

# Docker Notes

## 1) Docker Installation.

**apt install docker.io -y** (Command for docker installation on Ubuntu)

```
root@ip-172-31-57-88:/home/ubuntu# apt install docker.io -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base pigz runc ubuntu-fan
Suggested packages:
  ifupdown aufs-tools cgroupfs-mount | cgroup-lite debootstrap docker-buildx docker-compose-v2 docker-doc rinse
  zfs-fuse | zfsutils
The following NEW packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base docker.io pigz runc ubuntu-fan
0 upgraded, 8 newly installed, 0 to remove and 4 not upgraded.
Need to get 76.8 MB of archives.
After this operation, 289 MB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 pigz amd64 2.8-1 [65.6 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 bridge-utils amd64 1.7.1-1ubuntu2 [33.9 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 runc amd64 1.1.12-0ubuntu3 [8599 kB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 containerd amd64 1.7.12-0ubuntu4 [38.6 MB]
The following NEW packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base docker.io pigz runc ubuntu-fan
0 upgraded, 8 newly installed, 0 to remove and 4 not upgraded.
Need to get 76.8 MB of archives.
After this operation, 289 MB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 pigz amd64 2.8-1 [65.6 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 bridge-utils amd64 1.7.1-1ubuntu2 [33.9 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 runc amd64 1.1.12-0ubuntu3 [8599 kB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 containerd amd64 1.7.12-0ubuntu4 [38.6 MB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 dns-root-data all 2023112702-willsync1 [4450 B]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 dnsmasq-base amd64 2.90-2build2 [375 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 docker.io amd64 24.0.7-0ubuntu4 [29.1 MB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 ubuntu-fan all 0.12.16 [35.2 kB]
Fetched 76.8 MB in 1s (75.1 MB/s)
Preconfiguring packages ...
Selecting previously unselected package pigz.
(Reading database ... 114320 files and directories currently installed.)
Preparing to unpack .../0-pigz_2.8-1_amd64.deb ...
Unpacking pigz (2.8-1) ...
Selecting previously unselected package bridge-utils.
Preparing to unpack .../1-bridge-utils_1.7.1-1ubuntu2_amd64.deb ...
Unpacking bridge-utils (1.7.1-1ubuntu2) ...
Selecting previously unselected package runc.
Preparing to unpack .../2-runc_1.1.12-0ubuntu3_amd64.deb ...
Unpacking runc (1.1.12-0ubuntu3) ...
Selecting previously unselected package containerd.
Preparing to unpack .../3-containerd_1.7.12-0ubuntu4_amd64.deb ...
Unpacking containerd (1.7.12-0ubuntu4) ...
Setting up runc (1.1.12-0ubuntu3) ...
Setting up dns-root-data (2023112702-willsync1) ...
Setting up bridge-utils (1.7.1-1ubuntu2) ...
Setting up pigz (2.8-1) ...
Setting up containerd (1.7.12-0ubuntu4) ...
Created symlink /etc/systemd/system/multi-user.target.wants/containerd.service → /usr/lib/systemd/system/containerd.service.
Setting up ubuntu-fan (0.12.16) ...
Created symlink /etc/systemd/system/multi-user.target.wants/ubuntu-fan.service → /usr/lib/systemd/system/ubuntu-fan.service.
Setting up docker.io (24.0.7-0ubuntu4) ...
info: Selecting GID from range 180 to 999 ...
info: Adding group 'docker' (GID 114) ...
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /usr/lib/systemd/system/docker.socket.
Processing triggers for dbus (1.14.18-4ubuntu4) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
root@ip-172-31-57-88:/home/ubuntu#
```

**systemctl start docker** (Starts docker service)

```
root@ip-172-31-57-88:/home #
root@ip-172-31-57-88:/home/ubuntu# systemctl start docker
root@ip-172-31-57-88:/home/ubuntu#
```

