# Chapter 3 : docker

==========

## What is docker and containerization..?

- Docker is an open source centralized platform designed to create, deploy and run applications.
- Docker uses container on the host os to run applications . It allows applications to use the some linux kernel as a system on host computer, rather than creating a whole virtual os
- We can install docker on any os but docker engine runs natively on linux distribution
- Docker written in 'go' language
- Docker is a tool that performs os level virtualization , also known as containerization.
- Before docker, many users faces the problem that a particular code is running in the developer's system but not in the user's system.
- Docker is a set of platform as a service that uses os level virtualization whereas vmware uses hardware level virtualization.

## Advantages of Dockers

- No pre-alloation 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 physial HW/virtual HW or on Cloud
- you can re-use the image
- It took very less time to create container.

## Disadvantages:

- Docker is not a good solution for application that requires rich rich GUI.
- Difficult to manage large amount of conatiners.
- Docker does not provide cross-platform compatibility means if an applidation is designed to run in a docker conatainer on windows, then it can't run on linux or vice-versa
- Docker is suitable when the development OA and testing OS are same if the os is different, we should use VM.
- No solution for data recovery and backup

## Docker Architecture:

- Docker Ecosystem:(Set of s/w or packages)

# Chapter 3 : docker

==========

## Components of Docker:

- Docker Daemon/Server/Docker engine:
- Docker daemon runs on the Host OS.
- It is responsible for running containers to manages docker services.
- Docker Daemon can communicate with other daemons.

## Docker client:

- Docker users can interact with docker daemon through a client.
- Docker client uses 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 the docker

  i. Public Registry: Public registry is also called as docker hub.

  ii. Private Registry: It is used to share images within the enterprise.

## Docker Images:

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

  Ways to create an Images

  1. Take image from docker hub
  2. Create image from docker file
  3. Create image from existing docker containers.

# Chapter 3 : docker

==========

## Docker Container:

- Container hold the entire packages that is needed to run the application.
- In other words, we can say that the image is a template and the container is a copy of that template.
- Container is like a virtual machine.
- Images becomes container when they run on docker engine.

  Image ---> Run ---> Container.

- Container is Layer file system (execute task one by one or layer by layer)

```
systemctl start docker                                  … to start docker service
systemctl status docker                                 … to check service is start or not
docker -v                                               … to check docker version
docker –version                                         … to check docker version
docker info


docker images                                           … to see all images present in your local
docker search Jenkins                                   …to find out images in docker hub
docker pull Jenkins                                     …to download image from dockerhub to local machine
docker run -it –name vishalk17 image_name:tag /bin/bash …to give name to the container
                                                          -i = interactive mode
                                                          -t = terminal

docker start container_id/name                          … to start container
docker stop container_id/name                           … to stop container
docker attach container_id/name                         … to go inside container
docker ps -a                                            … to see all containers
docker ps                                               … to see only running containers
                                                           ps: process status
docker rm container_id/name                             … to delete container
```

# Chapter 3 : docker

==========

## Create Image from container :

## Terminal logs

```
[root@ip-172-31-2-0 ~]# docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES

>>>>>>>>>> run docker ubuntu image and give name to it that is ubuntu-vish <<<<<<<<<<<<

[root@ip-172-31-2-0 ~]# docker run --name vish-ubuntu -it ubuntu /bin/bash
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
e96e057aae67: Pull complete
Digest: sha256:4b1d0c4a2d2aaf63b37111f34eb9fa89fa1bf53dd6e4ca954d47caebca4005c2
Status: Downloaded newer image for ubuntu:latest

>>>>>>>>>> check docker images <<<<<<<<<<<<

root@aee3aca7d31d:/# [root@ip-172-31-2-0 ~]# docker images
REPOSITORY   TAG       IMAGE ID       CREATED      SIZE
ubuntu       latest    a8780b506fa4   8 days ago   77.8MB

[root@ip-172-31-2-0 ~]# docker ps -a
CONTAINER ID   IMAGE     COMMAND       CREATED          STATUS          PORTS     NAMES
aee3aca7d31d   ubuntu    "/bin/bash"   20 seconds ago   Up 19 seconds             vish-ubuntu

>>>>>>> Enter in container and create files <<<<<<<<<<

[root@ip-172-31-2-0 ~]# docker attach aee3aca7d31d

root@aee3aca7d31d:/# ls
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var

root@aee3aca7d31d:/# cd /mnt

root@aee3aca7d31d:/mnt# touch asd asd f asd fasd f asdfasd f asdfa f as fa sf as fas  fa f

root@aee3aca7d31d:/mnt# ls
as  asd  asdfa  asdfasd  f  fa  fas  fasd  sf

root@aee3aca7d31d:/mnt# exit
exit

>>>>>>> check difference in base image and current container change <<<<<<<<<<

[root@ip-172-31-2-0 ~]# docker diff vish-ubuntu
C /root
A /root/.bash_history
C /mnt
```

