**To SSH to Linux VM:**

Start🡪Run🡪CMD

ssh [nachiket@40.117.116.219](mailto:nachiket@40.117.116.219)

nachiket 🡪 User Name

40.117.116.219 🡪 Public IP Address of your Linux VM

Say “Yes” to the questions asked and press Enter key.

Key in the password.

Key in the following commands on Linux VM:

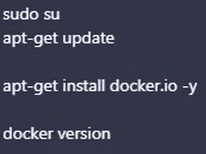
1) sudo su – Switches to super user (Like Administrator in Windows OS). You will see # prompt instead of $ prompt.  
2) apt-get update

To Install the Docker, Run the following Command:

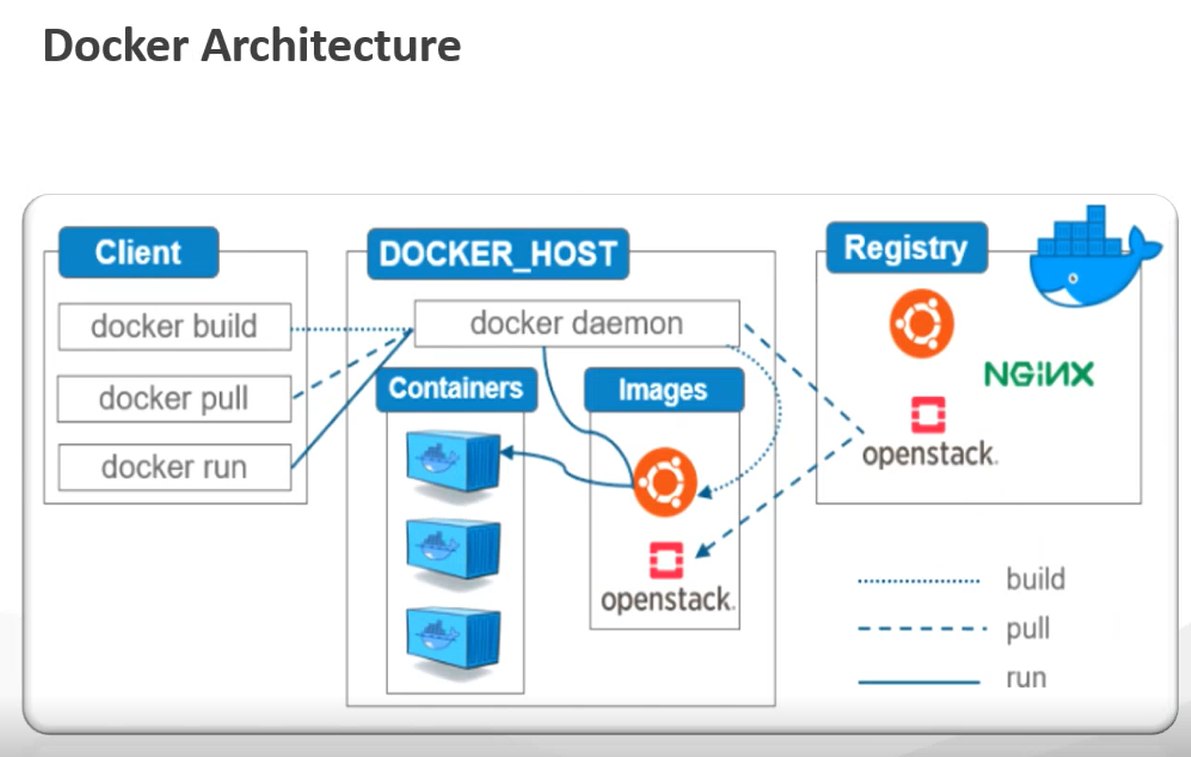
3) apt-get install docker.io -y

This will download and install Docker from the Docker official web site.

4) docker version



**Docker Architecture:**



**Docker Host:** The virtual machine or may be your laptop in which Docker is installed.

Once the Docker in Installed, a Background Service is Started. That Background Service is known as Docker Daemon. Docker Daemon takes care of managing our containers, managing your images etc.

Docker Host is the Machine on which the Docker Daemon is Running.

**Docker Daemon** is going to manage your Containers and Images.

**Docker Client (Docker CLI):** The One who is Using Docker or the One who is Giving the Docker Commands.

Docker Follows Client Server Architecture.

The machine on which Docker is installed, from the same machine you can give Docker commands.

The One who Issues Docker Commands OR Machine from Where you Give Docker Commands is Docker Client.

The Same Machine can be a Docker Host and can be a Docker Client.

