**Aim:** To understand Docker Architecture and Container Life Cycle, install Docker and execute docker commands to manage images and interact with containers**.**

Objective: Objective is Docker's architecture and the lifecycle of containers, along with mastering essential Docker commands for managing images and interacting with containers, individuals can leverage Docker's capabilities to package, distribute, and run applications consistently across different environments

Theory: Docker is a containerization technology that allows developers to package their applications and dependencies into lightweight, portable containers. These containers can then be run on any system that supports Docker, making it easier to deploy and manage applications in various environments.
Docker Architecture:
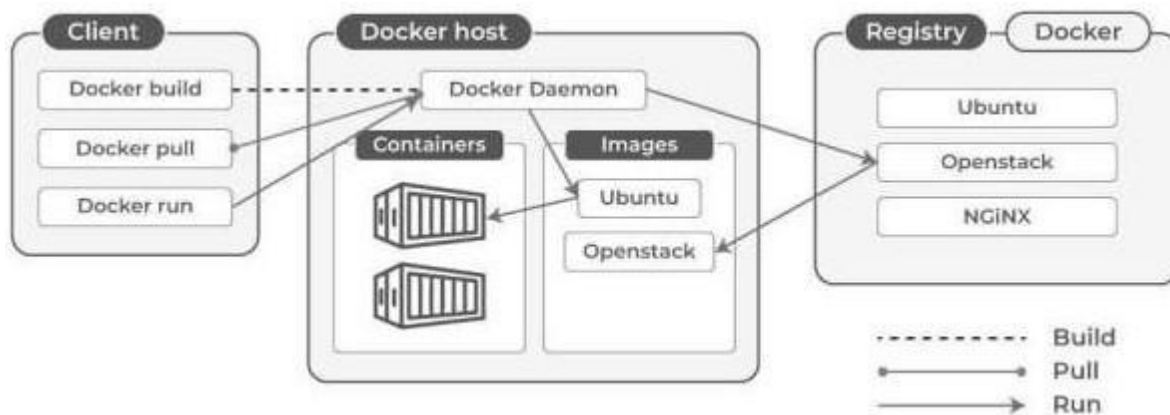
The Docker architecture consists of the following components:
Docker Client: This is the primary interface that developers use to interact with Docker. It sends commands to the Docker daemon and displays the output.
Docker Daemon: This is the background process that runs on the host machine and manages Docker objects such as images, containers, networks, and volumes.
Docker Registry: This is a storage and distribution system for Docker images. It allows developers to share and distribute their images with others.
**Docker Image:** This is a lightweight, standalone, executable package that includes everything needed to run an application, including the application code, runtime, libraries, dependencies, and system tools.
Docker Container: This is a runtime instance of a Docker image. It is isolated from the host system and other containers, providing a secure and predictable environment for the application to run in.

Container LifeCycle:

Docker Container Life Cycle
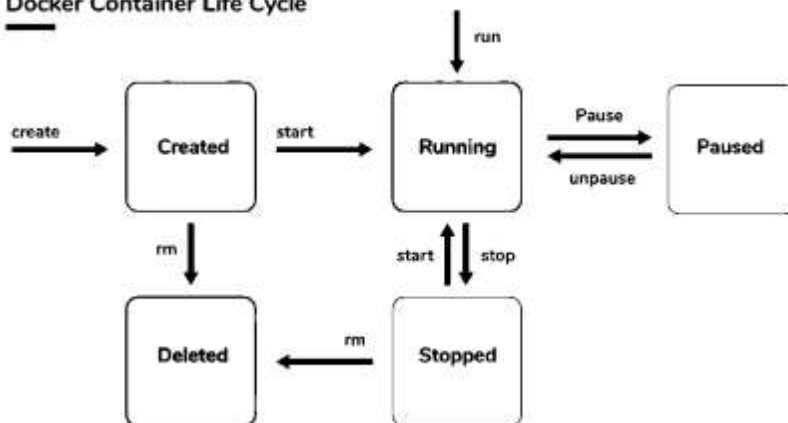
The lifecycle of a Docker container consists of the following stages:

Create: To create a container, you start by creating an image that includes all the necessary components to run the application. This image is then used to create the container.

Start: Once the container is created, you can start it using the docker start command. This launches the container and runs the application inside it.

Run: Once the container is started, you can interact with the application inside the container. You can run commands, access files, and make changes to the application.

