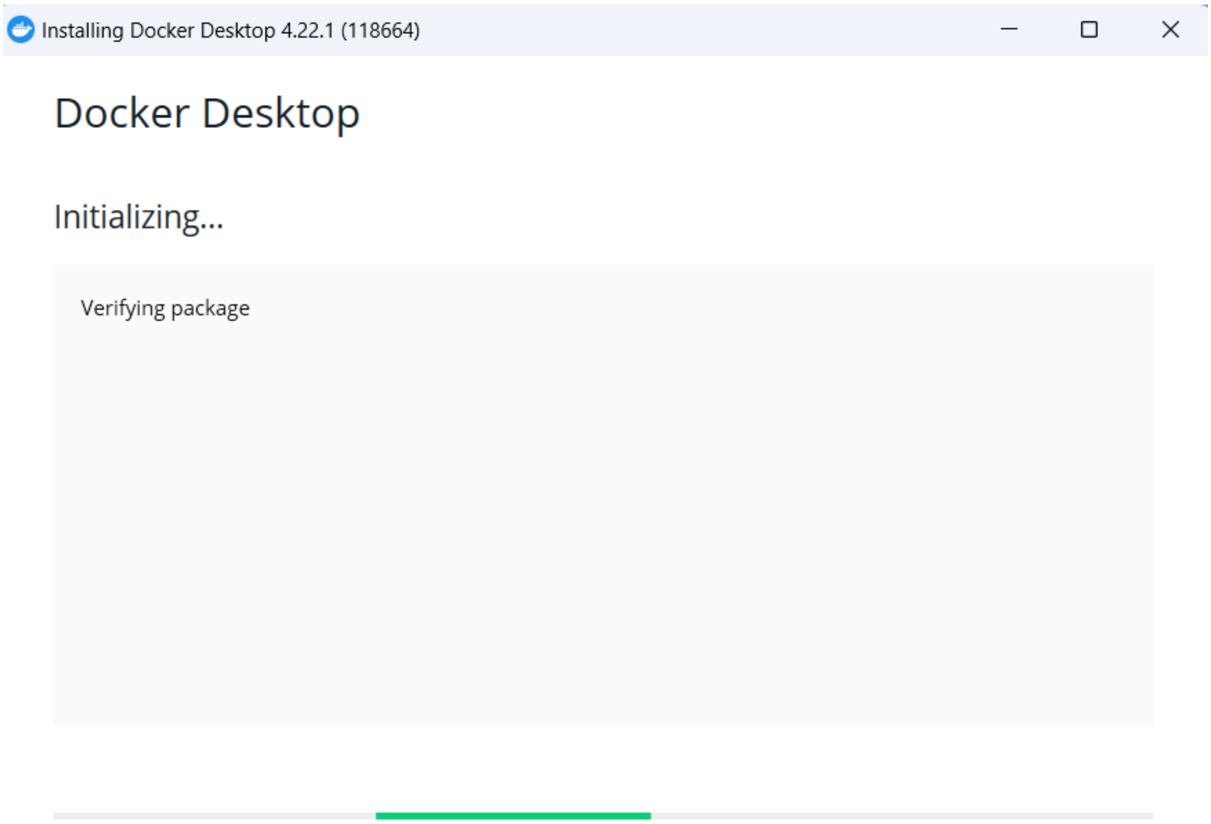
System requirements

WSL 2 backend | Hyper-V backend and Windows containers

- WSL version 1.1.3.0 or later.
- Windows 11 64-bit: Home or Pro version 21H2 or higher, or Enterprise or Education version 21H2 or higher.
- Windows 10 64-bit: Home or Pro 21H2 (build 19044) or higher, or Enterprise or Education 21H2 (build 19044) or higher.

By clicking "Accept All Cookies", you agree to the storing of cookies on your device to enhance site navigation, analyze site usage, and assist in our marketing efforts.

2. Run the Installer:
- Double-click the installer file to begin installation.



Installing Docker Desktop 4.22.1 (118664)

