# * devops *

→ It is a mashup of 2 word Development (Dev) and operationals (ops)

→ Devops speed delievary of Higher Quality S/w by combining and automating the work of S/w development & IT opes team

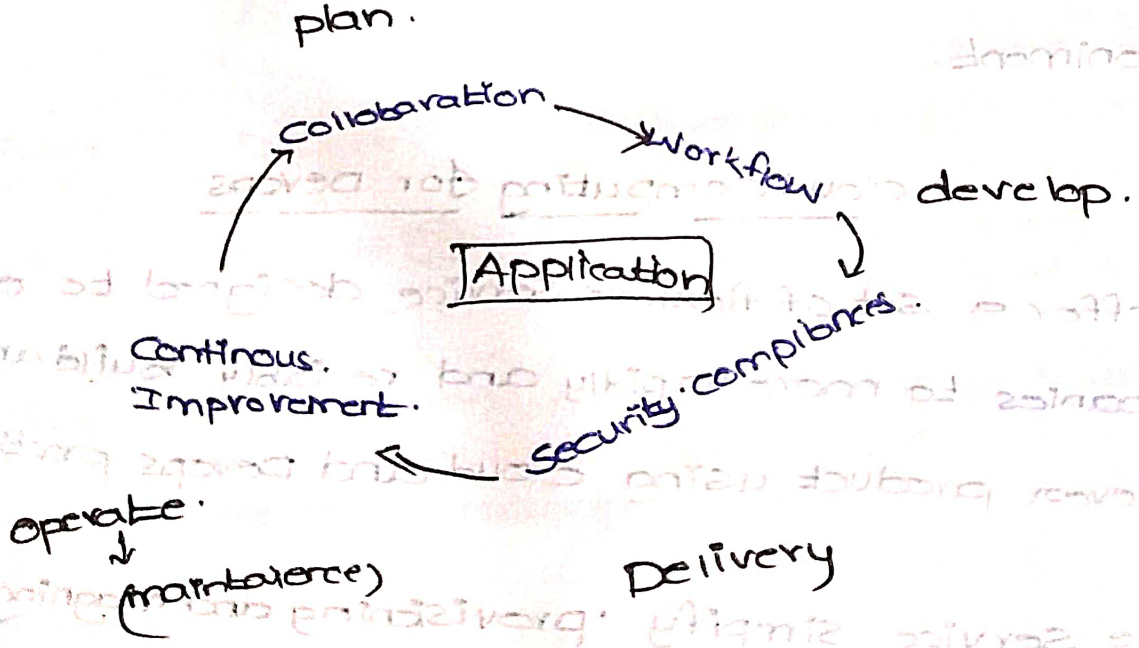
→ It help increase an organization speed to delievers applications and services.

**Benefits of devops:**

→ Speed. Reliability Improved collabaration.

→ Rapid delivary. Scale. Security.

**Devops lifecycle:-**




**plan:-** The team ideate, define and describe. features and capabilities of the applications and system they are buildings.

Cloud Watch : Monitoring Services → N/w speed, Read/write ops.
→ log files
→ Alarms

SNS : Event occurs, Notification

Could Trial : Auditing Services, → Everything is Recorded.

Ec2 Instance

* LightSail :-

*. Cloud front → CDN → frontend + backend,

→ CI/CD →

Image → pre-configured templated.

Snapshot → point-in-time backup. When migration is needed, then we need snapshot, [Need]

mastercopy

multiple copies.

OS + Additional S/w

* Snowball :- Service to migrate physical resource

data → terabytes
petabytes. → datasizes
megabytes

to one location to another location.

→ It is a mashup of 2 word Development (Dev) and operationals (ops)

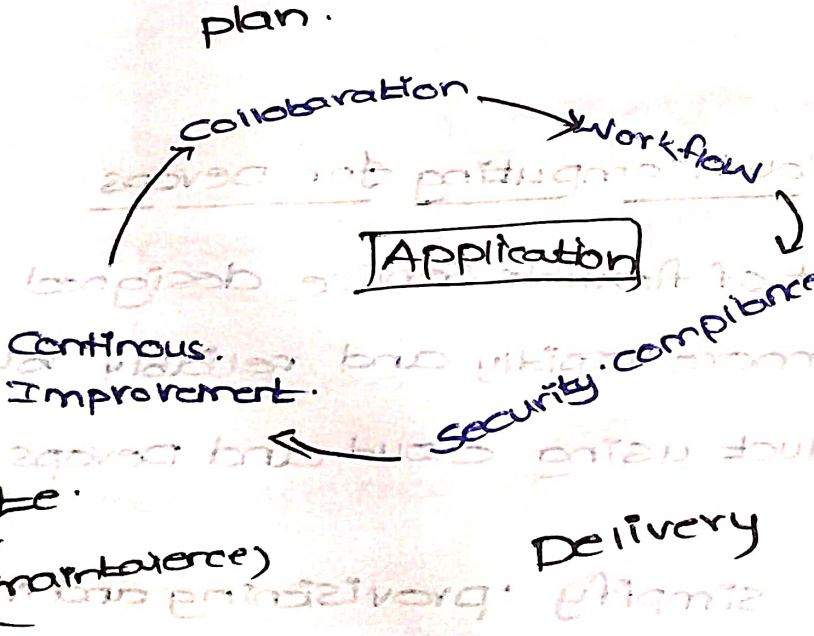
→ Devops speed delievary of Higher Quality S/w by combining and automating the work of S/w development & ITopes t

