Docker Deep Dive

echo? (Successfully run command)

docker –-version

service docker stop

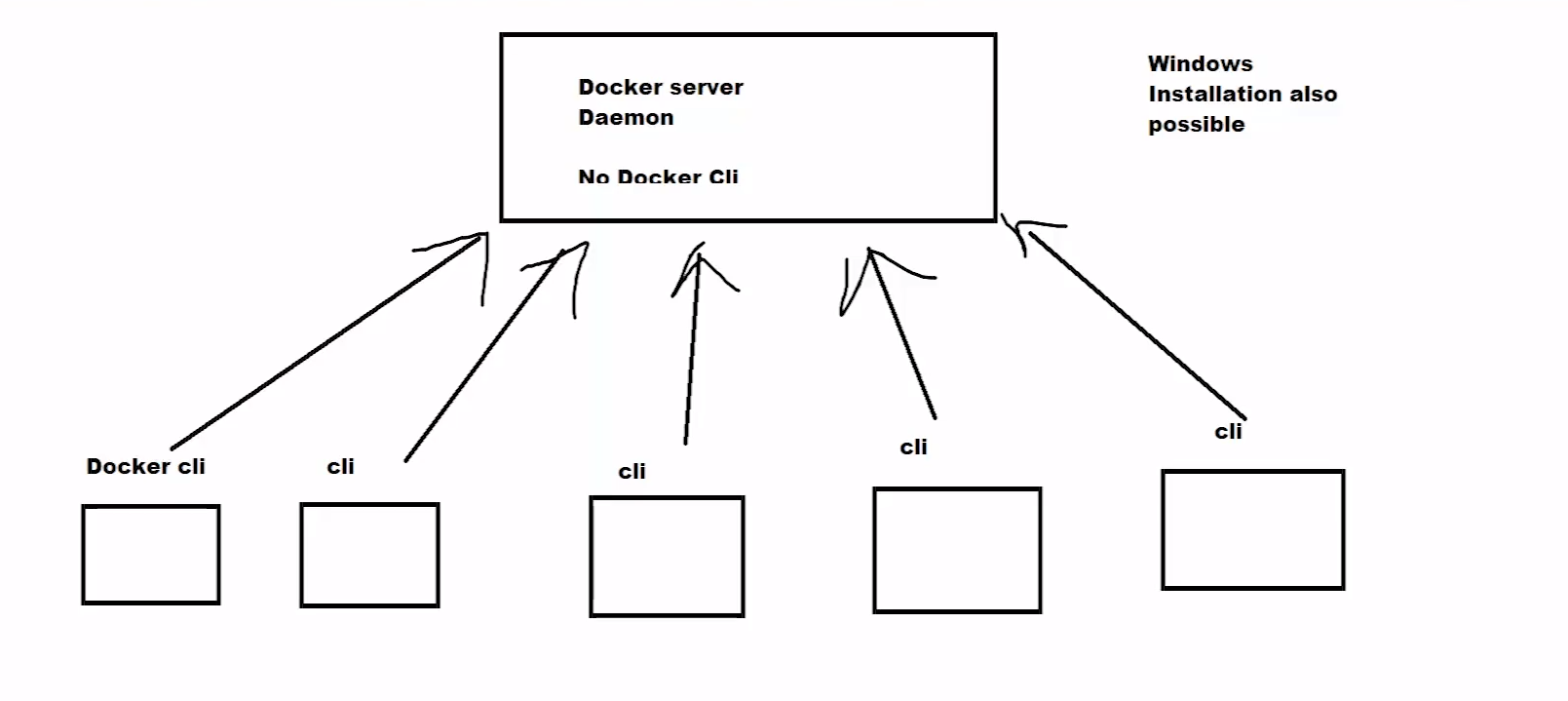
docker images

docker service start

**Interview Question:**

It is necessary to run docker server in same machine where we have docker client?

**Note: Docker server same machine - > Not necessary**

****

**Docker**

**Service docker start**

**Docker service status**

**Docker images**

**Docker container**

**Docker images ls**

**Docker ps (running containers)**

**Docker ps -a (all container)**

**Service docker stop**

**Lecture #3**

**Revision: -**

1. **We have seen 3 problems to find out docker important.**
2. **Images -> docker images only. Immutable can edit but execute.**
3. **Container: as a package of software because it contains application / Library / Dependencies, packages installation.**
4. **3 thing architecture: server, client, Registry**
5. **Docker installation**
6. **Some docker commands**
7. **Docker benefits**

**Question Time:**

1. **Is docker run command able to pull images.**

**Yes, it will pull images from docker registry.**

1. **How to list the docker images in the local (docker images)**

**Docker image ls and docker images**

1. **Steps to create a create docker images**

**We can write our own file -> docker images 🡪**

**If we don’t want write the docker file 🡪 We can pull images or download images from docker registry.**

**docker run hello-world**

**we have 2 more new images.**

**BusyBOX image: Stements print or any message use 🡪 docker run hello-world**

**Alpine image: Web server , Python , Java, mysql , promethoeous , and nagios etc.**

**Lightweighted and Secure.**

**CENTOS 5: 5 GB OS, Web server 1 GB , mysql 2 GB, Python 100MB ===8 to 9GB**

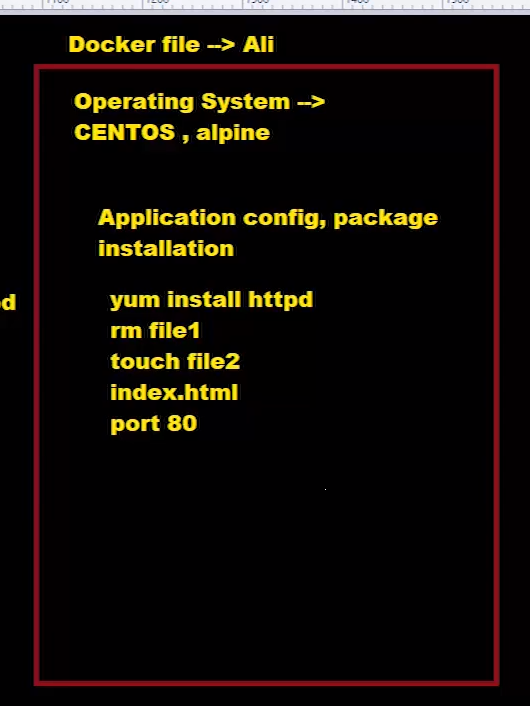
**Docker images 🡪 Dockefile -> or we can download from docker registry.**

**Docker Images:**

**We can docker container from docker images**

**Nginx, keyclock, RabbitMQ, mysql –5 to 6 Hours**

**Docker images: nginx, keyclock, rabbitMq, mysql 🡪 one time task**



Interview Question:

Yum install httpd 🡪 server start 🡪 systemctl start httpd

Install 🡪 Service start 🡪 web server 🡪 Docker file 🡪 Docker image 🡪 docker container as process

Docker registry: DockerHuB register

Docker pull hello-world

Docker rm containerid

Docker run –rm –name mycontainer hello-world

Docker pull imagesname

**Lecture# 04**

sudo docker run busybox echo “Hello from busy box…..”

docker images

docker ps

docker ps -a

docker busybox run busybox

docker images

docker rmi imageid

docker rm containername

When you want to delete images and container use **rmi** and **rm** command delete images for **rmi** and container use command **rm**

**Query Time:**

**#Docker run –name mycontainer busybox**

**Question:** What is use of above command.

**Answer:**

This will create container and execute with user define name like my container.