Docker Desktop

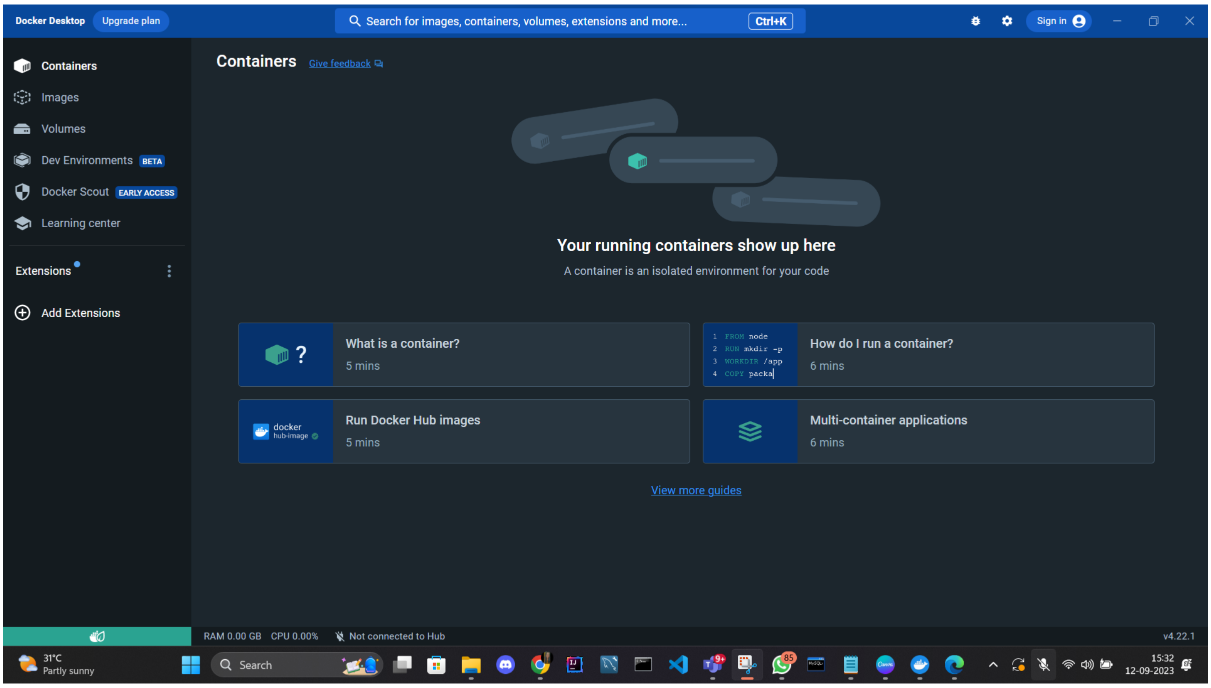
Initializing...

Verifying package

3. Configuration Options:

Shreya Kamath / DevOps Lab / T-13 / Roll no. 53

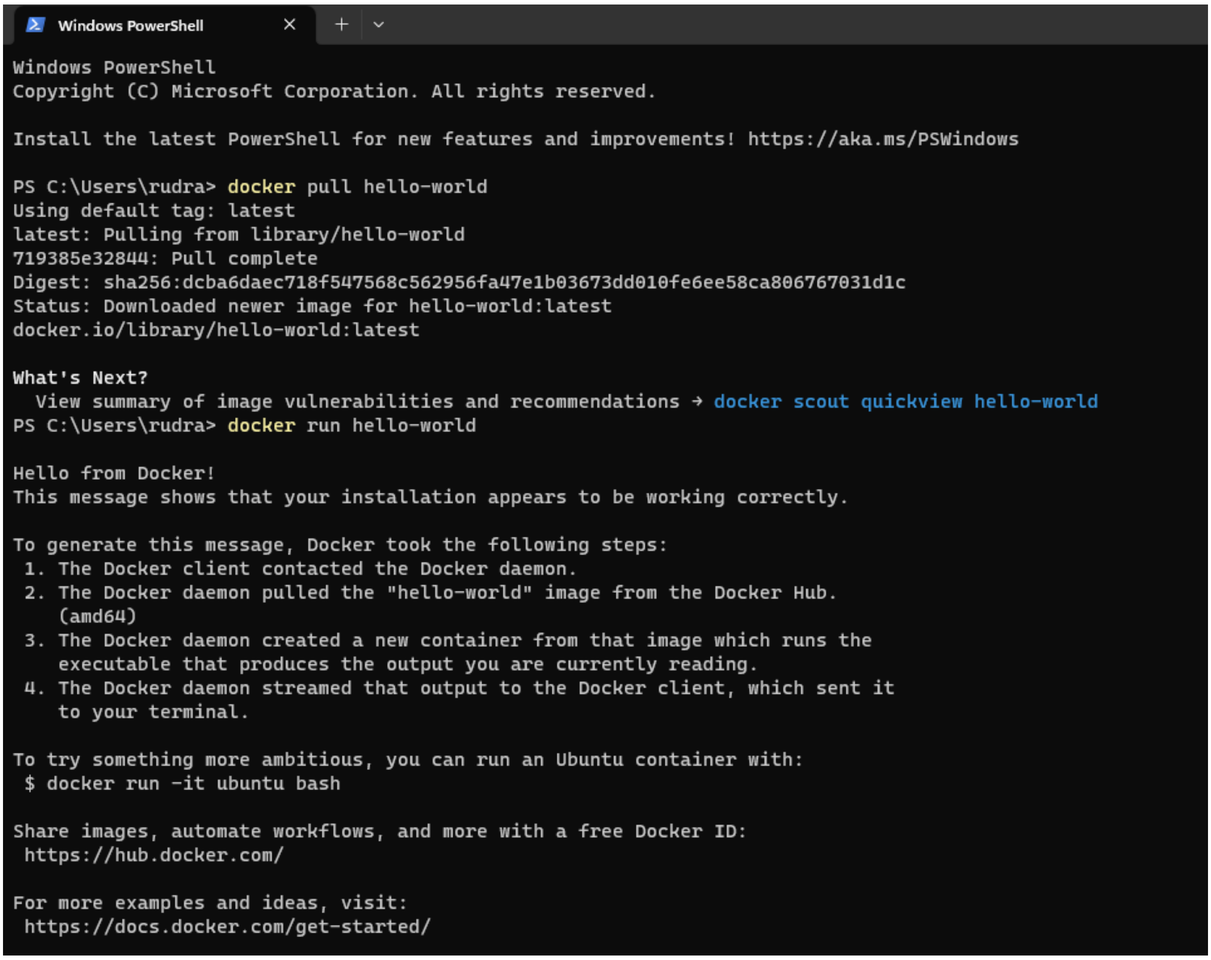
- After installation, access Docker settings by right-clicking the Docker icon on the desktop. Now, you're ready to configure Docker Desktop for Windows.