→ It help increase an organization speed to delieven applications and services.

Benefits of devops:

→ Speed.
→ Rapid delivary.
Reliability
Improved collabaration.
Scale.
Security.

Devops lifecycle:-




plan:- The team ideate, define and describe features and capabilities of the applications and system they are buildings.

- Develp :- It include all aspects of coding writing, testing, reviewing and the integration of code by team members as well as building that code into build artifacts (binarycode) that can be deployed into various environment.

→ Delever :- It is a process of depoying applications into production environment in a consistent and reliable way

→ In this phase, team define a release managment process with clear and manual approval stages.

→ operate :- The phase involve mainting, monitoring and troubleshooting application in the production environiment.

## cloud computing for Devops :

→ It offer a set of flexible service designed to enab companies to more rapidly and reliably build and deliever product using cloud and Devops practice

→ These service simpify provisioning and magning infrastructur, deploying applicationcode, automatic -ing s/w release processes and monitoring applicatons and infrasture performance.

# Tools for Devops.

→ Version Control — GIT, subversion, mercural, monotone, Bazar, visualcode, CVS, Rational clear case.

→ continous Integration & delivary [CICD] → Jenkins

→ configuration managment → Ansible

→ containerization → docker, kw

→ Infrastructure as a code. → Terraform

→ continous monitoring. → Nagios

→ cloud computing. → AWS, Azu

CICD → Jenkis, circleCI, Teamcity, Bamboo, gitlab. Travis CI

CM Tools: Anisible, Saltstack (google), Chef, puppet CF Engine, IBM Rational Synergy.

CN Tools: Docker, kubernetes [cluster] vagrant mesos [Apache] openshift, Azure container Instance.

Ic Tools: Terraform, cloudformation, ARM (Azur resource manager], CHef, puppet,

CM Tools: Nagios, JIRA, servicenow, SolarWinds, Logicmonitor, cloudwatch.

CM Tools: AWS, Azure, Gcp, Redhat, Ibm, Rackspace,

Notes: touch → timestrap.

→ VIm → texteditor, for modifying

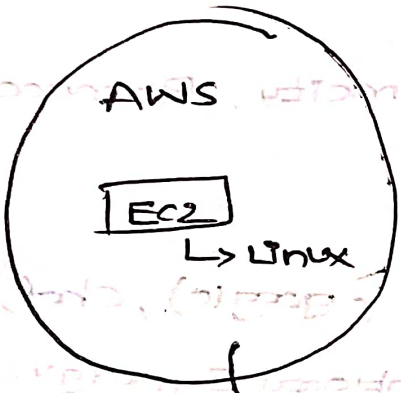
lsdsk -l → to list all the devices.

mkfs → filesystem

yum install → Redhat, centos,

aptget → ubuntun.

*




⇒ For security purpose

SSH

key-matching

keypair:- public
private.

abc → keypair (private key)

* [.pem
.ppk]

* Looses a private key (or) corruputed key-pair?

→ lifecycle policies

→ policy.

# * Dockers *

→ It is an "open-flatform tool for developing, shipping & running applications."

→ It provides the ability to package and run an application in a loosely "isolated environment" called container. [appdelivary technology]

→ Container is bit like a virtual machine, but unlike a vm, rather than creating a whole virtual o/s.

Docker: for :-

→ Fast consistent delivary of your application.

→ Responsive deployment & scaling.

→ Running more workloads on the same hardware.

## Architecture

dockerclient → support commands.

Input → given to Host / Install → deamons

→ Container → runnable instance.

