Docker

Docker version 20.10.25
DockerHub Username: jaiswal4docker

What is Docker Architecture & Container?

- ➔ Docker is an open-source centralized Platform designed to create, deploy and run applications.
- ➔ Docker uses containers on the host O.S to run applications. It allows applications to use the same linux kernel as a system on the host computer, rather than creating a whole virtual O.S.
- ➔ We can install docker on any O.S but the Docker engine runs natively on Linux distribution.
- ➔ Docker is written in 'go' language.
- ➔ Docker is a tool that performs O.S Level Virtualization, Also known as Containerization.
- ➔ Before Docker, Many users faced the problem that a particular code was running in the developer's system but not in the user's system.
- ➔ Docker was first released in March 2013. It was developed by Solomon Hykes and Sebastian Pahl.
- ➔ Docker is a set of Platform as a Service that uses O.S Level Virtualization. Whereas VMware uses Hardware Level Virtualization.

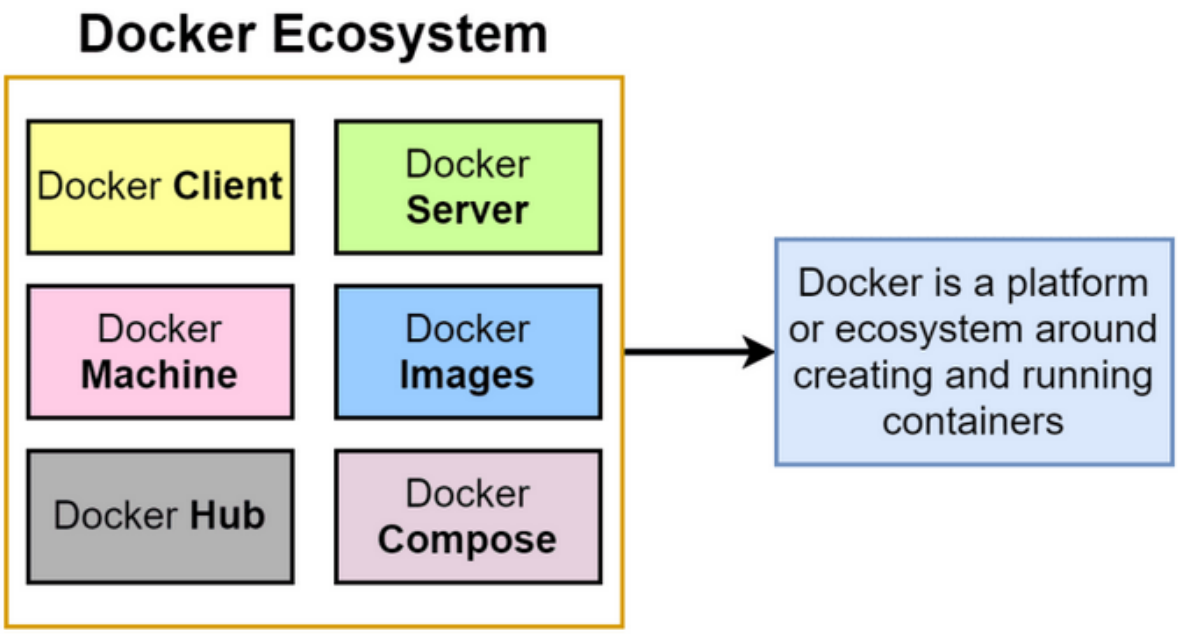
Advantages of Docker:

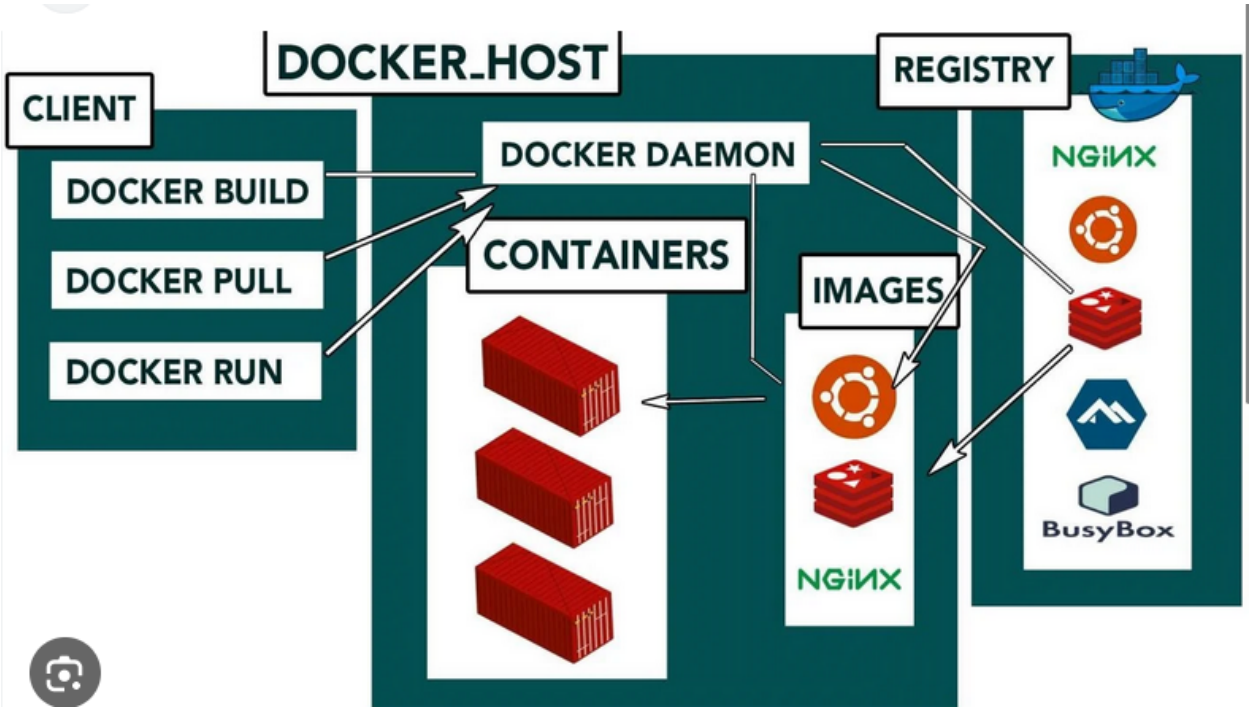
- ➢ No pre-allocation of RAM.
- ➢ CI Efficiency : Docker enables you to build a container image and use that same image across every step of the deployment process.
- ➢ Less Cost.
- ➢ It is light in weight.
- ➢ It can run on physical H/w or Virtual H/w or on Cloud.
- ➢ You can re-use the image.
- ➢ It took very less time to create the container.

Disadvantages of Docker

- ➢ Docker is not a good system for applications that require a rich GUI.
- ➢ Difficult to manage large amounts of Containers.

- Docker does not provide Cross-Platform compatibility means if an application is designed on windows then it can't run on Linux or vice-versa.
- Docker is suitable when the development O.S and Testing O.S are the same. If the O.S is different, We should use a VM.
- No solution for Data Recovery & Backup.





Docker Daemon/Engine:

- ➔ Docker Daemon runs on the Host O.S.
- ➔ It is responsible for running containers to manage docker services.
- ➔ Docker Daemon can communicate with other daemon.

Docker Client:

- ➔ Docker users can interact with docker daemon through a client.
- ➔ Docker client uses the commands and Rest API to communicate with the docker daemon.
- ➔ When a client runs any server command on the docker client terminal, the client terminal sends these docker commands to the docker daemon.
- ➔ It is possible for docker client to communicate with more than one daemon.

Docker Host:

- ➔ Docker Host is used to provide an environment to execute and run applications. It contains the docker daemon, images, containers, networks and storages.

Docker Hub/ Registry:

- ➔ Docker registry manages and stores the docker images.
- ➔ There are two types of registries in docker.
    1. Public Registry: Public registry is also called as docker hub.
    2. Private Registry: It is used to share images within enterprise.

Docker Images:

- ➔ Docker images are the read only binary templates used to create docker containers.
- ➔ Single file with all dependencies and configuration required to run a program.

- ➔ Way to create an Images
    1. Take image from docker hub.
    2. Create image from docker file.
    3. Create images from existing docker containers.

Docker Container:

- ➔ Container holds the entire packages that are needed to run the application.
- ➔ In other words, We can say that, The image is a template and the container is a copy of the template.
- ➔ Container is like a virtual Machine.
- ➔ Images become containers when they run on docker engine.

Basic commands in Docker

>> To see all images present in your local machine.
docker images

>> To find out images in docker hub.
docker search jenkins

```
root@ip-172-31-34-108:/home/ec2-user
[root@ip-172-31-34-108 ec2-user]# docker search jenkins
NAME                          DESCRIPTION                                     STARS     OFFICIAL   AUTOMATED
jenkins                       DEPRECATED; use "jenkins/jenkins:lts" instead   5657      [OK]
jenkins/jenkins               The leading open source automation server       3679
jenkins/jnlp-slave            a Jenkins agent which can connect to Jenkins…   157                  [OK]
jenkins/inbound-agent                                                         116
bitnami/jenkins               Bitnami Docker Image for Jenkins                65                   [OK]
jenkins/agent                                                                 62
jenkins/slave                 base image for a Jenkins Agent, which includ…   49                   [OK]
jenkins/ssh-agent             Docker image for Jenkins agents connected ov…   44
jenkins/ssh-slave             A Jenkins slave using SSH to establish conne…   39                   [OK]
jenkins/jnlp-agent-docker                                                     9
jenkins/jnlp-agent-maven      A JNLP-based agent with Maven 3 built in        9
jenkins/evergreen             An automatically self-updating Jenkins distr…   5
jenkins/pct                   Plugin Compat Tester - no longer published a…   5                    [OK]
jenkins/jnlp-agent-python     A JNLP-based agent with Python built in         3
jenkins/jenkins-experimental  Experimental images of Jenkins. These images…   3                    [OK]
jenkins/jnlp-agent-coresdk                                                    2
jenkins/jnlp-agent-alpine                                                     2
jenkins/jnlp-agent-node                                                       1
jenkins/jenkinsfile-runner    Jenkinsfile Runner packages                     1
jenkins/core-pr-tester        Docker image for testing pull-requests sent …   1
rancher/jenkins-jenkins                                                       1
jenkins/ath                   Jenkins Acceptance Test Harness                 1                    [OK]
jenkins/jnlp-agent-ruby                                                       1
rancher/jenkins-jnlp-slave                                                    0
rancher/jenkins-slave         Jenkins Build Slave                             0                    [OK]
[root@ip-172-31-34-108 ec2-user]#
```