# Chapter 3 : docker

==========

```
A /mnt/fasd
A /mnt/as
A /mnt/asdfa
A /mnt/asdfasd
A /mnt/fa
A /mnt/fas
A /mnt/asd
A /mnt/f
A /mnt/sf
```

A= add /append

D = delete

C =change

>>>>>>> **make new image from current container , new image contains all files that were created** <<<<<<<<<<<
>>>>>>> **docker commit old-image new-update-image-from-old-image** <<<<<<<<<<<

```
[root@ip-172-31-2-0 ~]# docker commit vish-ubuntu updateimage
sha256:01afa76111d58ef1f4370452d9ce7760ba6f13514b43e837d274f2f67e570e91
```

>>>>>>> **check available images** <<<<<<<<<<<

```
[root@ip-172-31-2-0 ~]# docker images
REPOSITORY    TAG       IMAGE ID       CREATED         SIZE
updateimage   latest    01afa76111d5   5 seconds ago   77.8MB
ubuntu        latest    a8780b506fa4   8 days ago      77.8MB
```

>>>>>>> **running new container from updated image and give it name "new container"** <<<<<<<<<<<
>>>>>>> **check whether in newcontainer (was created from update image) all files are exit or not** <<<<<<<<<<<

```
[root@ip-172-31-2-0 ~]# docker run -it --name newcontainer updateimage

root@688b0a33bcf5:/# ls
bin   dev  home  lib32  libx32  mnt  proc  run   srv  tmp  var
boot  etc  lib   lib64  media   opt  root  sbin  sys  usr

root@688b0a33bcf5:/# cd /mnt

root@688b0a33bcf5:/mnt# ls
as  asd  asdfa  asdfasd  f  fa  fas  fasd  sf

root@688b0a33bcf5:/mnt#
```

>>>>>>> **check all running and non running container** <<<<<<<<<<<

```
[root@ip-172-31-2-0 ~]# docker ps -a
CONTAINER ID   IMAGE         COMMAND       CREATED         STATUS                      PORTS     NAMES
688b0a33bcf5   updateimage   "/bin/bash"   7 minutes ago   Up 7 minutes                          newcontainer
aee3aca7d31d   ubuntu        "/bin/bash"   10 minutes ago  Exited (0) 8 minutes ago              vish-ubuntu
[root@ip-172-31-2-0 ~]#
```

Chapter 3 : docker

==========

## Docker-file creation

- dockerfile is basically a text file it contains some set of instruction
- automation of docker image creation

**Docker Components:**

FROM : for base image thus command must be on top of dockerfile

RUN : to execute commands it will create a layer in image

LABEL : Labeling like EMAIL, AUTHOR, etc.

MAINTAINER : author/owner/description

COPY: copy files from local system(docker vm). We need to provide source ,destination. (we cant download file from internet and any remote repo)

ADD : similar to copy but it provides a feature to download files from internet also we extract file at docker image side

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

WORKDIR : to set 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 flag

ENV : Environment Variables

```
# comments

FROM base-image
LABEL key=value
RUN command
COPY src dest
ADD src.tar dst
WORKDIR dir
ENTRYPOINT cmd
CMD args
```

# Chapter 3 : docker

==========

**Example 01:**

```
# pull image from docker hub with tag
# FROM image_name:tag_name

FROM ubuntu:latest

# add some text to sample.txt using RUN
# RUN command

RUN echo " hello hacker " >> /mnt/sample.txt
```