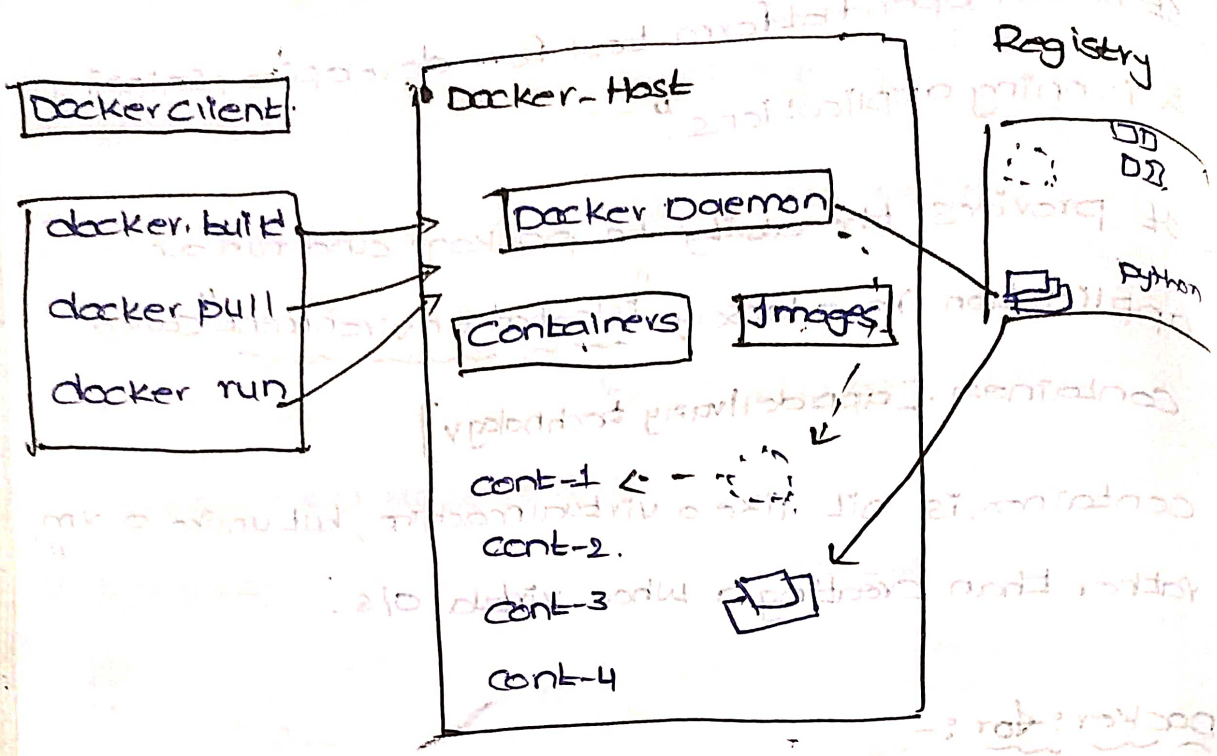
→ Registry → centralized (or) public hub. for image.

→ custom image → daemon.

↓

S/W

# Architecture:




Registry public → We need to main account, domain obtain doc.dockers.com,,

Docker Daemon:- (dockerd) → unix, tcp, and fd → /var/run/docker.sock

It listens for the docker API request and manages docker objects such as "Images, Containers, N/w & Volumes".

Docker client:

The docker client (docker) is the primary way that many dockers user interact with docker.

Docker registery:

It stores docker images. Docker Hub is a public registry that anyone can use and it is configured to look for images on docker Hub by default

# Docker objects :-

→ Image :- It is a read-only template with Infrastructure for created a Docker container.

→ It is an executable packages that include applicatb code, runtime, libraries, environment variable, cfg file

→ Containers :- It is a runnable instance of an image. You can create, Start, Stop, move (or) delete a container using the docker API (or) CLI.

→ Volumes :- They are directories and files that exist on the host file system outside of the dockercontainers.

→ These volumes are used to "perisist" data and share data between docker container.

Networks :-

→ It is a primarily used to establish communication between docker containers and the outside world via the host machine where the dockerdaemon is running.