**systemctl status docker** (Gives the status of docker service)

```
root@ip-172-31-57-88: /home X + v
root@ip-172-31-57-88:/home/ubuntu# systemctl start docker
root@ip-172-31-57-88:/home/ubuntu# systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; preset: enabled)
   Active: active (running) since Sat 2024-08-10 08:19:30 UTC; 22s ago
   TriggeredBy: ● docker.socket
     Docs: https://docs.docker.com
    Main PID: 4233 (dockerd)
      Tasks: 8
     Memory: 25.5M (peak: 25.6M)
        CPU: 293ms
    CGroup: /system.slice/docker.service
           └─4233 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Aug 10 08:19:29 ip-172-31-57-88 systemd[1]: Starting docker.service - Docker Application Container Engine...
Aug 10 08:19:29 ip-172-31-57-88 dockerd[4233]: time="2024-08-10T08:19:29.984929428Z" level=info msg="Starting up"
Aug 10 08:19:29 ip-172-31-57-88 dockerd[4233]: time="2024-08-10T08:19:29.986758323Z" level=info msg="detected 127.0.0.55"
Aug 10 08:19:30 ip-172-31-57-88 dockerd[4233]: time="2024-08-10T08:19:30.057040080Z" level=info msg="Loading containers"
Aug 10 08:19:30 ip-172-31-57-88 dockerd[4233]: time="2024-08-10T08:19:30.312462198Z" level=info msg="Loading containers"
Aug 10 08:19:30 ip-172-31-57-88 dockerd[4233]: time="2024-08-10T08:19:30.330746311Z" level=info msg="Docker daemon" com
Aug 10 08:19:30 ip-172-31-57-88 dockerd[4233]: time="2024-08-10T08:19:30.330884356Z" level=info msg="Daemon has complet
Aug 10 08:19:30 ip-172-31-57-88 dockerd[4233]: time="2024-08-10T08:19:30.374145032Z" level=info msg="API listen on /run
Aug 10 08:19:30 ip-172-31-57-88 systemd[1]: Started docker.service - Docker Application Container Engine.
[lines 1-21/21 (END)]
```

**Docker --version** (Give the Version of docker installed)

```
root@ip-172-31-57-88: /home X + v
root@ip-172-31-57-88:/home/ubuntu# docker --version
Docker version 24.0.7, build 24.0.7-0ubuntu4
```

**Docker version**

(If we give without – version, then it gives complete details of client and server version), By Default what is installed is client.

```
root@ip-172-31-57-88:/home/ubuntu# docker version
Client:
 Version:           24.0.7
 API version:       1.43
 Go version:        go1.22.2
 Git commit:        24.0.7-0ubuntu4
 Built:             Wed Apr 17 20:08:25 2024
 OS/Arch:           linux/amd64
 Context:           default

Server:
 Engine:
  Version:          24.0.7
  API version:      1.43 (minimum version 1.12)
  Go version:       go1.22.2
  Git commit:       24.0.7-0ubuntu4
  Built:            Wed Apr 17 20:08:25 2024
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.7.12
  GitCommit:
 runc:
  Version:          1.1.12-0ubuntu3
  GitCommit:
 docker-init:
  Version:          0.19.0
  GitCommit:
```

## 2) Docker pull tomcat

This command helps only in pulling/downloading the docker image from docker Hub. It does not create container.

In our case we are pulling docker image for tomcat. (for example)  
(Note: This is pulling the latest docker image only) Screenshot below shows the same

```
root@ip-172-31-57-88: /home X + v
root@ip-172-31-57-88:/home/ubuntu# docker pull tomcat
Using default tag: latest
latest: Pulling from library/tomcat
2b3981cac065: Pull complete
3e44a677d4d8: Pull complete
f561a59c5174: Pull complete
8fc851d1d586: Pull complete
4723684ec455: Pull complete
a6296e344bb6: Pull complete
4f4fb700ef54: Pull complete
7440b1c81218: Pull complete
Digest: sha256:3c87192b2e3627fe0c81718107062b6f3448aebb132d4006c18b4b2a54410156
Status: Downloaded newer image for tomcat:latest
docker.io/library/tomcat:latest
root@ip-172-31-57-88:/home/ubuntu#
```

### 3) Docker search tomcat

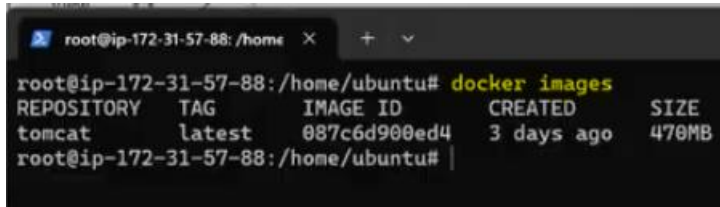
This command will give you all the docker images of a specific tool or product. In our case, the above command gives all the images of tomcat present on docker hub.

```
root@ip-172-31-57-88: /home X + v
Status: Downloaded newer image for tomcat:latest
docker.io/library/tomcat:latest
root@ip-172-31-57-88:/home/ubuntu# docker search tomcat
NAME                DESCRIPTION                STARS     OFFICIAL   AUTOMATED
tomcat              Apache Tomcat is an open source implementati... 3688      [OK]
tomEE              Apache TomEE is an all-Apache Java EE certif... 116       [OK]
vulhub/tomcat      0
lucee/lucee52      Lucee 5.2 application engine running on Apac... 3          [OK]
wnprcehr/tomcat    0
samlpy/tomcat-common 0
bitnami/tomcat     Bitnami container image for Tomcat             50          [OK]
lucee/lucee4        Lucee 4.5 application engine running on Apac... 8          [OK]
secoresearch/tomcat-varnish Tomcat and Varnish 5.0                       0          [OK]
paketobuildpacks/apache-tomcat 0
rapidfort/tomcat9-openjdk11-ib 0
eclipse/hadoop-dev 0
lucee/lucee51-nginx Lucee 5.1 with NGINX running on Apache Tomca... 1          [OK]
bitnamicharts/tomcat 0
lucee/lucee51      Lucee 5.1 application engine running on Apac... 0          [OK]
lucee/lucee52-nginx Lucee 5.2 with NGINX running on Apache Tomca... 11         [OK]
semoss/docker-tomcat Tomcat, Java, Maven, and Git on top of debian 0          [OK]
hivdb/tomcat-with-nucamino 0
lucee/lucee        Lucee application engine running on Apache T... 31
eclipse/rdf4j-workbench 9
chainguard/tomcat  Build, ship and run secure software with Cha... 0
rapidfort/tomcat10-openjdk17-ib 0
rancher/haproxy-tomcat 1
pgkb/pharmcat      Official docker image for running PharmCAT.    1
jelastic/tomcat     An image of the Tomcat Java application serv... 4
root@ip-172-31-57-88:/home/ubuntu#
```