**Output**

```
[root@ip-172-31-11-147 ~]# ls
Dockerfile
[root@ip-172-31-11-147 ~]# vi *
[root@ip-172-31-11-147 ~]# docker build -t vish-ubuntu .
Sending build context to Docker daemon  32.26kB
Step 1/2 : FROM ubuntu:latest
latest: Pulling from library/ubuntu
e96e057aae67: Pull complete
Digest: sha256:4b1d0c4a2d2aaf63b37111f34eb9fa89fa1bf53dd6e4ca954d47caebca4005c2
Status: Downloaded newer image for ubuntu:latest
 ---> a8780b506fa4
Step 2/2 : RUN echo " hello hacker " >> /mnt/sample.txt
 ---> Running in f433eb6a6f0a
Removing intermediate container f433eb6a6f0a
 ---> 2e6f27ccd20a
Successfully built 2e6f27ccd20a
Successfully tagged vish-ubuntu:latest
[root@ip-172-31-11-147 ~]# docker images
REPOSITORY    TAG       IMAGE ID       CREATED         SIZE
vish-ubuntu   latest    2e6f27ccd20a   3 seconds ago   77.8MB
ubuntu        latest    a8780b506fa4   10 days ago     77.8MB
[root@ip-172-31-11-147 ~]# docker run -it vish-ubuntu:latest /bin/bash
root@843108449f63:/# ls
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@843108449f63:/# cd /mnt
root@843108449f63:/mnt# ls
sample.txt
root@843108449f63:/mnt# cat *
 hello hacker
root@843108449f63:/mnt#
```

| | |
|---|---|
| docker build -t image_name_you_like_to_give:tag | (command use to build image from Dockerfile)<br>-t = tag |

# Chapter 3 : docker

==========

**Example 2:**

**vi Dockerfile**

```
# pull image from docker hub with tag
# FROM image_name:tag_name

FROM ubuntu:latest

# set working directory to /mnt
# whatever work we are going to do in this dir. only for now

WORKDIR /mnt

# add some text to sample.txt using RUN
# RUN linux_command

RUN echo " hello hacker " >> sample.txt

# set enviroment variables
# ENV key value

ENV who_is_hacker vishal

# write ENV to the xyz.txt file
# RUN linux_command

RUN echo "$who_is_hacker" >> xyz.txt

# copy file1.txt to /mnt
# COPY SRC DEST

COPY file1.txt /mnt/

# copy and extract *.tar.gz to /mnt

ADD data.tar.gz /mnt/
```

# Chapter 3 : docker

==========

```
4. docker                                   5. docker
[root@ip-172-31-11-147 ~]# pwd
/root
[root@ip-172-31-11-147 ~]# ls
Dockerfile
[root@ip-172-31-11-147 ~]# touch file1.txt
[root@ip-172-31-11-147 ~]# touch data.txt
[root@ip-172-31-11-147 ~]# tar -cvf data.tar data.txt
data.txt
[root@ip-172-31-11-147 ~]# ls
data.tar  data.txt  Dockerfile  file1.txt
[root@ip-172-31-11-147 ~]# gzip data.tar
[root@ip-172-31-11-147 ~]# ls
data.tar.gz  data.txt  Dockerfile  file1.txt
[root@ip-172-31-11-147 ~]# rm  data.txt
rm: remove regular empty file 'data.txt'? y
[root@ip-172-31-11-147 ~]# ls
data.tar.gz  Dockerfile  file1.txt
[root@ip-172-31-11-147 ~]# vi Docker*
[root@ip-172-31-11-147 ~]#
[root@ip-172-31-11-147 ~]# docker build -t ubuntu:v01 .
Sending build context to Docker daemon  46.08kB
Step 1/7 : FROM ubuntu:latest
latest: Pulling from library/ubuntu
Digest: sha256:4b1d0c4a2d2aaf63b37111f34eb9fa89fa1bf53dd6e4ca954d47caebca4005c2
Status: Downloaded newer image for ubuntu:latest
 ---> a8780b506fa4
Step 2/7 : WORKDIR /mnt
 ---> Running in 028005b9a720
Removing intermediate container 028005b9a720
 ---> 8221ff211fc5
Step 3/7 : RUN echo " hello hacker " >> sample.txt
 ---> Running in 73d2bb17fe49
Removing intermediate container 73d2bb17fe49
 ---> cc14e6890830
Step 4/7 : ENV who_is_hacker vishal
 ---> Running in 1a3974ff0aa0
Removing intermediate container 1a3974ff0aa0
 ---> e1b0454141e7
Step 5/7 : RUN echo "$who_is_hacker" >> xyz.txt
 ---> Running in c1b412486412
Removing intermediate container c1b412486412
 ---> b2e9c12edd5b
Step 6/7 : COPY file1.txt /mnt/
 ---> 0549bff65c81
Step 7/7 : ADD data.tar.gz /mnt/
 ---> 87e7fbcfb175
Successfully built 87e7fbcfb175
Successfully tagged ubuntu:v01
[root@ip-172-31-11-147 ~]#
```

```
[root@ip-172-31-11-147 ~]# ls
data.tar.gz  Dockerfile  file1.txt
[root@ip-172-31-11-147 ~]# docker images
REPOSITORY    TAG       IMAGE ID       CREATED              SIZE
ubuntu        v01       87e7fbcfb175   About a minute ago   77.8MB
vish-ubuntu   latest    2e6f27ccd20a   47 minutes ago       77.8MB
ubuntu        latest    a8780b506fa4   10 days ago          77.8MB
[root@ip-172-31-11-147 ~]# docker run -it ubuntu:v01 /bin/bash
root@ad1219413b62:/mnt# ls
data.txt  file1.txt  sample.txt  xyz.txt
root@ad1219413b62:/mnt# cat data*
root@ad1219413b62:/mnt# cat file1*
root@ad1219413b62:/mnt# cat sample.txt
 hello hacker
root@ad1219413b62:/mnt# cat xyz.txt
vishal
root@ad1219413b62:/mnt#
```

