# Docker

# Docker - Introduction

- Docker is a container management service. The keywords of Docker are **develop, ship** and **run** anywhere. The whole idea of Docker is for developers to easily develop applications, ship them into containers which can then be deployed anywhere.
- **Features of Docker**
- Docker has the ability to reduce the size of development by providing a smaller footprint of the operating system via containers.
- With containers, it becomes easier for teams across different units, such as development, QA and Operations to work seamlessly across applications.
- You can deploy Docker containers anywhere, on any physical and virtual machines and even on the cloud.
- Since Docker containers are pretty lightweight, they are very easily scalable.

# Docker – Architecture.

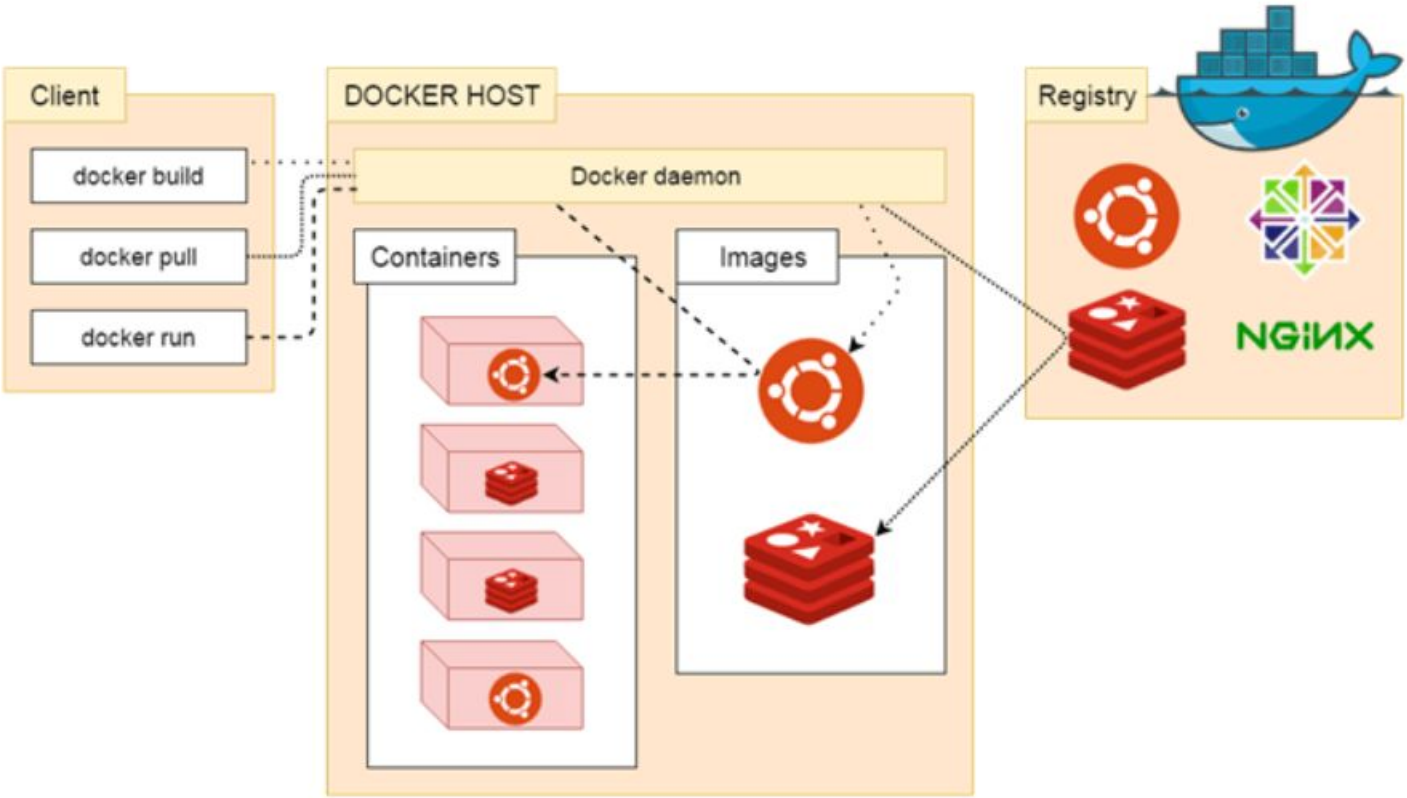
**What is Docker daemon?**

Docker daemon runs on the host operating system. It is responsible for running containers to manage docker services. Docker daemon communicates with other daemons. It offers various Docker objects such as images, containers, networking, and storage.

**Docker architecture**

Docker follows Client-Server architecture, which includes the three main components that are Docker Client, Docker Host, and Docker Registry.

# Docker – Architecture.



# Docker – Architecture.

**1. Docker Client**

Docker client uses commands and REST APIs to communicate with the Docker Daemon (Server). When a client runs any docker command on the docker client terminal, the client terminal sends these docker commands to the Docker daemon. Docker daemon receives these commands from the docker client in the form of command and REST API's request.

**2. Docker Host**

Docker Host is used to provide an environment to execute and run applications. It contains the docker daemon, images, containers, networks, and storage.

# Docker – Architecture.

**3. Docker Registry**

Docker Registry manages and stores the Docker images.

