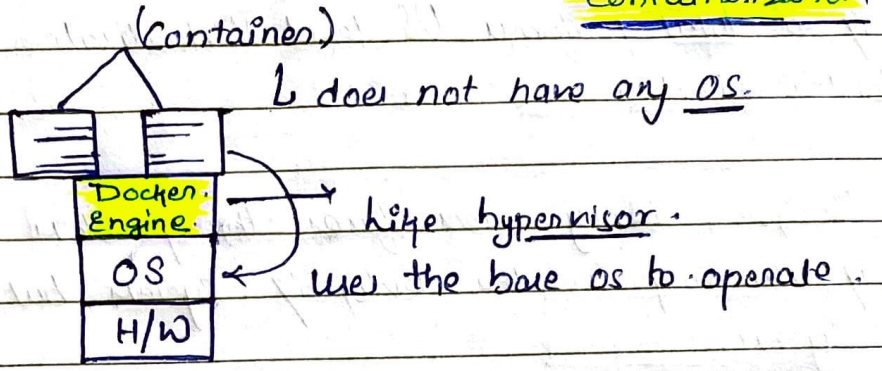
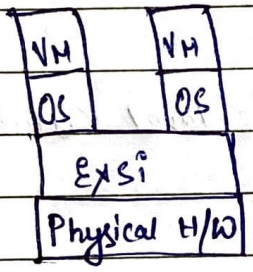
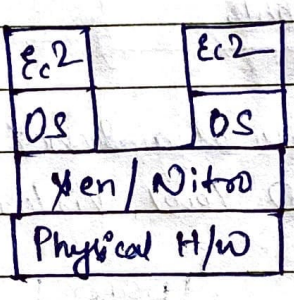
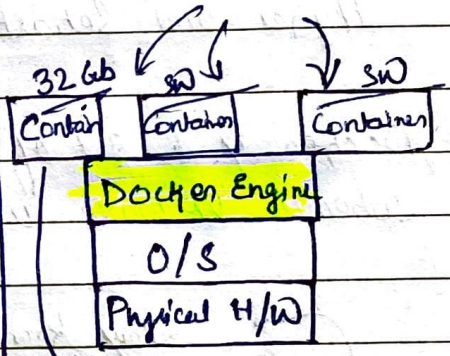
Lec - 01 What is Docker ?

+ Also a type of deployment tool.

→ Docker is a advanced Version of Virtualization or Containerization



(VM Ware)	(Aws Ec2)	(Docker hub)
		
* →	But they will hold the Resource even After use.	(128 gb RAM) Release Resource after used.

* Docker is an Open-Source Centralised platform designed to create, deploy and run applications.

* Docker uses Containers on the host OS to run applications. It allows applications to use the same Linux kernel as a system on the host computer (server), rather than creating a whole virtual O.S.

* Docker written in "GO" language.

Sawan Kr. Rauth.

Lec-02 What is Docker Architecture & Container.

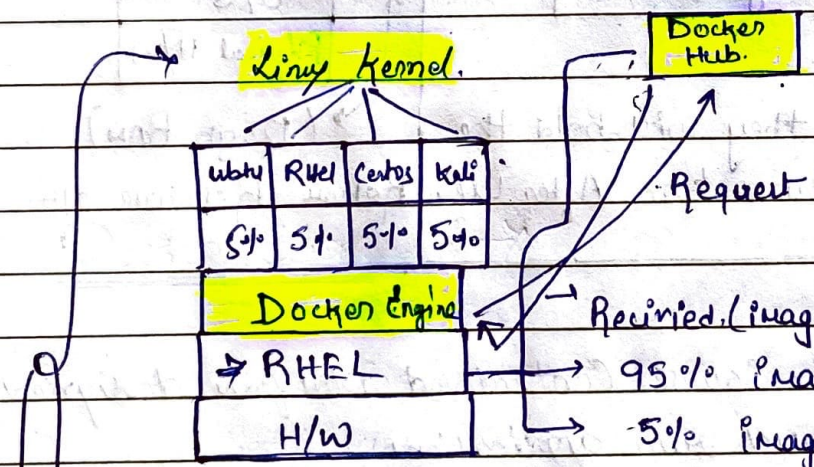
* We can Install docker on any O.S but Docker Engine runs ~~only~~ Natively on Linux distributions.