Chapter 3 : docker

==========

## Docker volume and how to share it ??

- volume is simply a directory inside our containers
- firstly we have to declare this directory as a volume and then share volume
- even if we stop container still we can access volume
- volume will be created in one container
- you can declare a directory as a volume only while creating container
- you cant create volume from existing container
- you can share one volume across any number of containers
- volumes will not included when you update an image
- you can mapped volume in two ways,

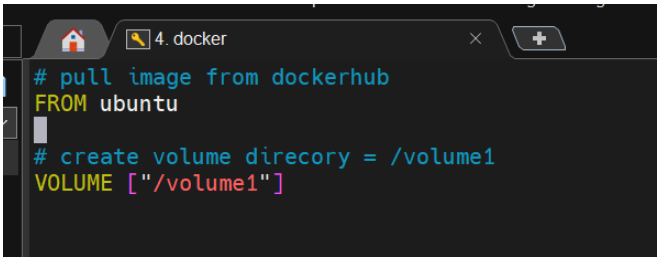
    1. container ←----→ container
    2. host ←----→ container

### ➢ Benefits of volume

- Decoupling container from storage
- Share volume among different containers
- Attach volume to containers
- On deleting container volume does not delete

Example 1 :

vi Dockerfile



# Chapter 3 : docker

==========

docker build -t myimage .

&

Make container and enter in it and check whether volume is created or not

&

Make new files in volume1

```
[root@ip-172-31-33-66 ~]# vi Dock*
[root@ip-172-31-33-66 ~]# docker build -t myimage .
Sending build context to Docker daemon  36.86kB
Step 1/2 : FROM ubuntu
latest: Pulling from library/ubuntu
e96e057aae67: Pull complete
Digest: sha256:4b1d0c4a2d2aaf63b37111f34eb9fa89fa1bf53dd6e4ca954d47caebca4005c2
Status: Downloaded newer image for ubuntu:latest
 ---> a8780b506fa4
Step 2/2 : VOLUME ["/volume1"]
 ---> Running in bd4307501e43
Removing intermediate container bd4307501e43
 ---> d931a20371a7
Successfully built d931a20371a7
Successfully tagged myimage:latest
[root@ip-172-31-33-66 ~]# docker images
REPOSITORY   TAG       IMAGE ID       CREATED          SIZE
myimage      latest    d931a20371a7   45 seconds ago   77.8MB
ubuntu       latest    a8780b506fa4   3 weeks ago      77.8MB
[root@ip-172-31-33-66 ~]#
[root@ip-172-31-33-66 ~]# docker run -it --name container1 myimage /bin/bash
root@d7744048c19b:/# ls
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var  volume1
root@d7744048c19b:/# cd volume1
root@d7744048c19b:/volume1# ls
root@d7744048c19b:/volume1# touch xyz aaa file5
root@d7744048c19b:/volume1# ls
aaa  file5  xyz
root@d7744048c19b:/volume1#
```

# Chapter 3 : docker

==========

**Example 2**: share volume of previous container with new one container & check whether files exits or not which were created in volume of container1