Stop: When you're done using the container, you can stop it using the docker stop command. This stops the application inside the container and shuts down the container.

Remove: Finally, when you no longer need the container, you can remove it using the docker rm command. This removes the container from the system and frees up any resources it was using.
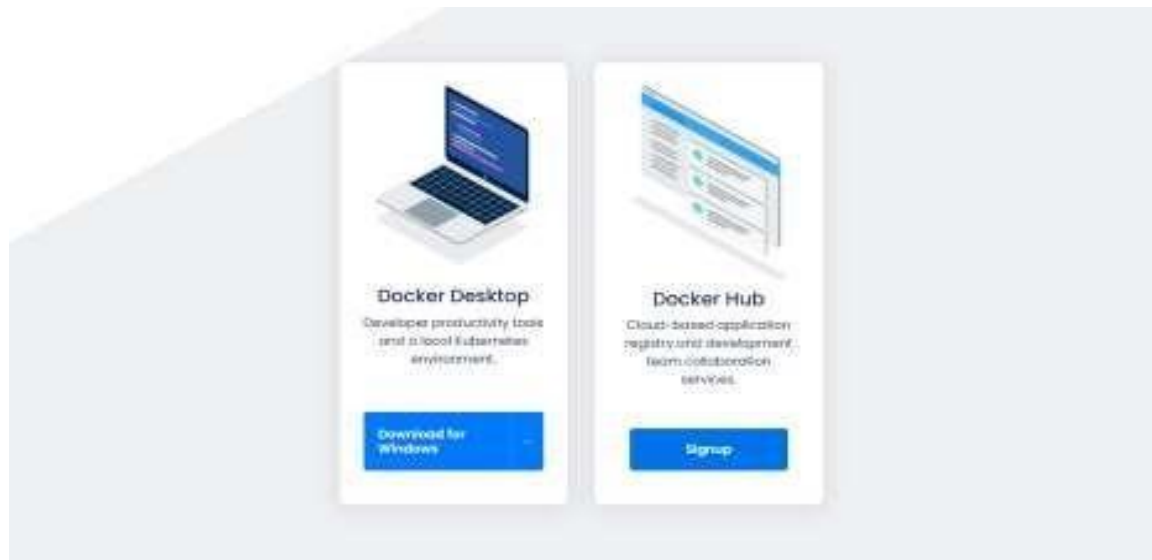
Steps for Installation:

Step 01: Open docker.com

Scroll down, Click on 'Get started for free' tab.
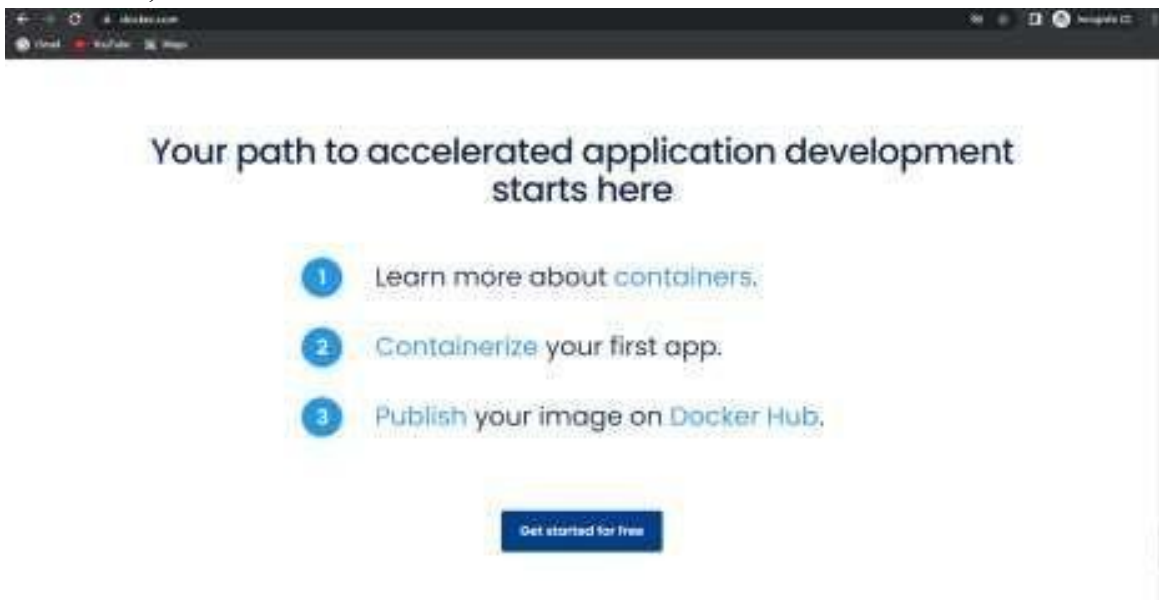
Step 02: Click on Docker Desktop, Download it.

