

LAB ASSIGNMENT NO-07

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

Theory:

1. What is docker?

Docker is a containerization platform that allows developers to package applications and their dependencies into lightweight, portable containers. These containers can run consistently across various environments, such as development laptops, testing servers, and production servers. Docker simplifies application deployment, scaling, and management by isolating applications within containers, ensuring they have everything needed to run, and abstracting away the underlying infrastructure.

Key points about Docker:

Containerization: Docker containers encapsulate an application and its dependencies, including libraries and configuration files, into a single package. This package is isolated from the host system and other containers, ensuring consistent and reliable application execution.

Portability: Docker containers are highly portable. Developers can build a container image on one machine and run it on another without worrying about compatibility issues, thanks to Docker's consistent environment.

Efficiency: Containers are lightweight compared to virtual machines (VMs) because they share the host operating system's kernel. This makes them more resource-efficient and faster to start compared to VMs.

Version Control: Docker images can be versioned and stored in repositories, making it easy to track changes, roll back to previous versions, and collaborate on application development.

Orchestration: Docker provides tools like Docker Compose and Kubernetes for orchestrating and managing containers at scale. This allows for automated deployment, load balancing, and scaling of containerized applications.

DevOps Integration: Docker plays a crucial role in DevOps practices by enabling continuous integration and continuous deployment (CI/CD) pipelines. Containers can be built and deployed automatically, streamlining the development and release process.

Ecosystem: Docker has a vast ecosystem of pre-built container images

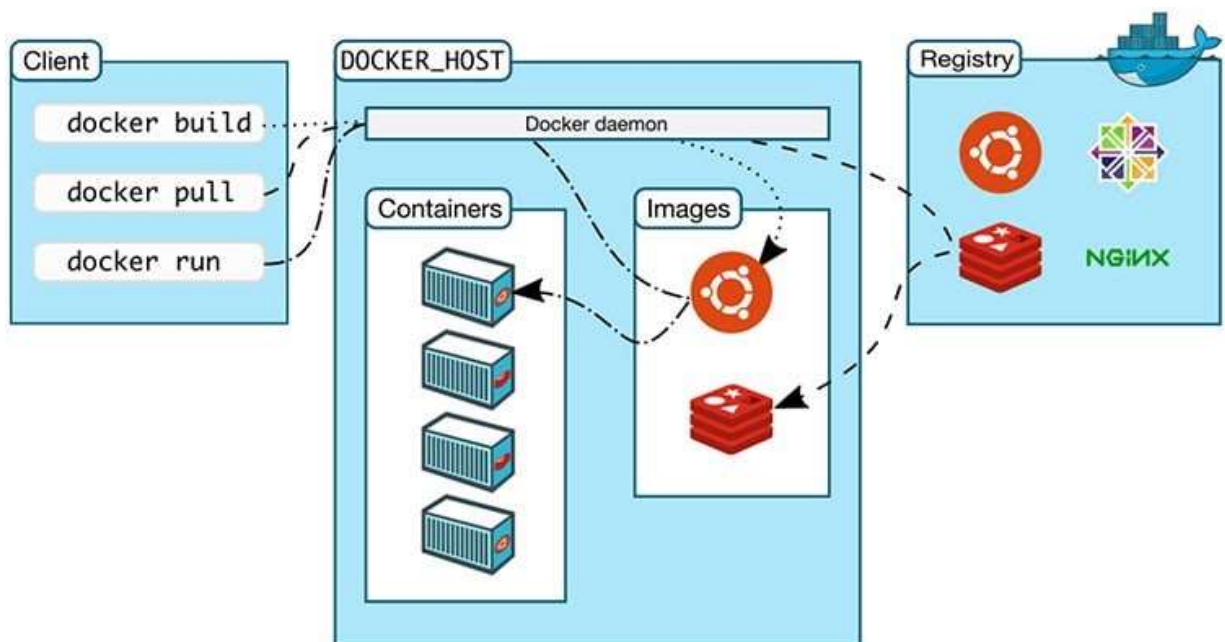
available on Docker Hub, which makes it easy to find and use popular software stacks and services.

Security: Docker containers provide isolation, but proper security practices are essential to ensure containerized applications are not vulnerable to attacks. Docker offers security features like user namespaces, seccomp profiles, and more.