Your running containers show up here

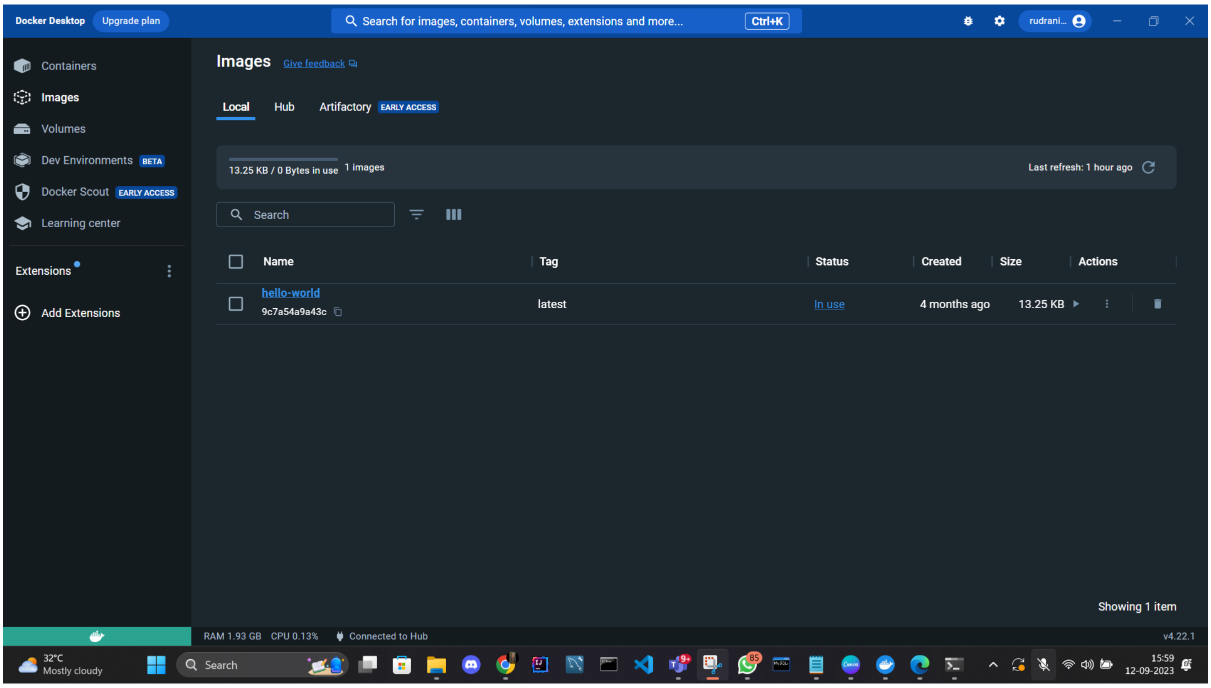
A container is an isolated environment for your code

```
FROM node
RUN mkdir -p
WORKDIR /app
COPY packa
```



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\rudra> docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
719385e32844: Pull complete
Digest: sha256:dcba6daec718f547568c562956fa47e1b03673dd010fe6ee58ca806767031d1c
Status: Downloaded newer image for hello-world:latest
docker.io/library/hello-world:latest

What's Next?
  View summary of image vulnerabilities and recommendations → docker scout quickview hello-world
PS C:\Users\rudra> docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