Installing Docker Desktop 4.6.0 (75818)

Docker Desktop 4.6.0

Unpacking files...

Unpacking file: resources/docker-desktop.iso
Unpacking file: resources/ddvp.ico
Unpacking file: resources/config-options.json
Unpacking file: resources/componentsVersion.json
Unpacking file: resources/bin/docker-compose
Unpacking file: resources/bin/docker
Unpacking file: resources/.gitignore
Unpacking file: InstallerCli.pdb
Unpacking file: InstallerCli.exe.config
Unpacking file: frontend/vk_swiftshader_icd.json
Unpacking file: frontend/v8_context_snapshot.bin
Unpacking file: frontend/snapshot_blob.bin
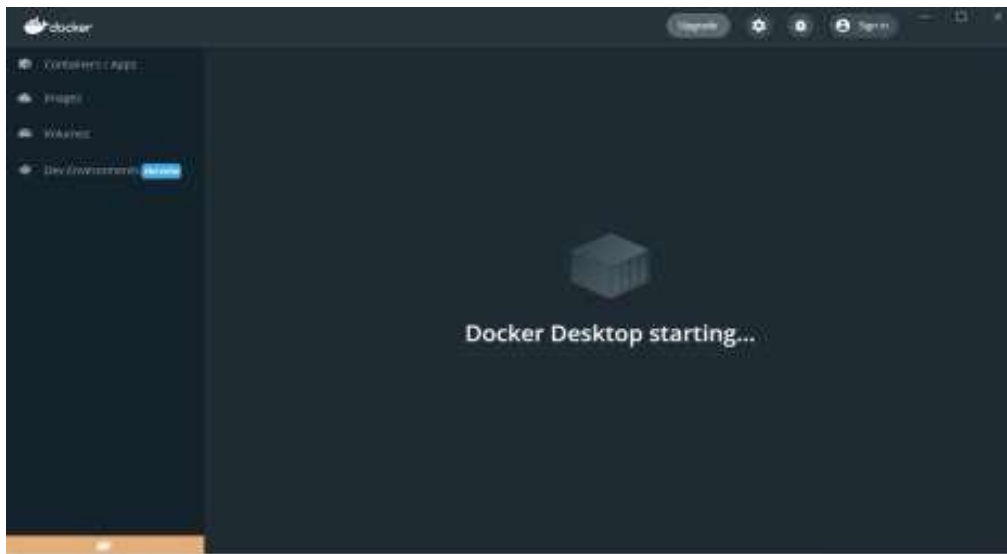Unpacking file: frontend/resources/regedit/vbs/util.vbs

Step 04: After Installation, Restart your device.
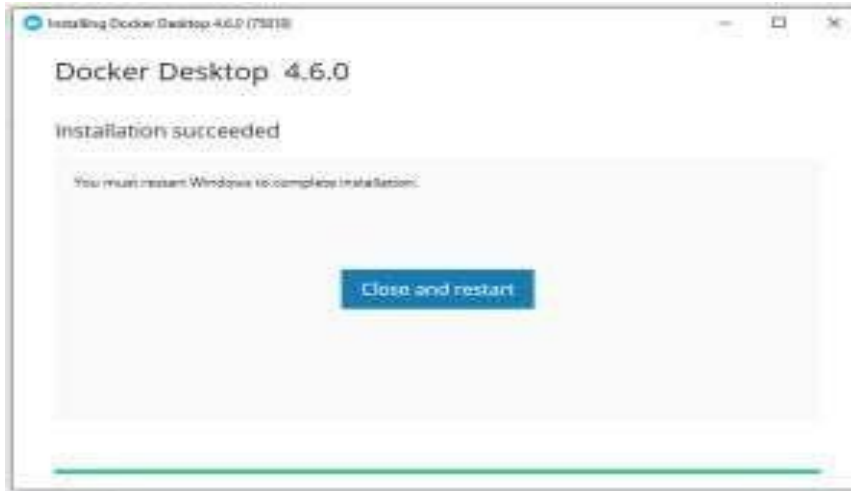
Step 05: Accept the terms and conditions, Click on Accept.

