How to Install the Docker and run the simple docker file on EC2

1..Update the server

sudo yum update

2..install the docker
sudo yum install docker

3.start the docker
sudo service docker start

4.check the docker
docker –v

5.Now create the dictionary
mkdir images

6. go to the dictionary
cd images/

7. Create the docker file
nano Dockerfile (for saving ctrl+o and for coming out from file ctrl+x)

8..docker build . (to build docker image)

9..docker images (to check whether docker image created or not)

10..Docker ps (to check created docker)

All about Docker basic on AWS server

To install docker

[ec2-user@ip-172-31-39-224 ~]$ sudo yum install docker

to start docker
[ec2-user@ip-172-31-39-224 ~]$ sudo service docker start

to get the information about docker

[ec2-user@ip-172-31-39-224 ~]$ sudo docker info

to create a docker container from ubuntu image from internet
[ec2-user@ip-172-31-39-224 ~]$ sudo docker container run ubuntu cat /etc/os-release

to check the docker images
[ec2-user@ip-172-31-39-224 ~]$ sudo docker image ls

REPOSITORY TAG IMAGE ID CREATED SIZE
ubuntu latest 93fd78260bd1 2 weeks ago 86.2MB

to check the running container
[ec2-user@ip-172-31-39-224 ~]$ sudo docker container ls

CONTAINER
ID IMAGE COMMAND CREATED STATUS PORTS N
AMES

to check all running+ non running container
[ec2-user@ip-172-31-39-224 ~]$ sudo docker container ls -a

CONTAINER
ID IMAGE COMMAND CREATED STATUS PORTS
NAMES
b9c5403afb49 ubuntu "cat /etc/os-release" 4 minutes ago Exited (0) 4 minutes
ago awesome_rosalind

to create one more container
[ec2-user@ip-172-31-39-224 ~]$ sudo docker container run ubuntu sleep 30

to check both container
[ec2-user@ip-172-31-39-224 ~]$ sudo docker container ls -a

```
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS                      PORTS     NAMES
aea80b8709e2   ubuntu    "sleep 30"               2 minutes ago    Exited (0) About a minute ago             mystifying_wing
b9c5403afb49   ubuntu    "cat /etc/os-release"    8 minutes ago    Exited (0) 8 minutes ago                  awesome_rosalind
```

How to Install Docker Compose on Amazon Server

On Linux systems, first install the Docker

```
[root@ip-172-31-84-176 ~]# sudo yum install docker
[root@ip-172-31-84-176 ~]# sudo service docker start
```

Run this command to download the current stable release of Docker Compose:

```
[root@ip-172-31-84-176 ~]# sudo curl -L
"https://github.com/docker/compose/releases/download/1.24.0/docker-compose-$(uname -s)-
$(uname -m)" -o /usr/local/bin/docker-compose
```

Apply executable permissions to the binary:

```
[root@ip-172-31-84-176 ~]# sudo chmod +x /usr/local/bin/docker-compose
```

Note: If the command docker-compose fails after installation, check your path. You can also create a symbolic link to /usr/bin or any other directory in your path.

```
[root@ip-172-31-84-176 ~]# sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose
```

Test the installation.

```
[root@ip-172-31-84-176 ~]# docker-compose --version
docker-compose version 1.24.0, build 0aa59064
```

10 Docker Command Part-1

1. To install docker in amazon ec2 linux server

[ec2-user@ip-172-31-39-224 ~]$ sudo yum install docker

2. To start docker

[ec2-user@ip-172-31-39-224 ~]$ sudo service docker start

3. To check the status

[ec2-user@ip-172-31-39-224 ~]$ sudo service docker status

4. To get the information about docker

[ec2-user@ip-172-31-39-224 ~]$ sudo docker info

5. To get the version of docker

[ec2-user@ip-172-31-39-224 ~]$ sudo docker version

6. To create a docker container from ubuntu image and get the release of ubuntu

[root@ip-172-31-92-105 ~]# docker container run ubuntu cat /etc/os-release

7. To create a docker container from ubuntu image and run sleep command for 30s

[root@ip-172-31-92-105 ~]# docker container run ubuntu sleep 30

8. To get list of docker image

[root@ip-172-31-92-105 ~]# docker image ls

9. To list all running container

[root@ip-172-31-92-105 ~]# docker container ls

10. To list all running and non-running container

[root@ip-172-31-92-105 ~]# docker container ls –a

10 Docker Command Part-2

1. To remove the stop container

[root@ip-172-31-92-105 ~]# docker container rm 4a79e7c7d036

2. To remove the multiple stop container

[root@ip-172-31-92-105 ~]# docker container rm afcd9e49df33 84d890f1eba0

3. To start the stop container

[root@ip-172-31-38-174 ~]# docker container start 2b4ed9cb2cbd

4. To stop the running container

[root@ip-172-31-38-174 ~]# docker container stop 2b4ed9cb2cbd

5. To run the container in the background

[root@ip-172-31-38-174 ~]# docker container run -d ubuntu sleep 30

6. To go inside the container

[root@ip-172-31-38-174 ~]# docker container run -it ubuntu /bin/bash

7. To come out of container but it will also stop the running container

root@eb1bd5486f0f:/# exit or ctrl+d

8. By use of this you will come out of container but your container will run in background

Ctrl+pq

9. This will give container id of all running and stop container

[root@ip-172-31-38-174 ~]# docker container ls –aq

10. This will delete all the container at the same time

root@ip-172-31-38-174 ~]# docker container rm $(docker container ls -aq)

10 Docker Command Part-3

1. This will create a container in which nginx web server will run in the background

[root@ip-172-31-38-174 ~]# docker container run -d nginx

2. This will give private ip and port of running container and by the help of this ip we can access the web server running on this container,

[root@ip-172-31-38-174 ~]# docker container inspect dd8442fd1087

Note:- 172.17.0.2 this ip is not accessible outside to access we need to do port mapping

3. This will give the logs of running container

[root@ip-172-31-38-174 ~]# docker container logs dd8442fd1087

4. This will give all the running process inside the container

[root@ip-172-31-38-174 ~]# docker container top dd8442fd1087

5. This will give how much cpu and memory are utilized by all running container

[root@ip-172-31-38-174 ~]# docker container stats

6. This will do the port mapping whatever request come to port 3600 it will forward the request to port 80

[root@ip-172-31-38-174 ~]# docker container run -d -p 3600:80 --name raj nginx

Now if give http://3.85.8.204:3600/(public ip of ec2 server) in browser then it will forward the request to container private ip(172.17.0.3 this is not ec2 server private ip it is container private ip which are running on ec2) on port 80 and we are able to access the web container

To check port mapping and name of container run the below command
[root@ip-172-31-38-174 ~]# docker container ls

CONTAINER
ID IMAGE COMMAND CREATED STATUS PORTS
NAMES

b5e091c626b3 nginx "nginx -g 'daemon of…" 4 minutes ago Up 4

minutes 0.0.0.0:3600->80/tcp raj

7. This will take you inside container to install any software inside the container

[root@ip-172-31-38-174 ~]# docker container exec -it b5e091c626b3 /bin/bash

root@b5e091c626b3:/# -------Now install any software you want

8. This will rename the docker container name

[root@ip-172-31-38-174 ~]# docker container rename dd8442fd1087 aws

9. This will restart the container

[root@ip-172-31-38-174 ~]# docker container restart b5e091c626b3

10. This command will take the running container from background to front screen

[root@ip-172-31-38-174 ~]# docker container attach 74094d60f4b3

10 Docker Command Part-4

1. This will kill the running container

[root@ip-172-31-38-174 ~]# docker container kill dd8442fd1087

2. This command will wait until container will stop and when container stop it will return stop id(0)

[root@ip-172-31-38-174 ~]# docker container wait b5e091c626b3

3. This will pause all the process running in the container and you are not able to access it

[root@ip-172-31-38-174 ~]# docker container pause b5e091c626b3

4. This will unpause the pause container

[root@ip-172-31-38-174 ~]# docker container unpause b5e091c626b3

5. It will delete all the non-running container

[root@ip-172-31-38-174 ~]# docker container prune

6. This will give port mapping details

[root@ip-172-31-38-174 ~]# docker container port b5e091c626b3

7. This will create container but not run it but run command create and run container

[root@ip-172-31-38-174 ~]# docker container create ubuntu sleep 60

8. This will start the created container

[root@ip-172-31-38-174 ~]# docker container start 3cd4449cbb68

9. This will give the difference between container with last update in container

[root@ip-172-31-38-174 ~]# docker container diff b5e091c626b3

10. This will copy file a.txt to container from local host

[root@ip-172-31-38-174 ~]# docker container cp a.txt 1982459689ef:/tmp

Docker Command Part-5

Export Command:-

[root@ip-172-31-94-183 ~]# docker container run -it ubuntu /bin/bash

root@e694e4e16ffd:/# apt-get update

root@e694e4e16ffd:/# apt-get install tree git –y

[root@ip-172-31-94-183 ~]# docker container ls

```
CONTAINER
ID        IMAGE          COMMAND          CREATED          STATUS          PORTS          N
AMES
e694e4e16ffd        ubuntu          "/bin/bash"        5 minutes ago       Up 5
minutes                      friendly_meitner
```

[root@ip-172-31-94-183 ~]# docker container export e694e4e16ffd >raj.tar
[root@ip-172-31-94-183 ~]# ls –lh

```
total 180M
-rw-r--r-- 1 root root 180M Jul 11 08:25 raj.tar

[root@ip-172-31-94-183 ~]#
```

or

[root@ip-172-31-94-183 ~]# docker container export e694e4e16ffd -o raj1.tar

[root@ip-172-31-94-183 ~]# ls -lh

```
total 359M
-rw------- 1 root root 180M Jul 11 08:27 raj1.tar
-rw-r--r-- 1 root root 180M Jul 11 08:25 raj.tar
[root@ip-172-31-94-183 ~]#
```

Now share this tar file with other which you want

Import Command:-

Now other need to convert above tar file into image to use and create container

[root@ip-172-31-94-183 ~]# docker image import raj.tar rajimage

sha256:9a7d5e7ed5a54191780d734004d6a8fe8ab35c5bf6c108e7a20a125afb95330c

[root@ip-172-31-94-183 ~]# docker image ls

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
rajimage	latest	9a7d5e7ed5a5	23 seconds ago	183MB
ubuntu	latest	4c108a37151f	3 weeks ago	64.2MB

[root@ip-172-31-94-183 ~]# docker container run -it rajimage /bin/bash
root@4f609111b319:/# ls

bin boot dev etc home lib lib64 media mnt opt proc root run sbin srv sys tmp usr var

root@4f609111b319:/# cd tmp/
root@4f609111b319:/tmp# tree
.

0 directories, 0 files

root@4f609111b319:/tmp# git –version

git version 2.17.1

root@4f609111b319:/tmp#

Docker Command Part-6

How to create image from running docker container and share that image to other to use and create new docker container

[root@ip-172-31-94-183 ~]# docker container run -it ubuntu /bin/bash

root@815b8c207540:/# cd tmp/

root@815b8c207540:/tmp# touch 1 2 3 4 56 7 8 90

root@815b8c207540:/tmp# ls

1 2 3 4 56 7 8 90

root@815b8c207540:/tmp# ls -lh
total 0
-rw-r--r-- 1 root root 0 Jul 11 08:47 1
-rw-r--r-- 1 root root 0 Jul 11 08:47 2
-rw-r--r-- 1 root root 0 Jul 11 08:47 3
-rw-r--r-- 1 root root 0 Jul 11 08:47 4
-rw-r--r-- 1 root root 0 Jul 11 08:47 56
-rw-r--r-- 1 root root 0 Jul 11 08:47 7
-rw-r--r-- 1 root root 0 Jul 11 08:47 8
-rw-r--r-- 1 root root 0 Jul 11 08:47 90

root@815b8c207540:/tmp# [root@ip-172-31-94-183 ~]#
[root@ip-172-31-94-183 ~]# docker container ls

CONTAINER
ID IMAGE COMMAND CREATED STATUS PORTS N
AMES
815b8c207540 ubuntu "/bin/bash" 2 minutes ago Up About a
minute dazzling_brown
e694e4e16ffd ubuntu "/bin/bash" 29 minutes ago Up 29
minutes friendly_meitner

[root@ip-172-31-94-183 ~]# docker container commit 815b8c207540 newimage
sha256:7b68b021455a6601029abda69f812f4a3fd45bb5800ea3c200ad295c13c8dda4

[root@ip-172-31-94-183 ~]# docker image ls
REPOSITORY TAG IMAGE ID CREATED SIZE
newimage latest 7b68b021455a 19 seconds ago 64.2MB
rajimage latest 9a7d5e7ed5a5 16 minutes ago 183MB
ubuntu latest 4c108a37151f 3 weeks ago 64.2MB

[root@ip-172-31-94-183 ~]#
[root@ip-172-31-94-183 ~]# docker container rm -f 815b8c207540
815b8c207540

[root@ip-172-31-94-183 ~]# docker container ls
CONTAINER
ID IMAGE COMMAND CREATED STATUS PORTS N
AMES
e694e4e16ffd ubuntu "/bin/bash" 36 minutes ago Up 36
minutes friendly_meitner

[root@ip-172-31-94-183 ~]# docker container run -it newimage /bin/bash

root@e872b02b6002:/# cd tmp/

root@e872b02b6002:/tmp# ls
1 2 3 4 56 7 8 90

root@e872b02b6002:/tmp#

Docker Command Part-7

By default latest version of image will download from docker hub...If you want to download a specific version of image then use the below command:-