**#docker run –rm –name mycon1 busybox**

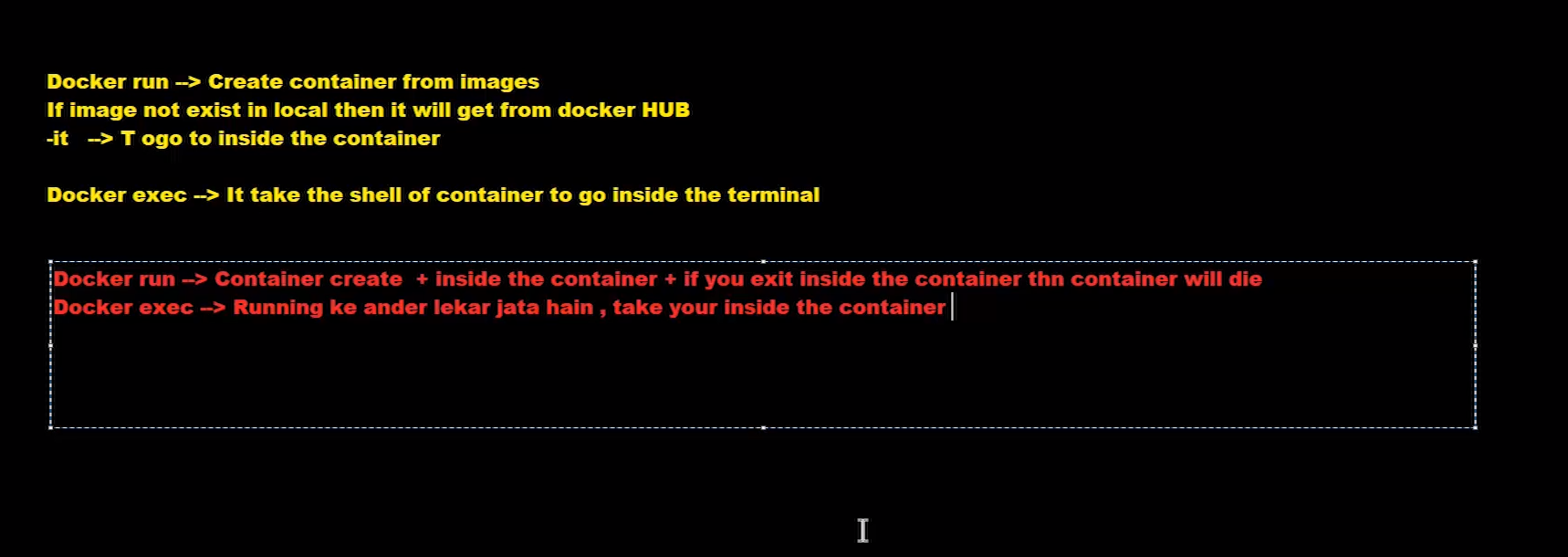
**Question:** What is use of above command**.**

**ANS:** Once the command will be executed then busybox container will remove from docker ps -a command or stopped container list.

-it stand for: interactive Terminal

Docker run -it –name container1 busybox /bin/sh

Docker start containername



Go inside the container then use below command.

Docker exec -it containername /bin/sh

When docker stop then use below command.

Docker stop containerId

List Concept: -

* Docker image 🡪 Are read only? Yes, we cannot edit image.
* Make container by using docker images. (Process + we can go inside the container and edit the file if you want)

**Reason: - When we create container it will be add one editable layer over image.**

**Docker images are readable and executable.**

**Lecture # 5**

**Sudo systemctl docker status**

**Docker Images**

* **Docker Image:**

**None writeable only readable file.**

**You can download readymade image in docker hub and docker registry**

**Commands:**

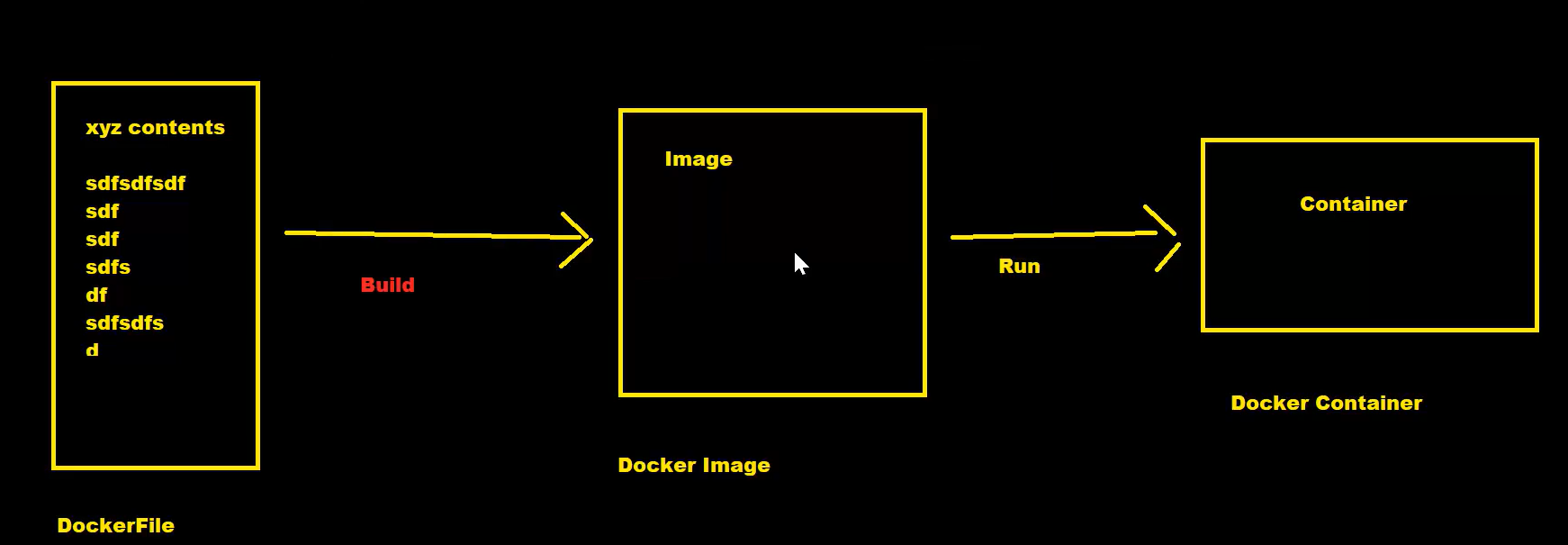
**Docker image pull**

**Docker pull image (First it will be found local from your machine then download from website like DockerHUB)**

**Docker run alpine (First it will be found from your local machine then download image and will create container)**

**Recipe: -**

1. **Dockerfile🡪 docker image**
2. **We should know docker file and what we need to write in docker file, how to write instruction**
3. **Create the image from dockerfile (Create dockerfile to docker image)**

