What is Docker?

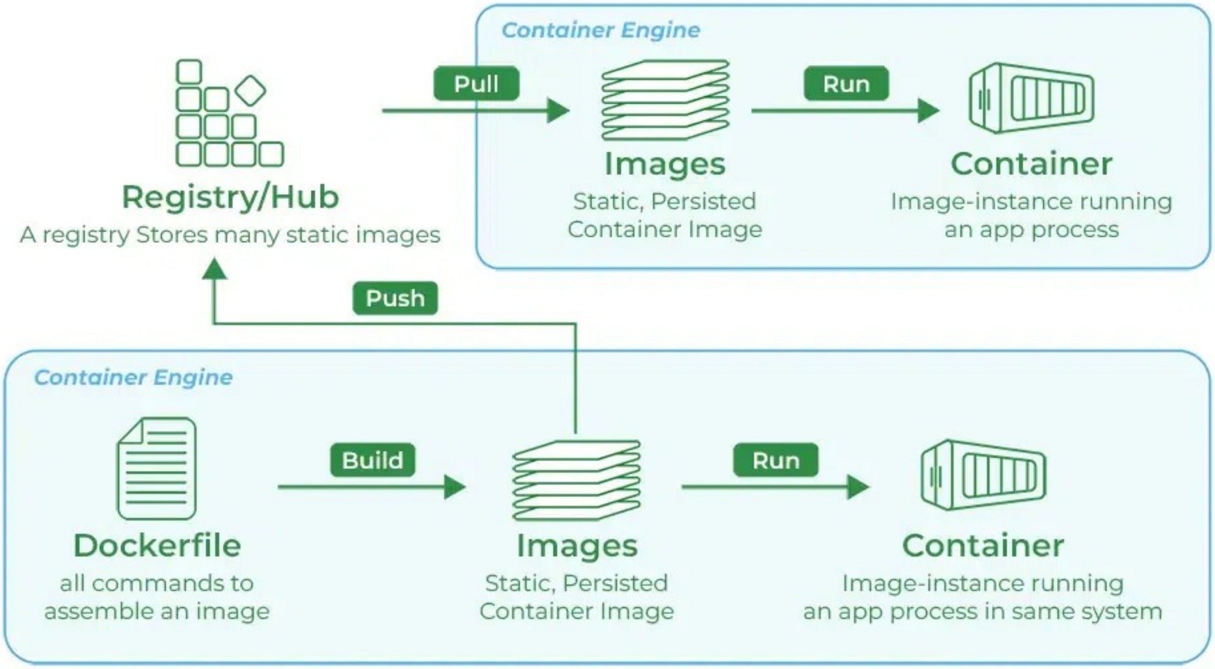
Docker is an open source container management software, it helps in managing the container life cycle such as creating, running, stopping, creating networks, volumes. It facilitates the developers to package their logical code with all its dependencies into a single executable bundle (Docker Image). Once the image has built, we can deploy it on any machine that supports docker acting as independent of the underlying OS.

What is Dockerhub?

Docker Hub is a repository service and it is a cloud-based service where people push their Docker Container Images and also pull the Docker Container Images from the **Docker Hub** anytime or anywhere via the internet. It provides features such as you can push your images as private or public.

Mainly DevOps team uses the Docker Hub. It is an open-source tool and freely available for all operating systems. It is like storage where we store the images and pull the images when it is required. When a person wants to push/pull images from the Docker Hub they must have a basic knowledge of Docker. Let us discuss the requirements of the Docker tool.

Docker is a tool nowadays enterprises adopting rapidly day by day. When a Developer team wants to share the project with all dependencies for testing then the developer can push their code on **Docker Hub** with all dependencies. Firstly create the **Images** and push the Image on Docker Hub. After that, the testing team will pull the same image from the Docker Hub eliminating the need for any type of file, software, or plugins for running the Image because the Developer team shares the image with all dependencies.



Why Should I use Docker hub?

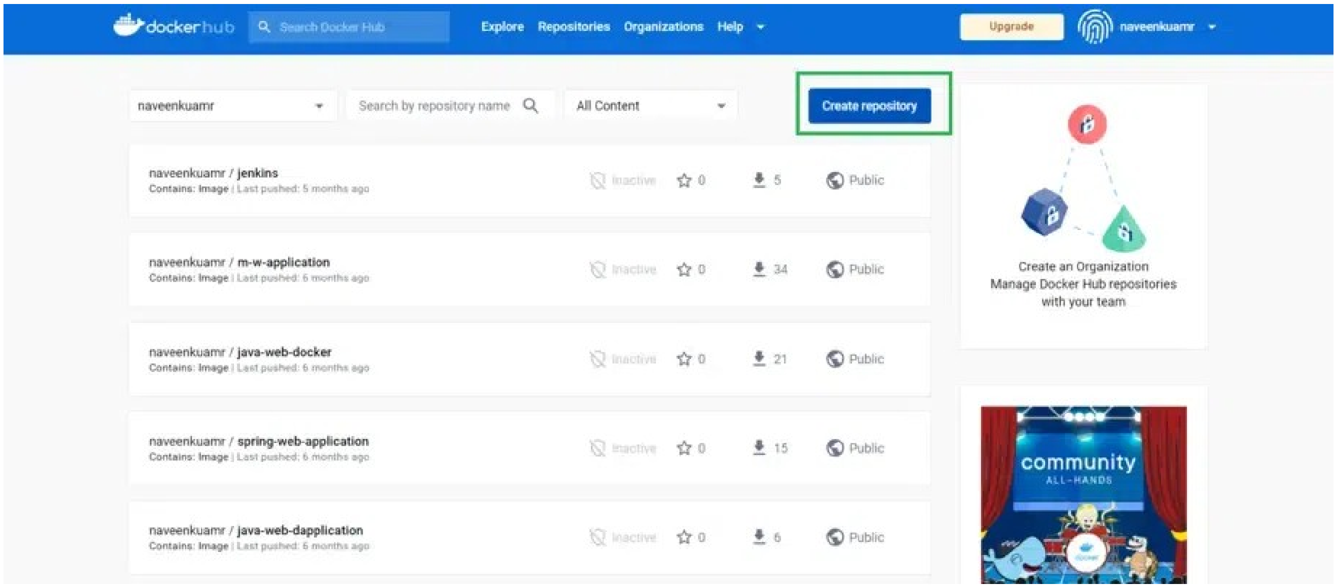
The following are some of the main aspects for using Dockerhub:

- **Efficient Image Management**: Docker Hub simplifies the storage, management, and sharing of Docker images, making it easy to organize and access container images from anywhere.
- **Enhanced Security**: It runs security checks on images and provides detailed reports on potential vulnerabilities, ensuring safer deployments.
- **Automation Capabilities**: With features like webhooks, Docker Hub can automate continuous deployment and testing processes, streamlining your CI/CD pipeline.
- **Integration and Collaboration**: Docker Hub integrates seamlessly with popular tools like GitHub and Jenkins, and allows managing permissions for users and teams, facilitating efficient collaboration.

How to Use Dockerhub and Create Repository? A Step-By-Step Guide

The following steps guide you in creating a first repository in Dockerhub using GUI:

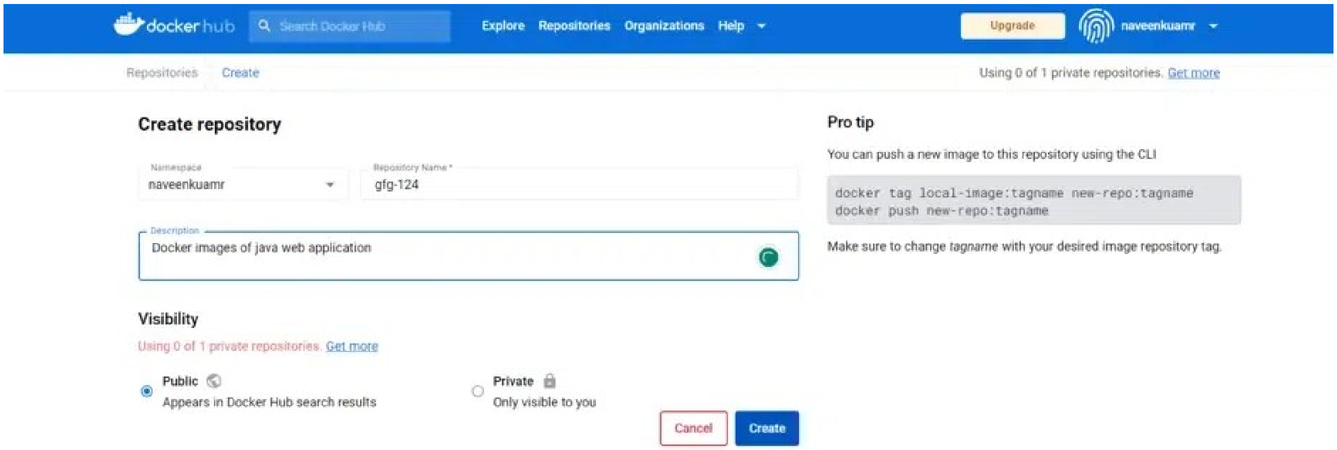
Step 1: Firstly navigate to the Dockerhub and sign in with your credentials and then select Create Repository.



naveenkuamr / **jenkins** Contains: Image | Last pushed: 5 months ago	Inactive	0	5	Public
naveenkuamr / **m-w-application** Contains: Image | Last pushed: 6 months ago	Inactive	0	34	Public
naveenkuamr / **java-web-docker** Contains: Image | Last pushed: 6 months ago	Inactive	0	21	Public
naveenkuamr / **spring-web-application** Contains: Image | Last pushed: 6 months ago	Inactive	0	15	Public
naveenkuamr / **java-web-dapplication** Contains: Image | Last pushed: 6 months ago	Inactive	0	6	Public

Step 2: After that, we will be taken to a screen for configuring the repository, where we must choose the namespace, repository name, and optional description.