[root@ip-172-31-94-183 ~]# docker pull ubuntu:14.04
[root@ip-172-31-94-183 ~]# docker image ls

ubuntu 14.04 2c5e00d77a67 8 weeks ago 188MB

To logging into Docker hub:-

[root@ip-172-31-94-183 ~]# 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: rajguptaaws
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

Now to push the own custom image to docker hub

[root@ip-172-31-94-183 ~]# docker image ls

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
newimage	latest	7b68b021455a	3 hours ago	64.2MB
rajimage	latest	9a7d5e7ed5a5	3 hours ago	183MB
ubuntu	latest	4c108a37151f	3 weeks ago	64.2MB
ubuntu	14.04	2c5e00d77a67	8 weeks ago	188MB

[root@ip-172-31-94-183 ~]# docker image tag rajimage rajguptaaws/raji123456
[root@ip-172-31-94-183 ~]# docker image ls

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
newimage	latest	7b68b021455a	3 hours ago	64.2MB
rajimage	latest	9a7d5e7ed5a5	3 hours ago	183MB
rajguptaaws/raji123456	latest	9a7d5e7ed5a5	3 hours ago	183MB
ubuntu	latest	4c108a37151f	3 weeks ago	64.2MB
ubuntu	14.04	2c5e00d77a67	8 weeks ago	188MB

[root@ip-172-31-94-183 ~]# docker push rajguptaaws/raji123456

The push refers to repository [docker.io/rajguptaaws/raji123456]
9079d3f9c52e: Pushed
latest: digest: sha256:790db81f22c4daa322fc1275ef948434290c00a8b7aaa0a449e36ee8a77c2194 size: 528

Docker Command Part-8

[root@ip-172-31-94-183 ~]# docker image ls

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
newimage	latest	7b68b021455a	4 hours ago	64.2MB
rajimage	latest	9a7d5e7ed5a5	5 hours ago	183MB
rajguptaaws/raji123456	latest	9a7d5e7ed5a5	5 hours ago	183MB
ubuntu	latest	4c108a37151f	3 weeks ago	64.2MB
ubuntu	14.04	2c5e00d77a67	8 weeks ago	188MB

[root@ip-172-31-94-183 ~]# docker image ls --format '{{.ID}} , {{.Repository}}'
7b68b021455a , newimage
9a7d5e7ed5a5 , rajimage
9a7d5e7ed5a5 , rajguptaaws/raji123456
4c108a37151f , ubuntu
2c5e00d77a67 , ubuntu

[root@ip-172-31-94-183 ~]# docker image ls --format '{{.ID}} , {{.Repository}} , {{.Tag}}'
7b68b021455a , newimage , latest
9a7d5e7ed5a5 , rajimage , latest
9a7d5e7ed5a5 , rajguptaaws/raji123456 , latest
4c108a37151f , ubuntu , latest
2c5e00d77a67 , ubuntu , 14.04

[root@ip-172-31-94-183 ~]#

To check the history of image

[root@ip-172-31-94-183 ~]# docker image history rajimage

IMAGE	CREATED	CREATED BY	SIZE	COMMENT
9a7d5e7ed5a5	5 hours ago		183MB	Imported from -

To remove the images which are not attached with any docker container
[root@ip-172-31-94-183 ~]# docker image rm ubuntu newimage

To remove the images which are attached with any docker container forcefully
[root@ip-172-31-94-183 ~]# docker image rm -f ubuntu newimage

To get the details of image like port number any many other information
[root@ip-172-31-94-183 ~]# docker image inspect newimage

It will delete all the images
[root@ip-172-31-94-183 ~]# docker image prune

Docker Command Part-9

Note:- save and export command both give the tar file to share with other , load and import both use to create image from tar file

If you want to share the image then you can also create tar file by the help of save command

[root@ip-172-31-94-183 ~]# docker image ls

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
<none>	<none>	7b68b021455a	5 hours ago	64.2MB
rajimage	latest	9a7d5e7ed5a5	5 hours ago	183MB
rajguptaaws/raji123456	latest	9a7d5e7ed5a5	5 hours ago	183MB
ubuntu	14.04	2c5e00d77a67	8 weeks ago	188MB

[root@ip-172-31-94-183 ~]# docker image save rajimage > rajimagenew.tar
[root@ip-172-31-94-183 ~]# ls
raj1.tar rajimagenew.tar raj.tar

[root@ip-172-31-94-183 ~]# docker image rm rajimage
Untagged: rajimage:latest

[root@ip-172-31-94-183 ~]# docker image ls

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
<none>	<none>	7b68b021455a	5 hours ago	64.2MB
rajguptaaws/raji123456	latest	9a7d5e7ed5a5	6 hours ago	183MB
ubuntu	14.04	2c5e00d77a67	8 weeks ago	188MB

[root@ip-172-31-94-183 ~]# docker image load < rajimagenew.tar
Loaded image: rajimage:latest

[root@ip-172-31-94-183 ~]# docker image ls

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
<none>	<none>	7b68b021455a	5 hours ago	64.2MB
rajimage	latest	9a7d5e7ed5a5	6 hours ago	183MB
rajguptaaws/raji123456	latest	9a7d5e7ed5a5	6 hours ago	183MB
ubuntu	14.04	2c5e00d77a67	8 weeks ago	188MB

Docker Command Part-10

How to create custom image from custom docker file

[root@ip-172-31-85-137 ~]# mkdir dockerfiles
[root@ip-172-31-85-137 ~]# cd dockerfiles/
[root@ip-172-31-85-137 dockerfiles]# vi Dockerfile
[root@ip-172-31-85-137 dockerfiles]# cat Dockerfile

FROM ubuntu:16.04

[root@ip-172-31-85-137 dockerfiles]# docker image build -t rajubuntu:1 .

Sending build context to Docker daemon 2.048kB
Step 1/1 : FROM ubuntu:16.04
16.04: Pulling from library/ubuntu
35b42117c431: Pull complete
ad9c569a8d98: Pull complete
293b44f45162: Pull complete
0c175077525d: Pull complete
Digest: sha256:a4d8e674ee993e5ec88823391de828a5e9286a1597b731eaecaaf9066cfdf539
Status: Downloaded newer image for ubuntu:16.04
 ---> 13c9f1285025
Successfully built 13c9f1285025
Successfully tagged rajubuntu:1

[root@ip-172-31-85-137 dockerfiles]# docker image ls

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
rajubuntu	1	13c9f1285025	3 weeks ago	119MB
ubuntu	16.04	13c9f1285025	3 weeks ago	119MB

[root@ip-172-31-85-137 dockerfiles]# docker container run -it rajubuntu:1
root@77099511689e:/# tree ----not found
bash: tree: command not found

root@77099511689e:/# exit
exit

[root@ip-172-31-85-137 dockerfiles]# vi Dockerfile
[root@ip-172-31-85-137 dockerfiles]# cat Dockerfile
FROM ubuntu:16.04
RUN apt-get update && apt-get install -y tree

[root@ip-172-31-85-137 dockerfiles]# docker image build -t rajubuntu:2 .

[root@ip-172-31-85-137 dockerfiles]# docker image ls

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
rajubuntu	2	13c9f1285025	3 weeks ago	119MB
rajubuntu	1	13c9f1285025	3 weeks ago	119MB
ubuntu	16.04	13c9f1285025	3 weeks ago	119MB

[root@ip-172-31-85-137 dockerfiles]# docker container run -it rajubuntu:2
root@1fd15222b539:/#tree ------already install

[root@ip-172-31-85-137 dockerfiles]# vi Dockerfile
[root@ip-172-31-85-137 dockerfiles]# cat Dockerfile

```
FROM ubuntu:16.04
RUN apt-get update && apt-get install -y tree
RUN touch /tmp/1.txt
RUN touch /tmp/2.txt
RUN touch /tmp/3.txt
RUN touch /tmp/4.txt
RUN touch /tmp/5.txt
RUN touch /tmp/6.txt
```

[root@ip-172-31-85-137 dockerfiles]# docker image build -t rajubuntu:3 .
[root@ip-172-31-85-137 dockerfiles]# docker image ls

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
rajubuntu	3	31a9e63bc1ee	3 minutes ago	146MB
rajubuntu	2	09a779b278b9	21 minutes ago	146MB
rajubuntu	1	13c9f1285025	3 weeks ago	119MB
ubuntu	16.04	13c9f1285025	3 weeks ago	119MB

[root@ip-172-31-85-137 dockerfiles]# docker container run -it rajubuntu:3
root@a4d906421e5a:/# cd tmp/

root@a4d906421e5a:/tmp# ls
1.txt 2.txt 3.txt 4.txt 5.txt 6.txt

root@a4d906421e5a:/tmp# tree .

```
.
|-- 1.txt
|-- 2.txt
|-- 3.txt
|-- 4.txt
|-- 5.txt
`-- 6.txt

0 directories, 6 files
root@a4d906421e5a:/tmp#
```

[root@ip-172-31-85-137 dockerfiles]# vi Dockerfile
[root@ip-172-31-85-137 dockerfiles]# cat Dockerfile

```
FROM ubuntu:16.04
RUN apt-get update && apt-get install -y tree
RUN touch /tmp/1.txt
RUN touch /tmp/2.txt
RUN touch /tmp/3.txt
RUN touch /tmp/4.txt
RUN touch /tmp/5.txt
RUN touch /tmp/6.txt
RUN echo "Raj Kumar Gupta"
RUN echo "Raj Kumar Gupta"
RUN echo "Raj Kumar Gupta"
RUN echo "Raj Kumar Gupta"
[root@ip-172-31-85-137 dockerfiles]# docker image build -t rajubuntu:4 .
```

Note:- When ever you want to add any or change any thing in docker file try to do at the end of file because if you do in begging or in middle then all the command after that will run but if add in the end then only that line will be run rest will take from cache and you will save your time

Docker Command Part-11

[root@ip-172-31-85-137 dockerfiles]# vi Dockerfile
[root@ip-172-31-85-137 dockerfiles]# cat Dockerfile

```
FROM ubuntu:16.04
LABEL name="Raj Gupta"
LABEL email=rajkumargupta14@gmail.com
```

[root@ip-172-31-85-137 dockerfiles]# docker image build -t rajubuntu:5 .

```
Sending build context to Docker daemon  2.048kB
Step 1/3 : FROM ubuntu:16.04
 ---> 13c9f1285025
Step 2/3 : LABEL name="Raj Gupta"
 ---> Running in c1ac8890fb2f
Removing intermediate container c1ac8890fb2f
 ---> a289a8b99ae9
Step 3/3 : LABEL email="rajkumargupta14@gmail.com"
 ---> Running in b14012af0d42
Removing intermediate container b14012af0d42
 ---> c1ab7f6e0b23
Successfully built c1ab7f6e0b23
Successfully tagged rajubuntu:5
```

[root@ip-172-31-85-137 dockerfiles]# docker image ls

```
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
rajubuntu           5                   c1ab7f6e0b23        50 seconds ago      119MB
rajubuntu           4                   f16cb6673695        25 minutes ago      146MB
rajubuntu           3                   31a9e63bc1ee        38 minutes ago      146MB
rajubuntu           2                   09a779b278b9        About an hour ago   146MB
rajubuntu           1                   13c9f1285025        3 weeks ago         119MB
ubuntu              16.04               13c9f1285025        3 weeks ago         119MB
[root@ip-172-31-85-137 dockerfiles]# docker image inspect rajubuntu:5

 "Labels": {
                "email": "rajkumargupta14@gmail.com",
                "name": "Raj Gupta"
            }
```

[root@ip-172-31-85-137 dockerfiles]# vi Dockerfile

[root@ip-172-31-85-137 dockerfiles]# cat Dockerfile

```
FROM ubuntu:16.04
LABEL name="Raj Gupta"
LABEL email="rajkumargupta14@gmail.com"
ENV NAME raj
ENV PASS password
```

[root@ip-172-31-85-137 dockerfiles]# docker image build -t rajubuntu:6 .

```
Sending build context to Docker daemon  2.048kB
Step 1/5 : FROM ubuntu:16.04
 ---> 13c9f1285025
Step 2/5 : LABEL name="Raj Gupta"
 ---> Using cache
 ---> a289a8b99ae9
Step 3/5 : LABEL email="rajkumargupta14@gmail.com"
 ---> Using cache
 ---> c1ab7f6e0b23
Step 4/5 : ENV NAME raj
 ---> Running in d3b63819ba08
Removing intermediate container d3b63819ba08
 ---> c7c7f71dc96c
Step 5/5 : ENV PASS password
 ---> Running in 93591c0824e7
Removing intermediate container 93591c0824e7
 ---> c8c5b5f76b30
Successfully built c8c5b5f76b30
Successfully tagged rajubuntu:6
```

[root@ip-172-31-85-137 dockerfiles]# docker container run -it rajubuntu:6
root@3ce98958a416:/# env

```
PASS=password
HOSTNAME=3ce98958a416
TERM=xterm
NAME=raj
```

[root@ip-172-31-85-137 dockerfiles]# vi Dockerfile
[root@ip-172-31-85-137 dockerfiles]# cat Dockerfile

```
FROM ubuntu:16.04
LABEL name="Raj Gupta"
LABEL email="rajkumargupta14@gmail.com"
ENV NAME raj
ENV PASS password
RUN pwd>/tmp/1st.txt
RUN cd /tmp/
RUN pwd>tmp/2nd.txt
```