>> To download images from docker hub to local machine.
docker pull jenkins/jenkins

```
root@ip-172-31-34-108:/home/ec2-user
[root@ip-172-31-34-108 ec2-user]# docker pull jenkins/jenkins
Using default tag: latest
latest: Pulling from jenkins/jenkins
012c0b3e998c: Pull complete
4b7f313cfec5: Pull complete
31d10e10fe4e: Pull complete
badc06ac8773: Pull complete
6fb10ca0446d: Pull complete
71a40217575c: Pull complete
1ba0db1f5cb7: Pull complete
f0c15308c44e: Pull complete
2fd6c9dfea90: Pull complete
e871ed2616ce: Pull complete
3f834d0be91b: Pull complete
cd0b14e7c3fa: Pull complete
b80e02465c88: Pull complete
Digest: sha256:8eadd547b5687de4682b93a0e84653545b94277cac4a96ca39867e00d8aafeea
Status: Downloaded newer image for jenkins/jenkins:latest
docker.io/jenkins/jenkins:latest
[root@ip-172-31-34-108 ec2-user]#
```

docker images

```
[root@ip-172-31-34-108 ec2-user]# docker images
REPOSITORY        TAG       IMAGE ID       CREATED         SIZE
jenkins/jenkins   latest    c9101035cede   43 hours ago    478MB
```

>> To give a name to the container.
docker run -it --name bhupinder ubuntu /bin/bash

```
root@0a724340a05e: /
[root@ip-172-31-34-108 ec2-user]# docker run -it --name bhupinder  ubuntu /bin/bash
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
445a6a12be2b: Pull complete
Digest: sha256:aabed3296a3d45cede1dc866a24476c4d7e093aa806263c27ddaadbdce3c1054
Status: Downloaded newer image for ubuntu:latest
```

Here bhupinder is container name
-it means Interactive mode terminal.

docker ps -a

```
[root@ip-172-31-34-108 ec2-user]# docker ps -a
CONTAINER ID   IMAGE     COMMAND       CREATED          STATUS                       PORTS     NAMES
0a724340a05e   ubuntu    "/bin/bash"   54 seconds ago   Exited (127) 10 seconds ago            bhupinder
[root@ip-172-31-34-108 ec2-user]#
```

>> To check service is start or not
service docker status

>> To start container
docker start bhupinder

```
root@ip-172-31-34-108:/home/ec2-user
[root@ip-172-31-34-108 ec2-user]# docker ps -a
CONTAINER ID   IMAGE     COMMAND       CREATED          STATUS                       PORTS     NAMES
0a724340a05e   ubuntu    "/bin/bash"   2 minutes ago    Exited (127) 2 minutes ago             bhupinder
[root@ip-172-31-34-108 ec2-user]#
[root@ip-172-31-34-108 ec2-user]#
[root@ip-172-31-34-108 ec2-user]#
[root@ip-172-31-34-108 ec2-user]# docker start bhupinder
bhupinder
[root@ip-172-31-34-108 ec2-user]#
[root@ip-172-31-34-108 ec2-user]#
[root@ip-172-31-34-108 ec2-user]# docker ps -a
CONTAINER ID   IMAGE     COMMAND       CREATED          STATUS          PORTS     NAMES
0a724340a05e   ubuntu    "/bin/bash"   3 minutes ago    Up 5 seconds              bhupinder
[root@ip-172-31-34-108 ec2-user]#
```

>> To go inside container
docker attach bhupinder

>> To see all containers
docker ps -a

>> To see only running containers
docker ps

ps means process status

>> To stop container
docker stop bhupinder

>> To delete container
docker rm bhupinder

Docker File Components & Diff Command

- Login into your AWS account and start your EC2 Instance. Access it from the putty.
- Now we have to create a container from our one image.
- Therefore, Create one container first.
- # docker run -it --name bhupicontainer ubuntu /bin/bash
- # cd tmp/
- Now, Create one file inside this tmp directory
- # touch myfile
- # cd ..
- # exit
- Now if you want to see the difference between the base image & changes on it then
- # docker diff bhupicontainer
- o/p ->