Docker Architecture

The Core components of the Docker consists of

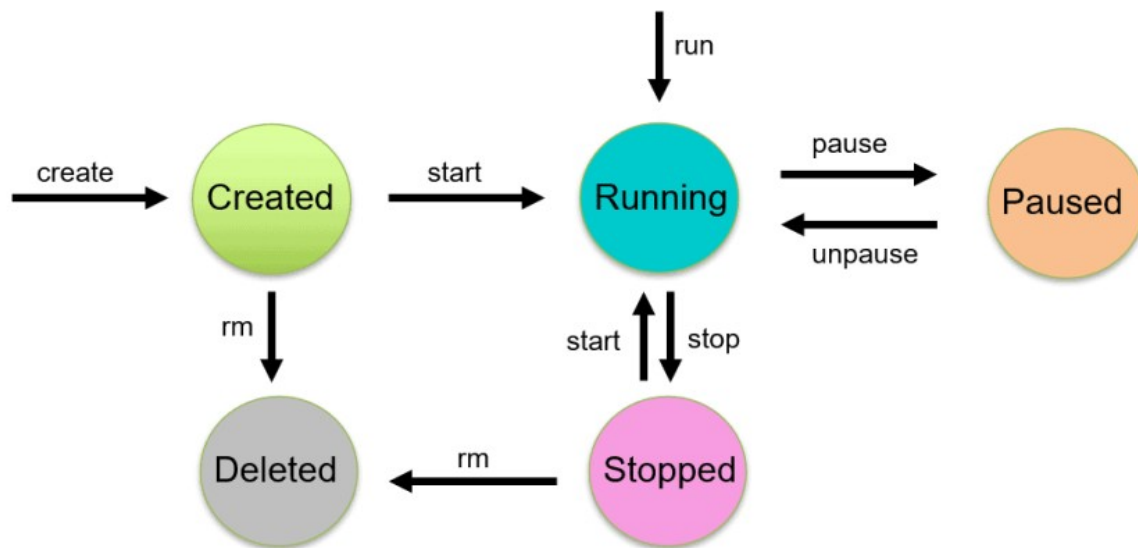
1. Docker daemon
2. Docker client
3. Docker Desktop
4. Docker registry
5. Docker images
6. Docker containers



Docker Container Lifecycle Management

The lifecycle of a docker container consists of five states:

1. Created state
2. Running state
3. Paused state/unpaused state
4. Stopped state
5. Killed/Deleted state



Created state

In the created state, a docker container is created from a docker image.
`docker create --name <name-of-container> <docker-image-name>`

Running state

In the running state, the docker container starts executing the commands mentioned in the image. To run a docker container, use the docker run command.

`docker run <container-id>`

or

`docker run <container-name>`

The docker run command creates a container if it is not present. In this case, the creation of the container can be skipped.

Paused state

In the paused state, the current executing command in the docker container is paused. Use the docker pause command to pause a running container.

`docker pause container <container-id or container-name>`

Note: docker pause pauses all the processes in the container. It sends the SIGSTOP signal to pause the processes in the container.

Unpaused state

In the unpaused state, the paused container resumes executing the commands once it is unpaused.

Use the docker unpause command to resume a paused container.

Then, Docker sends the SIGCONT signal to resume the process.

`docker unpause <container-id or container-name>`

Stopped state

In the stopped state, the container's main process is shutdown gracefully. Docker sends SIGTERM for graceful shutdown, and if needed, SIGKILL, to kill the container's main process.

Use the docker stop command to stop a container.

`docker stop <container-id or container-name>`

Restarting a docker container would translate to docker stop, then docker run, i.e., stop and run phases.

Killed/Deleted state

In the killed state, the container's main processes are shutdown abruptly.

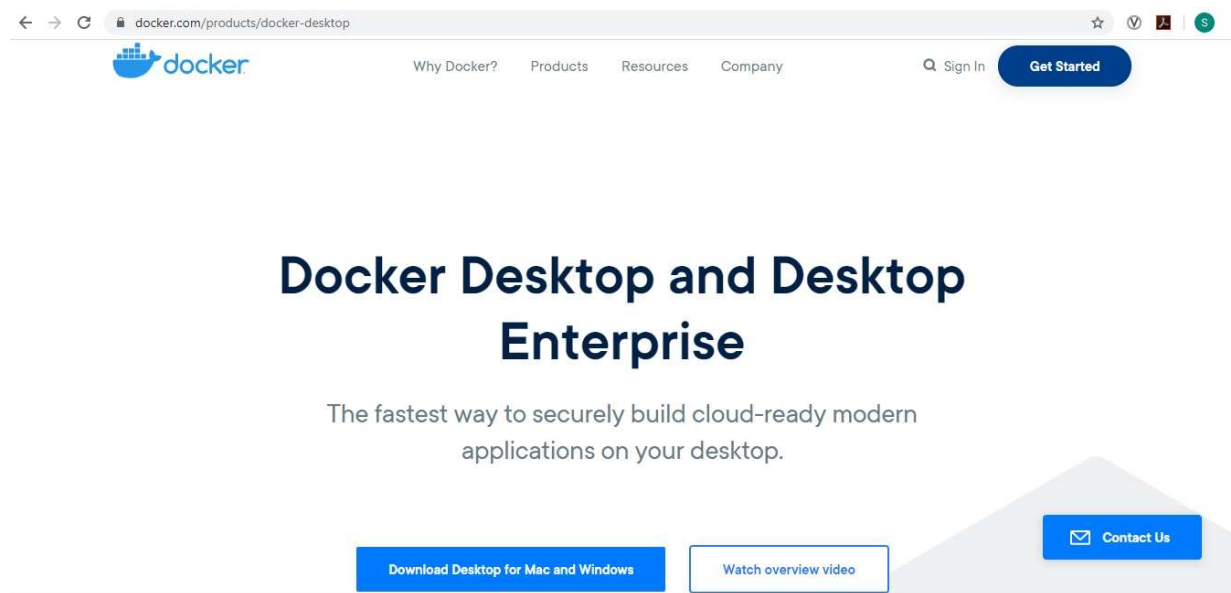
Docker sends a SIGKILL signal to kill the container's main process.

