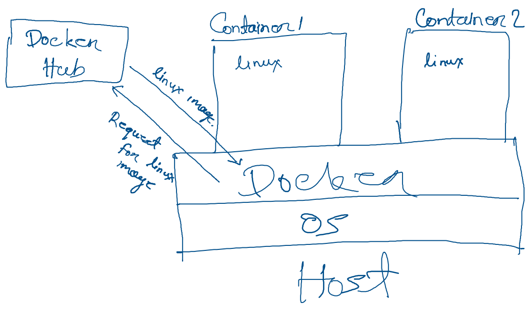
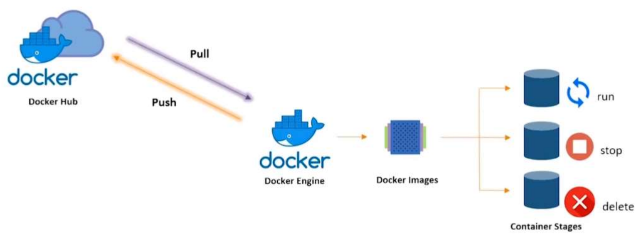
Docker

Tuesday, 2 August 2022 2:34 PM

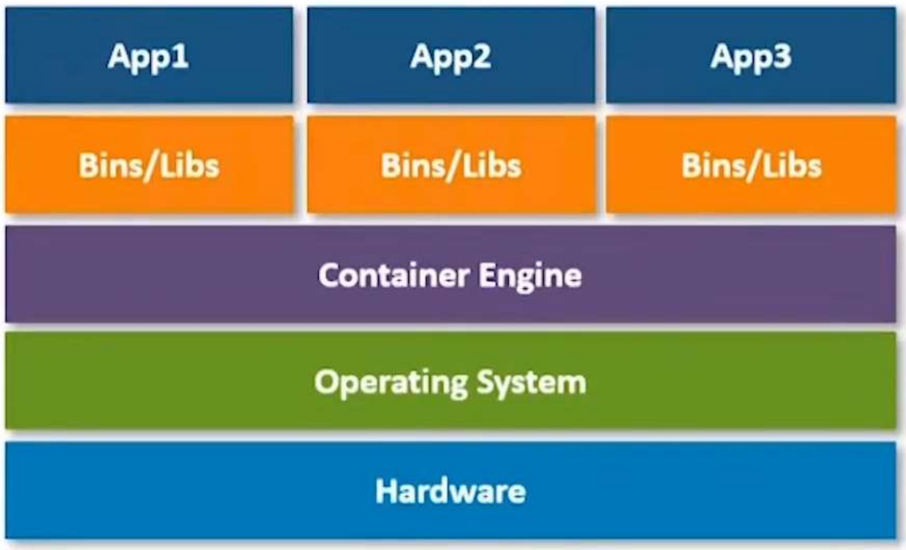
Docker is an open sourse centralised platform design to create, deploy run application.
Docker uses containers on the host OS to run application.
We can install docker on any OS but docker engine run natively on linux distribution.
Docker written on 'go' language.
Docker is a tool that perform OS level virtualization it is known as containerization.





When developer create container the install all dependency software on this container. Then create a image file of this container and send to operation team.

Operation team directly run the image file on the top of docker.



Docker installation.

→ Ubuntu → sudo apt-get install docker.io

→ Amazon linux → sudo amazon-linux-extras install docker

Docker command.

docker --version → To show which version of docker installed.

docker pull <image name> → It download docker image file from docker hub.

