Page No.
Date

# "Docker"

- Docker is an opensource centralised platform designed to create, deploy and run application.

- Docker uses container on the host OS to run applicati
it allows applications to use the some Linux Kernal as a system on the host computer, rather than creating a whole virtual O.S.

- we can install docker on O.S but docker engine runs notively on Linux distribution

- Docker written in 'go' Language

- Docker is a tool that performs O.S. Level virtualisation also knows 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 release in March 2013.
It is developed by Solomon Hykes and Sebastian Pahl

- Docker is a set of platform as a service that uses OS Level virtualisation whenever vmware use Hardware Level virtualization.

Akash Kale

Page No.
Date

## "Advantages of Docker"

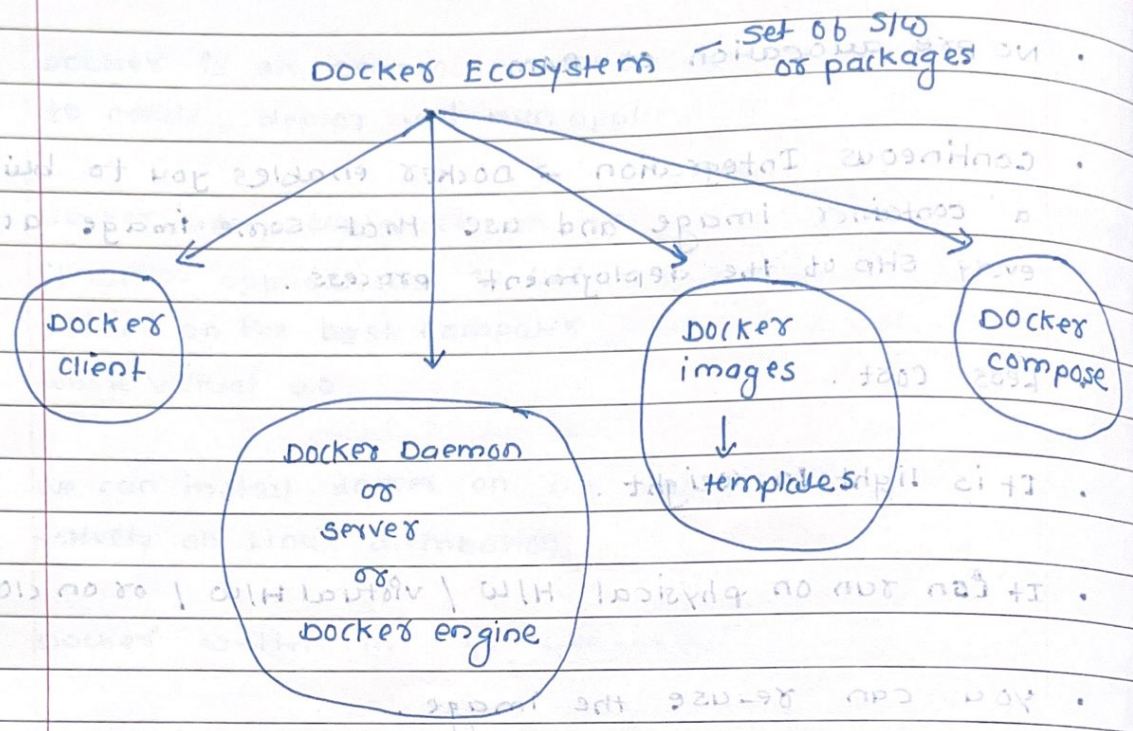
- No pre allocation of RAM
- Contineous Integration → Docker enables you to build a container image and use that same image across every step of the deployment process.
- Less cost.
- It is light in weight.
- It can run on physical H/W / virtual H/W / or on cloud.
- You can re-use the image.
- It took 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 container.
- Docker does not provide cross platform compatibility means if an application is designed to run in a docker container on windows, then it can't run on linux or vice-versa.
- Docker is suitable when the development O.S and testing O.S are same if the O.S is different, we should use V.M.
- No solution for Data recovery and Backup.

Page No.
Date

## "Docker Ecosystem"




## "Components of Docker"

① Docker Daemon

- Docker daemon runs on the host O.S.
- It is responsible for running containers to manages docker service
- Docker Daemon can communicate with other daemon.

② Docker client

- Docker users can interact with docker daemon through to client (CLI)
- Docker client uses command CLI and Rest API to communicate with the docker daemon.

- When a client runs only server command on the docker client terminal, the client terminal sends these docker.

commonds to the docker daemon
It is possible for docker client to communicate with more than one daemon

③ Docker Host

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

④ Docker Hub/Registry

Docker registry manages and stores the docker images.

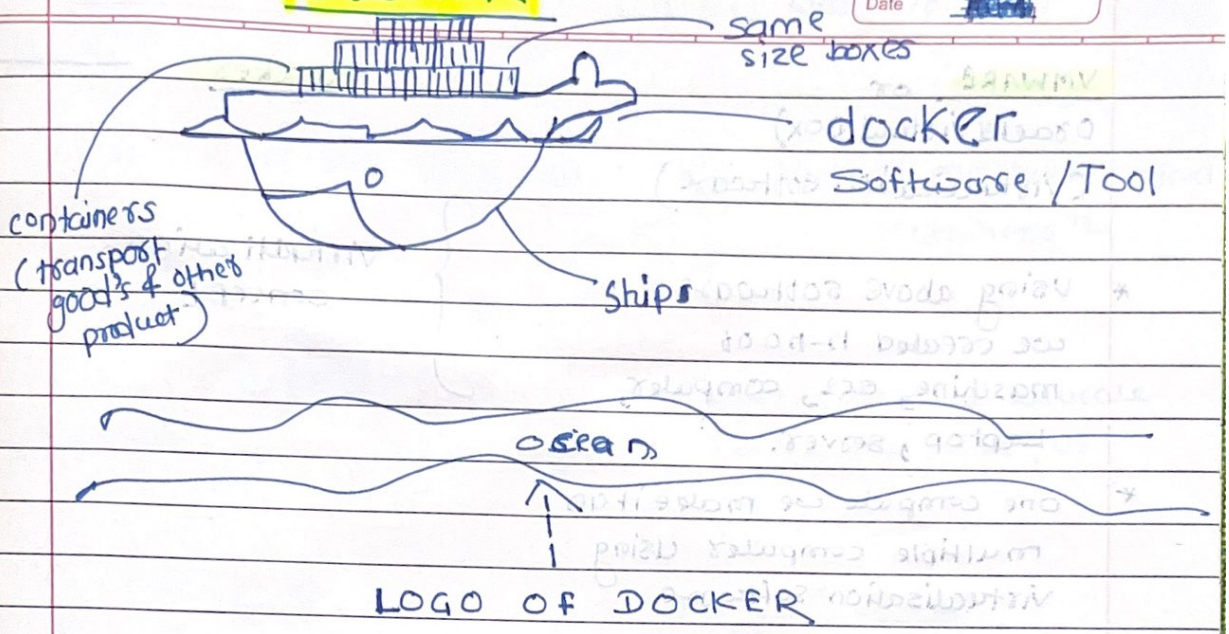
These are two types of registries in the docker.
① Public Registry
: Public registry is also called as docker hub.
② private Registry
: It is used to share images within the enterprise.

⑤ Docker images

Docker images are the read only templates used to create docker containers.
or
Singlefile with all dependencies and configuration required to run a program.

Akash Kale

# "DOCKER"




**Container** : * mean's one computer (normal - ec2 instance)
* using docker tool we create n-number of containers (machine).