But it is possible to have Docker Daemon on One Machine and Docker Client on Another Machine because it is a Client Server Architecture.

Whenever you issue the Docker command, those commands are Received by Docker Daemon and Docker Daemon acts on those commands.

This means the Client will use Docker CLI to give Docker Commands to the Docker Daemon.

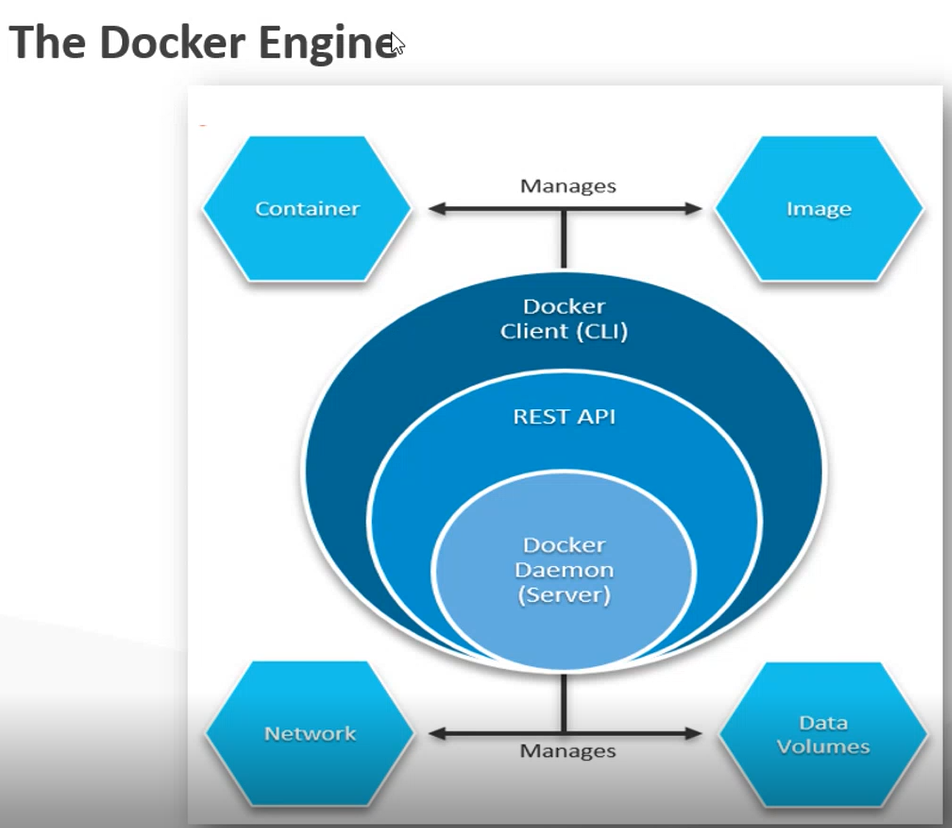
Docker Daemon is heart of Docker. Sometimes it is also known as Docker Engine. Daemon will take responsibility of Creating Containers, Creating Images, Creating Containers from Images,

Managing the Containers, Managing the Images.

**Registry:** Docker Hub. Collection of images. People create images and push it to the Docker Hub. Images are shared with other people by pushing them to the Docker Hub.

When you issue “docker pull” command, docker pull will tell Docker Daemon, please pull the specified image from the registry. So, Docker Daemon will connect to Docker Hub (Registry), pull that image

and give it you on your local machine and then you can create containers from that image.



**Docker Engine: Docker Client + Docker Daemon + REST API**

Docker CLI is a Collection of Docker Commands used by Client. E.g. – docker pull, docker version

Docker Commands have to be sent to Docker Daemon.

The commands keyed in by you on Docker CLI are given to the Docker Daemon by REST API.

Docker commands internally use REST API. This REST API is already installed when you install Docker. This REST API will forward those commands to the Daemon.

Docker Daemon can perform its task only when you give command to it. You give command to it via Docker CLI.

REST APIs are abstracted in the form of Docker Commands.

Docker Daemon does Container Management, Image Management, Volume Management and Network Management.