There are two types of registries in the Docker -

**Pubic Registry** - Public Registry is also called as Docker hub.

**Private Registry** - It is used to share images within the enterprise.

**Docker Objects**

There are the following Docker Objects -

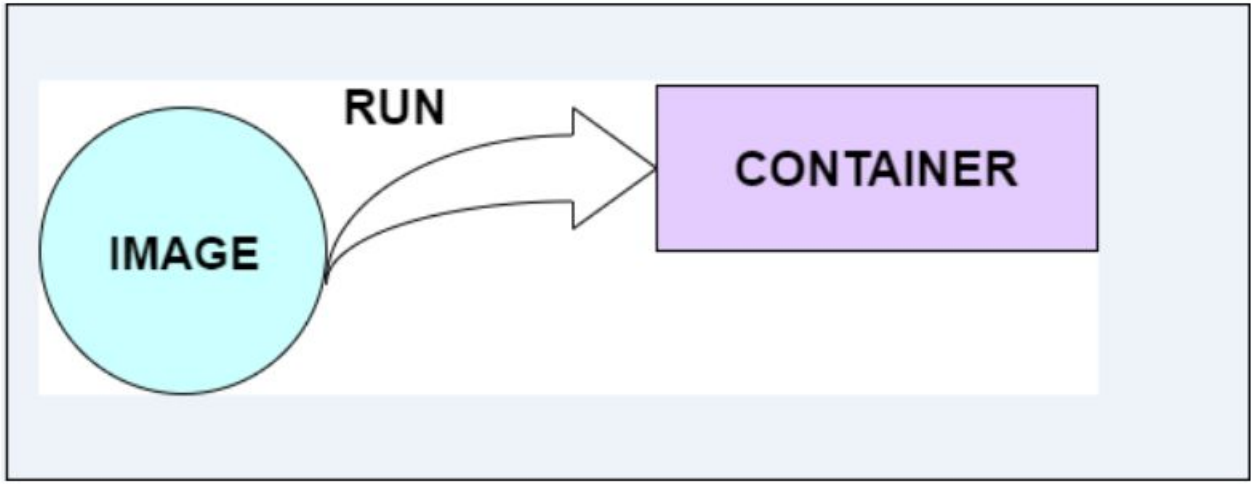
**Docker Images**

Docker images are the read-only binary templates used to create Docker Containers. It uses a private container registry to share container images within the enterprise and also uses public container registry to share container images within the whole world. Metadata is also used by docket images to describe the container's abilities.

**Docker Containers**

# Docker – Architecture.

**Image runs in a container...**



# Docker – Architecture.

**Docker Networking**

Using Docker Networking, an isolated package can be communicated. Docker contains the following network drivers -

**Bridge** - Bridge is a default network driver for the container. It is used when multiple docker communicates with the same docker host.

**Host** - It is used when we don't need for network isolation between the container and the host.

**None** - It disables all the networking.

**Overlay** - Overlay offers Swarm services to communicate with each other. It enables containers to run on the different docker host.

**Macvlan** - Macvlan is used when we want to assign MAC addresses to the containers.

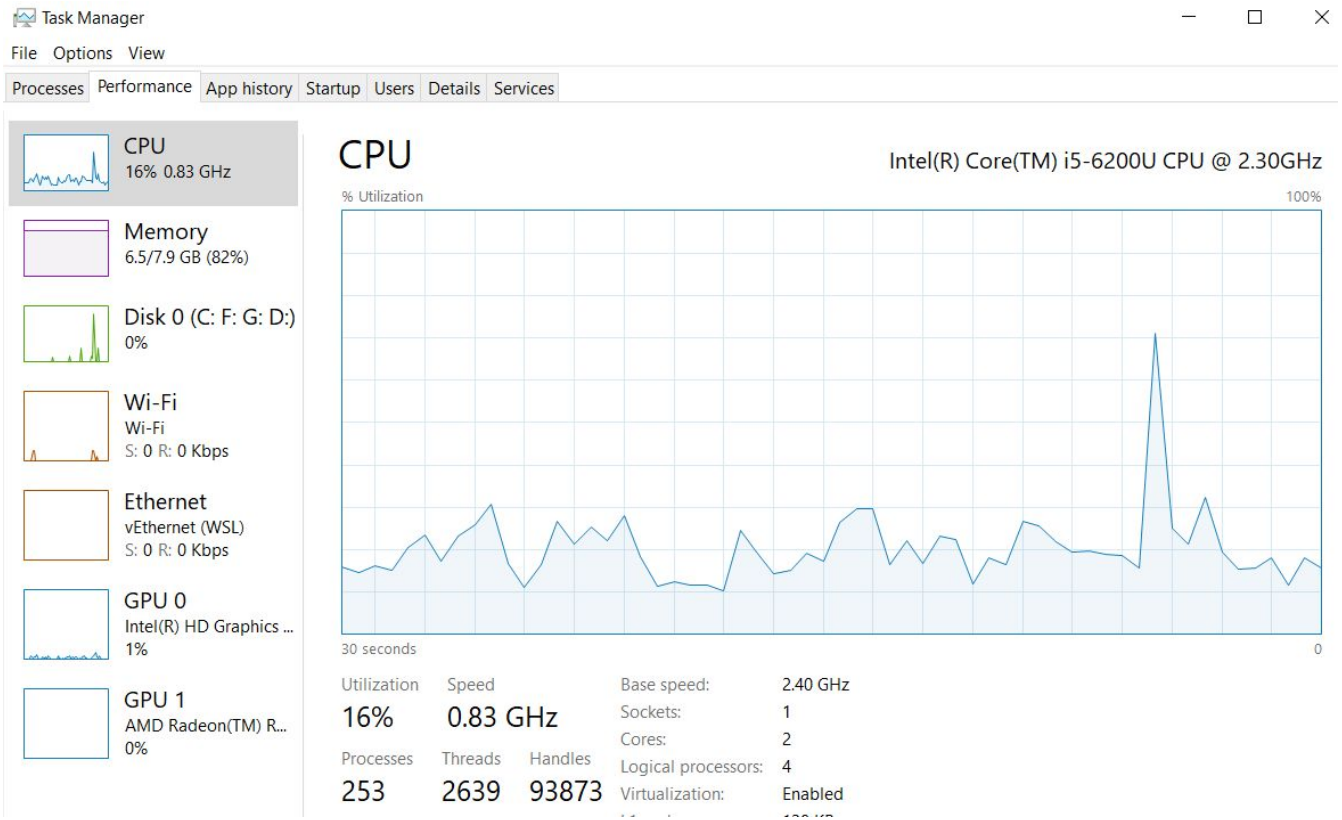
**Docker Storage**

Docker Storage is used to store data on the container. Docker offers the

# Docker - Installation

- Download docker hub for desktop and install it.