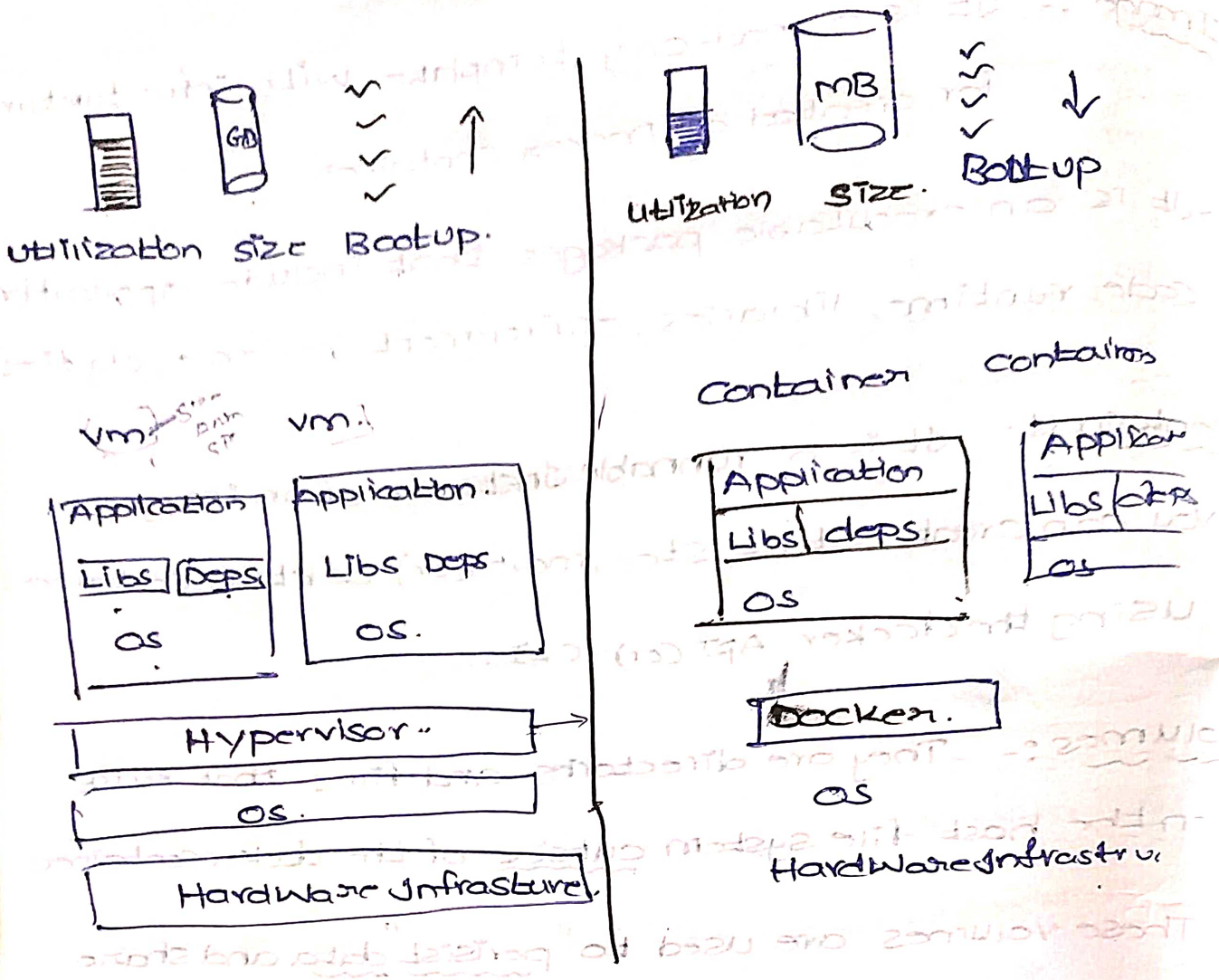
→ It include supporting for N/W containers through the use of Network drivers.

↓

Install drivers & communicate.

" container

# Vm vs container




* Adv:- Booktime decrease.
size. Utilization is less.

* containers workloads grouped in to cluster which is managed by kubernetis

Vm:→ Host based
It is a emulation of a physical computer.

→ It interact with physical computers by using lightweight S/W layers called hypervisors.

→ Hypervisor Separated vm from one another. allocated processor, memory and storage

→ as based

Containers. It is a light weight, more agile way of handling virtualization. → Since they don't have Hypervisors

→ Containers use a form of operating system (os) virtualization.

→ container → Application → Lib → dependencies.

→ Container are also ideal for automation & pipelines including CI/CD.

## Docker Installation:

Docker Engine :- It is an open-source containerization technology for building & containerizing your application

→ It acts as a client-server application with:

→ A server with a long-running daemon process ^ dockerd

→ API which specify interfaces that program can use to talk to and instruct the docker daemon.

→ A CLI client docker.

Support platform.

Docker Desktop mac, Windows.

Cloud platform: Aws, Azure, Gcp, Digitalocean

# Install docker desktop ON Window

* Window 11 (or) 11 64-bit : Home or pro 2004.

Enable WSL 2 feature on Window (Window Subsytem for linux, version 2)

Hardware properties:

64-bit, 4GB RAM, BIOS-level h/w Utilization.

## Install docker Engine on Linux:

→ RedHat (or) centos :-

To Install Docker Engined, you need a version:

RedHat / Centos 7, 8 (or) 9.

→ Install on ubuntu:

- Ubuntu Kinetic 22.10. → 64 bit
- Ubuntu Jammy 22.04 (LTS)
- " Focal 22.04 (LTS)
- " Bionic 18.04 (LTS)

* Installing docker on Amazon Ec2:

Step1 :- Sign in to AWS managment Console

Step2 :- Services → Compute → Ec2 (Scalble virtual Servers)

Step3 :- Ec2 dashboard → Instance → Launch Instances.

