**DOCKER**

**Container**: It is like a Virtual M/C.

**Docker**: It is tool to create containers.

Docker is an open source centralized platform design to create, deploy and run any type application.

Docker use containers on the host OS to run application, it allows application to use the same Linux kernel/OS used on the host computer rather than installing separate OS.

We can install Docker on any OS, but Docker engine runs natively on Linux distribution.

Docker is a tool that performs OS level virtualization, also known as "containerization".

It was first released in 2013 and was developed by Doctor In.

OS level virtualization: it used 95% OS of host computer and remaining 5% OS take from Docker hub in the form of image, we get that image in Docker engine and we copy that image in container. This is the way we get 100% OS in container.

Container has 5% OS in the form image which is very negligible that is why we called container doesn't have any OS.

Docker is a tool used to create virtual machine called "containers".

In container we can deploy any type of application easily.

Docker is a set of platform as a service that uses OS level virtualization, whereas VMware use hardware level virtualization.

**VMware vs Docker**

VMware we need to install OS then S/W.

Docker we doesn't need OS, We can directly install S/W.

VMware reserve ram and hard disk from base laptop.

Docker won't reserve any ram and hard disk, just used required resources and return to base laptop.

**Advantages of Docker**

1) No need to allocate any ram or Hard disk

2) Allow you to create container image and use that same image in every step of deployment process.

3) Less cost

4) Light in weight (need very less resources)

5) You can run on physical h/w, virtual/cloud,

6) You can reuse same image to create multiple container

7) Create container in very less time.

**Disadvantage of Docker**

Docker is suitable when development OS and testing OS are same.

“Container is layered file system everything happens in the form of layer one by one by one.”

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Architecture of Docker**

Developer create Docker file, it contain dependency and required software information.

Docker file we run in Docker engine and it creates image.

With the help of image you can create a container.

We can also share that image on Docker hub.

Other person can pull that image from Docker hub and create container.

**Components of Docker**

**Docker Demon**: it is Docker service , responsible for creating container.

**Docker Client**:  Where we run Docker command call DC.

**Docker Hub**: It contains all Docker Images.

**Docker Image**: It is read only file also call OS to run container + some software.

**Docker Container**: It is like VM which is created from Docker image.

**Docker File**: A text file it contains some set of instruction to build image

**Way to create image**

1) Take image from Docker hub.

2) Create image from existing Docker container.

3) Create image from Docker file.

**Docker container**

Docker container is like VM has complete infrastructure to run the application.

Image becomes container when we run on Docker engine.

"We cannot modify image but we can modify container and create a new image (modified image)"

**Lab**

**1) Take image from Docker hub.**

1) First create is Linux EC2 instance with **SSH** and **HTTP** port.

2) Update EC2 Instance: **#yum update -y**.

3) Install Docker package : **#yum install docker -y**.

4) Start Docker service :

**#systemctl start docker.service**

**#systemctl enable docker.service**

To see Docker package install or not

**#which docker**

To see doctor version

**#docker -v** or **#docker --version**

Docker version 20.10.13, build a224086

To see only running containers in your machine

**#docker ps**

To see all containers in your machine

**#docker ps -a**

To see list of Docker images present in your machine

**#docker images**

To find out image in Docker hub

**#docker search <imagename>**

Eg. **#docker search centos**

To download image from Docker hub

**#docker pull <imagename>**

Eg. **#docker pull centos**

To get image from Docker hub and create container and start container

**#docker run -it <imagename> /bin/bash**

i = interactive mode

t = terminal

To create container and start container but don't go inside it

**#docker run -dt <imagename> /bin/bash**

Replace i with d

To get image from Docker hub and create container give name and start container

**#docker run -it --name <name> <imagename> /bin/bash**

To rename a container

**#docker rename <old container name/ container id > <new cname>**

To start container

**#docker start <** **container name/** **container id>**

To go inside container

**#docker attach < container name/ container id>**

To stop container

**#docker stop < container name/ container id>**

To delete container

**#docker rm < container name/ container id>**

To delete image

**#docker rmi <imagename>**

"You cannot delete running container first stop the container and then delete the container"

"When you are creating a container it will search image in your machine docker engine if image is not there then first it will pull image from Docker hub and copy that image in your machine docker engine and from this docker engine image it will create container"

**2) To create image from our own container**

**#docker commit < container name/ container id > <imagename>**

To see what the activity is you have done in this container

**#docker diff < container name/ container id >**

**3) Create image from Docker file**

Docker file is basically a text file it contains some set of instruction to build image.