IMP :
- Docker is a OS - Level virtualisation.
- It follows union file system.
- Layered Architecture.

core operating system
↓
Windows ↔ Unix
- Windows → Not free → not open source code
- Unix → free & open source code
  - SUN - solaris
  - IBM AIX
  - Linux
  - HP - UX
  - MacOS
  - Centos, RedhadRL, fedora, Ubuntu

  (IBM AIX, Linux, HP-UX, MacOS: using S.Code of Unix created this OS.)

- Container :- A Container like a virtual machine.
- Docker :- Docker is a tool to create those virtual machine.
- Took from shiping containers.
- Docker is a tool designed to make containers in which we can deploy any type of application easily.

Page No.
Date

**VMWARE** or
Oracle(Virtual Box)
(Virtualisation software)

**Docker**

* Using above software we created n-no of maschine, ec2, computer, Laptop, server.

} virtualization concept

* one compute we make it on multiple computer using virtualisation software.

2GB ram for all vm machine

200 GB HD for 4 vm machine & other 100 GB HD for 2 vm

| VM1 | VM2 | VM3 | VM4 | VM5 | VM6 |
|---|---|---|---|---|---|
| S/W | S/W | S/W | S/W | S/W | S/W |
| OS | OS | OS | OS | OS | OS |

windows or Linux

Virtualization S/W — one computer as a multiple computer

Operating system — windows or Linux

base OS

12GB Hardware 1TB

computer

2GB ram | 200 GB HD or 100GB HD

Hypervisor

Host operating system

* not possible to create extra computer because Ram, HD space are complete.

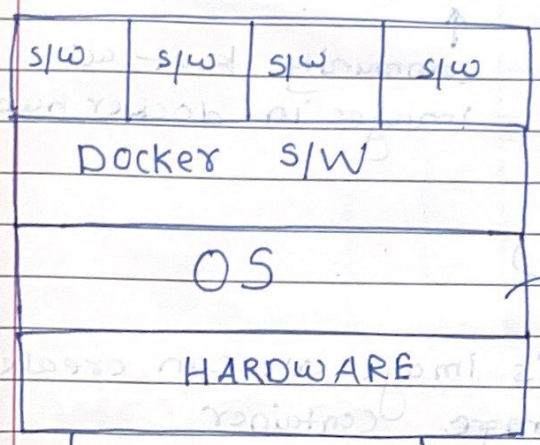
* Depends on our Ram capacity and HD capacity we are create this no of computer.

"Limited vm create"

* Docker is a tool that performs OS level virtulization, also knows as "containerization".
It was first released in 2013 and was deploy by Docker, Inc.

* Docker is tool used to create virtual machines called "containers".

containers or s/stem



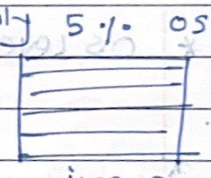

* you no need to allocate Ram, Hard disk, OS.

only Linux ← any favour

* you can create any no of computer that is the flexiblity of docker.

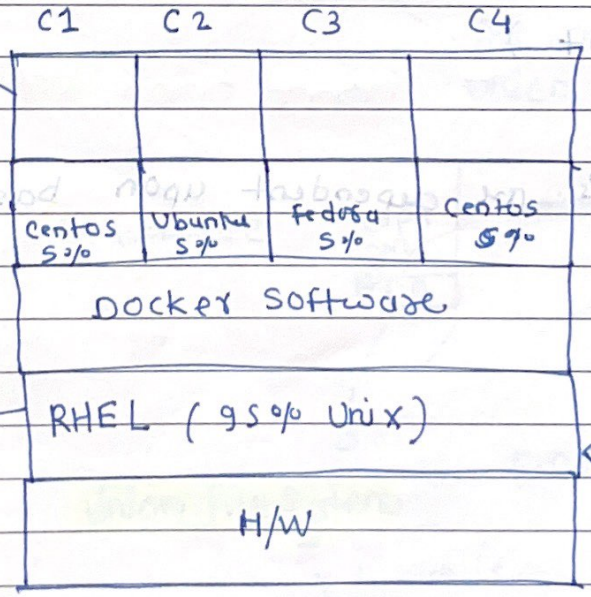
those images are in docker hub.

* concept is called OS level virtulization




each an every image is an OS.

you no need to install OS used image from docker hub.




5% of OS in the form of image

95% is this here

* It will not used any RAM & Hard disk for creating image.

Page No.
Date

* (hub.docker.com)

Docker Hub

Online Registry / online Repository

↑ So many images are available

create by docker community

↑ They create images

↑ community kept all images in docker hub.

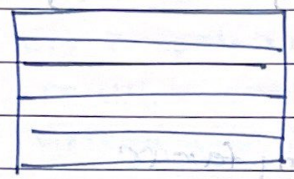


image
OS (5%)

* By using this image we can create ~~image~~ container

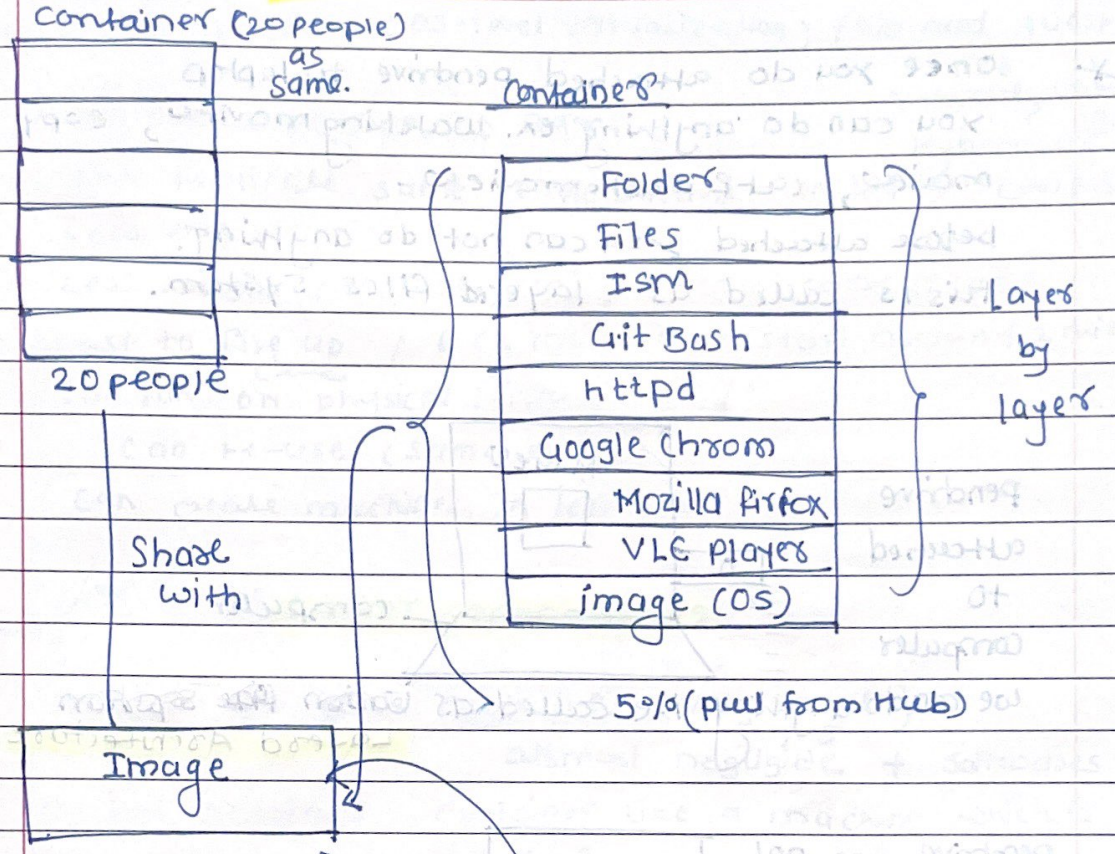
* Docker container occupy only 60- 70 MB space.

