INDEX

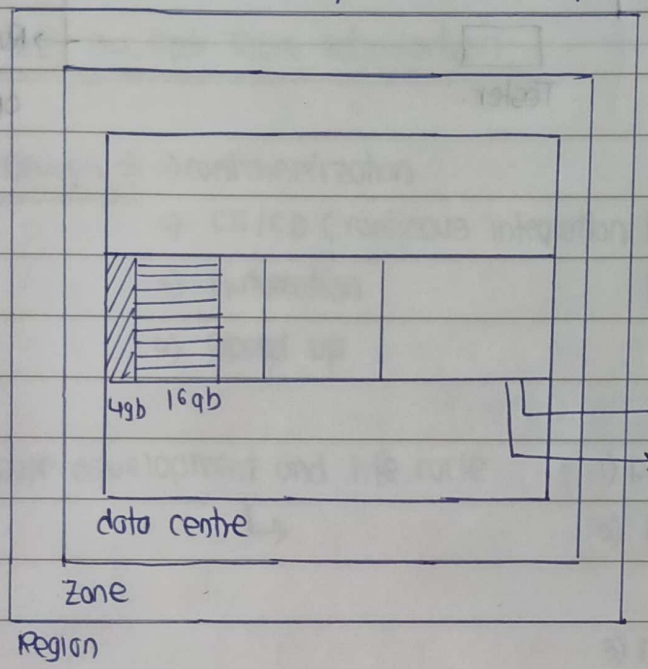
NAME: Sushant Aher STD.: _____ SEC.: _____ ROLL NO.: _____ SUB.: Devops.

S. No.	Date	Title	Page No.	Teacher's Sign / Remarks
Day 1)		Day-1 Inhoduction (virtualization, containirization)		
Day 2)		Day-2 Devops tool Inhoduction		
Day 3		Docker.		
Day 4		Docker		
Day 5		Docker (image pushing to Dockerhub and ECR)		
Day 6		Docker (volume, wordpress)		
Day 7		Docker (wordpress, Dockerfile)		
Day 8		Docker (Pythonapp with Dockerfile)		
Day 9		Docker (Network, Nodejs)		
Day 10		Docker (mysql with Dockerfile)		
Day 11		mysql with Dockerfile with .sh & .sql,		
		shell scripting nginx		
Day-12		multistage building, wordpress Docker-compose		
Day-13		volume mapped for synchronization		
Day-14		3 tier orchitecture with docker-compose file		
Day-15		kubernetes (why, Architecture of kubernetes)		
Day-16		why kubernetes, kubernetes object		
Day-17		ECS		
Day-18		Portainer		

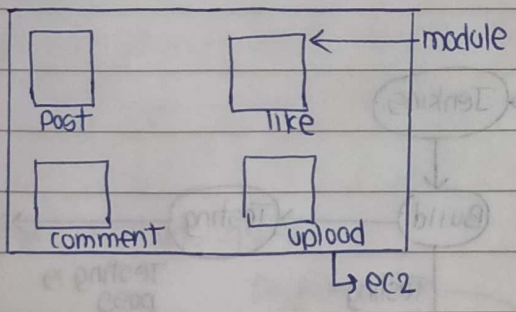
Day-01 Introduction (virtualization, containerization)

Virtualization = what is mean by virtualization

virtualization means that server is launch on physically storage by virtually with the help of the hypervisor.



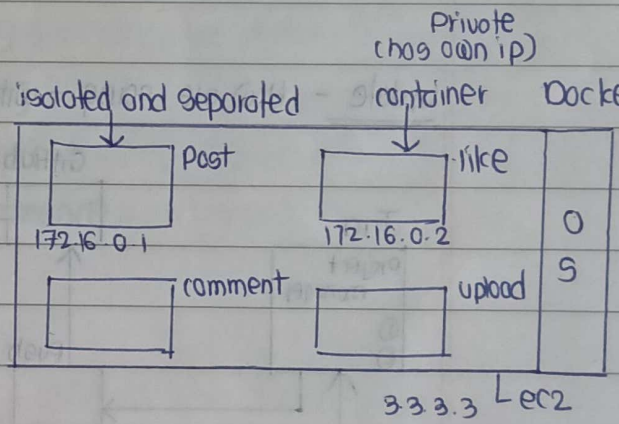
virtualization is using before devops.



Monolithic Architecture -

in one ec2 we created four many modules to separating code section but if one module is down then hole server is get downtime.

Solution - microservice Architecture



Microservice Architecture (containarization Architecture -

in one ec2 we created contain with help of docker, in it if comment get down then it not affect other containers they running live.

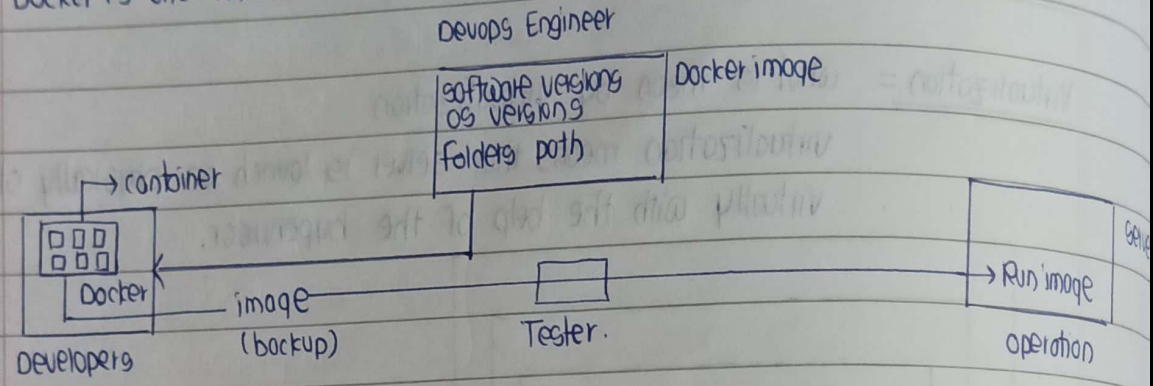
In microservices Architecture not dependencies between its containers.

Why need of devops ?

Why need of containarization ?

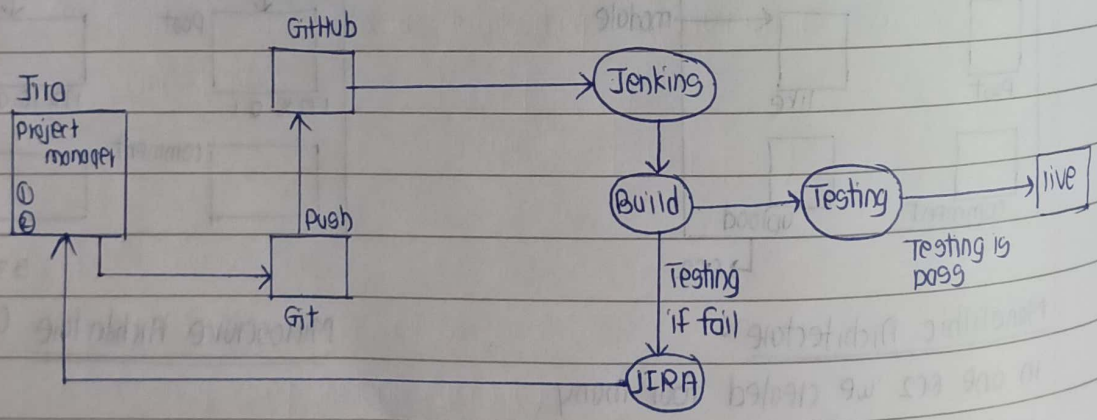
diffn between virtualization and containazation ?