[root@ip-172-31-85-137 dockerfiles]# docker image build -t rajubuntu:7 .

```
Sending build context to Docker daemon  2.048kB
Step 1/8 : FROM ubuntu:16.04
```

```
 ---> 13c9f1285025
Step 2/8 : LABEL name="Raj Gupta"
 ---> Using cache
 ---> a289a8b99ae9
Step 3/8 : LABEL email="rajkumargupta14@gmail.com"
 ---> Using cache
 ---> c1ab7f6e0b23
Step 4/8 : ENV NAME raj
 ---> Using cache
 ---> c7c7f71dc96c
Step 5/8 : ENV PASS password
 ---> Using cache
 ---> c8c5b5f76b30
Step 6/8 : RUN pwd>/tmp/1st.txt
 ---> Running in add60b90516f
Removing intermediate container add60b90516f
 ---> 006e5460add7
Step 7/8 : RUN cd /tmp/
 ---> Running in d510bd713aeb
Removing intermediate container d510bd713aeb
 ---> f4fc9d2ae8f4
Step 8/8 : RUN pwd>tmp/2nd.txt
 ---> Running in 5d78620e573b
Removing intermediate container 5d78620e573b
 ---> 69217d97a681
Successfully built 69217d97a681
Successfully tagged rajubuntu:7
```

[root@ip-172-31-85-137 dockerfiles]# docker container run -it rajubuntu:7
root@1d3ed6096e9b:/# cd tmp
root@1d3ed6096e9b:/tmp# ls
1st.txt 2nd.txt
root@1d3ed6096e9b:/tmp# cat 1st.txt
/
root@1d3ed6096e9b:/tmp# cat 2nd.txt
/
root@1d3ed6096e9b:/tmp#

[root@ip-172-31-85-137 dockerfiles]# vi Dockerfile
[root@ip-172-31-85-137 dockerfiles]# cat Dockerfile

```
FROM ubuntu:16.04
LABEL name="Raj Gupta"
LABEL email="rajkumargupta14@gmail.com"
ENV NAME raj
ENV PASS password
RUN pwd>/tmp/1st.txt
RUN cd /tmp/
RUN pwd>tmp/2nd.txt
WORKDIR /tmp
RUN pwd>/tmp/3rd.txt
```

```
[root@ip-172-31-85-137 dockerfiles]# docker image build -t rajubuntu:8 .
Sending build context to Docker daemon  2.048kB
Step 1/10 : FROM ubuntu:16.04
 ---> 13c9f1285025
Step 2/10 : LABEL name="Raj Gupta"
 ---> Using cache
 ---> a289a8b99ae9
Step 3/10 : LABEL email="rajkumargupta14@gmail.com"
 ---> Using cache
 ---> c1ab7f6e0b23
Step 4/10 : ENV NAME raj
 ---> Using cache
 ---> c7c7f71dc96c
Step 5/10 : ENV PASS password
 ---> Using cache
 ---> c8c5b5f76b30
Step 6/10 : RUN pwd>/tmp/1st.txt
 ---> Using cache
 ---> 006e5460add7
Step 7/10 : RUN cd /tmp/
 ---> Using cache
 ---> f4fc9d2ae8f4
Step 8/10 : RUN pwd>tmp/2nd.txt
 ---> Using cache
 ---> 69217d97a681
Step 9/10 : WORKDIR /tmp
 ---> Running in f3e6b4471ca6
Removing intermediate container f3e6b4471ca6
 ---> 71a0eaa3d0cb
Step 10/10 : RUN pwd>/tmp/3rd.txt
 ---> Running in 83417ed2c84d
Removing intermediate container 83417ed2c84d
 ---> 5985dfeffbbd
Successfully built 5985dfeffbbd
Successfully tagged rajubuntu:8

[root@ip-172-31-85-137 dockerfiles]# docker container run -it rajubuntu:8
root@4967807f35a2:/tmp# ls
1st.txt  2nd.txt  3rd.txt
root@4967807f35a2:/tmp# cat 3rd.txt
/tmp
root@4967807f35a2:/tmp# cat 1st.txt
/
root@4967807f35a2:/tmp# cat 2nd.txt
/
root@4967807f35a2:/tmp#
```

Docker Command Part-12

Copy the file from local system to docker container

[root@ip-172-31-82-31 dockerfiles]# vi Dockerfile
[root@ip-172-31-82-31 dockerfiles]# cat Dockerfile
FROM ubuntu:16.04
LABEL name="Raj Gupta"
LABEL email="rajkumargupta14@gmail.com"
ENV NAME raj
ENV PASS password
RUN pwd>/tmp/1st.txt
RUN cd /tmp/
RUN pwd>tmp/2nd.txt
WORKDIR /tmp
RUN pwd>/tmp/3rd.txt
RUN mkdir -p /tmp/project
COPY testproject /tmp/project/

[root@ip-172-31-82-31 dockerfiles]# mkdir testproject
[root@ip-172-31-82-31 dockerfiles]# cd testproject/
[root@ip-172-31-82-31 testproject]# touch 1.txt
[root@ip-172-31-82-31 testproject]# touch 2.txt
[root@ip-172-31-82-31 testproject]# cd ..
[root@ip-172-31-82-31 dockerfiles]# docker image build -t rajubuntu:14 .
[root@ip-172-31-82-31 dockerfiles]# docker container run -P -it rajubuntu:14
root@5e68fe116639:/tmp# ls
1st.txt 2nd.txt 3rd.txt project

root@5e68fe116639:/tmp# cd project/
root@5e68fe116639:/tmp/project# ls
1.txt 2.txt

We can do same thing by ADD command also but one more benefit of add command is if you copy any .tar file from local to container by using ADD command then it will extract the tar file but copy command simply copy the tar file.

root@5e68fe116639:/tmp/project# [root@ip-172-31-82-31 dockerfiles]#
[root@ip-172-31-82-31 dockerfiles]# clear
[root@ip-172-31-82-31 dockerfiles]# vi Dockerfile
[root@ip-172-31-82-31 dockerfiles]# cat Dockerfile
FROM ubuntu:16.04
LABEL name="Raj Gupta"

```
LABEL email="rajkumargupta14@gmail.com"
ENV NAME raj
ENV PASS password
RUN pwd>/tmp/1st.txt
RUN cd /tmp/
RUN pwd>tmp/2nd.txt
WORKDIR /tmp
RUN pwd>/tmp/3rd.txt
RUN mkdir -p /tmp/project2
ADD testproject /tmp/project2/
```

[root@ip-172-31-82-31 dockerfiles]# docker image build -t rajubuntu:81 .
[root@ip-172-31-82-31 dockerfiles]# docker container run -P -it rajubuntu:81

root@457e0e04200f:/tmp# ls
1st.txt 2nd.txt 3rd.txt project2
root@457e0e04200f:/tmp# cd project2/

root@457e0e04200f:/tmp/project2# ls
1.txt 2.txt
root@457e0e04200f:/tmp/project2#

Docker Command Part-13

When ever we want a command will run after every time container will start then use ENTRYPOINT

[root@ip-172-31-82-31 dockerfiles]# vi Dockerfile
[root@ip-172-31-82-31 dockerfiles]# cat Dockerfile

```
FROM ubuntu:16.04
ENV NAME raj
ENV PASS password123
RUN mkdir -p /var/run/sshd
RUN apt-get update
RUN apt-get install -y python tree
ENTRYPOINT [ "tree" ]
```

[root@ip-172-31-82-31 dockerfiles]#
[root@ip-172-31-82-31 dockerfiles]# docker image build -t rajubuntu:85 .
[root@ip-172-31-82-31 dockerfiles]# docker container run -P -itd rajubuntu:85

when you want output of particular command like tree its version only every time

[root@ip-172-31-82-31 dockerfiles]# cat Dockerfile

```
FROM ubuntu:16.04
ENV NAME raj
ENV PASS password123
RUN mkdir -p /var/run/sshd
RUN apt-get update
RUN apt-get install -y python tree
ENTRYPOINT [ "tree" ]
CMD ["--version"]
[root@ip-172-31-82-31 dockerfiles]# docker image build -t rajubuntu:90 .
[root@ip-172-31-82-31 dockerfiles]# docker container run -P -it rajubuntu:90
tree v1.7.0 (c) 1996 - 2014 by Steve Baker, Thomas Moore, Francesc Rocher, Florian Sesser,
Kyosuke Tokoro
```

[root@ip-172-31-82-31 dockerfiles]#

Docker Command Part-14

How to keep data permanent in docker container by using volume

[root@ip-172-31-93-105 ~]# docker volume ls

DRIVER VOLUME NAME

Create mysql container without password

[root@ip-172-31-93-105 ~]# docker container run -d --name mysql1 -e

MYSQL_ALLOW_EMPTY_PASSWORD=true mysql

[root@ip-172-31-93-105 ~]# docker volume ls
DRIVER VOLUME NAME
local bcf7678ec0c851930453fad10b43763caebc4c2e71d6c758863484f8060ad96f

Volume are mounted to

```
  "Volumes": {
              "/var/lib/mysql": {}
          },
```

above value we can get from command [root@ip-172-31-93-105 ~]# docker image inspect mysql

[root@ip-172-31-93-105 ~]# docker volume inspect

bcf7678ec0c851930453fad10b43763caebc4c2e71d6c758863484f8060ad96f

```
[
    {
        "CreatedAt": "2019-07-17T08:36:28Z",
        "Driver": "local",
        "Labels": null,
```

```
        "Mountpoint":
"/var/lib/docker/volumes/bcf7678ec0c851930453fad10b43763caebc4c2e71d6c758863484f8060ad9
6f/_data",
        "Name": "bcf7678ec0c851930453fad10b43763caebc4c2e71d6c758863484f8060ad96f",
        "Options": null,
        "Scope": "local"
    }
]
```

Volume are there in folder

[root@ip-172-31-93-105 ~]# cd
/var/lib/docker/volumes/bcf7678ec0c851930453fad10b43763caebc4c2e71d6c758863484f8060ad96
f/_data

[root@ip-172-31-93-105 _data]# ls
auto.cnf binlog.index client-
cert.pem ibdata1 ibtmp1 mysql.ibd public_key.pem sys
binlog.000001 ca-key.pem client-
key.pem ib_logfile0 #innodb_temp performance_schema server-cert.pem undo_001
binlog.000002 ca.pem ib_buffer_pool ib_logfile1 mysql private_key.pem server-
key.pem undo_002

[root@ip-172-31-93-105 _data]#

[root@ip-172-31-93-105 ~]# docker container ls
CONTAINER
ID IMAGE COMMAND CREATED STATUS PORTS
NAMES
52c3a40f6ef0 mysql "docker-entrypoint.s…" 28 minutes ago Up 28
minutes 3306/tcp, 33060/tcp mysql1

[root@ip-172-31-93-105 ~]# docker container exec -it 52c3a40f6ef0 bash
root@52c3a40f6ef0:/#

Logging into mysql and create database

root@52c3a40f6ef0:/# mysql
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.16 MySQL Community Server - GPL

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database raj;

```
Query OK, 1 row affected (0.01 sec)

mysql> create database test;
Query OK, 1 row affected (0.01 sec)

mysql> create database example;
Query OK, 1 row affected (0.01 sec)

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| example            |
| information_schema |
| mysql              |
| performance_schema |
| raj                |
| sys                |
| test               |
+--------------------+
7 rows in set (0.01 sec)

mysql> exit
Bye
root@52c3a40f6ef0:/#
root@52c3a40f6ef0:/# exit
exit
```

Now we are going to create new container after deleting the container created in above step

```
[root@ip-172-31-93-105 ~]# docker container ls
CONTAINER
ID       IMAGE             COMMAND                  CREATED           STATUS            PORTS
   NAMES
52c3a40f6ef0       mysql             "docker-entrypoint.s…"   43 minutes ago      Up 43
minutes       3306/tcp, 33060/tcp   mysql1

[root@ip-172-31-93-105 ~]# docker container rm -f 52c3a40f6ef0
52c3a40f6ef0

[root@ip-172-31-93-105 ~]# docker container ls
CONTAINER
ID       IMAGE             COMMAND           CREATED           STATUS            PORTS             N
AMES

[root@ip-172-31-93-105 ~]# docker container run -d --name mysql1 -e
MYSQL_ALLOW_EMPTY_PASSWORD=true mysql
4b50bc1137e5b849b00cbdcbb8314c9c11f5116453e016f6111c75dde1e1b28e

[root@ip-172-31-93-105 ~]# docker container exec -it 4b5 bash
```

root@4b50bc1137e5:/# mysql
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.16 MySQL Community Server - GPL

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.01 sec)

mysql> exit
Bye
root@4b50bc1137e5:/# exit
exit
[root@ip-172-31-93-105 ~]#
```

This means if you delete the container and if try to create a new container from same image also then it will not contains the same data because it will create with new volume
---

If you want to use same volume of your old container in place of new the use the below step...It will preserve your data