* OS Level virtulization - used 95% OS our base maschine & 5% download from dockerhub

* (Deployment)

* container's are dependent upon base maschine.

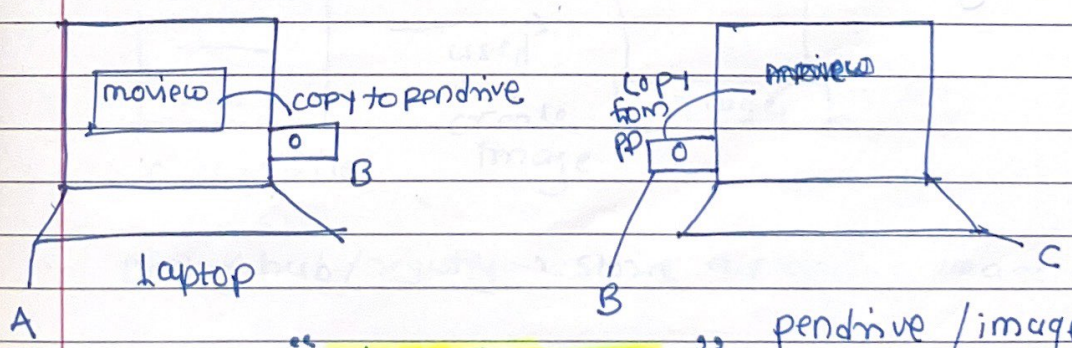
# "Layered Architecture"




VLC+mf+GC+httpd+Gitbash + Ism+Files+Folder

(OS + Slw + Files + Folder)
5 %

This image called as a Union File System.




# "Union File System"

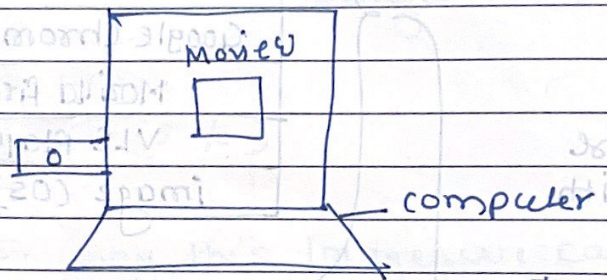
pendrive / image

* which is available in the form of image

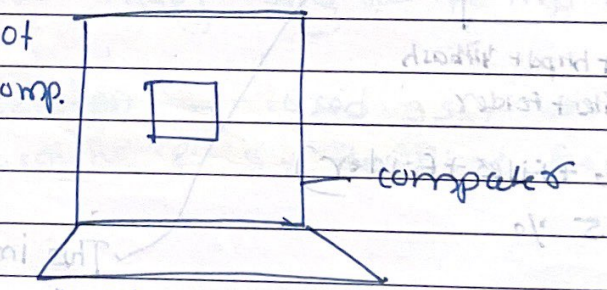
you can not do anything

Page No.
Date / /

* once you do attached pendrive to laptop you can do anything ex. watching movies, copy movies, cut & paste movies.
before attached you can not do anything.
this is called as layered files system.




pendrive attached to computer
we can do anything i.e called as ~~union file system~~ Layerd Architecture




pendrive can not attached to comp.
we can not do anything i.e called as ~~layerd system~~. Union file system

Page No.
Date

## "Way to create an Images"

1. Take image from docker hub
2. create image from docker file
3. create image from existing docker containers.

## "Docker container"

containers hold the entire packages that is needed to run the application

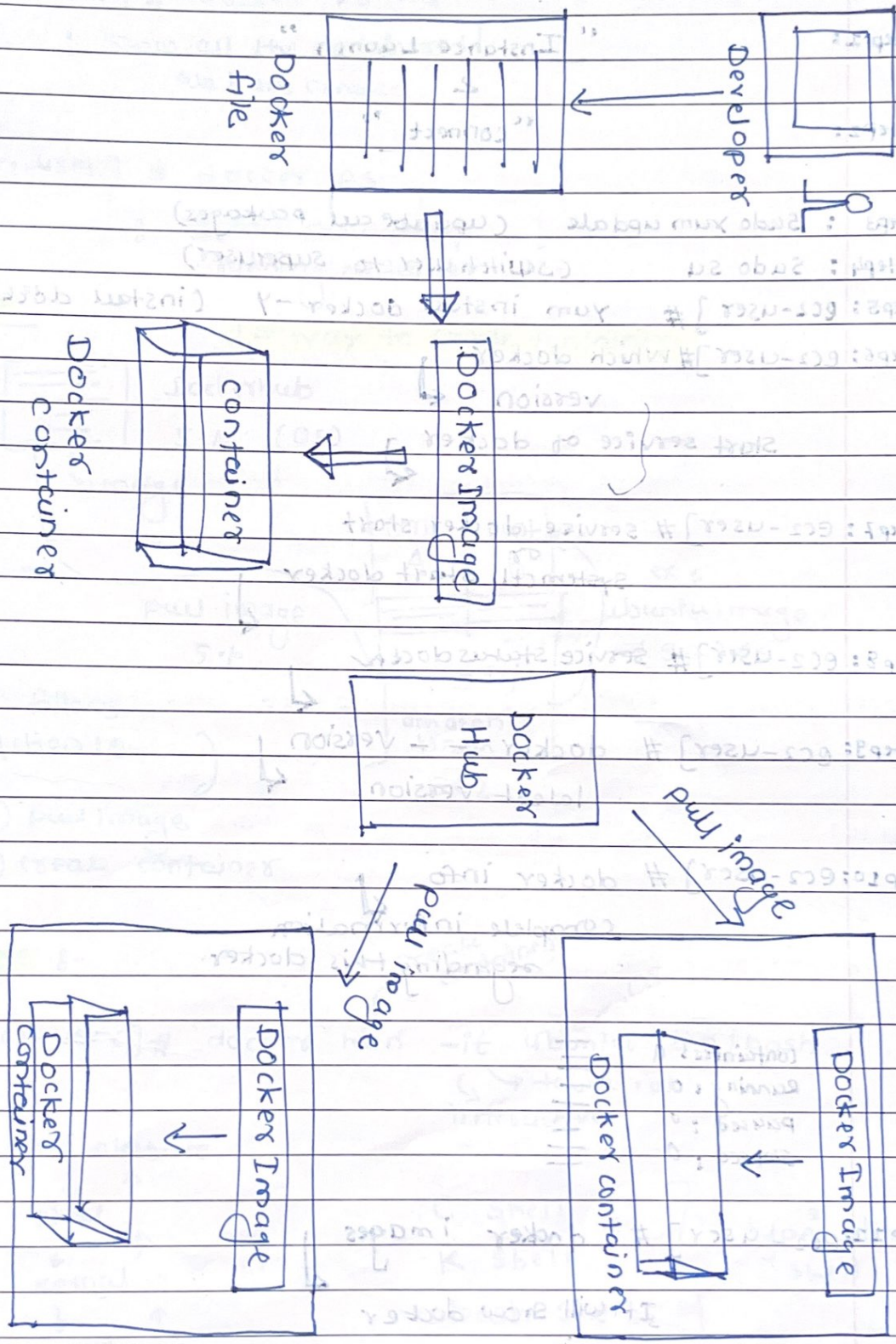
or

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

Images becomes container when they runs on docker engine

# "Docker work flow"



## "Basic commands in docker"

- To see all images present in your local maschine.