****

**Dockerfile syntax: -**

**FROM: to define base image for docker**

**RUN: to execute the command in dockerfile.**

**When complete docker file after that run below command.**

**Command:**

**Docker build -t**

**-t: stand for tag the image 🡪 Give name of the image.**

**Build 🡪 Create the image from docker file**

**Create a Docker file**

**Vim Dockerfile (Create file in Linux).**

**FROM alpine**

**RUN apt-get install python**

**RUN echo $(pwd)**

**RUN touch file.txt**

**RUN cd /tmp && touch rashid.txt**

**Second Example**

**FROM alpine**

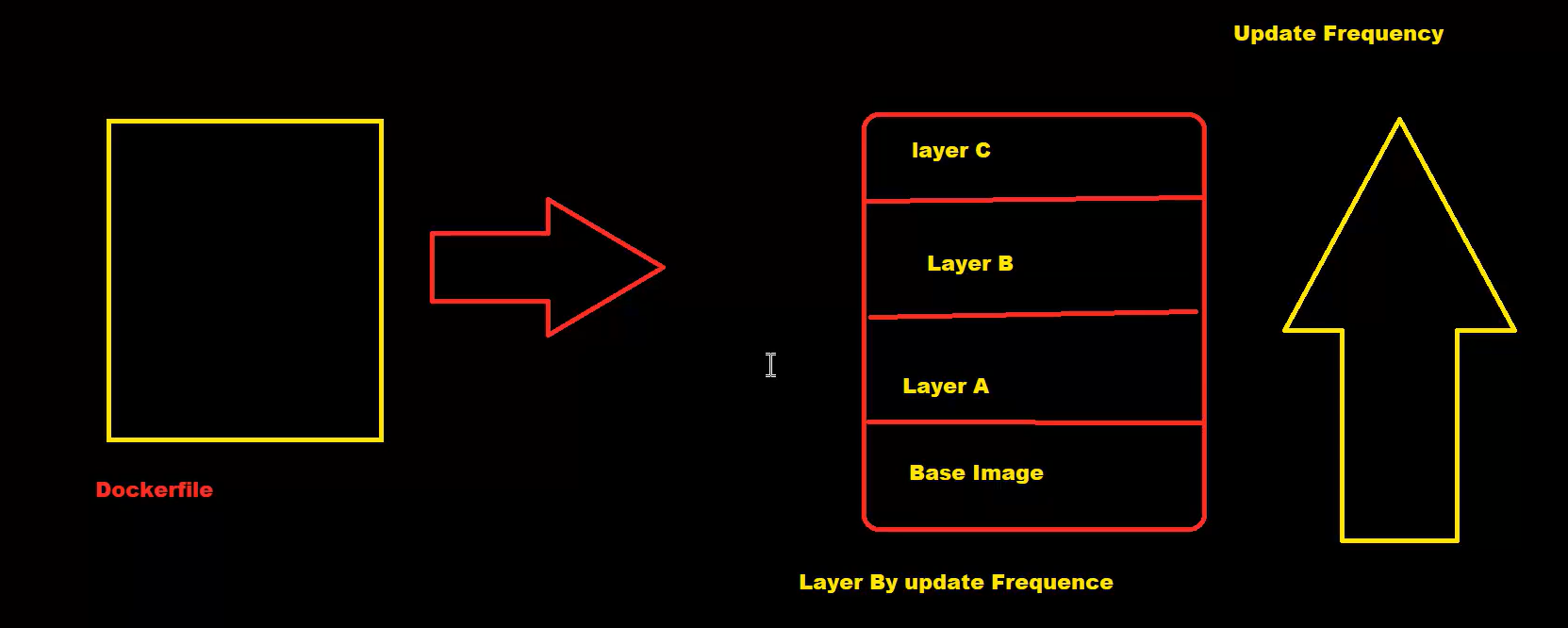
**echo “Hello this is a docker file”**

**RUN echo $(pwd)**

**RUN touch file.txt**

**Command:**

**Docker build -t firstimage . OR path**

****

**Docker run firstimage**

**Docker ps**

**Docker ps -a**

**Docker run –-name newcontainer firstimage**

**Docker run -it firstimage /bin/sh**

**Docker start imageid**

**Docker kill imageid**

**Docker rm delete container**

**Docker build -t mysecondimage:v1 .**

**Docker run -it imagename:v2 /bin/ash 🡪 (/bin/ash use for alpine image)**

**Interview Question:**

**If you want to reduce the count of image layers while building the docker image, what will be do?**

**Ans: User && (AND Operator)**

**What is version?**

**Webimage: v1,v2,v3 🡪 versioning**

**Docker version 20 , docker 21 etc.**

**Lecture # 06**

Command

docker run -d --rm -p 8080:8080 --name webserver busybox sh -c “while true; do {echo -e ‘Http/1.1 200 Ok\r\n’; echo ‘smallest http server’;} |nc-l -p 8080; done”

docker ps --no-trunc

-d is use for exit container when command will execute success

-p port number

sh -c is script

**Lecture # 07**

Commands

docker images

docker login

username:

password:

docker push

pull vs push

docker pull hello-world 🡪 hello-world download

docker push image name 🡪 Upload

Pull 🡺 Download

Push🡺 Upload

docker push aliimage: v2

docker tag <OldimageID: tag> <DockerUserID> /<NewImageID: tag>

docker tag rashidimage: v2 rashidliaquat/rashidimage: latest

docker tag busybox: latest rashidliaquat/mydemo:latest

docker push rashidliaquat/mydemo:latest

change in exiting image then

docker tag rashidliaquat/mydemo:latest rashidliaquat/mydemo:v1

docker push rashidliaquat/mydemo:v1

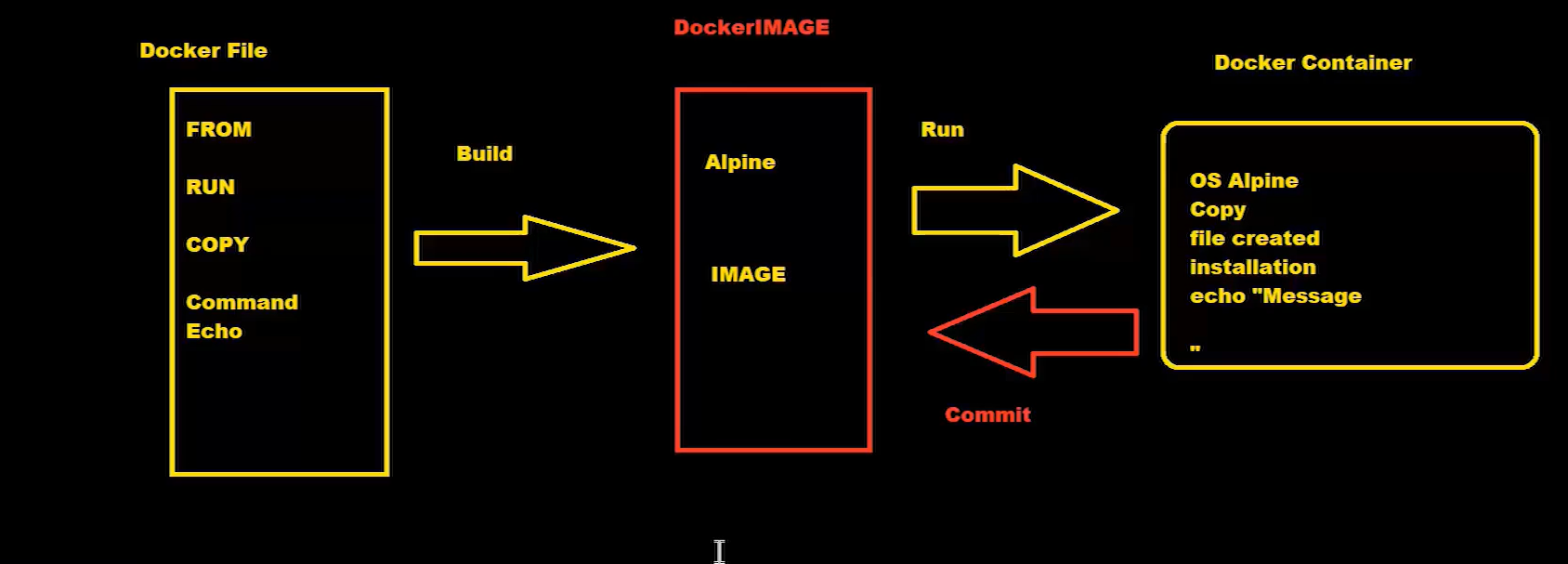
docker rmi rashidliaquat/mydemo:v1

docker pull rashidliaquat/mydemo:v1

docker images (see all images)

override: images use same name and version then exiting image data override images.

**Lecture # 08**

****

**Container to image**

**Reverse make container to image then use commit command**

**Commands**

**docker run -it --name my\_cont alpine /bin/sh**

make some file by using vi and touch command inside container then make a container to image

touch abc.txt

echo “Hello from docker” > sample.txt

ls -ltrh

apk add git

exit

docker ps

docker ps -a

docker commit

Docker commit: It will be created docker image from docker container.