  C /tmp
  A /tmp/myfile
  C /root
  A /root/.bash_history

Here
A= Append or Addition
C= Change
D= Deletion

- Now, Create image of this container
- # docker commit bhupicontainer updateimage
- # docker images

```
[root@ip-172-31-43-66 ec2-user]# docker run -it --name bhupicontainer ubuntu /bin/bash
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
445a6a12be2b: Pull complete
Digest: sha256:aabed3296a3d45cede1dc866a24476c4d7e093aa806263c27ddaadbdce3c1054
Status: Downloaded newer image for ubuntu:latest
root@ccfc2a1fe82c:/#
root@ccfc2a1fe82c:/#
root@ccfc2a1fe82c:/# ls
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@ccfc2a1fe82c:/# cd tmp/
root@ccfc2a1fe82c:/tmp# touch myfile
root@ccfc2a1fe82c:/tmp# cd ..
root@ccfc2a1fe82c:/# exit
exit
[root@ip-172-31-43-66 ec2-user]# docker diff bhupicontainer
C /tmp
A /tmp/myfile
C /root
A /root/.bash_history
[root@ip-172-31-43-66 ec2-user]#
[root@ip-172-31-43-66 ec2-user]# docker commit bhupicontainer updateimage
sha256:ca9dcabcb25a2e90bc864539201333040577b70b264abcb69696e242ef11a60b
[root@ip-172-31-43-66 ec2-user]#
[root@ip-172-31-43-66 ec2-user]# docker images
REPOSITORY    TAG       IMAGE ID       CREATED          SIZE
updateimage   latest    ca9dcabcb25a   44 seconds ago   77.8MB
ubuntu        latest    c6b84b685f35   4 weeks ago      77.8MB
[root@ip-172-31-43-66 ec2-user]#
```

- Now, create container from this image
- # docker run -it --name rajcontainer updateimage /bin/bash
- # ls
- # cd tmp/
- # ls
- o/p-> myfile { you will get all files back}

```
root@8cd90fa9f9cb: /
[root@ip-172-31-43-66 ec2-user]# docker images
REPOSITORY    TAG       IMAGE ID       CREATED              SIZE
updateimage   latest    ca9dcabcb25a   About a minute ago   77.8MB
ubuntu        latest    c6b84b685f35   4 weeks ago          77.8MB
[root@ip-172-31-43-66 ec2-user]#
[root@ip-172-31-43-66 ec2-user]#
[root@ip-172-31-43-66 ec2-user]# docker run -it --name rajputcontainer updateimage /bin/bash
root@8cd90fa9f9cb:/# ll tmp/
total 0
drwxrwxrwt. 1 root root 20 Sep 14 11:06 ./
drwxr-xr-x. 1 root root  6 Sep 14 11:11 ../
-rw-r--r--. 1 root root  0 Sep 14 11:06 myfile
root@8cd90fa9f9cb:/#
```

Dockerfile:

- ➔ Dockerfile is basically a text file. It contains some instructions.
- ➔ Automation of Docker image creation.

Docker Components:

FROM: For base image. This command must be on top of the dockerfile.

RUN: To execute commands, It will create a layer in image.

MAINTAINER: Author/ Owner/ Description

COPY: Copy files from local system (docker vm).
We need to provide a source, description.
(We can't download file from internet and any remote repo)

ADD:
Similar to COPY but, It provides a feature to download files from the internet, also we extract files at docker image side.

EXPOSE:
To expose ports such as port 8080 for tomcat, port 80 for nginx etc.

WORKDIR:
To set a working directory for a container.

CMD:
Execute commands but during container creation.

ENTRYPOINT:
Similar to CMD, but has higher priority over CMD, first commands will be executed by ENTRYPOINT only.

ENV:
Environment Variables.

ARG:
ARG is only available during the build of a Docker image (RUN etc), not after the image is created and containers are started from it (ENTRYPOINT, CMD). You can use ARG values to set ENV values to work around that.

Dockerfile

- ➔ Create a file named Dockerfile.
- ➔ Add instructions in Dockerfile.
- ➔ Build Dockerfile to create image.
- ➔ Run image to create container.

```
# vi Dockerfile
FROM ubuntu
RUN echo “Technical Guftgu” > /tmp/testfile
```

To Create image out of Dockerfile.
docker build -t test .

```
[root@ip-172-31-43-66 ec2-user]# vi Dockerfile
[root@ip-172-31-43-66 ec2-user]#
[root@ip-172-31-43-66 ec2-user]#
[root@ip-172-31-43-66 ec2-user]# docker build -t test .
Sending build context to Docker daemon   7.68kB
Step 1/2 : FROM ubuntu
 ---> c6b84b685f35
Step 2/2 : RUN echo "Welcome to Technical Guftgu" > /tmp/testfile
 ---> Running in 4f9707d96d10
Removing intermediate container 4f9707d96d10
 ---> 5e8d95b327fd
Successfully built 5e8d95b327fd
Successfully tagged test:latest
```

docker ps -a
o/p -> docker images

Now, Create container from the above image.