Docker is one containarization technology

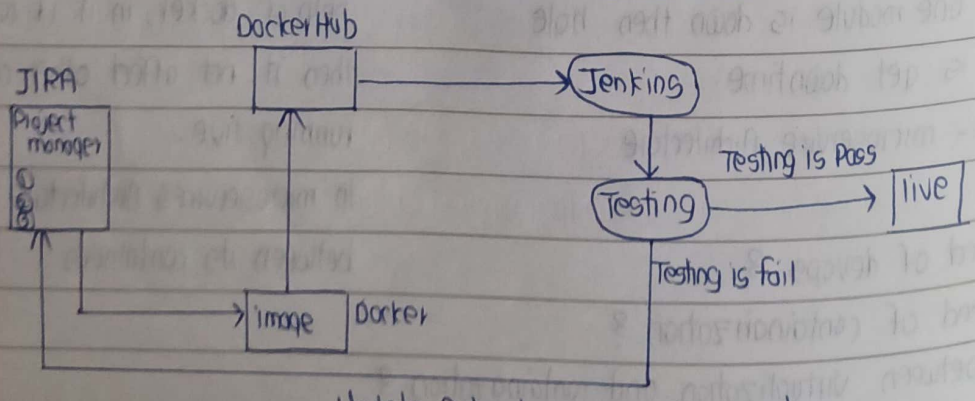


SDLC — requirement analysis
estimating feasibility
designing
coding
document
test
deployment
maintenance.

update - How are going - with Automation



Update Automation in virtualization



Update Automation in containirization

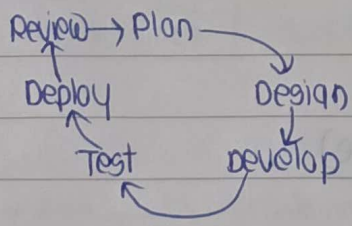
Why we need devOps ⇒ 1) containerization
2) version mismath and os
3) update
4) Automation

Day-02 DevOps Tools Introduction

Why Devops ⇒ 1) containerization
2) CI/CD (continous integration / continous Deployment)
3) Automation
4) Speed up

Software development and life cycle
↳
1) Requirement Gatering and analysis
2) Planning (Data modeling)
(Timeline)
3) Design
4) Coding
5) Testing
6) Deployment
7) Maintainance.

Agile methodology ⇒



- in each functionality with approved of client.
- Each step it gives feedback.
- decrease the cost.

Waterfall model ⇒ step, if one fail then restart from starting
cost increase.

Devops support to Agile methodology.

1) Docker - containerization (for Applying microservices)

2) kubernetes - contain orchitestration tool

- Autoscaling of
- load balancing
- Rollback / update
- version management
- Networking of

} Container

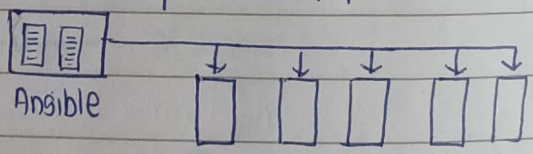
3) Git
- Source code management
- version control system

4) GitHub
- centralised / source code storage / management

4) Jenkins
- CI / CD pipeline
- Automation

5) Terraform
- creating infrastructure easily. HCL language used.

6) Ansible
- Ansible is used when we downloading software / update in many machine at same time.



7) Promethus and Grafana
- Timeseries database (Promethus)
- monitoring using Grafana (website)

8) JIRA
- Project troking (Project manager)

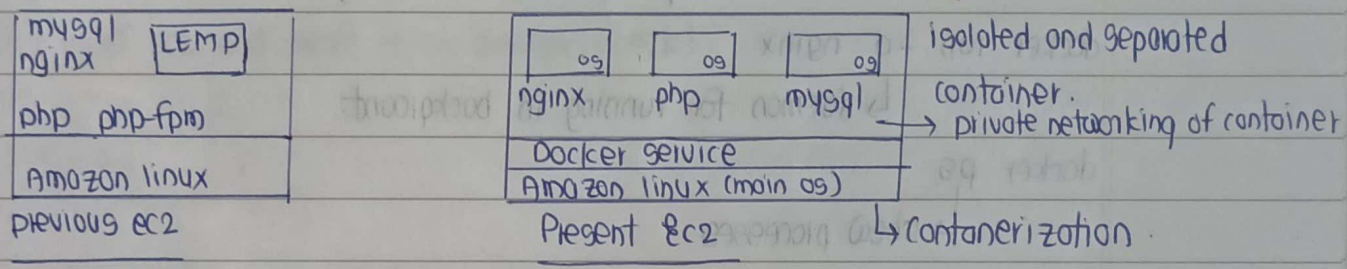
IPv4 - IP version 4 ⇒ 429, 49, 67, 296 Total IP in decimal IP 32 bit.

IPv6 - IP version 6 ⇒ 128 bit hexadecimal (0-9 a-f combination)

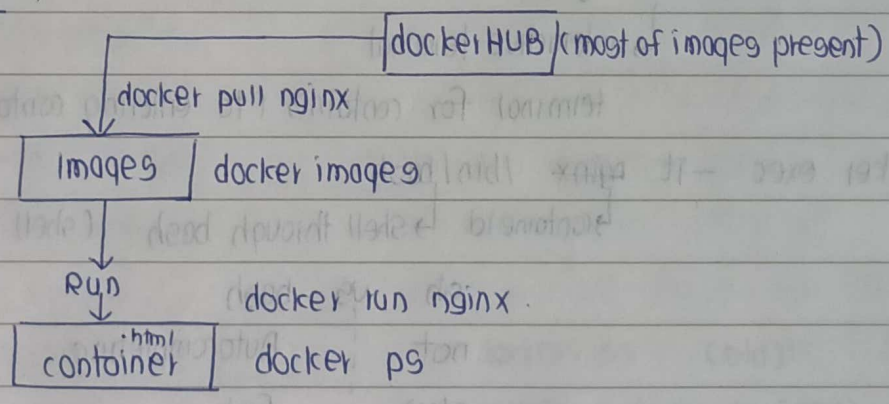
Day-03 Docker

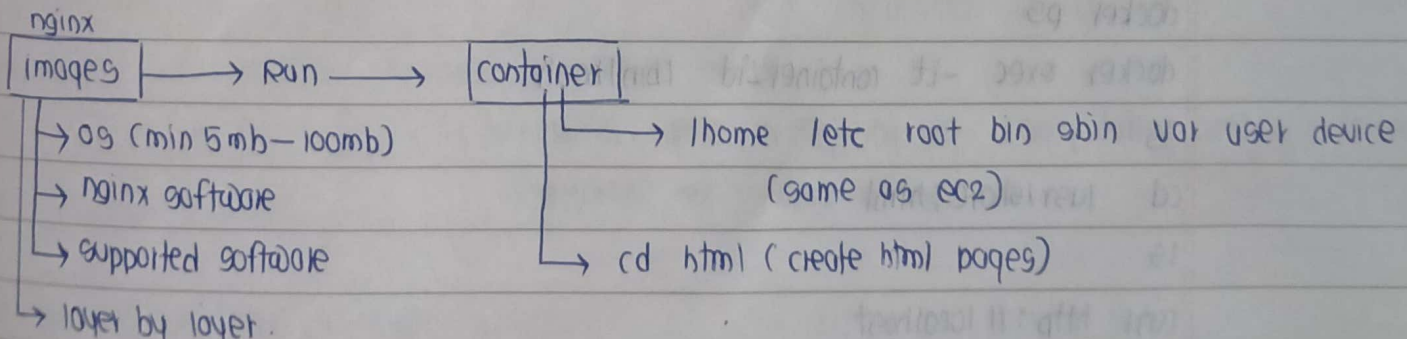
Docker
- create container
- manage container
- Deploy containers

sudo yum install docker
sudo service docker start
sudo service docker status.



Creating containers ⇒





sudo yum install docker -y

sudo service docker start

sudo service docker status

sudo su

docker pull nginx

apt update

docker images
↳ checking images are downloaded or not.

Repository Tag image ID created size

docker run nginx
↳ create actually containers in running

docker run -d nginx
↳ daemon for running in background

docker ps
↳ show processes

container id images command created status Ports Names

docker run -d -it nginx
↳ interactive terminal

terminal for container (to entering container)

docker exec -it ~~nginx~~ /bin/bash
↳ containe id ↳ shell through bash (shell means commands executing

sh	vs	bash
not		Autocompleting
slow		fast

docker ps

docker exec -it container_id /bin/bash

ls

cd /usr/share/html

ls

curl http://localhost

apt update

apt install nano -y

nano ravi.html

curl http://localhost/ravi.html

exit

docker stop containe_id/name

docker ps → running container

docker ps -a

↳ all (shows all container (Terminated, stop))

docker rm container-id

docker ps -a

docker run -d -p80:80 nginx

↳ mapping (only during creating container)

why mapping → (request → ec2 Public ip → container Private ip)

docker ps

copy public ip of ec2 and paste it on browser check!