Ex!- docker pull ubuntu

docker images → It display all downloaded docker image File.

docker run <image name> → This command help to run docker image and create container.
Ex:- docker run -it -d ubuntu

-it → make container interactive so I can write command
-d → It add docker to daemon Job so it should run in background.
-p 82:80 → [It map port 80 with port 82 of host machine
docker run -it -d -p 82:80 ubuntu]

docker ps → It list all the container which are running in the system.

docker ps -a → It display all stop container and also running container.

docker exec <container id> → It logging into container terminal / accessing container
Ex:- docker exec -it 233c96156D1 bash
↑ Interactive ↑ Container id ↑ which shell

exit → It exit from container shell to host system shell.

docker stop <container id> → It stop the running container.
Ex:- docker stop 233c96156D1

docker kill <container id> → in case the container is non responsive you directly kill container Job using kill command.
Ex:- docker kill 233c96156D1

docker rm <container-id> → It is use to remove a stopped container from the system.

docker rmi <image id> → To remove an image from the system we use the command "rmi"

docker commit <container-id> <set image name> → It create a new image (snapshot of container)

docker login → login to docker hub account.

docker push <image name> → It upload the image on your docker hub account.

Docker File

It is a text document that contain all the command a user could call on the command line to assemble an image. Using docker build users can create an automated build that executes several command line instructions in succession.

FROM

It define the base image, on which we will be building.
Syntax:- FROM ubuntu

ADD

ADD key work is used to add file to the container being built
Syntax:- ADD <source> <Destination address>

```
FROM ubuntu
ADD /Project /var/www/html
```
} It copy all File From project to html

RUN

The RUN keyword is used to add layers to the base image, by installing components. Each RUN statement, adds a new layer to the docker image.

Syntax:-

```
FROM ubuntu
RUN apt-get update
RUN apt-get -y install apache2.
ADD . /var/www/html
```

CMD

The CMD keyword is used to run commands on the start of the container.
These command run only when their is no argument specified while running the container.

Syntax:-

```
FROM ubuntu
RUN apt-get update
RUN apt-get -y install apache2
ADD . /var/www/html
DMD apachectl -D FOREGROUND
```

ENTRYPOINT

The ENTRYPOINT keyword is used to strictly run commands the the moment the container initilized.
ENTRYPOINT is same as CMD but it run whether the argument is specified or not.

Syntax:-

```
FROM ubuntu
RUN apt-get update
RUN apt-get -y install apache2
ADD . /var/www/html
ENTRYPOINT apachectl -D FOREGROUND
```

ENV

The ENV keyword is used to define environment variables in the container run time.

Syntax:-

```
FROM ubuntu
RUN apt-get update
RUN apt-get -y install apache2
ADD . /var/www/html
```

ENTRYPOINT apachectl -D FOREGROUND
ENV name Devops

Create → Write your docker code on a editor and save it as Dockerfile.
Run → docker build . t < image name> → It run the file.
Example → docker build . -t webserver

Docker Volumes

It is used to persist data across the life time of a container

docker run -it -v /home/ubuntu/dockerfile:/app -d ubuntu → This command create a container and mount the location.

This is source address of Host machine

This is destination of container machine.

- → Volume is a simple directory inside our container
- → Firstly, we have to declare this directory as a volume and the share volume
- → Even if we stop container still we can access volume.
- → volume will be created on one container.
- → You can declare a directory as a volume only while creating container.
- → You can't create volume from existing container.
- → You can share volume across any number of container.

We map volume in two ways
1) Container to container
2) Host to Host

Command

docker volume create <name of volume> → It create a volume.

docker volume ls → It display all volume present inside volume.

docker run -it --mount source=<name of volume>, target=<location of container folder> -d <image name>
It create a container and with a mount point which is mount with host.
source = location of host machine.
target = location of container
Ex:- docker run -it --mount source=test, target=/app -d ubuntu

Micro Services

What is monolithic application ?

Monilithic means every components in a single program.
Application are large and complex to understand.
If any new version is came then developer should re upload Full code.
If any bug found in any module it affect on entire application.
If the code written php it take to change code php to other language.

What is micro services?

In micro services architecture each and every module independent to each other.

All the services are loosely coupled.
Micro services are a software develepement architectural style that structures an application as a collection of loosely coupled services.

Application is distributed, hence easy to understand.
The code of only the microservice which is supported to be updated is changed.
Bug in one service doesnot affect other services.
No barrier to any specific technology.