docker run -it --name testcontainer test /bin/bash

cat tmp/testfile

```
[root@ip-172-31-43-66 ec2-user]# docker run -it --name testcontainer test /bin/b
ash
root@d06e39c64df0:/# ls
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@d06e39c64df0:/# cat tmp/testfile
Welcome to Technical Guftgu
```

vi Dockerfile

```
FROM ubuntu
WORKDIR /tmp
RUN echo “Subscribe to technical Guftgu” > /tmp/testfile
ENV myname bhupinder
COPY testfile1 /tmp
ADD test.tar.gz /tmp
```

touch testfile1
touch test
tar -cvf test.tar test
gzip test.tar
rm -fe test

docker build -t newimgage .

```
[root@ip-172-31-43-66 ec2-user]# docker build -t newimgage .
Sending build context to Docker daemon  9.216kB
Step 1/6 : FROM ubuntu
 ---> c6b84b685f35
Step 2/6 : WORKDIR /tmp
 ---> Using cache
 ---> 62595b0832f4
Step 3/6 : RUN echo “Subscribe to technical Guftgu” > /tmp/testfile
 ---> Using cache
 ---> 79ef2048109b
Step 4/6 : ENV myname bhupinder
 ---> Using cache
 ---> ecda7e262dbc
Step 5/6 : COPY testfile1 /tmp
 ---> c504d93871ac
Step 6/6 : ADD test.tar.gz /tmp
 ---> 95f24971a339
Successfully built 95f24971a339
Successfully tagged newimgage:latest
```

docker images

docker run -it --name newcontainer newimgage /bin/bash

```
root@a37bbdd61d4c: /tmp
[root@ip-172-31-43-66 ec2-user]# docker run -it --name newcontainer newimgage /bin/bash
root@a37bbdd61d4c:/tmp#
root@a37bbdd61d4c:/tmp#
root@a37bbdd61d4c:/tmp#
root@a37bbdd61d4c:/tmp# ll
total 4
drwxrwxrwt. 1 root root 18 Sep 14 12:08 ./
drwxr-xr-x. 1 root root  6 Sep 14 12:12 ../
-rw-r--r--. 1 root root  0 Sep 14 12:07 test
-rw-r--r--. 1 root root 36 Sep 14 12:05 testfile
-rw-r--r--. 1 root root  0 Sep 14 12:06 testfile1
root@a37bbdd61d4c:/tmp# cat testfile1
root@a37bbdd61d4c:/tmp# cat testfile
"Subscribe to technical Guftgu"
```

echo $myname
bhupinder

Docker Volume & How to Share it

- ➔ Volume is simply a directory inside our container.
- ➔ Firstly, We have to declare this directory as a volume and then share the volume.
- ➔ Even if we stop the container, still we can access volume.
- ➔ Volume will be created in one container.
- ➔ You can declare a directory as a volume only while creating a container.
- ➔ You can't create volume from an existing container.
- ➔ You can share one volume across any number of containers.
- ➔ Volume will not be included when you update an image.
- ➔ You can mapped volume in two ways.
    1. Container to Container
    2. Container to host (vice versa)

Benefits of Volume

- ➔ Decoupling Container from storage.
- ➔ Share volume among different Containers.
- ➔ Attach Volume to Containers.
- ➔ On deleting Container Volume does not delete.

Lab: Creating Volume from Dockerfile

Create a Dockerfile and write

```
FROM ubuntu
VOLUME ["/myvolume1"]
```

Then Create image from this Dockerfile

```
# docker build -t myimage .
```

```
[root@ip-172-31-34-125 ec2-user]# vim Dockerfile
[root@ip-172-31-34-125 ec2-user]# docker build -t myimage .
Sending build context to Docker daemon   7.68kB
Step 1/2 : FROM ubuntu
latest: Pulling from library/ubuntu
445a6a12be2b: Pull complete
Digest: sha256:aabed3296a3d45cede1dc866a24476c4d7e093aa806263c27ddaadbdce3c1054
Status: Downloaded newer image for ubuntu:latest
 ---> c6b84b685f35
Step 2/2 : VOLUME ["/myvolume1"]
 ---> Running in 4220bc5011a9
Removing intermediate container 4220bc5011a9
 ---> 207b76f54d96
Successfully built 207b76f54d96
Successfully tagged myimage:latest
[root@ip-172-31-34-125 ec2-user]#
```

