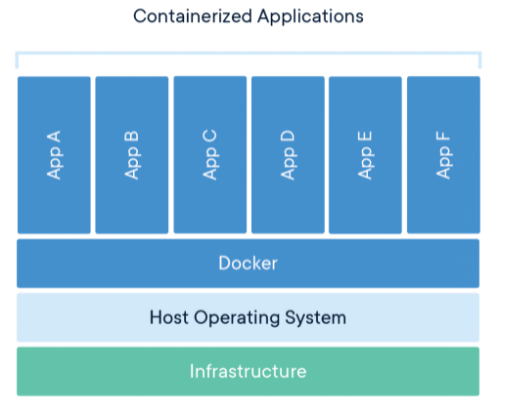
**Docker**:

**1)Whats is docker?**

**Docker** is an open source containerization platform. It enables developers to package applications into containers.

<https://www.docker.com/get-started>



Installation on Linux:-

<https://docs.docker.com/engine/install/>

#yum install docker

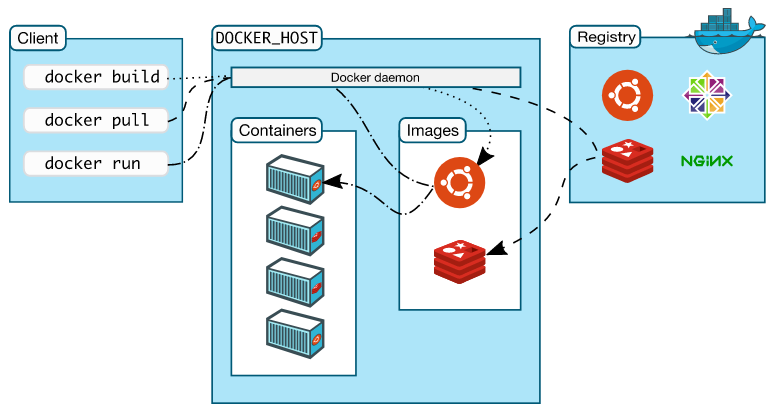
[root@ip-172-31-46-46 ~]# docker --version

Docker version 20.10.7, build f0df350

#Install docker on Windows:

<https://docs.docker.com/desktop/windows/install/>

**Docker Architecture:**



|  |
| --- |
| Management Commands: |
| builder - Manage builds |
| config - Manage Docker configs |
| container - Manage containers |
| context - Manage contexts |
| image - Manage images |
| manifest - Manage Docker image manifests and manifest lists |
| network - Manage networks |
| node - Manage Swarm nodes |
| plugin - Manage plugins |
| secret - Manage Docker secrets |
| service - Manage services |
| stack - Manage Docker stacks |
| swarm - Manage Swarm |
| system - Manage Docker |
| trust - Manage trust on Docker images |
| volume - Manage volumes |

|  |
| --- |
| Commands: |
| attach Attach local standard input, output, and error streams to a running container |
| build Build an image from a Dockerfile |
| commit Create a new image from a container's changes |
| cp Copy files/folders between a container and the local filesystem |
| create Create a new container |
| diff Inspect changes to files or directories on a container's filesystem |
| events Get real time events from the server |
| exec Run a command in a running container |
| export Export a container's filesystem as a tar archive |
| history Show the history of an image |
| images List images |
| import Import the contents from a tarball to create a filesystem image |
| info Display system-wide information |
| inspect Return low-level information on Docker objects |
| kill Kill one or more running containers |
| load Load an image from a tar archive or STDIN |
| login Log in to a Docker registry |
| logout Log out from a Docker registry |
| logs Fetch the logs of a container |
| pause Pause all processes within one or more containers |
| port List port mappings or a specific mapping for the container |
| ps List containers |
| pull Pull an image or a repository from a registry |
| push Push an image or a repository to a registry |
| rename Rename a container |
| restart Restart one or more containers |
| rm Remove one or more containers |
| rmi Remove one or more images |
| run Run a command in a new container |
| save Save one or more images to a tar archive (streamed to STDOUT by default) |
| search Search the Docker Hub for images |
| start Start one or more stopped containers |
| stats Display a live stream of container(s) resource usage statistics |
| stop Stop one or more running containers |
| tag Create a tag TARGET\_IMAGE that refers to SOURCE\_IMAGE |
| top Display the running processes of a container |
| unpause Unpause all processes within one or more containers |
| update Update configuration of one or more containers |
| version Show the Docker version information |
| wait Block until one or more containers stop, then print their exit codes |

Docker Command one By One:

List Running Container:

[root@ip-172-31-46-46 ~]# docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

[root@ip-172-31-46-46 ~]# docker container ls

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

[root@ip-172-31-46-46 ~]#

[root@ip-172-31-46-46 ~]# docker run centos

Unable to find image 'centos:latest' locally

latest: Pulling from library/centos

a1d0c7532777: Pull complete

Digest: sha256:a27fd8080b517143cbbbab9dfb7c8571c40d67d534bbdee55bd6c473f432b177

Status: Downloaded newer image for centos:latest

Bydefault tag is latest:

[root@ip-172-31-46-46 ~]# docker pull python

Using default tag: latest

latest: Pulling from library/python

0c6b8ff8c37e: Pull complete

412caad352a3: Pull complete

e6d3e61f7a50: Pull complete

461bb1d8c517: Pull complete

808edda3c2e8: Pull complete

724cfd2dc19b: Pull complete

8bd4965a24ab: Pull complete

fccd5fa208a8: Pull complete

af1ca64a0eec: Pull complete

Digest: sha256:a7a73f894e756267b2bac3b068e51ad50aa06f16855a9c6b208630d48937796f

Status: Downloaded newer image for python:latest

List downloaded images:

[root@ip-172-31-46-46 ~]# docker images

REPOSITORY TAG IMAGE ID CREATED SIZE

python latest e2e732b7951f 3 days ago 886MB

centos latest 5d0da3dc9764 4 months ago 231MB

[root@ip-172-31-46-46 ~]# docker run centos sleep 60

[root@ip-172-31-46-46 ~]# docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

678045cad1dd centos "sleep 60" 8 seconds ago Up 7 seconds compassionate\_carson

[root@ip-172-31-46-46 ~]# docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

678045cad1dd centos "sleep 60" 14 seconds ago Up 13 seconds compassionate\_carson

[root@ip-172-31-46-46 ~]# docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

678045cad1dd centos "sleep 60" 41 seconds ago Up 40 seconds compassionate\_carson

[root@ip-172-31-46-46 ~]# docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

678045cad1dd centos "sleep 60" 44 seconds ago Up 43 seconds compassionate\_carson

[root@ip-172-31-46-46 ~]# docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