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

```
Games  Sessions  View  Split  MultiExec  Tunneling  Packages  Settings  Help
4. docker
[root@ip-172-31-33-66 ~]# docker images
REPOSITORY   TAG       IMAGE ID       CREATED         SIZE
myimage      latest    d931a20371a7   4 minutes ago   77.8MB
ubuntu       latest    a8780b506fa4   3 weeks ago     77.8MB
[root@ip-172-31-33-66 ~]# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
[root@ip-172-31-33-66 ~]# docker ps -a
CONTAINER ID   IMAGE     COMMAND       CREATED         STATUS                      PORTS     NAMES
d7744048c19b   myimage   "/bin/bash"   3 minutes ago   Exited (127) 15 seconds ago             container1
[root@ip-172-31-33-66 ~]#
[root@ip-172-31-33-66 ~]# docker run -it --name container2 --privileged=true --volumes-from container1 ubuntu /bin/bash
root@97d43c6dfd07:/# ls
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var  volume1
root@97d43c6dfd07:/# cd volumes
bash: cd: volumes: No such file or directory
root@97d43c6dfd07:/#
root@97d43c6dfd07:/# cd v
var/     volume1/
root@97d43c6dfd07:/# cd volume1/
root@97d43c6dfd07:/volume1# ls
aaa  file5  xyz
root@97d43c6dfd07:/volume1#
root@97d43c6dfd07:/volume1#
```

# Chapter 3 : docker

==========

## Create container with volume using commands instead of docker file

```
5. docker
[root@ip-172-31-33-66 ~]# ls
Dockerfile
[root@ip-172-31-33-66 ~]# docker ps -a
CONTAINER ID   IMAGE     COMMAND       CREATED          STATUS                       PORTS     NAMES
97d43c6dfd07   ubuntu    "/bin/bash"   40 minutes ago   Up 40 minutes                          container2
d7744048c19b   myimage   "/bin/bash"   44 minutes ago   Exited (127) 40 minutes ago            container1
[root@ip-172-31-33-66 ~]# docker images
REPOSITORY   TAG       IMAGE ID       CREATED          SIZE
myimage      latest    d931a20371a7   45 minutes ago   77.8MB
ubuntu       latest    a8780b506fa4   3 weeks ago      77.8MB
[root@ip-172-31-33-66 ~]# docker run -it -v /volume2 container3 ubuntu /bin/bash
Unable to find image 'container3:latest' locally
docker: Error response from daemon: pull access denied for container3, repository does not exist or may require 'docker login': denied:
 requested access to the resource is denied.
See 'docker run --help'.
[root@ip-172-31-33-66 ~]# docker run -it -v /volume2 --name container3 ubuntu /bin/bash
root@b7f301467be4:/# ls
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var  volume2
root@b7f301467be4:/# cd volume3
bash: cd: volume3: No such file or directory
root@b7f301467be4:/# ls
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var  volume2
root@b7f301467be4:/# cd volume2
root@b7f301467be4:/volume2# ls
root@b7f301467be4:/volume2# touch sample1 sample2 sample3
root@b7f301467be4:/volume2# ls
sample1  sample2  sample3
root@b7f301467be4:/volume2#
```

```
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var  volume2
root@b7f301467be4:/# cd volume2
root@b7f301467be4:/volume2# ls
root@b7f301467be4:/volume2# touch sample1 sample2 sample3
root@b7f301467be4:/volume2# ls
sample1  sample2  sample3
root@b7f301467be4:/volume2# exit
exit
[root@ip-172-31-33-66 ~]# ls
Dockerfile
[root@ip-172-31-33-66 ~]# docker run -it --name container4 --privileged=true --volumes-from container3 ubuntu /bin/bash
root@23cb4bd36b8e:/# ls
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var  volume2
root@23cb4bd36b8e:/# cd volumes2
bash: cd: volumes2: No such file or directory
root@23cb4bd36b8e:/# cd volume2/
root@23cb4bd36b8e:/volume2# ls
sample1  sample2  sample3
root@23cb4bd36b8e:/volume2# touch sample4 file6
root@23cb4bd36b8e:/volume2# ls
file6  sample1  sample2  sample3  sample4
root@23cb4bd36b8e:/volume2# exit
exit
[root@ip-172-31-33-66 ~]# ls
Dockerfile
[root@ip-172-31-33-66 ~]# dockr images
-bash: dockr: command not found
[root@ip-172-31-33-66 ~]# docker ps -a
CONTAINER ID   IMAGE     COMMAND       CREATED          STATUS                          PORTS     NAMES
23cb4bd36b8e   ubuntu    "/bin/bash"   43 seconds ago   Exited (0) 9 seconds ago                  container4
b7f301467be4   ubuntu    "/bin/bash"   3 minutes ago    Exited (0) About a minute ago             container3
97d43c6dfd07   ubuntu    "/bin/bash"   45 minutes ago   Up 45 minutes                             container2
d7744048c19b   myimage   "/bin/bash"   49 minutes ago   Exited (127) 46 minutes ago               container1
[root@ip-172-31-33-66 ~]# docker start b7f301467be4
b7f301467be4
[root@ip-172-31-33-66 ~]# docker exec -it b7f301467be4 /bin/bash
root@b7f301467be4:/# ls
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var  volume2
root@b7f301467be4:/# cd volume2/
root@b7f301467be4:/volume2# ls
file6  sample1  sample2  sample3  sample4
root@b7f301467be4:/volume2#
```