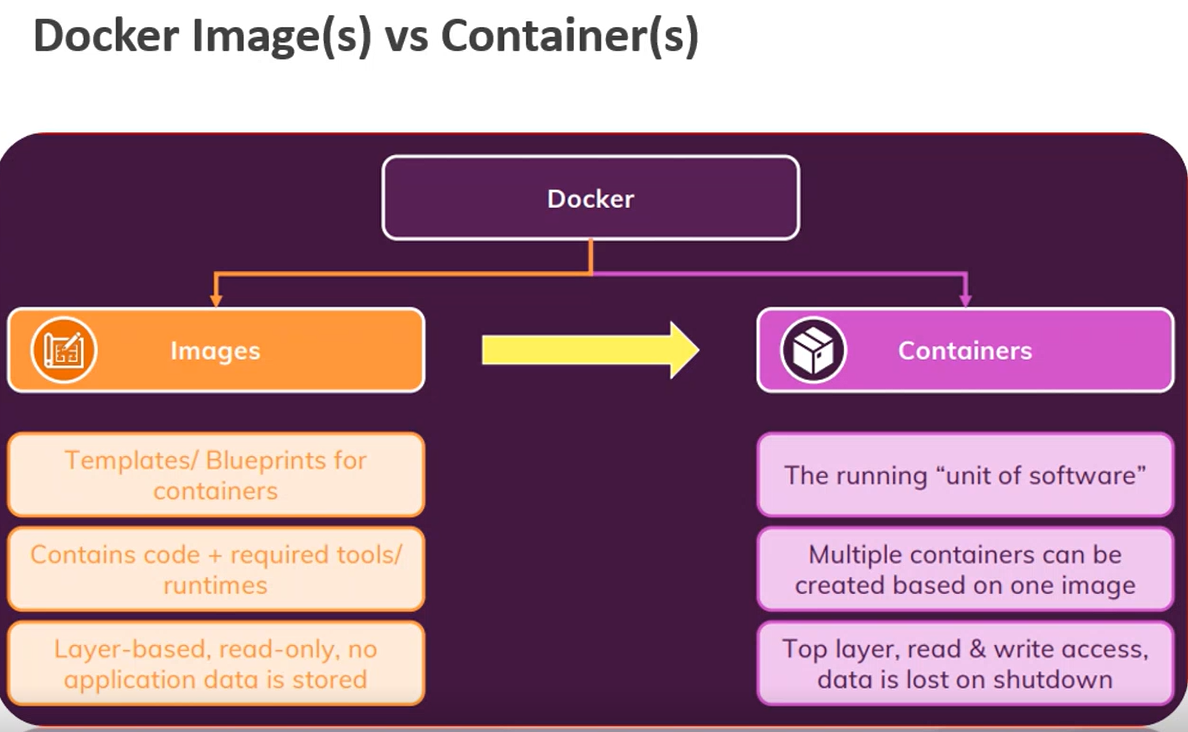


Image is a blueprint for a container. Image should define things such as When container is started what application it should host, on what port number the application should run, what dependencies it should have. Image is like creating a class. From one single image, you can create multiple containers.

Containers are like objects of the class.

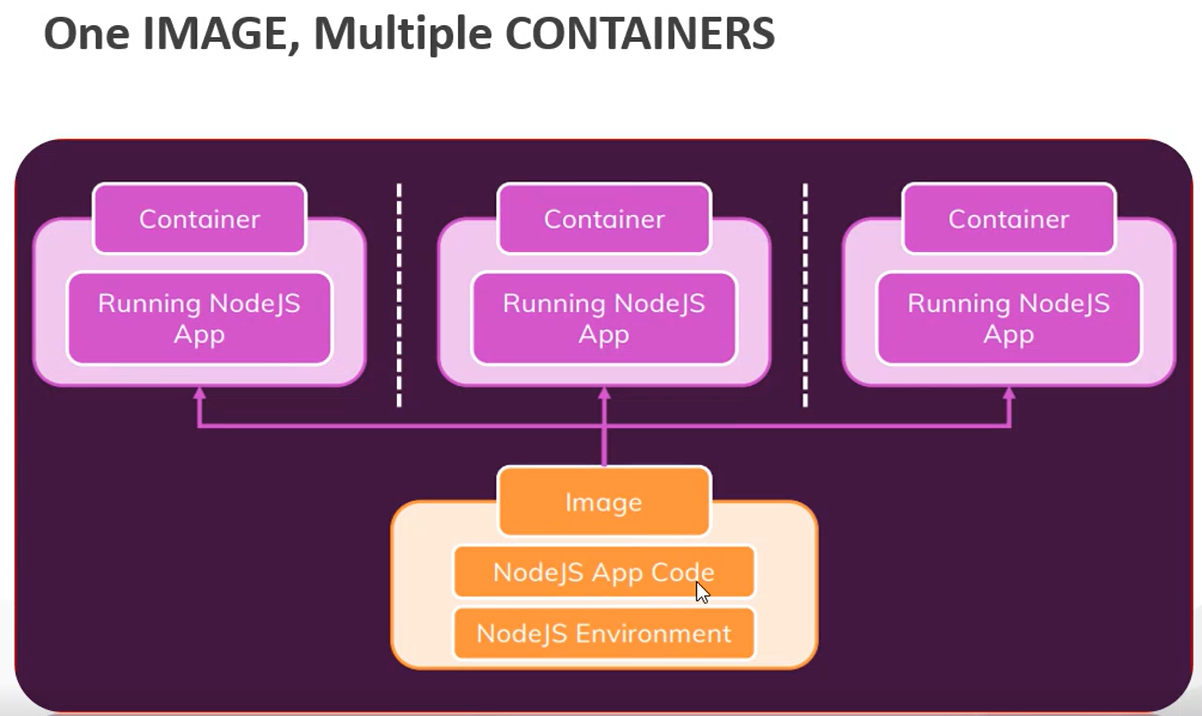
Images are shared. Containers are not shared.