Make sure that Virtualization is enabled in BIOS. After enabling, it shows as enabled in task manager. URL to follow the steps to enable to Virtulization: **https://www.minitool.com/news/enable-virtualization-windows-10.html**



# Docker – Installation and Docker hub.

- Install Ubuntu linux also.
- After installing, create a user for docker hub.
- Login with the credentials.
- **Docker Hub** is a registry service on the cloud that allows you to download Docker images that are built by other communities.  Let us download node.js from the community.
- From the ubuntu, get the node into docker hub.

```
charan@LAPTOP-ONEMN5DS:~$ docker pull node
Using default tag: latest
latest: Pulling from library/node
1e987daa2432: Pull complete
a0edb687a3da: Pull complete
6891892cc2ec: Pull complete
684eb726ddc5: Pull complete
b0af097f0da6: Pull complete
154aee36a7da: Pull complete
4dfe553b641e: Pull complete
ce0cc1c1f596: Pull complete
7eca74d68564: Pull complete
Digest: sha256:cb16b22cefb9a3a87f4d0bd1371e07fed4b8ff569b8f9f8c2efcc08edf11a854
Status: Downloaded newer image for node:latest
docker.io/library/node:latest
charan@LAPTOP-ONEMN5DS:~$
```

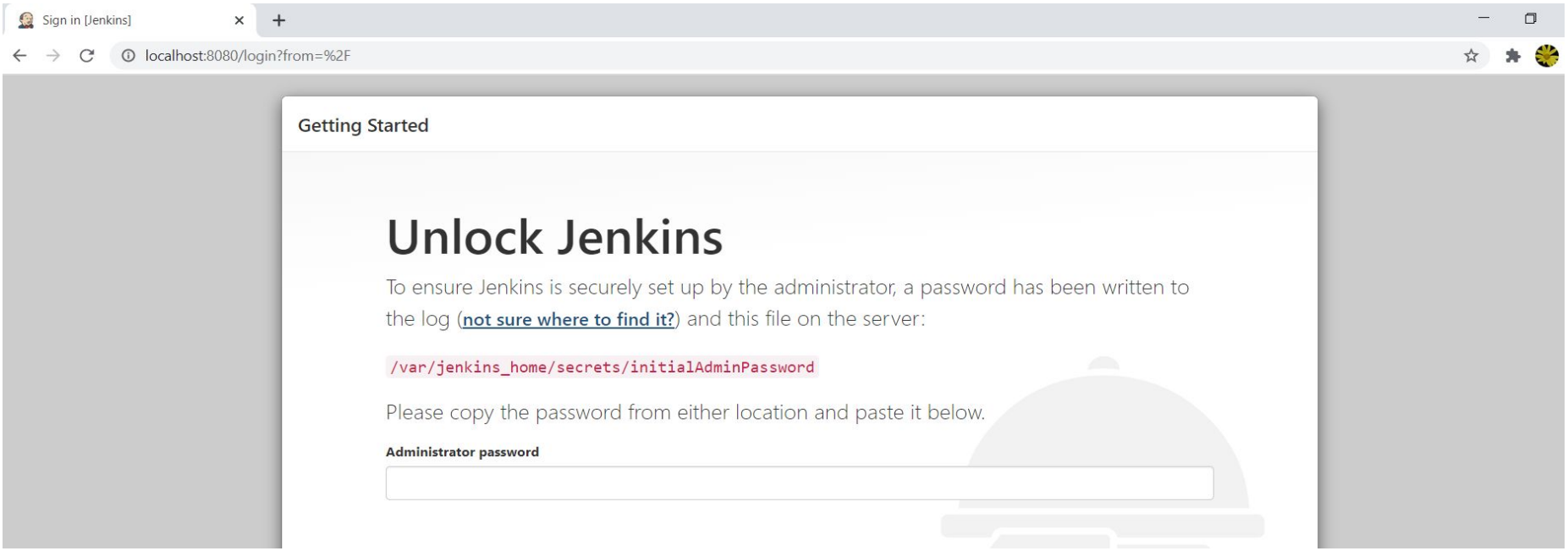
# Docker hub – downloading jenkins image and run it.

```
charan@LAPTOP-ONEMN5DS:~$
charan@LAPTOP-ONEMN5DS:~$ sudo docker pull jenkins/jenkins
Using default tag: latest
latest: Pulling from jenkins/jenkins
0ecb575e629c: Downloading [================>                                  ]  17.88MB/50.4MB
56ab0896877d: Downloading [=====================================>             ]  13.94MB/17.89MB
47139467bbb9: Downloading [=======>                                           ]  14.54MB/103.4MB
b21475076209: Waiting
ba9ed6a087d6: Waiting
5fab90f83725: Waiting
73d8c89cc98b: Waiting
097980007e32: Waiting
0addd00d8037: Waiting
a10e54e1aad6: Waiting
64b99ebb78df: Waiting
7de917672c68: Waiting
2c09ef2cf2a8: Waiting
d05c3f36b760: Waiting
b86f59316be8: Waiting
f7ac6306eee3: Waiting
```

```
charan@LAPTOP-ONEMN5DS:~$ sudo docker run -p 8080:8080 -p 50000:50000 jenkins/jenkins
Running from: /usr/share/jenkins/jenkins.war
webroot: EnvVars.masterEnvVars.get("JENKINS_HOME")
2021-02-18 14:51:19.604+0000 [id=1]     INFO    org.eclipse.jetty.util.log.Log#initialized: Logging initialized @684ms t
o org.eclipse.jetty.util.log.JavaUtilLog
2021-02-18 14:51:19.849+0000 [id=1]     INFO    winstone.Logger#logInternal: Beginning extraction from war file
2021-02-18 14:51:24.555+0000 [id=1]     WARNING o.e.j.s.handler.ContextHandler#setContextPath: Empty contextPath
2021-02-18 14:51:24.666+0000 [id=1]     INFO    org.eclipse.jetty.server.Server#doStart: jetty-9.4.35.v20201120; built:
2020-11-20T21:17:03.964Z; git: bdc54f03a5e0a7e280fab27f55c3c75ee8da89fb; jvm 1.8.0_282-b08
2021-02-18 14:51:25.184+0000 [id=1]     INFO    o.e.j.w.StandardDescriptorProcessor#visitServlet: NO JSP Support for /,
did not find org.eclipse.jetty.jsp.JettyJspServlet
2021-02-18 14:51:25.288+0000 [id=1]     INFO    o.e.j.s.s.DefaultSessionIdManager#doStart: DefaultSessionIdManager worke
rName=node0
2021-02-18 14:51:25.288+0000 [id=1]     INFO    o.e.j.s.s.DefaultSessionIdManager#doStart: No SessionScavenger set, usin
g defaults
2021-02-18 14:51:25.290+0000 [id=1]     INFO    o.e.j.server.session.HouseKeeper#startScavenging: node0 Scavenging every
 660000ms
2021-02-18 14:51:26.194+0000 [id=1]     INFO    hudson.WebAppMain#contextInitialized: Jenkins home directory: /var/jenki
ns_home found at: EnvVars.masterEnvVars.get("JENKINS_HOME")
2021-02-18 14:51:26.427+0000 [id=1]     INFO    o.e.j.s.handler.ContextHandler#doStart: Started w.@24f43aa3{Jenkins v2.2
80,/,file:///var/jenkins_home/war/,AVAILABLE}{/var/jenkins_home/war}
2021-02-18 14:51:26.497+0000 [id=1]     INFO    o.e.j.server.AbstractConnector#doStart: Started ServerConnector@29d80d2b
{HTTP/1.1, (http/1.1)}{0.0.0.0:8080}
2021-02-18 14:51:26.498+0000 [id=1]     INFO    org.eclipse.jetty.server.Server#doStart: Started @7578ms
2021-02-18 14:51:26.507+0000 [id=22]    INFO    winstone.Logger#logInternal: Winstone Servlet Engine running: controlPor
t=disabled
```