Step in Instance:

1. Name & tags → Docker.

2. Application and os image (AMI) → Redhat linux.

3. Instance type :- Combination of cpu, memory, storage, N/W.

t2-micro → 1CPU, 1GiB. RAM: [4 terabyte (4000 GiB) High-end application, ecommerce website]

4. key pair (login) :-

Create new key pair → keypair name docker

keypair type : • RSA • ppk → putty private key.

use of putty : It is used to connect from window to linux

CLI → GUI

5. → N/W setting :

Create Security group (SG) → (virtual firewall)

☑ Allow SSH traffic [port 22]

6. Launch Instance → backend

Downloading

## * Downloading putty :-

It is used to connect unix (or) linux machine from window.

→ Website :- putty.org → download putty. →

Alternative binary files → 64-bit x86 (putty

put on desktop. [ dockervkeyvalt

→ How to connect instance :-

Select instance. → copy public IPV4 Address → then

open putty → past on Host Name (port) 22.

→ Expand SSH under Categories → Select Auth →

Browser → docker.ppk → open → login.

Login :- ec2 - user then Enter.

default user → Normal

Sudo - i → Super user (To switch) N → Super

## * version :- cat /etc/redhat-release of redhat

→ 9.1

Update the Red Hat server :- yum update -y

→ docker install

Google → docker install → Docker Engine install

overview → RHEL

install method:

1. yum install -y yum-utils.

2. Enable dockery repristory :- yum-config-manag
\ --add-repo\ http://download.docker.com/linu
/rhel/docker-ce.repo

3. Again update the servers → yum update -y

Resolving respority Error :-

→ cd /etc/yum.repos.d/

→ ls

→ vi docker-ce.repo [edit]

→ Remove the Existence content in the file, then add some lines.

→ [docker-ce-stable]

name = Docker CE Stable - $basearch

baseurl = https://download.docker.com/linux
/centos/$releasever/$basearch/stable.

enable = 1

gpgcheck=1.

gpg key = https://download.docker.com/linux

Step3: yum install docker-ce docker-ce-cli containerd.io docker-compose-plugin.

* systemctl status docker.
  systemctl start docker.
  systemctl enable docker.

* Stop Every Instances: → Instance State → Stop

* public Ip. →
  Username → Username@Ipaddress
  keypair (password)

  ⇒ putty
  → puttygen

  • pem → permission enahced mai;

→ key-pair location.

* manage docker as a Non-root user.

→ groupadd groupname (docker)

→ add user → create a user.

useradd username (raju)

→ set password :- passwd username (raju)

→ Newpass :- ——— [Hemabathai]

Imp step: Adding user in the group. →

usermod -aG docker raju. → add → Groupname.

→ Identify with group :- id -Gn raju.

User belong to which group.

→ update groupname :- newgrp docker.

Switch user :- su -raju

account.

whoami → check which user.

* --version → to known which version (or) not

docker Helpoptions:

→ docker → list all docker command

docker --help → help of a particular command

docker images --help [ List images ]

* docker rmi --help: [removes all images]

Docker Syntax:-

docker [option] command

manage Docker Images:

-> Docker image is a "read only - template" with instru for creating a docker container.

-> An image is based on another image, with some additional customization.

-> Image is an "executable package" that include everything needed to run application : code, runtime libaries, env, variables, configure files.

Syntax:

$ docker image command

- build
- history
- import
- inspect
- load
- ls
- prune
- pull
- push
- rm
- save
- tag

Image usage:

Syntax: docker image [options] [Repository[:Tag]

available images:- docker images | docker image ls
in docker

1 -> How to search in docker hub for images :-

docker search ubuntu (Image name) → ubuntu, java, python, centos

2 -> Download the image for hub :-

docker pull ubuntu.

3 -> To verify -> docker images.

-> docker pull centos :7 -> 7th version it download

-> List entire id docker images. for full description
↓ short
↓
to long:-

docker images --no-trunc.

To check image history:

docker image history ubuntu (Imagname (or) ImgId.

* Notes:

SSH -> bash -> CLI -> low to high level

container.

Check syntax & commands

*. In linux default shell -> bash

Advance shell of sh. (bourne shell)

bash:- bourne Again Shell

* To check image details : → comple depth info

docker image inspect Ubuntu.

* docker images with label name :-

docker images --filter "label=centos: pyth

* Removing images:

docker images.

docker rmi ubuntu.

docker image rm ubuntu

* Dangling images :-

It is a an image that is not tagged and it is not used by an container.