- In the visibility area, as indicated in the picture, there are two options: Public and Private. We can choose any of them depending on the type of organization you are in.
- If you chose Public, everyone will be able to push-pull and use the image because it will be accessible to everyone. If you select the private option, only those with access to that image can view and utilize it. it.



Using 0 of 1 private repositories. Get more

Create repository

Namespace: naveenkuamr

Repository Name *: gfg-124

Description: Docker images of java web application

Visibility

Using 0 of 1 private repositories. Get more

● Public — Appears in Docker Hub search results

○ Private — Only visible to you

Pro tip

You can push a new image to this repository using the CLI

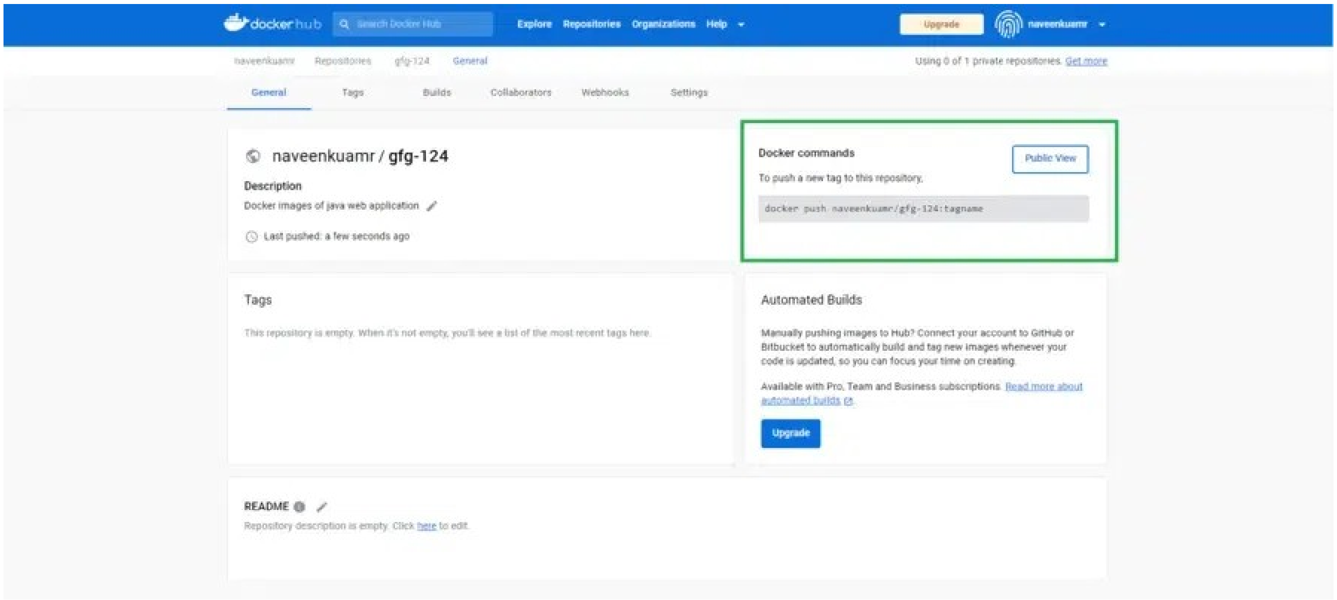
```
docker tag local-image:tagname new-repo:tagname
docker push new-repo:tagname
```

Make sure to change *tagname* with your desired image repository tag.

Step 3: At finally repository is created with the help of the Docker Commands we can push or pull the image.

- The following command is used for pushing the docker image that exists in local to the Dockerhub.

```
docker push <your-username>/my-testprivate-repo>.
```



Using 0 of 1 private repositories. Get more

naveenkuamr / gfg-124

Description

Docker images of java web application

Last pushed: a few seconds ago

Docker commands

To push a new tag to this repository,

```
docker push naveenkuamr/gfg-124:tagname
```

Tags

This repository is empty. When it's not empty, you'll see a list of the most recent tags here.

Automated Builds

Manually pushing images to Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating.

Available with Pro, Team and Business subscriptions. Read more about automated builds

README

Repository description is empty. Click here to edit.

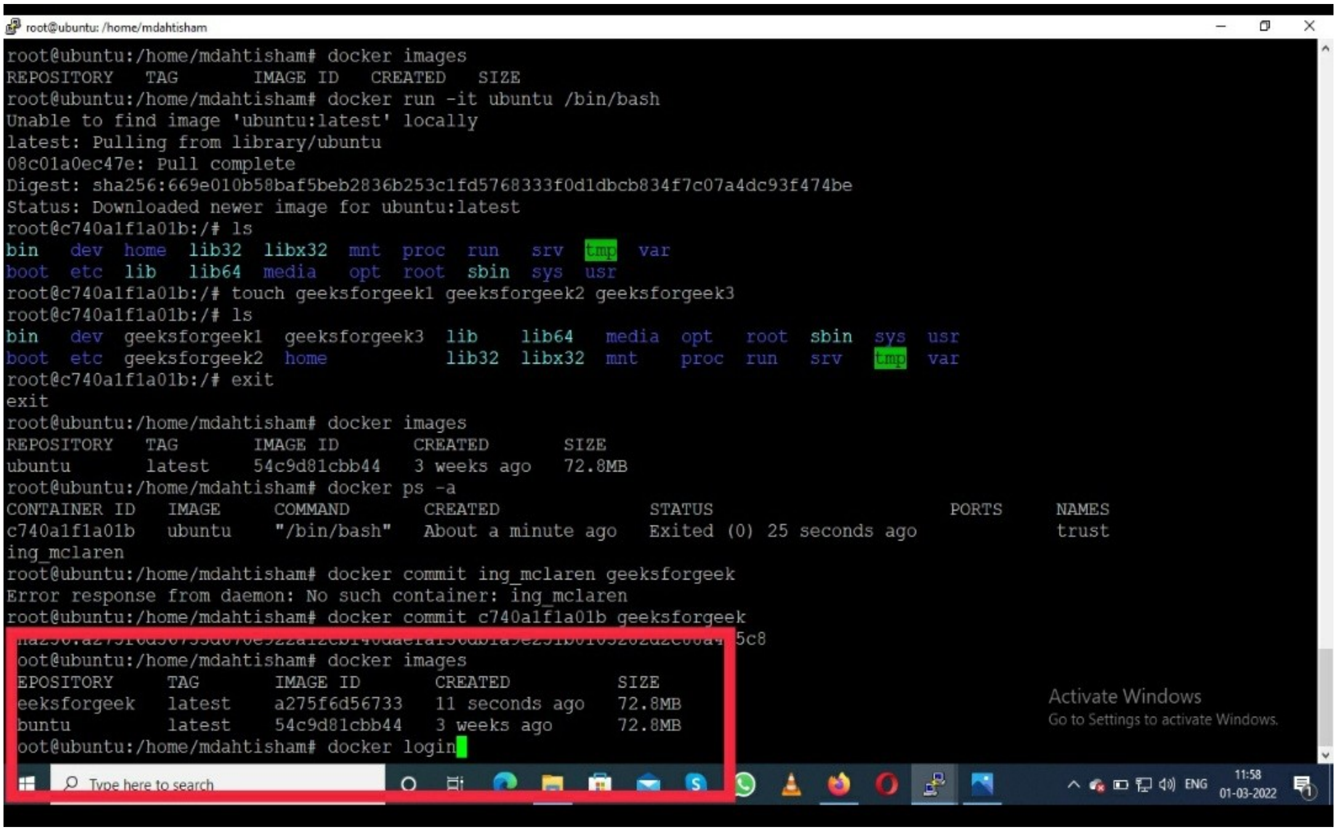
How To Push Docker Images to Docker Hub?

The push command as the name suggests itself is used to pushing a docker image onto the docker hub. Try to Follow this example to get an idea of the push command:

Step 1: Open Docker in your system.