#### 4) Docker images

This command shows you all the images that have been downloaded.

In our case, we have only downloaded tomcat, so it shows tomcat image only.

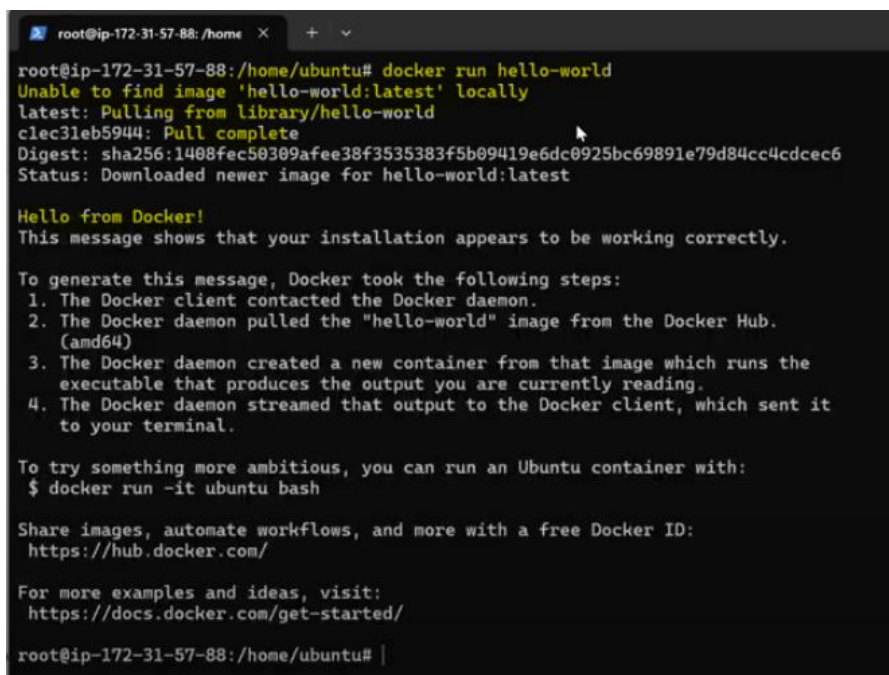


```
root@ip-172-31-57-88: /home X + v
root@ip-172-31-57-88:/home/ubuntu# docker images
REPOSITORY    TAG       IMAGE ID      CREATED       SIZE
tomcat        latest    087c6d900ed4  3 days ago    470MB
root@ip-172-31-57-88:/home/ubuntu#
```

#### 5) Docker run hello-world

This command helps in pulling/downloading the docker image and creating the container from that image and runs the container also.

In our case it is pulling/downloading the hello-world docker image and creating a container from it and runs it also.



```
root@ip-172-31-57-88: /home X + v
root@ip-172-31-57-88:/home/ubuntu# docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:1408fec50309afee38f3535383f5b09419e6dc0925bc69891e79d84cc4cdcec6
Status: Downloaded newer image for hello-world:latest

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/
root@ip-172-31-57-88:/home/ubuntu#
```

We can verify that hello-world image is downloaded by writing Docker images. Below is the output from the screen.