# Docker – Jenkins is up and running.

```
2021-02-18 14:51:58.033+0000 [id=47]    INFO    h.m.DownloadService$Downloadable#load: Obtained the updated data file for hudson.tasks.Maven.MavenInstall
er
2021-02-18 14:51:58.034+0000 [id=47]    INFO    hudson.util.Retrier#start: Performed the action check updates server successfully at the attempt #1
2021-02-18 14:51:58.041+0000 [id=47]    INFO    hudson.model.AsyncPeriodicWork#lambda$doRun$0: Finished Download metadata. 23,679 ms
2021-02-18 14:52:12.807+0000 [id=27]    INFO    jenkins.InitReactorRunner$1#onAttained: Completed initialization
2021-02-18 14:52:12.931+0000 [id=21]    INFO    hudson.WebAppMain$3#run: Jenkins is fully up and running
2021-02-18 17:20:55.235+0000 [id=76]    INFO    hudson.model.AsyncPeriodicWork#lambda$doRun$0: Started Periodic background build discarder
2021-02-18 17:20:55.264+0000 [id=76]    INFO    hudson.model.AsyncPeriodicWork#lambda$doRun$0: Finished Periodic background build discarder. 9 ms
2021-02-18 23:26:44.918+0000 [id=85]    INFO    hudson.model.AsyncPeriodicWork#lambda$doRun$0: Started Fingerprint cleanup
2021-02-18 23:26:44.942+0000 [id=85]    INFO    hudson.model.AsyncPeriodicWork#lambda$doRun$0: Finished Fingerprint cleanup. 22 ms
```

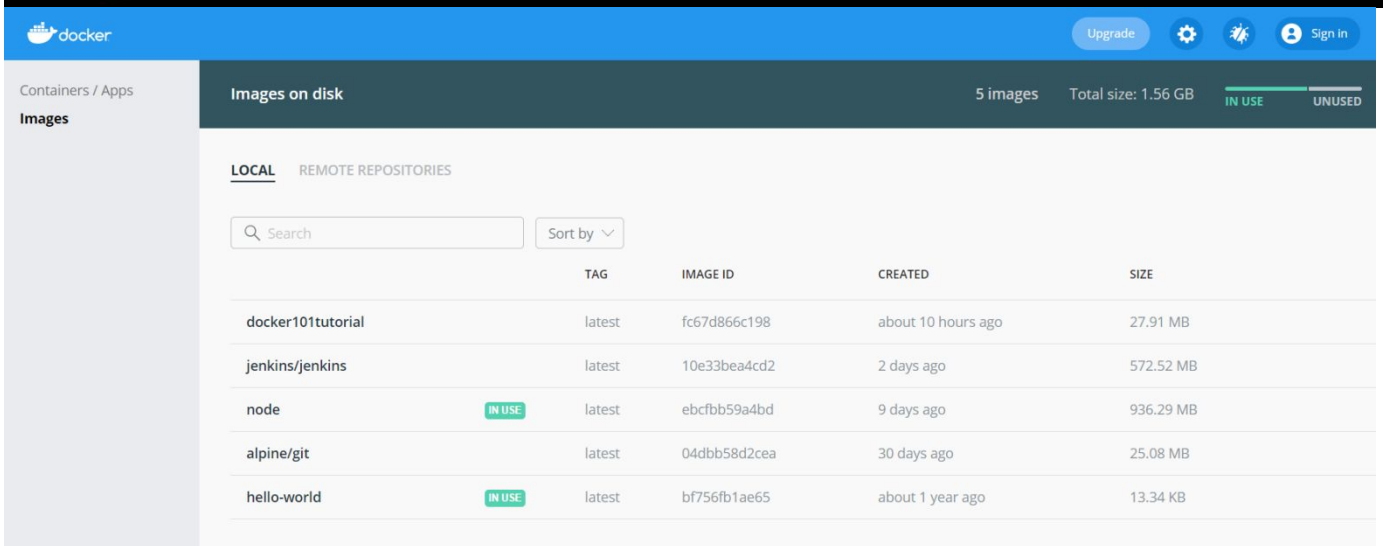


# Docker - Images

- **A Docker image** is a read-only template that contains a set of instructions for creating a container that can run on the Docker platform.
- **A Docker container** is a standard unit of software that stores up code and all its dependencies so the application runs fast and reliably from one computing environment to different ones.
- In Docker, everything is based on Images. An image is a combination of a file system and parameters.
- $ docker images  Will display all the images stored in the repository.
- Docker run hello-world  Will the run the program hello-world.