The following window should pop up.
Click on the link - https://aka.ms/wsl2kernel.
(Do not close this window).

Download the WSL2 Linux kernel update package for x64 machines.

Step 4 - Download the Linux kernel update package

1. Download the latest package:

   - WSL2 Linux kernel update package for x64 machines

   ⓘ Note

   If you're using an ARM64 machine, please download the **ARM64 package** instead. If you're not sure what kind of machine you have, open Command Prompt or PowerShell and enter: `systeminfo | find "System Type"`. **Caveat:** On non-English Windows versions, you might have to modify the search text, translating the "System Type" string. You may also need to escape the quotations for the find command. For example, in German `systeminfo | find '"Systemtyp"'`.

2. Run the update package downloaded in the previous step. (Double-click to run - you will be prompted for elevated permissions, select 'yes' to approve this installation.)
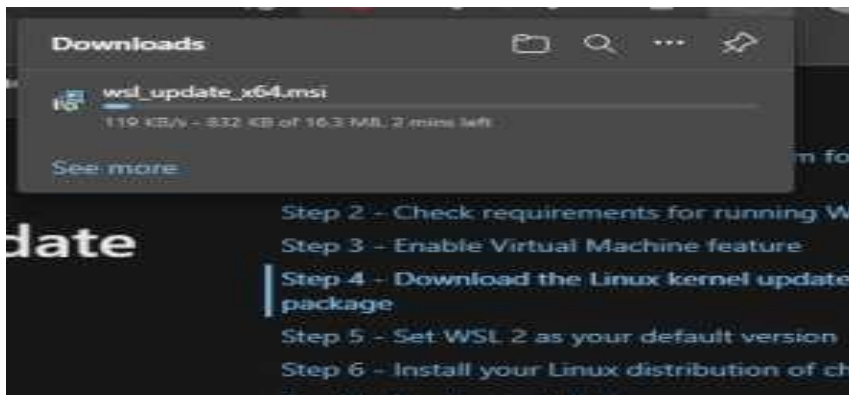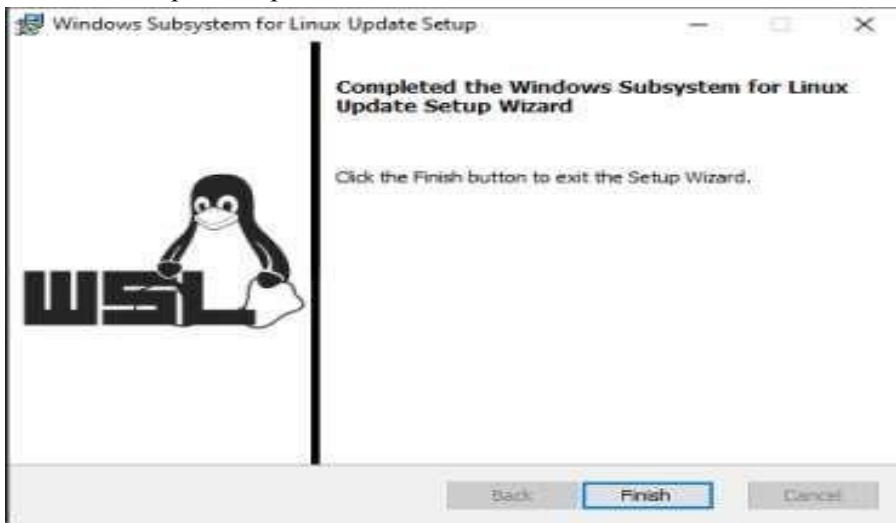