Now, Create a Container from this image & run

```
# docker run -it --name container1 myimage /bin/bash
```

```
root@ip-172-31-34-125:/home/ec2-user
[root@ip-172-31-34-125 ec2-user]# docker run -it --name container1 myimage /bin/bash
root@d1baf2f7d2da:/# ls
bin   dev  home  lib32  libx32  mnt        opt   root  sbin  sys  usr
boot  etc  lib   lib64  media   myvolume1  proc  run   srv   tmp  var
root@d1baf2f7d2da:/# cd myvolume1/
root@d1baf2f7d2da:/myvolume1# touch filex filey filez
root@d1baf2f7d2da:/myvolume1# ls
filex  filey  filez
root@d1baf2f7d2da:/myvolume1# exit
exit
[root@ip-172-31-34-125 ec2-user]#
```

Now, do ls, you can see myvolume1

Now, Share Volume with another Container

Container < — > Container

docker run -it --name container2 --privileged=true --volumes-from container1 ubuntu /bin/bash

```
root@d3929e891f1f: /myvolume1
[root@ip-172-31-34-125 ec2-user]# docker run -it --name container2 --privileged=true
--volumes-from container1 ubuntu /bin/bash
root@d3929e891f1f:/# ls
bin   dev  home  lib32  libx32  mnt        opt   root  sbin  sys  usr
boot  etc  lib   lib64  media   myvolume1  proc  run   srv   tmp  var
root@d3929e891f1f:/# cd myvolume1/
root@d3929e891f1f:/myvolume1# ls
filex  filey  filez
root@d3929e891f1f:/myvolume1#
```

Now after creating container2, myvolume1 is visible. Whatever you do in one volume, Can see from the other volume.

touch /myvolume1/samplefile
docker start container1
docker attach container1
ls /myvolume1

You can see samplefile here
exit

Now, Try to create volume by using Command

docker run -it --name container3 -v /volume2 ubuntu /bin/bash
ls
cd /voume2

Now, Create one file cont3file and exit
touch cont3file
exit

Now, create one more container, and share volume2

docker run -it --name container4 --privileged=true --volumes-from container3 ubuntu /bin/bash

Now, you are inside container, do ls you can see volume2
Now, Create one file inside this volume and then check in container 3, you can see that file.

cd /volume2
touch sample1 sample2 sample3

```
[root@ip-172-31-34-125 ec2-user]# docker run -it --name container3 -v /volume2 u
buntu /bin/bash
root@bc388881f951:/# ls
bin   dev  home  lib32  libx32  mnt  proc  run   srv  tmp  var
boot  etc  lib   lib64  media   opt  root  sbin  sys  usr  volume2
root@bc388881f951:/# cd volume2/
root@bc388881f951:/volume2# ls
root@bc388881f951:/volume2# touch vol1 vol2 vol3
root@bc388881f951:/volume2# ls
vol1  vol2  vol3
root@bc388881f951:/volume2# exit
exit
[root@ip-172-31-34-125 ec2-user]# docker run -it --name container4 --privileged=true --vo
lumes-from container3 ubuntu /bin/bash
root@076412715dd7:/# ls
bin   dev  home  lib32  libx32  mnt  proc  run   srv  tmp  var
boot  etc  lib   lib64  media   opt  root  sbin  sys  usr  volume2
root@076412715dd7:/# cd volume2/
root@076412715dd7:/volume2# ls
vol1  vol2  vol3
root@076412715dd7:/volume2# touch vol4
root@076412715dd7:/volume2# ls
vol1  vol2  vol3  vol4
```

```
[root@ip-172-31-34-125 ec2-user]# docker start container3
container3
[root@ip-172-31-34-125 ec2-user]# docker attach container3
root@bc388881f951:/# ls
bin   dev  home  lib32  libx32  mnt  proc  run   srv  tmp  var
boot  etc  lib   lib64  media   opt  root  sbin  sys  usr  volume2
root@bc388881f951:/# cd volume2/
root@bc388881f951:/volume2# ls
vol1  vol2  vol3  vol4
root@bc388881f951:/volume2#
```

Volume Share (Host-Container)

➔ Verify files in /home/ec2-user

- # docker run -it --name hostcont -v /home/ec2-user:/rajput --privileged=true ubuntu /bin/bash
- # cd /rajput
- Do ls , now you can see all files of host machine
- # touch rajputfile
- # exit
- Now, check in EC2 machine, you can see this file

Some other commands

- # docker volume ls

```
[root@ip-172-31-34-125 ec2-user]# docker volume ls
DRIVER    VOLUME NAME
local     34152ff0b7effb62b4649fbf7bcd6d9b84744bc298711bd5c63010c6095455f8
local     d74eacde5684d6cef052aa4b8d8372608914b085b9d052722bd0d5199b241754
[root@ip-172-31-34-125 ec2-user]#
```