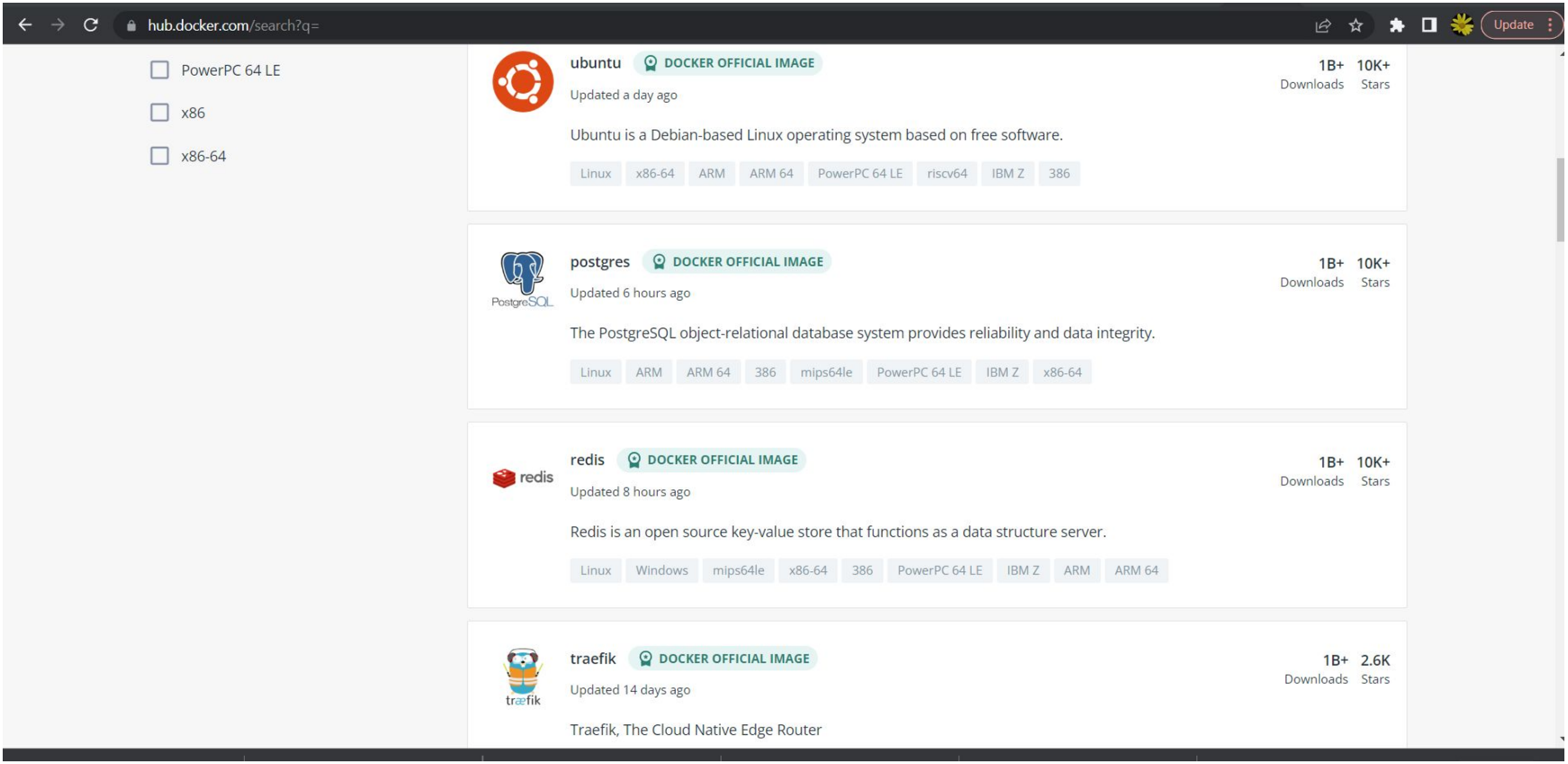
# Docker - image

```
charan@LAPTOP-ONEMN5DS:~$ docker images
REPOSITORY          TAG       IMAGE ID       CREATED          SIZE
docker101tutorial   latest    fc67d866c198   10 hours ago     27.9MB
jenkins/jenkins     latest    10e33bea4cd2   2 days ago       573MB
node                latest    ebcfbb59a4bd   9 days ago       936MB
alpine/git          latest    04dbb58d2cea   4 weeks ago      25.1MB
hello-world         latest    bf756fb1ae65   13 months ago    13.3kB
charan@LAPTOP-ONEMN5DS:~$ 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/

charan@LAPTOP-ONEMN5DS:~$
```



# List of docker hub softwares...

To display all the docker software for downloading are :
https://hub.docker.com and click on **explore** menu item.  Its a repoisotry which can be pulled as an image from docker hub.