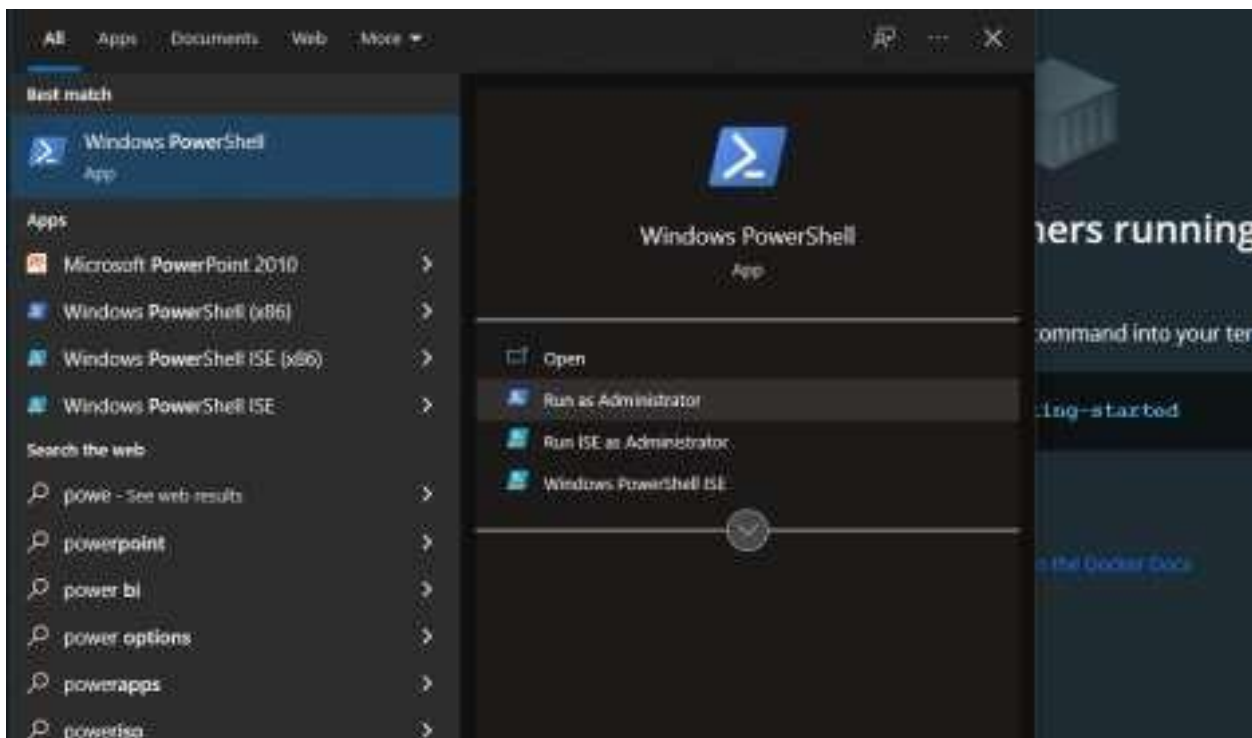After Download is complete, Run the .msipackage.

Click on next.
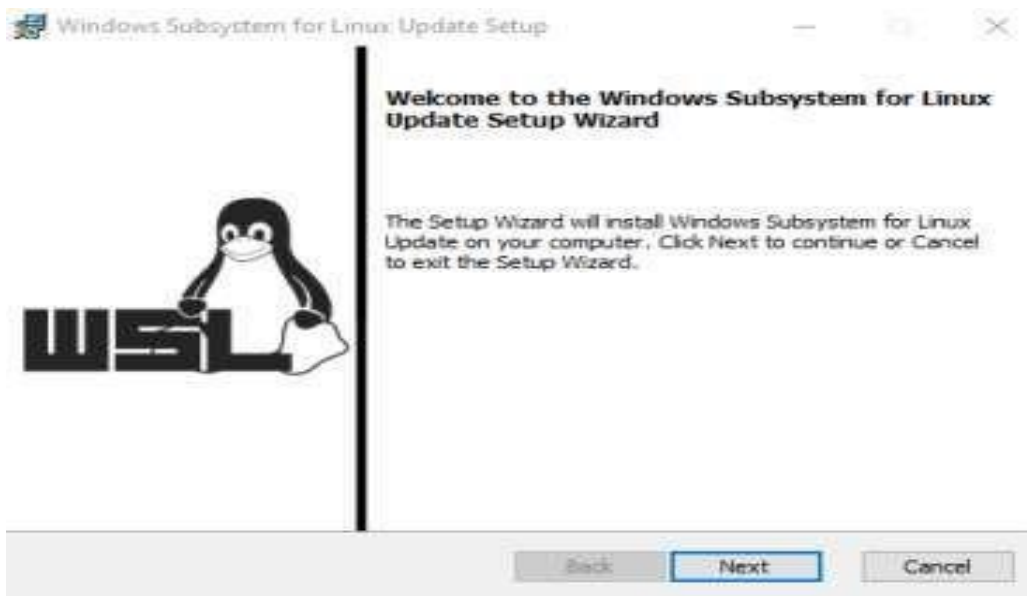
After, the setup is complete, Click on finish.