Day-04 Docker

Images stored in

- DockerHub (Docker)
- ECR (Aws)
- ACR (Azure)
- GCR (Google)
- Third party

docker images (old)] same docker image ls (new)

docker ps (old)] same docker container ls (new)

docker ran -d -p80:80 --nome mywebsite nginx

↳ container name putting by we. (only during creating container)

name should not rename of container.

docker stop container id / name (old)

docker container stop container-id (new)

docker stop $ (docker ps -q)

↳ stopping multiple container (q → container id)

docker rm $(docker ps -aq)
↳ removing stopping container at one time multiple container.
a - all q - id

docker run -d -p 80:80 mywebsite nginx
docker run -d -p80:80 otherwebsite nginx
↳ error - ec2 port number already mapped
Solution → docker run -d -p90:80 otherwebsite nginx
↳ http://ip:90

docker run
↳ docker pull
↳ docker create
↳ docker start
} (Automatically fired command in background)

docker create
↳ only create container from image (container in stop condition)
↳ docker pull
↳ docker create
} (fired in background)

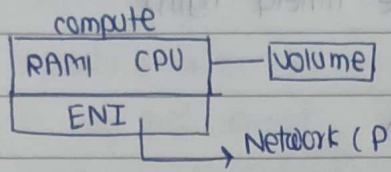
docker start
↳ starting stopped container
↳ starting created container

diffn betn docker start and docker run ?
diffn betn docker create and docker start ?

images in DockerHub
↳ Docker official images
↳ Verified Publisher (Public but owned by owner)
↳ Sponsored OSS (Private)

docker run -d httpd
docker ps
docker logs contain_name
↳ seeing logs of specified container. (docker debugging)
↳ for troubleshooting

container



docker inspect containe-name
- ↳ for troubleshooting
- ↳ provide all information about container

docker rmi image-id or docker image rm image-id
- ↳ removing image

docker image prune
- ↳ removing image all image at one time
  - condition if image is running not remove. (only remove unused image)
  - if we want forcefully delete used -f

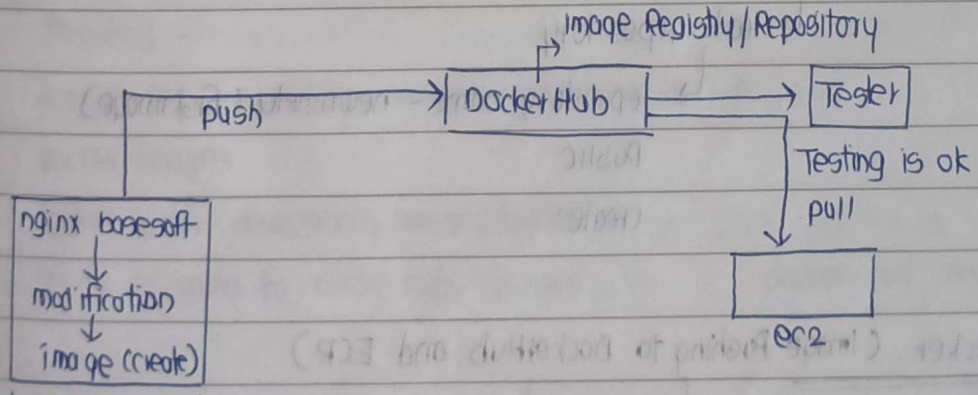
docker create httpd
- ↳ first check image is present in localmachine

docker start httpdcontainername

docker run -d -p80:80 --name apachecola httpd

docker ps

copy public ip and checked.



docker commit containername newimagebackup
- ↳ for creating backup/image of container

docker run -d -p90:80 --name ninety nginx

docker ps

docker exec -it ninety /bin/bash

apt update

apt install nano

rm index.html

cd /usr/share/nginx/html

rm index.html

nano index.html

nano navi.html

exit

docker commit ninety newninety:1.0

↳ creating images

docker images

docker run -d -p93:80 --name three newninety

docker ps -a

publicip:93 (adding 93 in a security group ec2)

creating account in DockerHub

Pushing images to DockerHub

↳ DockerHub

↳ Repositories

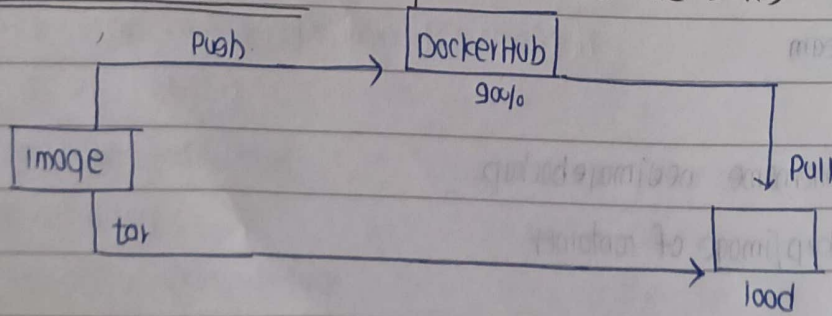
↳ create repository

↳ repository name - newninety (first image)

Public

create

Day-05 Docker. (Image Pushing to DockerHub and ECR)



Pushing images in DockerHub ⇒

↳ create repository in DockerHub Devopsbaby/mywebsite

↳ logging in our Aws account

docker login -u devopsbaby

Password - (Personal access token)

↳ why - for security purpose we can not give him to Password of dockerHub.

↳ docker tag mywebsite devopsbaby/mywebsite

↳ local image —rename→ ↳ DockerHub uploading image ↳ creating new image with repository name for pushing to DockerHub

↳ docker push devopsbaby/mywebsite

↳ uploading image towards DockerHub

Protocol →

DockerHub

↳ Account setting

↳ Personal Access Token

↳ Generate New Token

↳ name Read & write Generate.

↳ copy Personal Access token in our system.

docker login -u devopsbaby

Password -

docker tag neoninety devopsbaby/mynewsite:1.0

docker images

docker push devopsbaby/mynewsite:1.0

Verify to going to DockerHub Account. docker pull devopsbaby/mywebsite:1.0

image → tar → tar → image docker load

docker save

docker save -o myweb.tar nginx (o → output)

ls ↳ tar file name ↳ image name.

downloading to laptop scp

docker load -i myweb.tar → image loading through tar. (i - input)

ECR
↳ Elastic container registory
Paid

Aws account (creating repository in ECR)
↳ ECR
↳ create repository
↳ name- mywebsite
↳ create
↳ select repository
↳ view push commands

logging to ec2 to aws account
↳ aws configure
Access key -
Secret key -
region name- ap-south-1
faimat- json
aws s3 ls
copy and paste ecr logging command
copy and paste tag command change image name -newninety
docker images
docker push image_name

mysql image ⟶ container
↳
docker run -d -e MYSQL-ROOT-PASSWORD=
Pass@123 mysql

Parameter ⇒
-d → --daemon --name
-p → -- publish
-e → -- environment
↳ environment variable

docker ps
docker exec -it mysqlcontainerid /bin/bash
mysql -u root -p
Password -
show databases;

Day-06 Docker (Docker volume, wordpress,)

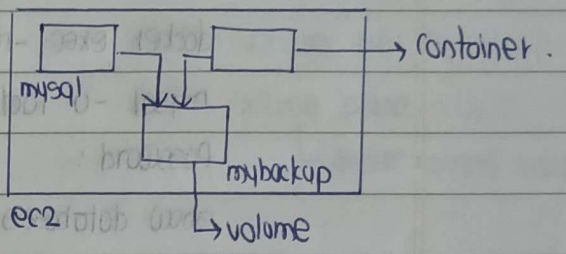
Docker volumes
- 1) Host Path (Bind mount)
- 2) Named volumes
- 3) Anonymous volumes

1) Host Path (Bind mount)

why used volume ⇒

we used volume as backup of container data

It is mouted to ec2 volume or anothe volume for backup.



If container crashed our backup data is stored to volume.

```
docker run -d --name majhosql -v /home/ec2-user/mybackup:/var/lib/mysql mysql
```

↳ used to create new volume with attached to new volume as backup., like mouting

Protical

↳ sudo su

```
sudo service docker start
mkdir mybackup
docker run -d --name majhosql -v /home/ec2-user/mybackup:/var/lib/mysql
    -e MYSQL_ROOT_PASSWORD=Pass@123
docker ps
docker exec -it majhosql /bin/bash
mysql -u root -p
Password:
create database facebook;
use facebook;
create table posts (id int, name varchar(100));
insert into posts values (1, "ravi");
exit;
exit
cd mybackup/
ls
```

```
docker stop majhosql
docker rm majhosql
docker ps -a
cd mybackup/
ls          → our backup not affected if container is crashed
docker run -d --name tajhosql -v /home/ec2-user/mybackup/:/var/lib/mysql
    -e MYSQL_ROOT_PASSWORD=Pass@123
docker ps
docker exec -it tajhosql /bin/bash
mysql -u root -p
Password:
show databases;
select * from posts;
exit;
exit
```

2) Named volume ⇒

```
docker volume create myvol
                ↳ Here docker create volume in path /var/lib/docker/volume/data
docker manage these volume
```

Practical

```
↳ sudo su
cd /var/lib/docker/volumes/
ls
cd
docker volume create ourmysqlvolume
                ↳ creating volume by using docker
cd /var/lib/docker/volumes
ls
cd ourmysqlvolume
ls
docker run -d -v ourmysqlvolume:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=Pass@...
docker ps
```

docker volume ls
↳ seeing list of volume

docker run -d --name majbasql1 -v ourSqlvolume : /var/lib/mysql
-e MYSQL_ROOT_PASSWORD = Pass@123

docker exec -it majbasql1 /bin/bash

mysql -u root -p

Password =

create database facebook;

show databases;

exit;

docker stop majbasql1

docker rm majbasql1

cd /var/lib/docker/volumes

ls

cd ourSqlvolume

ls
↳ see our database is not destroyed.

docker volume inspect myvol

docker volume rm myvol

docker volume prune
↳ delete unused volume.