docker commit my\_cont myfirstimage\_cont

docker rm my\_cont

then again create image

docker run -it --name mysecond\_container myfirstimg\_count /bin/sh

**Question Time**

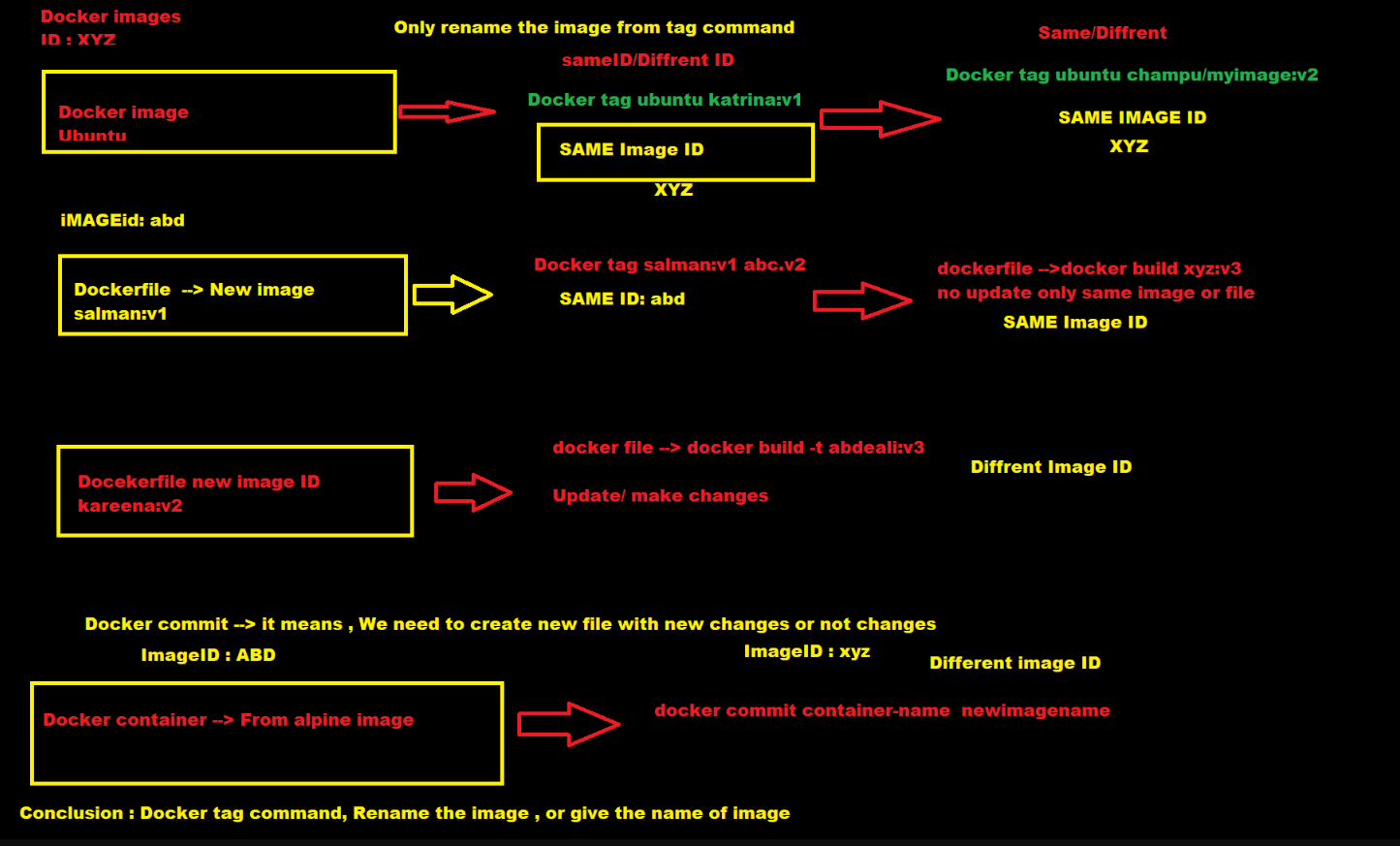
**Question:** I take ubuntu, create the container from Ubuntu image and when I run the ubuntu container the it exited.

Can I run this command.

**docker commit <Ubuntu containerId> <Dockerimage>**

**Question:** Will the image-ID same? Means Ubuntu imageid and new image created both are same? Docker commit command.

Image Id difference



**Image Id difference use case**

**Docker commit🡪 it means, we need to create new file with new changes or not changes.**

**Use Case**

**Question:** Make two containers by using alpine image

Container name

Container1 and container2 then use commit command for make again image for these containers, both containers id same or not?

**Ans**: different

docker run --name cont1 alpine

docker run --name cont2 alpine

docker commit cont1 image1

docker commit cont2 image2

**Make a docker file**

mkdir demo

cd demo/

ls -ltrh

vim Dockerfile

FROM alpine

RUN apk add git

RUN touch /tmp/file.txt

Make image by using above docker file

docker build . (Use . for current directory)

Update dockerfile again run docker build command.

Both created images contain different image Id.

**Dangling Image?**

If repo and tag name is none but image ID present in your docker images then it’s called dangling images.

**Question:** Show only dangling images.

docker images --filter “dangling=true”

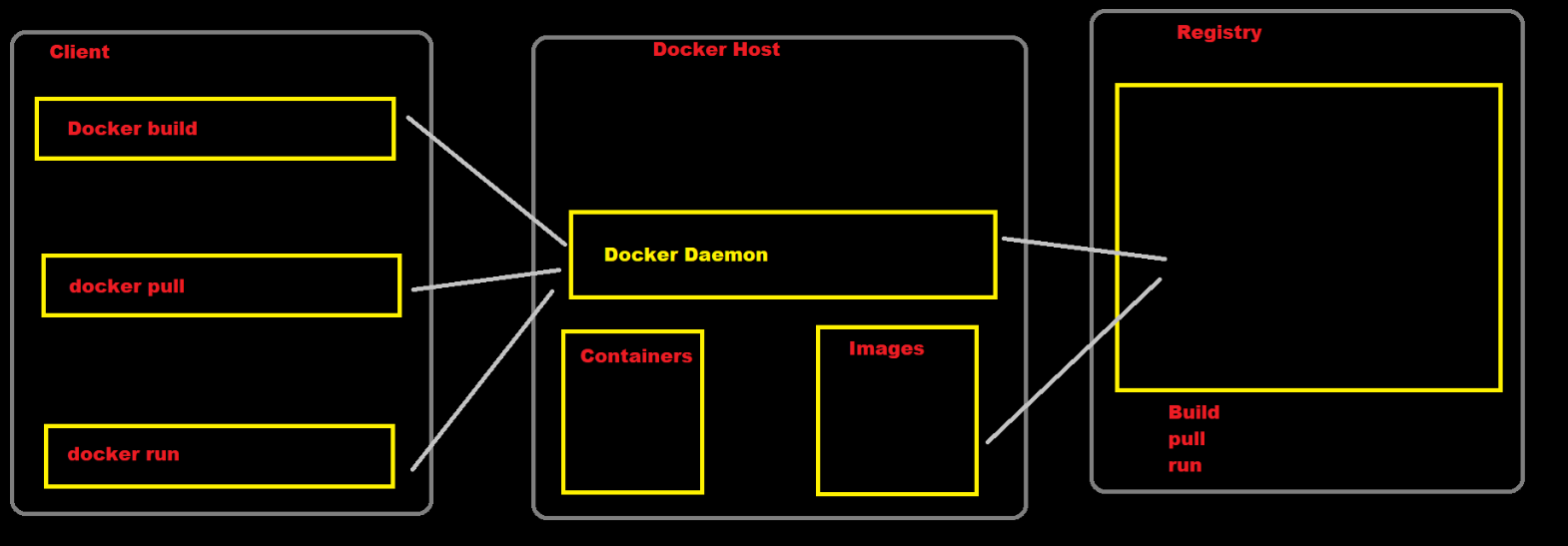
docker ps -a

docker ps -a -q (Only show container Id)

Delete all container

docker rm $(docker ps -a -q) (Delete all stopped containers)

**Lecture 9: Docker Notes**

****

Command

systemctl stop docker. socket (docker run this command then docker services not working all service)