  [ ] # docker images

- To find out images in docker hub

  [ ] # docker search (image name)

- To download image from dockerhub to local maschine

  [ ] # docker pull (image name)

- To give name to container

  docker run -it --name (container name) (image name) /bin/bash

  interactive terminal

- To start container

  docker start (container name)

- To go inside container

  docker attach (container name)

- To see all containers

  docker ps -a

Page No.
Date

- To see only running containers

  docker ps ← process status

- To stop container

  docker stop (container name)

- To delete container

  docker rm (container name)

Akash Kale

**practicle :**

Step 1: "Instance Launch"
&
Step2: "connect"

Step3 : Sudo yum update (update all packages)
Step4 : Sudo su (switch user to superuser)
Step5 : ec2-user ]# yum install docker -Y (install docker)
Step6 : ec2-user ]# which docker
version
Start service of docker

Step7 : ec2-user ]# service docker start
or
systemctl start docker

Step8 : ec2-user ]# service status docker

Step9 : ec2-user ]# docker - - version
latest version.

Step10 : ec2-user ] # docker info
complete information
regarding this docker

containers : 0
Running : 0
Paused : 0
Stopped : 0

Step 11 : ec2 - user ] # docker images
It will show docker
images.

Step 12: ec2-user] # docker ps -a

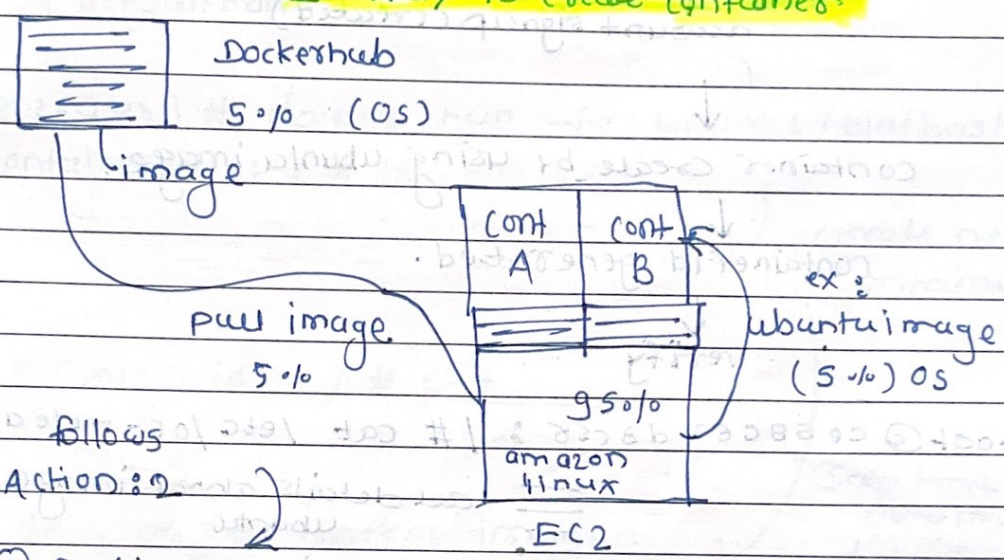
: Show all the containers we have created.

Step 13: ec2-user] # docker ps

: It will show only running container

"concept"

"1st way to create container"




① pull image
② create container

practicle :-

Step 14: ec2-user]# docker run -it ubuntu /bin/bash

create container
OS
terminal
interactive

* Linux Architecture

user interface
↓ ↑
shell
↓ ↑
kernal
↓ ↑
hardware

C shell
K shell
bash shell

types of shell

Page No.
Date

run command
↳ locally image not found because not download os (ubuntu image) or not pull any image.

↓

pull request complete.

↓

open docker hub account
(hub.docker.com)

↓

account signup (created)

↓

container create by using ubuntu image.

↓

container id generated.

↓

verify

Step 15: root@ce59c62d3c56 : /# cat /etc/os-release

all details about image ubuntu

Step 16: root@ce59c62d3c56 : / # exit
exit

Step 17: ec2-user ] # cat /etc/os-release

amazon linux (ec2 instance)

Verify

Step 18: ec2-user ] # docker ps

only running container show.

exit
kelyamale
container show
nomar
ollte

Page No.
Date

Step 19: ec2-user] # docker image

| Repository | TAG | IMAGEID | CREATED | SIZE |
|---|---|---|---|---|
| ubuntu | latest | ------ | 7 days ago | 72.8MB |

Step 20: ec2-user ] # docker ps - a

It will show all the containers.

"name" ← follow algorithm.
(defaultname)

step 21: ec2-user ] # docker run -it ubuntu /bin/bash
container id : / # ls

create new container

step 22: container id : / # exit
exit

Seen here new image details

step 23: ec2-user] # docker image

latest container
details of container
oldest container

* for searching image by using cmd

Step 24: ec2-user ] # docker search ubuntu

100 of images you can search

already available this software in amazon linux

Page No.
Date

Step 25:
* search image ex: chef *
In docker hub you can search on search tab.

ec2-user ] # docker run -it chef/chefdk /bin/bash

pulling complete

successfully downloaded

& create container.

Step 26: container id : / # cat / etc / os-release

chef software by defaulut available in this container.

Step 27: container id : / # which chef

Step 28: container id : / # chef -v

version details & dependancy software.

Step 29: container id : / # exit

exit ← exit from container

verify

Step 30: ec2-user ] # docker images

Step 31: ec2-user ] # docker ps-a

container details.

Page No.
Date

* Requirnment to download Jenkins *

Step32: [ec2-user] # docker pull jenkins/jenkins

Pull complete

Download image for jenkins/jenkins: latest

Step33: [ec2-user] # docker images

ubunty
chef / chefok
Jenkins / Jenkins
} 3 containers.

Step34: [ec2-user] # docker ps -a

name by defaut
name by defaut
name by defaut.
} It will show all the container.

Step35: If rename this name then use command

[ec2-user] # docker rename angry_mestorf devops cont
(old name) (new name)

verify

Step36: [ec2-user] # docker ps -a

"name should be change"

* ec2-user] # docker run -it --name saicontainer

Step37: ubuntu /bin/bash

direct jump to respeted container. →

Step38:
Container id:
~~ec2-user~~ # exit
exit

Step39:
* ec2-user ] # docker ps -a

Stopped container

* Requirnment to start container

Step 40: ec2-user] # docker start saicontainer
saicontainer

Step41: ec2-user ] # docker ps → only running container show

* Go instead of this container

Step42: ec2-user] # docker attach (container id or name).
exit

* Stop container

Step43: ec2-user] docker stop (container id or name).

Step44: ec2-user ] docker ps
stop not show this container

Page No.
Date

* Remove container or delete of container.