678045cad1dd centos "sleep 60" 46 seconds ago Up 45 seconds compassionate\_carson

[root@ip-172-31-46-46 ~]# docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

678045cad1dd centos "sleep 60" 48 seconds ago Up 46 seconds compassionate\_carson

[root@ip-172-31-46-46 ~]# docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

678045cad1dd centos "sleep 60" 49 seconds ago Up 48 seconds compassionate\_carson

[root@ip-172-31-46-46 ~]# docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

678045cad1dd centos "sleep 60" 50 seconds ago Up 49 seconds compassionate\_carson

[root@ip-172-31-46-46 ~]# docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

678045cad1dd centos "sleep 60" 58 seconds ago Up 57 seconds compassionate\_carson

[root@ip-172-31-46-46 ~]# docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

[root@ip-172-31-46-46 ~]#

Detach Mode:

[root@ip-172-31-46-46 ~]# docker run -d ubuntu sleep 60

Unable to find image 'ubuntu:latest' locally

latest: Pulling from library/ubuntu

ea362f368469: Pull complete

Digest: sha256:b5a61709a9a44284d88fb12e5c48db0409cfad5b69d4ff8224077c57302df9cf

Status: Downloaded newer image for ubuntu:latest

dbcb6de560190af826739cf2c882d9c40e18b28b0845438fd5b43ec41284dcae

[root@ip-172-31-46-46 ~]#

[root@ip-172-31-46-46 ~]# docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

dbcb6de56019 ubuntu "sleep 60" 11 seconds ago Up 10 seconds happy\_hawking

List all stopped and running container:

[root@ip-172-31-46-46 ~]# docker ps -a

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

dbcb6de56019 ubuntu "sleep 60" About a minute ago Exited (0) 39 seconds ago happy\_hawking

678045cad1dd centos "sleep 60" 23 minutes ago Exited (0) 22 minutes ago compassionate\_carson

6a9928081c87 centos "sleep 60" 24 minutes ago Exited (0) 23 minutes ago trusting\_montalcini

6f5122b91ba2 centos "/bin/bash" 28 minutes ago Exited (0) 28 minutes ago quizzical\_heisenberg

Interactive Mode:

[root@ip-172-31-46-46 ~]# docker run -d -it ubuntu

e5272ed8038ff7159c9541c6191de2ae9f1b304943465158055aea8db58468bb

[root@ip-172-31-46-46 ~]# docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

e5272ed8038f ubuntu "bash" 3 seconds ago Up 2 seconds determined\_dewdney

[root@ip-172-31-46-46 ~]# docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

e5272ed8038f ubuntu "bash" 5 seconds ago Up 5 seconds determined\_dewdney

[root@ip-172-31-46-46 ~]#

**Stop Container:**

[root@ip-172-31-46-46 ~]# docker container stop e5272ed8038f

e5272ed8038f

**Start Container**

[root@ip-172-31-46-46 ~]# docker container start e5272ed8038f

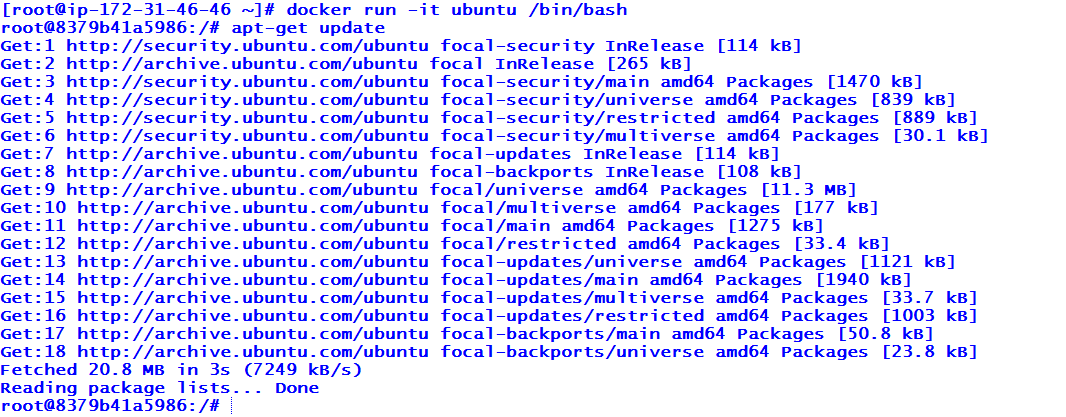
e5272ed8038f

[root@ip-172-31-46-46 ~]# docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

e5272ed8038f ubuntu "bash" 4 minutes ago Up 3 seconds determined\_dewdney

**Start Container and login**



Install apache inside container:



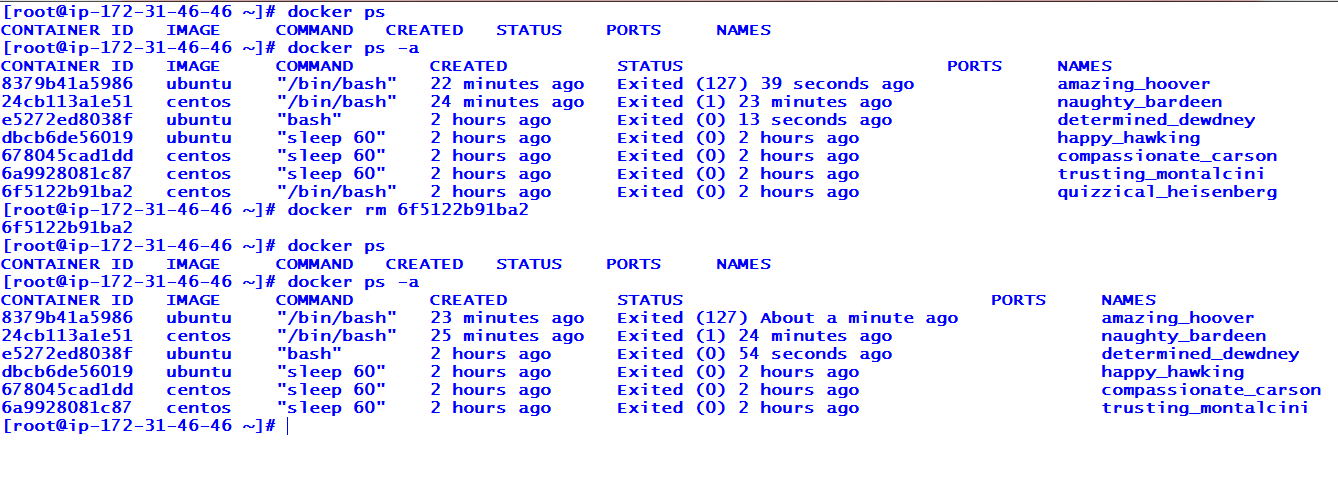
Shortcut to logout from container: ctr+p+q