`docker kill <container-id or container-name>`

Installing Docker

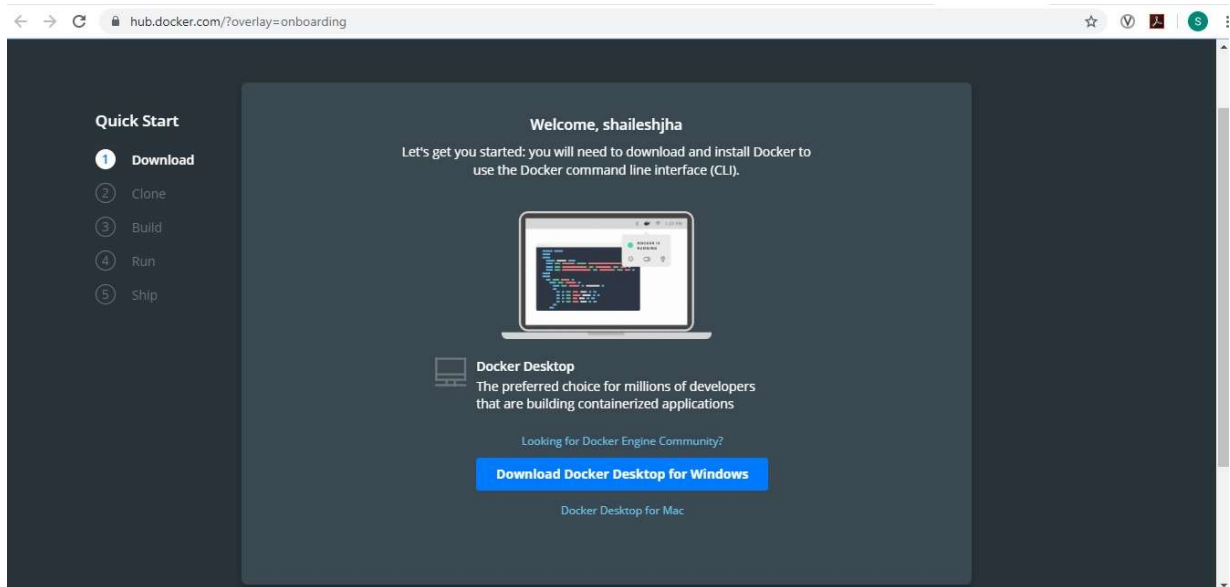
Step 1 – Download Docker

Click on Products-> Docker Desktop on the menu bar. This will take you to the docker desktop product page.