Shreya Kamath / DevOps Lab / T-13 / Roll no. 53



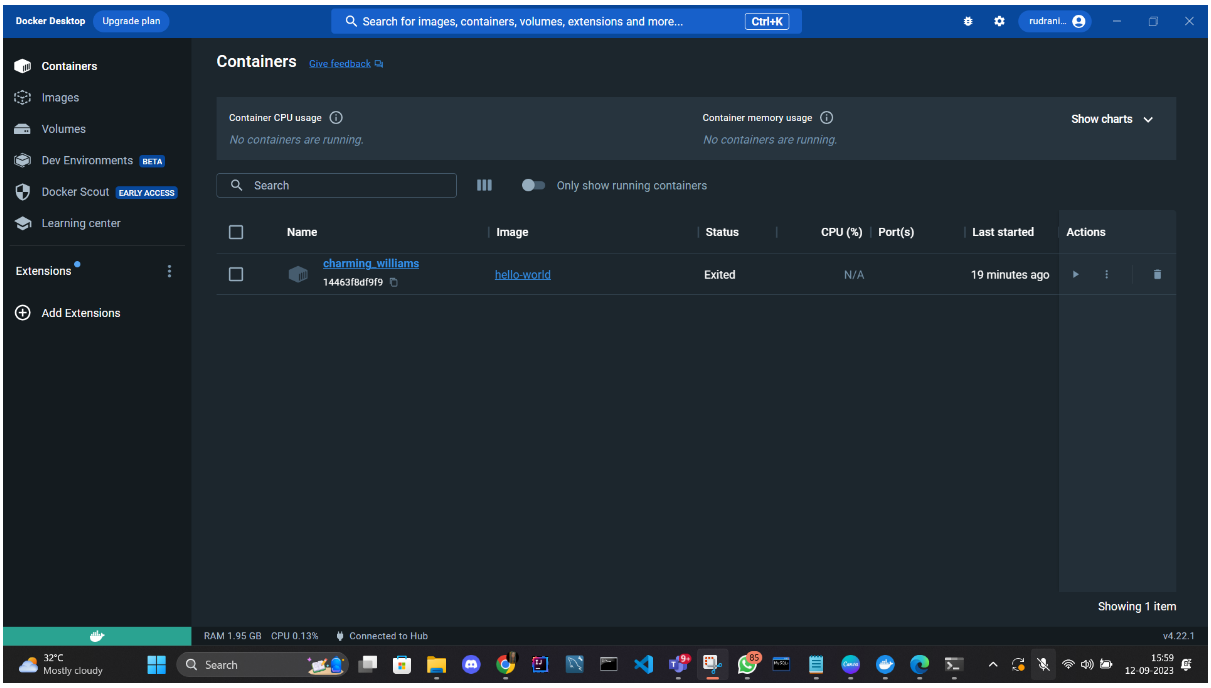
Images

13.25 KB / 0 Bytes in use 1 images

Last refresh: 1 hour ago

Name	Tag	Status	Created	Size	Actions
hello-world 9c7a54a9a43c	latest	In use	4 months ago	13.25 KB	

Showing 1 item



Containers

Container CPU usage

No containers are running.

Container memory usage

No containers are running.

Only show running containers

Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
charming_williams 14463f8df9f9	hello-world	Exited	N/A		19 minutes ago	

Showing 1 item

CONCLUSION:

Docker simplifies the process of developing, testing, and deploying applications because it ensures that what works on your development machine will also work in other environments, like a production server, without the "it works on my machine" problem. It's especially valuable in modern software development and deployment workflows, where consistency and scalability are essential.

Shreya Kamath / DevOps Lab / T-13 / Roll no. 53

LAB ASSIGNMENT 8

AIM: : Deploy static web application on docker.

LAB OUTCOME:
LO1, LO5 Mapped.

THEORY:
To deploy a static web application on Docker, you can follow these steps:

1. Install Docker Desktop:
If you haven't already, download and install Docker Desktop for Windows. You can get it from the official Docker website: https://www.docker.com/products/docker-desktop

2. Verify Docker Installation:
After installation, open Docker Desktop to ensure that it's running correctly. You should see the Docker icon in your system tray.

3. Create a Dockerfile:
Create a Dockerfile in the root directory of your web application. This file is used to define how your application should be built and run within a Docker container. Here's a simple example of a Dockerfile for a static web application:

```
 Dockerfile
   # Use an official Nginx image as the base image
   FROM nginx:alpine

   # Copy your static web application files to the container
   COPY ./path/to/your/app /usr/share/nginx/html

   # Expose port 80 to the host
   EXPOSE 80
```

4. Build the Docker Image:
Open a terminal and navigate to the directory containing your Dockerfile. Run the following command to build a Docker image:

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

Replace `my-web-app` with your desired image name, and don't forget the period at the end, which indicates the current directory.