```
[root@ip-172-31-93-105 ~]# docker volume ls
DRIVER              VOLUME NAME
local               bcf7678ec0c851930453fad10b43763caebc4c2e71d6c758863484f8060ad96f   ----old
local               e700479341274ebe6837f6f4c20f564a6c93d91cf16490c5d825b2081c1ee07f -----new
[root@ip-172-31-93-105 ~]# docker container run -itd -v
bcf7678ec0c851930453fad10b43763caebc4c2e71d6c758863484f8060ad96f:/var/lib/mysql mysql
0384601accbcc5e836a859ee075ff26c05bf3eb5a4f74bb36c509c63c6ed2aa9
[root@ip-172-31-93-105 ~]# docker container exec -it 0384 bash
root@0384601accbc:/# mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.16 MySQL Community Server - GPL
```

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| example            |
| information_schema |
| mysql              |
| performance_schema |
| raj                |
| sys                |
| test               |
+--------------------+
7 rows in set (0.01 sec)

mysql>
```

Docker Command Part-15

If you are going to create a volume that does not exit like below abc...then it will simply going to create a volume with same same

[root@ip-172-31-93-105 ~]# docker volume ls

```
DRIVER              VOLUME NAME
local               8c275dfbba8a8134169997142989597ca98a91afdc8bcde841e0fa3dea9f6770
local               bcf7678ec0c851930453fad10b43763caebc4c2e71d6c758863484f8060ad96f
local               e700479341274ebe6837f6f4c20f564a6c93d91cf16490c5d825b2081c1ee07f
```

[root@ip-172-31-93-105 ~]# docker container run -d --name mysql1 -v abc:/var/lib/mysq4 -e
MYSQL_ALLOW_EMPTY_PASSWORD=true mysql
8e578149e45721964154480f9b8ca9d4d6132a98623d092a4f3d9ba9c1607090

[root@ip-172-31-93-105 ~]# docker container ls

```
CONTAINER
ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
  NAMES
8e578149e457        mysql               "docker-entrypoint.s…"   28 seconds ago      Up 27
seconds        3306/tcp, 33060/tcp   mysql1
```

[root@ip-172-31-93-105 ~]# docker volume ls

```
DRIVER              VOLUME NAME
local               488241083b58309e800e51e6e586b5dae695b5969333231c8928d0b7e9cf56f1
local               8c275dfbba8a8134169997142989597ca98a91afdc8bcde841e0fa3dea9f6770
local               abc
local               bcf7678ec0c851930453fad10b43763caebc4c2e71d6c758863484f8060ad96f
local               e700479341274ebe6837f6f4c20f564a6c93d91cf16490c5d825b2081c1ee07f
[root@ip-172-31-93-105 ~]#
```

To delete or remove the volume used below command

To delete one by one
[root@ip-172-31-93-105 ~]# docker volume ls
DRIVER VOLUME NAME
local 488241083b58309e800e51e6e586b5dae695b5969333231c8928d0b7e9cf56f1
local 8c275dfbba8a8134169997142989597ca98a91afdc8bcde841e0fa3dea9f6770
local abc
local bcf7678ec0c851930453fad10b43763caebc4c2e71d6c758863484f8060ad96f
local e700479341274ebe6837f6f4c20f564a6c93d91cf16490c5d825b2081c1ee07f
local mytest

[root@ip-172-31-93-105 ~]# docker volume rm
488241083b58309e800e51e6e586b5dae695b5969333231c8928d0b7e9cf56f1

488241083b58309e800e51e6e586b5dae695b5969333231c8928d0b7e9cf56f1

[root@ip-172-31-93-105 ~]# docker volume ls
DRIVER VOLUME NAME
local 8c275dfbba8a8134169997142989597ca98a91afdc8bcde841e0fa3dea9f6770
local 92217cb240d362ca3b62fa07ffebf50e28e67b31f907d8b87e51cceb340cafb2
local abc
local bcf7678ec0c851930453fad10b43763caebc4c2e71d6c758863484f8060ad96f
local e700479341274ebe6837f6f4c20f564a6c93d91cf16490c5d825b2081c1ee07f
local mytest
local xyz

To delete more then one at same time
[**root@ip-172-31-93-105 ~]# docker volume ls**
DRIVER VOLUME NAME
local 92217cb240d362ca3b62fa07ffebf50e28e67b31f907d8b87e51cceb340cafb2
local abc
local bcf7678ec0c851930453fad10b43763caebc4c2e71d6c758863484f8060ad96f
local e700479341274ebe6837f6f4c20f564a6c93d91cf16490c5d825b2081c1ee07f
local mytest
local xyz

[root@ip-172-31-93-105 ~]# docker volume rm abc mytest
abc
mytest

[root@ip-172-31-93-105 ~]# docker volume ls
DRIVER VOLUME NAME
local 92217cb240d362ca3b62fa07ffebf50e28e67b31f907d8b87e51cceb340cafb2
local bcf7678ec0c851930453fad10b43763caebc4c2e71d6c758863484f8060ad96f
local e700479341274ebe6837f6f4c20f564a6c93d91cf16490c5d825b2081c1ee07f
local xyz

To delete all unused volume that volume not used by any container use prune command

[root@ip-172-31-93-105 ~]# 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**
Deleted Volumes:
bcf7678ec0c851930453fad10b43763caebc4c2e71d6c758863484f8060ad96f

Total reclaimed space: 183.6MB

[root@ip-172-31-93-105 ~]# docker volume ls
DRIVER VOLUME NAME
local 92217cb240d362ca3b62fa07ffebf50e28e67b31f907d8b87e51cceb340cafb2
local e700479341274ebe6837f6f4c20f564a6c93d91cf16490c5d825b2081c1ee07f
local xyz

[root@ip-172-31-93-105 ~]#

To remove the used volume of running container ...First we need to kill the container then we need to remove it then only we are able to delete volume

[root@ip-172-31-93-105 ~]# docker container ls
CONTAINER
ID IMAGE COMMAND CREATED STATUS PORTS
NAMES
ee6ecf5b36f6 mysql "docker-entrypoint.s…" 33 minutes ago Up 33
minutes 3306/tcp, 33060/tcp mysql1

[root@ip-172-31-93-105 ~]# docker container kill ee6ecf5b36f6
ee6ecf5b36f6

[root@ip-172-31-93-105 ~]# docker container ls
CONTAINER
ID IMAGE COMMAND CREATED STATUS PORTS N
AMES

[root@ip-172-31-93-105 ~]# docker volume ls
DRIVER VOLUME NAME
local 92217cb240d362ca3b62fa07ffebf50e28e67b31f907d8b87e51cceb340cafb2
local e700479341274ebe6837f6f4c20f564a6c93d91cf16490c5d825b2081c1ee07f
local xyz

[root@ip-172-31-93-105 ~]# docker container ls -a
CONTAINER
ID IMAGE COMMAND CREATED STATUS PORTS
NAMES
ee6ecf5b36f6 mysql "docker-entrypoint.s…" 37 minutes ago Exited (137) 2
minutes ago mysql1

52dd7437728e mysql "docker-entrypoint.s…" 3 hours ago Exited (1) 3 hours ago admiring_shtern

[root@ip-172-31-93-105 ~]# docker container rm ee6ecf5b36f6 52dd7437728e
ee6ecf5b36f6
52dd7437728e

[root@ip-172-31-93-105 ~]# docker container ls -a
CONTAINER
ID IMAGE COMMAND CREATED STATUS PORTS N
AMES

[root@ip-172-31-93-105 ~]# 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
Deleted Volumes:
xyz
92217cb240d362ca3b62fa07ffebf50e28e67b31f907d8b87e51cceb340cafb2
e700479341274ebe6837f6f4c20f564a6c93d91cf16490c5d825b2081c1ee07f

Total reclaimed space: 183.6MB

[**root@ip-172-31-93-105 ~]# docker volume ls**
DRIVER VOLUME NAME
[root@ip-172-31-93-105 ~]#

Docker Command Part-15

How to bind any folder from local system to any container to access it. This will just create a link to folder in place of copying all the data, So in this away we can avoid copy same data in multiple place and same the memory

[root@ip-172-31-93-105 ~]# mkdir bind
[root@ip-172-31-93-105 ~]# ld
ld: no input files

[root@ip-172-31-93-105 ~]# ls
bind
[root@ip-172-31-93-105 ~]# cd bind/
[root@ip-172-31-93-105 bind]# vi index.html
[root@ip-172-31-93-105 bind]# cat index.html
<html>
<head>
<title>test</title>
</head>
<body>
<h1 align="center">Docker BindMount Point</h1>
</body>
</html>

[root@ip-172-31-93-105 bind]# pwd
/root/bind

[root@ip-172-31-93-105 bind]# docker container run -it -v /root/bind:/tmp/test/ ubuntu:14.04 bash

root@212c5cb66eb3:/#
[root@ip-172-31-93-105 bind]# pwd
/root/bind

[root@ip-172-31-93-105 bind]# docker container run -rm -it -v
unknown shorthand flag: 'r' in -rm
See 'docker container run --help'.

[root@ip-172-31-93-105 bind]# docker container run -it -v /root/bind:/tmp/test/ ubuntu:14.04 bash
Unable to find image 'ubuntu:14.04' locally
14.04: Pulling from library/ubuntu
a7344f52cb74: Pull complete
515c9bb51536: Pull complete
e1eabe0537eb: Pull complete
4701f1215c13: Pull complete
Digest: sha256:2f7c79927b346e436cc14c92bd4e5bd778c3bd7037f35bc639ac1589a7acfa90
Status: Downloaded newer image for ubuntu:14.04

root@212c5cb66eb3:/# cd /tmp/
root@212c5cb66eb3:/tmp# ls
test
root@212c5cb66eb3:/tmp# cd test/
root@212c5cb66eb3:/tmp/test# ls
index.html
root@212c5cb66eb3:/tmp/test# cat index.html
<html>
<head>
<title>test</title>
</head>
<body>
<h1 align="center">Docker BindMount Point</h1>
</body>
</html>
root@212c5cb66eb3:/tmp/test#

if you change any thing in local system same will reflect in container also

Note:-

In place of this command

docker container run -it -v /root/bind:/tmp/test/ ubuntu:14.04 bash

You can also use

docker container run -it --mount type=bind,source= /root/bind,target=/tmp/test/ ubuntu:14.04 bash

Docker Command Part-16

[root@ip-172-31-40-217 ~]# docker network ls
NETWORK ID NAME DRIVER SCOPE
39450de642d6 bridge bridge local
f6345e9bd840 host host local
514a936c83c1 none null local

[root@ip-172-31-40-217 ~]# docker network inspect bridge

"Containers": {}, -----no container attached till now to bridge network

[**root@ip-172-31-40-217 ~]# docker container ls**

CONTAINER
ID IMAGE COMMAND CREATED STATUS PORTS N
AMES

[root@ip-172-31-40-217 ~]#
[root@ip-172-31-40-217 ~]# docker container run -itd nginx

Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
0a4690c5d889: Pull complete
9719afee3eb7: Pull complete
44446b456159: Pull complete
Digest: sha256:b4b9b3eee194703fc2fa8afa5b7510c77ae70cfba567af1376a573a967c03dbb
Status: Downloaded newer image for nginx:latest
d6110871fab5dbd0503bdde818e597e40860f7397ed868f1d7da72afe3edbefd

[root@ip-172-31-40-217 ~]# docker container ls
CONTAINER
ID IMAGE COMMAND CREATED STATUS PORTS
NAMES
d6110871fab5 nginx "nginx -g 'daemon of…" 23 seconds ago Up 22
seconds 80/tcp suspicious_hamilton

[root@ip-172-31-40-217 ~]# docker network inspect bridge


```
 "Containers": {
         "d6110871fab5dbd0503bdde818e597e40860f7397ed868f1d7da72afe3edbefd": {
            "Name": "suspicious_hamilton",
            "EndpointID":
"af174e6c181f987c4c24318b911eb067b7f2442f9ea151179041c46038ab179f",
            "MacAddress": "02:42:ac:11:00:02",
            "IPv4Address": "172.17.0.2/16",
            "IPv6Address": ""
         }
      },
```


Now one container attached with bridge network

[root@ip-172-31-40-217 ~]# docker container run -it ubuntu:14.04 bash
root@12848ca91085:/#

root@12848ca91085:/# ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=47 time=1.62 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=47 time=1.64 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=47 time=1.63 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=47 time=1.63 ms

This means that this container has internet access.

