Once installed the docker inside the server

here you can see **TO RUN THE DOCKER RELATED WORK, WITHOUT SUDO ACCESS, ADD THE CURRENT USER TO DOCKER GROUP** if not commands will not work without permissions.

Now try to enter docker commands to check

command: **docker image ls**
output: permission denied

try with other command also to check the containers

command: docker container ls
output: permission denied

So To run Docker commands there is group called docker to that docker group add your user ubuntu

command: sudo usermod -aG <group> <user-name>
command: sudo usermod -aG docker ubuntu

then you need to logout from server

command: logout

Then after log into the same server again and now check with docker commands are working or not

Before, when i am entering this commands output shows permission denied
now check with same commands below

command: docker image ls (It will list all the images)
output: Repository   TAG   IMAGE ID   CREATED   SIZE

to check the containers
command: docker container ls   (IT WILL LIST THE Running CONTAINERS)

output: container ID IMAGE COMMAND CREATED STATUS PORTS NAMES

command: docker container ls -a   (IT WILL LIST all THE CONTAINERS which stopped also)
output: container ID IMAGE COMMAND CREATED STATUS PORTS NAMES

Now lets see How to start with containers
- To create the containers first we need to fetch the images
- Because to create a container image is must

Open Browser - hub.docker.com in this registry you can find all images
now **search** an simple image called **hello-world**
click on **Hello-world** (its a official image all official images will have this star tag)

if i want this image means you can see on right side command: **docker pull hello-world**

if i run this command, it will download the image in our server, and also you can search the image in our server also
command: docker search hello-world
<u>output: here also you can see official **ok** and stars</u>

now i want this **hello-world image** go and copy the command from docker.hub

open git bash
First check we have any image

command: docker image ls
<u>output: no images</u>

now pull the image
command: docker pull hello-world
<u>output: downloaded latest</u>

command: docker image ls (It will list all the images)
```
output: Repository    TAG    IMAGE ID      CREATED       SIZE
        hello-world   latest  d2c94e               12 months ago       13 KB
```

Now using this **hello-world image** create a **container**

command: docker container run hello-world
<u>output- read this now</u>
here you can see (<u>you can run an UBUNTU container with: docker run -it ubuntu bash</u>)

now list the container
command: docker container ls
<u>output: no container</u>

again try with **-a** list all containers
command: docker container ls -a

```
output: container ID    IMAGE    COMMAND  CREATED    STATUS       PORTS       NAMES
          c3c740      hello-world     '/hello'          2 minutes     exited(0) 2 minutes recursing-villani
(RANDOM CONTAINER NAME)
```

So this how docker is working with the images and creating containers also
Now will try with other image called ubuntu

browse: docker.hub
search- ubuntu image and open that
here you can see the tags, tags are nothing but an versions

note : from here you can pull an image, operating system (os) and also application also,

let's check in server open git bash

command: cat /etc/os-release
output: here you can see the os information

now check the disk usage
command: df -h

```
output: filesystem size used available use mounted on
          /dev/root  15G   2.3G  13G     16%   /
```

now check the processors running in your machine
command: ps -ef
output: This all are running in your ubuntu machine

browse docker.hub

now i want **version 22.04 version** if i won't give version means it will take latest version
so open git bash

command: docker image pull ubuntu:22.04
output: pull complete

command : docker image ls
output: ubuntu 22.04 image ID created size

Now by using ubuntu image i want to create a container
command: docker container run ubuntu:22.04

now check the containers

command: docker container ls
<u>output: no containers</u>

command: docker container ls -a
<u>output: container ID ubuntu:22.04 status-exited name</u>

Again containers are stopped, why ?
Here you need to know, when you are running the containers, you need to give modes.

- if you don't specify mode, it will always exit.

here you can see we created a ubuntu container and status is exited

So here what i want to conclude here means, if you want to create the containers you need to give the modes.