5. Run the Docker Container:
After building the image, you can start a Docker container based on that image using the following command:

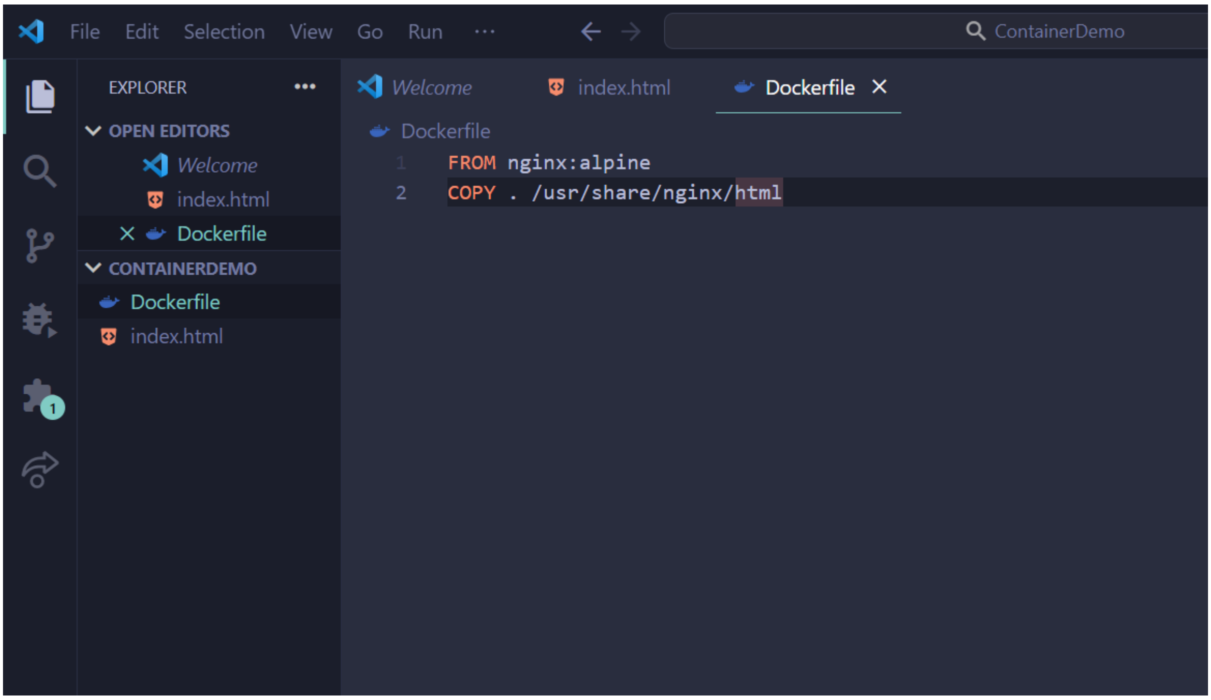
```
docker run -d -p 8080:80 my-web-app
```

This command runs the container in detached mode (`-d`) and maps port 8080 on your host to port 80 in the container. You can choose a different port if you like.

6. Access Your Web Application:
Open a web browser and navigate to `http://localhost:8080` (or the port you specified in step 5). You should be able to access your static web application running inside the Docker container.

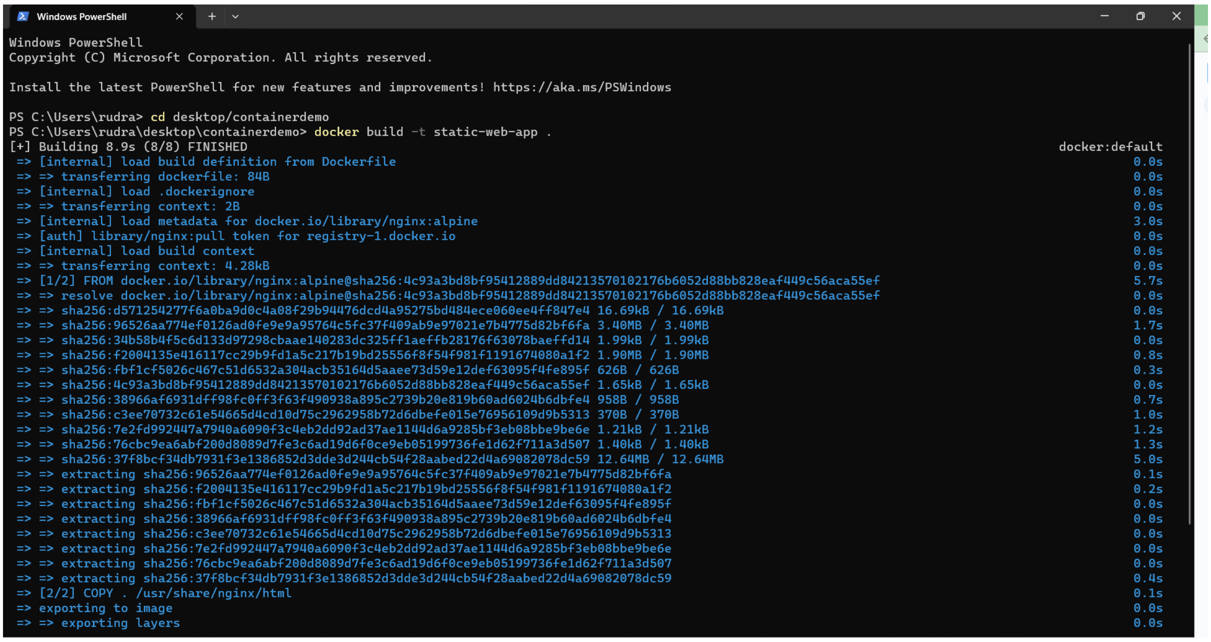
7. Manage Docker Containers:
You can manage your Docker containers using Docker commands like `docker ps` to list running containers, `docker stop <container_id>` to stop a container, and `docker rm <container_id>` to remove a container.