Check container memory:





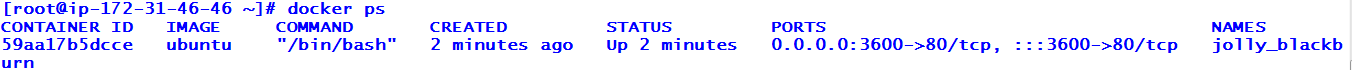
How to remove container:



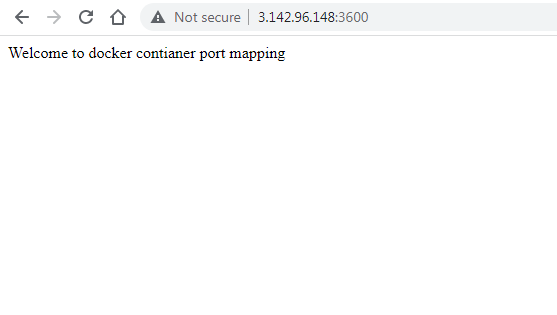
Expose Port/Port Mapping:

Step:1 #docker run -it -p 3600:80 ubuntu /bin/bash

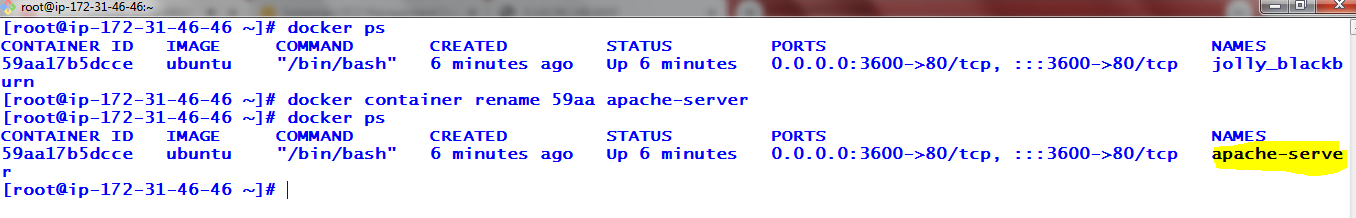
3600 is docker host port and 80 is container port

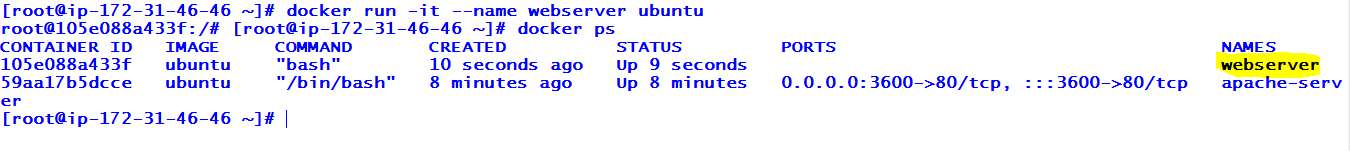


<http://3.142.96.148:3600/>

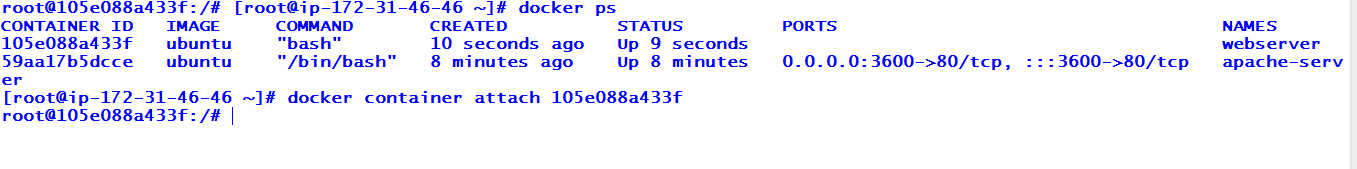


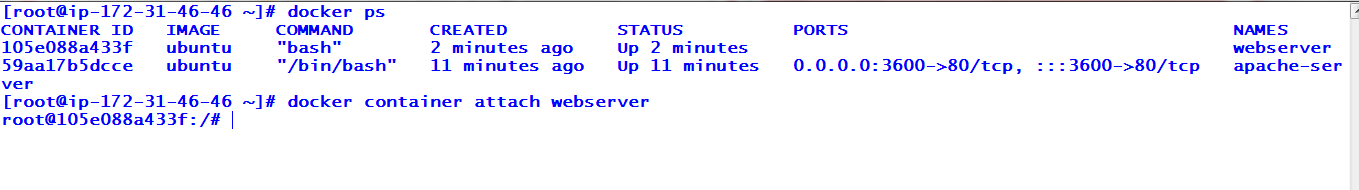
Rename container Name:



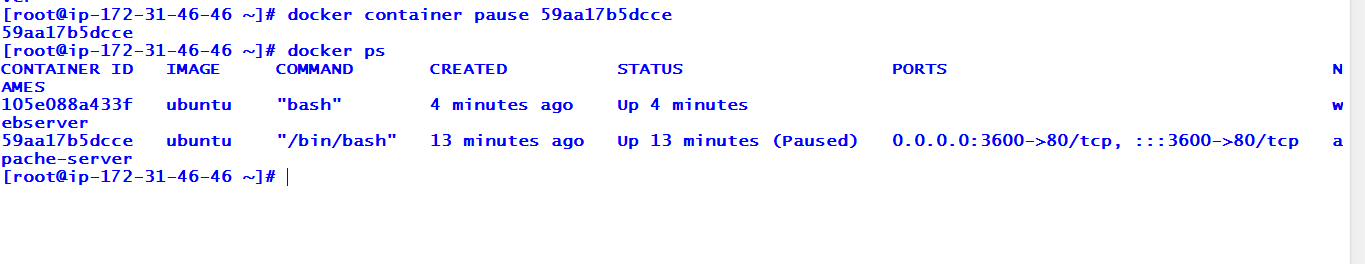


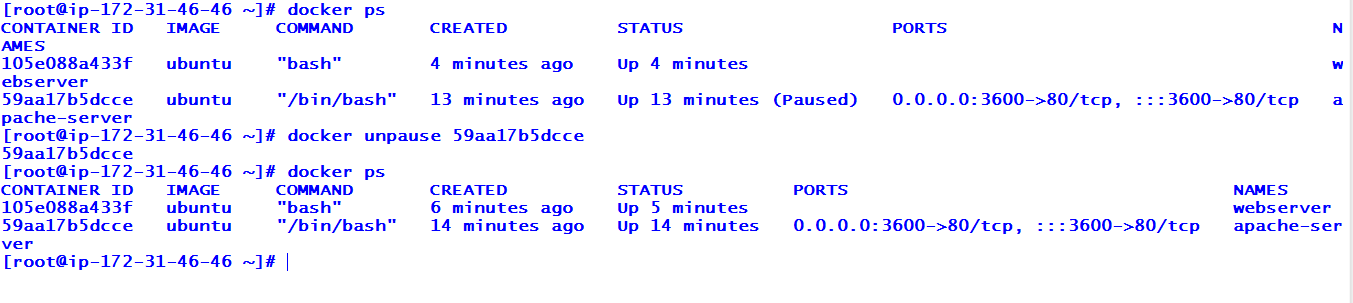
Login to docker container using id and name:





Pause/Unpause Container:

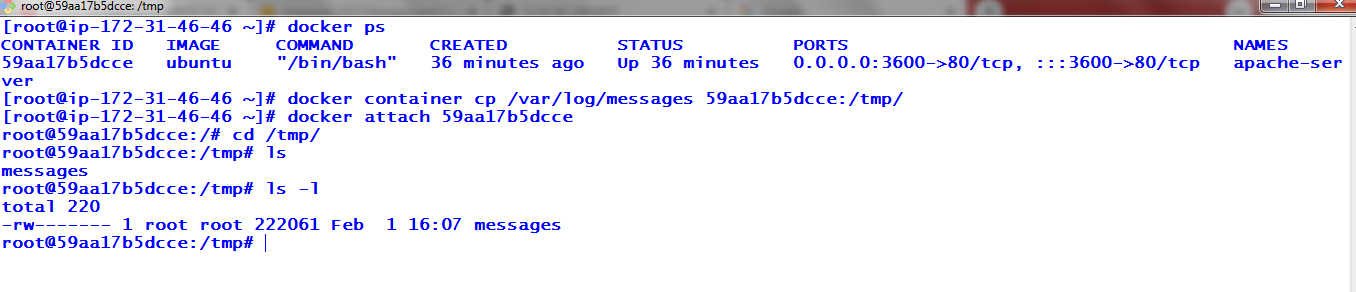




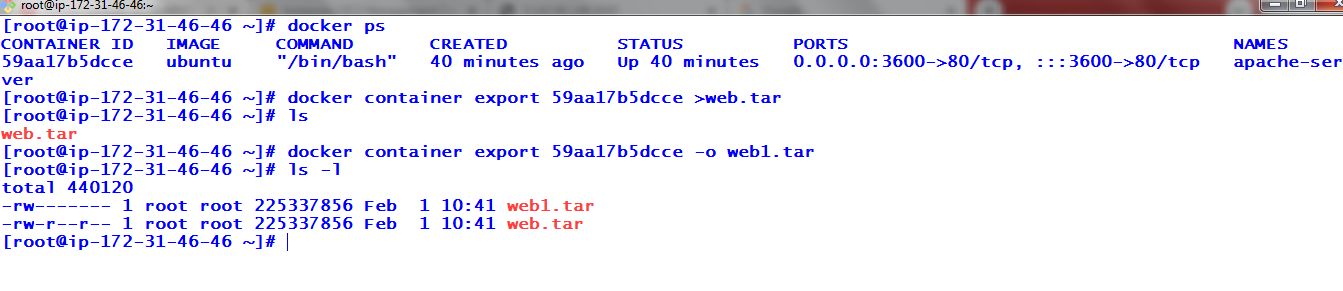
Kill Container:

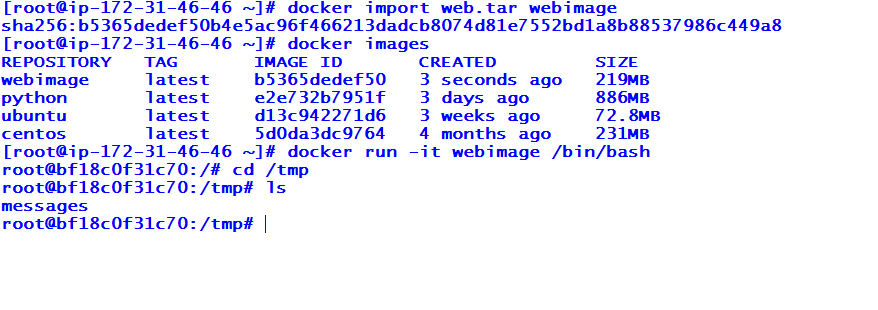
[root@ip-172-31-46-46 ~]# docker container kill 105e088a433f

Copy File inside container:



Export and Import Container:





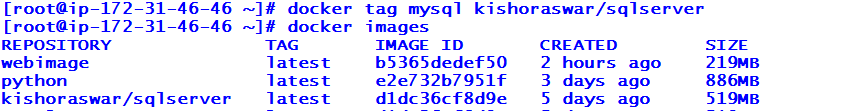
Docker Hub:

<https://hub.docker.com/>

Download image from dockerhub

docker pull httpd

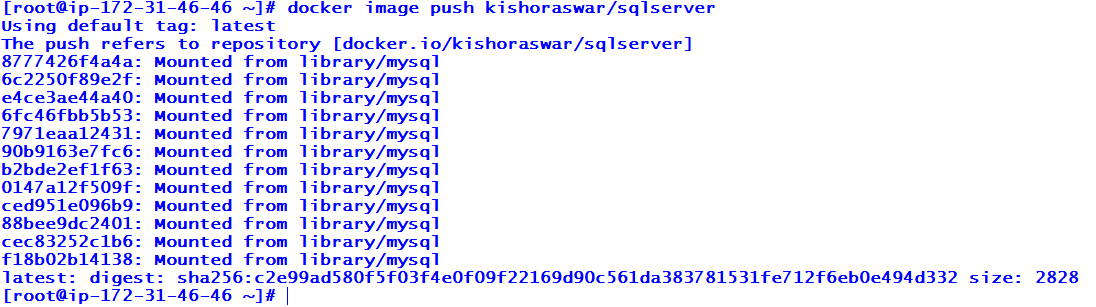
Assign new tag to image.