Modes two types
- Interactive mode or foreground mode (-it)
- detached mode or background mode (-dt)

Now try to create a container in <u>interactive mode</u>

c**ommand: docker container run -it ubuntu:22.04**
output: root (now your inside the container)

<u>open one more git bash software</u>
log into same server once again
Command: ssh -i key.pair username@ipaddress

command: docker container ls
output: now ubuntu 22.04 is running

now check using -a
command: docker container ls -a

now check what's the difference between inside the container and outside the container

Inside the container
give **command: cat /etc/os-release**
output: # showing same in both

Outside means other g new git bash page

command: cat /etc/os-release

output: showing same in both

NOTE: difference is because Docker images are all minimal applications and its not a full fledged operating system inside the container

lets verify with below commands now;

Inside the container check the process with below command
command: ps -ef

```
output: UID      PID     PPID    C        STIME           TTY    TIME   CMD
        root     1       0       0        03:40   pts/      00:00  /bin/bash
        root     10      1       0                                 ps -ef
```

outside the container use process command
command: ps -ef
output: this many are running

Note inside the containers you will not find any software's that generally find in your host

host is nothing but a outside which we operating on host

outside the container
command: python3 -v
output: python is available here in our host level   (to exit give ctrl + d)

command: vi
output: exit click on esc :q

command: ssh
output: available

command: git
output: available

command: apt
output: available

inside the container
command: python -v
output: command not found

command: vi
output: command not found

command: ssh
output: command not found

command: git
output: command not found

command: apt
output: available (Because **apt** we use for installations)

This a major difference between a VM and inside the container
- A VM will have heavy weight
- A container will have very very light weight and that's the reason why this is very fast

now lets understand about interactive mode

open **outside container**
command: docker container ls or docker ps (both are same commands)
output: ubuntu:22.04 is running

now open **inside** the **container**
now give a command exit to exit from a container
command: exit

now again open outside the container
command: docker container ls
output: its exited again

this is how it works when you're connected inside the container it will run upon, once exit from container it will disconnect and it will exit from container.

So when you work with detached mode (-dt) it will always run, they will not exit when you exit the container, so simply running continuously run in the background

command: docker container run -dt ubuntu:22.04
output: bbabe5ae674f some big Id its showing

note: we are not inside the container still we on host but we know now its works

command: docker container ls
output: bbabe5ae674f

note: now it will not exit unless you go and stop the container.
- how i created the container so i can go and stop container and start the container also

to **stop** container
command: docker container stop ID or name anything you can give here

start the container
command: docker container start ID or name

Note: Here when you want to remove the container we use rm but rm works only when you stop the container only it works
- when a container is in start and upon running rm command will not work to remove but if you want to remove running container there is a one more command we use rm -f forcely removing the container

when container is in stop you remove the container with below command
command: docker container rm ID or name

When container is running you cant remove the container with **rm** so add **rm -f** it remove forecly to running containers
command: docker container rm -f ID or name

if you don't want the containers which are not running
command: docker container ls -a
output: here we can see stopped and exited containers

i want all containers needs to be deleted
command: docker container rm <all containers with Id or names> you can give here

we always go with detached mode to run our container applications needs to run 24/7 *lets created a named container using ubuntu image in -dt mode*

command: docker container run -dt --name user-1 ubuntu:22.04
<u>output: container ID created</u>

command: docker container ls
<u>output: here you can see the name also what we have created</u>

Note: whenever you create a container without using any mode lets see what happen

command: docker container run ubuntu:22.04
command: docker container ls and also l**s -a**
<u>output: it will exited automatically</u>

lets create a container now and (exec)connect with container from outside only not entering into inside container

command: docker container run -dt --name user-1 ubuntu:22.04
command: docker container exec user-1 uname
<u>output: Linux </u>(this output is coming from inside the container)

Now let's check with command which are running on the container
command: docker container exec user-1 ps -rf
<u>output: showing processors which are running on container</u>

Now let's connect container with interactive mode without stopping after when i exit from container also
command: docker container exec -it user-1 bash (bash is shell it will the access to connect to the containers
<u>output: #ba4jdgscwn</u> (now inside the container again not on host if you want to verify)

command: ps -ef
<u>output: /bin/bash/</u>
<u>bash</u>
ps -ef

command: git
<u>output: not found</u>

Now exit from the container and let's check if the container is down or not ?
command: exit

command: docker container ls
<u>output: user-1 is running without exit</u>

Lets work on web application

i want to go a head and deploy web application and i want to access that application

how do we install our web applications

Command: docker container exec -it user-1 bash
<u>output: #</u> inside the container