* Docker is a tool performs OS level Virtualization, also known as Containerization.

* Before Docker, Many users faces the problem that a particular Code is Running in the developer's System but not in the user's System.

* Docker was first Released in Mar 2013 & it is developed by Solomon Hykes and Sebastian Pahl.

* Docker is a set of platform as a Service that uses O.S level Virtulaisation whereas VM Ware uses H/W level Virtualisation.



→ that's Why Container is called Light Weight

→ O/S level Virtulization.

Lec 3. :- Advantages, Disadvantages & Architecture of Docker.

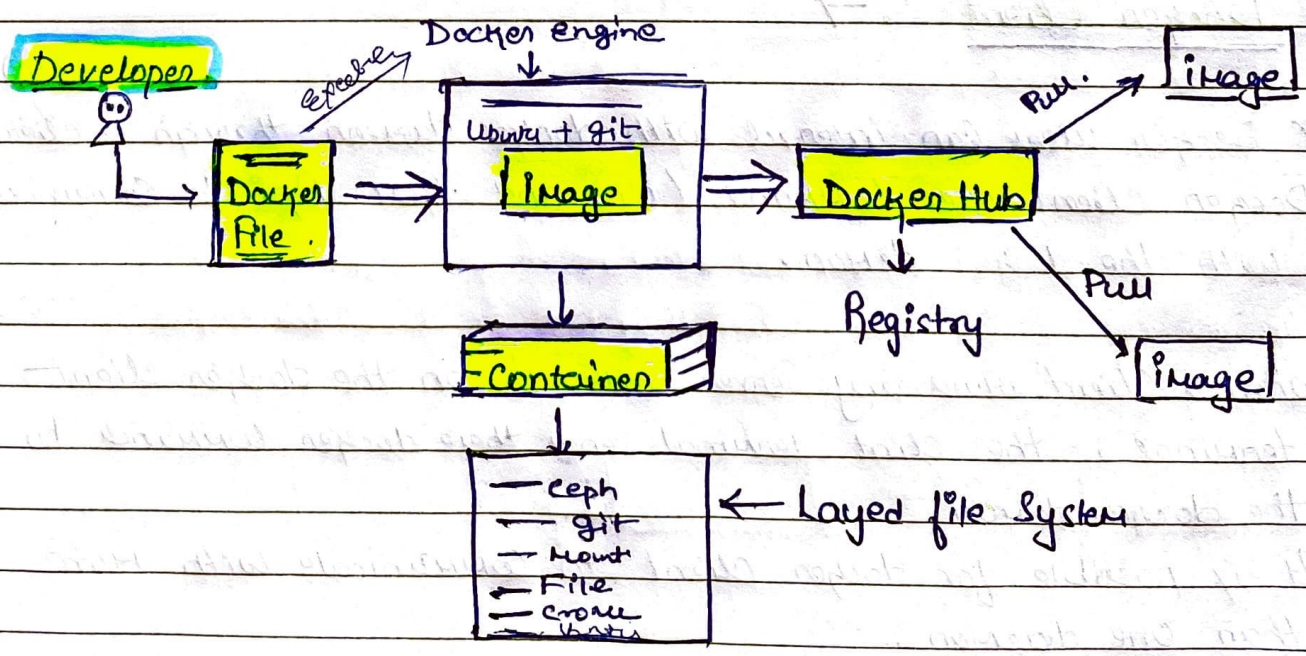
Advantages of Docker

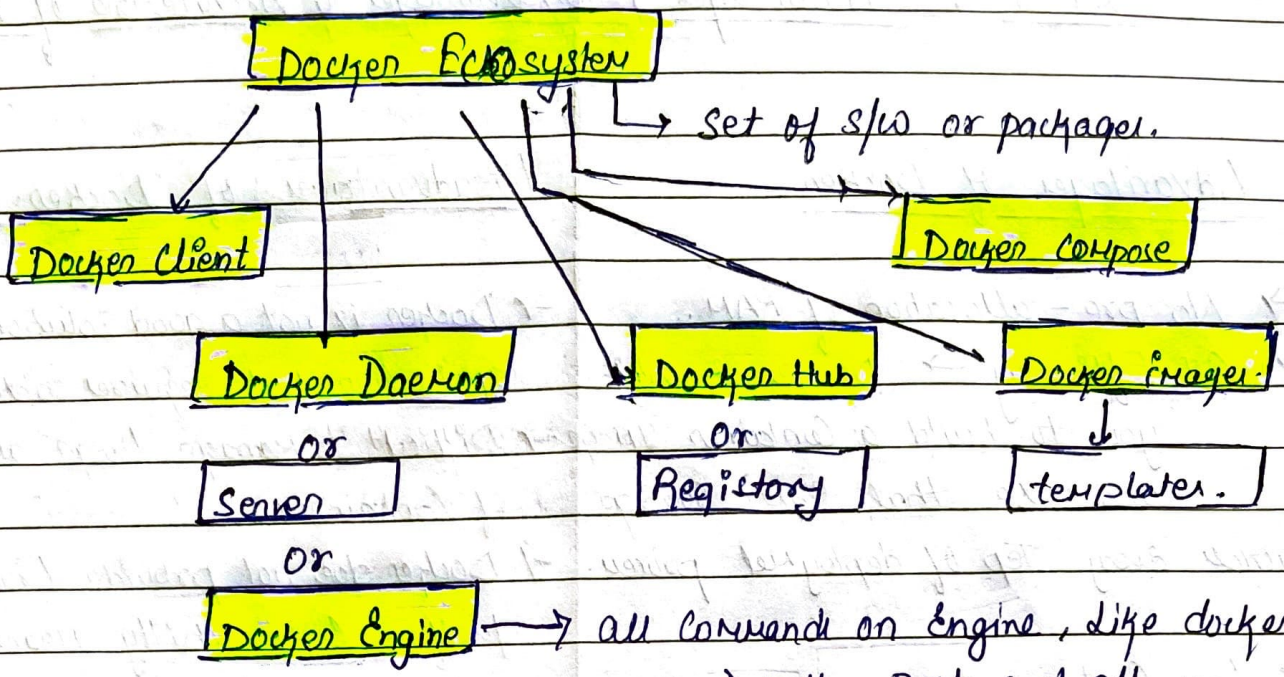
→ No pre-allocation of RAM.
→ CI Efficency → Docker Enables you to build a Container Image and use that same Image across every step of deployment process.
→ Less Cost
→ It is light in weight.
→ It can run on physical H/W or Virtual H/W or on Cloud.
→ You can re-use the Image.
→ It took very less time to Create Container.

Disadvantages of Docker:

→ Docker is not a good solution for application that requires rich GUI.
→ Difficult to Manage large amount of Containers.
→ Docker does not provide Cross-platform Compatibility means if an application is designed to run in Container on Windows, then it can't run on Linux or vice-versa.
→ NO solution for Data Recovery and Bkp
→ Docker is suitable when the development OS and testing OS are same if the OS is different we should use VM.

Architecture of Docker





→ all Commands on Engine, like docker → pull, Push and all.

Components of Docker.

*) **Docker Daemon / Docker Engine** :→

→ Docker daemon run on the Host O.S
→ It is responsible for running Containers to Manage docker Services.
→ Docker Daemon can Communicate with other daemons.

*) **Docker Client** :→

(cli)

→ Docker users can interact with docker daemon through a client
→ Docker client uses Commands (cli) and Rest Api to Communicate with the docker demon.

