#List local Images
docker images

#Log in to a remote registry
docker login RegistryURL -u user -p password

#Pull an image from a remote registry
docker pull registry.redhat.io/rhel8/httpd-24

#Run a container with Podman
docker run -d -it -p 80:80 -v /nfs:/var/www/html registry.redhat.io/rhel8/httpd-24

#List the running containers
docker ps

#Execute a command in a running container
docker exec -it containerid /bin/bash

#Execute a command by root in a running container
docker exec -u 0 -it a34b5708919d /bin/bash

#Display the logs of a container
docker logs containerid

#Save an image
docker save -o /path containerid

#Load an image
docker load -i /path

#Start an existing container
docker start containerid

#Stop an existing container
docker stop containerid

#Restart an existing container
docker restart containerid

#Remove a container
docker rm containerid

#Remove a container image
docker rmi containerimages

#Podman version
docker version

#Log out
docker logout

#Create a new image based on the current state of a running container
docker run -rm -it [–name name] image:tag command

#Create a new image based on the current state of a running container
docker commit container newImage:tag

#Create (but don’t start) a container from an image
docker create [-name name] image:tag

#Restart an existing container
docker restart container

#Wait on one or more containers to stop
docker wait container1 [container2…]

#Stop a running container gracefully
docker stop container

#Send a signal to a running container
docker kill container

#Remove a container (use -f if the container is running)
docker rm [-f] container

#Display a live stream of a container’s resource usage
docker stats container

#Return metadata (in JSON) about a running container
docker inspect container

#Processors running on Docker
docker top container-id

#Delete volume forcefully
docker volume prune

#Delete container forcefully
docker container prune

#Build docker images
docker build -t yourusername/repository-name .

#Specify the docker file if having other name
docker build -f dockerfile.dev .

############### Docker File ####################

```
FROM ubuntu:18.04    #Add base image
LABEL name vaibhav  #Add name labels to container
LABEL email vvgadhave@gmail.com    #Add email label
ENV NAME vaibhav    #Add enviroment variables
ENV PASS Passwd@123#    #Add enviroment variables
RUN pwd > /home/test.txt  # (/) Run command
WORKDIR /tmp   # Change working directory, default may be /
RUN pwd > /home/test.txt  #(/tmp) output after changing working directory
USER vaibhavg  # Switch user from root to vaibhavg
COPY test.txt /tmp/  #Copy local test.txt file from local to container /tmp/
ADD test.txt.tar /tmp  #Extract contents from test.txt.tar in /tmp
CMD ["python"]  #Run any specific command in container
CMD ["/bin/bash"]  # If having multiple CMD in docker file then last CMD will execute when
container is up
ENTRYPOINT ["test.sh"] # Executes as soon as the container is up
```

##################Docker compose file##################

```
version: '2.2'
services:
  node01:
    image: docker.elastic.co/elasticsearch/elasticsearch:7.11.1
    container_name: node01
    environment:
      - node.name=node01
      - cluster.name=es-cluster-7
      - discovery.type=single-node
      - "ES_JAVA_OPTS=-Xms128m -Xmx128m"
    ulimits:
      memlock:
        soft: -1
        hard: -1
    volumes:
      - es-data01:/usr/share/elasticsearch/data
    ports:
      - 9200:9200
    networks:
      - es-network

  kibana:
    image: docker.elastic.co/kibana/kibana:7.11.1
    environment:
      ELASTICSEARCH_HOSTS: http://node01:9200
    ports:
      - 5601:5601
    networks:
      - es-network
    depends_on:
      - node01
```

```
    restart: always     #(no = never restart | always = if container stops for nay reason | on-failure =
only restart if stops by any error code | unless-stopped = always restart wnless we forcefully stop it )

  heartbeat:
    image: docker.elastic.co/beats/heartbeat:7.11.1
    environment:
      ELASTICSEARCH_HOSTS: http://node01:9200
    volumes:
      - /home/vaibhavg/kibana/heartbeat.yml:/usr/share/heartbeat/heartbeat.yml:ro
    networks:
      - es-network
    depends_on:
      - node01


volumes:
  es-data01:
    driver: local

networks:
  es-network:
    driver: bridge
#################### JFYI ######################

# Install ssh server in conatiner
RUN apt install openssh-server -y
EXPOSE 22
CMD ["/usr/sbin/sshd", "-D"]

############### Docker-Machine ####################

#URL for deocker-machine
https://docker-docs.netlify.app/machine/install-machine/

#Check Version
docker-machine version

#List docker machines
docker-machine ls

#Create docker machines
docker-machine create --driver virtualbox machine1

#You need to tell Docker to talk to the new machine. You can do this with the docker-machine env
command.
docker-machine env machine1

#Connect your shell to the new machine.
eval "$(docker-machine env machine1)"

#Start a Docker machine
docker-machine start machine1
```

