Docker is a container Engine which contain Docker container and Images. Docker images is a Blue Print of a container. And container is a Small size running Virtual Machine.

In old Virtualization System there is hypervisor that provide a Hardware Virtualization for a Gest OS. There is new Kernel install on the hypervisor. If there is 3 OS are install on a Hypervisor, then 2 Different kernel are install on Hypervisor. All Images(.iso) file contain OS libraries with Kernel. So it sizes reach to 1 GB to 5 GB. And we need minimum 5 to10 GB of Space and 512 MB to 8 GB of Ram to install OS and run it.

Instead of all this we have solution of Docker.

Docker container use host system kernel and it contains only required libraries. It images size is also small. Docker engine gives same environment as virtual Machine.

Yes, but Question is this Why we need Docker in a DevOps Work flow?

Docker provide vitalization architecture so we can use same environment on Developer, Tester and Production Server. Developer Code on their Laptop on a container then it can check with a same configuration with tester side and it also deployed in Production Server.

Docker installation

sudo apt-get update

sudo apt-get install docker.io

docker –version

docker pull < image-name>

docker pull ubuntu

docker images

docker run < image-name >

docker run ubuntu

run container interactive and demon (run in background)

docker run –it –d ubuntu

view all running container

docker ps

view all container in system

docker ps –a

stop container

docker stop <container id>

working with (in) container

docker exec –it <container id > bash

exit to container

exit

stop container force fully

docker kill <container id>

remove container

docker rm <container id>

remove image

docker rmi <image id>

<https://hub.docker.com>

remove forcefully container

docker rm –f <container id>

save changes in container

docker commit <container id > <new image name>

remove all running container

docker rm –f $(docker ps –a -d)

port mapping there is one more flag

docker run –it –p 82:80 –d <imagename>

-p<OS port: container port>

For push images to docker hub

docker login

docker push <imagename>

note: when you push image on docker hub you image name in username/imagename

Docker file

FROM ubuntu //<image name>

ADD . /var/www/html ////<source> <destination in container>

RUN apt-get update // run conde

CMD apachectl –D FOREGROUND

ENTRYPOINT apachectl –D FOREGROUND

ENV name value

To build cocker file

docker build <source of dockerfile> –t <name of dockerfile>

Docker Volume

docker volume create <name of volume>

docker run –it --mount source=<name of volume>, destination=<dest add in > -d <image name>

docker volume ls

Docker Volume Bind Mount

docker run –it –v <source-directory>:<destination-directory> -d <container-name>

docker run –it –v /home/ubuntu/mount:/demo –d ubuntu

Define name of continer

docker run –it --name container1 –d ubuntu

docker run –it --name container2 --ink container1 –d ubuntu

cat /etc/hosts

apt-get update

apt-get install ping

apt-get install iputils-ping

ping container1

Docker compose:

sudo curl -L "https://github.com/docker/compose/releases/download/1.23.1/dockercompose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

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

mkdir docker

nano docker-compose.yml

version: '3'

services:

sample1:

image: 'httpd'

ports:

- "80:80"

sample2:

image: 'nginx'

docker-compose up –d

docker-compose down

server

docker swarm init --advertise-addr=<ip-address-of-leader>

docker swarm leave --force

docker node ls

docker service create –name <name-of- service> --replicas <number-of-replicas> <image-name>

docker service create –name nginx –repilcas 3 –p 80:80 nginx

docker service ls

sudo usermod –aG docker $USER