Upload image to dockerhub

Step-1) execute command docker login

Step-2) docker image push.



docker pull kishoraswar/myimage:cc-nginx

**Docker Volume:**

[root@ip-172-31-46-46 ~]# docker volume --help

Usage: docker volume COMMAND

Manage volumes

Commands:

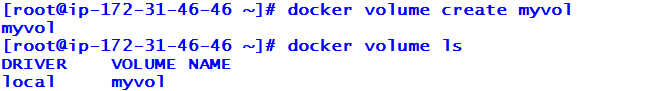
create Create a volume

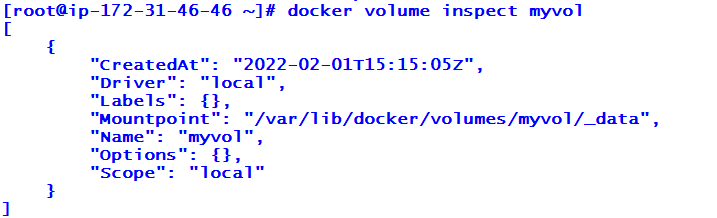
inspect Display detailed information on one or more volumes

ls List volumes

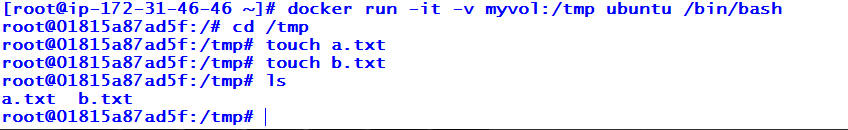
prune Remove all unused local volumes

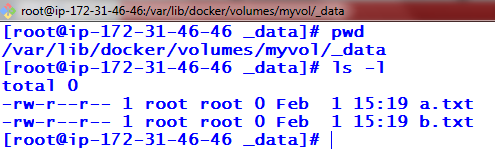
rm Remove one or more volumes





Created container and attached volume..



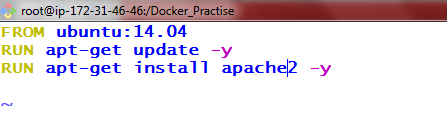


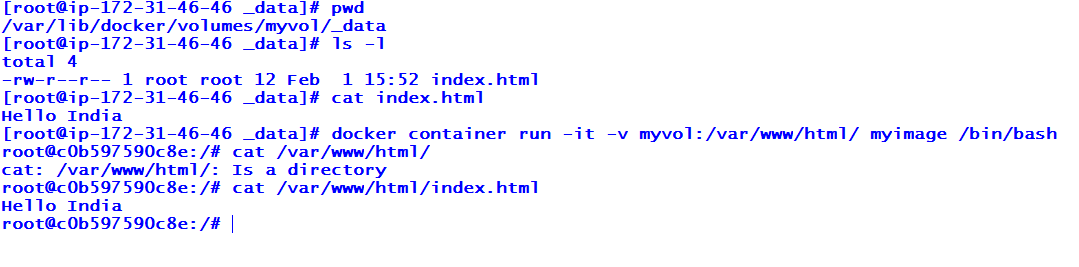
**Build Simple Custom Image::**

Create file named Dockerfile and below commands.

Execute below command on host:

docker image build -t myimage .





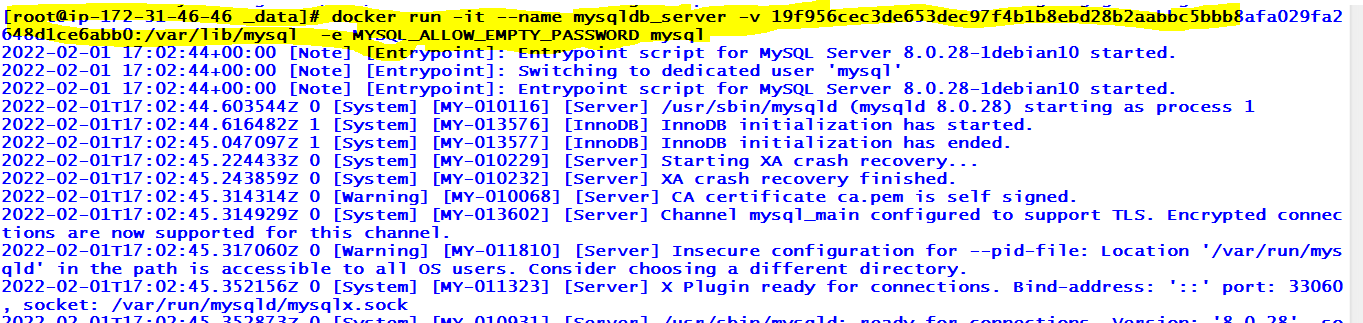
We have created index.html inside our volume.

POC:

Pull mysql image

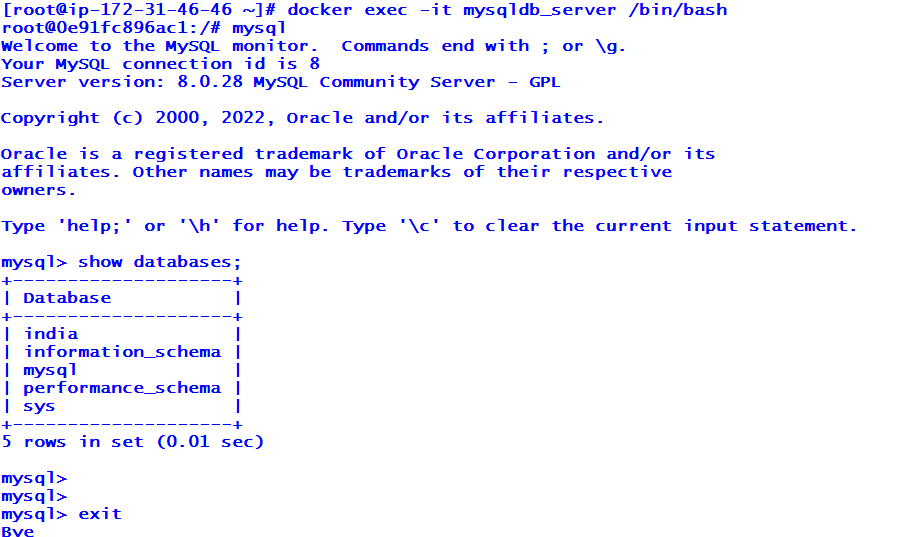
Create container

Create db

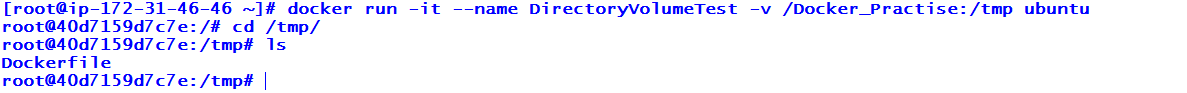


Inspect image you will find mount point for db.

Attach same to another image.