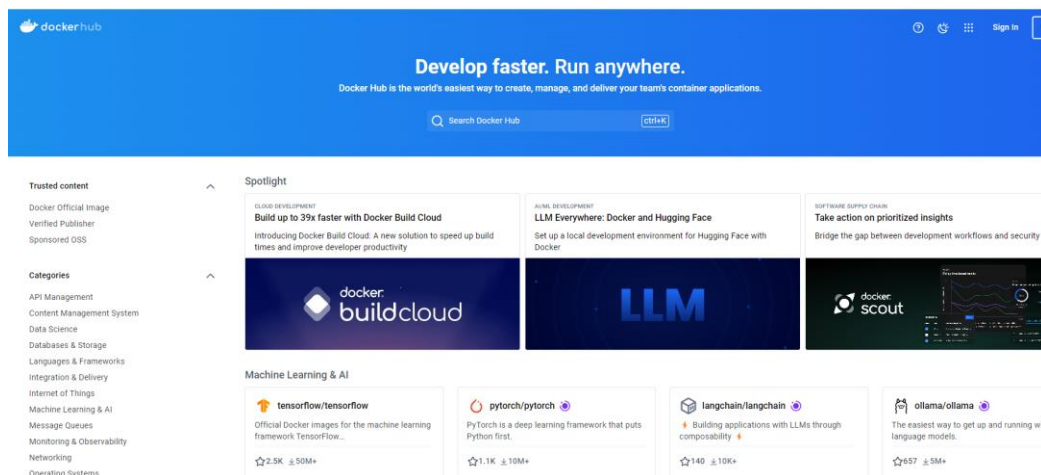
```
root@ip-172-31-57-88: /home
root@ip-172-31-57-88:/home/ubuntu# docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
tomcat         latest    087c6d90ed4   3 days ago    470MB
hello-world    latest    d2c94e258dcb   15 months ago 13.3kB
root@ip-172-31-57-88:/home/ubuntu#
```

## 6) Docker ps -a or -i

This command gives all the containers that are created from the image and also its status. In our case hello-world container. Here container ran the command and then it dies (if Status in below screenshot shows Exited, then container has died after running, it's not running continuously.)

```
root@ip-172-31-57-88:/home/ubuntu# docker ps -a
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS      PORTS   NAMES
58cc72155798   hello-world "/hello"                About a minute ago    Exited (0) About a minute ago    relaxed_gauss
root@ip-172-31-57-88:/home/ubuntu#
```

## 7) Docker Repository (Docker Hub) – It is the place where we can store 1 or more versions of a specific docker image. An image can have 1 or more versions (tags).



## 8) Docker Registry (Docker Trusted Registry) – It's an enterprise-grade storage solution for Docker Images. In other words, its an image storage service similar to GitHub but for Docker Images.



- 9) Docker Engine - Docker Engine is an open-source containerization technology for building and containerizing your applications. Docker Engine acts as a client-server application with:
- A server with a long-running daemon process [dockerd](#).
  - APIs which specify interfaces that programs can use to talk to and instruct the Docker daemon.
  - A command line interface (CLI) client [docker](#).
- 10) Master Server - If the container is created on the docker server, then it's called master server.
- 11) Worker Server - If the container is created on the different server, then it's called worker server.
- 12) DockerFile - Docker can build images automatically by reading the instructions from a Dockerfile. A Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image.
- 13) Docker Container Lifecycle

Below are the steps involved in Docker Container Lifecycle:

**Create container**

```
$ docker create --name ubuntu-cont ubuntu
```

**Run docker container**

```
$ docker run -itd ubuntu
```

```
$ docker run -itd --name ubuntu-cont ubuntu
```

**Pause container**

```
$ docker pause <container-id/name>
```

**Unpause container**

```
$ docker unpause <container-id/name>
```

**Start container**

```
$ docker start <container-id/name>
```

### Stop container

\$ docker stop <container-id/name>

### Restart container

\$ docker restart <container-id/name>

### Kill container

\$ docker kill <container-id/name>

### Destroy container

\$ docker rm <container-id/name>

- 14) Docker Image Layer Caching - Docker Image layer caching is process where if have an image which has some contents in it like application or something and if we want to make a small change in the application, then I will not create the whole image again, it will only recreate that layer of change in the application since rest other things are already cached in the image.

### 15) Docker Info

This 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.

```
root@ip-172-31-93-222: /home/ubuntu
Init Binary: docker-init
containerd version:
runc version:
init version:
Security Options:
  apparmor
  seccomp
    Profile: builtin
  cgroupns
Kernel Version: 6.8.0-1009-aws
Operating System: Ubuntu 24.04 LTS
OSType: linux
Architecture: x86_64
CPUs: 1
Total Memory: 957.4MiB
Name: ip-172-31-93-222
ID: 55985be4-d50f-4e26-8aee-ef366e61cdd3
Docker Root Dir: /var/lib/docker
Debug Mode: false
Experimental: false
Insecure Registries:
  127.0.0.0/8
Live Restore Enabled: false
root@ip-172-31-93-222: /home/ubuntu#
```

## 16) Docker rmi (image name) – docker rmi ubuntu

This command is used to delete the downloaded images. In our case we are deleting ubuntu image.

```
root@ip-172-31-93-222: /home/ubuntu
root@ip-172-31-93-222:/home/ubuntu# docker rmi ubuntu
Untagged: ubuntu:latest
Untagged: ubuntu@sha256:2e863c44b718727c860746568e1d54afd13b2fa71b160f5cd9058fc436217b30
Deleted: sha256:35a88802559dd2077e584394471ddaa1a2c5bfd16893b829ea57619301eb3908
Deleted: sha256:a30a5965a4f7d9d5ff76a46eb8939f58e95be844de1ac4a4b452d5d31158fdea
root@ip-172-31-93-222:/home/ubuntu# docker images_
```