Chapter 3 : docker

==========

## Share volume , container to host and vice versa (bind mount)

Here I m binding host directory (/home/ec2-user) with container volume (vishalk17)

```
3. docker
[root@ip-172-31-33-66 ~]#
[root@ip-172-31-33-66 ~]#
[root@ip-172-31-33-66 ~]# docker images
REPOSITORY   TAG       IMAGE ID       CREATED        SIZE
myimage      latest    d931a20371a7   25 hours ago   77.8MB
ubuntu       latest    a8780b506fa4   3 weeks ago    77.8MB
[root@ip-172-31-33-66 ~]# docker run -it --name hostcont -v /home/ec2-user:/vishalk17 ubuntu /bin/bash
root@bfaf72186690:/# ;s
bash: syntax error near unexpected token `;'
root@bfaf72186690:/# ls
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  media  mnt  opt  proc  root  run  sbin  srv  sys
root@bfaf72186690:/#
root@bfaf72186690:/# ls
bin   dev  home  lib32  libx32  mnt  proc  run   srv  tmp  var
boot  etc  lib   lib64  media   opt  root  sbin  sys  usr  vishalk17
root@bfaf72186690:/# cd vishalk17
root@bfaf72186690:/vishalk17# ls
Dockerfile  docker.docx
root@bfaf72186690:/vishalk17# exit
exit
[root@ip-172-31-33-66 ~]# ls
Dockerfile
[root@ip-172-31-33-66 ~]# cd /home/ec2-user
[root@ip-172-31-33-66 ec2-user]# ls
docker.docx  Dockerfile
[root@ip-172-31-33-66 ec2-user]#
```

| | |
|---|---|
| docker volume create volume_name | ..create volume |
| docker volume ls | .. see list of volumes |
| docker volume rm volume_name | ..remove volume |
| docker volume inspect volume_name | ..detail about volume |
| docker volume prune | ..remove unused volume |

Chapter 3 : docker

==========

## Docker port expose & publish

- service docker start
- docker run -td –techserver -p 90:80 ubuntu (make sure you already have enabled 90 port in security group)
  p = publish
- docker ps
- docker port techserver (check port assigned to techserver)

- docker exec -it techserver /bin/bash ( you will be inside in techserver container)
  - apt-get update
  - apt-get install apache2 -y
  - cd /var/www/html
  - echo " hi its vishalk17 " >> index.html
  - service apache2 start

now check whether you are able to access webpage from the container on port no. 90 or not

ip_address:90 [check in any browser]

# Chapter 3 : docker

==========

### Docker exec :

It is specifically for running new things in already started container
or
it creates a new process in the container's environment while " docker attach" just connect the standard input/output of its main process inside the container .

### Difference between docker expose and docker port

Basically you have three option :

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 accessible 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, 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.