- Locate the Images that you want to push using the below command:

```
docker images
```


root@ubuntu: /home/mdahtisham
Type here to search
Activate Windows
Go to Settings to activate Windows.
ENG
11:58
01-03-2022

```
root@ubuntu:/home/mdahtisham# docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
root@ubuntu:/home/mdahtisham# docker run -it ubuntu /bin/bash
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
08c01a0ec47e: Pull complete
Digest: sha256:669e010b58baf5beb2836b253c1fd5768333f0d1dbcb834f7c07a4dc93f474be
Status: Downloaded newer image for ubuntu:latest
root@c740a1f1a01b:/# ls
bin   dev  home  lib32  libx32  mnt    proc  run   srv  tmp  var
boot  etc  lib   lib64  media   opt    root  sbin  sys  usr
root@c740a1f1a01b:/# touch geeksforgeek1 geeksforgeek2 geeksforgeek3
root@c740a1f1a01b:/# ls
bin   dev  geeksforgeek1  geeksforgeek3  lib    lib64   media  opt   root  sbin  sys  usr
boot  etc  geeksforgeek2  home           lib32  libx32  mnt    proc  run   srv   tmp  var
root@c740a1f1a01b:/# exit
exit
root@ubuntu:/home/mdahtisham# docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
ubuntu       latest    54c9d81cbb44   3 weeks ago   72.8MB
root@ubuntu:/home/mdahtisham# docker ps -a
CONTAINER ID   IMAGE     COMMAND       CREATED              STATUS                      PORTS     NAMES
c740a1f1a01b   ubuntu    "/bin/bash"   About a minute ago   Exited (0) 25 seconds ago             trust
ing_mclaren
root@ubuntu:/home/mdahtisham# docker commit ing_mclaren geeksforgeek
Error response from daemon: No such container: ing_mclaren
root@ubuntu:/home/mdahtisham# docker commit c740a1f1a01b geeksforgeek
[illegible]5c8
oot@ubuntu:/home/mdahtisham# docker images
EPOSITORY     TAG       IMAGE ID       CREATED          SIZE
eeksforgeek   latest    a275f6d56733   11 seconds ago   72.8MB
buntu         latest    54c9d81cbb44   3 weeks ago      72.8MB
oot@ubuntu:/home/mdahtisham# docker login
```

- The above command will list all the images on your system.

Step 2: Go to the browser and search *hub.docker.com.*

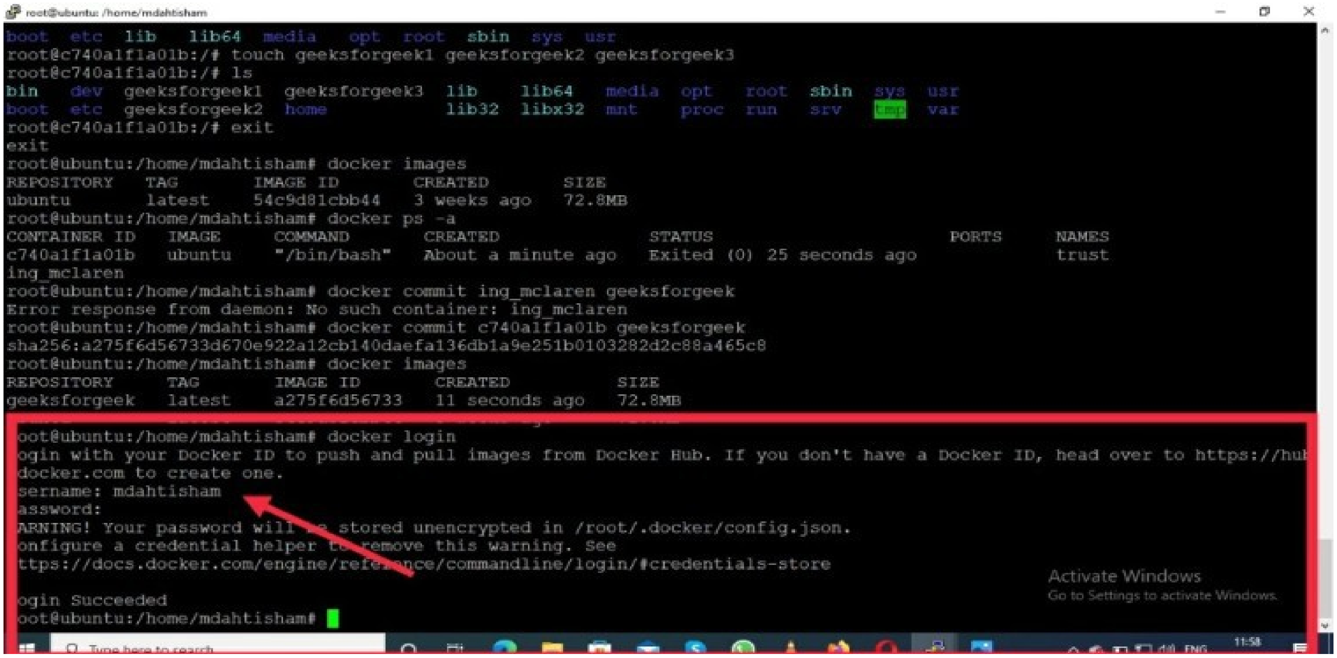
Step 3: Sign up on the docker hub if you do not have a docker hub account, after login on to docker hub.

Step 4: Back to the docker terminal and execute the below command:

```
docker login
```

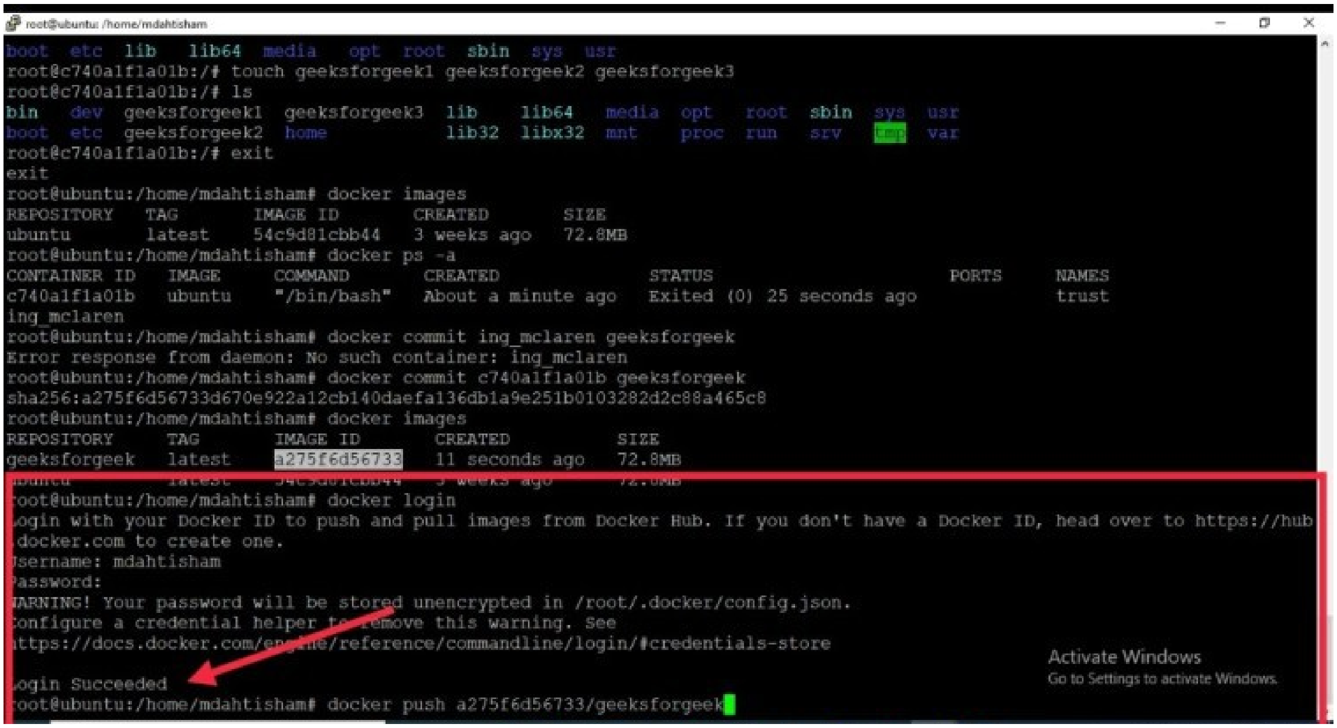
Step 5: Then give your credential and type in your docker hub username or password.