system start docker. socket

docker status docker. socket

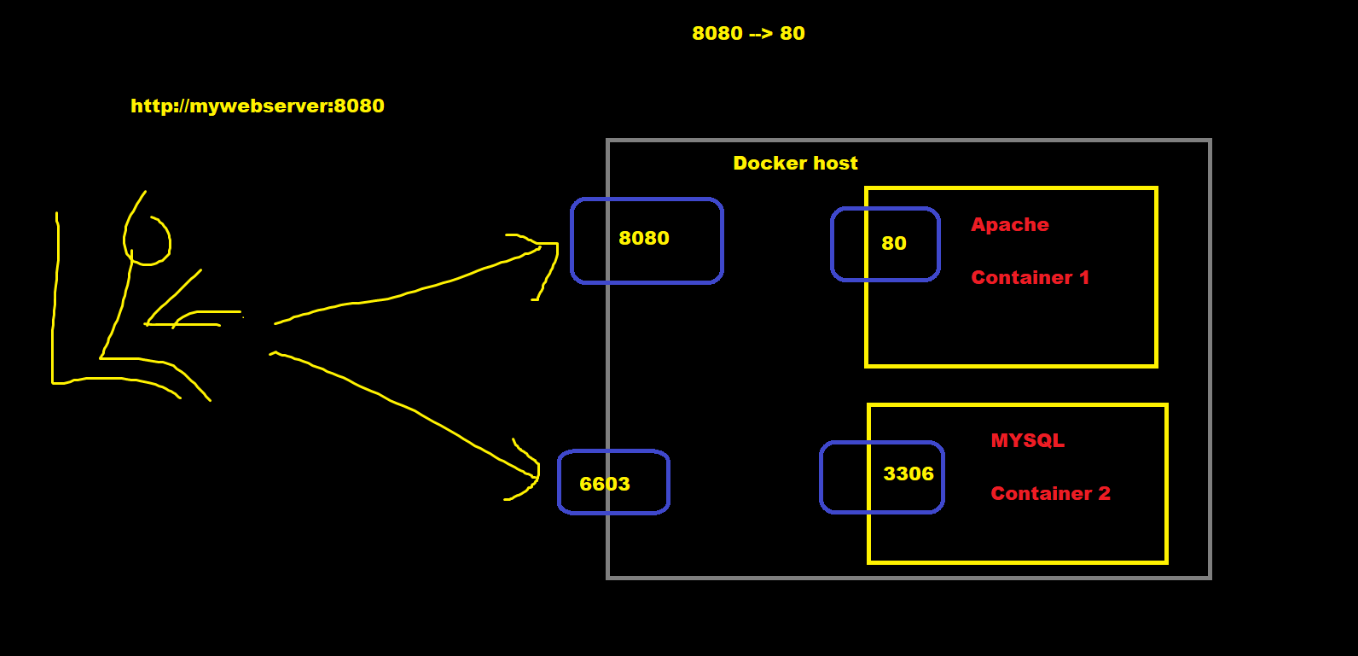
**Docker mapping**

Port 🡺 [www.google.com](http://www.google.com) 80 number on

telnet🡺 23🡺 ssh🡺 22

ftp🡺 21 tftp🡺 69

dhcp 🡺 67, 68 dns 🡺 53



Port Mapping (Client to host)🡺 Docker host 🡺 docker connector

**Command**

docker run -it -p 9000:80 –name mywebserver nginx: latest

docker run -it -d -p 9000:80 –name mywebserver nginx: latest (Run on background)

curl <http://localhost:9000>

netstat -tulpn | grep 90 (Show all service Active Internet connections (Only servers))

Run exit for terminate webserver.

Docker run – -name web-NEW nginx: latest

Above command will not create any binding between host and connector port.

for mapping, we need to use below command

Docker run – -name web-NEW -p 9000:80 nginx: latest

-p flag for port mapping

Docker run -it -p 80:80 –name web1 nginx🡺 Running?: yes

Docker run -it -p 80:80 –name web2 nginx 🡺 Running?: No

Docker run -it -p 8080:80 - -name web2 nginx 🡪 Running ? No (Name conflict)

Docker run -it -p 8081:80 - -name web4 nginx 🡺 Running? YES

**Home work**

1. Socket read about it
2. IT tables / Networking as per Linux (IP address / subnet: Good to have)
3. Docker push documents 🡺 Refer all images

Docker components

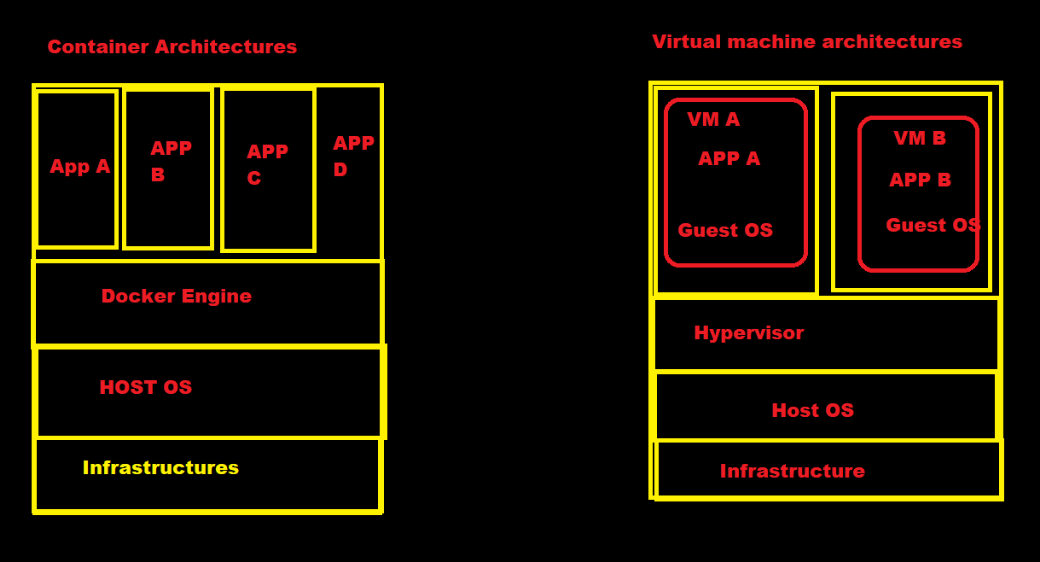
Docker socker file

Docker port mapping using nginx images

Docker conflict in container name or port

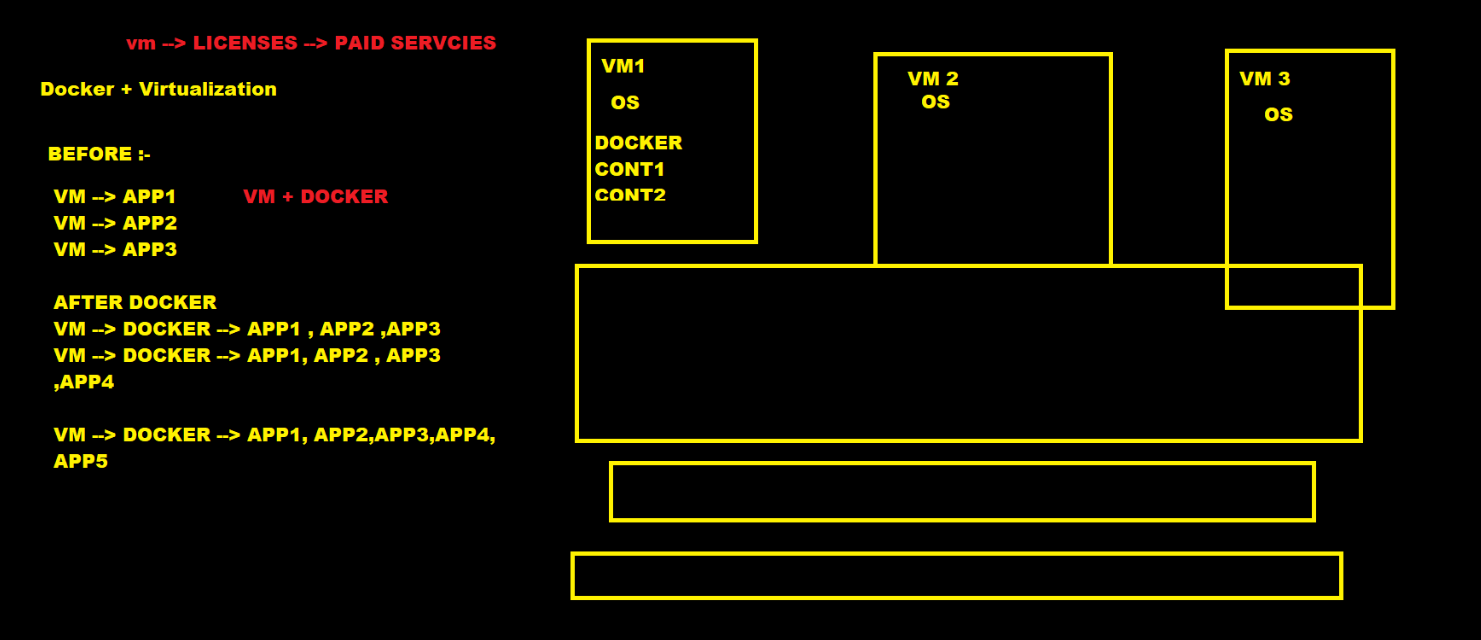
**Lecture 10: Docker Notes**

**Application Deploy**

****

**Container vs virtual Architecture**

**Use both ways for deployment**

****

**Vm and docker Architecture**

Dockerfile, Images, and containers

**Dockerfile:** 🡺A dockerfiles used to build docker image.

It is plain text contains and its series of instruction telling docker what os what app source code will do

**Docker images: 🡺** Docker image is static artifact that build from dockerfile and its tagged and publish to registry (DockerHub).