# Docker - Containers

- Containers are instances of Docker images that can be run using the Docker run command. The basic purpose of Docker is to run containers.
- $ Docker ps –a  This commands will show all the containers in the docker hub.

charan@LAPTOP-ONEMN5DS: ~

```
charan@LAPTOP-ONEMN5DS:~$ docker run node
charan@LAPTOP-ONEMN5DS:~$ docker ps -a
CONTAINER ID   IMAGE         COMMAND                  CREATED          STATUS                      PORTS     NAMES
dbd8fb535a6e   node          "docker-entrypoint.s…"   13 seconds ago   Exited (0) 8 seconds ago              affectionate_hofstadter
d2533ac20f55   hello-world   "/hello"                 24 minutes ago   Exited (0) 24 minutes ago             jovial_rhodes
1f131ddeffda   hello-world   "/hello"                 25 minutes ago   Exited (0) 25 minutes ago             quirky_beaver
charan@LAPTOP-ONEMN5DS:~$
```

# Docker – Working with containers

- $ docker top <container id> ⇒ This command will show the list of processes running in that container.

```
charan@LAPTOP-ONEMN5DS: ~
charan@LAPTOP-ONEMN5DS:~$ docker ps -a
CONTAINER ID   IMAGE             COMMAND                  CREATED          STATUS                     PORTS                                          NAMES
c94ea4a6f1b0   jenkins/jenkins   "/sbin/tini -- /usr/…"   38 seconds ago   Up 33 seconds              0.0.0.0:8080->8080/tcp, 0.0.0.0:50000->50000/tcp   xenodochial_ganguly
d2533ac20f55   hello-world       "/hello"                 30 minutes ago   Exited (0) 30 minutes ago                                                     jovial_rhodes
1f131ddeffda   hello-world       "/hello"                 32 minutes ago   Exited (0) 32 minutes ago                                                     quirky_beaver
charan@LAPTOP-ONEMN5DS:~$
charan@LAPTOP-ONEMN5DS:~$
charan@LAPTOP-ONEMN5DS:~$ docker top c94ea4a6f1b0
UID                 PID                 PPID                C                   STIME               TTY                 TIME                CMD
1000                3363                3343                0                   00:09               ?                   00:00:00            /sbin/tini -- /usr/local/bin/jenkins.sh
1000                3400                3363                99                  00:09               ?                   00:01:28            java -Duser.home=/var/jenkins_home -Djenkins.model.Je
ns.slaveAgentPort=50000 -jar /usr/share/jenkins/jenkins.war
```

# Docker – docker container commands.

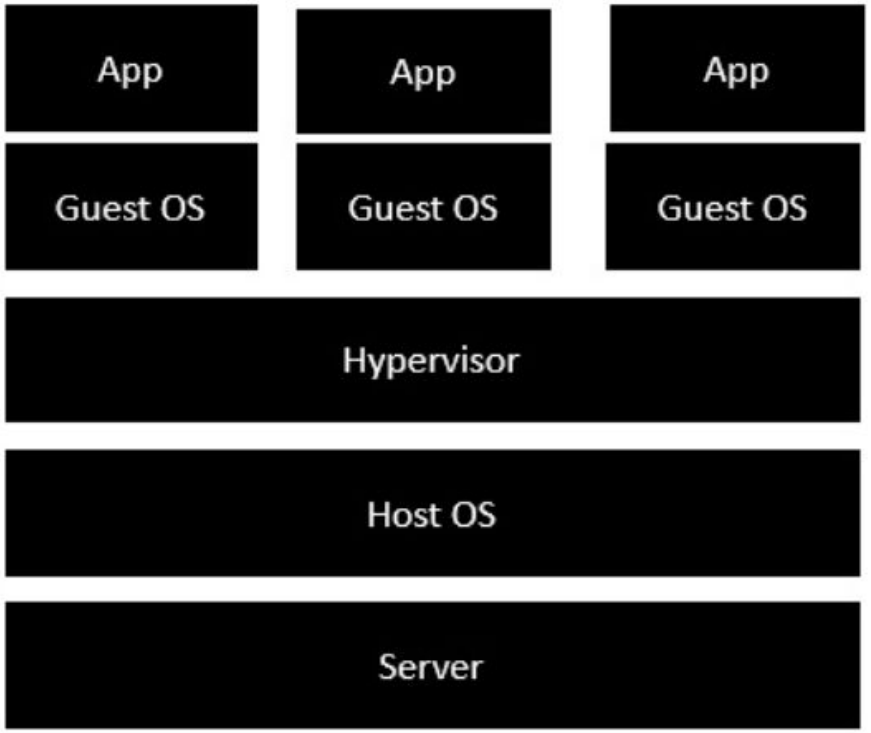
- **$docker stop <container id>** Will stop running this container.

```
charan@LAPTOP-ONEMN5DS:~$
charan@LAPTOP-ONEMN5DS:~$ docker stop c94ea4a6f1b0
c94ea4a6f1b0
charan@LAPTOP-ONEMN5DS:~$ docker ps -a
CONTAINER ID   IMAGE             COMMAND                  CREATED          STATUS                        PORTS     NAMES
c94ea4a6f1b0   jenkins/jenkins   "/sbin/tini -- /usr/…"   5 minutes ago    Exited (143) 17 seconds ago             xenodochial_ganguly
d2533ac20f55   hello-world       "/hello"                 35 minutes ago   Exited (0) 35 minutes ago               jovial_rhodes
1f131ddeffda   hello-world       "/hello"                 36 minutes ago   Exited (0) 36 minutes ago               quirky_beaver
charan@LAPTOP-ONEMN5DS:~$ docker top c94ea4a6f1b0
Error response from daemon: Container c94ea4a6f1b0b62d1ab9b39abcbfbd108c652ae63ee32414dfc85de2d3751964 is not running
charan@LAPTOP-ONEMN5DS:~$
```

- $docker rm <container id> Will remove the container from the hub.

```
charan@LAPTOP-ONEMN5DS:~$ docker ps -a
CONTAINER ID   IMAGE             COMMAND                  CREATED          STATUS                        PORTS     NAMES
c94ea4a6f1b0   jenkins/jenkins   "/sbin/tini -- /usr/…"   7 minutes ago    Exited (143) 2 minutes ago              xenodochial_ganguly
d2533ac20f55   hello-world       "/hello"                 37 minutes ago   Exited (0) 37 minutes ago               jovial_rhodes
1f131ddeffda   hello-world       "/hello"                 38 minutes ago   Exited (0) 38 minutes ago               quirky_beaver
charan@LAPTOP-ONEMN5DS:~$
charan@LAPTOP-ONEMN5DS:~$ docker rm c94ea4a6f1b0
c94ea4a6f1b0
charan@LAPTOP-ONEMN5DS:~$ docker ps -a
CONTAINER ID   IMAGE         COMMAND    CREATED          STATUS                        PORTS     NAMES
d2533ac20f55   hello-world   "/hello"   37 minutes ago   Exited (0) 37 minutes ago               jovial_rhodes
1f131ddeffda   hello-world   "/hello"   38 minutes ago   Exited (0) 38 minutes ago               quirky_beaver
charan@LAPTOP-ONEMN5DS:~$
```

# Docker – Architecture




The server is the physical server that is used to host multiple virtual machines.