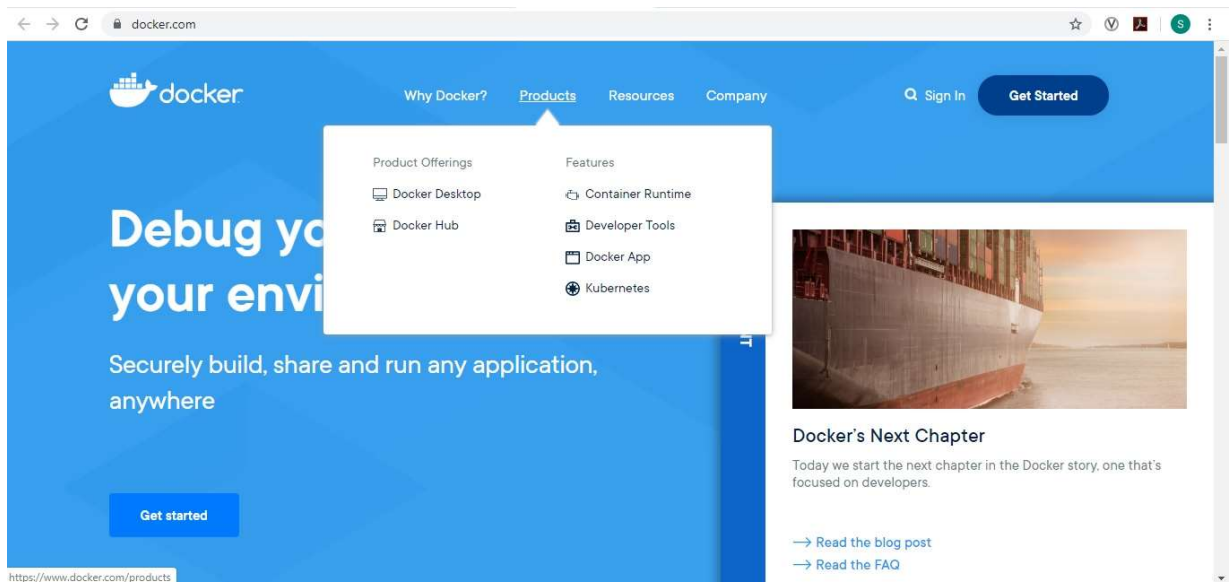


Docker Desktop Product Webpage

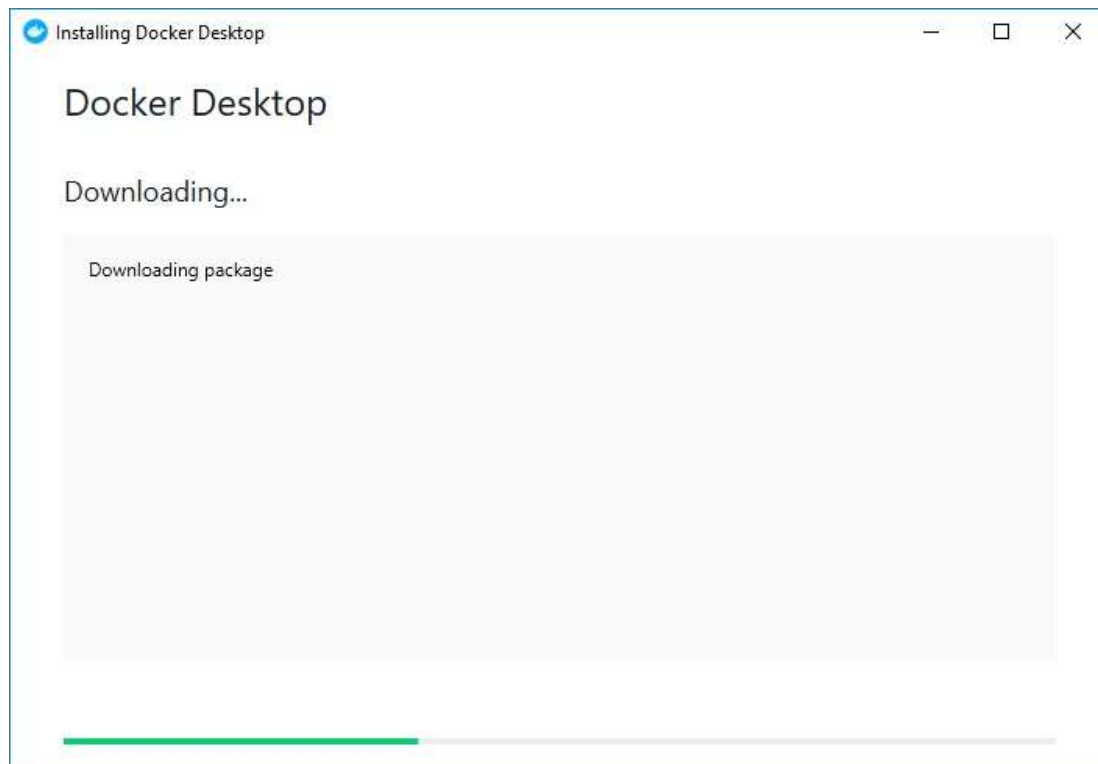
Click on Download Desktop for Mac and Windows.



Docker Hub – download docker web page
The installer file for windows is around 914 MB.