Note: If there is a container running from the image downloaded and you try to delete that image, it will not delete, you will have to delete the container first in order to delete the image. Below screenshot shows the same.

```
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
hello-world   latest    d2c94e258dcb   15 months ago  13.3kB
root@ip-172-31-93-222:/home/ubuntu# docker rmi hello-world
Error response from daemon: conflict: unable to remove repository reference "hello-world" (must force)
- container 60413e0b06db is using its referenced image d2c94e258dcb
root@ip-172-31-93-222:/home/ubuntu#
```

## 17) Docker rm

This command is used to delete the containers created from an image.

## 18) Docker ps -l

This command will only show the last executed container from an image.

```
root@ip-172-31-93-222:/home/ubuntu# docker ps -l
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS
NAMES
3d128c9aadf5   ubuntu   "cat /etc/passwd"       About a minute ago  Exited (0) About a minute ago
xenodochial_swanson
```

## 19) Docker start (container ID)

This command can start the container again with the command that was used to create it.



```

root@ip-172-31-93-222:/home/ubuntu# docker start 3d128c9adf5
3d128c9adf5

```

## 20) Docker Stop (Name or Container ID)

This command can stop the running container.

## 21) Docker run --name myname ubuntu

This command will help to give every container a unique name so that you don't have to remember the container ID.

```

root@ip-172-31-93-222:/home/ubuntu# docker run --name test ubuntu
root@ip-172-31-93-222:/home/ubuntu# docker ps -l

```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
59c67996462a	ubuntu	"/bin/bash"	13 seconds ago	Exited (0) 11 seconds ago		test

Below screenshot shows container created with our given name.

```

root@ip-172-31-93-222:/home/ubuntu# docker ps -a

```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
59c67996462a	ubuntu	"/bin/bash"	56 seconds ago	Exited (0) 54 seconds ago		test
3d128c9adf5	ubuntu	"cat /etc/passwd"	7 minutes ago	Exited (0) 4 minutes ago		xenodochial_swanson
b8dd2cd04c40	ubuntu	"cat /etc/issue"	8 minutes ago	Exited (0) 8 minutes ago		clever_chatelet
60413e0b06db	hello-world	"/hello"	24 minutes ago	Exited (0) 24 minutes ago		determined_jemison

## 22) Docker run -it ubuntu bash

This command is used to run the container in interactive mode, container will not die even after executing its task.

```
root@ip-172-31-93-222: /home/ubuntu
root@ip-172-31-93-222:/home/ubuntu# docker run -it ubuntu bash
root@99a512aa1c77:/# cd /tmp
root@99a512aa1c77:/tmp# touch abcd.txt
root@99a512aa1c77:/tmp# ls -ltr
total 0
-rw-r--r-- 1 root root 0 Aug 11 07:46 abcd.txt
root@99a512aa1c77:/tmp# root@ip-172-31-93-222:/home/ubuntu# docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
99a512aa1c77	ubuntu	"bash"	About a minute ago	Up About a minute
wonderful_elbakyan	ubuntu	"/bin/bash"	3 minutes ago	Exited (0) 3 minutes ago
59c67996462a	ubuntu	test	10 minutes ago	Exited (0) 7 minutes ago
3d128c9aadf5	ubuntu	"cat /etc/passwd"	11 minutes ago	Exited (0) 11 minutes ago
xenodochial_swanson	ubuntu	"cat /etc/issue"	27 minutes ago	Exited (0) 27 minutes ago
b8dd2cd04c40	hello-world	"/hello"		
clever_chatelet	determined_jemison			
60413e0b06db				

- **-i** is used to start an interactive session.
- **-t** allocates a tty and attaches stdin and stdout.
- **ubuntu** is the image that we used to create the container.
- **bash** (or **/bin/bash**) is the command that we are running inside the Ubuntu container.

To quit and return to host from the running container session you must type **exit** command. The exit command terminates all the container processes and stops it.

# exit

If you're interactively logged on container terminal prompt and you need to keep the container in running state but exit from the interactive session, you can quit the console and return to host terminal by pressing **Ctrl p** and **q** keys.

## 23) Docker attach <container id >

This command is used to reconnect to the running container, for that we need the container ID or name. We can check the container ID by running **docker ps** command and then run **docker attach** command with container ID to connect back to that container.

## 24) Docker kill <container id>

This command is used to stop a running container from the host session.

## 25) Docker Swarm

- For creating Docker Swarm, we need two Ec2 instances which have **docker install** on both of them, one is Master (m) and other is worker (w)

<input checked="" type="checkbox"/>	m	i-01af49df712fb6a70	Running	t2.micro	Initializing	<a href="#">View alarms</a>
<input type="checkbox"/>	w	i-0392498b4d9093804	Running	t2.micro	Initializing	<a href="#">View alarms</a>

- Define their hostname in ubuntu terminal so that there is no confusion.
- Now setup password less ssh between master and worker by following below steps.

On master: Type **ssh-keygen** and follow all steps as follows to generate public key and private key