Docker Compose

It is a tool it can be used to create multiple docker application at same time.
You use a YML File to configure your application's services. Then with a single command, you create and start all the services from your configuration.
Run docker-compose up and compose starts and runs your entire app.
↑
It run yml File

Docker swarm

It is a tool who monitor all container health, if any container unable to response then it create a clone of same container to make sure container always alive.
It is also know as Container Orchestration.
In such case it stopping the unhealthy container and create a new one in backend. end user not aware of it.

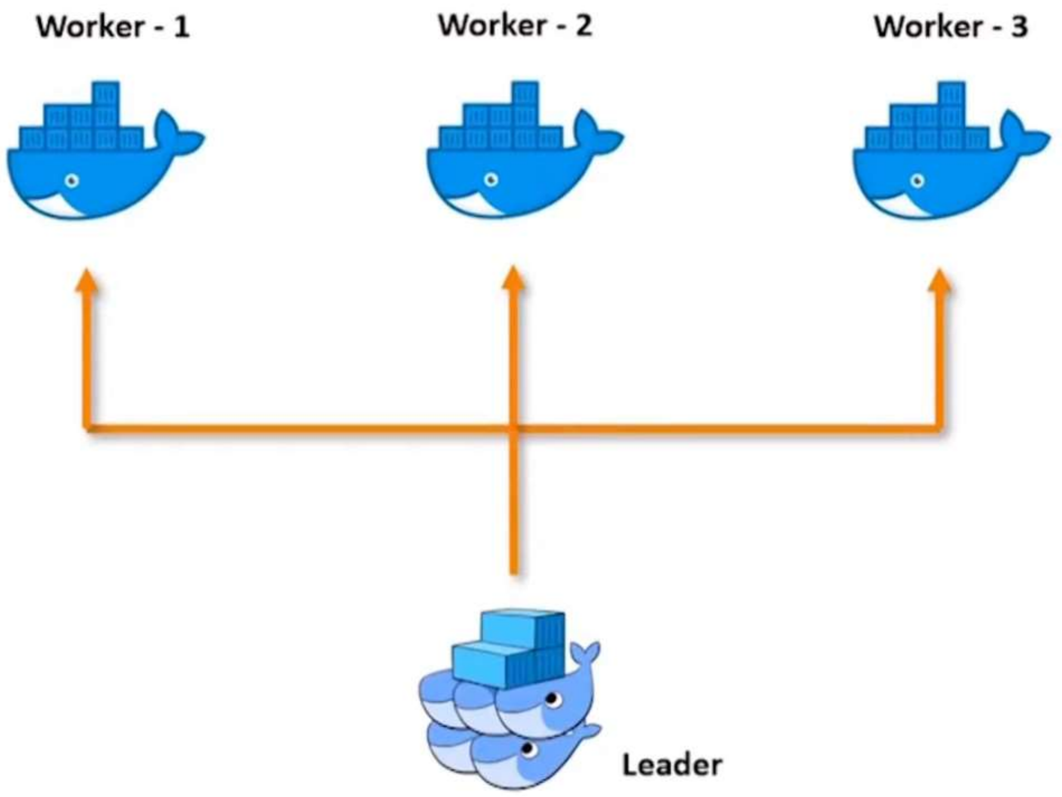
Commands

docker swarm init --advertise-addr = <private ip of master instance> → This command run on master instance. and in address add master instance private IP address then it return another command to add worker on it.

docker node ls → It display all present worker cluster (This command run on master instance)

docker swarm leave → Through this command worker instance leave the master node. (This command run on worker instance)

docker swarm leave --Force → Through this command master leave the swarm. (This command run on worker instance)



Service

Container on the cluster are deployed using services on docker swarm.
A service is a long-running Docker container that can be deployed to any node worker.

Command

```
docker service create -name <name of service> --replicas <number of replicas> <image name>
```
This command create service.

Ex:- `docker service create --name apache --replicas 7 -p 83:80 ubuntu`

```
docker service scale <name of the service>=<no of replicas>
```
This command scale up and scale down the replica.

Ex:- `docker service scale apache=2` → It down the replica 5 to 2.

Ex:- `docker service scale apache= 8` → It up the replica 2 to 8.

`docker services rm <name of the service>` → It remove the service.