SCREENSHOTS:

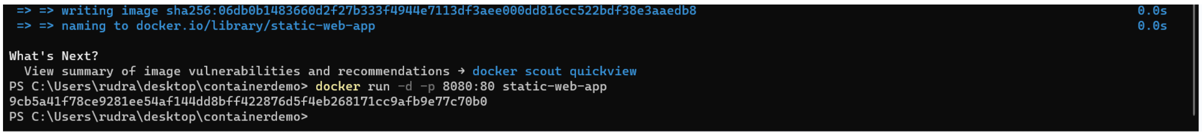


```
FROM nginx:alpine
COPY . /usr/share/nginx/html
```

Shreya Kamath / DevOps Lab / T-13 / Roll no. 53

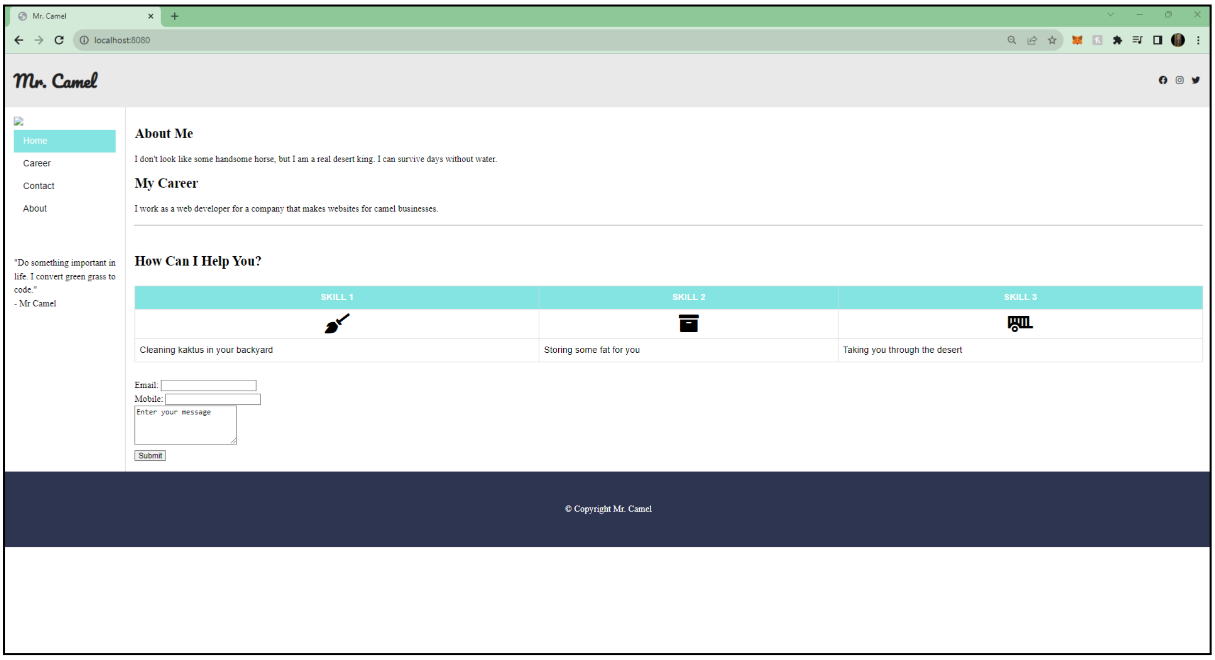


```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\rudra> cd desktop/containerdemo
PS C:\Users\rudra\desktop\containerdemo> docker build -t static-web-app .
[+] Building 8.9s (8/8) FINISHED                                                                                                            docker:default
 => [internal] load build definition from Dockerfile                                                                                                  0.0s
 => => transferring dockerfile: 84B                                                                                                                   0.0s
 => [internal] load .dockerignore                                                                                                                     0.0s
 => => transferring context: 2B                                                                                                                       0.0s
 => [internal] load metadata for docker.io/library/nginx:alpine                                                                                       3.0s
 => [auth] library/nginx:pull token for registry-1.docker.io                                                                                          0.0s
 => [internal] load build context                                                                                                                     0.0s
 => => transferring context: 4.28kB                                                                                                                   0.0s
 => [1/2] FROM docker.io/library/nginx:alpine@sha256:4c93a3bd8bf95412889dd84213570102176b6052d88bb828eaf449c56aca55ef                                 5.7s
 => => resolve docker.io/library/nginx:alpine@sha256:4c93a3bd8bf95412889dd84213570102176b6052d88bb828eaf449c56aca55ef                                 0.0s
 => => sha256:d571254277f6a0ba9d0c4a08f29b94476dcd4a95275bd484ece060ee4ff847e4 16.69kB / 16.69kB                                                      0.0s
 => => sha256:96526aa774ef0126ad0fe9e9a95764c5fc37f409ab9e97021e7b4775d82bf6fa 3.40MB / 3.40MB                                                        1.7s
 => => sha256:34b58b4f5c6d133d97298cbaae140283dc325ff1aeffb28176f63078baeffd14 1.99kB / 1.99kB                                                        0.0s
 => => sha256:f2004135e416117cc29b9fd1a5c217b19bd25556f8f54f981f1191674080a1f2 1.90MB / 1.90MB                                                        0.8s
 => => sha256:fbf1cf5026c467c51d6532a304acb35164d5aaee73d59e12def63095f4fe895f 626B / 626B                                                            0.3s
 => => sha256:4c93a3bd8bf95412889dd84213570102176b6052d88bb828eaf449c56aca55ef 1.65kB / 1.65kB                                                        0.0s
 => => sha256:38966af6931dff98fc0ff3f63f490938a895c2739b20e819b60ad6024b6dbfe4 958B / 958B                                                            0.7s