- username
- password



```
root@ubuntu: /home/mdahtisham
boot  etc  lib   lib64  media   opt  root  sbin  sys  usr
root@c740a1f1a01b:/# touch geeksforgeek1 geeksforgeek2 geeksforgeek3
root@c740a1f1a01b:/# ls
bin   dev  geeksforgeek1  geeksforgeek3  lib    lib64   media  opt   root  sbin  sys  usr
boot  etc  geeksforgeek2  home           lib32  libx32  mnt    proc  run   srv   tmp  var
root@c740a1f1a01b:/# exit
exit
root@ubuntu:/home/mdahtisham# docker images
REPOSITORY   TAG       IMAGE ID       CREATED        SIZE
ubuntu       latest    54c9d81cbb44   3 weeks ago    72.8MB
root@ubuntu:/home/mdahtisham# docker ps -a
CONTAINER ID   IMAGE     COMMAND       CREATED              STATUS                      PORTS     NAMES
c740a1f1a01b   ubuntu    "/bin/bash"   About a minute ago   Exited (0) 25 seconds ago             trust
ing_mclaren
root@ubuntu:/home/mdahtisham# docker commit ing_mclaren geeksforgeek
Error response from daemon: No such container: ing_mclaren
root@ubuntu:/home/mdahtisham# docker commit c740a1f1a01b geeksforgeek
sha256:a275f6d56733d670e922a12cb140daefa136db1a9e251b0103282d2c88a465c8
root@ubuntu:/home/mdahtisham# docker images
REPOSITORY     TAG       IMAGE ID       CREATED          SIZE
geeksforgeek   latest    a275f6d56733   11 seconds ago   72.8MB
oot@ubuntu:/home/mdahtisham# docker login
ogin with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub
docker.com to create one.
sername: mdahtisham
assword:
ARNING! Your password will be stored unencrypted in /root/.docker/config.json.
onfigure a credential helper to remove this warning. See
ttps://docs.docker.com/engine/reference/commandline/login/#credentials-store

ogin Succeeded
oot@ubuntu:/home/mdahtisham# 
```

Step 6: After that hit the Enter key you will see login success on your screen.



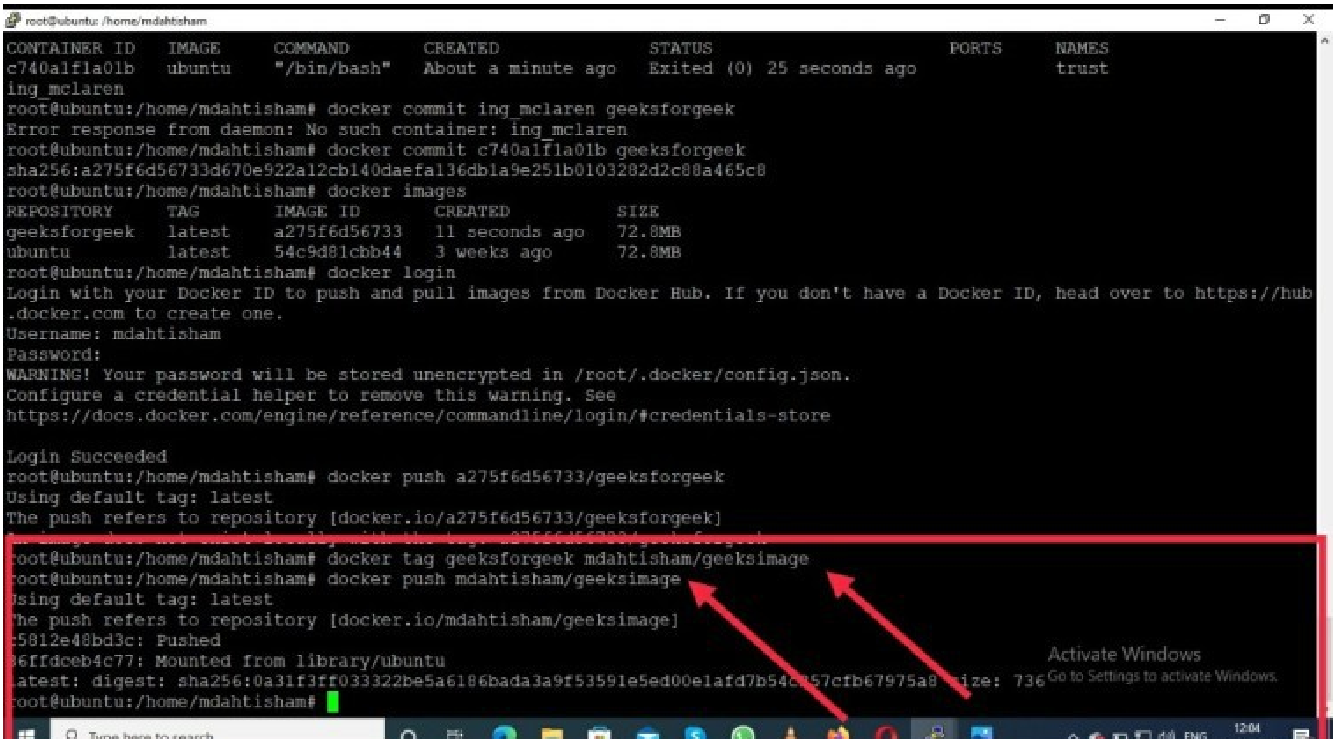
```
root@ubuntu: /home/mdahtisham
boot  etc  lib   lib64  media   opt  root  sbin  sys  usr
root@c740a1f1a01b:/# touch geeksforgeek1 geeksforgeek2 geeksforgeek3
root@c740a1f1a01b:/# ls
bin   dev  geeksforgeek1  geeksforgeek3  lib    lib64   media  opt   root  sbin  sys  usr
boot  etc  geeksforgeek2  home           lib32  libx32  mnt    proc  run   srv   tmp  var
root@c740a1f1a01b:/# exit
exit
root@ubuntu:/home/mdahtisham# docker images
REPOSITORY   TAG       IMAGE ID       CREATED        SIZE
ubuntu       latest    54c9d81cbb44   3 weeks ago    72.8MB
root@ubuntu:/home/mdahtisham# docker ps -a
CONTAINER ID   IMAGE     COMMAND       CREATED              STATUS                      PORTS     NAMES
c740a1f1a01b   ubuntu    "/bin/bash"   About a minute ago   Exited (0) 25 seconds ago             trust
ing_mclaren
root@ubuntu:/home/mdahtisham# docker commit ing_mclaren geeksforgeek
Error response from daemon: No such container: ing_mclaren
root@ubuntu:/home/mdahtisham# docker commit c740a1f1a01b geeksforgeek
sha256:a275f6d56733d670e922a12cb140daefa136db1a9e251b0103282d2c88a465c8
root@ubuntu:/home/mdahtisham# docker images
REPOSITORY     TAG       IMAGE ID       CREATED          SIZE
geeksforgeek   latest    a275f6d56733   11 seconds ago   72.8MB
ubuntu         latest    54c9d81cbb44   3 weeks ago      72.8MB
oot@ubuntu:/home/mdahtisham# docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub
.docker.com to create one.
Username: mdahtisham
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
oot@ubuntu:/home/mdahtisham# docker push a275f6d56733/geeksforgeek
```

Step 7: Then type the tag images name, docker hub username, and give the name it appears on the docker hub using the below command:

```
# docker tag geeksforgeek mdahtisham/geeksimage
         geeksforgeek - Image name
         mdahtisham - Docker hub username
         geeksimage - With this name Image will appear on the docker hub
```