```
root@m:/home/ubuntu# ssh-keygen
Generating public/private ed25519 key pair.
Enter file in which to save the key (/root/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_ed25519
Your public key has been saved in /root/.ssh/id_ed25519.pub
The key's fingerprint is:
SHA256:eSWJYcS10FhPuSI4wWcyW3xrzXPWB3v7KzcsVPKQAds root@m
The key's randomart image is:
+--[ED25519 256]--+
|  . . o=.oo. |
|  = +o*.oo.. |
|  X..++=o.Eoo |
|  + . +o=oo+o.o |
|  . oS..+ =o. |
|  . . . o |
|  . . . |
|  o +. |
|  +.o |
+-----[SHA256]-----+
```

Now do **cat /root/.ssh/id\_ed25519.pub** and copy all the contents inside it.

```
root@m:/home/ubuntu# cat /root/.ssh/id_ed25519.pub
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIFSsYlGy8mc9GtH1yp6qI+STmo74Z8yqRHZviTgSu/Qy root@m
```

On Worker: Do **cat >> authorized\_keys** (append authorized\_keys file and paste the contents copied from master server)

```
root@w:/home/ubuntu# cat>>/root/.ssh/authorized_keys
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIFSsYlGy8mc9GtH1yp6qI+STmo74Z8yqRHZviTgSu/Qy root@m
^C
root@w:/home/ubuntu# |
```

Now we will ssh to worker node from master using the Public IP address

```
root@m:/home/ubuntu# ssh 52.91.179.231
The authenticity of host '52.91.179.231 (52.91.179.231)' can't be established.
ED25519 key fingerprint is SHA256:HnLaE5TZWC0msF0wdFs09zU/XgdBeCcJ3hJrssBjjHs.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])?
```

```
root@w:~# |
```

If we exit the worker, we will return back to the master node.

```
root@w:~# exit
logout
Connection to 52.91.179.231 closed.
root@m:/home/ubuntu# |
```

- Now run the below command to master node to make it manager IP

**Docker swarm init --advertise-addr (Public IP of Master Node)**

Below screenshot gives the output

```
root@m:/home/ubuntu# docker swarm init --advertise-addr 54.161.156.162
Swarm initialized: current node (8vpuygcs875423ahh35j0rlih) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-0x13gqyebm2qssi6fltfp0562zbcwq5wfyvakf7jim5c9jj9ut-d1mzbmqu0tff53xx4scp8iye1 54.161.156.162:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.
```

- Now run the next command highlighted the above screenshot on the worker node so that worker node joins swarm as worker.

Command differs scenario to scenario as token will be different in different cases.

```
ubuntu@ip-172-31-25-32:~$ sudo su
root@w:/home/ubuntu# docker swarm join --token SWMTKN-1-0x13gqyebm2qssi6fltfp0562zbcwq5wfyvakf7jim5c9jj9ut-d1mzmq0tff5
3xx4scp8iye1 54.161.156.162:2377
This node joined a swarm as a worker.
```

- To verify whether the node is joined as worker, we will run the below command from master.

### Docker node ls

```
root@m:/home/ubuntu# docker node ls
ID                HOSTNAME    STATUS    AVAILABILITY    MANAGER STATUS    ENGINE VERSION
8vpuygcs875423ahh35j0r1ih * m          Ready     Active           leader            24.0.7
l8bbi4c6s6r0jdd3zab80mo74 w          Ready     Active           -                 24.0.7
root@m:/home/ubuntu#
```

The above output gives you both manager (leader) and worker details.

- Now we will create services (pod in Kubernetes) in docker swarm by running the below command on master node.

### Docker service create --replicas 3 -p 80:80 --name web nginx

Above command will create 3 replicas (containers) on both manager and worker with the image name web and container is created for nginx with port mapping 80:80. (1<sup>st</sup> one is VM port and 2<sup>nd</sup> one after: is your container port)

```
root@m:/home/ubuntu# docker service create --replicas 3 -p 80:80 --name web nginx
p3vvkvuvlllofvwk2fx8o37tne
overall progress: 3 out of 3 tasks
1/3: running [=====]
2/3: running [=====]
3/3: running [=====]
verify: Service converged
root@m:/home/ubuntu#
```



- Next command will list the service (containers) is created or not.

### Docker service ls

