**Docker**

docker system prune

**List all images**

docker image ls

**To stop all Containers**

docker kill $(docker ps -q)

docker rm $(docker ps -a -q)

docker rmi $(docker images -q)

**list all container**

docker ps -aq

docker stop ####

docker rm $(docker ps -a -q)

docker rmi [image] << remove an image

**List all container (running, all, in quite mode)**

docker container ls

docker container ls -all

docker container ls -aq

docker built -t hello-world . // create image using dockerfile

docker run -p 400:80 helloworld // run container map port 400 to 80

docker run -d -p 400:80 helloworld // run container map port 400 to 80 in detach mode

**Stop all running container**

docker stop $(docker ps -aq)

**Remove all container**

docker rm $(docker ps -a -q)

**Remove all images**

docker rmi $(docker images -q)

docker pull [image]

docker build -t[ag]

docker start [container]

docker stop [container]

**Network**

docker network ls

remove network

docker network rm [network]

docker network inspect [network] << show information of the network

**Developing with Docker Containers:**

* **docker create [image]**: Create a new container from a particular image.
* **docker login**: Log into the Docker Hub repository.
* **docker pull [image]**: Pull an image from the [Docker Hub repository](https://hub.docker.com/).
* **docker push [username/image]**: Push an image to the Docker Hub repository.
* **docker search [term]**: Search the Docker Hub repository for a particular term.
* **docker tag [source] [target]**: Create a target tag or alias that refers to a source image.

**Running Docker Containers**

* **docker start [container]**: Start a particular container.
* **docker stop [container]**: Stop a particular container.
* **docker exec -ti [container] [command]**: Run a shell command inside a particular container.
* **docker run -ti—image [image] [container] [command]**: Create and start a container at the same time, and then run a command inside it.
* **docker run -ti—rm—image [image] [container] [command]**: Create and start a container at the same time, run a command inside it, and then remove the container after executing the command.
* docker pause [container]: Pause all processes running within a particular container.

**Using Docker Utilities:**

* **docker history [image]**: Display the history of a particular image.
* **docker images**: List all of the images that are currently stored on the system.
* **docker inspect [object]:** Display low-level information about a particular Docker object.
* **docker ps**: List all of the containers that are currently running.
* **docker version**: Display the version of Docker that is currently installed on the system.

**Cleaning Up Your Docker Environment:**

* **docker kill [container]**: Kill a particular container.
* **docker kill $(docker ps -q)**: Kill all containers that are currently running.
* **docker rm [container]**: Delete a particular container that is not currently running.
* **docker rm $(docker ps -a -q)**: Delete all containers that are not currently running.

**Start a container in background**

docker run -d jenkins

**Start an interactive container**

docker run -it ubuntu bash

**Start a container automatically removed on stop**

docker run --rm ubuntu bash

**Export port from a container**

docker run -p 80:80 -d nginx

**Start a named container**

docker run --name mydb redis

**Restart a stopped container**

docker start mydb

**Stop a container**

docker stop mydb

**Add metadata to container**

docker run -d \ label=traefik.backend=jenkins jenkins

**To show all instances of these images (called containers), That will list all containers, including stopped ones:**

docker ps

docker ps -a

**List your containers (-a shows stopped ones too):**

docker container ls -a (docker ps is deprecated)

**List your images:**

docker images

**It is useful to log into your containers sometimes. To do that, use this:**

docker exec -ti <container ID or name> sh

**Check the logs from your containers:**

docker logs <container ID or name>

**Inspect your container(s):**

docker inspect <container ID or name>

**View and manage volumes,**

docker volume ls

docker volume inspect <name>

**View and manage networks,**

docker network ls

docker network inspect <name>

**Remove stopped containers:**

docker rm $(docker ps -a q)

**Remove all unused containers, volumes, networks and images (both dangling and unreferenced):**

docker system prune (add -a to remomve unnused images, not just dangling)

**Delete the entire vm, and run compose again**

docker-machine rm default  
docker-machine create --driver virtualbox default  
eval (docker-machine env default)  
docker-compose up

**Remove all stopped containers:**

docker container rm $(docker container la -a -q)

**Remove all dangling images:**

docker image rm $(docker images -f dangling=true)

**Remove all unused containers, volumes, networks and dangling images (add -a to remove any unreferenced images as well):**

docker system prune

**# Delete all containers**

docker rm $(docker ps -a -q)

**# Delete all images**

docker rmi $(docker images -q)

**docker container run --detach --publish 80:80 nginx:alpine**

Running an Interactive Ubuntu Container

**docker container run --interactive --tty ubuntu:16.04**

**Docker Compose**

docker-compose up -d

docker-compose down

**A Dozen Dockerfile Instructions**

FROM—specifies the base (parent) image.

LABEL —provides metadata. Good place to include maintainer info.

ENV—sets a persistent environment variable.

RUN —runs a command and creates an image layer. Used to install packages into containers.

COPY—copies files and directories to the container.

ADD—copies files and directories to the container. Can upack local .tar files.

CMD—provides a command and arguments for an executing container. Parameters can be overridden. There can be only one CMD.

WORKDIR—sets the working directory for the instructions that follow.

ARG—defines a variable to pass to Docker at build-time.

ENTRYPOINT—provides command and arguments for an executing container. Arguments persist.

EXPOSE—exposes a port.

VOLUME—creates a directory mount point to access and store persistent data.