→ When a Client runs any Server Commands on the docker client terminal, the client terminal sends these docker Commands to the docker demon.
→ It is possible for docker client to Communicate with more than one daemon.

Docker Host :-

→ Docker Host is used to provide an environment to execute and run applications. It contains the docker daemon, Images, Containers, Network and Storages.

Docker Hub / Registry

→ Docker registry manages and stores the docker images.

There are two types of registry in the docker.

i) **Public Registry** :- public registry is also called as docker hub

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

Docker Images

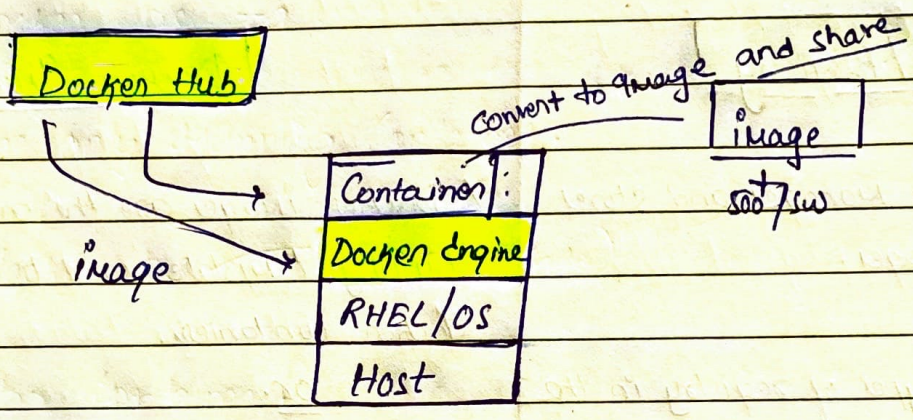
→ Docker image are the read only binary templates used to create docker containers.

or.

→ Single file with all dependencies and configuration required to run a Container / Programme.

*) Ways to Create or access an Images.

=> Take images from docker hub.
=> Create image from docker file.
=> Create image from existing docker container.



*) Docker Container.

=> Container hold the entire packages that is needed to run the application.

=> In other words, we can say that the image is a template and the container is a copy of that template.

=> Container is like a Virtual Machine (Not actual like)
=> Images become container when they run on docker Engine.

Image → runs → Container

(Modification Not possible) (Modification possible)

Lec-4 → Basic Commands in Docker.

⇒ To see all images present in your local Machine.

$#> docker images.

⇒ To find out images in docker hub.

$#> docker search Jenkins
(image Name)

⇒ To download image from dockerhub to local Machine.

$#> docker pull Jenkins
(image name)

⇒ To give name to container.

$#> docker run it --name Sawan ubuntu /bin/bash.
(interactive mode) (image name)

⇒ To check services is start or not.

$#> Service docker status.

⇒ To start Container

$#> docker start sawan
(Container name)

⇒ To go inside a Container.

$#> docker attach Sawan
↳ (Container Name)

⇒ To see all Containes state.

$#> docker ps -a

⇒ To see only running Containes.

$#> docker ps — (previous state)

⇒ To stop Container.

$#> docker stop Sawan → (Container name)

⇒ To delete Container.

$#> docker rm ~~bhupi~~ Sawan → (Container name)

⇒ Stop all running Container.

$#> docker stop $(docker ps -a -q)

⇒ Delete all stopped Container.

$#> docker rm $(docker ps -a -q)

⇒ Delete all image.

$#> docker rmi -F $(docker images -q)

Lec -05 Dockerfile Components & diff Command.

⇒ Login to Aws account and start your Ec2 Instance. Access it from putty.

⇒ Now we have to create Container from our own image, therefore Create one Container first.

```
$#} docker run -it --name Sawan Ubuntu /bin/bash.
                    $(Container name)  (image name)
```

⇒ Now Create one file inside this tmp directory.

```
$#} cd /tmp
$#} touch filename.
```

⇒ Now if we want to see the difference between the base image and Changes on it then.

```
$#} docker diff Sawan
                ↳ (Container name)
```

```
O/p → C /root
      A /root/.bash_history
      C /tmp
```

Dockerfile :- Dockerfile is basically a text file it contains some set of Instruction.