Open Powershell as an Administrator.

Run the following Command:
wsl --set-default-version 2

```
PS C:\WINDOWS\system32> wsl --set-default-version 2
For information on key differences with WSL 2 please visit https://aka.ms/wsl2
The operation completed successfully.
PS C:\WINDOWS\system32> _
```

Now, Click on Restart

WSL 2 installation is incomplete.

The WSL 2 Linux kernel is now installed using a separate MSI update package.
Please click the link and follow the instructions to install the kernel update:
https://aka.ms/wsl2kernel.

Press Restart after installing the Linux kernel.

Docker should now restart.
Click on Start.

**Get started with Docker
in a few easy steps!**

ESTIMATED TIME: 2 minutes
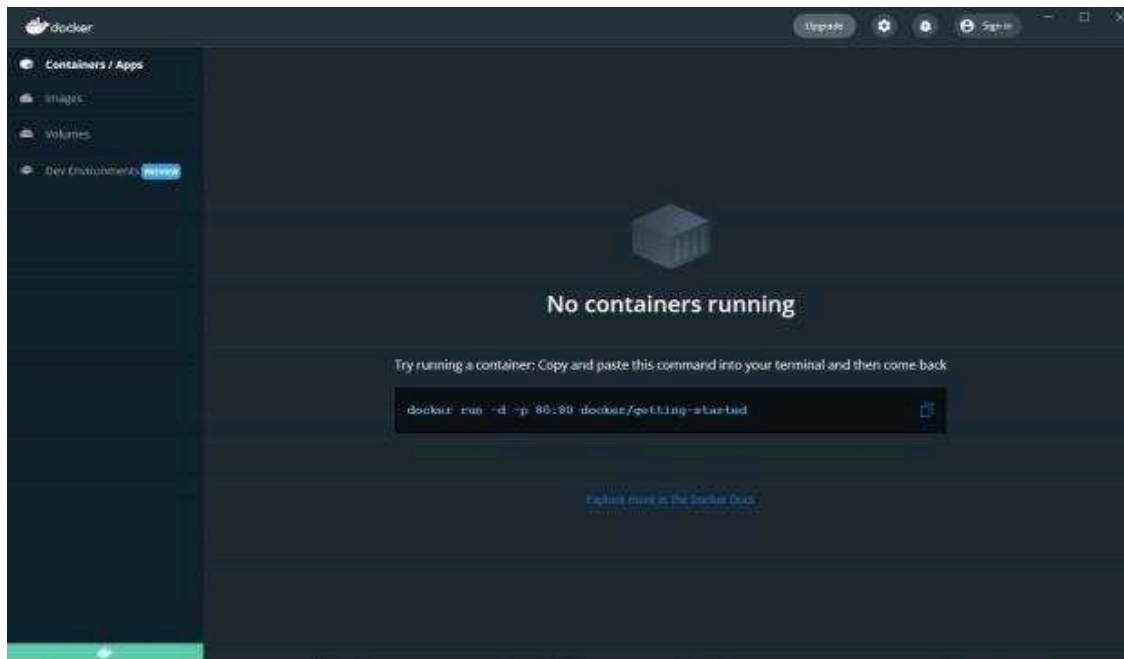
Start

Skip tutorial

The following window should pop up.
This means, Installation is now complete.

Output:

Open Command Prompt, run the following commands:

1)   To check the version of

Docker: docker --version

2)   To install image of

ubuntu docker pull ubuntu

```
PS C:\Users\Lenovo> docker --version
Docker version 20.10.13, build a224086
PS C:\Users\Lenovo> docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
4d32b49e2995: Pull complete
Digest: sha256:bea6d19168bbfd6af8d77c2cc3c572114eb5d113e6f422573c93cb605a0e2ffb
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest
PS C:\Users\Lenovo>
```

3)   Check downloaded

images, docker images

```
PS C:\Users\Lenovo> docker images
REPOSITORY    TAG        IMAGE ID       CREATED       SIZE
ubuntu        latest     ff0fea8310f3   4 days ago    72.8MB
PS C:\Users\Lenovo> _
```

4) Run ubuntu OS

docker run -it ubuntu /bin/bash

```
PS C:\Users\Lenovo> docker run -it ubuntu /bin/bash
root@f45775828da6:/# _
```

5) Open another Command Prompt and follow the steps shown below.