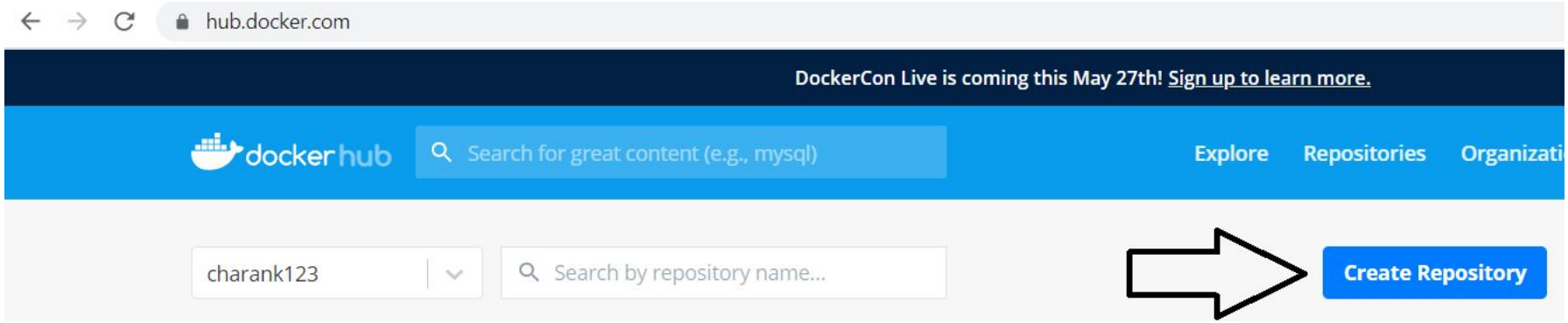
The Host OS is the base machine such as Linux or Windows.

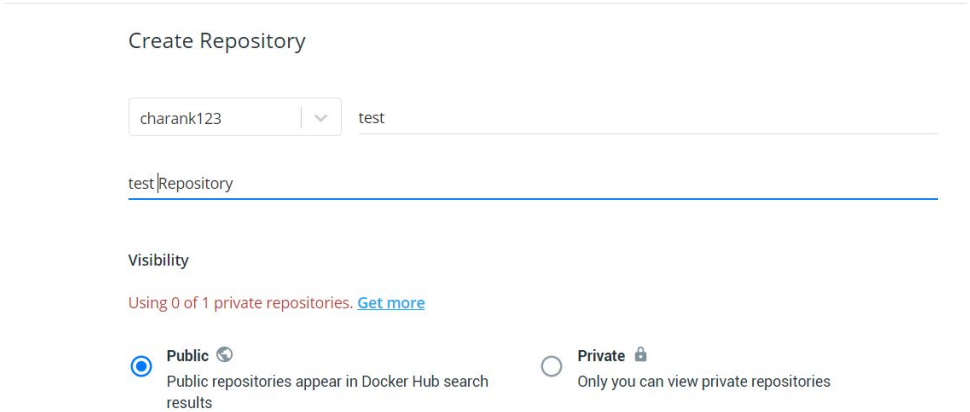
The Hypervisor is either

# Docker – Public Repositories

- Public repositories can be used to host Docker images which can be used by everyone else.
- Visit the URL : https://hub.docker.com/ and login.
- Create a repository.
- Tag the image from local hub and push it in the created repository.
- Result can be verified in the docker hub.
- Created image can be run on a container.

# Docker – Public repository




```
charan@LAPTOP-ONEMN5DS:~$ docker images
REPOSITORY          TAG       IMAGE ID       CREATED         SIZE
docker101tutorial   latest    fc67d866c198   46 hours ago    27.9MB
jenkins/jenkins     latest    10e33bea4cd2   3 days ago      573MB
node                latest    ebcfbb59a4bd   10 days ago     936MB
hello-world         latest    bf756fb1ae65   13 months ago   13.3kB
charan@LAPTOP-ONEMN5DS:~$
charan@LAPTOP-ONEMN5DS:~$ docker tag bf756fb1ae65 charank123/test
charan@LAPTOP-ONEMN5DS:~$
```

```
charan@LAPTOP-ONEMN5DS:~$ docker images
REPOSITORY          TAG       IMAGE ID       CREATED         SIZE
docker101tutorial   latest    fc67d866c198   46 hours ago    27.9MB
jenkins/jenkins     latest    10e33bea4cd2   3 days ago      573MB
node                latest    ebcfbb59a4bd   10 days ago     936MB
charank123/test     latest    bf756fb1ae65   13 months ago   13.3kB
hello-world         latest    bf756fb1ae65   13 months ago   13.3kB
charan@LAPTOP-ONEMN5DS:~$
charan@LAPTOP-ONEMN5DS:~$ docker tag bf756fb1ae65 charank123/test:2.0
charan@LAPTOP-ONEMN5DS:~$
charan@LAPTOP-ONEMN5DS:~$ docker push charank123/test:2.0
The push refers to repository [docker.io/charank123/test]
9c27e219663c: Layer already exists
2.0: digest: sha256:90659bf80b44ce6be8234e6ff90a1ac34acbeb826903b02cfa0da11c82cbc042 size: 525
charan@LAPTOP-ONEMN5DS:~$ docker images
REPOSITORY          TAG       IMAGE ID       CREATED         SIZE
docker101tutorial   latest    fc67d866c198   46 hours ago    27.9MB
jenkins/jenkins     latest    10e33bea4cd2   3 days ago      573MB
node                latest    ebcfbb59a4bd   10 days ago     936MB
hello-world         latest    bf756fb1ae65   13 months ago   13.3kB
charank123/test     2.0       bf756fb1ae65   13 months ago   13.3kB
charank123/test     latest    bf756fb1ae65   13 months ago   13.3kB
charan@LAPTOP-ONEMN5DS:~$ docker run charank123/test:2.0
```

```
charan@LAPTOP-ONEMN5DS:~$ docker ps -a
CONTAINER ID   IMAGE                 COMMAND    CREATED          STATUS                      PORTS     NAMES
e613e9ce16cf   charank123/test:2.0   "/hello"   46 seconds ago   Exited (0) 42 seconds ago           hungry_perlman
6220f8ffa881   bf756fb1ae65          "/hello"   23 minutes ago   Exited (0) 23 minutes ago           distracted_turing
d2533ac20f55   hello-world           "/hello"   36 hours ago     Exited (0) 36 hours ago             jovial_rhodes
1f131ddeffda   hello-world           "/hello"   36 hours ago     Exited (0) 36 hours ago             quirky_beaver
charan@LAPTOP-ONEMN5DS:~$
```