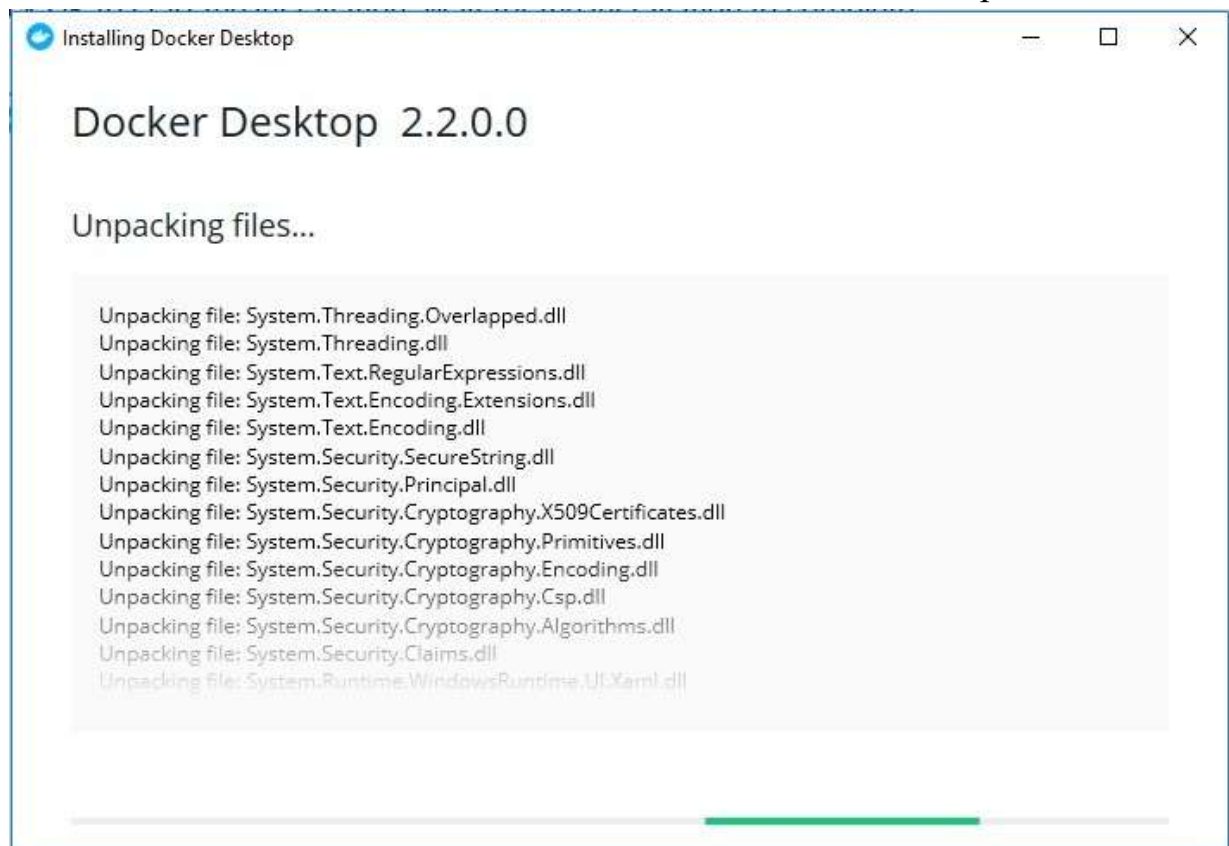
Step 2 – Run the installer



Docker Installation

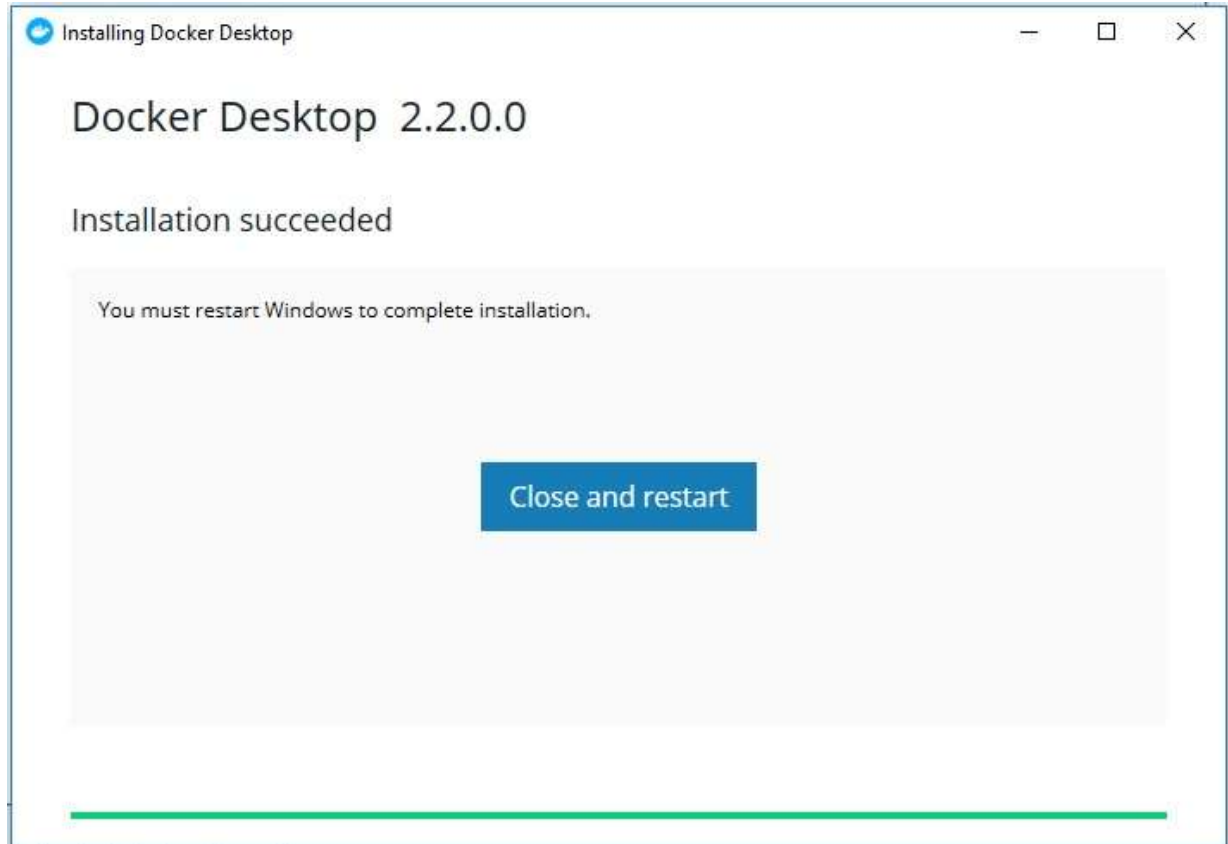
Step 3 – Configuration Settings

Click OK to start the installation. Wait for the installation to complete.