#Stop a Docker machine
docker-machine stop machine1

#Restart a Docker machine
docker-machine restart machine1

#Configuration for a Docker machine
docker-machine config machine1

#Inspect a Docker machine
docker-machine inspect machine1

#Get ip of a Docker machine
docker-machine ip machine1

#Kill a Docker machine
docker-machine kill machine1

#1.Set the hostname on the instance to the name Machine addresses it by, such as default.
2. Install Docker if it is not present already.
3.Generate a set of certificates (usually with the default, self-signed CA) and configure the daemon to accept connections over TLS.
4. Copy the generated certificates to the server and local config directory.
5. Configure the Docker Engine according to the options specified at create time.
6. Configure and activate Swarm if applicable.
--> docker-machine provision machine1
#regenerated certificates for a Docker machine
docker-machine regenerate-certs machine1

#ssh a Docker machine
docker-machine ssh machine1

#Status a Docker machine
docker-machine status machine1

#upgrade a Docker machine
docker-machine upgrade machine1

#Stop a Docker machine
docker-machine url machine1

############## Docker Swarm ####################

#To create a new docker swarm leader
docker swarm init

#To chek the list of nodes connected to docker swarm
docker node ls

#To get the docker initialise link again to join worker
docker swarm join-token worker

#To get the docker initialise link again to join mananger
docker swarm join-token manager

#To exit from docker swarm manager (Run this comand on worker)
docker swarm leave

#To remove any worker from group (Run this command on manager)
docker node rm worker2

#To inspect worker information
docker node inspect worker1

#Docker always recommends to promote manager as 1,3,5,7.. (as odd no)

#To promote docker worker to manager
docker node promote worker1 worker2

#To demote docker worker to manager
docker node demote worker1 worker2

#To create a service
docker service create -d alpine ping 8.8.8.8

#To create a service with replicas
docker service create -d --replicas 4 alpine ping 8.8.8.8

#To check the running serices.
docker service ls

#To check where the serices are running
docker service ps ajdlkjdlii211

#To scale docker conatainers
docker service scale asdekjjkajdf=4 dasdewdqw=4

#To run the serice globally
docker service create --mode global alpine ping 8.8.8.8

#To run container on manager only
docker service create --constraint="node.role==manager" alpine ping 8.8.8.8

#To run container on any particular worker only
docker service create --constraint="node.role==worker" alpine ping 8.8.8.8

#To add any label to any worker (Labels are required because you can run containers on any particular workers by this)
docker node update --label-add="ssd=true" worker1

#TO remove any label from worker
docker node update --label-rm ssd node2

#To run any service on particular worker by using labels
docker service create --constraint="node.labels.ssd==true" --replicas=3 -d alpine ping 8.8.8.8

#To pause any running worker node
docker node update --availability=pause worker2

#To unpause any running worker node
docker node update --availability=active worker2

#To move all containers from running worker node to other automatically
docker node update --availability=drain worker2