# Docker – public repositories

**$docker images** Will display all the images.

**$docker tag bf756fb1ae65 charank123/test:2.0**

Any image will have tag. Tag of the image can be seen in docker images command. This image tag has to be attached to created repository. Charank123 is the login, test is the name of the repository and 2.0 is the tag name.

**$ docker push charank123/test:2.0**

This command will push the image from local hub to test repository with a tag of 2.0

**$docker images** Check whether a new image is created in test repository.

**$ docker run charank123/test:2.0**

This command will run the image with a tag of 2.0 under test repository in a container.

# Docker - info

- docker info command - displays system wide information regarding the Docker installation. Information displayed includes the kernel version, number of containers and images. The number of images shown is the number of unique images. The same image tagged under different names is counted only once.

charan@LAPTOP-ONEMN5DS: ~

```
charan@LAPTOP-ONEMN5DS:~$ docker info
Client:
 Context:    default
 Debug Mode: false
 Plugins:
  app: Docker App (Docker Inc., v0.9.1-beta3)
  buildx: Build with BuildKit (Docker Inc., v0.5.1-docker)
  scan: Docker Scan (Docker Inc., v0.5.0)

Server:
 Containers: 4
  Running: 0
  Paused: 0
  Stopped: 4
 Images: 4
 Server Version: 20.10.2
 Storage Driver: overlay2
  Backing Filesystem: extfs
  Supports d_type: true
  Native Overlay Diff: true
 Logging Driver: json-file
 Cgroup Driver: cgroupfs
 Cgroup Version: 1
 Plugins:
  Volume: local
  Network: bridge host ipvlan macvlan null overlay
  Log: awslogs fluentd gcplogs gelf journald json-file local logentries splunk syslog
 Swarm: inactive
 Runtimes: io.containerd.runc.v2 io.containerd.runtime.v1.linux runc
 Default Runtime: runc
 Init Binary: docker-init
 containerd version: 269548fa27e0089a8b8278fc4fc781d7f65a939b
 runc version: ff819c7e9184c13b7c2607fe6c30ae19403a7aff
 init version: de40ad0
 Security Options:
  seccomp
   Profile: default
 Kernel Version: 5.4.72-microsoft-standard-WSL2
```

# Docker – Setting Node.js

- Pull the node image.
- After that verify using the command docker images.
- Create a HelloWorld.js and type the code.
- Run this file in the node container.

charan@LAPTOP-ONEMN5DS: ~

```
charan@LAPTOP-ONEMN5DS:~$ docker images
REPOSITORY          TAG       IMAGE ID       CREATED        SIZE
docker101tutorial   latest    fc67d866c198   3 days ago     27.9MB
jenkins/jenkins     latest    10e33bea4cd2   5 days ago     573MB
node                latest    ebcfbb59a4bd   12 days ago    936MB
charank123/test     2.0       bf756fb1ae65   13 months ago  13.3kB
charank123/test     latest    bf756fb1ae65   13 months ago  13.3kB
hello-world         latest    bf756fb1ae65   13 months ago  13.3kB
charan@LAPTOP-ONEMN5DS:~$
charan@LAPTOP-ONEMN5DS:~$ cat HelloWorld.js
console.log ("Hey this is Charan");
charan@LAPTOP-ONEMN5DS:~$
charan@LAPTOP-ONEMN5DS:~$ docker run -it --rm --name=HelloWorld -v "$PWD":/usr/src/app -w /usr/src/app node node HelloWo
rld.js
Hey this is Charan
```

# Docker – run HelloWorld.js file.

- $ docker run -it --rm --name=HelloWorld -v "$PWD":/usr/src/app -w /usr/src/app node node HelloWorld.js

- -it – Runs in interactive mode.
- -rm – Removes the node from container after running the code.
- --name=HelloWorld  Giving the name “HelloWorld” to the container.
- "$PWD":/usr/src/app  Setting the present working directory as /usr/src/app.
- -w  Working directory used by node.js
- First node is used to run the node image.
- Second node is used to run the node and HelloWorld.js in the node container.

# Docker - Cloud

The Docker Cloud is a service provided by Docker in which you can carry out the following operations −

Nodes − You can connect the Docker Cloud to your existing cloud providers such as Azure and AWS to spin up containers on these environments.

Cloud Repository − Provides a place where you can store your own repositories.

Continuous Integration − Connect with Github and build a continuous integration pipeline.

Application Deployment − Deploy and scale infrastructure and containers.

Continuous Deployment − Can automate deployments.

# Docker – Continuous Integration

Docker has integrations with many Continuous Integrations tools, which also includes the popular CI tool known as Jenkins. Within Jenkins, you have plugins available which can be used to work with containers.

# Docker – Kubernetes Architecture

# Docker – Working of Kubernetes

# Docker – CLI

- All the docker commands can be seen in this site: https://docs.docker.com/engine/reference/commandline/docker/

Thank you!!