Docker Command Part-17

How to create own network in docker container

[root@ip-172-31-40-217 ~]# docker network create -d bridge test
5e62d8b1f783f6b0f6cad70f5b74bc992010c56bed543073b17ca56051f219eb

[root@ip-172-31-40-217 ~]# docker network ls

NETWORK ID	NAME	DRIVER	SCOPE
39450de642d6	bridge	bridge	local
f6345e9bd840	host	host	local
514a936c83c1	none	null	local
5e62d8b1f783	test	bridge	local

[root@ip-172-31-40-217 ~]#
[root@ip-172-31-40-217 ~]# ifconfig

br-5e62d8b1f783 Link encap:Ethernet HWaddr 02:42:70:A0:F7:87
inet addr:172.18.0.1 Bcast:172.18.255.255 Mask:255.255.0.0
UP BROADCAST MULTICAST MTU:1500 Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:0 (0.0 b) TX bytes:0 (0.0 b)

Now attach docker to our own created network

[root@ip-172-31-40-217 ~]# docker container run -it --network test ubuntu:14.04 bash
root@75b823d59160:/#
root@75b823d59160:/# ifconfig
eth0 Link encap:Ethernet HWaddr 02:42:ac:12:00:02
inet addr:172.18.0.2 Bcast:172.18.255.255 Mask:255.255.0.0
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:17 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:1326 (1.3 KB) TX bytes:0 (0.0 B)

above connected with below virtual network on system..Once container deleted below entry also deleted

[root@ip-172-31-40-217 ~]# ifconfig
veth8193796 Link encap:Ethernet HWaddr B2:01:04:AC:7F:D4
inet6 addr: fe80::b001:4ff:feac:7fd4/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:12 errors:0 dropped:0 overruns:0 frame:0
TX packets:28 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:1064 (1.0 KiB) TX bytes:2280 (2.2 KiB)

Docker Command Part-18

How to ping a container by using container id

[root@ip-172-31-40-217 ~]# docker network create test2

70b88d7ee77d84967516ec50b184eebc54695076a93ca706ea17e8f2079959e0

[root@ip-172-31-40-217 ~]# docker container run -it --network=test2 ubuntu:14.04 bash
root@18ae8aa73cbf:/# hostname

18ae8aa73cbf

root@18ae8aa73cbf:/# [root@ip-172-31-40-217 ~]#

[root@ip-172-31-40-217 ~]# docker container run -it --network=test2 ubuntu:14.04 bash

root@daf01887e65e:/# ping 18ae8aa73cbf
PING 18ae8aa73cbf (172.19.0.2) 56(84) bytes of data.
64 bytes from 18ae8aa73cbf.test2 (172.19.0.2): icmp_seq=1 ttl=255 time=0.070 ms
64 bytes from 18ae8aa73cbf.test2 (172.19.0.2): icmp_seq=2 ttl=255 time=0.047 ms
64 bytes from 18ae8aa73cbf.test2 (172.19.0.2): icmp_seq=3 ttl=255 time=0.052 ms
64 bytes from 18ae8aa73cbf.test2 (172.19.0.2): icmp_seq=4 ttl=255 time=0.056 ms
64 bytes from 18ae8aa73cbf.test2 (172.19.0.2): icmp_seq=5 ttl=255 time=0.053 ms
64 bytes from 18ae8aa73cbf.test2 (172.19.0.2): icmp_seq=6 ttl=255 time=0.050 ms

64 bytes from 18ae8aa73cbf.test2 (172.19.0.2): icmp_seq=7 ttl=255 time=0.052 ms
^C
--- 18ae8aa73cbf ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6151ms
rtt min/avg/max/mdev = 0.047/0.054/0.070/0.008 ms
root@daf01887e65e:/#

So its pinging by using host name....It means DNS is enable by default in custom network(network created by us but not in default network) ....We can also ping by using container ID, container name,

If you want by default DNS is enable whenever you create container without giving any network name then first we need to delete default network(bridge) then create your custom network with same name(bridge) ..then what ever container will create it will attach with your custom network(bridge) by default..

Docker Command Part-19

whenever you give network name as host then whatever you create container it will use same network as your host computer network

[root@ip-172-31-40-217 ~]# docker network ls

NETWORK ID	NAME	DRIVER	SCOPE
39450de642d6	bridge	bridge	local
f6345e9bd840	host	host	local
514a936c83c1	none	null	local

[root@ip-172-31-40-217 ~]# docker container run -it --network=host ubuntu:14.04 bash
root@ip-172-31-40-217:/# ifconfig ------ This will give same output as your host computer

at a time host driver can attach with only one network

[root@ip-172-31-40-217 ~]# docker network create -d host test
Error response from daemon: network with name test already exists
[root@ip-172-31-40-217 ~]#

The main benefit of using host network is we do not need to do port mapping to access the any web server running in container we can access it by using the host ip directly (in below case by using public ip ec2 server without opening any port

[root@ip-172-31-40-217 ~]# docker container run -itd --network=host nginx
111c466853a3d4118390f0a559a7cf3d0bc86302c4134265148a86bc8abed720

Docker Command Part-20

[root@ip-172-31-40-217 ~]# docker network ls

NETWORK ID	NAME	DRIVER	SCOPE
39450de642d6	bridge	bridge	local
f6345e9bd840	host	host	local
514a936c83c1	none	null	local

1. If we create any container by default it will attached with bridge network.
2. you can also create your custom network and attach your container to custom network.
3. If you attach your container with host network then all the property of host computer network will assign
4. if you do not want to assign any network to your container then use null network

[root@ip-172-31-40-217 ~]# docker container run -it --network=none ubuntu:14.04 bash
root@9d3666594a18:/# ifconfig

```
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
```

```
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

root@9d3666594a18:/#
```

Docker Command Part-21

How to attach a container to multiple network(multiple NIC card)

[root@ip-172-31-40-217 ~]# docker network ls

```
NETWORK ID          NAME                DRIVER              SCOPE
39450de642d6        bridge              bridge              local
f6345e9bd840        host                host                local
514a936c83c1        none                null                local
5e62d8b1f783        test                bridge              local
70b88d7ee77d        test2               bridge              local
```

[root@ip-172-31-40-217 ~]# docker container run -it --network bridge ubuntu:14.04 bash
root@581ec2e1364b:/# ifconfig

```
eth0      Link encap:Ethernet  HWaddr 02:42:ac:11:00:04
          inet addr:172.17.0.4  Bcast:172.17.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:8 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:656 (656.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
```

```
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

root@581ec2e1364b:/#

Now we are going connect test network also with our container

[root@ip-172-31-40-217 ~]# docker network connect test 581ec2e1364b
[root@ip-172-31-40-217 ~]# docker container exec -it 581ec2e1364b bash
root@581ec2e1364b:/# ipconfig

```
bash: ipconfig: command not found
root@581ec2e1364b:/# ifconfig
eth0      Link encap:Ethernet  HWaddr 02:42:ac:11:00:04
          inet addr:172.17.0.4  Bcast:172.17.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:12 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:936 (936.0 B)  TX bytes:0 (0.0 B)

eth1      Link encap:Ethernet  HWaddr 02:42:ac:12:00:02
          inet addr:172.18.0.2  Bcast:172.18.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:13 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1046 (1.0 KB)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

Now we have two NIC card(eth0 and eth1)

Now if you want to detach any network from your container then

[root@ip-172-31-40-217 ~]# docker network disconnect test 581ec2e1364b
[root@ip-172-31-40-217 ~]# docker container exec -it 581ec2e1364b bash
root@581ec2e1364b:/# ifconfig

```
eth0      Link encap:Ethernet  HWaddr 02:42:ac:11:00:04
          inet addr:172.17.0.4  Bcast:172.17.255.255  Mask:255.255.0.0
```

```
        UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
        RX packets:13 errors:0 dropped:0 overruns:0 frame:0
        TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:0
        RX bytes:1006 (1.0 KB)  TX bytes:0 (0.0 B)

lo      Link encap:Local Loopback
        inet addr:127.0.0.1  Mask:255.0.0.0
        UP LOOPBACK RUNNING  MTU:65536  Metric:1
        RX packets:0 errors:0 dropped:0 overruns:0 frame:0
        TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

root@581ec2e1364b:/#

Now we have only one NIC card

If you want to delete any network then

```
[root@ip-172-31-40-217 ~]# docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
39450de642d6        bridge              bridge              local
f6345e9bd840        host                host                local
514a936c83c1        none                null                local
5e62d8b1f783        test                bridge              local
70b88d7ee77d        test2               bridge              local
[root@ip-172-31-40-217 ~]# docker network rm test
test
[root@ip-172-31-40-217 ~]# docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
39450de642d6        bridge              bridge              local
f6345e9bd840        host                host                local
514a936c83c1        none                null                local
70b88d7ee77d        test2               bridge              local
[root@ip-172-31-40-217 ~]#
```

To remove all networks not used by at least one container use the below command

[root@ip-172-31-40-217 ~]# docker network prune
WARNING! This will remove all networks not used by at least one container.
Are you sure you want to continue? **[y/N] y**
[root@ip-172-31-40-217 ~]#

[root@ip-172-31-40-217 ~]# docker network --help

Usage: docker network COMMAND

Manage networks

Commands:
  connect   Connect a container to a network
  create   Create a network
  disconnect Disconnect a container from a network
  inspect   Display detailed information on one or more networks
  ls   List networks
  prune   Remove all unused networks
  rm   Remove one or more networks

Run 'docker network COMMAND --help' for more information on a command.

Docker Command Part-22

How we can create own private repository like docker hub to pull and push own image

[root@ip-172-31-93-16 ~]# docker container run -d -p 5000:5000 --name simple_registry registry

[root@ip-172-31-93-16 ~]# docker container ls

CONTAINER
ID   IMAGE   COMMAND   CREATED   STATUS   PORTS
      NAMES
2f2ccf162441   registry   "/entrypoint.sh /etc…"   22 seconds ago   Up 22
seconds   0.0.0.0:5000->5000/tcp   simple_registry

[root@ip-172-31-93-16 ~]# docker pull redis
[root@ip-172-31-93-16 ~]# docker image ls

REPOSITORY   TAG   IMAGE ID   CREATED   SIZE
redis   latest   598a6f110d01   7 days ago   118MB
registry   latest   f32a97de94e1   4 months ago   25.8MB
[root@ip-172-31-93-16 ~]# docker image tag redis 127.0.0.1:5000/redis
[root@ip-172-31-93-16 ~]# docker container ls
CONTAINER
ID   IMAGE   COMMAND   CREATED   STATUS   PORTS
      NAMES

```
2f2ccf162441        registry            "/entrypoint.sh /etc…"   12 minutes ago      Up 12
minutes       0.0.0.0:5000->5000/tcp   simple_registry
[root@ip-172-31-93-16 ~]# docker image ls
REPOSITORY             TAG                 IMAGE ID            CREATED             SIZE
127.0.0.1:5000/redis   latest              598a6f110d01        7 days ago          118MB
redis                  latest              598a6f110d01        7 days ago          118MB
registry               latest              f32a97de94e1        4 months ago        25.8MB
[root@ip-172-31-93-16 ~]# docker image push 127.0.0.1:5000/redis
The push refers to repository [127.0.0.1:5000/redis]
ecfdefa27746: Pushed
178539e30c1b: Pushed
866b8e9e04ba: Pushed
0cd777ef23ac: Pushed
ec2fceb1c8e2: Pushed
d8a33133e477: Pushed
latest: digest: sha256:9815a0d456dbbef05a5fd5efe4406db003d32a2f91de40f01b62457562e1d7f6
size: 1572
```

you can verify on below location on local system the redis image will be there
http://127.0.0.1:5000/v2/_catalog

Now we are going to delete redis image from system then going to pull from my own repository

```
[root@ip-172-31-93-16 ~]# docker image ls
REPOSITORY             TAG                 IMAGE ID            CREATED             SIZE
127.0.0.1:5000/redis   latest              598a6f110d01        7 days ago          118MB
redis                  latest              598a6f110d01        7 days ago          118MB
registry               latest              f32a97de94e1        4 months ago        25.8MB

[root@ip-172-31-93-16 ~]# docker image rm redis
Untagged: redis:latest
Untagged:
redis@sha256:8888f6cd2509062a377e903e17777b4a6d59c92769f6807f034fa345da9eebcf

[root@ip-172-31-93-16 ~]# docker container ls
CONTAINER
ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
     NAMES
2f2ccf162441        registry            "/entrypoint.sh /etc…"   21 minutes ago      Up 21
minutes       0.0.0.0:5000->5000/tcp   simple_registry


[root@ip-172-31-93-16 ~]# docker image pull 127.0.0.1:5000/redis
Using default tag: latest
```

latest: Pulling from redis
Digest: sha256:9815a0d456dbbef05a5fd5efe4406db003d32a2f91de40f01b62457562e1d7f6
Status: Image is up to date for 127.0.0.1:5000/redis:latest

Docker Command Part-23

How to access private repository/registry insecure(HTTP) way without any issue

When you are going to create your own private repository then only secure repository(HTTPS) are allowed by docker, expect for 127.0.0.0/8 this is insecure but by default it allowed by docker.

Other then this you can allow by doing below

```
[root@ip-172-31-93-16 ~]# cd /etc/docker/
[root@ip-172-31-93-16 docker]# vi daemon.json
[root@ip-172-31-93-16 docker]# cat daemon.json
{
  "insecure-registries" : ["10.0.2.15:5000"]
}

[root@ip-172-31-93-16 docker]# service docker restart
Stopping docker:                                           [  OK  ]
Starting docker:        .                                  [  OK  ]