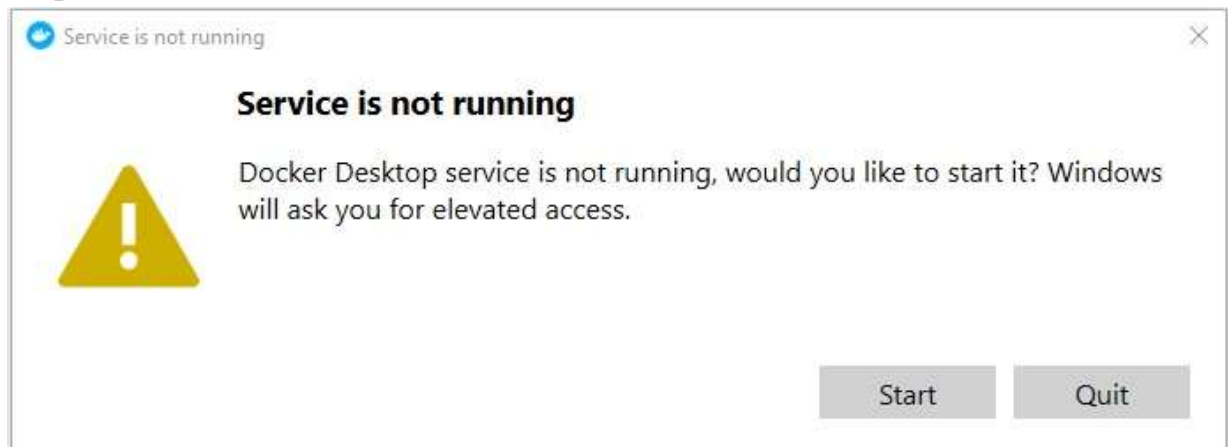
Docker Installation progress

Once the installation completes, click on Close and restart to complete the installation.