**Docker container: 🡺** It is running instances of docker images.

**Conclusion:🡺** Docker images combines source code with dependencies required to run application.

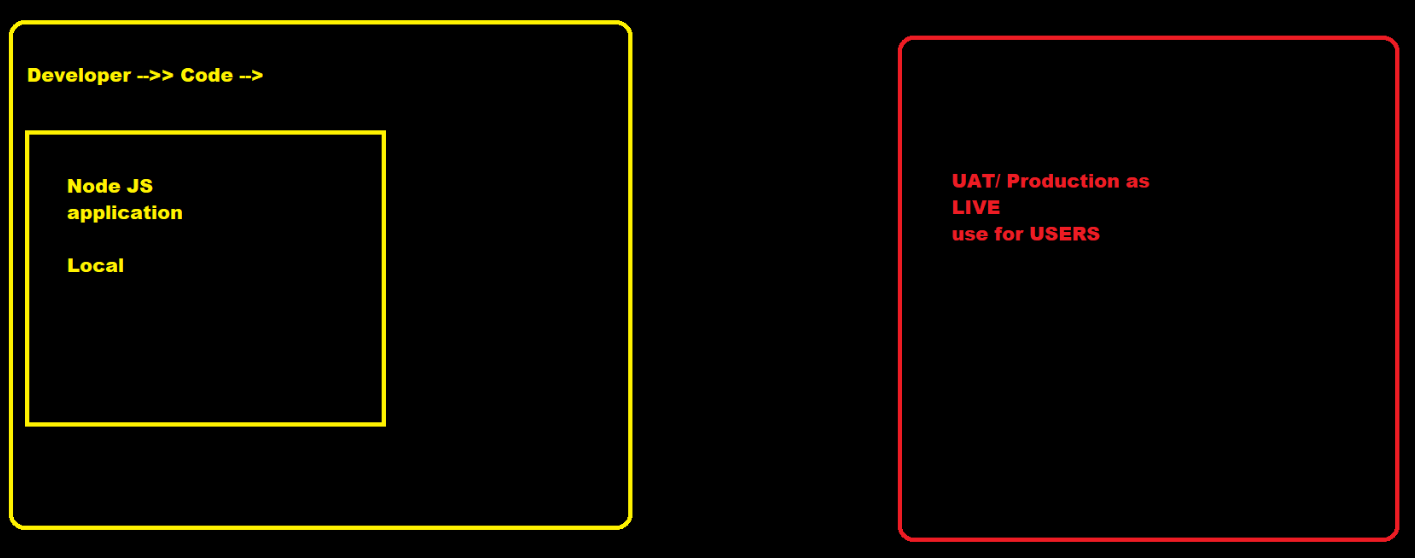
Lightweighted images, portable, developers we can shared.

**Dockerfile🡺 Docker image🡺 docker container**

**Step: 1** Tale the code from developer then run the application at your machine as local

* Node installed, dependencies, yum install nodejs
* Node version5
* Project

**How to deploy node application step by, step?**



**Command**

mkdir lect-10

cd lect-10/

vim package. Json

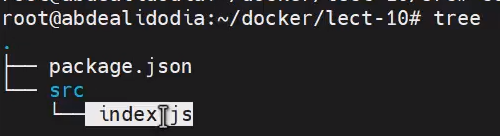
mkdir src

cd src/

vim index.js

Then move to main directory lect-10 check tree.

tree



For testing

mkdir testing

cp -vrf package. json src/

then check all file in testing folder

**Node JS installation process**

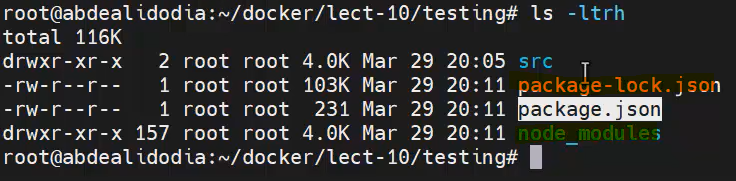
**Command**

curl -fsSL <http://deb.nodesource.com/setup_12.x> | sudo -E bash –

sudo apt-get install -g nodemon

npm install express

after npm install



npm start

npm run

npm init

package install🡺 np package installs 🡺 npm start🡺 npm open

Make docker file🡺 make image from docker file 🡺 run container then deploy application through docker.

**Dockerfile**

FROM node:15

ARG PORT=8000

ENV PORT=$PORT

WORKDIR app

Copy src src

Copy package. json .

RUN npm install

EXPOSE $PORT

CMD npm start

After complete dockerfile run below command for making Dockerimage

docker build -t my\_node\_app .

docker run - -name my\_node\_app1 -p 8000:8000 -d my\_node\_app: latest

docker exec -it containerId

docker logs (Check all logs)

ctrl +p + q (Exit without stop the container)

**Assign new port**

my\_port=5000

docker run –name my\_node\_app1 -p 9000: $my\_port -d -e PORT=$my\_port my\_node\_app: latest

curl localhost:9000

**Lecture-11**

****

**Practical phase 1 and 2 document.**

When you type run command using docker run 🡺 you will press or type exit command inside the container then your container will be died.

When you type exec command when container is already running mode then you press ctrl + p + q

Exit command inside the container then your container will not be died.

docker run - -name my\_deamonized -d ubuntu /bin/sh -C “while true; do each my deamonized container; sleep 1; done”

**Lecture-12**

**Practical phase 2 with Solution**

****

mkdir miniapp

#Vi package.json

{

"name": "docker\_web\_app",

"version": "1.0.0",

"description": "Node.js on Docker",

"author": "First Last <first.last@example.com>",

"main": "server.js",

"scripts": {

"start": "node server.js"

},

"dependencies": {

"express": "^4.16.1"

}}

**vim server.js**

'use strict';

const express = require('express');

**vim Dockerfile**

FROM node:14

# Create app directory

WORKDIR /usr/src/app

# Install app dependencies

# A wildcard is used to ensure both package.json AND package-lock.json are copied

# Where available (npm@5+)

COPY package\*.json ./

RUN npm install

# If you are building your code for production

# RUN npm ci --only=production

# Bundle app source

COPY . .

EXPOSE 8080

CMD [ "node", "server.js" ]

**vim .dockerignore**

node\_modules

npm-debug.log

docker build -t rashidliaquat/mynode\_application .

docker images

docker run -p 49160:8080 -d rashidliaquat/mynode\_application

docker logs containerId

docker exec -it containerId /bin/bash

curl localhost: 8080

ctrl + p + q

curl -I localhost:49160

curl localhost: 49160

**Q3: -**

docker container run --rm --name nginx --publish target=80,published=127.0.0.1:8081,protocol=tcp -d nginx

docker container run --rm --name nginx -p 80:127.0.0.1:8081/tcp -d nginx

docker container port containerId

**Q4**

docker run -it busybox /bin/sh

mkdir -p /data/html

cd /data/html/

echo “This is busybox container” index.html

cat index.html

ctrl + p + q

docker commit -p -a “Rashid Liaquat<[rashid.liaquat68@gmail.com](mailto:rashid.liaquat68@gmail.com)”> -m “Changes on index.html file” containerId or name rashidliaquat/myboximage:latest

docker ps

docker inspect containerId

docker run -it - -name rashidliaquat/myboximage /bin/sh

ls

httpd -h

ctrl + p + q

docker ps

docker commit -a “rashid<rashid.liaquat@gamil.com>” -c ‘CMD [“/bin/httpd”, “-f“, ”-h”, “/data/html”]’ -p rashid1 rashidhidliaquat/mysecond\_box:v2

docker run –name myboxcontainer rashidliaquat/mysecond\_box:v2

docker inspect containerId | grep IPAddress

curl 172.17.0.5

**Q5**

mkdir Q5

cd Q5/

vim index.html

<h2>It Works</h2>

<h3>Welcome to Alnafi</h3>

**Vim Dockerfile**

FROM tecadmin/ubuntu-ssh:16.04

LABEL maintainer=”[rashid.liaquat49@gmail.com](mailto:rashid.liaquat49@gmail.com)”