command: apt install -y nginx
<u>output: E:</u> <u>unable to locate package ngin</u>x   (because its out dated)

lets update our system inside
command: apt update -y
<u>output: updated</u>

now install nginx

command: apt install -y nginx
<u>output: installed</u>

lets check its running or not

command: service nginx status
<u>output: not running</u>

now run the nginx
command: service nginx start

now again check status
command: service nginx status
output: running

now where did installed all this - Inside a container not on host

now exit from container
command: exit
<u>output: outside of container</u>

let's check is it running still
command: docker container ls
<u>output: user-1 is running</u>
now i will try to access it

command: sudo ss -nptl (here we can see is there any port 80 is running)
<u>output: 53 22 22 is running no 80 is running</u>

take your IP address and open in browser its not responding
-because you have installed nginx inside the container not on host

now let's check the properties inside the container
command: docker container inspect user-1
<u>output: what the information is showing this all properties of container</u>

here you can IPAddress: 172.17.0.2 have address but its a private ip address
- if i try to access from outside world on browser it not gonna work
- but i will go to my container and say curl 172.17.0.2 this address

command: curl 172.17.0.2
<u>output: welcome to nginx page opens inside the container</u>

But this is long approach we doing

There is One More approach in creating a container and downloading the image in a single command using detached mode

first lets check do we have nginx image
command: docker image ls
output: no nginx

lets create a container with image

command: docker container run -dt --name user-2 nginx

<u>output: nginx is not there means it will the pull the nginx image from docker hub registry and downloads</u>

command: docker image ls
<u>output: nginx available</u>

let's check the container
command: docker container ls
<u>output: user-2 with nginx image</u>

now user-2 have an nginx image without installing separately and lets verify it available or not

command: docker container inspect user-2
<u>output: take the private IPAddress 172.17.0.3</u>

command: curl ipaddress 172.17.0.3
<u>output: welcome to nginx</u>

now here also by this approach we get the same thing welcome to nginx this is how in docker things work differently

now lets delete the containers and check pages open
command: docker container rm -f user-1 user-2
<u>output: deleted</u>

now check the response
command: curl 172.17.0.3
<u>output: failed</u>

Now lets check the difference between -
create a container with same names with user-1 with ubuntu image and user-2 with nginx image

now inspect both
command: docker container inspect user-1
<u>output: 172.17.0.2 same</u>

command: docker container inspect user-2
<u>output: 172.17.0.3 same</u>

check the response
command: curl 172.17.0.2

output: failed again we need to install software's inside to update system again install nginx and start nginx this so its a long process

command:172.17.0.3
outrput: welcome to nginx its working

let's check the port 80 is running
command: sudo ss -nptl
output: no 80 port

why is it not showing port 80 because web applications run on nginx only ? yes
note: host not occupied port 80 and still nginx is running why because it's isolated

-------------------Note--------------------

Now think your Application is there inside the container, would you ask your customer to come into the server like this and verify the application? No na
- Generally Customers will see the application from the browser ? for that what we will do we will go server take the IPAddress and will give to them to access the application to use

So Here comes Docker Networking, lets see

DOCKER NETWORKING -

It enables communication and connectivity between containers and external network to interact with each other and the outside world.

while creating instance we have added some port numbers

That port number will call in different names

PORT FORWARD/ MAPPING/ PUBLISH

Port Forward: Port Forwarding in Docker allows you to expose container ports to the outside world.

here you can see
-P (capital P) = predefined range 32768 to 61000. it generates its own port when we add -P
-p (small p) = custom range 8080-9090. if i want to add my port numbers i will go with -p

So now how can i create a port mapping with existing container user-2, No its not possible with already having container user-1 we cannot add port to that.

now lets create a new container with port
command: docker container run -dt --name user-3 -P nginx
Output: container Id created

command: docker container ls
output: user-3 run on predefined port

now lets go back to the aws account copy the IPaddress

open browser- ipaddress:port example- 34.246.54.45:32768
output: welcome to nginx

now try with random port number -p
command: docker container run -dt --name user-4 -p 8080:80 nginx
output: container Id created

command: docker container ls
output: user-3 run on 8080:80

browse- ipaddress:8080
output: welcome to nginx