→ docker images --filter "dangling=true"

↓

list image which are unsed

→ delete dangling images:

docker image prune

Docker container :- It is a "runnable instance" of an image. You can create, stop, move or delete

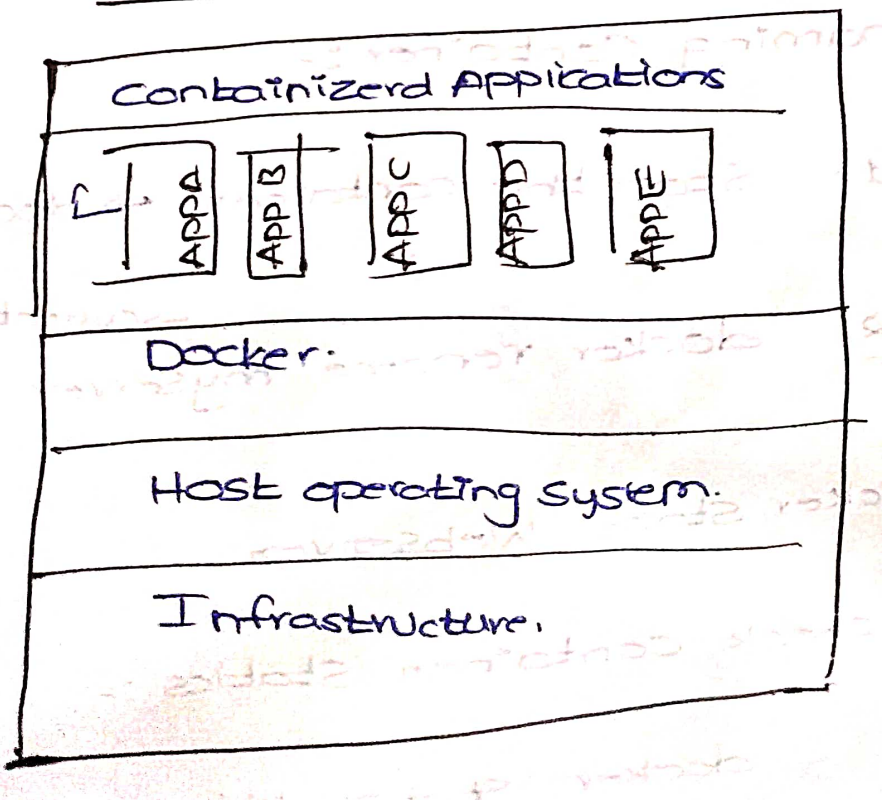
-> It is a standard unit of s/w that packages up code and all its dependencies. So the application runs Quickly and reliably from one computing env to another

Docker containers are :

containerized

1. Flexible :- most complex applications can be
2. Light weight :- Containers leverage and share host kernel
3. Interchangeable : can deploy updates & upgrade on-the-fly (while running only)
4. portable :- Can build locally, deploy on cloud, run anywhere
5. Scalable :- Can increase automatically distribute container replicas
6. Stackable :- Can stack services vertically and on the fly.

Container architecture :



## * Manupbting docker images :-

1. docker run -i -t <imgname> : <tag> /bin/

-i → interactive shell

-t → will allocate a pseudo-tty.

-d → deamon mode.

2. docker ps → [List running containers]

3. docker run -it -d centos

4. docker run --name myserver -it -d Ubuntu
(name)

5. docker stop myserver [To stop a container]

6. To list the stoped container: docker ps -a.

7. docker start myserver. → Start the container

8. Renaming Container :-

Step1. Stop the container → docker stop myser

Step2: docker rename myServer WebServer
(current) (Next Con

9. docker start Webserver.

10. To check container Statics :-

^c

docker stats Webserver. [to close

11. * To monitor the container : docker top webserver

12. pausing the container : docker container pause webserver.

To known the pause (or) not → docker ps.

unpause: docker container unpause webserver.

13. To kill a container :- docker kill container name (or) container id

14. To check containers logs :-

docker logs webserver.

Logs, → to monitor containers logs continuously.

docker logs -f webserver. [^c] ctrl+c

15. docker system df → How many images, containers are created ?

16. docker system events. → 2-terminal [open 2-term

on the 1 terminal run the command [dockersystem events]

on the 2 " " Just, execute the command

start & stop, verify the events logs

* Run a command in a running container :

→ docker ps.

→ docker exec -it webserver /bin/bash. Enter into a container

→ pwd

→ ls → bin boot dev etc home lib lib32 lib Media mnt opt proc root run sbin srv sys

→ Ubuntu → apt-get install apache2 -y

→ exit → to logout the container.

* Without login, in to container to list.

docker exec -it webserver ls.

ls /opt → direct

pwd.

* directly log in to container :-

docker run -it centos /bin/bash.

* docker run -it roboxes/rhel8 /bin/bash

↓ image.

2 process: 1. It searches in the local reprasitry.

2. orelse it pull the image from hub

*

Ravi → docker inspect :- It provides detailed information of container.

→ · docker ps.

docker Images.

→ detach mode → container name

docker run -it -d --name myserver ubuntu

docker ps

→ docker inspect containername (or) Id.

