Docker Installation

For RHEL :

yum install docker -y

service docker status

service docker start

docker –version

systemctl enable docker

For Ubuntu :

apt-get update

apt-get install docker.io -y

service docker status

service docker start

docker –version

systemctl enable docker

Docker Container Commands:

CID = container ID or name

docker container ls or ps  
docker container ls -a or ps -a  
docker container rm CID  
docker container rm -f CID  
docker container start CID  
docker container stop CID  
docker container restart CID  
docker container rename CID <new\_name>  
docker container exec -it CID bash or u an provide sh as well instead of bash  
docker container logs CID  
#docker container attach CID  
docker container kill CID  
docker container prune => to delete stopped containers  
docker container ls -aq => to list all container id  
docker container inspect <container\_id or name>  
docker image inspect <image\_id>

docker container cp <file> <container\_id>@<path>

ex:

docker container cp index.html mynginxContainer:/usr/share/nginx/html/index.html

docker container run -d –name subodhContainer nginx => create container in detached mode with container name as subodhContainer

docker container stop `docker container ps -q` => to stop all running containers  
docker container rm `docker container ps -aq` => to delete all stopped containers

docker container export cid > a.tar  
docker image import a.tar myImage  
docker run -dit myImage bash

docker image tag myImage subodhdere77/imageName:v1  
docker image push subodhdere77/imageName:v1

================================================================

Port Mapping:

Check free port using

netstat -anp | grep <port>

Ex:

docker container -d -p <host\_port>:<container\_port> nginx

docker container run -d -p 1234:80 nginx

How to access from browser:

http://<server>:<port>

http://localhost:1234

================================================================

Env Variable:

docker container run -d -p 3456:8080 -e APP\_COLOR=red kodekloud/webapp-color

How to check env variables:

docker container exec <container\_id> **env**

================================================================

Dockerfile  
  
FROM ubuntu:16.04  
RUN apt-get update && apt-get install tree -y  
RUN apt-get update && apt-get install -y openssh-server && apt-get install -y python  
RUN touch /tmp/1.txt  
ENV PASS password  
ENV NAME subodh  
RUN pwd > /tmp/5.txt  
RUN cd /tmp  
RUN pwd > /tmp/6.txt  
WORKDIR /tmp  
RUN pwd > /tmp/7.txt  
RUN whoami > /tmp/8.txt   
RUN useradd -d /home/subodh -g root -G sudo -m -p $(echo "$PASS" | openssl -1 stdin) $NAME  
RUN whoami > /tmp/9.txt   
COPY dirName /tmp/  
ADD a.tar /tmp  
EXPOSE 88  
ENTRYPOINT [ "echo",”subodh” ]  
CMD ["dere"]

Docker Volume Commands:

docker volume create - Create a volume

docker volume inspect - Display detailed information on one or more volumes

docker volume ls - List volumes

docker volume prune - Remove all unused local volumes

docker volume rm - Remove one or more volumes

docker run –itd –v /tmp/test/:/usr/share/nginx/html nginx

Docker Network Commands:

docker network connect - Connect a container to a network

docker network create - Create a network

docker network disconnect - Disconnect a container from a network

docker network inspect - Display detailed information on one or more networks

docker network ls - List networks

docker network prune - Remove all unused networks

docker network rm - Remove one or more networks

docker run –itd –network=<new\_network\_name> nginx

Docker Compose Commands:

Installation:

sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

sudo chmod +x /usr/local/bin/docker-compose

sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose

File : docker-compose.yaml

version: "3"

services:

web:

build: .

ports:

- "5000:5000"

volumes:

- .:/code

- logvolume01:/var/log

links:

- redis

redis:

image: redis

volumes:

logvolume01: {}

=============================

docker-compose is a tool for defining and running multi-­con­tainer Docker applic­ations. With Compose, you use a YAML file to configure your applic­ation’s services.

Then, with a single command, you create and start all the services from your config­ura­tion.

Commands:

docker-compose build => only builds the images, does not start the containers

docker-compose up =>

docker-compose up => use docker-compose.yaml

docker-compose -f <file\_name\_local.yaml> up => use custom yaml files

-d, --detach => background detached mode

--build => forcefully Build images before starting contai­ners

--no-build => skips the image build process

--forc­e-r­ecreate => Recreate containers even if their config­uration and image haven’t changed.

--scale SERVIC­E=NUM => Scale SERVICE to NUM instances. Overrides the scale setting in the Compose file if present.

docker-compose down =>

Stops containers and removes containers, networks, volumes, and images

By default, the only things removed are:

- Containers for services defined in the Compose file

- Networks defined in the networks section of the Compose file

- The default network, if one is used

created by up. Networks and volumes defined as external are never removed.

use -v to remove volumes also along with other things

docker-compose stop => Stop an existing service container

docker-compose start => Start an existing service container

docker-compose restart => Restart an existing service container

docker-compose pause => Pauses running containers of a service

docker-compose unpause => Unpauses paused containers of a service

docker-compose ps => Shows list of containers for a service

docker-compose logs => Displays log output from services

docker-compose top => View the processes running within each service container

docker-compose pull => Pulls an image associated with a service defined in a docker-compose.yaml file, but does not start containers based on those images.

docker-compose rm =>

Removes stopped service contai­ners.By default, anonymous volumes attached to containers are not removed. You can override this with -v. To list all volumes, use docker volume ls.

-f, --force => Don’t ask to confirm the removal

-s, --stop => Stop the contai­ners, if required, before removing

-v => Remove any anonymous volumes attached to containers

docker-compose version => Prints the version of docker­compose

docker-compose push => Pushes images for services to their respective regist­ry/repository

docker­compose config => Validate and view the Compose file

docker­compose kill => Forces running containers to stop by sending a SIGKILL signal