Chapter 3 : docker

==========

## How to push docker image in docker hub ?

- service docker start
- docker run -it ubuntu /bin/bash (You will be inside in container)
    - now create some files in container
- docker commit container_id image1

## now create acc. In hun.docker.com

- docker login (enter your username and pass of docker hub)
- docker tag image1 docker_username/new_image
- docker push docker_username/new_image

**now you can see this image in docker hub acc.**

**Now create one instance in another region and pull image from hub to check**

- docker pull docker_username/new_image
- docker run -it –name new_container docker_username/new_image /bin/bash

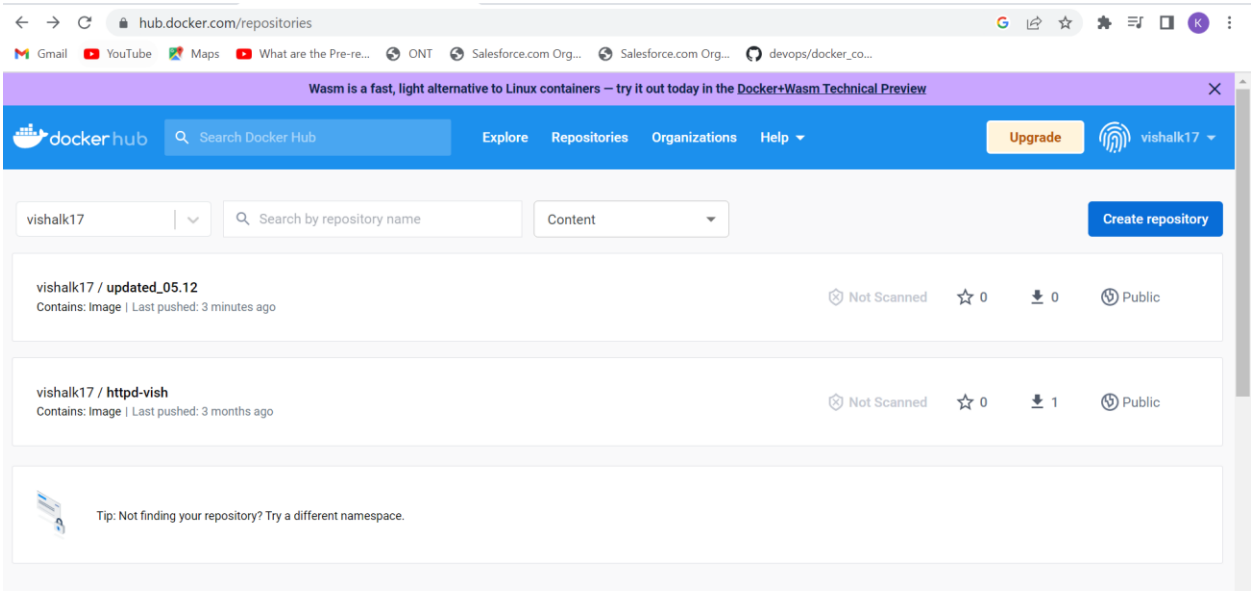
```
4. docker
[root@ip-172-31-33-66 ~]# clear
[root@ip-172-31-33-66 ~]#
[root@ip-172-31-33-66 ~]#
[root@ip-172-31-33-66 ~]# service docker start
Redirecting to /bin/systemctl start docker.service
[root@ip-172-31-33-66 ~]# ls
Dockerfile
[root@ip-172-31-33-66 ~]# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED        STATUS        PORTS                                   NAMES
3ae1d019ab4c   ubuntu    "bash"    22 hours ago   Up 22 hours   0.0.0.0:90->80/tcp, :::90->80/tcp   techserver
[root@ip-172-31-33-66 ~]# docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
ubuntu       latest    a8780b506fa4   4 weeks ago   77.8MB
[root@ip-172-31-33-66 ~]# docker 3ae*
docker: '3ae*' is not a docker command.
See 'docker --help'
[root@ip-172-31-33-66 ~]# ls
Dockerfile
[root@ip-172-31-33-66 ~]# docker exec -it 3ae /bin/bash
root@3ae1d019ab4c:/# ls
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@3ae1d019ab4c:/# touch file1 file2 file3
root@3ae1d019ab4c:/# read escape sequence
[root@ip-172-31-33-66 ~]# ls
Dockerfile
[root@ip-172-31-33-66 ~]# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED        STATUS        PORTS                                   NAMES
3ae1d019ab4c   ubuntu    "bash"    22 hours ago   Up 22 hours   0.0.0.0:90->80/tcp, :::90->80/tcp   techserver
[root@ip-172-31-33-66 ~]# docker commit 3ae update_v1:latest
sha256:a8dbc83e2c09d8da5de2cd26b3aa9497727de4de7802f62015ec59a9d97ccc8c
[root@ip-172-31-33-66 ~]# docker images
REPOSITORY   TAG       IMAGE ID       CREATED         SIZE
update_v1    latest    a8dbc83e2c09   5 seconds ago   226MB
```