```
root@m:/home/ubuntu# docker service ls
ID                NAME    MODE     REPLICAS  IMAGE      PORTS
p3vvkvuvllof     web     replicated 3/3        nginx:latest *:80->80/tcp
root@m:/home/ubuntu#
```

In the above command, we can see under replicas that it is showing 3/3, which means all 3 containers are ready, it shows service name is web, image is nginx-latest, and port is 80:80

- We will test nginx is running on containers or not by checking the public IP on port 80 on web console. Below screenshot shows, it worked for master node public IP. Same we will verify for worker node Public IP also and that also worked.

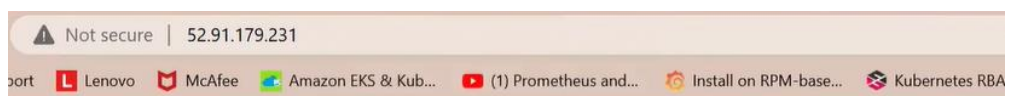


## Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to [nginx.org](https://nginx.org).  
Commercial support is available at [nginx.com](https://nginx.com).

*Thank you for using nginx.*



## Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to [nginx.org](https://nginx.org).  
Commercial support is available at [nginx.com](https://nginx.com).

*Thank you for using nginx.*

- Next command will check how many services are running on master and worker out of 3 services (containers) that we have created.

Docker service ps web (where web is the service name that we have created)

```
root@m:/home/ubuntu# docker service ps web
```

ID	NAME	IMAGE	NODE	DESIRED STATE	CURRENT STATE	ERROR	PORTS
zdkv4gw43n8p	web.1	nginx:latest	w	Running	Running 3 minutes ago		
t5xqqouq9nnq	web.2	nginx:latest	m	Running	Running 3 minutes ago		
9doelvbu6liz	web.3	nginx:latest	w	Running	Running 3 minutes ago		

```
root@m:/home/ubuntu#
```

Above command shows that 2 service (containers) are running on worker node and 1 is running on master node. So, the general working model is that master also acts as worker sometimes, but it always has the least load. It also depends on the load on worker nodes, if it is too much on worker nodes, then also service (container) is created on master node.

- Now if we want to scale the replicas up or down, then we will use the below command.

**Docker service scale web=5**

```
root@m:/home/ubuntu# docker service scale web=5
web scaled to 5
overall progress: 5 out of 5 tasks
1/5: running [=====>]
2/5: running [=====>]
3/5: running [=====>]
4/5: running [=====>]
5/5: running [=====>]
verify: Service converged
root@m:/home/ubuntu#
```

- Now we will install ctop tool for docker monitoring.
  - We will run the below command to install ctop for monitoring docker.

**wget**

<https://github.com/bcicen/ctop/releases/download/v7.0.1/ctop-0.7.1-linux-amd64> -O /usr/local/bin/ctop

```

root@m:/home/ubuntu# wget https://github.com/bcicen/ctop/releases/download/v0.7.1/ctop-0.7.1-linux-amd64 -O /usr/local/bin/ctop
--2024-08-17 07:01:54-- https://github.com/bcicen/ctop/releases/download/v0.7.1/ctop-0.7.1-linux-amd64
Resolving github.com (github.com)... 140.82.113.4
Connecting to github.com (github.com)|140.82.113.4|:443... connected.
HTTP request sent, awaiting response...

```

**chmod +x /usr/local/bin/ctop**

```

root@m:/home/ubuntu# chmod +x /usr/local/bin/ctop
root@m:/home/ubuntu#

```

Now we will run **ctop** command and will see the below output

ctop - 07:02:41 UTC 3 containers							
NAME	CID	CPU	MEM	NET RX/TX	IO R/W	PIDS	
web.2.t5xqgouq9n...	cb6dfa6a9c38	0%	2M / 957M	2K / 1K	0B / 0B	2	
web.4.zrllly07cmw...	5f1687b9d89e	0%	2M / 957M	962B / 0B	0B / 0B	2	
web.5.jcpav9zi9q...	0ad7bf25262b	0%	2M / 957M	962B / 0B	0B / 0B	2	

If we run the above command ctop in master, then it will show only containers that are created in master node, not the worker node. To see the containers on worker node, we need to install and run the ctop command on the worker node.

- One thing regarding leader election in docker swarm is that if the leader is not available, then a new leader will be elected from one of the worker nodes.

## 26) Set up a local registry

A registry can be considered private if pulling requires authentication.

**Docker run -d -p 50000:5000 --restart always --name my-registry registry:latest**

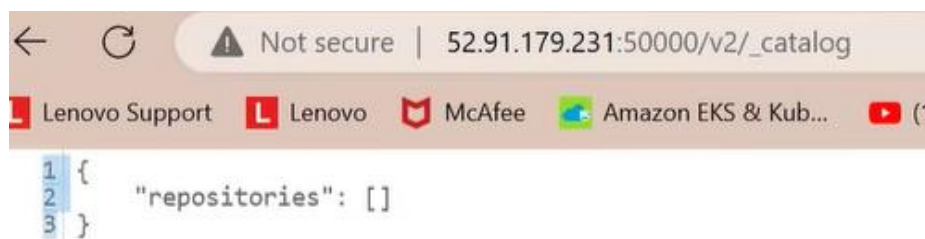
```