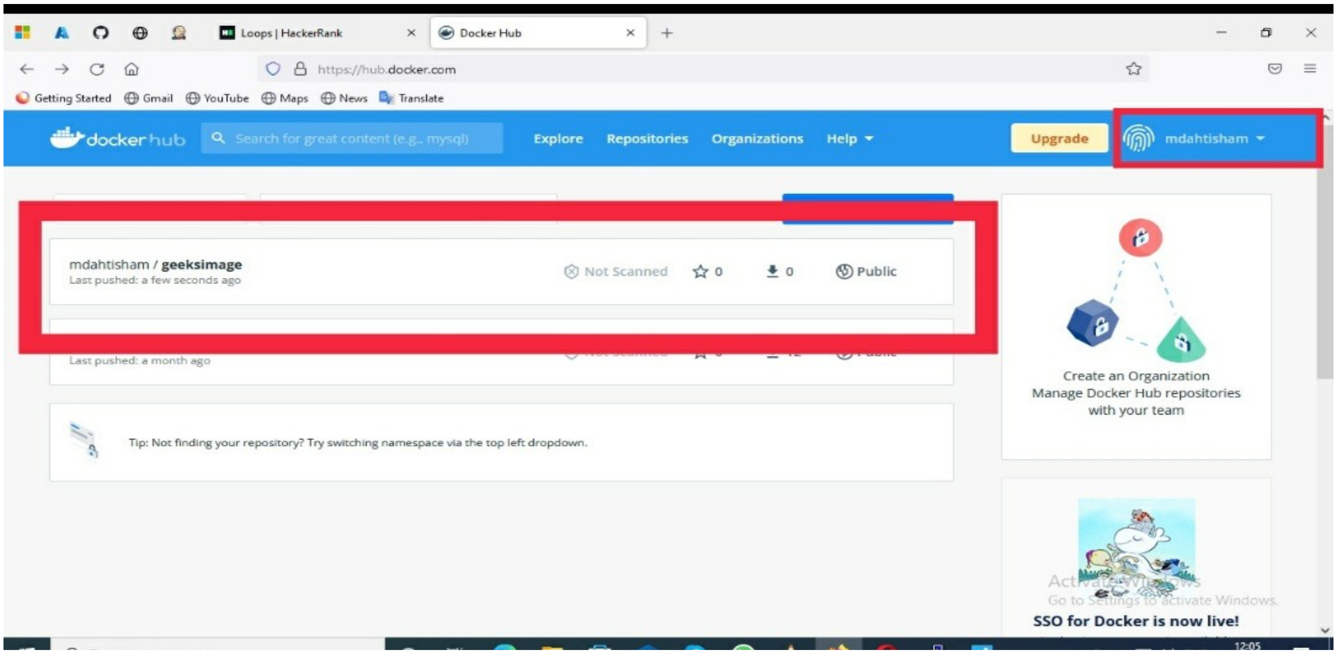
Step 8: Now push your image using the below command:

```
# docker push mdahtisham/geeksimage
```



```
CONTAINER ID   IMAGE     COMMAND       CREATED              STATUS                     PORTS     NAMES
c740a1f1a01b   ubuntu    "/bin/bash"   About a minute ago   Exited (0) 25 seconds ago            trust
ing_mclaren
root@ubuntu:/home/mdahtisham# docker commit ing_mclaren geeksforgeek
Error response from daemon: No such container: ing_mclaren
root@ubuntu:/home/mdahtisham# docker commit c740a1f1a01b geeksforgeek
sha256:a275f6d56733d670e922a12cb140daefa136db1a9e251b0103282d2c88a465c8
root@ubuntu:/home/mdahtisham# docker images
REPOSITORY     TAG       IMAGE ID       CREATED          SIZE
geeksforgeek   latest    a275f6d56733   11 seconds ago   72.8MB
ubuntu         latest    54c9d81cbb44   3 weeks ago      72.8MB
root@ubuntu:/home/mdahtisham# docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub
.docker.com to create one.
Username: mdahtisham
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
root@ubuntu:/home/mdahtisham# docker push a275f6d56733/geeksforgeek
Using default tag: latest
The push refers to repository [docker.io/a275f6d56733/geeksforgeek]
root@ubuntu:/home/mdahtisham# docker tag geeksforgeek mdahtisham/geeksimage
root@ubuntu:/home/mdahtisham# docker push mdahtisham/geeksimage
Using default tag: latest
The push refers to repository [docker.io/mdahtisham/geeksimage]
5812e48bd3c: Pushed
6ffdceb4c77: Mounted from library/ubuntu
latest: digest: sha256:0a31f3ff033322be5a6186bada3a9f53591e5ed00e1afd7b54c57cfb67975a8 size: 736
root@ubuntu:/home/mdahtisham#
```

Note: Below you can see the Docker Image successfully pushed on the docker hub: mdahtisham/geeksimage



mdahtisham / **geeksimage**
Last pushed: a few seconds ago
Not Scanned ☆ 0 ⬇ 0 Public

Last pushed: a month ago

Tip: Not finding your repository? Try switching namespace via the top left dropdown.

Create an Organization
Manage Docker Hub repositories with your team

SSO for Docker is now live!

How To Pull Docker Images from Docker Hub?

The pull command is used to get an image from the Docker Hub to the local docker environment. Follow this example to get an overview of the pull command in Docker:

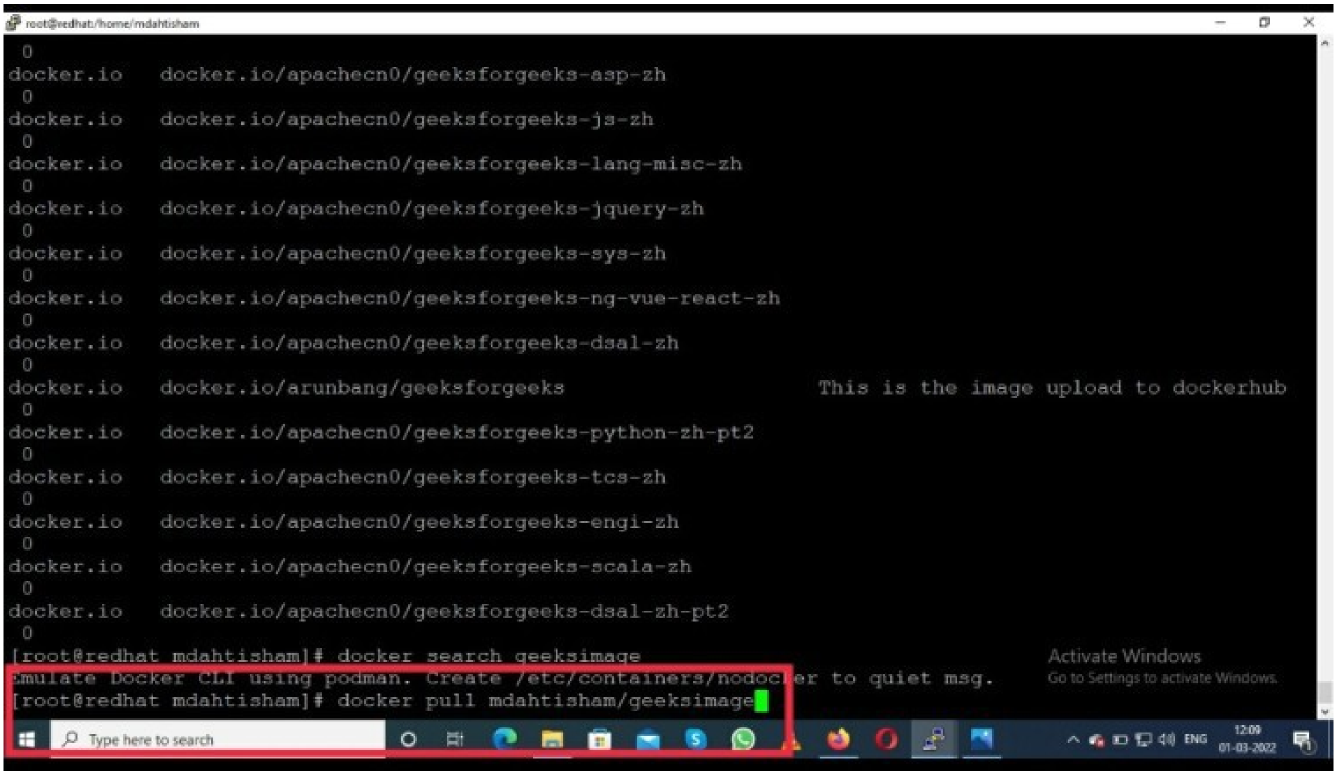
Step 1: Now you can search the image using the below command in docker as follows:

```
# docker search imagename
```

- One can see all images on your screen if available images with this name.One can also pull the images if one knows the exact name

Step 2: Now pull the image see the below command.

```
# docker pull mdahtisham/geeksimage
   mdahtisham - Docker Hub username
   geeksimage - With this name Image will appear on the docker hub
```