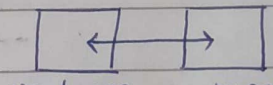
3) Anonymous volume

In this volume we not give named to volume, so this is called Anonymous

cd /var/lib/docker/volumes

docker run -d -v /var/lib/mysql -e MYSQL_ROOT_PASSWORD = Pass@123
↳ folder was created with anonymous name.

Wordpress with container ⇒



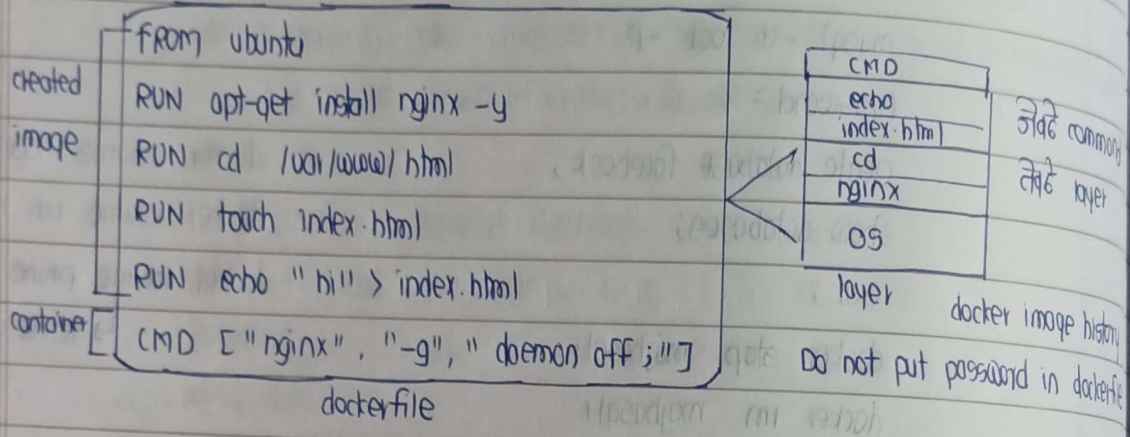
1) docker run -d -e MYSQL_ROOT_PASSWORD = Pass@123 -e MYSQL_DATABASE = wordpressdb
--name majbasql mysql

2) docker run -d -p80:80 -e WORDPRESS_DB_HOST = majbasql (contain-private-ip)
-e WORDPRESS_DATABASE = wordpressdb -e WORDPRESS_DB_username = root
-e WORDPRESS_DB_PASS = Pass@123 --link majbasql : mysql wordpress
↳ container name ↳ img name

docker file =>

writing some command in file and then executing this file, then and then docker image is created.

In Docker we work as writing Docker file. → work of Devops engineer



docker build .

docker images

Day-07 Docker (wordpress, Dockerfile)

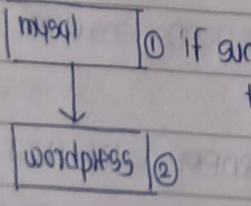
Wordpress with help of container.

docker run -d --name wpsql -e MY-SQL-ROOT-PASSWORD=Pass@123
-e MYSQL-DATABASE=wordpressdb -v /home/ec2-user/wordpressbackup:/var/lib/sql mysql

docker ps

docker run -d -p80:80 -e WORDPRESS-DB-HOST=wpsql (↳ container name)
-e WORDPRESS-DB-USER=root (↳ user name) -e WORDPRESS-DB-NAME=wordpressdb
-e WORDPRESS-DB-PASSWORD=Pass@123
--link wpsql:mysql --name wordpress wordpress

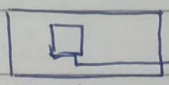
copy public-ip and check.



Dockerfile ⇒

↳ creating our own docker image.

EC2-user



docker build .

docker build -t myimg .

docker build -t myimg:1.0 .

docker build -t myimg:beta .

} for giving name to image

docker build -f myfile .

docker build -f myfile -t myimg .

} for changed name.

```
FROM ubuntu
RUN apt update ~~install~~ -y
RUN apt install nginx -y
RUN cd /var/www/html
RUN rm index.html
RUN touch index.html
RUN echo "hi ..." >index.html
CMD ["nginx","-g","daemon off;"]
```

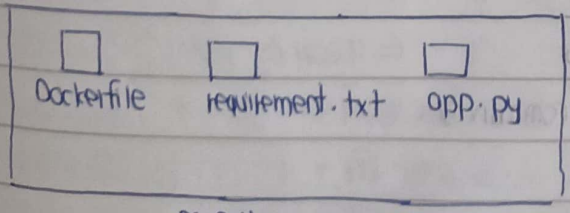
Dockerfile

docker build .

Day-08 Docker (Pythonapp with dockerfile)

Python website

- → Flask framework
- → pip
- → requirement.txt



MyPython App

mkdir pythonapp

cd pythonapp

nano Dockerfile

```
FROM python:3.9-slim
WORKDIR /app
COPY requirements.txt /app/.
RUN pip install -r requirements.txt
COPY . /app/
EXPOSE 5000
CMD ["python", "app.py"]
```

ls

nano requirements.txt

↳ required software

nano app.py

↳ Python code

ls

sudo su

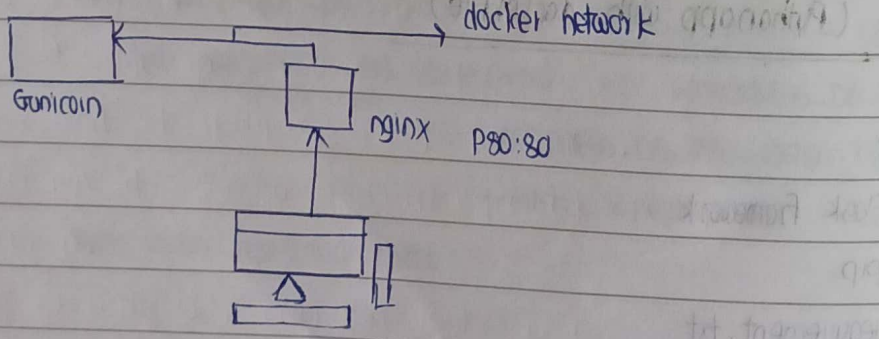
docker build -t mypythonimg .

docker images

docker run -d -p80:5000 --name pythonwala mypythonimg

docker ps

copy public ip and Test it.



docker exec -it container_name commands

↳ only for one commands.

```
sudo su
docker run -d nginx
docker exec -it container-id cat /etc/nginx/conf.d/default.conf
        ↳ copy nginx conf page code
cd
cd /home/ec2-user/mypythonapp/
ls
nano default.conf
        ↳ Add in location  proxy-pass  http://container_Pythonapp_ip:5000;
ls
nano nginxdockerfile
        ↳ FROM nginx
          COPY default.conf /etc/nginx/conf.d/
          EXPOSE 80
          CMD ["nginx", "-g", "daemon off;"]
docker build -t majhanginximg -f nginxdockerfile .
docker images
docker rm $(docker ps -aq)
```

Docker Network ⇒

- 1) bridge (default)
- 2) HOST
- 3) NoNe

1) bridge Network ⇒

- Container च्या आतल्या आत मध्ये केलेले communication
- जेव्हा network तयार केला जातो, त्या Network मध्ये जेवढे container असतील तेवढ्याच container ला एकमेकांशी communication करता येईल.
- जर मी two bridge network तयार केले तर एक network मधल्या container ला दुसऱ्या network मधल्या container शी communication करता येत नाही.
- This Network is used for security.