[root@ip-172-31-93-16 docker]#
```

Now this will allow this also

How to make private repository/registry secure(HTTPS) in other word how to add certificate to private repository/registry to make secure access

[root@ip-172-31-93-16 ~]# mkdir certs
[root@ip-172-31-93-16 ~]# openssl req -newkey rsa:4096 -nodes -sha256 -keyout certs/domain.key

-x509 -days 365 -out certs/domain.crt
Generating a 4096 bit RSA private key
.....++
.....++
writing new private key to 'certs/domain.key'

You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.

Country Name (2 letter code) [XX]:
State or Province Name (full name) []:
Locality Name (eg, city) [Default City]:
Organization Name (eg, company) [Default Company Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:repo.docker.local
Email Address []:

[root@ip-172-31-93-16 ~]#

[root@ip-172-31-93-16 ~]# ls
certs
[root@ip-172-31-93-16 ~]# cd certs/
[root@ip-172-31-93-16 certs]# ls
domain.crt domain.key

[root@ip-172-31-93-16 certs]# cd /etc/docker/
[root@ip-172-31-93-16 docker]# ls
key.json

[root@ip-172-31-93-16 docker]# mkdir certs.d
[root@ip-172-31-93-16 docker]# ls
certs.d key.json

[root@ip-172-31-93-16 docker]# cd certs.d/
[root@ip-172-31-93-16 certs.d]# mkdir repo.docker.local:5000
[root@ip-172-31-93-16 certs.d]# cd .
[root@ip-172-31-93-16 certs.d]# cd
[root@ip-172-31-93-16 ~]# ls
Certs

[root@ip-172-31-93-16 ~]# cp certs/domain.crt /etc/docker/certs.d/repo.docker.local\:5000/ca.crt
[root@ip-172-31-93-16 ~]# service docker restart

Stopping docker: [OK]
Starting docker: . [OK]

[root@ip-172-31-93-16 ~]#

Now create repository with secure

[root@ip-172-31-93-16 ~]# docker container run -d -p 5000:5000 --name secure_registry -v

$(pwd)/certs/:/certs -e REGISTRY_HTTP_TLS_CERTIFICATE=/certs/domain.crt -e REGISTRY_HTTP_TLS_KEY=/certs/domain.key registry
7bfd9f1aea673a98f37a211c37bc727b92c1ab3aae613103eb897613c9cd0de6

[root@ip-172-31-93-16 ~]#

Now we are going to push one image

[root@ip-172-31-93-16 ~]# docker image ls
REPOSITORY TAG IMAGE ID CREATED SIZE
mariadb latest f55f3a2a2d81 3 days ago 354MB
registry latest f32a97de94e1 4 months ago 25.8MB
[root@ip-172-31-93-16 ~]#

[root@ip-172-31-93-16 ~]# docker image tag mariadb repo.docker.local:5000/mariadb
[root@ip-172-31-93-16 ~]# docker image ls
REPOSITORY TAG IMAGE ID CREATED SIZE
mariadb latest f55f3a2a2d81 3 days ago 354MB
repo.docker.local:5000/mariadb latest f55f3a2a2d81 3 days ago 354MB
registry latest f32a97de94e1 4 months ago 25.8MB
[root@ip-172-31-93-16 ~]#

[root@ip-172-31-93-16 ~]# docker image push repo.docker.local:5000/mariadb
The push refers to repository [repo.docker.local:5000/mariadb]
An image does not exist locally with the tag: repo.docker.local:5000/mariadb

[root@ip-172-31-93-16 ~]#

To resolve above we need to add repo.docker.local in path /etc/hosts

[root@ip-172-31-93-16 ~]# vi /etc/hosts
[root@ip-172-31-93-16 ~]# cat /etc/hosts
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1   localhost6 localhost6.localdomain6
172.31.93.16 repo.docker.local
[root@ip-172-31-93-16 ~]#

Now we are able to push the image to private repository in secure away

[root@ip-172-31-93-16 ~]# docker image push repo.docker.local:5000/mariadb
The push refers to repository [repo.docker.local:5000/mariadb]
0a9738aacc8d: Pushed
189fe2319039: Pushed
0aff0ac22d66: Pushed
6c7632269b32: Pushed
829531ae5233: Pushed
69faac9fc0dc: Pushed
3419e6db06bd: Pushed
00f4fc732ccd: Pushed
68ed6b608570: Pushed
38d8a1d432cd: Pushed
75e70aa52609: Pushed
dda151859818: Pushed
fbd2732ad777: Pushed
ba9de9d8475e: Pushed
latest: digest: sha256:86bbf5dffd86bca75ba91cec9a3e08ae3efbef1af233fc19d6b4924079e83f33 size: 3240
[root@ip-172-31-93-16 ~]#

Now our Secure docker repository setup are done

Docker Registry with basic authentication

[root@ip-172-31-93-16 ~]# mkdir auth
[root@ip-172-31-93-16 ~]# docker container run --entrypoint htpasswd registry -bnB raj password
>auth/htpasswd

[root@ip-172-31-93-16 ~]# cat auth/htpasswd
raj:$2y$05$rIV1sexUtv8yPsLZFxEd.uTENSP6Ik95S/y0MZpIiXVu6LGQ44JrO

**[root@ip-172-31-93-16 ~]# docker container run -d **
> -p 5000:5000 \

```
>  --name registry_basic \
>  -v "$(pwd)"/auth:/auth \
>  -v "$(pwd)"/certs:/certs \
>  -e "REGISTRY_AUTH=htpasswd" \
>  -e "REGISTRY_AUTH_HTPASSWD_REALM=Registry Realm" \
>  -e "REGISTRY_AUTH_HTPASSWD_PATH=/auth/htpasswd \
>  -e "REGISTRY_HTTP_TLS_CERTIFICATE=/certs/domain.crt \
>  -e "REGISTRY_HTTP_TLS_KEY=/certs/domain.key \
>  registry
```

Now our privite repositry is scure so to access it we need to logging into it

```
[root@ip-172-31-93-16 ~]# docker login repo.docker.local:5000
Username: raj
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-172-31-93-16 ~]#
[root@ip-172-31-93-16 ~]# docker image push repo.docker.local:5000/mariadb
The push refers to repository [repo.docker.local:5000/mariadb]
0a9738aacc8d: Layer already exists
189fe2319039: Layer already exists
0aff0ac22d66: Layer already exists
6c7632269b32: Layer already exists
829531ae5233: Layer already exists
69faac9fc0dc: Layer already exists
3419e6db06bd: Layer already exists
00f4fc732ccd: Layer already exists
68ed6b608570: Layer already exists
38d8a1d432cd: Layer already exists
75e70aa52609: Layer already exists
dda151859818: Layer already exists
```

Now we are able to push our image to secure repository

Docker Command Part-24

WordPress By using Docker Container

We are going to create 2-tiers container (web and database) like WordPress

Database :-

[root@ip-172-31-95-65 ~]# docker container run --name some-mysql -e
MYSQL_ROOT_PASSWORD=mypassword -d mysql:5.7

061d5275947a25c9b3ae75d3cefb2cb97f5e65246c6dd86b72bd71bb73942c52
[root@ip-172-31-95-65 ~]# docker container inspect 061

So database IP is "IPAddress": "172.17.0.2",

Now Web tiers

[root@ip-172-31-95-65 ~]# docker container run --name some-wordpress -e
WORDPRESS_DB_HOST=172.17.0.2:3306 -e WORDPRESS_DB_USER=root -e
WORDPRESS_DB_PASSWORD=mypassword -d wordpress

30e5b2fc98ee47d9f63ca52b85dae9ee294e14a8fae91b2e70294bc8b7213dae

[root@ip-172-31-95-65 ~]# docker container inspect 30e

So WordPress IP is "IPAddress": "172.17.0.3",

So WordPress installation done by using container
Now to access this container from outside we need to do port mapping

Docker Command Part-25

How to create nginx container by using docker compose

[ec2-user@ip-172-31-95-65 ~]$ sudo -i
[root@ip-172-31-95-65 ~]# vi docker-compose.yml
[root@ip-172-31-95-65 ~]# cat docker-compose.yml

```
version: '3'
services:
  webapp1:
    image: nginx
    ports:
      - "8000:80"
```

[root@ip-172-31-95-65 ~]# docker-compose up -d ------To run the docker compose file
[root@ip-172-31-95-65 ~]# docker container ls

```
CONTAINER
ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
    NAMES
```

89992e0f59e7 nginx "nginx -g 'daemon of…" About a minute ago Up About a minute 0.0.0.0:8000->80/tcp root_webapp1_1

[root@ip-172-31-95-65 ~]# docker network ls
NETWORK ID NAME DRIVER SCOPE
7263963a2ee3 bridge bridge local
6e7b7c015f1e host host local
a8d46c14262d none null local
3c94972e3a5f root_default bridge local

Now to delete all the resource which are created by docker compose file

[root@ip-172-31-95-65 ~]# docker-compose down
Stopping root_webapp1_1 ... done
Removing root_webapp1_1 ... done
Removing network root_default

[root@ip-172-31-95-65 ~]#

Now create two container by use of docker compose

[root@ip-172-31-95-65 ~]# vi docker-compose.yml
[root@ip-172-31-95-65 ~]# cat docker-compose.yml
```
version: '3'
services:
  webapp1:
    image: nginx
    ports:
      - "8000:80"
  webapp2:
    image: nginx
    ports:
      - "8001:80"
```

[root@ip-172-31-95-65 ~]# docker-compose up

If you change any particlar line then only taht container will re-create and reaming will be same like i am going to cahne port nuber of conatiner two

[root@ip-172-31-95-65 ~]# vi docker-compose.yml
[root@ip-172-31-95-65 ~]# cat docker-compose.yml

```
version: '3'
services:
  webapp1:
    image: nginx
    ports:
```

```
    - "8000:80"
  webapp2:
    image: nginx
    ports:
      - "8002:80"
```

[root@ip-172-31-95-65 ~]# docker-compose up -d
root_webapp1_1 is up-to-date
Recreating root_webapp2_1 ... done

Docker Command Part-26

When ever we run the docker compose command then it will find the file docker-compose.yml file, If it will not found then it will give the error

[root@ip-172-31-95-65 ~]# mv docker-compose.yml docker-compose2.yml
[root@ip-172-31-95-65 ~]# ls
docker-compose2.yml

[root@ip-172-31-95-65 ~]# docker-compose up
ERROR:
Can't find a suitable configuration file in this directory or any
parent. Are you in the right directory?

Supported filenames: docker-compose.yml, docker-compose.yaml

[root@ip-172-31-95-65 ~]#

If you want to give any other name in place of docker-compose.yml then

[root@ip-172-31-95-65 ~]# docker-compose -f docker-compose2.yml up -d
root_webapp1_1 is up-to-date
root_webapp2_1 is up-to-date

In this case it will not look for default file(docker-compose.yml) in place of it what ever file you will give it will take that one only.

[root@ip-172-31-95-65 ~]# docker-compose -f docker-compose2.yml down

Stopping root_webapp2_1 ... done
Stopping root_webapp1_1 ... done
Removing root_webapp2_1 ... done
Removing root_webapp1_1 ... done
Removing network root_default

[root@ip-172-31-95-65 ~]#

Docker compose will take json format code also you can convert yml to json in below link

http://convertjson.com/yaml-to-json.htm

[root@ip-172-31-95-65 ~]# vi docker-compose.json
[root@ip-172-31-95-65 ~]# cat docker-compose.json

```
{
  "version": "3",
  "services": {
    "webapp1": {
      "image": "nginx",
      "ports": [
        "8000:80"
      ]
    },
    "webapp2": {
      "image": "nginx",
      "ports": [
        "8001:80"
      ]
    }
  }
}
```

[root@ip-172-31-95-65 ~]# docker-compose -f docker-compose.json up -d
Creating network "root_default" with the default driver
Creating root_webapp1_1 ... done
Creating root_webapp2_1 ... done

Docker Command Part-27

The create command will only create the container but it will not run the container and not create any network

[root@ip-172-31-95-65 ~]# mv docker-compose2.yml docker-compose.yml
[root@ip-172-31-95-65 ~]# docker-compose create
[root@ip-172-31-95-65 ~]# docker network ls

```
NETWORK ID          NAME                DRIVER              SCOPE
7263963a2ee3        bridge              bridge              local
6e7b7c015f1e        host                host                local
a8d46c14262d        none                null                local
[root@ip-172-31-95-65 ~]# docker container ls
CONTAINER
ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
    NAMES
```

To remove the container use the rm command

[root@ip-172-31-95-65 ~]# docker-compose rm

Going to remove root_webapp2_1
Are you sure? [yN] y
Removing root_webapp2_1 ... done
[root@ip-172-31-95-65 ~]#

To create the container and also network run the below command but it will not run the container

[root@ip-172-31-95-65 ~]# docker-compose up --no-start

To start the all stop container use the below command.

[root@ip-172-31-95-65 ~]# docker-compose start

To stop the running container

[root@ip-172-31-95-65 ~]# docker-compose stop

To remove use

[root@ip-172-31-95-65 ~]# docker-compose rm

Docker Command Part-28

To list down image

[root@ip-172-31-95-65 ~]# docker-compose images
Container Repository Tag Image Id Size

[root@ip-172-31-95-65 ~]# docker-compose up -d
Creating root_webapp2_1 ... done
Creating root_webapp1_1 ... done

[root@ip-172-31-95-65 ~]# docker-compose images
Container Repository Tag Image Id Size
---
root_webapp1_1 nginx latest 98ebf73aba75 104 MB
root_webapp2_1 nginx latest 98ebf73aba75 104 MB
[root@ip-172-31-95-65 ~]#

To check the status

```
[root@ip-172-31-95-65 ~]# docker-compose ps
    Name              Command          State         Ports
-------------------------------------------------------------------
root_webapp1_1   nginx -g daemon off;   Up      0.0.0.0:8000->80/tcp
root_webapp2_1   nginx -g daemon off;   Up      0.0.0.0:8002->80/tcp
```

To pause the docker conatiner

```
[root@ip-172-31-95-65 ~]# docker-compose pause
Pausing root_webapp2_1 ... done
Pausing root_webapp1_1 ... done

[root@ip-172-31-95-65 ~]# docker-compose ps
    Name              Command          State         Ports
-------------------------------------------------------------------
root_webapp1_1   nginx -g daemon off;   Paused   0.0.0.0:8000->80/tcp
root_webapp2_1   nginx -g daemon off;   Paused   0.0.0.0:8002->80/tcp
```

To unpause the docker container

```
[root@ip-172-31-95-65 ~]# docker-compose unpause
Unpausing root_webapp1_1 ... done
Unpausing root_webapp2_1 ... done