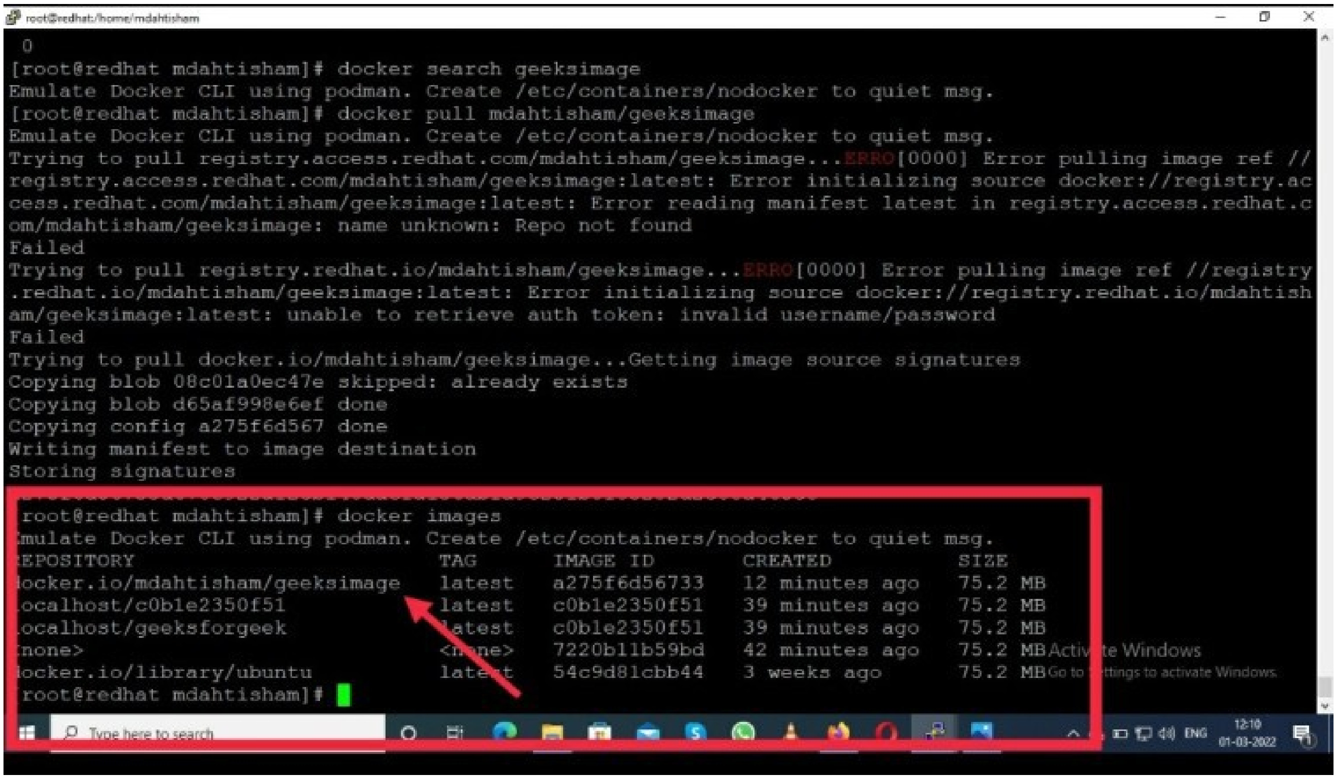


root@redhat:/home/mdahtisham
Activate Windows
Go to Settings to activate Windows.
Type here to search
ENG
12:09
01-03-2022

```
 0
docker.io   docker.io/apachecn0/geeksforgeeks-asp-zh
 0
docker.io   docker.io/apachecn0/geeksforgeeks-js-zh
 0
docker.io   docker.io/apachecn0/geeksforgeeks-lang-misc-zh
 0
docker.io   docker.io/apachecn0/geeksforgeeks-jquery-zh
 0
docker.io   docker.io/apachecn0/geeksforgeeks-sys-zh
 0
docker.io   docker.io/apachecn0/geeksforgeeks-ng-vue-react-zh
 0
docker.io   docker.io/apachecn0/geeksforgeeks-dsal-zh
 0
docker.io   docker.io/arunbang/geeksforgeeks                    This is the image upload to dockerhub
 0
docker.io   docker.io/apachecn0/geeksforgeeks-python-zh-pt2
 0
docker.io   docker.io/apachecn0/geeksforgeeks-tcs-zh
 0
docker.io   docker.io/apachecn0/geeksforgeeks-engi-zh
 0
docker.io   docker.io/apachecn0/geeksforgeeks-scala-zh
 0
docker.io   docker.io/apachecn0/geeksforgeeks-dsal-zh-pt2
 0
[root@redhat mdahtisham]# docker search geeksimage
Emulate Docker CLI using podman. Create /etc/containers/nodocker to quiet msg.
[root@redhat mdahtisham]# docker pull mdahtisham/geeksimage
```

Step 3: Now check for the pulled image in the local docker environment using the below command:

```
# docker images
```


root@redhat:/home/mdahtisham
Type here to search
ENG
12:10
01-03-2022
Activ te Windows
Go to ttings to activate Windows.

```
 0
[root@redhat mdahtisham]# docker search geeksimage
Emulate Docker CLI using podman. Create /etc/containers/nodocker to quiet msg.
[root@redhat mdahtisham]# docker pull mdahtisham/geeksimage
Emulate Docker CLI using podman. Create /etc/containers/nodocker to quiet msg.
Trying to pull registry.access.redhat.com/mdahtisham/geeksimage...ERRO[0000] Error pulling image ref //
registry.access.redhat.com/mdahtisham/geeksimage:latest: Error initializing source docker://registry.ac
cess.redhat.com/mdahtisham/geeksimage:latest: Error reading manifest latest in registry.access.redhat.c
om/mdahtisham/geeksimage: name unknown: Repo not found
Failed
Trying to pull registry.redhat.io/mdahtisham/geeksimage...ERRO[0000] Error pulling image ref //registry
.redhat.io/mdahtisham/geeksimage:latest: Error initializing source docker://registry.redhat.io/mdahtish
am/geeksimage:latest: unable to retrieve auth token: invalid username/password
Failed
Trying to pull docker.io/mdahtisham/geeksimage...Getting image source signatures
Copying blob 08c01a0ec47e skipped: already exists
Copying blob d65af998e6ef done
Copying config a275f6d567 done
Writing manifest to image destination
Storing signatures

 root@redhat mdahtisham]# docker images
mulate Docker CLI using podman. Create /etc/containers/nodocker to quiet msg.
EPOSITORY                     TAG      IMAGE ID       CREATED          SIZE
ocker.io/mdahtisham/geeksimage   latest   a275f6d56733   12 minutes ago   75.2 MB
ocalhost/c0b1e2350f51         latest   c0b1e2350f51   39 minutes ago   75.2 MB
ocalhost/geeksforgeek         atest    c0b1e2350f51   39 minutes ago   75.2 MB
none>                         <n ne>   7220b11b59bd   42 minutes ago   75.2 MB
ocker.io/library/ubuntu       late t   54c9d81cbb44   3 weeks ago      75.2 MB
root@redhat mdahtisham]# 
```

Difference between Github and Dockerhub

The following are the difference between github and dockerhub:

Feature	GitHub	Docker Hub
Primary Purpose	Code Repository and Version Control	Docker Image Repository and Management
Content	Source Code, Documentation	Docker Container Images
Integration	Works with CI/CD tools like Jenkins, Travis CI	Integrates with CI/CD tools and Docker itself
Visibility	Public and Private Repositories	Public and Private Repositories
Security	Code scanning and vulnerability alerts	Image security scans and vulnerability reports

Difference between Dockerhub and Docker Registry

The following are the differences between Dockerhub and Docker Registry:

Feature	Docker Hub	Docker Registry
Service Type	Cloud-based repository service	Self-hosted registry service
Accessibility	Public and private image repositories	Primarily private, customizable
Integration	Integrates with GitHub, Jenkins, and more	Can be integrated with various CI/CD tools
Security	Built-in security scans and vulnerability reports	Security depends on implementation
Automation	Supports webhooks for CI/CD automation	Requires manual setup for automation

Features of Docker Hub

The following are the features of dockerhub:

- Storage, management, and sharing of images with others are made simple via Docker Hub.
- Docker Hub runs the necessary security checks on our images and generates a full report on any security flaws.
- Docker Hub can automate the processes like Continuous deployment and Continuous testing by triggering the Webhooks when the new image is pushed into Docker Hub.
- With the help of Docker Hub, we can manage the permission for the users, teams, and organizations.
- We can integrate Docker Hub into our tools like GitHub, Jenkins which makes workflows easy

Advantages of Docker Hub

The following are the advantages of Docker hub:

- Docker Container Images are light in weight.
- We can push the images within a minute and with help of a command.
- It is a secure method and also provides a feature like pushing the private image or public image.
- Docker hub plays a very important role in industries as it becomes more popular day by day and it acts as a bridge between the developer team and the testing team.
- If a person wants to share their code, software any type of file for public use, you can just make the images public on the docker hub.

Why would someone use Docker?

> *Docker simplifies application deployment by containerizing apps with all their dependencies.*

Why use Docker instead of GitHub?

> *Docker is for containerizing applications, whereas GitHub is for source code version control.*

Is Docker Hub the same as GitHub?

> *No, Docker Hub is for container images, while GitHub is for source code.*

Is Docker Hub private?

> *Docker Hub offers both private and public repositories.*

How do I push an image to Docker Hub?

> *Use the `docker push <username>/<repository>` command.*

Docker Commands Cheat Sheet

The Docker cheat sheet will help you as a reference guide from where you can quickly read of mostly used common commands of Docker. The cheat sheet will help as a handy guide for developers and other system administrations who are working with Docker. Let’s get started:

Installation Commands

Name	**Command**
Installation on Linux	curl -sSL https://gcurl -fsSL https://get.docker.com -o get-docker.sh && sudo sh get-docker.sh

Docker Login Commands

Name	**Command**
Log in to a Registry	docker login
Logout from a Registry	docker logout

Image Management Commands

Docker images are self-contained software packages that contain all the necessary components to run an application. These components include the code, runtime, system tools, system libraries, and settings. Docker images are lightweight and easy to use.

Name	**Command**
Build an image	docker build -t <image_name>
Pulling an Image	docker image pull nginx
Pulling an Image Example	docker image pull <Name of The Image>:<Tag>

Image Transfer Commands

Name	**Command**
Pushing an Image	docker image push <usernameofregistry:Imagename: tag>
Pushing an Image Example	docker image push eon01/nginx localhost:5000/myadmin/nginx

Docker Hub Commands

Docker Hub is a service provided by Docker for finding and sharing container images with your team. Learn more and find images at “***https://hub.docker.com”.***

Name	Command
Login into Docker	-docker login -u <username>
Publish an image to Docker Hub	-docker push <username>/<image_name>
Search Hub for an image	-docker search <image_name>
Pull an image from a Docker Hub	-docker pull <image_name>

General Docker Commands

Name	Command
Start the docker daemon	docker -d
Get help with Docker. Can also use –help on all subcommands	docker –help
Display system-wide information	docker info

Containers Management Commands

CONTAINERS

A docker image’s runtime instance is referred to as a container. The container remains consistent regardless of the infrastructure in use. This isolation of software from its environment guarantees uniformity in function, even in cases where there are discrepancies between development and staging.