→ Automation of Docker Image Creation.

Docker Components.

FROM :- for base image this command must be on top of the docker file.

RUN :- To execute Commands, it will create a layer in Image.

MAINTAINER :- Author / Owner / Description.

COPY :- Copy files from local system (docker VM) we need to provide Source, destination.
(we can't download file from internet and any remote repo)

ADD :- Similar to Copy but, it provides a feature to download files from internet, also we extract file at docker image side.

EXPOSE :- To expose ports such as port 8080 for tomcat; port 80 nginx etc.

CMD :- Execute command but during Container Creation.

ENTERY POINT :- Similar to CMD, but has higer priority over CMD, first CMD will be executed by ENTERYPOINT only.

ENV :- Enviroment Variables.

Dockerfile.

1. Create a file named Dockerfile (D should be in capital)
2. Add instructions in Dockerfile.
3. Build Dockerfile to create Image.
4. Run Image to create Container.

vi Dockerfile.

```
FROM ubuntu
RUN 'echo " Sawan Rauth " > /tmp/testfile.
```

5. To run the Dockerfile :—

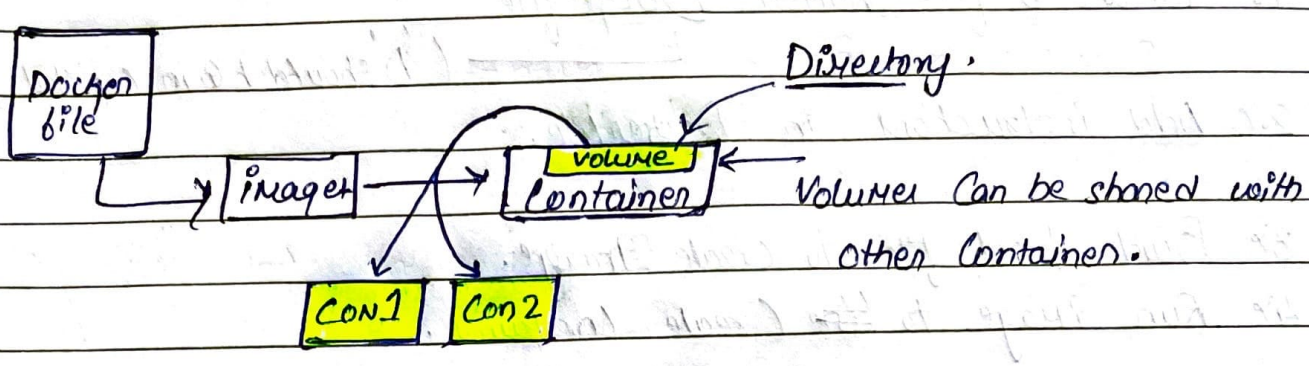
```
#$ docker build -t myimage .
```
→ (Dot to execute in current dir)
↳ (image name which we want to give)

6. docker ps -a
7. docker images.

8. Now, Create Container from the above Image.

```
#$ docker run -it --name mycontainer. myimage /bin/bash
```
(container name) (image name)

Lec - 06 :- Docker Volume & How to Share it.



⇒ Volume is simply a directory inside our container.

⇒ Firstly, we have to declare this directory as a Volume and then share Volume.

⇒ Even if we stop Container, still we can access Volume.

⇒ Volume will be created in One Container.

⇒ You can declare a directory as a Volume only while Creating Container.

⇒ you can't Create Volume from existing Container.

⇒ you can share One Volume across any number's of Container's.

⇒ Volume will not be included when you update an image.

⇒ You can Mapped ~~Container~~ Volume in two ways :—

1.⇒ Container ⟷ Container.

2.⇒ Host ⟷ Container.

Lec-07 Creating Volume from Dockerfile.