We can attach any directory with container using below option



Note: Remove all stopped container

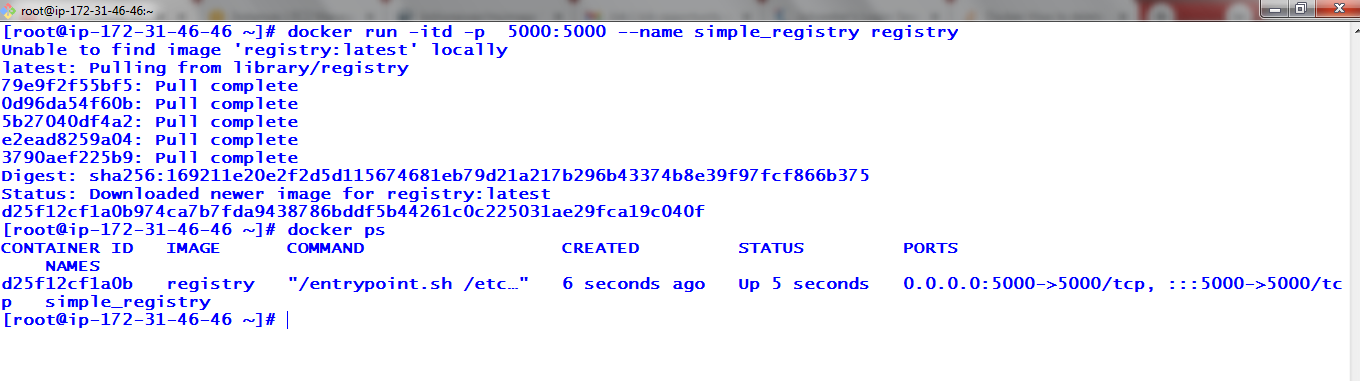
docker rm $(docker ps -a -q)

docker image prune --all --force

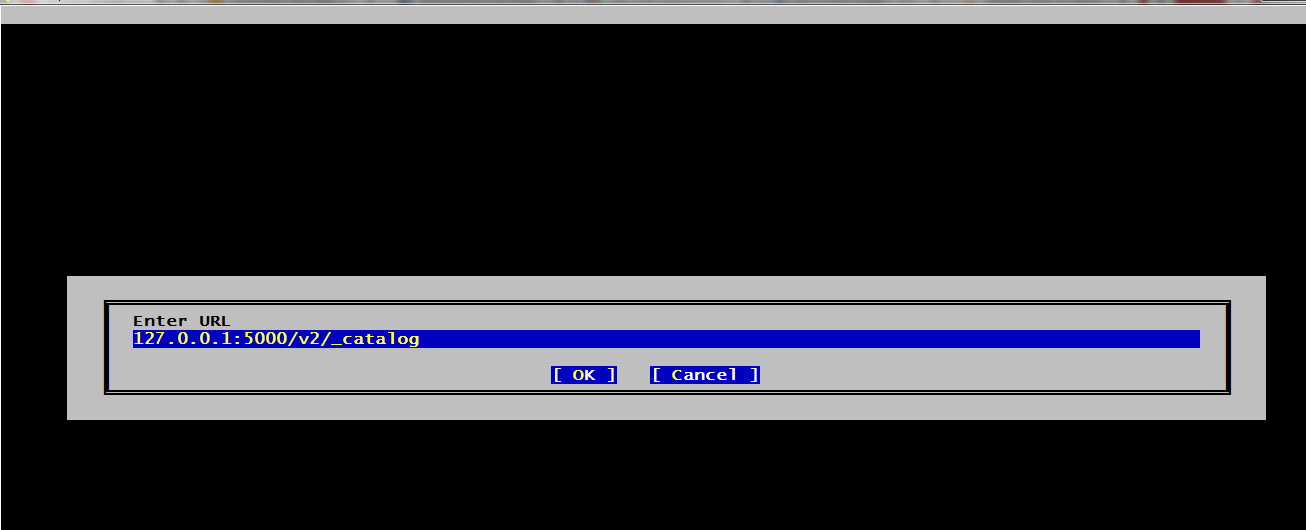
Docker Private Registry:

Steps:

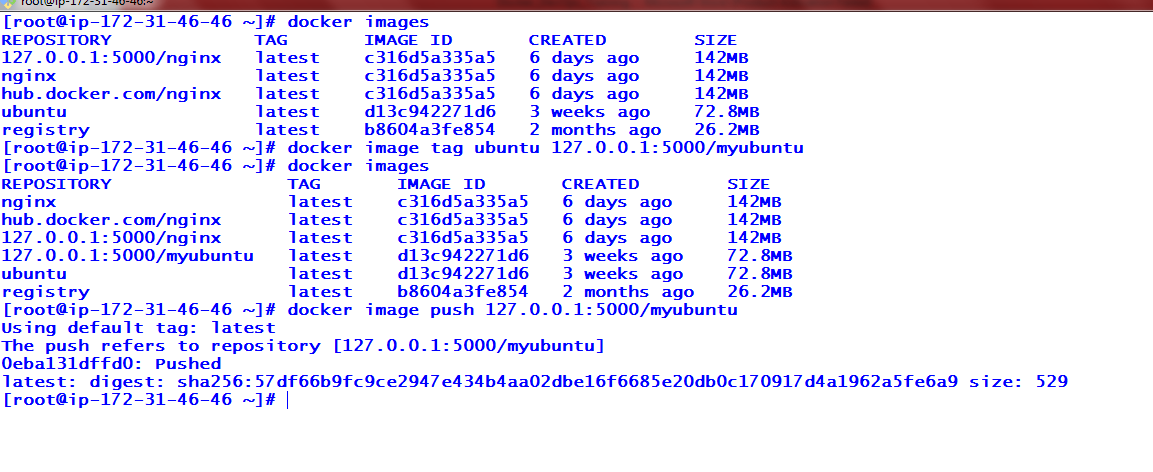
1. Create ec2 instance.
2. Set root password and make password authentication yes in /etc/ssh/sshd\_config file
3. Restart sshd service
4. Install docker package
5. Install elinks



Execute elinks command



Here we have uploaded image to local repo:



To use http we need to add below file in /etc/docker location

[root@ip-172-31-46-46 docker]# cat daemon.json

{

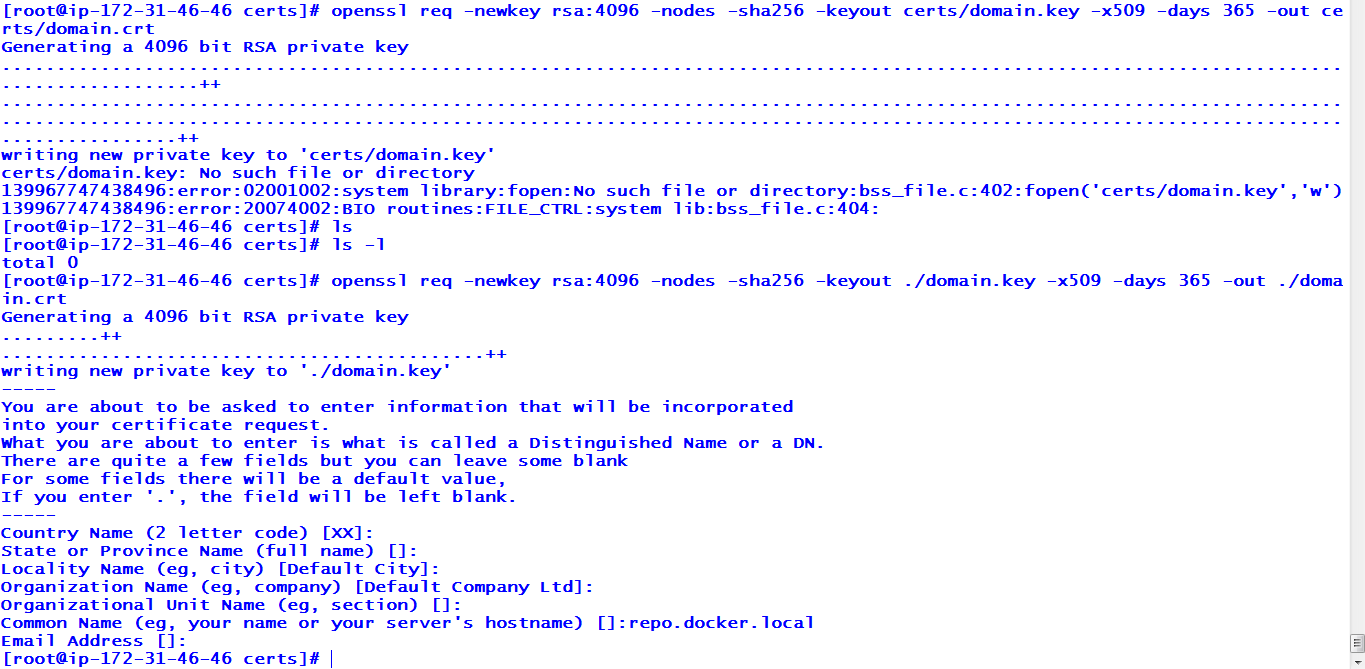
\*\*insecure-registries\* : [\*172.31.46.46:5000\*]

}

[root@ip-172-31-46-46 docker]#

Using https:

openssl req -newkey rsa:4096 -nodes -sha256 -keyout ./domain.key -x509 -days 365 -out ./domain.crt



[root@ip-172-31-46-46 certs]# cp domain.crt /etc/docker/certs.d/repo.docker.local\:5000/ca.crt

[root@ip-172-31-46-46 certs]# systemctl restart docker

[root@ip-172-31-46-46 certs]# docker container run -d -p 5000:5000 secure\_registry -v $(pwd)/certs:/certs -e REGISTRY\_HTTP\_TLS\_CERTIFICATE=/certs/domain.crt -e REGISTRY\_HTTP\_TLS\_KEY=/certs/domain.key registry

Unable to find image 'secure\_registry:latest' locally

docker: Error response from daemon: pull access denied for secure\_registry, repository does not exist or may require 'docker login': denied: requested access to the resource is denied.

**Dockerfile:**

USER – to switch user

WORKDIR – default working directory

RUN -

LABEL

COPY 🡪 copy and place to destination

ADD 🡪 it will extract on destination location/ it work as wget.. to download remote file inside image

Ex: ADD <https://www.free-css.com/assets/files/free-css-templates/download/page274/arizona.zip> /var/www/html

ENV 🡪 to set env variable

CMD 🡪 TO execute command on shell

Ex: CMD [“tree”]

Ex: CMD [“python”]

Note: Always execute last CMD and override it on previous command

EXPOSE 22

SSH to Container:

Create image using below package

RUN yum install openssh-client –y && yum install openssh-server

RUN mkdir –p /var/run/sshd

EXPOSE 22

CMD [“/usr/sbin/sshd”, “-D”]

#ssh user@container-ip



FROM centos:latest

MAINTAINER kishor aswar

RUN yum install –y httpd\

zip \

unzip

ADD <https://www.free-css.com/assets/files/free-css-templates/download/page274/arizona.zip> /var/www/html

WORKDIR /var/www/html

RUN unzip Arizona.zip

RUN cp –rvf Arizona/\* .

RUN rm –rf Arizona.zip

CMD [“/usr/sbin/httpd”, “-D”, “FOREGROUND”]

EXPOSE 80

CMD vs Entrypoint

CMD: override cmd instruction

Entrypoint : its not overriding

CMD is an instruction that is best to use if you need a default command which users can easily override. If a Dockerfile has multiple CMDs, it only applies the instructions from the last one.

On the other hand, **ENTRYPOINT** is preferred when you want to define a container with a specific executable. You cannot [override an ENTRYPOINT](https://phoenixnap.com/kb/docker-run-override-entrypoint) when starting a container unless you add the **--entrypoint** flag.

**Combine ENTRYPOINT with CMD** if you need a container with a specified executable and a default parameter that can be modified easily. For example, when [containerizing an application](https://phoenixnap.com/kb/how-to-containerize-applications) use **ENTRYPOINT** and **CMD** to set environment-specific variables.

Docker Compose:

[root@ip-172-31-46-46 Docker\_Practise]# docker container rm -f $(docker container ls -aq)

68e963035fdc

42ee6402d7be

3984e9752d70

fd079a45ce01

95e2b586242a

cbbc6b013dd3

d25f12cf1a0b

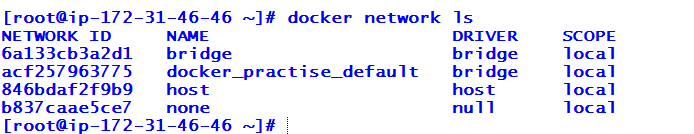
docker container run --name db-mysql -e MYSQL\_ROOT\_PASSWORD=mypassword -d mysql:5.7

docker container run -d -p 8000:80 --name my-wordpress -e WORDPRESS\_DB\_HOST=172.17.0.2:3306 -e WORDPRESS\_DB\_USER=root -e WORDPRESS\_DB\_PASSWORD=mypassword wordpress

DB\_HOST=ip 🡪its container ip

<https://docs.docker.com/samples/wordpress/>

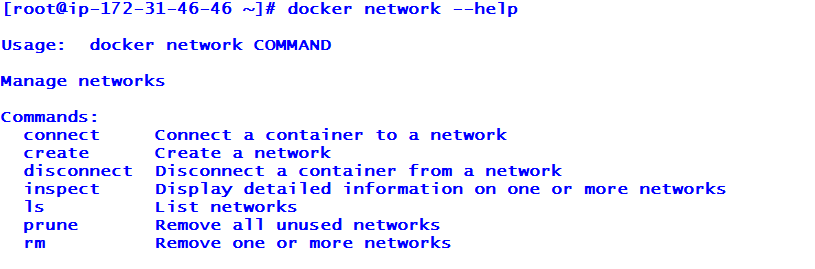
**Docker Network:**

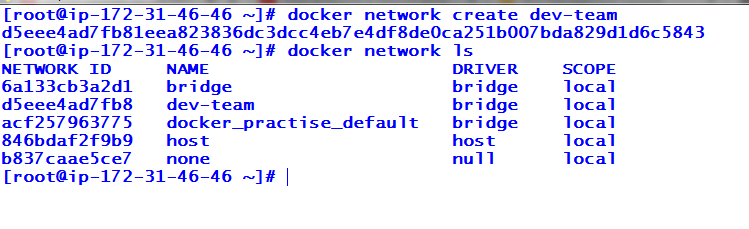


Here we are passing hostname and dns.



