Images are used to package and distribute applications, while containers are the runtime instances of those images.

* Docker run <image-name> 🡪 to run an image
* Docker run –name <container name> <image-name> 🡪 to create and name the container.
* Docker pull <image-name> 🡪 to pull any image from the official Docker hub
* Docker ps 🡪 to list all the running containers
* Docker ps –a 🡪 to list all the running and stopped containers
* Docker ps –l 🡪 to list the latest container
* Docker ps -q 🡪 shows the id of the container
* Docker stop <container ID/ name> 🡪 to stop the container
* Docker start <container ID/name> 🡪 to start the container
* Docker rm {options} <container-id/container-name> 🡪 to remove the container

-f flag: remove the container forcefully.

-v flag: remove the volumes.

-l flag: remove the specific link mentioned.

* Docker rmi <image ID/ image name> 🡪to delete image in Docker.
* Docker exec {options} --This command allows us to run new commands in a running container.

-d flag: for running the commands in the background.

-i flag: it will keep STDIN open even when not attached.

-e flag: sets the environment variables

Ex: docker exec <cont-id> cat /etc/\*release\*

* Docker login 🡪login to docker hub
* Docker push <image name/ image-id> 🡪 to push inti hub
* Docker build -t <image-name: tag > 🡪 to build docker image
* Docker images -q | wc -l 🡪 no of images in local in number
* Docker rmi $(docker images -aq) 🡪to delete all the images at a time
* Docker images <image-name> - to see details of particular image
* Docker image <image-image.> --no-trunc 🡪 to display Docker Images, with full-length Image Ids, you use the –no-trunc flag.
* Docker run <image-name> sleep 10 🡪 to sleep the container in 10 sec after 10 sec the container automatically will stop
* docker attach <cont-id> 🡪 before attaching ensure the container is running
* sudo docker --filter=reference='Ubuntu' -- to filter the images

docker build –t <image name > . 🡪 to build an image

Q. Run a container named blue-app using image kodekloud/simple-webapp and set the environment variable APP\_COLOR to blue. Make the application available on port 38282 on the host. The application listens on port 8080.

docker run -p 38282:8080 --name blue-app -e APP\_COLOR -d kodekloud/simple-webapp

To know the env field from within a webapp container, run docker exec -it blue-app env

Q. Deploy a mysql database using the mysql image and name it mysql-db.

Set the database password to use db\_pass123. Lookup the mysql image on Docker Hub and identify the correct environment variable to use for setting the root password.

docker run -d -e MYSQL\_ROOT\_PASSWORD=db\_pass123 --name mysql-db mysql

docker image prune –a -🡪 to delete all the images.