1.) Create a Dockerfile and write.

```
FROM ubuntu
VOLUME ["/myxyz"]
```

2.) Then Create image from this Dockerfile.

$ ⇒ docker build -t myimage .
(image name)

3.) Now Create a Container from this image & Run.

$# ⇒ docker run -it --name container1 myimage /bin/bash.
(container Name) (image name)

Now do "ls" under root diretory and u can see the myxyz volume name.

4.) Now Share Volume with another Container Container1 - Con2

$# ⇒ docker run -it --name container2 --privileged=true
--volume from container1 ubuntu /bin/bash.

Now after Creating Container2, myxyz is visible whatever you do in One Volume, Can see from other Voulume.

$# ⇒ touch /myxyz/testfile $# ⇒ docker start container1
$# ⇒ docker attach container ⇒ then go to myxyz all files will be there.

5.) Now, try Create Volume by using Command.

$#) docker run -it --name Container3 -v /volume2 ubuntu /bin/bash

Do "ls" → cd /volume2

Now Create One file Cont3file and exit.

6.) Now Create One More Container, and share Volume2.

$#) docker run -it --name Container4 --privileged=true --volumes-from Container3 ubuntu /bin/bash

Now you are inside the Container, do "ls" you can see Volume 2.

Then Create One file inside this Volume and then check in Container 3, you can see that file.

Lec -08 Volume Sharing (HOST – CONTAINER)

#⇒ go to /home/ec2-user

#⇒ docker run -it --name hostcontainer -v /home/ec2-user:/filexyz --privileged=true ubuntu /bin/bash

(container name) (HOST share folder path) (container share folder path) (image name) (bash shell to login)

$#⇒ docker run -it --name hostcontainer -v /home/ec2-user:/filexyz --privileged=true ubuntu /bin/bash

(This is the full command to share volume from Host to Container)

#⇒ Now cd /filexyz
Do "ls", now we can see all the host file under the container it's synced.

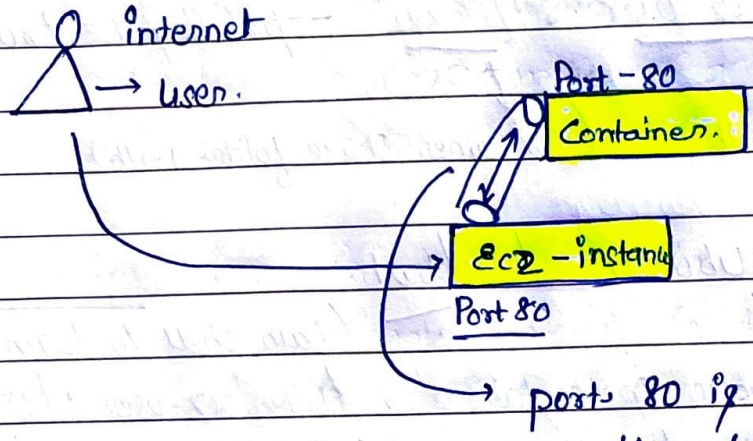
#⇒ Now Create touch xyzfile

#⇒ Exit from the container.

#⇒ Now go to /home/ec2-user
and "ls" you can see the xyzfile there which is created under the container because its Mapped From HOST ↔ CONTAINER.

Lec-09 Docker Port Expose.

0 - 65535 → logical ports.
Under System.
(works under transport layer)



→ port 80 is mapped Host Machine to the Container port 80.

*) generally we can change the port no. under the host or instance but can't change under the Container.

→ Login into Aws account Create One Linux instance.

→ Now go to putty → Login as Ec2-user.
→ sudo -i
→ yum update -y.
→ yum install docker -y.
→ service docker start.

(host 80 port is mapped with container 80 port)
(host) (container)

→ docker run -td --name techServer -p 80:80 ubuntu
-td → daemon
techServer → Container Name
-p (Port)

=> docker ps.

=> docker port techServer
↳ (container)

=> docker exec -it techServer. /bin/bash.
↳ (container)

=> apt-get update.