 => => sha256:c3ee70732c61e54665d4cd10d75c2962958b72d6dbefe015e76956109d9b5313 370B / 370B                                                            1.0s
 => => sha256:7e2fd992447a7940a6090f3c4eb2dd92ad37ae1144d6a9285bf3eb08bbe9be6e 1.21kB / 1.21kB                                                        1.2s
 => => sha256:76cbc9ea6abf200d8089d7fe3c6ad19d6f0ce9eb05199736fe1d62f711a3d507 1.40kB / 1.40kB                                                        1.3s
 => => sha256:37f8bcf34db7931f3e1386852d3dde3d244cb54f28aabed22d4a69082078dc59 12.64MB / 12.64MB                                                      5.0s
 => => extracting sha256:96526aa774ef0126ad0fe9e9a95764c5fc37f409ab9e97021e7b4775d82bf6fa                                                             0.1s
 => => extracting sha256:f2004135e416117cc29b9fd1a5c217b19bd25556f8f54f981f1191674080a1f2                                                             0.2s
 => => extracting sha256:fbf1cf5026c467c51d6532a304acb35164d5aaee73d59e12def63095f4fe895f                                                             0.0s
 => => extracting sha256:38966af6931dff98fc0ff3f63f490938a895c2739b20e819b60ad6024b6dbfe4                                                             0.0s
 => => extracting sha256:c3ee70732c61e54665d4cd10d75c2962958b72d6dbefe015e76956109d9b5313                                                             0.0s
 => => extracting sha256:7e2fd992447a7940a6090f3c4eb2dd92ad37ae1144d6a9285bf3eb08bbe9be6e                                                             0.0s
 => => extracting sha256:76cbc9ea6abf200d8089d7fe3c6ad19d6f0ce9eb05199736fe1d62f711a3d507                                                             0.0s
 => => extracting sha256:37f8bcf34db7931f3e1386852d3dde3d244cb54f28aabed22d4a69082078dc59                                                             0.4s
 => [2/2] COPY . /usr/share/nginx/html                                                                                                                0.1s
 => exporting to image                                                                                                                                0.0s
 => => exporting layers                                                                                                                               0.0s
```



```
 => => writing image sha256:06db0b1483660d2f27b333f4944e7113df3aee000dd816cc522bdf38e3aaedb8                                                           0.0s
 => => naming to docker.io/library/static-web-app                                                                                                      0.0s

What's Next?
  View summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\Users\rudra\desktop\containerdemo> docker run -d -p 8080:80 static-web-app
9cb5a41f78ce9281ee54af144dd8bff422876d5f4eb268171cc9afb9e77c70b0
PS C:\Users\rudra\desktop\containerdemo>
```

Shreya Kamath / DevOps Lab / T-13 / Roll no. 53



Mr. Camel

localhost:8080

Mr. Camel

- Home
- Career
- Contact
- About

"Do something important in life. I convert green grass to code."
- Mr Camel

About Me

I don't look like some handsome horse, but I am a real desert king. I can survive days without water.