# Chapter 3 : docker

==========

```
4. docker
CONTAINER ID   IMAGE     COMMAND   CREATED        STATUS        PORTS                                      NAMES
3ae1d019ab4c   ubuntu    "bash"    22 hours ago   Up 22 hours   0.0.0.0:90→80/tcp, :::90→80/tcp            techserver
[root@ip-172-31-33-66 ~]# docker commit 3ae update_v1:latest
sha256:a8dbc83e2c09d8da5de2cd26b3aa9497727de4de7802f62015ec59a9d97ccc8c
[root@ip-172-31-33-66 ~]# docker images
REPOSITORY   TAG       IMAGE ID       CREATED          SIZE
update_v1    latest    a8dbc83e2c09   5 seconds ago    226MB
ubuntu       latest    a8780b506fa4   4 weeks ago      77.8MB
[root@ip-172-31-33-66 ~]# docker tag update_v1 vishalk17/updated_05/12
[root@ip-172-31-33-66 ~]# docker images
REPOSITORY                TAG       IMAGE ID       CREATED          SIZE
update_v1                 latest    a8dbc83e2c09   36 seconds ago   226MB
vishalk17/updated_05/12   latest    a8dbc83e2c09   36 seconds ago   226MB
ubuntu                    latest    a8780b506fa4   4 weeks ago      77.8MB
[root@ip-172-31-33-66 ~]# 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.do
o create one.
Username: vishalk17
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-172-31-33-66 ~]# docker push vishalk17/updated_05/12
Using default tag: latest
The push refers to repository [docker.io/vishalk17/updated_05/12]
f1b91811c9f8: Preparing
f4a670ac65b6: Preparing
denied: requested access to the resource is denied
[root@ip-172-31-33-66 ~]# docker tag update_v1 vishalk17/updated_05.12
[root@ip-172-31-33-66 ~]# docker push vishalk17/updated_05.12
Using default tag: latest
The push refers to repository [docker.io/vishalk17/updated_05.12]
f1b91811c9f8: Pushed
f4a670ac65b6: Pushed
latest: digest: sha256:f4fecfeee6537ea54892596b3326952add6613cd15ac5fb96abb64f456742502 size: 741
[root@ip-172-31-33-66 ~]#
```



# Chapter 3 : docker

==========

```
4. docker        7. new_server
[ec2-user@ip-172-31-33-66 ~]$
[ec2-user@ip-172-31-33-66 ~]$ sudo su -
Last login: Mon Dec  5 02:29:17 UTC 2022 on pts/0
[root@ip-172-31-33-66 ~]#
[root@ip-172-31-33-66 ~]# docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
[root@ip-172-31-33-66 ~]# docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
[root@ip-172-31-33-66 ~]#
[root@ip-172-31-33-66 ~]# docker run -it --name container1 vishalk17/updated_05.12 /bin/bash
Unable to find image 'vishalk17/updated_05.12:latest' locally
latest: Pulling from vishalk17/updated_05.12
e96e057aae67: Pull complete
18fb5b36196b: Pull complete
Digest: sha256:f4fecfeee6537ea54892596b3326952add6613cd15ac5fb96abb64f456742502
Status: Downloaded newer image for vishalk17/updated_05.12:latest
root@e15654d790f8:/#
root@e15654d790f8:/# ls
bin   dev  file1  file3  lib    lib64  media  opt   root  sbin  sys  usr
boot  etc  file2  home   lib32  libx32 mnt    proc  run   srv   tmp  var
root@e15654d790f8:/#
root@e15654d790f8:/# [root@ip-172-31-33-66 ~]#
[root@ip-172-31-33-66 ~]# docker ps -a
CONTAINER ID   IMAGE                    COMMAND       CREATED          STATUS          PORTS     NAMES
e15654d790f8   vishalk17/updated_05.12  "/bin/bash"   30 seconds ago   Up 29 seconds   80/tcp    container1
[root@ip-172-31-33-66 ~]#
```

**Some Important commands:**

- docker stop $(docker ps -a -q) -- stop all running containers
- docker rm $(docker ps -a -q) -- delete all stop containers
- docker rmi -f $(docker ps -a -q) -- delete all images