**>systemctl start/stop/status docker🡪 start/stop/status docker services**

**>docker info🡪display the docker system details**

**>docker images –a 🡪 show all docker images in local repo.**

**>docker ps 🡪display running containers, docker ps –a🡪display stopped containers**

**>docker search ubuntu 🡪search images from docker repository**

**>docker pull ubuntu 🡪pull images from repo**

**>docker run –tid -–name=test –net <network name> -v /test:/var/log –p 80:80 –-ip 172.9.0.0 –h web1.example.com ubuntu:16.04 /bin/bash🡪create a container**

**>docker attach [container name/ID]🡪 to attach to the detached container**

**>docker inspect [container name/ID]🡪 information of the running docker container**

**>docker logs [container name/ID]🡪display the logs present at time of execution.**

**>docker start/stop [container name/ID] 🡪start/stop a container**

**>docker inspect -f '{{ .NetworkSettings.IPAddress }}' <container\_name> 🡪 Check container's IP Address**

**>docker rename test test1 🡪rename a container**

**>docker pause/unpause [container name/ID] 🡪Pause/unpause all running processed inside a container.**

**>docker top [container name/ID]🡪display running processes of a container**

**>docker stats [container name/ID]🡪 Display a live stream of containers**

**>docker history ubuntu 🡪show the history/containers of the image**

**>docker kill [container name/ID] 🡪kill a container**

**>docker kill $(docker ps –q) 🡪Kill all containers**

**>docker rm $(docker ps –a –q) 🡪 delete all stopped containers**

**>docker rmi $(docker images –q –f dangling=true) 🡪 Delete all none images**

**>docker rmi $(docker images –q) 🡪delete all images**

**>docker rmi –f ‘docker images’ 🡪forcefully remove images from local repo.**

**>docker ps –aq | xargs docker rm –f 🡪delete all containers running and non running**

**>docker diff [container name/ID]🡪 Inspect changes to files or directories on a container’s filesystem**

**>docker cp [file path] [(container name/ID):path]🡪 Copy files/folders between a container and the local filesystem.**

**> docker ps --format="{{.ID}} {{.Names}}"🡪to view all the container name and ID**

**>docker network ls🡪 list all the available docker networks**

**Type of networks:🡪Bridge,host and none**

**Bridge - The bridge network is provided by default and this is the network to which a container created on this host connects. This maps the “docker0” bridge on the host.**

**Host - When a container connects to the Host network , it gets the same configuration of the host. We can start a container with the host network and then inspect for details.**

**None - This is used to attach a container to a no network.Containers attached to this network will not have a IP address and will be stand-alone. The none network is useful to create containers that need to be standalone batch jobs without any dependency on other networks.**

**> docker network inspect bridge🡪to check the bridge network interface**

**>docker network create –driver <driver name> [network name] –subnet=172.19.0.0/16**

**>docker exec -it 2031be9aeaa6 ifconfig**

**> docker run -dit --net=none ifconfig-ubuntu /bin/bash🡪 Start a Container with the None network. Now if we run the ifconfig command inside the container we can see no IP address,>docker exec -it ab10aef79d5b ifconfig**

**>docker network connect/disconnect <Network Name> <Container> 🡪Connect or disconnect the container to the network specified**

**>docker network rm <network name> 🡪remove the docker network**

**> docker exec -it 7f9993262794 ping -c 4 b624040483f4🡪 pinged the second container from the first container**

**>docker volume ls🡪to list all the volumes**

**>docker volume create <volume name>**

**docker run -it --volumes-from 0ff56994a111 ubuntu bash**

**>docker build –t test1 . 🡪 create an image from docker file**

**>docker commit [container name/ID] milan211/[image name]🡪create image from existing container**

**>docker push milan211/[image name]🡪push image to docker repo**

**> docker-compose up –d🡪docker-compose run in background after creating docker-compose.yaml**

**> docker-compose up --force-recreate🡪docker-compose force recreate**

**>docker swarm init –listen-addr 172.31.91.142:2377🡪**

**>docker node ls🡪to view all the nodes in the cluster**

**>docker node promote/demote [node-name/ID]🡪To add/remove node in managers list should execute on master Leader node only.**

**>docker node rm [node-name/ID]🡪remove nodes from swarm manager should execute on master Leader node only.**

**>docker node inspect self🡪node information as JSON**

**>docker info**

**>docker service create –p 8080:8080 --name <service name> --replicas 10 tomcat:8.0🡪 to download the image and run the container containers at a time in all the nodes**

* **--replicas are how many containers we want run the nodes simultaneously**

**>docker ps**

**>docker service scale <service name>=3🡪scaling up and down**

**Or docker service update –-replicas=5 <service name>**

**>docker service update –-image ngnix1.14.0 <service name>🡪update an service**

**>docker service ls🡪 to view number of services running in you managers nodes**

**>docker service ps <service name>**

**>docker service update –-rollback <service name>🡪roll back to previous version of a service. --update-delay 0s to execute the rollback without a delay between tasks**

**>shutdown –h now🡪to bring down a node**

**Docker File**

|  |  |
| --- | --- |
| **ADD** | **Copies a file from the host system onto the container** |
| **CMD** | **The command that runs when the container starts** |
| **ENTRYPOINT** |  |
| **ENV** | **Sets an environment variable in the new container** |
| **EXPOSE** | **Opens a port for linked containers** |
| **FROM** | **The base image to use in the build. This is mandatory and must be the first command in the file.** |
| **MAINTAINER** | **An optional value for the maintainer of the script** |
| **ONBUILD** | **A command that is triggered when the image in the Dcokerfile is used as a base for another image** |
| **RUN** | **Executes a command and save the result as a new layer** |
| **USER** | **Sets the default user within the container** |
| **VOLUME** | **Creates a shared volume that can be shared among containers or by the host machine** |
| **WORKDIR** | **Set the default working directory for the container** |

**FROM ubuntu:15.04**

**MAINTAINER Ivan Krizsan, https://github.com/krizsan**

**RUN apt-get update && \**

**apt-get upgrade -y && \**

**apt-get install -y software-properties-common && \**

**add-apt-repository ppa:webupd8team/java -y && \**

**apt-get update && \**

**echo oracle-java7-installer shared/accepted-oracle-license-v1-1 select true | /usr/bin/debconf-set-selections && \**

**apt-get install -y oracle-java8-installer && \**

**apt-get clean**

**nagios- 5666**