* docker diff → It is used to inspect changes to files (or) directories on a container file system.

docker ps → docker diff ~~Image~~ containerID.

→ Inside the container

↓

(duplicate session)(login) → docker exec -it id bash

create a file → touch /opt/abc.

ls /opt/ → output: C/opt
A /opt/abc

abc

→ ~~apt-get~~

→ docker diff e4a1cbc88bac

apt-get update -y

options in output: Symbol description

A → A file (or) directory was added.

D → A file (or) directory was deleted.

C → A file (or) directory was updated (or) changed (*)

** docker cp → copy files (or) folders b/w a container and the local host file system.

makefile → touch aws → ls. → copy file to ubuntu container.

docker ps.

docker cp ./aws e4a1bc88bab:/opt → current location

docker exec e4a1bc88bab ls /opt.

abc
aws

24. Coping container to Local filesystem

file in container
→ docker cp eualabs-boo : /opt/abc /tmp.

ls/ ls /tmp

25. docker info :- It displays System Wide informat

26. Removing containers:

docker ps.

docker ps -a.

step1: Stop the container: docker stop "containerID"

step2: Delete container:- docker container rm CID
(containers)

docker container rmi CID
(images)

27. To removing all Stoped container

docker container prune.

docker ps.

docker ps -a.

## * PORT Forwarding :- [Imp]

port forwarding (or) mapping redirects a communication request from one Ipaddress & port number combination to another.

→ Services are exposed, to the applications residing outside of the host internal Network.

Syntax :- docker run -d -p <host-port> : <container-port> <image> :<tag>

→ To Host a Webpage → nginx.

nginx

process : 1. docker pull nginx.

2. docker images. → container ID

Run a container 3. docker run -d --name Webserver -p 8000:80 nginx

8000 : 80
↓ ↓
host port container

→ 4. Go to Ec2 Instance → publicIpv4 → co

Go to Instance Security Group.
↓
Edit Inbound rule → Add rule.
↓
CustomTcp rule : 8000
(Anywhere) → Save rule.

(Troubleshooting - issues)

→ Webserver :- publicIpv4 : 8000
↓
Nginx

## * Allocate memory & RAM for a container:

→ (Limitation of "RAM" in a container)

docker run -d --name Webserver1 -p 8001:80

-m 512m nginx. (m=memory)

docker stats

→ (Limitation of "cpu" in a container)

docker run -d --name Webserver2 -p 8002:80

--cpu-quota = 50000 nginx (50%)

docker stats.

Both (RAM + cpu) allocation

docker run -d --name Webserver3 -p 8003:80.

-m 300m --cpu-quota=5000 nginx.

## * 30. docker update:

It is used to dyramically updates Container config on the fly

Syntax:

=> docker update -m 300m Webserve1

docker Stats

To change cpu limit share.

docker update --cpu-shares 512 webserver1

docker stats.

## Dockerfile:-

→ Dockerfile is the core file that contains instructions to be performed when an image is build.

→ It is a text document that consists all the command a user could call on the CLI to assemble an image.

→ The docker build commands builds an image from a dockerfile and a context

Syntax:- Instruction arguments

### * Dockerfile Instruction & Arguments.

1. Form: A dockerfile must start with a FROM instruction. The FROM instruction initialize a new build stage and set the Base image for subsequent instruction

Ex: FROM centos:latest

2. MAINTAINER: It is used to specify about the author who create new docker image for the support

Ex: MAINTAINER RAJU mraju4u@gmail.com

How to create dockerfile :-

vi Dockerfile → file open → i (insert mode)

File content :

```
# This is a docker first project

FROM ubuntu:latest
MAINTAINER RAJU rnraju4u@gmail.
LABEL "Name"="Webserver"
esc : wq!  → Save
RUN apt-get update -y
```

Instructions

→ is

3. Label :- It is used to provide (specify) metadata (→ containing data) Information to an image, which is a "key-value" pair

Ex: LABEL "APP-Env" = "production"

4. RUN :- It is used to execute any commands on top of the current image and this will create a new layer.

1. SHELL Form :

RUN yum update -y

2. Executable form:

RUN ["yum", "update"]

* Building images using dockerfile :-

-> The "docker build" command builds Docker images from a dockerfile and a "context"

-> A build context is the set of files located in the specified PATH (or) URL.

docker build.

docker build -f <path.to-dockerfile> -t <REPOSITORY>:<TAG>.

docker build -t apache:latest . → CWP → Syntax

docker images.

-> launch container:

[While execution instruction if smtg goes wrong, it uses cache"]

docker run -it -d --name WebServer apache:lat

docker ps.

* Create instruction
Build
Run
] 3 steps to create docker images

## * Build an Image apache 2.4 Webserver.

open dockerfile → vi

FROM ubuntu:latest.

MAINTAINER RAJU rnraju4u@gmail.com.