Create Docker Network

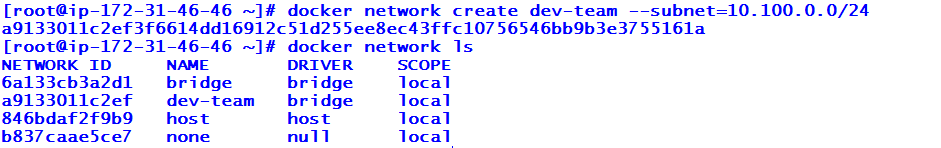


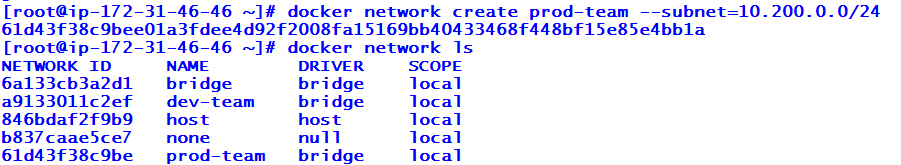


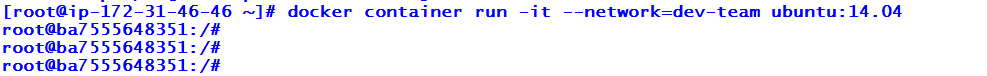
Remove network:



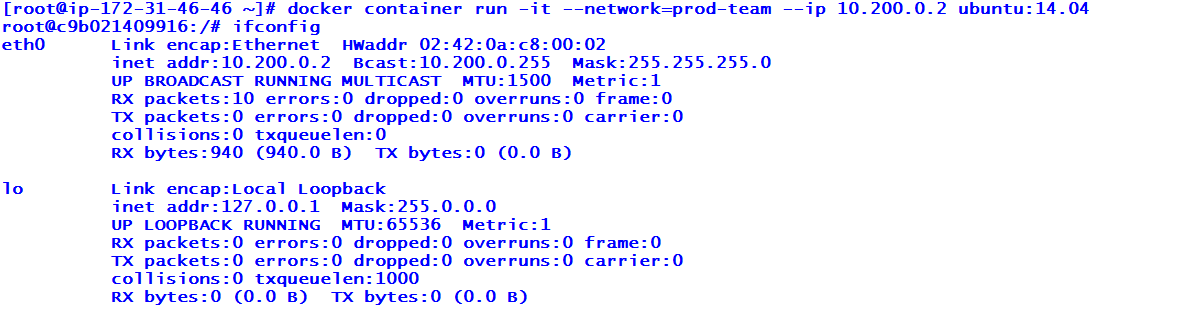
Create network with subnet



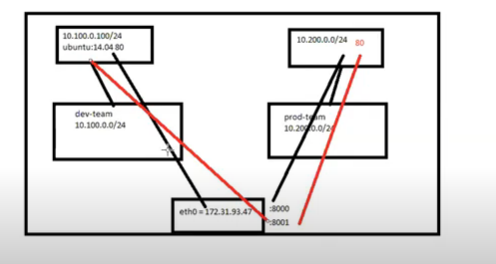


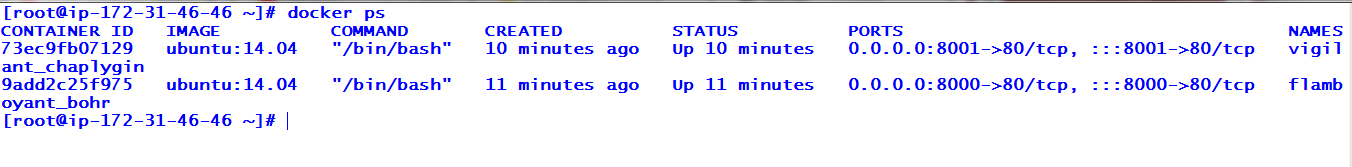


Assign specific ip to container



Communicate two different network containers:





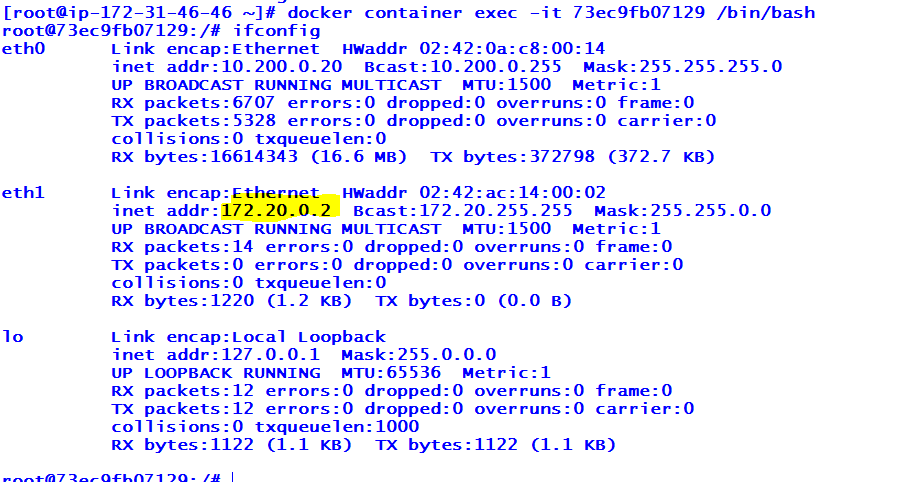




Without PORT mapping .. container can communicate

Create new network

docker network connect 82ede80eb0(network id) 73ec9fb07129(container id)



Disconnect From Network:

docker network disconnect 82ede80eb0(network id) 73ec9fb07129(container id)

**Docker Swarm:**

Its node cluster.

We can make all node as manager and worker.