-docker ps

docker container ls –a

docker container rm b71e3e6b1118 //copy docker id for remove but first (Use your container ID in the above command)

stop your docker

- docker container stop b71e3e6b1118

- docker container rm b71e3e6b1118

- docker ps

- docker //list all docker commands

- docker images

-        docker image rm ff0fea8310f3 // copy image id from previous

output (Use your image ID in the above command)

- docker run -it ubuntu /bin/bash //check output

```
PS C:\Users\Lenovo> docker ps
CONTAINER ID   IMAGE    COMMAND      CREATED          STATUS            PORTS     NAMES
f45775828da6   ubuntu   "/bin/bash"  About a minute ago  Up About a minute          nostalgic_elion
PS C:\Users\Lenovo> docker container ls -a
CONTAINER ID   IMAGE    COMMAND      CREATED          STATUS            PORTS     NAMES
f45775828da6   ubuntu   "/bin/bash"  About a minute ago  Up About a minute          nostalgic_elion
PS C:\Users\Lenovo> docker container rm f45775828da6
Error response from daemon: You cannot remove a running container f45775828da6297e793470cd07835cf764532a3d5eded8e4094ffc
0bc0f687858. Stop the container before attempting removal or force remove
PS C:\Users\Lenovo> docker container stop f45775828da6
f45775828da6
PS C:\Users\Lenovo> docker container rm f45775828da6
f45775828da6
PS C:\Users\Lenovo> docker ps
CONTAINER ID   IMAGE    COMMAND    CREATED   STATUS    PORTS     NAMES
PS C:\Users\Lenovo> docker images
REPOSITORY   TAG      IMAGE ID      CREATED      SIZE
ubuntu       latest   ff0fea8310f3  4 days ago   72.8MB
PS C:\Users\Lenovo> docker image rm f45775828da6
Error: No such image: f45775828da6
PS C:\Users\Lenovo> docker image rm ff0fea8310f3
Untagged: ubuntu:latest
Untagged: ubuntu@sha256:bea6d19168bbfd6af8d77c2cc3c572114eb5d113e6f422573c93cb605a0e2ffb
Deleted: sha256:ff0fea8310f3957d9b1e6ba494f3e4b63cb348c76160c6c15578e65995ffaa87
Deleted: sha256:867d0767a47c392f80acb51572851923d6d3e55289828b0cd84a96ba342660c7
PS C:\Users\Lenovo> docker images
REPOSITORY   TAG      IMAGE ID   CREATED   SIZE
PS C:\Users\Lenovo> _
```

Conclusion:

Q1. What is the difference between containerization and virtualization?
Ans:Containerization and virtualization are both technologies used to isolate and manage applications and their dependencies, but they differ in their approach and level of abstraction.Containerization, exemplified by platforms like Docker, involves encapsulating an application and its dependencies into a lightweight, portable container. These containers share the host operating system's kernel but run in isolated user space environments. Containers are more lightweight and efficient compared to virtual machines, as they don't require a separate operating system instance for each application. They provide consistency across different environments, enabling applications to run reliably regardless of the underlying infrastructure.Virtualization, on the other hand, creates multiple virtual machines (VMs) on a single physical machine, each with its own operating system instance. Hypervisors, such as VMware or VirtualBox, manage these VMs, enabling multiple operating systems to run simultaneously on the same hardware. Virtualization provides stronger isolation between applications but requires more resources and overhead due to the duplication of operating systems.

Q2. What is Docker Daemon?
Ans:The Docker Daemon is a background process that manages Docker objects such as images, containers, networks, and volumes. It listens for Docker API requests and handles container lifecycle events, such as starting, stopping, and monitoring containers. The Docker Daemon runs on the host machine and communicates with the Docker client to execute commands and manage Docker resources. It's responsible for building Docker images, running containers, managing storage volumes, and networking configurations. The Docker Daemon plays a crucial role in the Docker architecture, enabling users to interact with Docker containers and manage their lifecycle effectively.