RUN apt-get update && apt-get install -y apache2

COMPY html/\* /var/www/html/

WORKDIR /var/ww/html

CMD [“apachectl”, “-D”, “FOREGROUND”]

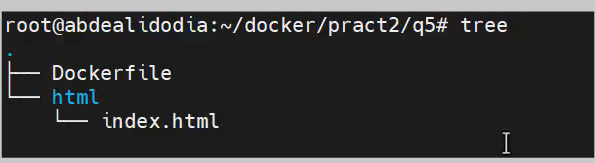
EXPOSE 90

**End Dockerfile**

mkdir html

mv index.html html/

tree



docker build -t my\_ubuntu\_apache .

docker run -it my\_ubuntu\_apache /bin/bash

ls

cat index.html

ctrl + p + q

**Q.6**

docker run ubuntu:18.04 /bin/sh -c “while true; do echo Rashid Liaquat; sleep 1; done”

docker run -d ubuntu:18.04 /bin/sh -c “while true; do echo Rashid Liaquat; sleep 1; done”

**Q.7**

docker kill containerId (Single container terminate)

docker kill $(docker ps -q) (for all container terminate)

docker container prune (for all container terminate) dangers command

docker rm -v $(docker ps -aq -f status=exited)

**Q.8**

mkdir q8

docker pull busybox

docker run -it busybox /bin/sh

ls

touch abd.txt

cat > abd.txt

Rashid

Liaquat

From

Alnafi

docker commit containerId rashidliaquat/my\_box\_abd:version1

docker run -it rashidliaquat/my\_box\_abd:version1 /bin/sh

cat>abd2.txt

Rashid

Liaquat

From

Alnafi

Ctrl + p + q

docker commit –change “ENV DEBUG=true” containerId rashidliaquat/my\_box\_abd:version

docker inspect -f “{{ .Config.Env }}” rashidliaquat/my\_box\_abd:version2



docker inspect -f “{{ .Config.Env }}” rashidliaquat/my\_box\_abd:version1



**Q.9**

mkdir q9

cd q9

vim Dockerfile

FROM alpine

RUN touch /tmp/abd.txt

**End dockerfile**

cat Dockerfile

docker build -t image1: v1 .

docker build -t image1: v2 .

docker build -t image1: v3 .

docker images

docker tag image1: v1 rashidliaquat/image1: v1

docker tag image1: v2 rashidliaquat/image1: v2

docker tag image1: v3 rashidliaquat/image1: v3

docker login

docker push rashidliaquat/image1: v1

docker push rashidliaquat/image1: v1

docker push rashidliaquat/image1: v1

docker rmi rashidliaquat/image1: v1

docker rmi rashidliaquat/image1: v2

docker rmi rashidliaquat/image1: v3

docker pull rashidliaquat/image1: v2

docker run -it rashidliaquat/image1: v2 /bin/ash

docker rmi rashidliaquat/image1: v2

docker pull –all-tags rashidliaquat/image1

**Q.10**

**Go to DockerHub**

docker pull jenkinsci/Jenkins

docker run -d –name my\_jenkins -p 7000:8080 jenkinsci/jenkins

curl localhost:7000

docker exec -it containerId /bin/bash

ls

cd /var/jenkins\_home/secrets/

cat initialAdminPassword

pwd

**Lecture # 13**

**Volume topic**

I am losing data from container 🡪 Persistent

Container remove, kill or stop or anything 🡪 never loss your data

Container as process

Docker volume

Three-way data types stored?

**Docker storge types: -**

Docker Volume (Important)

Bind Mount (Important)

tmpfs (not Important)

1. **Volume**: it is also called docker volume, data will be stored via docker area into base machine
2. **Bind mount**: it is also called docker bind, Container data will be stored into file system like vfs, zfs, overlay, overlay2
3. **tmpfs**: it’s stored into RAM when reboot your machine then your data will wipe out.

Docker area 🡺 Docker home

Docker home🡺 Docker database

Docker Area OR Docker home OR Docker database🡺 path /var/lib/docker/volumes

docker volume ls

docker volume path🡺 /var/lib/docker/volumes

docker images Path🡺 /var/lib/docker/image/overlay2/imagedb/metadata/sha256

systemctl status docker.service

service docker status

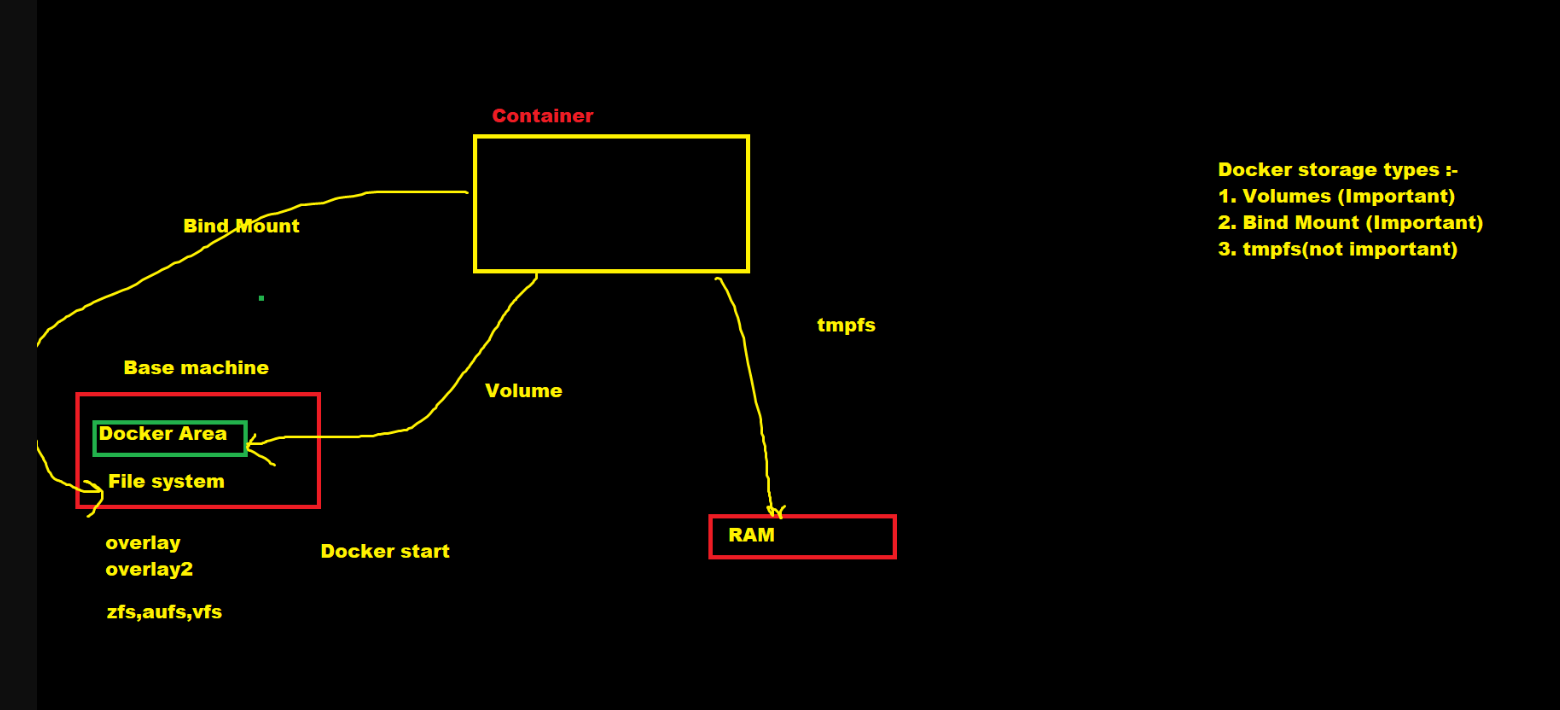
Docker storage: -

1. Docker volumes: - These are stored on the host file system that docker managers, on my ubuntu system path is /var/lib/docker/volumes

2. Bind Mounts: - These allow storage to be mounted from anywhere on the host system. for example, /home/aadmin/data. you could even mount /data,