root@m:/home/ubuntu# docker run -d -p 50000:5000 --restart always --name my-registry registry
Unable to find image 'registry:latest' locally
latest: Pulling from library/registry
930bdd4d222e: Pull complete
a15309931e05: Pull complete
6263fb9c821f: Pull complete
86c1d3af3872: Pull complete
a37b1bf6a96f: Pull complete
Digest: sha256:12120425f07de11a1b899e418d4b0ea174c8d4d572d45bdb640f93bc7ca06a3d
Status: Downloaded newer image for registry:latest
219eff391bd816ec4e8649b72c21ca3af6572fb18167c747ebca217957bd536d

```

- Above command downloads the registry image which is tagged latest. This tag references the latest version of the registry at the time of this writing.
- Exposes 5000 port to the host, under the same port
- Gives the container the name registry instead of assigning it a random name.
- -d states that the docker container creation from image will run in the background.
- --restart always states that container will remain running.
- --name, you will give the name of your registry.
- Registry: latest, it pull and run the latest image of registry.
- Now we need to access the registry from the URL. For that we need to access it using the below link and verify whether the registry is up and running and that you have no repositories pushed to it. If below screenshot is there then registry is up and running.

[http://localhost \(public IP\):50000/v2/\\_catalog](http://localhost (public IP):50000/v2/_catalog)



- Now we will create a tag on nginx or any image and push the image to our registry.

**Docker tag [Image] [Public IP]:5000 [port of container]/ [tag name]**

Below screenshot gives you the example.

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
nginx	latest	5ef79149e0ec	2 days ago	188MB
registry	latest	cfb4d9904335	10 months ago	25.4MB
root@w:/home/ubuntu# docker tag nginx 52.91.179.231:5000/nginx:10				
root@w:/home/ubuntu# docker images				
REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
52.91.179.231:5000/nginx	10	5ef79149e0ec	2 days ago	188MB
nginx	latest	5ef79149e0ec	2 days ago	188MB
registry	latest	cfb4d9904335	10 months ago	25.4MB
root@w:/home/ubuntu#				

- Now we will push the image that is tagged into our registry.

Docker push [Public IP]:5000/nginx

## 27) Docker File and Docker Image creation from Docker File

- We will now create a docker file by doing the below.

Vi Dockerfile

FROM ubuntu

MAINTAINER your\_name Sourabh

RUN apt-get -y install apache2

RUN echo "Hello Apache Server on Ubuntu Docker" > /var/www/html/index.html

EXPOSE 80

CMD /usr/sbin/apache2ctl -D FOREGROUND

```
FROM ubuntu
MAINTAINER your_name ajay
RUN apt-get -y install apache2
RUN echo "Hello Apache server on Ubuntu Docker" > /var/www/html/index.html
EXPOSE 80
CMD /usr/sbin/apache2ctl -D FOREGROUND
```

```
root@w:/home/ubuntu# ls -la
total 36
drwxr-x--- 4 ubuntu ubuntu 4096 Aug 17 08:05 .
drwxr-xr-x 3 root   root   4096 Aug 17 06:45 ..
-rw----- 1 ubuntu ubuntu   8 Aug 17 06:56 .bash_history
-rw-r--r-- 1 ubuntu ubuntu 220 Mar 31 08:41 .bash_logout
-rw-r--r-- 1 ubuntu ubuntu 3771 Mar 31 08:41 .bashrc
drwx----- 2 ubuntu ubuntu 4096 Aug 17 06:47 .cache
-rw-r--r-- 1 ubuntu ubuntu  807 Mar 31 08:41 .profile
drwx----- 2 ubuntu ubuntu 4096 Aug 17 06:45 .ssh
-rw-r--r-- 1 ubuntu ubuntu   0 Aug 17 06:47 .sudo_as_admin_successful
-rw-r--r-- 1 root   root   187 Aug 17 08:05 Dockerfile
```

- Once the dockerfile is created we will create the image.

Docker build -t my-simple-web-app/webapplication:1.0 . (This command will not be supported in the latest version of docker) as shown in below screenshot also.

```
root@w:/home/ubuntu# docker build -t my-simple-web-app/webapplication:1.0 .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
Install the buildx component to build images with BuildKit:
https://docs.docker.com/go/buildx/

"docker build" requires exactly 1 argument.
See 'docker build --help'.

Usage:  docker build [OPTIONS] PATH | URL | -

Build an image from a Dockerfile
root@w:/home/ubuntu#
```