Docker file name must and should be "Dockerfile"

**Docker file components**

**FROM**: For base image, this command must be on top of the Docker file.

**RUN**: To execute command, it will create layer in image.

**MAINTAINER**: Name of author or description.

**COPY**: Copy file from base OS, need to provide source and destination (we can't download file from internet and any remote repo)

**ADD**: It provides a feature to download files from internet, also we can extract file Docker image side.

**EXPOSE**: To open port (mention port no)

**WORKDIR**: To set working director (after creation of container you would be landing this particular directory)

**CMD**: Execute commands during container creation.

**ENTRYPOINT**: Same as CMD but high in priority (executed before CMD)

**ENV**: Variable

**VOLUME**: TO create docker volume

**ARG**:

**Steps**

1) Create file named Dockerfile

2) Add instruction in Dockerfile

3) Build docker file to create image

4) Run image to create container

#vim Dockerfile

FROM centos

RUN echo "Hello Rahul" > /tmp/test

To create image from docker file

**#docker build -t <imagename> .**

**(dot means Dockerfile is in current directory)**

#vim Dockerfile

FROM centos

WORKDIR /tmp

RUN echo "Hello Rahul" > /tmp/test

ENV myname rahul

COPY testfile1 /tmp

ADD test.tar.gz /tmp

If you use rahulfile name instead of Dockerfile then used following command to build image

**#docker build -t testimage . -f rahulfile**

**Docker Volume**

Volume is simply a directory inside our container and we can share this directory with other containers.

Firstly we have to declare this directory as a volume and then we share volume.

You can declared directory as a volume only while creating contains, you can't create volume in already existing container.

You can share one volume with any number of containers but at the time of container creation only.

Even if we stop container still other containers can access the volume from shared containers.

Volume will be created in one container and share with the other containers no need to create volume again and again in each and every container.

When you create an image from your existing container which contain volume and from that image if you create any other container then you get the data which present inside your volume directory but no connection between new container volume and old container volume, it will treated as normal directory in new container.

You can share volume in two ways container to container and host machine to container

**Advantage of volume**

1) We can share volume among different containers

2) Attach volume to containers

3) On deleting containers volume does not delete

**Lab**

**Create volume from Dockerfile**

1) Create Dockerfile

#vim Dockerfile

FROM centos

VOLUME ["/myvolume"]

2) Now build image from Docker file and Create container1 from this image.

3) Now created container 2 and share volume at a time of creation using following command

**#docker run -it --name container2 --privileged=true --volumes-from container1 <imagename> /bin/bash**

**Create volume by using command**

**#docker run -it --name container3 -v /myvolume2 <imagename> /bin/bash**

Sharing same as above.

How share volume with host?

**#docker run -it --name <cname> -v /root:/volume3 --privileged=true <image> /bin/bash**

-v /root:/volume3 link /root of host and /volume3 of container

Suppose we are sharing volume with two containers and if you delete volume directory from container1 then container2 volume directory won't get deleted.

If we delete data from container1 volume directory then container2 volume directory data will get delete.

**Docker Port Expose**

Container does not have any public IP, having only private IP.

Mapping port of host to port of container (Open container port to internet to access website).

If you stop or terminate container port will be unmapped automatically.

You can take any port number in host but container port should be default. (80=80,81=80)

To create and run container but don't go inside container and map host and container port.

**#docker run -td --name webserver -p 80:80 centos**

To check map port

**#docker port <cname>**

To go inside container

**#docker exec -it <cname> /bin/bash**

Now install webpage and create index document.

**Difference between docker attach and docker exec.**

Docker exce create a new process even if you come out of container using exit won’t stop container.

Docker attach connect to main process if you come out of container using exit it stop container.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**How to push docker image in docker hub?**

1) Create image of your containers.

2) Create account in docker hub.

3) Run command in ec2

**#docker login.**

4) Enter your docker hub username and password.

5)Create tag

**#docker tag <image name in your docker engine> <docker hub username>/<image name u want to keep in docker hub>**

6) Push image to docker hub

**#docker push <docker hub username>/<image name u want to keep in docker hub>**

If images is private in docker hub then you need to provide username and password using **#docker login** command then only you can pull that image.(access within company)

If you want to make image private and doctor hub click the image go to setting make it private

To logout from docker hub

**#docker logout**

Stop all running containers

**#docker stop $(docker ps -a -q)**

Delete all stopped container

**#docker rm $(docker ps -a -q)**

Delete all images

**#docker rmi -f $(docker images -q)**