/home/aadmin/data --> changes happened

/data --> changes will replicated



**Docker volume types**

**Lecture # 14**

1. IT company 🡺 Development 🡺 Code developed
2. Code Development 🡪 Containerization 🡺 (Dockerization, Kubernetes)
3. Scratch application run 🡺 locally🡺 Package/Port/Open install
4. Dockerfile create your own🡺 That’s why there are hire you
5. Docker image 🡺 policy need to apply🡺 register every company 🡺 DockerHub🡺 Private repository
6. Docker image pull 🡺 Like run the container
7. Application port change 🡺 container/Host post 🡺 ENV, variable then you need to run docker.
8. Docker data persistent 🡺 Permanent data saved into your shared or local drive.
   * + - 1. Docker volume
         2. Bind mount
         3. tmpfs

Command

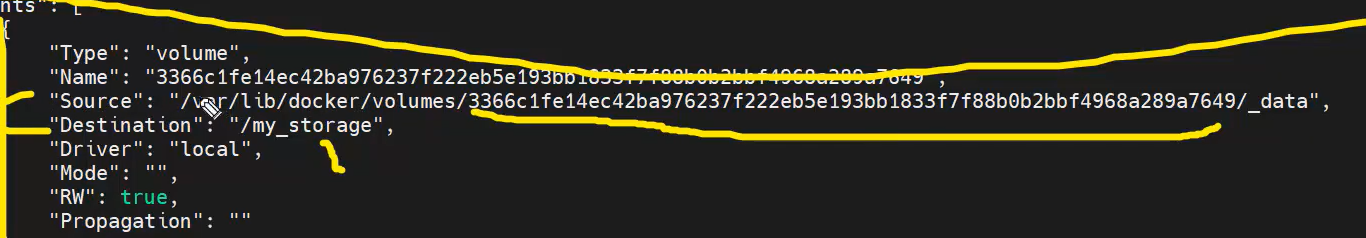
docker run -it --name container1 -v /my\_storage alpine/bin.sh

cd /var/lib/docker/volumes/

docker volume ls

docker start containerId

docker inspect containername



**Method🡺1 docker volume**

docker exec -it container1 /bin/sh

docker volume –help

create (Create volume)

inspect (Display details)

ls (List volume)

prune (Remove all unused local volumes)

rm (Remove one or more volumes)

docker run -it --name container2 -v /my\_storage alpine/bin/sh

**Method🡺 2 Bind mound**

mkdir /home/aadmin/my\_data/

one way

docker run -it --name contaier3 -v /home/aadmin/my\_data/:/opt/data alpine /bin/sh

2nd way

Docker run -it –name container3 -v /home/aadmin/my\_data/:/opt/data alpine /bin/sh

**Method🡺 3 tmpsf**

One way

docker run -d -it --name tmptest1 –mount type=tmpfs, destination=/app alpine /bin/sh

docker exec -it tmptest1 /bin/sh

create something inside container.

2nd way

docker run -d -it –name tmptest2 –tmpfs /app alpine /bin/sh

**Example**

Create image by using below command

docker build -t node\_app .

docker images

docker run –name mynode1 -p 8000:8000 -d -v /root/docker/lect-10/src/:/app/src node\_app

docker exec -it mynode1 /bin/sh

ls

pwd

curl localhost:8000

**Lecture # 15**

**Docker Image variants and change default docker Path.**

**Command**

docker pull node:15

docker pull node:15-slim

docker pull node:15-alpine

docker images | grep -I node

docker images | grep -w node

Two types of images

1. Alpine
2. Busybox

Best size you need mange for docker admin or devops admin

**Docker default path**

cd /var/lib/docker/

Key file location : /etc/docker/key.json

vim key.json

vim /etc/docker/daemon.json

mkdir /home/docker

vim /etc/docker/daemon.json

{

“data-root”: “/home/docker”

}

then go to stop docker services

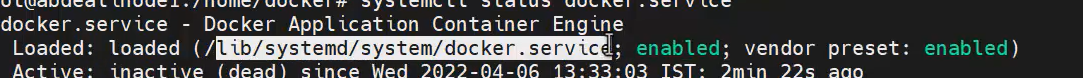
systemctl stop docker. socket

systemctl stop docker. service

cp -axT /var/lib/docker/ /home/docker/

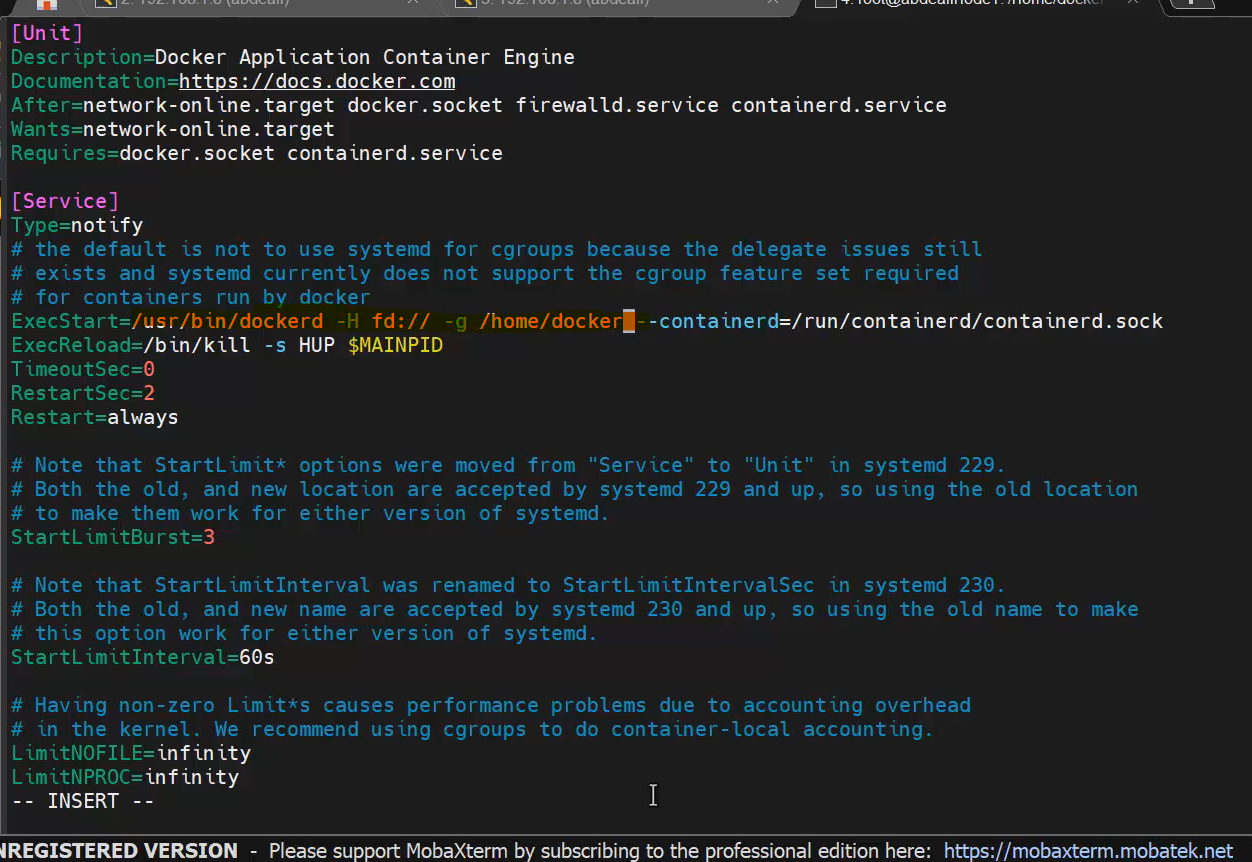
then check status

systemctl status docker. service



Then open above file mention in screenshot.

vim /lib/system/system/docker.service



Update above file add -g and required path for example /home/docker

then restart docker service

systemctl start docker. service

then reload the daemon by using below command

systemctl daemon-reload

systemctl start docker. service

systemctl start docker. socket

vim /lib/system/system/docker.service

systemctl restart docker. service

New Path or updated path: /home/docker/image/overlay2/imagedb/content/sha256

systemctl status docker. service

**Lecture # 16 (Docker Multi stages)**

Docker file 🡺 Production, UAT, DEV