=> apt-get install apache2.

=> cd /var/www/html

=> Echo "Sawan New website under Docker Container" > index.html

=> Service apache2 start.

=> docker run -td --name my jenkins -p 8080:8080 Jenkins.

*) Difference between docker attach and docker exec ?

Docker Exec :- Docker exec Creates a new process in the Container Environment while docker Attach.
- Just Connect the Standard Input / output of the main process in side the Container to Corresponding Standard input / output Error of Current terminal.

Docker Exec :- is Specifically for running new things in a already started Container, be it a shell or Same other process.

*) What is the difference between **Expose** and **publish** CMD in docker ?

Basically you have three Options :—

1.) Neither Specify Expose nor -p

2.) Only Specify Expose.

3.) Specify Expose and -p. (→ written in Next page.)

i) if you specify neither Expose or -p, the Service in the Container will only by accible from inside the Containes itself.

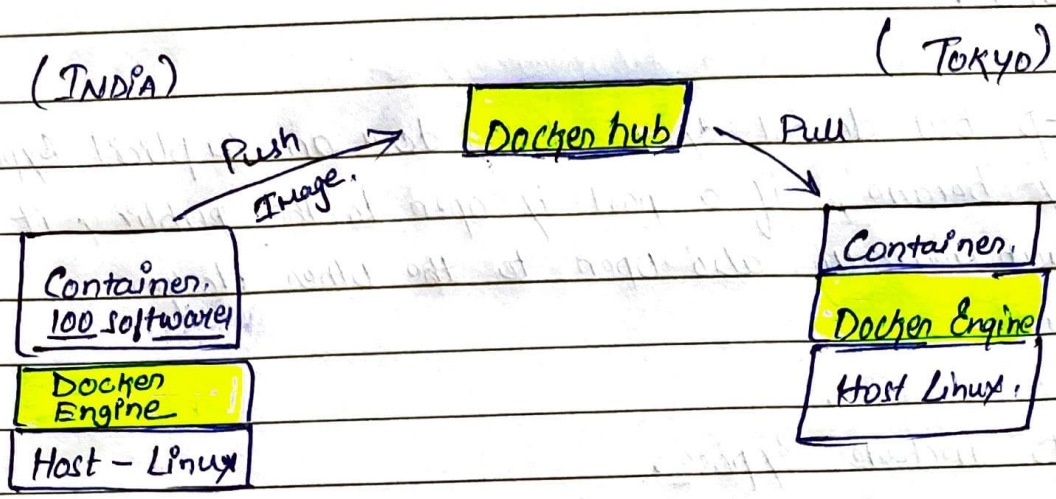
ii) if you Expose a port, the Service in the Container is not accisable from outside docker, but from inside other docker Containers, so thish is good for inter-Container Communication.

iii) if you Expose and -p a port, the service in the Container is accessiable from any where, even Outside the docker Container.

if you do -p but do not expose docker does an implicit Expose this is because, if a port is open to the public, it is automatically also open to the other docker container.

Hence -p includes Expose.

Lec-10 :- How to push docker img in dockerhub.



⇒ go to Aws account → Select Amazon Linux.

⇒ Now go to putty → Login as → ec2-user.

⇒ sudo su

⇒ yum install docker -y.

⇒ Servie docker start.

⇒ docker run -it ubuntu bin/bash.

Now create some files inside Container Now Create image of this Container.

⇒ docker Commit Container1 Image1

Now Create account in hub.docker.com.

Now go to Ec2 instance.

⇒ docker login
(Enter your username and password.)

Now give tag to your image.

⇒ docker tag image1: dockerid / new image.
(your docker userid) ↳ (Provide any name it will show in docker hub)

⇒ docker push dockerid / newimage.
↳ (your docker userid)

Now you can see this image in docker hub account.

Now Create one instance in tokyo region and pull image from hub.

⇒ docker pull dockerid / newimage.

⇒ docker run -it --name mycon dockerid / New image
/bin/bash.