Name	Command
Starting Containers	docker container start nginx
Stopping Containers	docker container stop nginx
Restarting Containers	docker container restart nginx
Pausing Containers	docker container pause nginx
Unpausing Containers	docker container unpause nginx
Blocking a Container	docker container wait nginx

Sending SIGKILL Containers	docker container kill nginx
Sending another signal	docker container kill -s HUP nginx
Connecting to an Existing Container	docker container attach nginx
Check the Containers	docker ps
To see all running containers	docker container ls
Container Logs	docker logs infinite
'tail -f' Containers' Logs	docker container logs infinite -f
Inspecting Containers	docker container inspect infinite
Inspecting Containers for certain	docker container inspect –format '{{ .NetworkSettings.IPAddress }}' $(docker ps -q)
Containers Events	docker system events infinite
docker system events infinite	docker container port infinite
Running Processes	docker container top infinite
Container Resource Usage	docker container stats infinite
Inspecting changes to files or directories on a container's filesystem	docker container diff infinite

Docker Image Management Commands

Name	Command
Listing Images	docker image ls
Building Images	docker build.
From a Remote GIT Repository	docker build github.com/creack/docker-firefox
Instead of Specifying a Context, You Can Pass a Single Dockerfile in the URL or Pipe the File in via STDIN	docker build – < Dockerfile
Building and Tagging	docker build -t eon/infinite.

Building a Dockerfile while Specifying the Build Context	docker build -f myOtherDockerfile.
Building from a Remote Dockerfile URI	curl example.com/remote/Dockerfile | docker build -f – .
Removing an Image	docker image rm nginx
Loading a Tarred Repository from a File or the Standard Input Stream	docker image load < ubuntu.tar.gz
Saving an Image to a Tar Archive	docker image save busybox > ubuntu.tar
Showing the History of an Image	docker image history
Creating an Image From a Container	docker container commit nginx
Tagging an Image	docker image tag nginx eon01/nginx
Pushing an Image	docker image push eon01/nginx

Docker Network Commands

Name	Command
Creating an Overlay Network	docker network create -d overlay MyOverlayNetwork
Creating a Bridge Network	docker network create -d bridge MyBridgeNetwork
Creating a Customized Overlay Network	docker network create -d overlay \ –subnet=192.168.0.0/16 \ –subnet=192.170.0.0/16 \ –gateway=192.168.0.100 \ –gateway=192.170.0.100 \ –ip-range=192.168.1.0/24 \ –aux-address=”my-router=192.168.1.5" –aux-address=”my-switch=192.168.1.6" \ –aux-address=”my-printer=192.170.1.5" –aux-address=”my-nas=192.170.1.6" \ MyOverlayNetwork
Removing a Network	docker network rm MyOverlayNetwork
Listing Networks	docker network ls

Getting Information About a Network	docker network inspect MyOverlayNetwork
Connecting a Running Container to a Network	docker network connect MyOverlayNetwork nginx
Connecting a Container to a Network When it Starts	docker container run -it -d –network=MyOverlayNetwork nginx
Disconnecting a Container from a Network	docker network disconnect MyOverlayNetwork nginx

Docker Exposing Ports Commands

Name	Command
Exposing Ports	EXPOSE <port_number>
Mapping Ports	docker run -p $HOST_PORT:$CONTAINER_PORT –name <container_name> -t <image>

Docker Commands Removing Containers, Images, Volumes, And Networks

Name	Command
Removing a Running Container	docker container rm nginx
Removing a Container and its Volume	docker container rm -v nginx
Removing all Exited Containers	docker container rm $(docker container ls -a -f status=exited -q)
Removing All Stopped Containers	docker container rm `docker container ls -a -q`
Removing a Docker Image	docker image rm nginx
Removing Dangling Images	docker image rm $(docker image ls -f dangling=true -q)
Removing all Images	docker image rm $(docker image ls -a -q)
Removing all Untagged Images	docker image rm -f $(docker image ls | grep “^” | awk “{print $3}”)
Stopping & Removing all Containers	docker container stop $(docker container ls -a -q) && docker container rm $(docker container ls -a -q)

Removing Dangling Volumes	docker volume rm $(docker volume ls -f dangling=true -q)
Removing all unused (containers, images, networks and volumes)	docker system prune -f
Clean all	docker system prune -a

Docker Swarm Commands

Name	Command
Installing Docker Swarm	curl -ssl https://get.docker.com | bash
Initializing the Swarm	docker swarm init –advertise-addr 192.168.10.1
Getting a Worker to Join the Swarm	docker swarm join-token worker
Getting a Manager to Join the Swarm	docker swarm join-token manager
Listing Services	docker service ls
Listing nodes	docker node ls
Creating a Service	docker service create –name vote -p 8080:80 instavote/vote
Listing Swarm Tasks	docker service ps
Scaling a Service	docker service scale vote=3
Updating a Service	docker service update –image instavote/vote:movies vote
Updating a Service	docker service update –force –update-parallelism 1 –update-delay 30s nginx

Docker file Commands

Command	Description	Example
FROM	Specifies the base image for the build	FROM ubuntu:latest
RUN	Executes a command inside the container during build time	RUN apt-get update && apt-get install -y curl
CMD	Specifies the default command to run when the container starts	CMD [“npm”, “start”]

EXPOSE	Informs Docker that the container listens on specific network ports at runtime	EXPOSE 80/tcp
ENV	Sets environment variables inside the container	ENV NODE_ENV=production
COPY	Copies files or directories from the build context into the container	COPY app.js /usr/src/app/
ADD	Similar to COPY but supports additional features like URL retrieval and decompression	ADD https://example.com/file.tar.gz /usr/src/
WORKDIR	Sets the working directory for subsequent instructions	WORKDIR /usr/src/app
ARG	Defines variables that users can pass at build-time to the builder with the docker build command	ARG VERSION=1.0
ENTRYPOINT	Configures a container to run as an executable	ENTRYPOINT [“python”, “app.py”]
VOLUME	Creates a mount point and assigns it to a specified volume	VOLUME /data
USER	Sets the user or UID to use when running the image	USER appuser
LABEL	Adds metadata to an image in the form of key-value pairs	LABEL version=”1.0" maintainer=”John Doe
ONBUILD	Configures commands to run when the image is used as the base for another build	ONBUILD ADD . /app/src

Docker Volume Commands

Command	Description	Example
volume create	Creates a named volume	docker volume create mydata
volume ls	Lists the available volumes	docker volume ls
volume inspect	Displays detailed information about a volume	docker volume inspect mydata
volume rm	Removes one or more volumes	docker volume rm mydata
volume prune	Removes all unused volumes	docker volume prune

Docker CP commands

Command	Description	Example
docker cp [OPTIONS] SRC_PATH CONTAINER:DEST_PATH	Copies files or directories from the local filesystem to the specified container	docker cp myfile.txt mycontainer:/usr/src/app/
docker cp [OPTIONS] CONTAINER:SRC_PATH DEST_PATH	Copies files or directories from the specified container to the local filesystem	docker cp mycontainer:/usr/src/app/result.txt /tmp/result/

Docker Security Commands (Docker Scout)

Command	Description	Example
docker scout compare	[experimental] Compare two images and display differences	docker scout compare image1:tag image2:tag
docker scout cves	Display CVEs identified in a software artifact	docker scout cves image: tag
docker scout Quickview	Quick overview of an image	docker scout quickview image: tag
docker scout recommendations	Display available base image updates and remediation recommendations	docker scout recommendations image:tag
docker scout version	Show Docker Scout version information	docker scout version

1. What is the architecture of Docker?

Answer:

> *Docker follows a client-server architecture. The Docker client communicates with the Docker daemon, which is responsible for building, running, and managing Docker containers. The client and daemon can run on the same host, or the client can connect to a remote daemon.*

2. Which language is Docker built on?

Answer:

> *Docker is built using Go programming language because of its advantage of several features of the Linux kernel to deliver its functionality.*

3. Does Docker require coding?

Answer:

No, Docker does not require any prior coding knowledge. It is a containerization platform that enables developers to package, deploy, and run applications using containers.

4. Are Docker secrets safe?

Answer:

You can use Docker secrets to centrally manage this data and securely transmit it to only those containers that need access to it. Secrets are encrypted during transit and at rest in a Docker swarm.