[root@ip-172-31-95-65 ~]# docker-compose ps
    Name              Command          State         Ports
-------------------------------------------------------------------
root_webapp1_1   nginx -g daemon off;   Up      0.0.0.0:8000->80/tcp
root_webapp2_1   nginx -g daemon off;   Up      0.0.0.0:8002->80/tcp

[root@ip-172-31-95-65 ~]#
```

Docker Command Part-29

To kill all the running container created by docker compose

```
[root@ip-172-31-93-32 ~]# docker-compose kill
Killing root_webapp1_1 ... done

[root@ip-172-31-93-32 ~]# docker-compose ps
    Name              Command          State     Ports
-----------------------------------------------------------
root_webapp1_1   nginx -g daemon off;   Exit 137
```

Now again if we want to start

```
[root@ip-172-31-93-32 ~]# docker-compose start
Starting webapp1 ... done
```

[root@ip-172-31-93-32 ~]# docker-compose ps
```
    Name              Command          State          Ports
-------------------------------------------------------------------
root_webapp1_1   nginx -g daemon off;   Up      0.0.0.0:8000->80/tcp
[root@ip-172-31-93-32 ~]#
```

To know our docker container port 80 is mapped with which port outside world

[root@ip-172-31-93-32 ~]# docker-compose port webapp1 80
0.0.0.0:8000

To see the log coming to docker container

[root@ip-172-31-93-32 ~]# docker-compose logs -f

To get the help use

[root@ip-172-31-93-32 ~]# docker-compose --help

To run any command in the container use the exec command

[root@ip-172-31-93-32 ~]# docker-compose ps
```
    Name              Command          State          Ports
-------------------------------------------------------------------
root_webapp1_1   nginx -g daemon off;   Up      0.0.0.0:8000->80/tcp
[root@ip-172-31-93-32 ~]# docker container ls -a
CONTAINER
ID        IMAGE             COMMAND                  CREATED           STATUS            PORTS
   NAMES
c834fe29a103        nginx             "nginx -g 'daemon of…"   16 minutes ago      Up 13
minutes       0.0.0.0:8000->80/tcp   root_webapp1_1
```

[root@ip-172-31-93-32 ~]# docker-compose exec webapp1 ls --------This ran ls command inside running container webapp1

```
bin   dev  home  lib64  mnt  proc  run   srv  tmp  var
boot  etc  lib   media  opt  root  sbin  sys  usr
[root@ip-172-31-93-32 ~]#
```

[root@ip-172-31-93-32 ~]# docker-compose run webapp1 ls
```
bin   dev  home  lib64  mnt  proc  run   srv  tmp  var
```

```
boot  etc  lib   media  opt  root  sbin  sys  usr

[root@ip-172-31-93-32 ~]# docker-compose ps
    Name            Command         State         Ports
------------------------------------------------------------------------
root_webapp1_1   nginx -g daemon off;   Up      0.0.0.0:8000->80/tcp
[root@ip-172-31-93-32 ~]# docker container ls -a
CONTAINER
ID        IMAGE             COMMAND                  CREATED           STATUS                       PORTS
          NAMES
10d52c52b364        nginx             "ls"                     23 seconds ago      Exited (0) 22 seconds
ago                       root_webapp1_run_816095b9c49c
c834fe29a103        nginx             "nginx -g 'daemon of…"   22 minutes ago      Up 19
minutes             0.0.0.0:8000->80/tcp   root_webapp1_1
[root@ip-172-31-93-32 ~]#
```

Note:-- So difference between run and exec is that

exec command ------> run the command in the same running container
run command--------> run the command in new container and after running the command kill the container

---

To restart the container

```
[root@ip-172-31-93-32 ~]# docker-compose restart
Restarting root_webapp1_1 ... done
[root@ip-172-31-93-32 ~]#
```

To download the image from docker hub

```
[root@ip-172-31-93-32 ~]# docker-compose pull
Pulling webapp1 ... done
[root@ip-172-31-93-32 ~]#
```

To check the docker compose version

```
[root@ip-172-31-93-32 ~]# docker-compose --version
docker-compose version 1.24.0, build 0aa59064
[root@ip-172-31-93-32 ~]#
```

Docker Command Part-30

To use scale command and create docker container as per our requirement

[root@ip-172-31-93-32 ~]# vi docker-compose.yml
[root@ip-172-31-93-32 ~]# cat docker-compose.yml

```
version: '3'
services:
  webapp1:
    image: nginx
  webapp2:
    image: nginx
```

[root@ip-172-31-93-32 ~]# docker-compose up -d
Recreating root_webapp1_1 ... done
Creating root_webapp2_1 ... done

[root@ip-172-31-93-32 ~]# docker-compose ps

Name	Command	State	Ports
root_webapp1_1	nginx -g daemon off;	Up	80/tcp
root_webapp2_1	nginx -g daemon off;	Up	80/tcp

[root@ip-172-31-93-32 ~]# docker-compose scale webapp1=4 webapp2=2
WARNING: The scale command is deprecated. Use the up command with the --scale flag instead.
Starting root_webapp1_1 ... done
Creating root_webapp1_2 ... done
Creating root_webapp1_3 ... done
Creating root_webapp1_4 ... done
Starting root_webapp2_1 ... done
Creating root_webapp2_2 ... done

Total 6(4+2) container are created

[root@ip-172-31-93-32 ~]# docker-compose ps

Name	Command	State	Ports
root_webapp1_1	nginx -g daemon off;	Up	80/tcp
root_webapp1_2	nginx -g daemon off;	Up	80/tcp
root_webapp1_3	nginx -g daemon off;	Up	80/tcp
root_webapp1_4	nginx -g daemon off;	Up	80/tcp
root_webapp2_1	nginx -g daemon off;	Up	80/tcp
root_webapp2_2	nginx -g daemon off;	Up	80/tcp

Now to delete all

[root@ip-172-31-93-32 ~]# docker-compose down
Stopping root_webapp2_2 ... done
Stopping root_webapp1_2 ... done
Stopping root_webapp1_3 ... done
Stopping root_webapp1_4 ... done
Stopping root_webapp1_1 ... done
Stopping root_webapp2_1 ... done
Removing root_webapp2_2 ... done
Removing root_webapp1_2 ... done
Removing root_webapp1_3 ... done
Removing root_webapp1_4 ... done
Removing root_webapp1_1 ... done
Removing root_webapp2_1 ... done
Removing root_webapp1_run_816095b9c49c ... done
Removing network root_default
[root@ip-172-31-93-32 ~]#

Top command:- It will give the all the running process

[root@ip-172-31-93-32 ~]# docker-compose up -d
Creating network "root_default" with the default driver
Creating root_webapp1_1 ... done
Creating root_webapp2_1 ... done

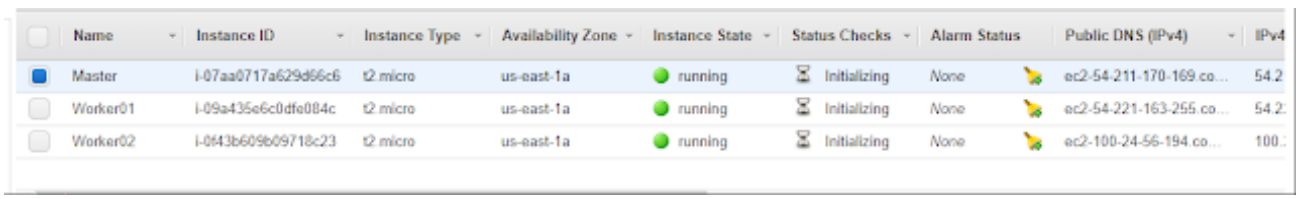
[root@ip-172-31-93-32 ~]# docker-compose top
root_webapp1_1
UID PID PPID C STIME TTY TIME CMD
---
root 332 32749 0 08:57 ? 00:00:00 nginx: master process nginx -g daemon off;
101 471 332 0 08:57 ? 00:00:00 nginx: worker process

root_webapp2_1
UID PID PPID C STIME TTY TIME CMD
---
root 380 358 0 08:57 ? 00:00:00 nginx: master process nginx -g daemon off;
101 508 380 0 08:57 ? 00:00:00 nginx: worker process
[root@ip-172-31-93-32 ~]#

How to install Docker Swarm

Take 3 amazon EC2 server in which we are going to make one server as master and reaming two server as worker

Note:- docker mast be already install in all the server



keep all machine in same network so that they are able to ping each other

Master:-

[root@ip-172-31-40-90 ~]# docker info
Swarm: inactive

[root@ip-172-31-40-90 ~]# **docker swarm init**
Swarm initialized: current node (rby7qdb8hc3ebuuy78vpl0i4v) is now a manager.

To add a worker to this swarm, run the following command:

docker swarm join --token SWMTKN-1-4z0bipbsxzoy2ccn5m22eiems1w0a5du8rlyt6nbvdq3pfegm8-1z76g4wwivmk1s0cp8dethxav 172.31.40.90:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.

[root@ip-172-31-40-90 ~]# docker node ls

ID HOSTNAME STATUS AVAILABILITY MANAGER STATUS ENGINE VERSION
rby7qdb8hc3ebuuy78vpl0i4v * ip-172-31-40-90 Ready Active Leader 18.06.1-ce

[root@ip-172-31-40-90 ~]#

After running the above red mark command on all worker

[root@ip-172-31-40-90 ~]# docker node ls
ID HOSTNAME STATUS AVAILABILITY MANAGER STATUS ENGINE VERSION
q0yqpe0fsihtmoaj1834s1tnf ip-172-31-35-189 Ready Active 18.06.1-ce
rby7qdb8hc3ebuuy78vpl0i4v * ip-172-31-40-90 Ready Active Leader 18.06.1-ce
pzcz2fs38ks9vrwecn30txuv9 ip-172-31-43-91 Ready Active 18.06.1-ce
[root@ip-172-31-40-90 ~]#

So both the node are added with master as worker

[root@ip-172-31-40-90 ~]# docker info
Swarm: active

Worker01:-

[root@ip-172-31-35-189 ~]# docker info
Swarm: inactive

[root@ip-172-31-35-189 ~]# docker swarm join --token SWMTKN-1-
4z0bipbsxzoy2ccn5m22eiems1w0a5du8rlyt6nbvdq3pfegm8-1z76g4wwivmk1s0cp8dethxav
172.31.40.90:2377
This node joined a swarm as a worker.

[root@ip-172-31-35-189 ~]#

[root@ip-172-31-35-189 ~]# docker info
Swarm: active

Worker02:-

[root@ip-172-31-43-91 ~]# docker info
Swarm: inactive

[root@ip-172-31-43-91 ~]# docker swarm join --token SWMTKN-1-
4z0bipbsxzoy2ccn5m22eiems1w0a5du8rlyt6nbvdq3pfegm8-1z76g4wwivmk1s0cp8dethxav
172.31.40.90:2377
This node joined a swarm as a worker.
[root@ip-172-31-43-91 ~]#

[root@ip-172-31-43-91 ~]# docker info
Swarm: active

How to Promote(Master) and Demote(Worker) a docker swarm node

Master:-

To get more details about any node run the inspect command like below for Worker01

[root@ip-172-31-40-90 ~]# docker node inspect u9sg7rw8yvb75o5ybrnppeglo

```
[
    {
        "ID": "u9sg7rw8yvb75o5ybrnppeglo",
        "Version": {
            "Index": 33
```

```
},
"CreatedAt": "2019-07-24T09:41:36.258168494Z",
"UpdatedAt": "2019-07-24T09:41:36.330025943Z",
"Spec": {
    "Labels": {},
    "Role": "worker",
    "Availability": "active"
},
"Description": {
    "Hostname": "ip-172-31-43-91",
    "Platform": {
        "Architecture": "x86_64",
        "OS": "linux"
    },
    "Resources": {
        "NanoCPUs": 1000000000,
        "MemoryBytes": 1033723904
    },
    "Engine": {
        "EngineVersion": "18.06.1-ce",
```

Now to Promote(Master) any node run the below command (promoted both worker node as master)

[root@ip-172-31-40-90 ~]# docker node promote af3y6rhgp328s777kw26g7co2

u9sg7rw8yvb75o5ybrnppeglo
Node af3y6rhgp328s777kw26g7co2 promoted to a manager in the swarm.
Node u9sg7rw8yvb75o5ybrnppeglo promoted to a manager in the swarm.

[root@ip-172-31-40-90 ~]# docker node ls

ID	HOSTNAME	STATUS	AVAILABILITY	MANAGER STATUS	ENGINE VERSION
af3y6rhgp328s777kw26g7co2	ip-172-31-35-189	Ready	Active	Reachable	18.06.1-ce
rby7qdb8hc3ebuuy78vpl0i4v *	ip-172-31-40-90	Ready	Active	Leader	18.06.1-ce
u9sg7rw8yvb75o5ybrnppeglo	ip-172-31-43-91	Ready	Active	Reachable	18.06.1-ce

[root@ip-172-31-40-90 ~]#

Now we can run any command(like ls) on any node because both worker node become as master also

---

Likewise we can Demote(Worker) any node just change promote command to demote

[root@ip-172-31-40-90 ~]# docker node demote af3y6rhgp328s777kw26g7co2
u9sg7rw8yvb75o5ybrnppeglo
Manager af3y6rhgp328s777kw26g7co2 demoted in the swarm.

Manager u9sg7rw8yvb75o5ybrnppeglo demoted in the swarm.

[root@ip-172-31-40-90 ~]# docker node ls