Containers are temporary. They are created and destroyed.

Images are permanent.

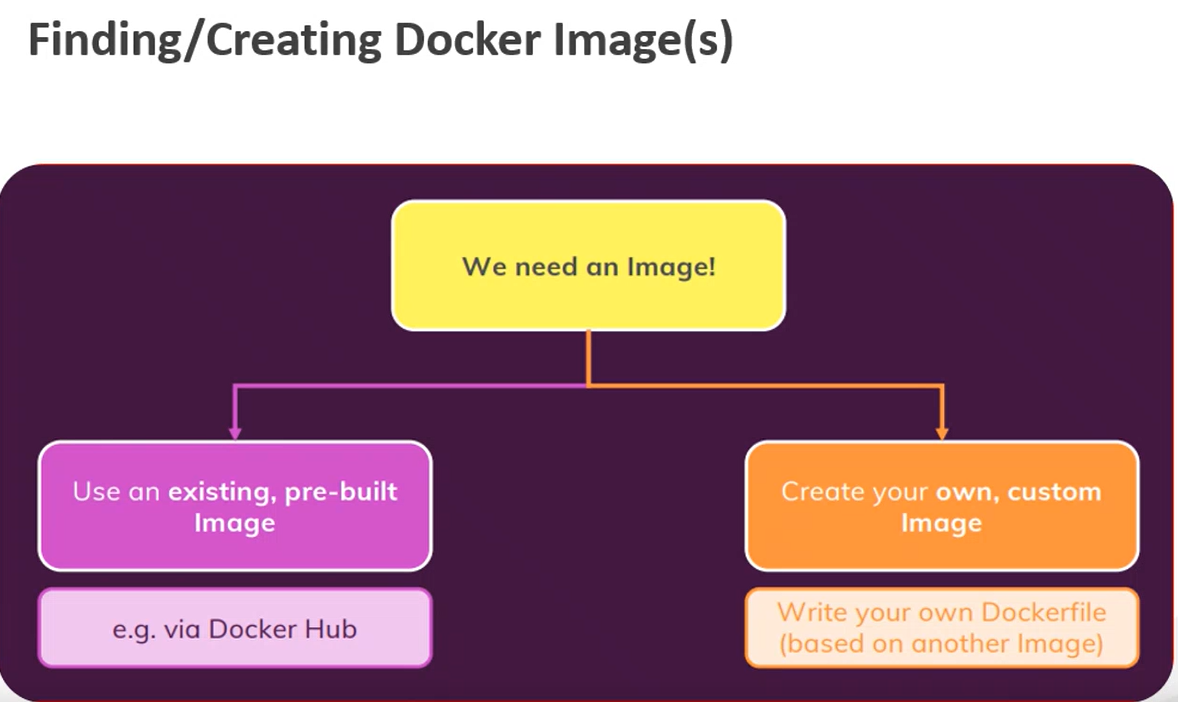
As long as you have an image, you can create a container from it. Containers are lightweight. They share host machine’s Operating System and Kernel. Hence you can spin up multiple container in no time.

You put an image in Docker Hub. People pull that image from Docker Hub and create containers from that image.



**Microservices are Hosted Inside a Container.**

In above diagram, you are hosting 3 instances of your application.



**Docker Commands:**

To check the Docker Installation:

docker - -version

To List the Images:

docker images

To Pull the Hello-World Image from Docker Hub:

docker pull hello-world

To Pull the Hello-World Image from DockerHub and Create the Container from it:

docker run --name helloworldcon hello-world

To List the Running Containers:

docker ps

To List All the Containers (Running and Exited):

docker ps -a