Step45:
ec2-user] # docker rm saicontainer
saicontainer
verify

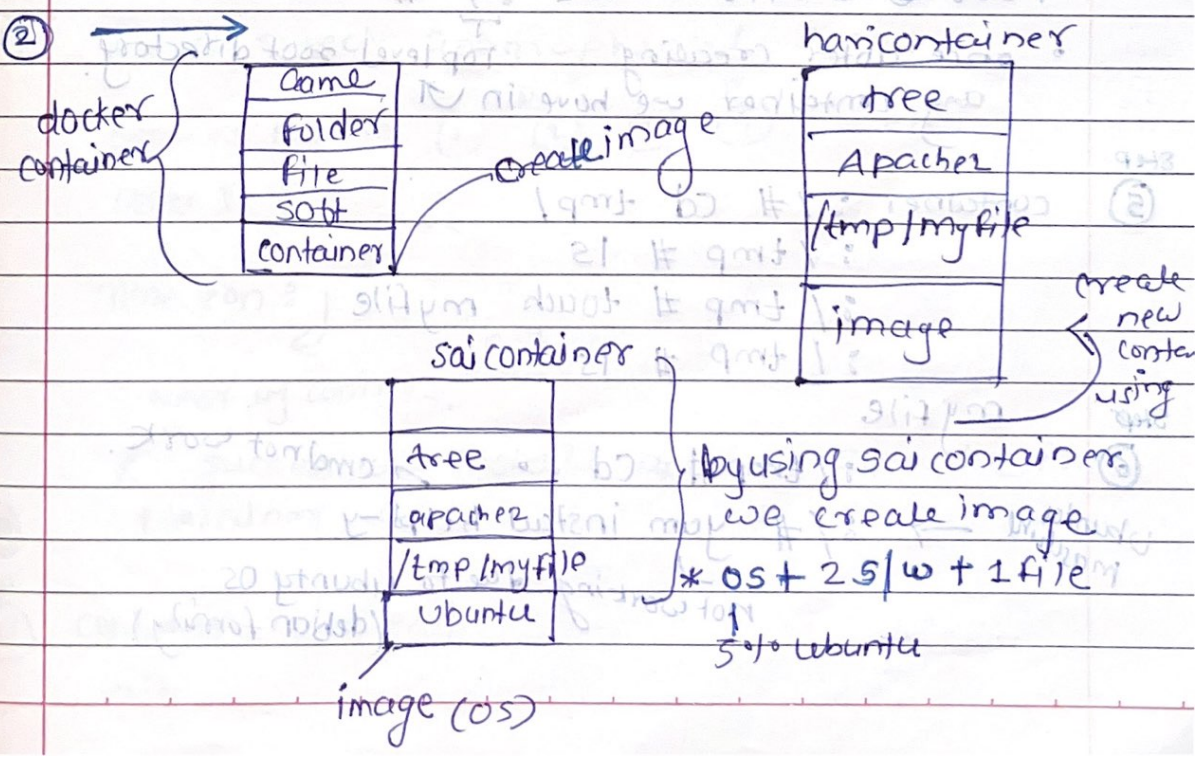
Step46: ec2-user] # docker ps -a
— notshowing this container.

- Stop — start
  — attached

- stop - rm

Class 30

Second way : container by using docker image
: Launch Instance, Amazon EC2 Instance
(1) Take a docker image from docker hub - way.
(2)



Page No.
Date

**practicle :-**

Step ① ec2-user] # docker run -it --name sai container (create) ubuntu /bin/bash

by using ↗

not ok (error)

* Docker service not start so error is occured.
* first start docker service then run above command.

Step ② ec2-user] # service docker start

Step ③ ec2-user] # ckkconfig docker on

Step ④ → execute this command & successfully create container & by default top level root directory

root @ 01aaf6f1fab2 :/ #

top level root directory.

• once after creating any container we have in ↗

Step ⑤ container :/ # cd tmp/

:/tmp # ls

:/tmp # touch myfile

:/tmp # ls

myfile

Step ⑥ :/tmp # cd .. → cmd not work.

ubuntu machine → :/ # yum install httpd -y

not working due to ubunty OS. (debian family)

* debian family - apt-get install s/w name -y
* Redhad family - yum install s/w name -y

Page No.
Date

Step (7) container : # apt-get install apache2 -y

Done

• OS not update so not successfully install software

Step (8) container : / # apt-get install tree -y

Done

• OS not update so not success installation software

Step (9) container : / # apt-get update -y

for update

enter

successfully update

Step (10) container : / # apt-get install tree -y

successfully install

Step (11) container : / # apt-get install apache2 -y

successfully install.

Step (12) please select Geographical Arear

contries Name (1) (2) (3) (4) - etc.

select :

Time zon :

enter

near by contries.

* successfully install apache2 software in container.

Step (13) container : / # exit

exit

Step (14) * create one image by using sai container

ec2-user] # docker commit sai container saiimage

use commit command for creating container

---

ec2-user] # docker image

Details of this image

Step (15) * create container by using image (newly created).

ec2-user] # docker run -it --name hamcontainer saiimage /bin/bash

I have created

new create container

Step (16) which tree

container:/# which tree

Step (17) container:/# which apache2

Step (18) container:/# ls

tmp

Step (19) container:/# ls tmp/

my file

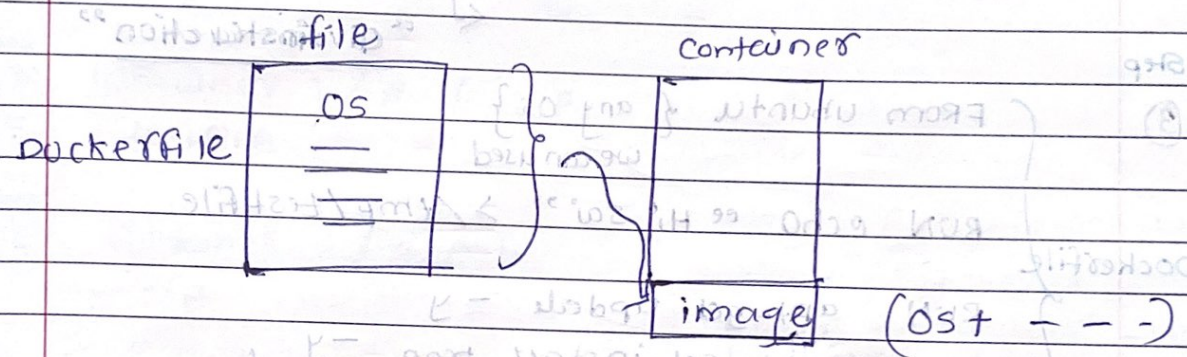
container:/# cd tmp/

container:/tmp # ls

myfile

Container : /tmp# cd ..
container : / #
container : / # exit
exit .

## Third way "Dockerfile"




- Dockerfile :
  - A text file with instructions to build an image
  - Automation of Docker Image creation
    - FROM
    - RUN
    - CMD
- Step 1 : create a file named Dockerfile.
- Step 2 : Add instruction in Dockerfile.
- Step 3 : Build dockerfile to create image
- Step 4 : Run image to create container.

# Dockerfile components & Diff commands

## Dockerfile

Dockerfile is a 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 dockerfile.
- 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 feture to downloads files from internet, also we extract file at docker image side.
- EXPOSE → To expose port such as port 8080 for tomcat, port 80 for nginx etc.
- WORKDIR → To set working directory for a contain

Page No.
Date

- CMD → Execute commands but during Container creation.
- ENTRYPOINT → Similar to CMD, but has higher priority over CMD, first commands will be executed by ENTRYPOINT only.
- ENV → Enviornment variables.

"Dockerfile creation"

1) create a file named Dockerfile (capital)
2) Add instruction in Dockerfile
3) Build dockerfile to create image
4) Run image to create container

```
vi Dockerfile
:

FROM Ubuntu
RUN echo "Testfile" > /tmp/testfile
WORKDIR /tmp
ENV myname (name)
COPY testfile1 /tmp

ADD test.tar.gz /tmp
```

Page No.
Date

To create image out of dockerfile

- docker build -t (image name)

docker ps -a
docker images } verify

Now create container from the above image

- docker run -it --name (containername) (imagename) /bin/bash

Akash Kale

Page No.
Date

## practide

Step
① create Docker file

ec2-user ] # ls
nofile

Step
② ec2-user ] # vim Dockerfile ↵
command line mod
press i button ↵
↵ "Give instruction"

Step
③ FROM ubuntu { any OS } we can used
RUN echo "Hi sai" >/tmp/testfile
Dockerfile
RUN apt-get update -y
RUN apt-get install tree -y ↵
press enter
ctr+L Cleaning the screen.

Docker file

| |
|---|
| OS-ubuntu |
| testfile-Hi sai |
| tree |

image (OS + File +s/w)

testcontainer

| |
|---|
| OS-ubuntu |
| tree |
| testfile (Hi sai) |
| test |

image

Step
④ ec2-user ] # docker build -t test .