2) HOST ⇒

- Host मध्ये container चा Network च असते.
- ती ec2 च्याच network सोबत map होतो.
- येथे mapping ची गरज नसते.

docker network create mynetwork

⇛ None Network ⇒

જેટલા આવવાવાળા container એ કોઈપણ રીતે communication નહિ કરશે.

Creating Network ⇒

↳ docker network create -d Host/None

docker network create mynetwork
↳ bridg network

docker run --network mynetwork
↳ during creating container

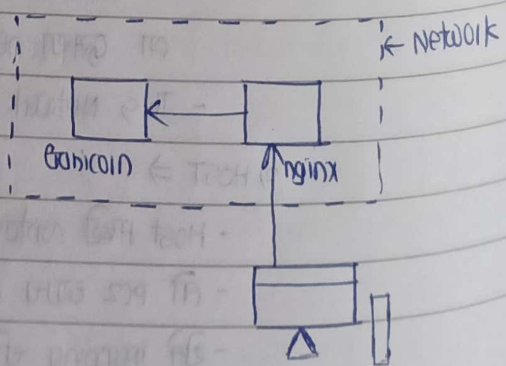
docker network connect mynetwork mypythonwala
↳ if container is already exit

docker network disconnect mynetwork mypythonwala
↳ disconnecting from network.

docker network create mynetwork
docker network ls
docker run -d --network mynetwork --name mypythonapp mypythonimg
docker ps
docker run -d -p80:80 --network mynetwork ✗
docker inspect python_container_id
↳ copy ip

ls
nano default.conf
↳ change ip

docker build -t newnginx -f nginxdockerfile .
docker run -d -p80:80 --network mynetwork newnginx
docker ps
copy ip (Public ip of ec2) and Test.



Day-09 Docker. (Docker Network, Nodejs)

Docker Network ⇒

```
docker network create mynet
docker run --network mynet
docker network connect mynet container_id

docker network ls
docker network inspect network_name
                        ↳ All about network.
docker images
docker run -d -- name pythonwala mypythonimg
docker ps
docker network ls
docker network inspect bridge
                ↳ see container
docker ps
docker inspect container_name
                ↳ See Network id
docker network connect mynetwork pythonwala
docker network inspect mynetwork
docker run -d -p80:80 --name nginxwala newnginx
docker ps
docker network connect mynetwork nginxwala
docker network inspect mynetwork
copy public_ip of ec2 and Test.
docker network disconnect mynetwork pythonwala
docker network inspect mynetwork.
docker network rm mynetwork
                ↳ remove network

docker network create yournet --subnet 10.0.0.0/16
                                        ↳ giving CIDR block
docker network ls
docker network inspect yournet
```

docker network connect yournet pythonbala

docker network inspect yournet
↳ see ip in containers section.

docker stop $(docker ps -q)

docker rm $(docker ps -aq)

docker run -d --network host nginx
↳ used host network.

docker ps

copy public-ip and Test

docker network ls

docker stop $(docker ps -q)

docker rm $(docker ps -aq)

Nodejs ⇒

sudo su

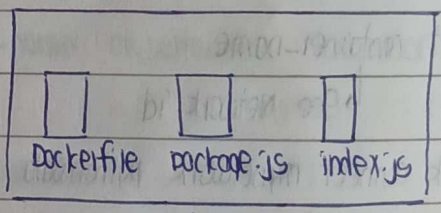
mkdir mynodeapp

cd mynodeapp/

service docker start

nano Dockerfile
↳

```
FROM node
WORKDIR /usr/src/app
COPY package.json .
COPY . .
RUN npm install
EXPOSE 3000
CMD ["node", "index.js"]
```



nano package.js

nano index.js

ls

docker build -t mynodeimg .

docker images

docker ps

docker stop $(docker ps -q)

docker rm $(docker ps -aq)

docker run -d -p80:3000 mynodeimg
↳ without proxy

docker ps

copy ip and Test.

for proxy →

copy nginxdockerfile and paste to mynodeapp

Also copy default.conf and paste to mynodeapp

sudo nano default.conf

↳ change ip copy node-container ip and put in this

:3000

sudo nano nginxdockerfile

docker stop $(docker ps -q)

docker rm $(docker ps -aq)

docker run -d --name nodewala mynodeimg

docker inspect nodewala

↳ check ip

docker build -t nodenginx -f nginxdockerfile .

docker ps

docker stop nodenginx-container

docker rm nodenginx-container

docker run -d -p80:80 --name nginxnode nodenginx

docker ps

copy ip and Test.

Day-10 Docker (mysql with Dockerfile)

mysql with help of docker file ⇒

```
FROM mysql
ENV MYSQL_ROOT_PASSWORD = Pass@123
ENV MYSQL_DATABASE = facebook
EXPOSE 3306
CMD ["mysqld"]
```

sudo su

mkdir mysqlwala

cd mysqlwala

nano Dokerfile

ls

docker build -t mysqling .

docker images

docker run -d --mysqlwala --name mysqling

↳ without giving environment variable, environment variable is get in during building images in dockerfile.

docker exec -it mysqlwala /bin/bash

mysql -u root -p

Pass@123

exit

docker run -d --name mysqlwala1 -e MYSQL_ROOT_PASSWORD = ravi@123 mysqling

↳ Overwriting environment variable.

whenever creating container, environment variable overwrite during container.

when we entering database used this password.

docker ps

docker exec -it mysqlwala1 /bin/bash

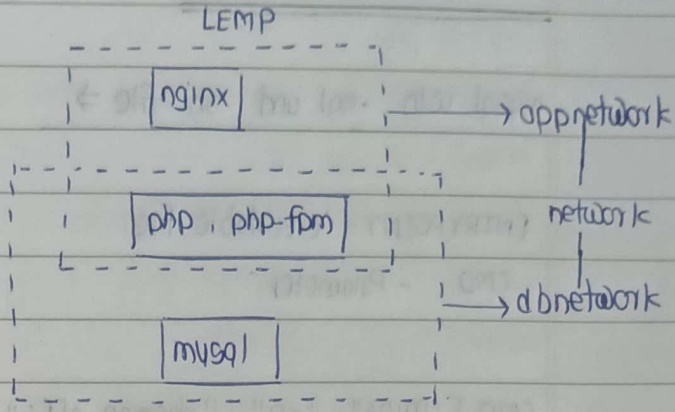
mysql -u root -p

ravi@123

exit;

Docker compose file:→

three-tier architecture with 3 container



Used to define and manage multi-container docker application in simplified, declarative manner.

run multiple container with help of single docker compose file.

```
version : 3.0
services : web
    image : nginx
    network : appnetwork    depends on :
services : app
    image : php-fpm
    network : - appnetwork
              - dbnetwork
services : db
    image : mysql
    network : dbnetwork
    volume : myvol
Volume :
    myvol :
network :
  - appnetwork
  - dbnetwork
```

↳ Docker compose file
↳ write in yml
↳ yml is space sensitive language.

Day-11 Docker (mysql Dockerfile with .sh & .sql , shell scripting nginx)

mysql with .sql and .sh file ⇒

ENTRYPOINT - executable file

CMD - Parameter

CMD ["nginx", "-g", "daemon off;"] ⟶ ENTRYPOINT ["nginx"]

CMD ["python", "my.py"] ⟶ CMD ["-g", "daemon off;"]

CMD ["node", "index.js"] ⟶ ENTRYPOINT ["python"]

CMD ["my.py"]

ENTRYPOINT ["node"]

CMD ["index.js"]

```
sudo su
cd mysqlwala
nano Dokerfile
        ↳ remove database
            Add   copy  init.sql  /docker-entrypoint-initdb.d/
                  copy  init.sh   /docker-entrypoint-initdb.d/
nano init.sql
    ↳ create database insta;
        use insta;
        create table users (id int, name varchar(20));
        insert into users values (1, "ravi"),(2, "aliya");
nano init.sh
    ↳ echo " i am inside shell script file"
ls
docker build -t mysqlimg .
docker images
docker run -d --name mysqlwala mysqlimg
docker ps
docker exec -it mysqlwala /bin/bash
```

mysql -u root -p

Password:

> use insta;

select * from users;

exit;

Donot write password in Dockerfile

solution → Argument (ARG)

→ Aws secreate manager

Custom scripting file with nginx ⇒

mkdir customsh

cd customsh

nano Dockerfile

```
FROM nginx
COPY custom.sh /usr/local/bin/custom.sh
RUN chmod +x /usr/local/bin/custom.sh
EXPOSE 80
ENTRYPOINT ["/bin/bash", "-c", "/usr/local/bin/custom.sh && nginx -g 'daemon off;'"]
```

nano custom.sh

```
cd /usr/share/nginx/html
rm index.html
touch index.html
echo "i am in nginx container" > index.html
```

docker build -t mynginxing .