My Career

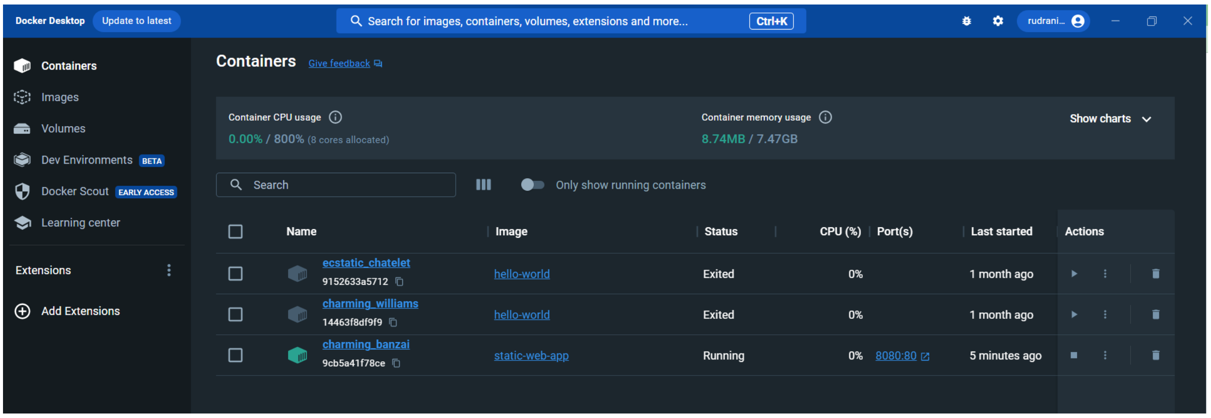
I work as a web developer for a company that makes websites for camel businesses.

How Can I Help You?

SKILL 1	SKILL 2	SKILL 3
Cleaning kaktus in your backyard	Storing some fat for you	Taking you through the desert

Email:
Mobile:
Enter your message
Submit

© Copyright Mr. Camel



Docker Desktop Update to latest

Search for images, containers, volumes, extensions and more... Ctrl+K

rudrani...

Containers Give feedback

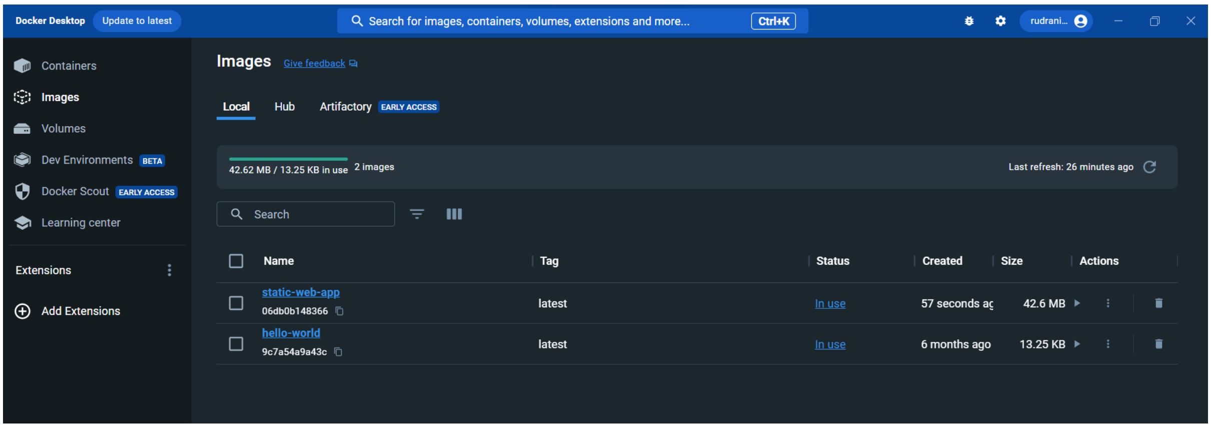
Container CPU usage
0.00% / 800% (8 cores allocated)

Container memory usage
8.74MB / 7.47GB

Show charts

Search Only show running containers

Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
ecstatic_chatelet 9152633a5712	hello-world	Exited	0%		1 month ago	
charming_williams 14463f8df9f9	hello-world	Exited	0%		1 month ago	
charming_banzai 9cb5a41f78ce	static-web-app	Running	0%	8080:80	5 minutes ago	



Docker Desktop Update to latest

Search for images, containers, volumes, extensions and more... Ctrl+K

rudrani...

Images Give feedback

Local Hub Artifactory EARLY ACCESS

42.62 MB / 13.25 KB in use 2 images

Last refresh: 26 minutes ago

Search

Name	Tag	Status	Created	Size	Actions
static-web-app 06db0b148366	latest	In use	57 seconds ag	42.6 MB	
hello-world 9c7a54a9a43c	latest	In use	6 months ago	13.25 KB	

CONCLUSION:

In summary, deploying a static web application on Docker in Windows 11 is a straightforward process. By installing Docker Desktop, creating a Dockerfile, building an image, and running a container, you can host your web app with ease. Managing containers and cleaning up resources is also manageable, making it an efficient and scalable solution for web application deployment.