} successfully we have created image.

Page No.
Date

Step (5) ec2-user ] # docker images

} seen images

Step (6) ec2-user ] # docker run -it --name test container test /bin/bash ↵ enter

Step (7) container :/# ls

Step (8) container :/# ls tmp
testfile

Step (9) container :/# exit
exit

Step (10) same docker file → modification.

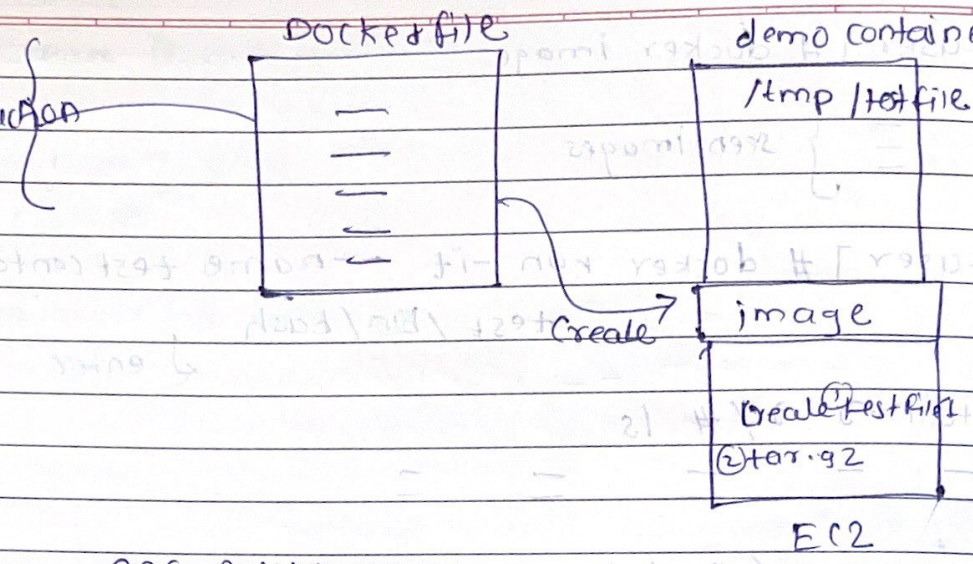
ec2-user] # vi dockerfile ↵ enter
:
↵

Delete this last line (modification).

modify DOCKERFILE

```
FROM ubuntu
WORKDIR /tmp
RUN echo "Hi sai" > tmp/testfile
ENV mynamehas
COPY testfile /tmp
ADD test.tar.gz
```

(Add = remove + copy)




esc : wq
enter

Step 11 ec2-user] # ls
• Dockerfile
Step 12 ec2-user] # touch testfile 1
Step 13 ec2-user] # ls
• Dockerfile testfile1
Step 14 ec2-user] # touch test
Step 15 e2-user] # ls
Step 16 Docker file test testfile1
Step 17
* ec2-user] # tar -cvzf test.tar.gz
ec2-user] # ls
Docker file test testfile1 test.tar.gz
Step 18
* ec2-user] # docker build -t demo.

} successfully build image

Step 19
* ec2-user] # docker images

Akash Kale

Page No.
Date

create container

Step20: ec2-user ] # docker run -it --name demo contain
demo /bin/bash

Step21: container : /tmp # ls
test testfile testfile1

Step22: container : /tmp # echo $myname
hari

Step23: container : /tmp # cat testfile
Hi sai

control + L

clear

## "Docker Volume and How to share"

Page No.
Date

- Volume is a simply a directory inside our container
- 
- Firstly, we have to decidre 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 volumes from existing container
- You can share one volume across any number of containers.

## "Create volume from Dockerfile"

Create a Dockerfile and write instruction.

```
vi Dockerfile
FROM ubuntu
VOLUME ["/myvolume"]
```

Then create image from this dockerfile

```
docker build -t myimage (image name)
```

Now create a container from this image & run

```
docker run -it --name (container name) (image name) /bin/bash
```

(myimage)

Page No.
Date

Now do is, you can see myvolume I

Now share volume with another container

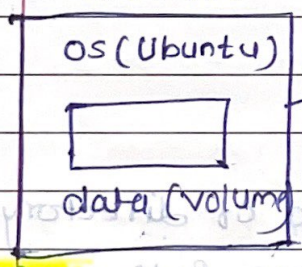
# Docker Volumes

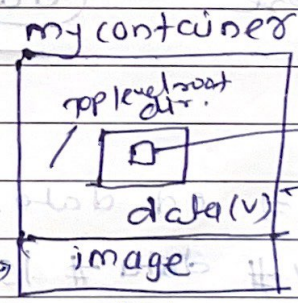
Class 31

**1st way :**

Docker file

- create one docker file
- from dockerfile i want to create one image.




**Practicle**

Step 1: ~]$ sudo su

Step 2: ec2-user] # service docker status

Active & running

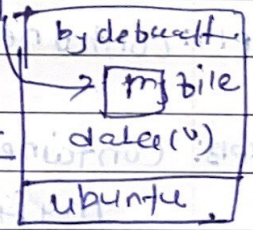
Step 3: ec2-user] # ls

Dockerfile testfile1 test.tar.gz

Step 4: ec2-user] # vi Dockerfile

press i button

Delete previous instruction




I want to change this dockerfile

Step5 :
Dockerfile

FROM ubuntu
VOLUME["/data"] ← create volume under top level root directory.

esc : wq ↵

Step6 :

← create image

ec2-user]# docker build -t myimg . ↵ enter

successfully we have created image.

↵ verify

Step7 :

ec2-user]# docker image ↵

Step8 :

↓ create one container using myimage.

ec2-user]# docker run -it --name myconT1 myimg /bin/bash

Step9 : container :/# ls ↵

seen volume here (data)

Step10 : container :/# cd data/

Instead of directory create one file

Step11 : container :/# data # ls ↵

Step12 : container : /data # ↵ touch myfile

Step13 : container : / data # ls ↵

myfile ↵

Step14 : exit

* I want to create mycont2 reberance from mycont1.

Step15: ec2-user] # docker run -it --name mycont2 --privileged = true --volumes-from ubuntu /bin/bash mycont1 ↵ enter

successfully created mycont2.

Step16: container : / # ls ↵

Data folder — —
— — — — —

Step17: container : / # exit exit.

another file created in mycontainer2 i.e. test. called volume map.

verify container are exited or start ↵

Step18: ec2-user] # docker ps -a ↵

— mycont1 exited
—

Step19: ec2-user] # docker start mycont1 ↵

mycont1

Step20: ec2-user] # docker ps ↵

container details here

Step21: ec2-user] # docker attach mycont1 ↵

container : / # ls ↵

data — —
— — —

Page No.
Date

Step22: container :/# ls data/
myfile
container :/# exit
exit

not created my file

Step23: ec2-user ]# docker start mycont2 reverse step
mycont2

Step24: ec2-user ]# docker attach mycont2

Step25: container :/# cd data/

Step26: container :/ data# ls
myfile ~~test~~

Step27: container :/ ~~data# exit~~ data # touch test

Step28: container :/data # ls
myfile test

Step29: container :/data # exit
exit

Step30: ec2-user ]# docker start mycont1 ← start cont1
my cont1

we can share the volume

Step31: ec2-user ]# docker attach mycont1
container :/# ls data/
myfile test

mapping volume between one container to another.