docker run -d -p80:80 --name nginxcola mynginxing

nano Dockerfile

```
ARG Password
ENV MY-SQL-ROOT-PASSWORD = ${Password}
```

docker build --build-arg Password=Pass@123 -t argsqling .

docker images

docker run -d --name argoola argsal:img

docker ps

docker exec -it argoola /bin/bash

mysql -u root -p

Password: Pass@123

exit

exit

docker history argsal:img

↳ seeing all layers (included password)

solution → multistage building

ADD

↳ ADD is same work as copy

but it copy from another website also ADD url

FROM	LABEL
RUN	MAINTAINER
WORKDIR	HEALTHCHECK
CMD	USER
ONBUILD	VOLUME
ENTRYPOINT	
ARG	
COPY	
ADD	
ENV	
EXPOSE	

↳ most used dockerfile command

Day-12 Docker (multistage- building, wordpress Docker-compose file)

Multi stage- Building ⇒

↳ Hiding sensitive information from image (Hide history)
Reduced docker file size. (copy only executable files)

sudo su
service docker start.
mkdir multistagewala
cd multistagewala
nano Dockerfile

↳

```
FROM node AS build
WORKDIR /usr/src/app
COPY package.json .
COPY . .
RUN npm install
CMD ["node", "index.js"]

FROM node AS production
WORKDIR /usr/src/app
COPY --from=build /usr/src/app ./dist
COPY --from=build /usr/src/app/package*.json ./
RUN npm install
EXPOSE 3000
CMD ["node", "dist/index.js"]
```

nano package.json
nano index.js
copy IP and Test.

Docker compose file ⇒

↳ for creating container more than one and whenever this container needs networks.

Wordpress with Dockercomposefile ⇒

sudo su

curl -SL https: ____

↳ copy url from dockerHub

Downloading Docker compose

chmod +x /usr/local/bin/docker-compose

docker-compose --version

mkdir composewala cd composewala

nano wpwala.yml

```
version : '3.9'
services :
    db :
        image : mysql
        ports :
            - 3306
        volumes
            - myvol : /var/lib/sql
        networks :
            - mynet
        environment :
            - MYSQL_ROOT_PASSWORD = Pass@123
            - MYSQL_DATABASE = wordpressdb
    website :
        image : wordpress
        ports :
            - 80 : 80
        networks :
            - mynet
        environment :
            - WORDPRESS_DB_HOST = db
            - WORDPRESS_DB_USER = root
            - WORDPRESS_DB_PASSWORD = Pass@123
```

```
      - WORDPRESS_DB_NAME= wordpressdb
networks:
  mynet:
    driver: bridge
volumes:
  myvol:
```

docker-compose -f wpwala.yml up

Day-13 Docker (Docker-compose three tier architecture.)

cd composewala

docker-compose -f wpwala.yml up -d

↳ in background

docker-compose -f wpwala.yml ps

docker ps

Copy IP and Test.

Three Tier architecture with Docker-compose-file ⇒

docker-compose -f wpwala.yml down

↳ To removing container, networks stopping

when we file name write docker-compose.yml we did not write a file name during up docker file. docker-compose up -d

Source code changes - directory changed in live server ⇒

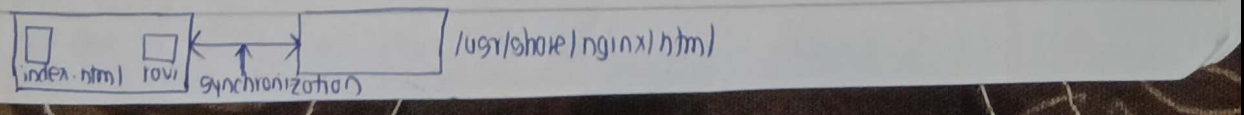
for this we used volume mapped and folder contain source code.

mkdir mywebsite cd mywebsite

nano index.html

nano ravi.html

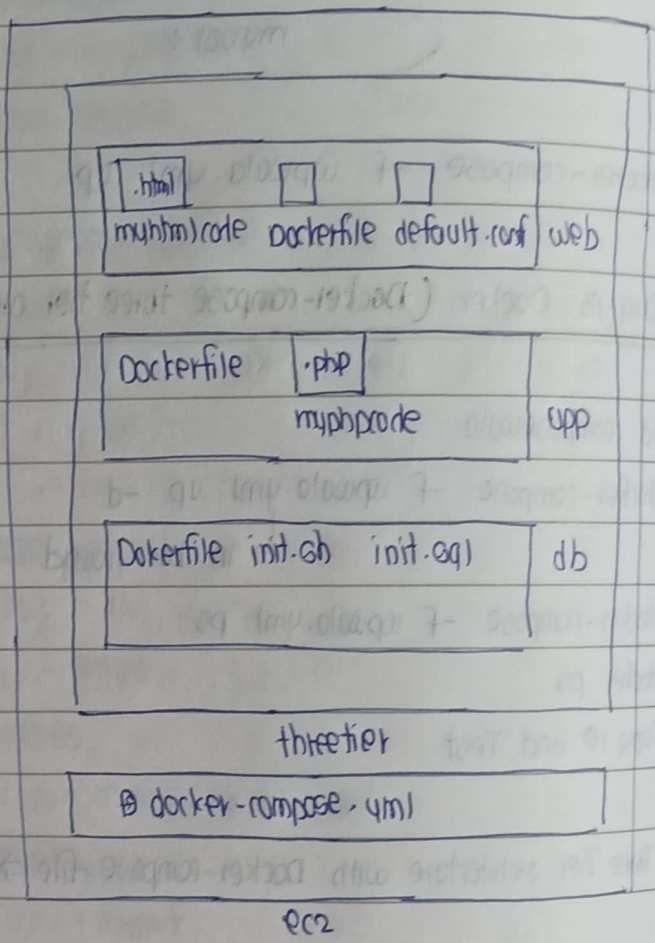
docker run -d -p80:80 -v /home/ec2-user/mywebsite:/usr/share/nginx/html nginx



Day-14 Docker

Three-tier architecture with docker_compose file

```
mkdir threetier
cd threetier
mkdir opp web db
cd web
nano Dockerfile
nano index.html
nano default.conf
cd..
cd appl
nano Dokerfile
nano index.php x
mkdir myphpcode
cd myphpcode
nano index.php
cd..|..
cd db
nano Dokerfile
nano init.sh
nano init.sql
```



Day-15 kubernetes (why kubernetes, Architecture of kubernetes)

```
docker run --cpu 0.5 --memory 512mb -d
```

↳ defining cpu and memory to container

kubernetes is very easy

↳ by google

why kubernetes ⇒

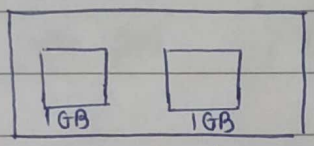
1) Auto restart of container

In this kubernetes help to auto restarting container, when container is crashed / exited state or if container is okay, but in under container web server is not working well then auto start these container.

2) healthcheck

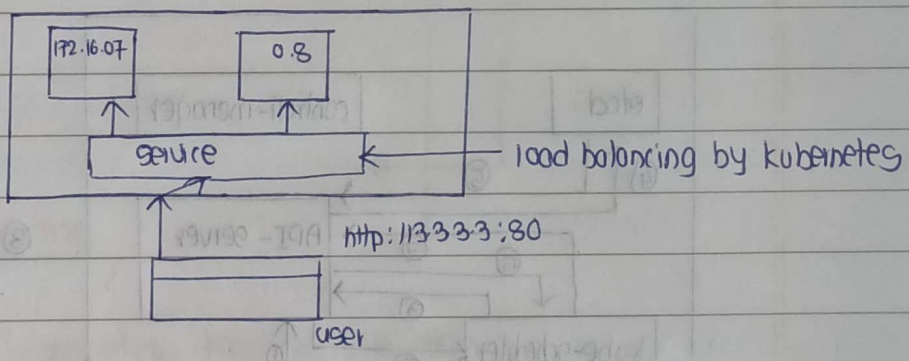
kubernetes also functionality of healthcheck, that helps checking web server working or not by kubernetes. (if nginx is not working)