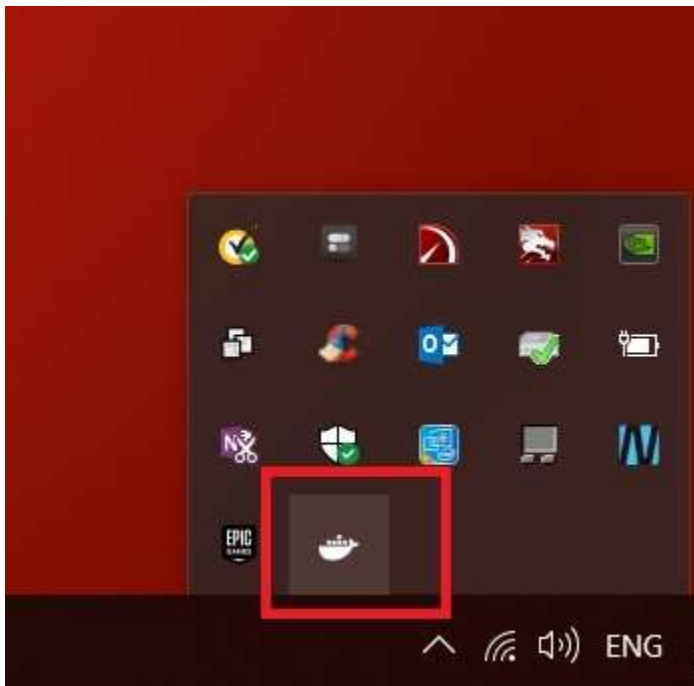


Docker Installation Complete

Step 4 – Run Docker



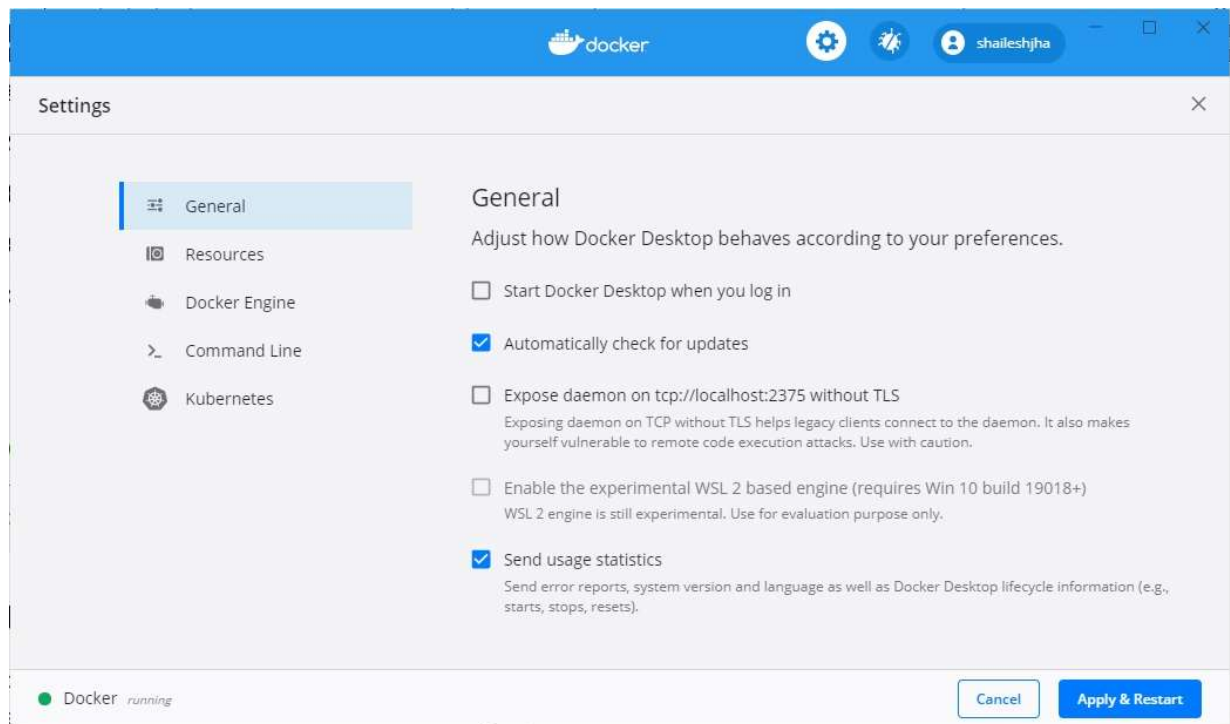
Docker Service not running warning



Docker Icon – Windows task bar



Step 5 – Disable Start Docker at Startup



Docker Desktop settings

Step 6 – Check the version of Docker Installed

Open Powershell or command prompt and enter the command:

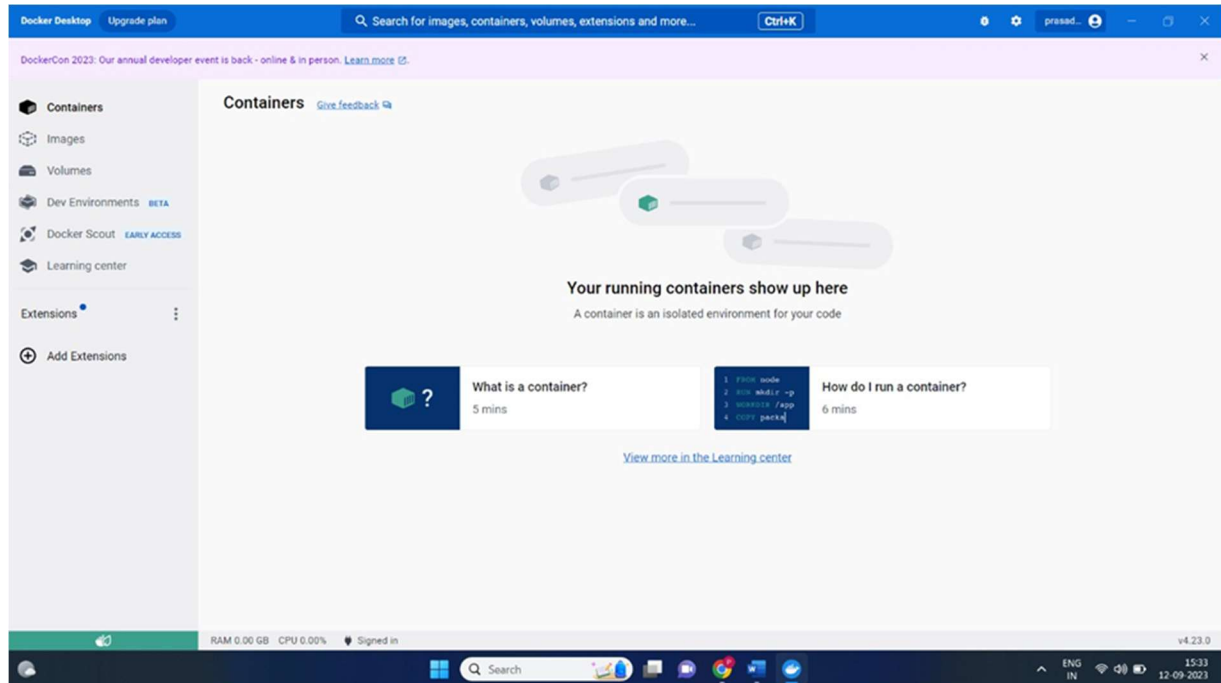
`docker version`

You should see something like this.

```
Command Prompt
C:\Users\shailesh.j>docker version
Client: Docker Engine - Community
Version: 19.03.5
API version: 1.40
Go version: go1.12.12
Git commit: 633a0ea
Built: Wed Nov 13 07:22:37 2019
OS/Arch: windows/amd64
Experimental: false

Server: Docker Engine - Community
Engine:
Version: 19.03.5
API version: 1.40 (minimum version 1.12)
Go version: go1.12.12
Git commit: 633a0ea
Built: Wed Nov 13 07:29:19 2019
OS/Arch: linux/amd64
Experimental: false
containerd:
Version: v1.2.10
GitCommit: b34a5c8af56e510852c35414db4c1f4fa6172339
runc:
Version: 1.0.0-rc8+dev
GitCommit: 3e425f80a8c931f88e6d94a8c831b9d5aa481657
docker-init:
Version: 0.18.0
GitCommit: fec3683
C:\Users\shailesh.j>
```

Docker Container:



```
PS C:\Users\Pratik Arote> docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
PS C:\Users\Pratik Arote> docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
719385e3284d: Pull complete
Digest: sha256:dcb6daec718f547568c562956fa47e1b03673dd010fe6ee58ca806767031d1c
Status: Downloaded newer image for hello-world:latest
docker.io/library/hello-world:latest

What's Next?
View a summary of image vulnerabilities and recommendations + docker scout quickview hello-world
PS C:\Users\Pratik Arote> docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
hello-world latest 9c7a54a9a43c 4 months ago 13.3kB
PS C:\Users\Pratik Arote> docker run hello-world

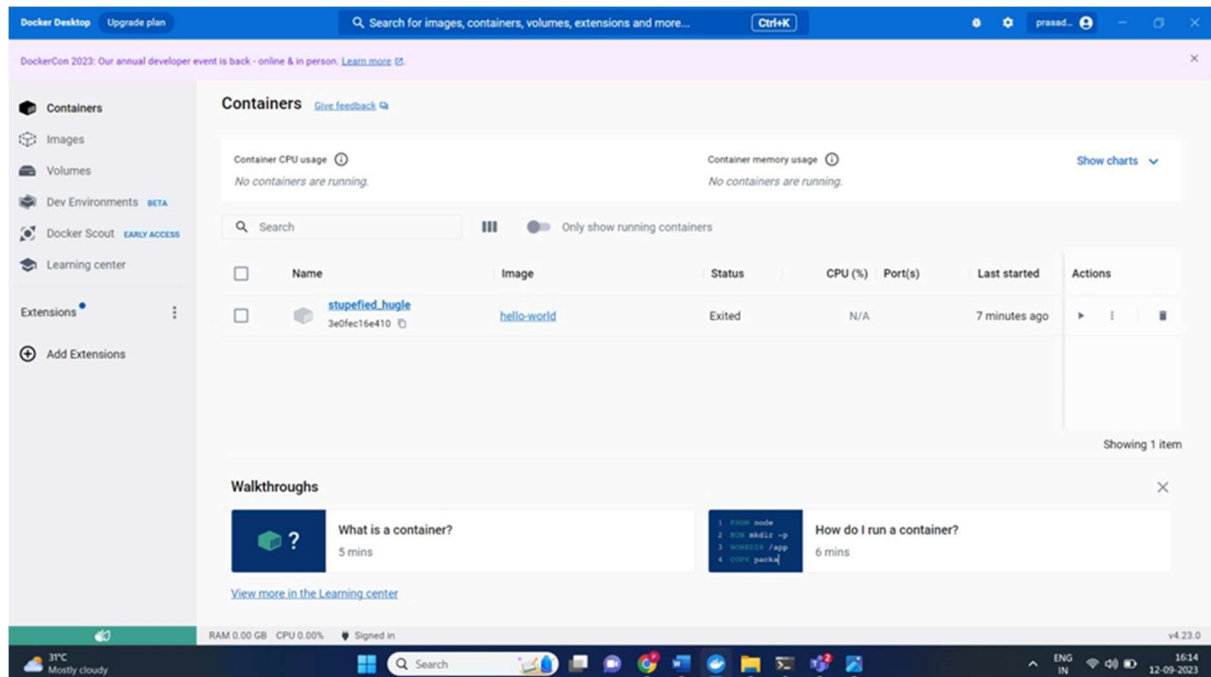
Hello from Docker!
This message shows that your installation appears to be working correctly.

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/
PS C:\Users\Pratik Arote> |
```



Conclusion: In this experiment We have installed and deployed a container in 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 concept of containerization and Analyze the containerization of OS images and deployment of applications over Docker.