Step32: container :/# exit
exit

"container1 ⟷ container 2"

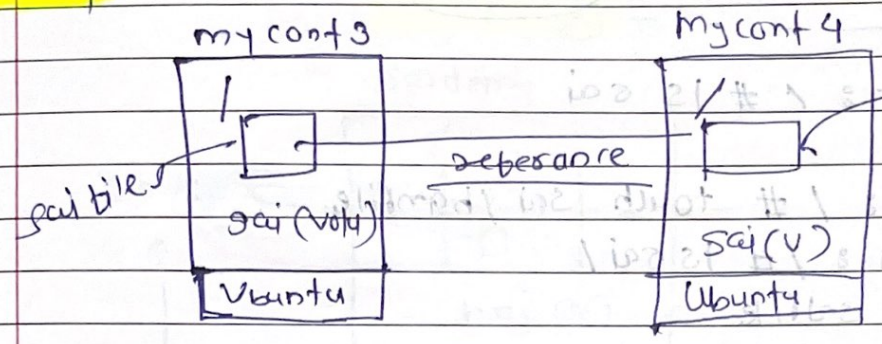
* docker run -it --name container2 --privileged=true --volume-from container1 ubuntu /bin/bas

Now after creating container 2, myvolume1 is visible. whatever you do in one volume, can see from other volume.

→ touch /myvolume1/samplefile
→ docker start container1
→ docker attach container1
→ ls /myvolume1
you can see samplefile here
exit.

Page No.
Date

**2nd way :**




**Practicle**

Step33:
ec2-user] # docker run -it --name mycont3 -v /sai ubuntu /bin/bash

Step34: Container : / # ls
sai — — — —

Step35: container : / # cd sai /

Step36: container : / sai # touch saifile

Step37: container : / sai # ls
saifile

Step38: container : / sai # exit
exit

* I want to create one cont3 by using ubuntu image. from referance mycont3.

Step39: ec2-user]
docker run -it --name mycont4 --volume-from mycont3 ubuntu /bin/bash enter
--privileged = true.

successfully we have created container

verify

Step40:
container : / # ls

Step41:
* container to container volume mapping
sharing volume

Page No.
Date

— — — — — saifile
— — — —

Step42: container : / # ls sai
saifile

Step43: container : / # touch sai/hamfile

Step44: container : / # ls sai/
hamfile saifile

Step45: container : / # exit
exit

start cont3

Step46: ec2-user] # docker start mycont3
mycont3 ~~docker~~

Step47: ec2-user] # docker attach mycont3

Step48: container : / # ls
— — — —
— — — —

verify

Step49: Container : / # ls sai
hamfile saifile

Step50: Container : / # sai exit
exit

—o/—

" Now create volume ~~by~~ using command "

*

docker run -it --name container3 -v /volume2 ubuntu /bin/bash

DO ls → cd /volume2

Now create one file container3 file and exit

Now create one more container, and share volume2.

Page No.
Date

* docker run -it --name container4 --priviledged=true
--volume-from container 3 ubuntu /bin/bash

Now you are inside container, do ls, you can see volume 2.

Now create one file inside this volume and the check in container can see that file.

## "Volumes (Host - container)"

verify files in /home/ec2-user

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

(/bash: container; /home/ec2-user: host)

Cd /rajput

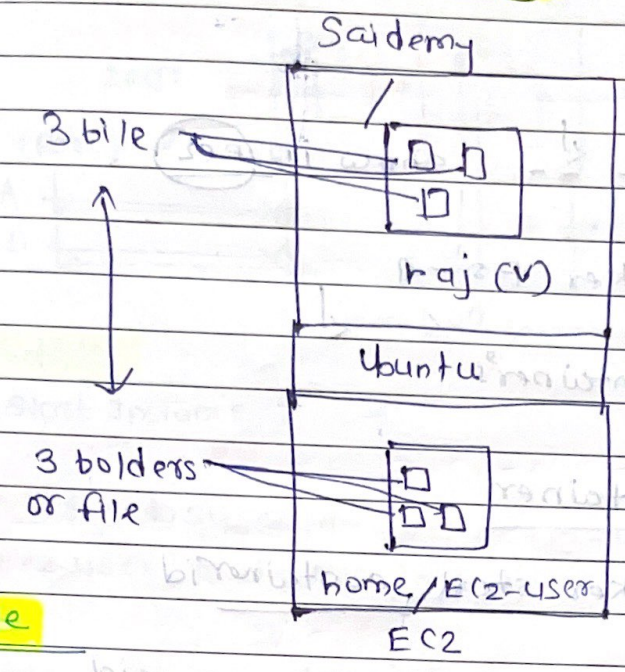
Do ls, now you can see all files at host machine.

touch dockfile (in container)

exit

Now check in EC2 machine, you can see the file

# "Host To Container mapping"




**Practide**

Step1:

ec2-user] # ls ← verify 3 file in ec2-user.

Dockerfile testfile1 test.tar.gz

Step2:

ec2-user] # docker run -it --name Saidemycont -v /home/ec2-user:/raj Ubuntu /bin/bash

successfully create saidemy container. ↵ enter

Step3:

Container:/ # ls

— — — — — raj

— — — — —

Step4:

container:/ # cd raj

Step5: container:/raj # ls

Dockerfile test.tar.gz testfile1

Step6: Container:/raj # touch rajfile.

Step7: Container:/raj # ls

→ extra bile seen raj

Step8: Container : /raj # exit ↵
exit

Step9: ec2-user] # ls
Dockerfile — — — ↵ show in ec2

Step10: ec2-user]# docker ps -a ↵
—
—
—
—
Container's

Start container

Step11: ec2-user] # docker start containerid.
container id
ec2-user] # docker attach containerid.

Container id : / # ls
— — — —
— — — —
Container : / # exit
exit.

Step12:
* ec2-user ] # docker inspect saidemy cont. ↵ enter
—
—
—
—
—
—
—
—

Page No.
Date

# "Docker port Expose"

→ Login with AWS account create one Linux instance
Now go to putty → Login as → ec2-user

→ sudo su
→ yum update -y
→ yum install docker -y
→ service docker status
→ service docker start
→ docker run -td --name techserver -p 80:80 ubuntu

(-td: daemon (detatched mode); -p 80:80: port, host; ubuntu: container)

→ docker ps
→ docker port (techserver) — container name
→ o/p : 80/TCP → 0.0.0.0/80

→ docker exec -it techserver /bin/bash

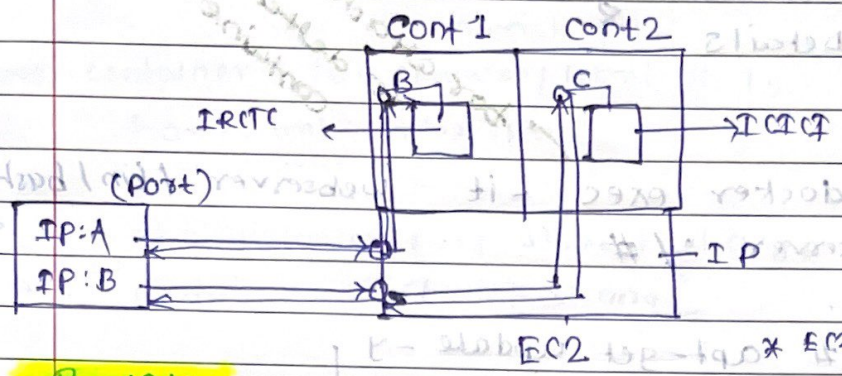
→ apt-get update
→ apt-get install apache2 -y
→ cd var/www/html
→ echo "This is docker port file" > index.html
→ service apache2 start

→ docker run -td --name myjenkins -p 8080:8080 jenkins

Class 32

Page No.