3) Autoscaling

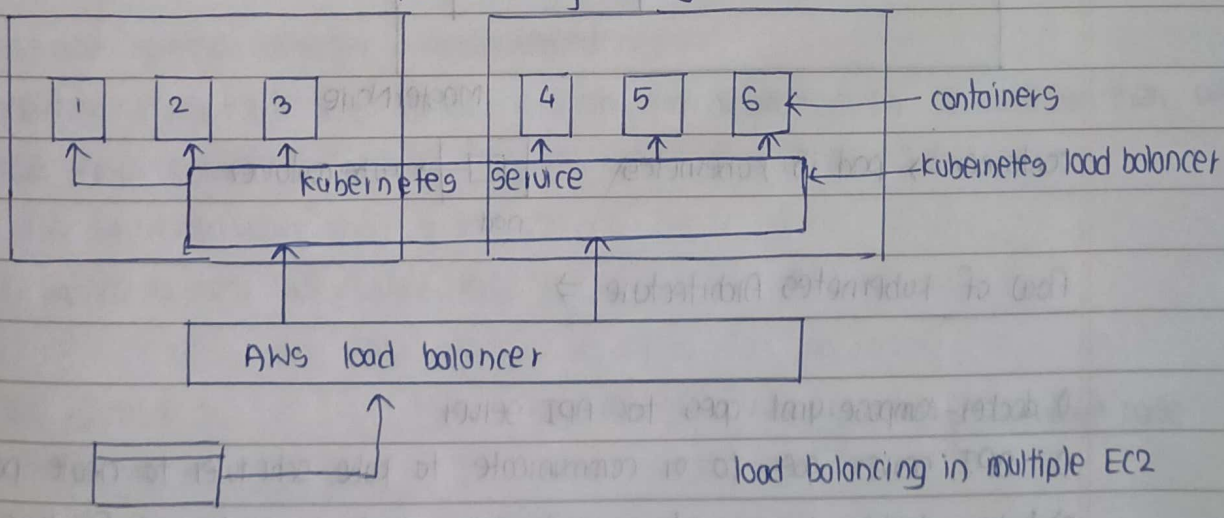


kubernetes Autoscaled container if container ram is fulled, then kubernetes scaling container

4) Load balancing



load balancing in single ec2



load balancing in multiple EC2

docker run --cpu 0.25 --memory 512 mb -d

↳ in docker

request 0.25 (min)

limit 0.5 (max) → in kubernetes

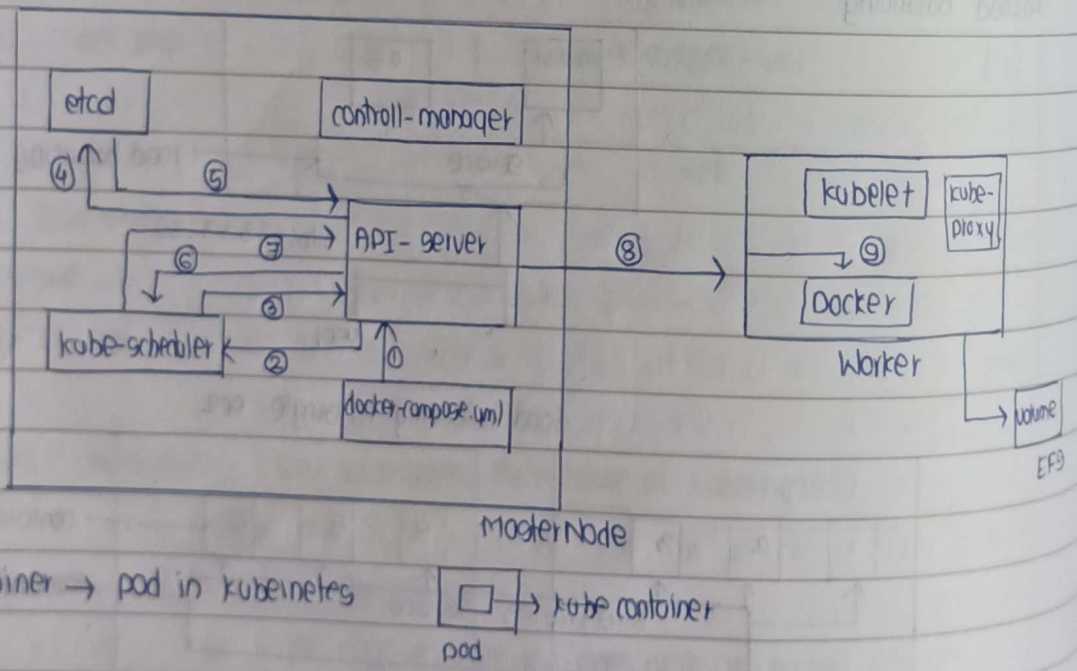
ECS → elastic container service

EKS → elastic kubernetes service

Why kubernetes ⇒

1) Auto start of container
2) healthcheck
3) Autoscaling
4) load balancing
5) Network
6) Auto - Healing
7) Auto Node Allocating
8) updates/new release / deployment
9) secreate / config (master node)

Kubernetes Architecture ⇒ Component of K8's ⇒



container → pod in kubernetes

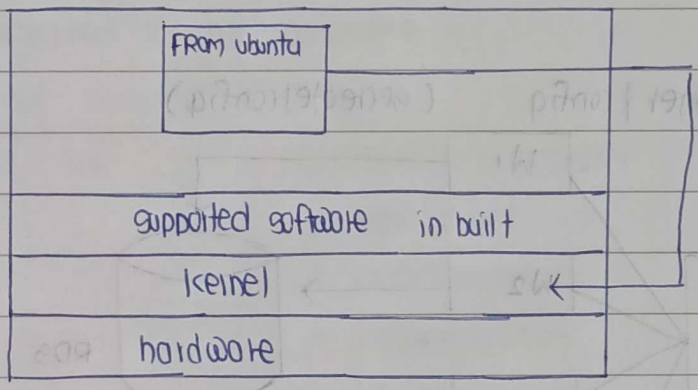
flow of kubernetes Architecture ⇒

1) docker-compose.yml goes to API server
2) API server goes to or communicate to kube-scheduler to create pod
3) kube-scheduler has no data in which worker empty to create pod for that communicate

4) API-server communicate to etcd and get data
5) etcd send data to API server.
6) API server data to kube-schedular.
7) kub-schedular decide in which worker create pod these send to API server
8) API server send to worker in worker through kubelet
9) kubelet create pod through Docker image file.
Hence our pod will create.

etcd -: no-sql non relational json
control-manager ⇒ control manager managed the pod, pod is crashed or dishoyed or not responding this information sends the API server and then create new pod.

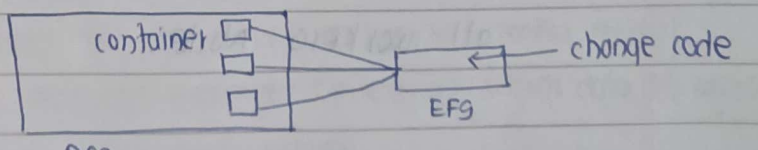
container is used host os and container os what is difference in os ?
⇒



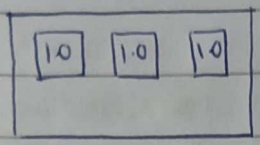
Day-16 kubernetes (why kubernetes, kubernetes objects)

8) Updates | new release manager. | deployment.
if we have bug in our containers or any modification in containers then we release new update for new features. for that two ways
1) by volume
2) with help of kubernetes

1) by volume ⇒
disadvantage → get more time to update code

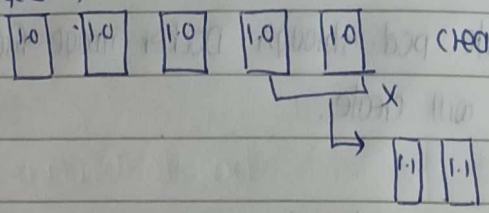


2) by kubernetes ⇒



change to 1.1

how kubernetes changes ⇒ थोड़े थोड़े करके तेज करके container create करते.