LABEL "Name" = "webserver" \

"project" = "Airtel" \

"Env" = "production.

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

(Installation & upgradation purpose)

→ docker build -t webserver:2.4 → cmp.

↓

tagname.

→ docker run -it -d --name Apachewebserver

Webserver:2.4

→ docker ps:

5th: CMD → [Regular command like ls, rm]

→ It is used to set a command to be executed

When running a container.

→ There must be one CMD in Dockerfile. If more than one CMD is listed, only the last cmd takes effect. [" cmd] ping www.google.com

6. ENTRYPOINT :- It is used to configure and run a container as an executable.

Ex: ENTRYPOINT ping google.com

NOTE: If user specifies any arguments (commands) at the end of "docker run" command, the specified command (override).

→ Entrypoint instruction is not overwritten by the "Command" in the file

→FROM
→main
→label →RUN

* vi dockerfile. →CMD ping google.com

docker build -t googping .

docker images

docker run -it googping bash

Solving: ↓

ping google.com [Command Not found]

Rundockercontainer: docker run -it -d containerid

docker ps.

docker exec -it containerid ping google.com (→ command)

## Cmd example

```
FROM centos:latest

MAINTAINER RAJU rnraju444@gmail.com.

LABEL "Name" = "Webserver" \
      "project" = "Airtel" \
      "Env" = "production.

CMD ping google.com.
```

→ docker run test.

- docker run test pwd → it works. (overide)

After Run command any command override

## Entrypoint

```
FROM
MAINTAINER
LABEL
ENTRYPOINT ping google.com.
```

: we can't change runtime command (fixed)

docker build -t test1 .

docker run test1

docker run test1 pwd.

↓

google

7. Copy: It is used to copy files, directories and remote url files to the destination (dockerimage) within the filesystem of the docker images.

Eg: Copy src dest.

Note: If the 'src' argument is compressed file (tar, zip, bzip2 - - etc) then it copy and doesn't extract.

6. Add:- It is used to copy files, directories and remote URL files to the destination within the filesystem of the dockerimage.

Ex: ADD src dest.

Note:

It will extract (zip, archive) file, automatically inside a destination in the docker image.

* makefile cat >aws → Hi, Hello (ctrl+d) Save.

cat > azure → hello.

cat > gcp → Ravin

zipfiles

→ tar -cvf cloud.tar aws azure gcp.

→ ls.

Copy: Vim Dockerfile (Normal file)

→ destination of container

→ COPY aws /opt

↳ Source.

→ docker build -t Sample .

↑ imagename.

→ docker run Sample ls /opt. → output → aws file.

2ndEx [zipfile]

COPY cloud.tar /opt.

docker build Sample1.

docker run Sample1 ls /opt > cloud.tar.

Add Example:-

ADD azure /opt → built → run → output
↓ azure

2ndEx. ADD cloud.tar /opt → aws, azure, gcp } output

Copy → Copy the sample files to destination
↓
cloud.tar

Add → To extract the files.
↓
aws
azure.
gcp.

9. WORKDIR :- It is used to set the working directory. Ex:- WORKDIR /tmp.

10. EXPOSE:- This instruction inform Docker that the container listens on the specified network port at runtime.

→ By default EXPOSE assume "TCP" & need to specify UDP.

→ To establish, the port running the container. We use the -p flag on the container.

→ vi Dockerfile :

→ WORKDIR /opt.

→ EXPOSE 8080/tcp.

→ docker build -t cloud .

→ docker build cloud pwd

11 USER.

The user instruction lets you specify the username to be used when a command is run.

→ It is used to set the username, groupname, UID, GID, for running subsequent command else root user will used Ex: USER ltebadmin

12. Env: It is used to set environmental variable with "key value" set

Ex: Env username cor) admin.

13. ARG: It is also used to set Env variables with key-value, but this variable will set only during the image build not on the container

Ex: ARG temp-ver 2.0

14. ONBUILD: It lets you stash a set of commands that will be used when the image is used again as a base image for a container.

Ex: ONBUILD ADD
ON BUILD RUN

Project

Installing python Flask

Vim Dockerfile:

From centos: latest (or) 7

MAINTAINER : Raju rraju4u@gmail.com

RUN yum update -y

RUN yum install -y python3-pip wget

RUN pip3 install Flask.

ADD hello.py /home/hello.py

WORKDIR /home. → save & quit

→ create application → cat > hello.py

print C" Hello welcome to python"]

↓ ctrl+d

docker build -t python .

project: Creating Webserver Images.

→ Create a docker file. (*) ← Webpage (default location) ↓ /var/www/html

From centos:7

RUN yum install httpd -y

Copy index.html /var/www/html/

CMD ["/usr/sbin/httpd", "-D", "FOREGROUND"]

Expose 80.

File index.html vim index.html

<html>