# "PORT MAPPING" / "PORT EXPORT" / "Expose"




* EC2 mapping with cont

## Practicle

Start Instance :

Step1 :

~ ]$ sudo su

Step2: ec2-user] # docker images

Step3: ec2-user] # service docker status

— — service is allready running

Step4: ec2-user] # docker ps -a

— — } all containers show

Step5: ec2-user] # docker ps

only show running container

Step6:

* ec2-user] # docker run -td --name webserver -p 2

80 : 80 ubuntu

host port — container port

enter

(-td terminal detached)

(-it - interactive terminal)

→ not interactive to cont.

that why td used.

dnt not give : /bin/bash - don't want to enter any folder

don't want to interact with terminal.

Page No.
Date

Step 7: ec2-user] # docker ps

container Details
webserver.

for attaching to detached mode container

Step8: ec2-user] # docker exec -it webserver /bin/bash

container webserver :/#

*
Update:
Step9: container :/# apt-get update -y

— Done

*
install:

Step10: container :/# apt-get install apache2 -y

select Geographical Area
&
select Time zone,

Done.

Step11: container :/# ls

var

Step12: container :/# cd /var/
Step13: container :/var# ls

www

Step14: container :/var# cd www/
Step15: container :/var/www# ls
html

Step16: container :/var/www# cd html/
Step17: container :/var/www/html#

} Debant data or contain

Page No.
Date

Step18: container : /var/www/html # echo "Hello Docker" > hari.html

Step19: container : /var/www/html # ls
hari.html index.html

* ec2 instance port should change.
* container port not change.

Step20: container : /var/www/html # cd ../../..

Step21: ——/— : /# exit
exit

Step22: ec2-user ] # docker ps
running container

PORTS
0.0.0.0:80->80/tcp
container port

* create detaching container

Page No.
Date

"Difference between docker attach and docker exec?"

Docker exec creates a new process in the container's environment while docker attach just connect the standard Input/output of the main process inside 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 some other process.

pid - process id
ppid - present processid.

"What is difference between expose and publish a docker?"

Basically you have three option :-

1) Neither specify expose nor -P

if you specify neither expose nor -P, the service in the container will only be accessible from inside the container itself.

2) only specify expose

if you expose a port, the service in the container is not accessible from outside docker, but from inside other docker container, so this is good for inter-container communication.

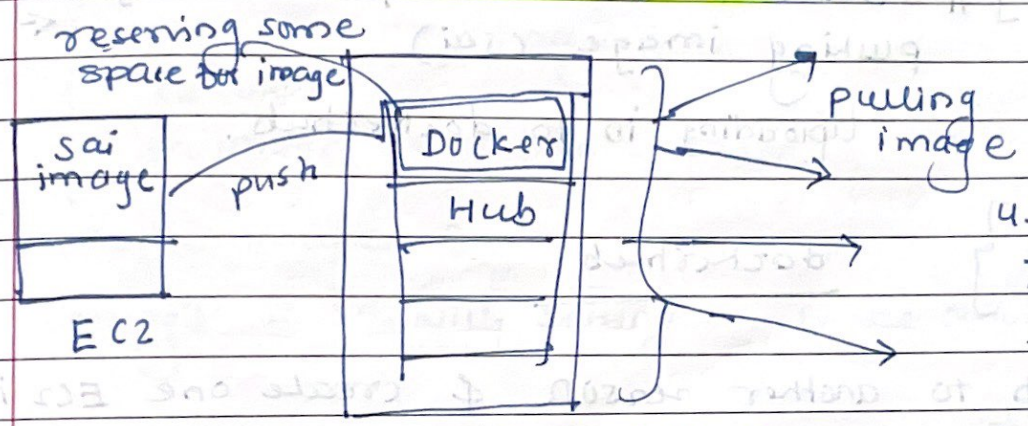
If you do -p but do not expose docker does ~~not~~ implicit expose. This is because, if a port is open to the public,

it is automatically also open to the other docker containers hence -p, include expose

3) Specify expose and -p

If you expose and -p a port, the service in the container is accessible from anywhere, even outside docker.

# "Docker Hub"




= } first 3 role very important for reserve some space in this hub.

Akash kale

Page No.
Date

## Practicle

open: hub.docker.com
create account : dockerhub.com

Step1: ec2-user] # docker image

Step2: ec2-user] # docker login

Username : hari2SCS
password : .....

Step3: Login Succeeded

Step4: ec2-user] # docker tag saiimage hari2SCS/newsaiimage

Step5: ec2-user] # docker images

Step6: ec2-user] # docker push hari2SCS/newsaiimage
pushing image (sai)
Uploading in to dockerhub.

Check in docker hub } dockerhub

Step7:
* switch to another reason & create one EC2 instance

EC2 - Japan

Launch EC2 Instance Tokeyo

Page No.
Date

Run this command :

Step 8: sudo su / yum update / yum install docker -y

Start the service :

Step 9: ec2-user ] # service docker start

= } service start

Step 10: ec2-user ] # docker images

No any image because new ec2 instance

Step 11: ec2-user ] # docker ps -a

No any container

Step 12:

pulling } → ec2-user ] # docker pull hari2scs/newsaiimage

pulling

Step 13: ec2-user ] # docker images

image created

Step 14: ec2-user ] # docker run -it --name japancontainer hari2scs/newsaiimage /bin/bash

successfully create container using sai image.

* { currently saiimage is a public image } so anyone can pull this image.

Step 15:

Make private image :

{ Setting's - Visibility setting - make private }

Step 16: ec2-user ] # docker ps -a

* container details

for container

remove container Step 17: { ec2-user ] # docker rm japancontainer

japancontainer.

Page No.
Date

Step18: ec2-user] # docker ps -a
Blue name image, container

Step19: ec2-user] # docker images
---- saiimage

Step20: ec2-user] # docker rmi hari2scs/newsaiimage
bari image

Step21: ec2-user] # docker images
can not see any images

Step22: ec2-user] # docker ps-a
you can not see any container

* we tried to pull private image

Step23: ec2-user] # docker pull hari2scs/newsaiimage
error (private)

Step24: ec2-user] # docker login
username :
password :
Login successeded
} provide username & pas to user

Step25: ec2-user] # docker pull hari2scs/newsaiimage
pulling
Download Complete

Step26: Delete Repository
1→ setting → Delete repository

Page No.
Date

Step 27:

ec2-user] # docker stop $ (docker ps -a -q)

all  quit

i/p  o/p

Successfully Stop

* remove all container — first remove container

ec2-user] # docker rm $ (docker ps -a -q)

- everything we have delete -

* remove all image & secondly remove containers

ec2-user] # docker rmi -f $ (docker images -q)

i/p of this command.  o/p

Deleting all the images.

verify :- docker images / docker ps -a (Command).

"How to push docker image in dockerhub"

Go to AWS account → Select Amazon Linux
Now go to putty → Login as - ec2-user

sudo su
yum update -y
yum install docker -y
service docker start
docker run -it ubuntu /bin/bash

Now create some files inside container
Now create image of this container

docker commit (container name) (imagename)
container1 image1

Now create account in hub.docker.com
Now go to EC2 instance

docker login
enter your username & password

Now give tag to your image

docker tag image1 dockerid / newimage
anyname
project

docker push dockerid / newimage

Now you can see this image in docker hub account

Now create one instance in other region and pull merge from hub.

Date

docker pull dockerid / newimage

Create new container from this image

docker run -it --name mycontainer dockerid /newImage /bin/bash

Page No.
Date

## "some other commands"

docker volume ls

docker volume create <volume name>

docker volume rm <volume name>

docker volume prune

(It removed all unused docker volume)

docker volume inspect <volume name>

docker container inspect <container name>

① Stop all running container :

docker stop $(docker ps -a -q)

② Delete all stopped container

docker rm $(docker ps -a -q)

③ Delete all images :

docker rmi -f $(docker images -q)