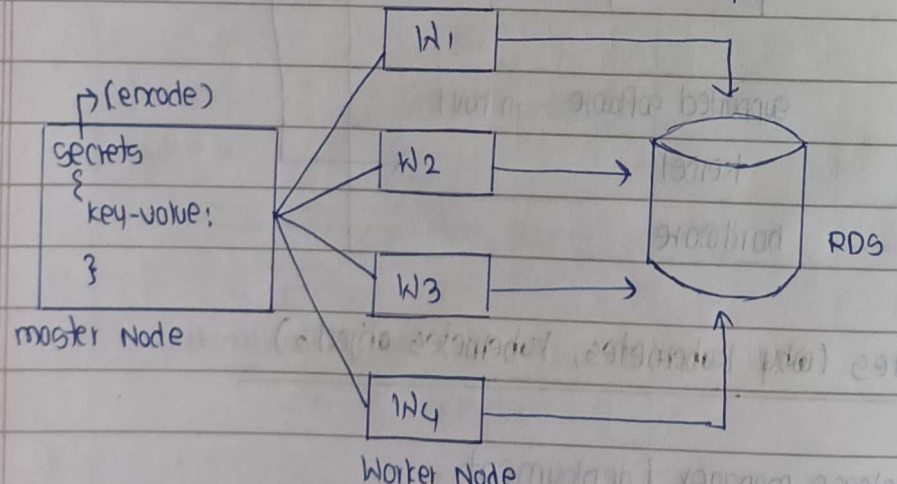
Roollback ⇒

If we new release manager and these are not working propery then rollback functionality is used in kubernetes that rollback to our previous up stable update's.

what is version ⇒

only changes in source code or software (software versions)

9) Password manager / config (secreate/config)



If hacker knows my RDS password then how we change from our containers at one time?

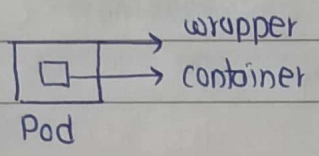
→ first change RDS password and then this password put in master node under that secreate (section key-value change Password then this password is send to worker node and changes automatically to all workers Node.

Also we can changes config in master node and this changes will change to all workers node

kubernetes objects ⇒

- 1) Pod
- 2) Service
- 3) volume
- 4) Network
- 5) configmap
- 6) secreate
- 7) Ingress

1) Pod ⇒



Why pod ?

pod, under pod we can put any container, any container means different software created containers.

pod supported to all containers.

Types of pods ⇒

- 1) Pod
- 2) replication controller
- 3) replica set
- 4) Deployment
- 5) statefull set
- 6) Daemon set

Pod → simple pod, wickness pod do not autostart containers.

replication controller → if pod can crashed, then they create automatically new

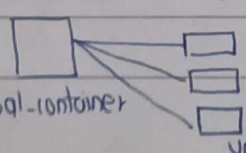
replica set → they also start new pod, in short query we can see count or details of pod in list.

weakness in replication controller and replica set they can not support to update/release manager and rollback.

Deployment → Autostart, update/rollback, load balancing

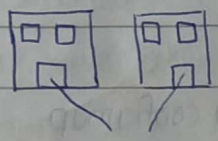
this will use in Industries

statefull set → for using for state level container mostly mysql



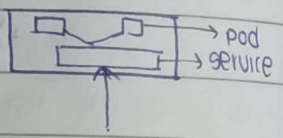
they can remember (indexing) which data is store in which volume.

daemon set → in all worker, each worker create container and this is used for logs



2) service

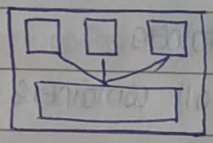
connectivity with from users/client and load balancing



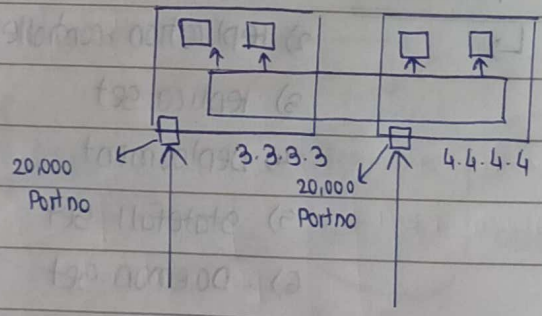
Types ⇒ 1) cluster IP
2) Node Port
3) Load balancer

cluster IP →

load balancing only under ec2, they are not connect to other world



Node Port →



load balancer → most used (mostly used)

3) Ingress

for domain name assigning
for multiple services.

Q) In one pod, can we stored multiple containers?
→ Yes, this will not microservice architecture

```
apiversion : apps/v1          → kubernetes files (manifest file)
kind       : Deployment       → kubernetes object
Metadata :                    → information about deployment
    name : nginxdeployment
    labels :
        apps : nginx          → To finding deployment
spec :
    replicas : 2              → two create pod
    selectors :
        MatchLabels :
            app : nginx       → to find pod
    template :
        metadata :
            labels :
                app : nginx   → To finding pod
spec :
    containers :
        - name : nginx        → container (may be more)
          image : nginx
          ports :
            - containerPort : 80
```

.yml file

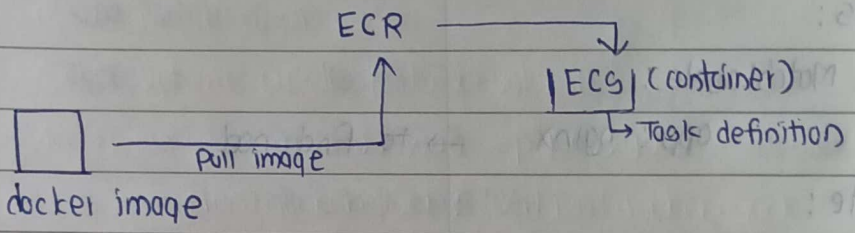
```
apiversion :
kind : service
metadata :
    name : myservice
    labels :
        app : nginx
spec :
    selector :
        app : nginx           → mapped to pod
    Ports :
    - protocol : TCP      targetport : 80
      port     : 80       type :  nodeport
```

Day-17 ECS

elastic container Selvice

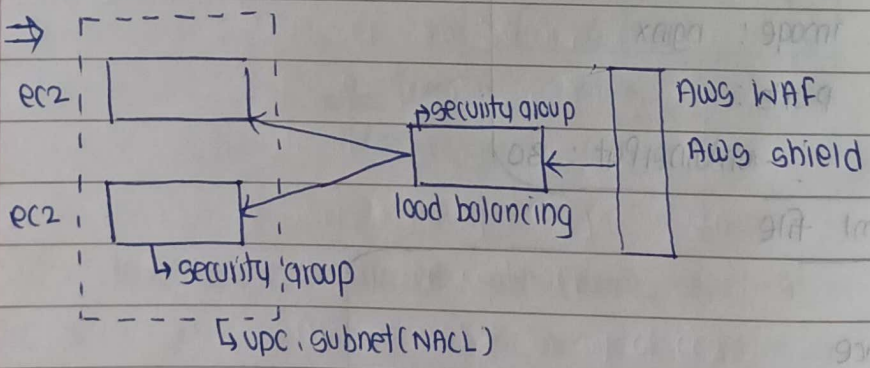
what is difference betn Docker and ECS ?

Docker	ECS
No Autoscoling	Autoscoling
No load balancing	load balancing



Task definition
↳ 1) Task (not always (stopped if work stop))
2) Selvices (24x7) (web application, database)

How secure a system ?



AWS WAF
↳ web application firewall
↳ spyware, sql injection
malware, Scripting Attack

Aws shield
↳ DOS, DDOS

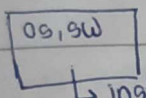
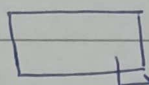
vpc
↳ flow logs used three-tier architecture.

ECS launch Type

↳ 1) EC2 (infrastructe mode us)

2) fargate

↳ serverless

Infrastructe	serverless
 ↳ installed us	 ↳ os/software managed by aws
	- if request has comes then containers launch
	- bill depends on request
	- if request is gone then containers remove for CPU and ram used for another containers.
	- response time is less than ec2.

Protcol %

ECS

↳ create cluster

↳ mycluster Aws fargate → create

under mycluster

↳ service

↳ create

compute option - launch type

Task definition

↳ Create new task definition

mytaskdef

1CPU 2GB

Container 1

Name nginx Image uri → uri (copied from ECR

Port mapping

80 TCP name

Amout 21 → create

select task definition

Seivice name - myseivice → create copy IP from Task.

delete-service

- cluster

Day-18 Portainer

way of using
- 1) on ec2
- 2) laptop with docker-desktop

1) In EC2
- start ec2

docker image prun
docker volume create portainervolume
docker run -d -p8000:8000 -p9000:9000 --name portainerwala
-v /var/run/docker.sock:/var/run/docker.sock
-v portainervolume:/data portainer/portainer-ce:latest

open port no. 8000 and 9000 to our ec2
and search PublicIP:9000
- username - admin
- password - password1234
- container Add container
  - Name- puspha
  - image- httpd
  - Port mapping 90 80
  - deploy the container
  - Add 90 Port no in sg of ec2
  - container → exec (ioacon)
  - cd /usr/local/apache2/htdocs
  - touch puspa.html
  - echo "shrivali" > puspa.html

2) Laptop ⇒
- download docker desktop
  - windows AMD64
  - install restart.