- # docker volume create < volume name>
- # docker volume rm <volume name>

```
[root@ip-172-31-34-125 ec2-user]# docker volume create prince
prince
[root@ip-172-31-34-125 ec2-user]# docker volume ls
DRIVER    VOLUME NAME
local     34152ff0b7effb62b4649fbf7bcd6d9b84744bc298711bd5c63010c6095455f8
local     d74eacde5684d6cef052aa4b8d8372608914b085b9d052722bd0d5199b241754
local     prince
[root@ip-172-31-34-125 ec2-user]# docker volume rm prince
prince
[root@ip-172-31-34-125 ec2-user]# docker volume ls
DRIVER    VOLUME NAME
local     34152ff0b7effb62b4649fbf7bcd6d9b84744bc298711bd5c63010c6095455f8
local     d74eacde5684d6cef052aa4b8d8372608914b085b9d052722bd0d5199b241754
```

- # docker volume prune {It removed all unused docker volume}
- # docker volume inspect <volume name>

```
[root@ip-172-31-34-125 ec2-user]# docker volume prune
WARNING! This will remove all local volumes not used by at least one container.
Are you sure you want to continue? [y/N] y
Total reclaimed space: 0B
[root@ip-172-31-34-125 ec2-user]# docker volume ls
DRIVER    VOLUME NAME
local     34152ff0b7effb62b4649fbf7bcd6d9b84744bc298711bd5c63010c6095455f8
local     d74eacde5684d6cef052aa4b8d8372608914b085b9d052722bd0d5199b241754
[root@ip-172-31-34-125 ec2-user]#
[root@ip-172-31-34-125 ec2-user]# docker volume inspect 34152ff0b7effb62b4649fbf7bcd6d9b84744bc298711bd5c63010c6095455f8
[
    {
        "CreatedAt": "2023-09-14T16:31:04Z",
        "Driver": "local",
        "Labels": null,
        "Mountpoint": "/var/lib/docker/volumes/34152ff0b7effb62b4649fbf7bcd6d9b84744bc298711bd5c63010c6095455f8/_data",
        "Name": "34152ff0b7effb62b4649fbf7bcd6d9b84744bc298711bd5c63010c6095455f8",
        "Options": null,
        "Scope": "local"
    }
]
```

- # docker container inspect <Container name>

Docker Port Expose

Lab:

- ➔ Login into AWS account. Create one Linux instance.
- ➔ Now go to putty -> login as ec2-user
- ➔ # sudo su
- ➔ # yum update -y
- ➔ # yum install docker -y
- ➔ # systemctl start docker
- ➔ # systemctl status docker
- ➔ # docker run -it --name techserver -p 80:80 ubuntu
- ➔ # docker ps
- ➔ # docker port techserver
- ➔ o/p: 80/tcp -> 0.0.0.0:80
  80/tcp -> :::80
- ➔ # docker exec -it techserver /bin/bash
- ➔ # apt-get update
- ➔ # apt-get install apache2 -y
- ➔ # cd /var/www/html/
- ➔ # echo “Welcome to technical guftgu” > index.html
- ➔ # service apache2 start
- ➔ Now, Go to browser and paste ec2 public ip to access the webserver

- ➔ # docker run -td --name myjenkins -p 8080:8080 jenkins/jenkins
- ➔ Please enable 8080 custom port in security group of ec2

>> Difference between **docker attach** and **docker exec**?

docker exec creates a new process in the container’s environment while **docker attach** just connects the standard Input/Output of the main process inside the container to corresponding standard input/output error of the current terminal.

docker exec is specifically for running new things in an already started container, be it a shell or some other process.

>> What is the difference between **expose** and **publish** a docker?

Basically you have three options:-

1. Neither specify **expose** nor **-p**
2. Only specify **expose**

3. Specify **expose** and **-p**

1. If you specify neither **expose** nor **-p** , the service in the container will only be available from inside the container itself.
2. If you **expose** a port, the service in the container is not accessible from outside docker, but from inside other docker containers, so this is good for inter-container communication.
3. If you **expose** and **-p** a port, the service in the container is accessible from anywhere, even outside docker.
4. If you do **-p** but do not **expose** docker does an implicit expose. This is because, If a port is open to the public, It is automatically also open to the other docker containers.
5. Hence **'-p'** includes expose.

How to push docker image in docker hub

- Go to the AWS account and create an EC2 instance. Now login as ec2-user via putty.
- # sudo su
- # yum update -y
- # yum install docker -y
- # service docker start
- # docker run -it --name container1 ubuntu /bin/bash
- Now create some files inside the container.
- Now create an image of this container.
- # docker commit container1 image1
- Now create account in hub.docker.com
- Now go to ec2 instance
- # docker login
- Enter your username and password.
- Now give a tag to your image.
- # docker tag image1 dockerid/newimage
- # docker push dockerid/newimage

```
[root@ip-172-31-36-248 ec2-user]# docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
ubuntu       latest    c6b84b685f35   4 weeks ago   77.8MB
[root@ip-172-31-36-248 ec2-user]#
[root@ip-172-31-36-248 ec2-user]# docker commit container1 image1
sha256:fa2e13c7baddad4850e7b2760955fa3923c50eeedc522a95f157959e628f724a
[root@ip-172-31-36-248 ec2-user]# docker images
REPOSITORY   TAG       IMAGE ID       CREATED         SIZE
image1       latest    fa2e13c7badd   5 seconds ago   77.8MB
ubuntu       latest    c6b84b685f35   4 weeks ago     77.8MB
[root@ip-172-31-36-248 ec2-user]# docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't
have a Docker ID, head over to https://hub.docker.com to create one.
Username: jaiswal4docker
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
[root@ip-172-31-36-248 ec2-user]# docker tag image1 jaiswal4docker/project1
[root@ip-172-31-36-248 ec2-user]# docker push jaiswal4docker/project1
Using default tag: latest
The push refers to repository [docker.io/jaiswal4docker/project1]
ff8c7afbb453: Pushed
dc0585a4b8b7: Mounted from library/ubuntu
latest: digest: sha256:ba39507aabe142c0b123438c6f5811e9f0b8fe399ad2d40cf9d9ce771
4f7c8a4 size: 736
[root@ip-172-31-36-248 ec2-user]#
```