```
ID                            HOSTNAME          STATUS    AVAILABILITY    MANAGER
STATUS    ENGINE VERSION
af3y6rhgp328s777kw26g7co2     ip-172-31-35-
189   Ready               Active                          18.06.1-ce
rby7qdb8hc3ebuuy78vpl0i4v *   ip-172-31-40-
90    Ready               Active          Leader          18.06.1-ce
u9sg7rw8yvb75o5ybrnppeglo     ip-172-31-43-
91    Ready               Active                          18.06.1-ce
```

[root@ip-172-31-40-90 ~]#

Worker01:-

Worker02:-

What is docker service(create,ls,logs) in docker swarm ---part 1

after running below command on Master it will create a container on Worker01(172.31.88.27) then it will start pining to it

[root@ip-172-31-83-166 ~]# docker container run -it alpine ping 172.31.88.27

Unable to find image 'alpine:latest' locally
latest: Pulling from library/alpine

```
9d48c3bd43c5: Pull complete
Digest: sha256:72c42ed48c3a2db31b7dafe17d275b634664a708d901ec9fd57b1529280f01fb
Status: Downloaded newer image for alpine:latest
PING 172.31.88.27 (172.31.88.27): 56 data bytes
64 bytes from 172.31.88.27: seq=0 ttl=254 time=0.942 ms
64 bytes from 172.31.88.27: seq=1 ttl=254 time=0.644 ms
64 bytes from 172.31.88.27: seq=2 ttl=254 time=0.572 ms
64 bytes from 172.31.88.27: seq=3 ttl=254 time=0.635 ms
64 bytes from 172.31.88.27: seq=4 ttl=254 time=0.616 ms
^Z64 bytes from 172.31.88.27: seq=5 ttl=254 time=0.708 ms
64 bytes from 172.31.88.27: seq=6 ttl=254 time=0.716 ms
64 bytes from 172.31.88.27: seq=7 ttl=254 time=0.765 ms
64 bytes from 172.31.88.27: seq=8 ttl=254 time=0.645 ms
64 bytes from 172.31.88.27: seq=9 ttl=254 time=0.629 ms
^C
--- 172.31.88.27 ping statistics ---
10 packets transmitted, 10 packets received, 0% packet loss
round-trip min/avg/max = 0.572/0.687/0.942 ms

[root@ip-172-31-83-166 ~]#
```

[root@ip-172-31-83-166 ~]# docker service --help

```
Usage:  docker service COMMAND

Manage services

Commands:
  create      Create a new service
  inspect     Display detailed information on one or more services
  logs        Fetch the logs of a service or task
  ls          List services
  ps          List the tasks of one or more services
  rm          Remove one or more services
  rollback    Revert changes to a service's configuration
  scale       Scale one or multiple replicated services
  update      Update a service

Run 'docker service COMMAND --help' for more information on a command.
[root@ip-172-31-83-166 ~]#
```

This is also create a container on worker and run the ping command

To create the docker service

[root@ip-172-31-83-166 ~]# docker service create alpine ping 172.31.81.43

docker service ls will list the created services

[root@ip-172-31-83-166 ~]# docker service ls
ID NAME MODE REPLICAS IMAGE PORTS
8ybsu9bgen5e confident_poincare replicated 1/1 alpine:latest
lc7l8xvo9a0o nifty_albattani replicated 1/1 alpine:latest

To see the detail of docker service

[root@ip-172-31-83-166 ~]# docker service inspect 8ybsu9bgen5e

To check the logs of service

[root@ip-172-31-83-166 ~]# docker service logs 8ybsu9bgen5e

Docker service Part-2

To create the four copy of container

[root@ip-172-31-83-166 ~]# docker service create -d --replicas 4 alpine ping 172.31.83.166
befib2ehy1bkmpduqv98qip60

[root@ip-172-31-83-166 ~]# docker service ls
ID NAME MODE REPLICAS IMAGE PORTS

8ybsu9bgen5e confident_poincare replicated 1/1 alpine:latest
lc7l8xvo9a0o nifty_albattani replicated 1/1 alpine:latest
befib2ehy1bk optimistic_kilby replicated **4/4** alpine:latest

[root@ip-172-31-83-166 ~]#

To get the details of all process of service

[root@ip-172-31-83-166 ~]# docker service ps befib2ehy1bk
ID NAME IMAGE NODE DESIRED STATE CURRENT STATE ERROR PORTS
of1nea5konsa optimistic_kilby.1 alpine:latest ip-172-31-81-43 Running Running 6 minutes ago
x2vcdvzoh0q0 optimistic_kilby.2 alpine:latest ip-172-31-81-43 Running Running 6 minutes ago
hcky0bjjk9v3 optimistic_kilby.3 alpine:latest ip-172-31-88-27 Running Running 6 minutes ago
b7xovnxo2u6p optimistic_kilby.4 alpine:latest ip-172-31-83-166 Running Running 6 minutes ago

replicas services will act like auto-scaling in above case if some delete the 1 or more container then master will create again new container so that total count will be four

Docker service(scale,port mapping) Part-3

on master
[root@ip-172-31-83-166 ~]# docker service create -d --replicas 2 alpine ping 172.31.83.166

uz034gaq968jxsriuuob4lxsa

[root@ip-172-31-83-166 ~]#

on worker02

[root@ip-172-31-83-166 ~]# docker service create -d --replicas 3 alpine ping 172.31.81.43

wkk7l8bacrrld2qaupubdr0m2
[root@ip-172-31-83-166 ~]#

[root@ip-172-31-83-166 ~]# docker service ls

ID	NAME	MODE	REPLICAS	IMAGE	PORTS
8ybsu9bgen5e	confident_poincare	replicated	1/1	alpine:latest	
uz034gaq968j	laughing_rosalind	replicated	**2/2**	alpine:latest	
lc7l8xvo9a0o	nifty_albattani	replicated	1/1	alpine:latest	
befib2ehy1bk	optimistic_kilby	replicated	4/4	alpine:latest	
wkk7l8bacrrl	sharp_hawking	replicated	**3/3**	alpine:latest	

[root@ip-172-31-83-166 ~]#

now to increase and decrease replicas

[root@ip-172-31-83-166 ~]# docker service scale wkk7l8bacrrl=7
wkk7l8bacrrl scaled to 7
overall progress: 7 out of 7 tasks
1/7: running [==>]
2/7: running [==>]
3/7: running [==>]
4/7: running [==>]
5/7: running [==>]
6/7: running [==>]
7/7: running [==>]
verify: Service converged

[root@ip-172-31-83-166 ~]#

At same time

[root@ip-172-31-83-166 ~]# docker service scale wkk7l8bacrrl=5 befib2ehy1bk=8
wkk7l8bacrrl scaled to 5
befib2ehy1bk scaled to 8
overall progress: 5 out of 5 tasks
1/5: running [==>]
2/5: running [==>]
3/5: running [==>]
4/5: running [==>]
5/5: running [==>]
verify: Service converged
overall progress: 8 out of 8 tasks
1/8: running [==>]
2/8: running [==>]
3/8: running [==>]

```
4/8: running   [==================================================>]
5/8: running   [==================================================>]
6/8: running   [==================================================>]
7/8: running   [==================================================>]
8/8: running   [==================================================>]
verify: Service converged
[root@ip-172-31-83-166 ~]#
```

[root@ip-172-31-83-166 ~]# docker service ls

```
ID             NAME                MODE         REPLICAS   IMAGE          PORTS
8ybsu9bgen5e   confident_poincare  replicated   1/1        alpine:latest
uz034gaq968j   laughing_rosalind   replicated   2/2        alpine:latest
lc7l8xvo9a0o   nifty_albattani     replicated   1/1        alpine:latest
befib2ehy1bk   optimistic_kilby    replicated   8/8        alpine:latest
wkk7l8bacrrl   sharp_hawking       replicated   5/5        alpine:latest
```

```
[root@ip-172-31-83-166 ~]#
```

Now to remove service

[**root@ip-172-31-83-166 ~]# docker service rm befib2ehy1bk wkk7l8bacrrl**

```
befib2ehy1bk
wkk7l8bacrrl
```

[**root@ip-172-31-83-166 ~]# docker service ls**

```
ID             NAME                MODE         REPLICAS   IMAGE          PORTS
8ybsu9bgen5e   confident_poincare  replicated   1/1        alpine:latest
uz034gaq968j   laughing_rosalind   replicated   2/2        alpine:latest
lc7l8xvo9a0o   nifty_albattani     replicated   1/1        alpine:latest
[root@ip-172-31-83-166 ~]#
```

Port mapping:-

[root@ip-172-31-83-166 ~]# docker service create -d -p 8090:80 nginx

```
r3emkkfmh4a37wuxrneh0ovru
```

[root@ip-172-31-83-166 ~]# docker service ls

```
ID             NAME                MODE         REPLICAS   IMAGE          PORTS
r3emkkfmh4a3   tender_chaplygin    replicated   1/1        nginx:latest   *:8090->80/tcp
```

[root@ip-172-31-83-166 ~]#

now you are able to access it from any nodes(master,worker01,worker02)
http://54.210.168.62:8090/

Docker Service Mode (Replicated, global)

[root@ip-172-31-83-166 ~]# docker service create -d --replicas=3 alpine ping 172.31.83.166

gnbq42aqu083vcsf1f8w8d7uz

[root@ip-172-31-83-166 ~]# docker service ls

```
ID             NAME             MODE         REPLICAS   IMAGE          PORTS
gnbq42aqu083   gallant_lamport  replicated   3/3        alpine:latest
ip6hivkzqohi   happy_thompson   replicated   1/1        alpine:latest
r3emkkfmh4a3   tender_chaplygin replicated   1/1        nginx:latest   *:8090->80/tcp
```

If you want to see the visual diagram of all your master and node then run the below command on master

[root@ip-172-31-83-166 ~]# docker service create --name=viz --publish=8080:8080/tcp --constraint=node.role==manager --
mount=type=bind,src=/var/run/docker.sock,dst=/var/run/docker.sock dockersamples/visualizer
6iamsonw3jrsedeudqhun1bmx
overall progress: 1 out of 1 tasks
1/1: running [==>]
verify: Service converged
[root@ip-172-31-83-166 ~]#

Now you in browser use the blow to see

http://34.203.202.233:8080/

If you want to create container service on all nodes even if you add any node in future then

[root@ip-172-31-83-166 ~]# docker service create --mode=global alpine ping 8.8.8.8
snrsqqnnblwx39tgccjp3w4ts
overall progress: 3 out of 3 tasks
vvj0e7oybvg7: running [==>]
ly6ri9vtmku7: running [==>]
nlkj6kdokxk3: running [==>]
verify: Service converged

[root@ip-172-31-83-166 ~]# docker service ls

```
ID             NAME             MODE         REPLICAS   IMAGE                          PORTS
snrsqqnnblwx   festive_edison   global       3/3        alpine:latest
gnbq42aqu083   gallant_lamport  replicated   3/3        alpine:latest
ip6hivkzqohi   happy_thompson   replicated   1/1        alpine:latest
r3emkkfmh4a3   tender_chaplygin replicated   1/1        nginx:latest                   *:8090->80/tcp
6iamsonw3jrs   viz              replicated   1/1        dockersamples/visualizer:latest *:8080->8080/tcp
```

[root@ip-172-31-83-166 ~]#

Docker swarm Label and Constraint

We can create label in two way

1. Node labels ----for this we need to run on master, we need access on master
2. Engine labels ----we need to run on worker, so we don't required access of master.

If you want to create some service on only master then

[root@ip-172-31-83-166 ~]# docker service create --replicas=3 --constraint="node.role==manager"

alpine ping 172.31.83.166
mr0nf8wjykhd8vhzzpexs20vl
overall progress: 3 out of 3 tasks
1/3: running [==>]
2/3: running [==>]
3/3: running [==>]
verify: Service converged
[root@ip-172-31-83-166 ~]#

---

and to create on only worker

[root@ip-172-31-83-166 ~]# docker service create --replicas=3 --constraint="node.role==worker"

alpine ping 172.31.83.166
qixhwn2mje7guhqlcazl56bxt
overall progress: 3 out of 3 tasks
1/3: running [==>]
2/3: running [==>]
3/3: running [==>]
verify: Service converged
[root@ip-172-31-83-166 ~]#

In this way we can add label to node

[root@ip-172-31-83-166 ~]# docker node update --label-add="ssd=true" ip-172-31-81-43
ip-172-31-81-43
[root@ip-172-31-83-166 ~]#

In this way we can create service on only particular label node

[root@ip-172-31-83-166 ~]# docker service create --constraint="node.labels.ssd==true" --replicas=3 -d alpine 172.31.81.43

0a6ezsi9x2qhy7d523ctca0g8
[root@ip-172-31-83-166 ~]#

We can also create label on worker by using below

Run the below command on worker02

[root@ip-172-31-81-43 ~]# vi /etc/docker/daemon.json
[root@ip-172-31-81-43 ~]# cat /etc/docker/daemon.json
{
"labels : ["name=raj"]
}
[root@ip-172-31-81-43 ~]# service docker restart

Now run the below command on master it will create on service on worker02 only

[root@ip-172-31-83-166 ~]# docker service create --constraint="engine.labels.name==raj" --
replicas=3 -d alpine 172.31.81.43
mcvert6gvmw3u464ikhjwid2y
[root@ip-172-31-83-166 ~]#