- Now you can see this image in docker hub account.
- Now create one instance in the tokyo region and pull image from docker hub.
- # docker pull docekrid/newimage
-

```
[root@ip-172-31-45-57 ec2-user]# docker pull jaiswal4docker/project1
Using default tag: latest
latest: Pulling from jaiswal4docker/project1
445a6a12be2b: Pull complete
1003afc6e8d1: Pull complete
Digest: sha256:ba39507aabe142c0b123438c6f5811e9f0b8fe399ad2d40cf9d9ce7714f7c8a4
Status: Downloaded newer image for jaiswal4docker/project1:latest
docker.io/jaiswal4docker/project1:latest
[root@ip-172-31-45-57 ec2-user]#
```

- # docker run -it --name mycon dockerid/newimage /bin/bash
- # ls

```
[root@ip-172-31-45-57 ec2-user]# docker run -it --name mycont jaiswal4docker/project1 /bin/bash
root@3b0222876545:/#
root@3b0222876545:/#
root@3b0222876545:/# ls
bin  boot  dev  etc  file1  file2  home  lib  lib32  lib64  libx32  media  mnt  opt  proc  root  run  sbin  srv  sys  test1  test2
root@3b0222876545:/#
```

If we change project1 image private then we try to pull will get bellow error

```
[root@ip-172-31-45-57 ec2-user]# docker run -it --name mycont jaiswal4docker/pro
ject1 /bin/bash
docker: Error response from daemon: Conflict. The container name "/mycont" is al
ready in use by container "3b0222876545a8fbbe2b08073ee3cb450372edaf8f3957e62b4fa
ccb51e53941". You have to remove (or rename) that container to be able to reuse
that name.
See 'docker run --help'.
[root@ip-172-31-45-57 ec2-user]#
```

Now First login to docker hub then try to access the private image

```
[root@ip-172-31-45-57 ec2-user]# docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Dock
er ID, head over to https://hub.docker.com to create one.
Username: jaiswal4docker
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
[root@ip-172-31-45-57 ec2-user]# docker pull jaiswal4docker/project1
Using default tag: latest
latest: Pulling from jaiswal4docker/project1
Digest: sha256:ba39507aabe142c0b123438c6f5811e9f0b8fe399ad2d40cf9d9ce7714f7c8a4
Status: Image is up to date for jaiswal4docker/project1:latest
docker.io/jaiswal4docker/project1:latest
[root@ip-172-31-45-57 ec2-user]#
[root@ip-172-31-45-57 ec2-user]# docker images
REPOSITORY                TAG       IMAGE ID       CREATED          SIZE
jaiswal4docker/project1   latest    fa2e13c7badd   18 minutes ago   77.8MB
[root@ip-172-31-45-57 ec2-user]#
```

Some Important Commands

1. Stop all running containers
   # docker stop $(docker ps -a -q)

2. Delete all stopped containers
   # docker rm $(docker ps -1 -q)

3. Delete all images
   # docker rmi -f $(docker images -q)

```
[root@ip-172-31-45-57 ec2-user]#  docker rm $( docker ps -a -q)
3b0222876545
[root@ip-172-31-45-57 ec2-user]# docker rmi -f $( docker images -q)
Untagged: jaiswal4docker/project1:latest
Untagged: jaiswal4docker/project1@sha256:ba39507aabe142c0b123438c6f5811e9f0b8fe399ad2d40cf9
d9ce7714f7c8a4
Deleted: sha256:fa2e13c7baddad4850e7b2760955fa3923c50eeedc522a95f157959e628f724a
Deleted: sha256:8d48645dcb719f743b71eeea1822e4452c2630b2d489978e613d114fd44ab28e
Deleted: sha256:dc0585a4b8b71f7f4eb8f2e028067f88aec780d9ab40c948a8d431c1aeadeeb5
[root@ip-172-31-45-57 ec2-user]#
```

Notes Created by : Prince Jaiswal (princejaiswal0007@gmail.com)