5. How many types of volumes are there in Docker?

Answer:

Docker supports three types of volumes:

a) ***Named Volumes:*** *These are volumes with a user-defined name that can be used across multiple containers.*

b) ***Bind Mounts:*** *These are directories on the host machine that are mounted into a container, allowing direct access to the host's file system.*

c) ***tmpfs Mounts:*** *These are volumes stored in the host's memory, allowing fast read and write operations but with limited size and durability.*

6. What is the flag in Docker?

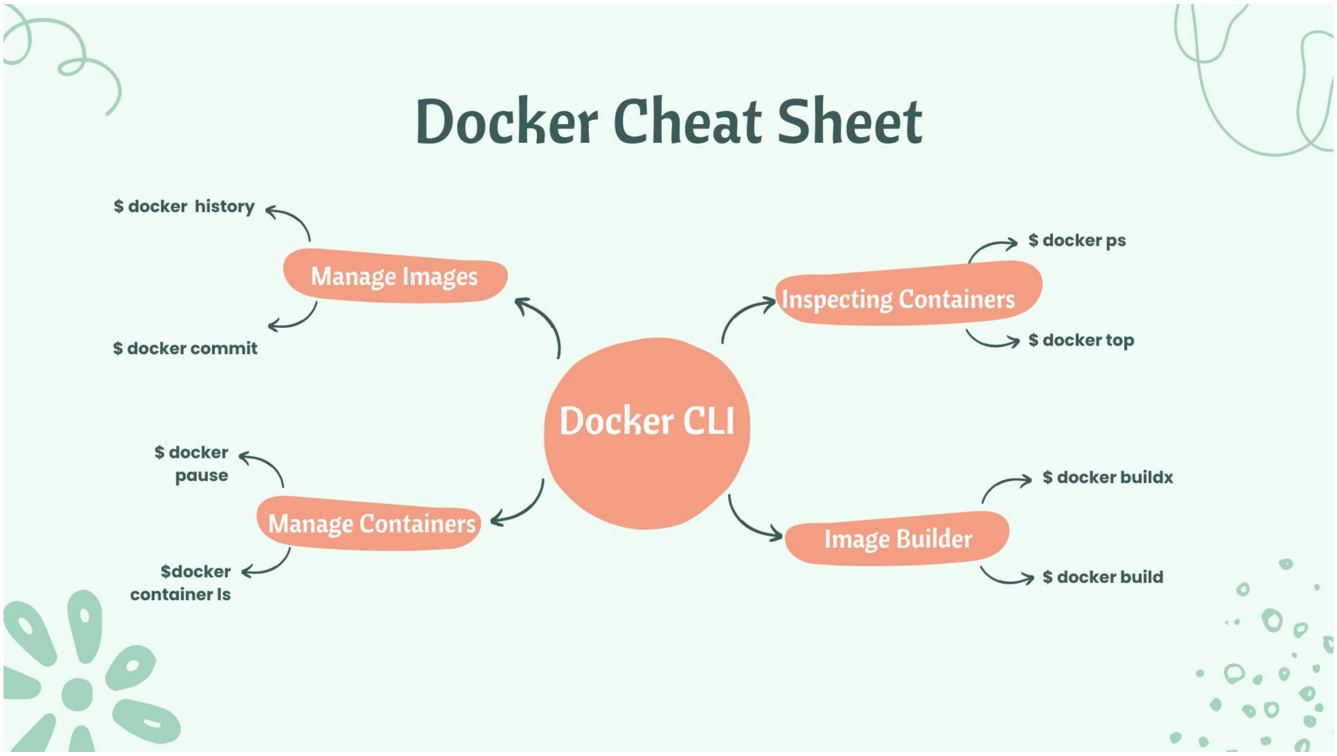
Answer:

In Docker, a flag is a command-line option that modifies the behavior of a Docker command. Flags are used to provide additional instructions or parameters to Docker commands, allowing you to customize the execution according to your needs.

7. Why is Docker used in DevOps?

Answer:

Docker is widely used in DevOps practices due to its ability to create reproducible and portable environments. With Docker, developers can package their applications and dependencies into containers, ensuring consistent behavior across different stages of the software development lifecycle. Docker also facilitates the automation of deployment, testing, and scaling processes, enabling faster and more reliable software delivery in DevOps pipelines.



Docker Cheat Sheet

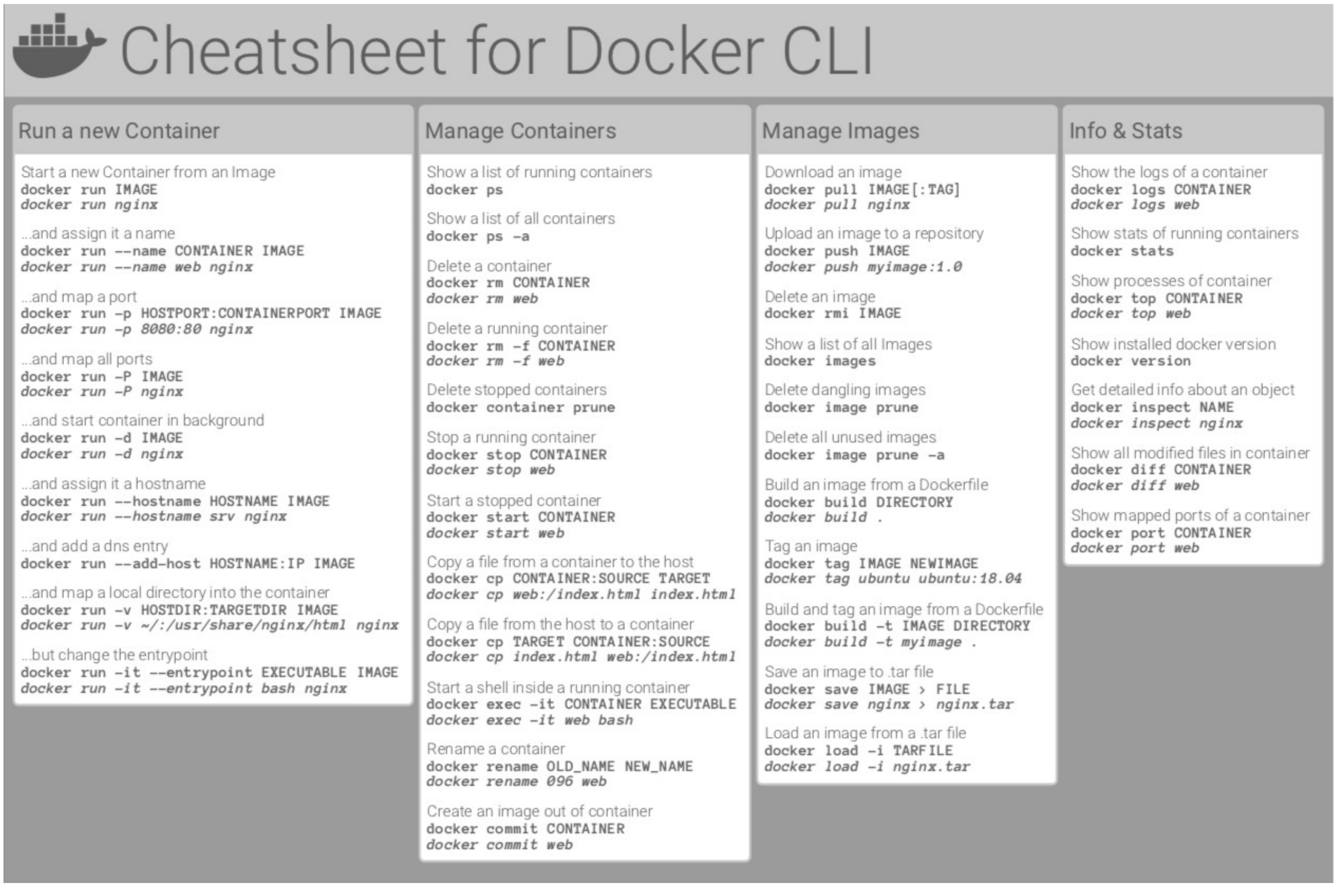
Docker CLI

Manage Images
- $ docker history
- $ docker commit

Inspecting Containers
- $ docker ps
- $ docker top

Manage Containers
- $ docker pause
- $docker container ls

Image Builder
- $ docker buildx
- $ docker build



Cheatsheet for Docker CLI

Run a new Container

Start a new Container from an Image
```
docker run IMAGE
docker run nginx
```

...and assign it a name
```
docker run --name CONTAINER IMAGE
docker run --name web nginx
```

...and map a port
```
docker run -p HOSTPORT:CONTAINERPORT IMAGE
docker run -p 8080:80 nginx
```

...and map all ports
```
docker run -P IMAGE
docker run -P nginx
```

...and start container in background
```
docker run -d IMAGE
docker run -d nginx
```

...and assign it a hostname
```
docker run --hostname HOSTNAME IMAGE
docker run --hostname srv nginx
```

...and add a dns entry
```
docker run --add-host HOSTNAME:IP IMAGE
```

...and map a local directory into the container
```
docker run -v HOSTDIR:TARGETDIR IMAGE
docker run -v ~/:/usr/share/nginx/html nginx
```

...but change the entrypoint
```
docker run -it --entrypoint EXECUTABLE IMAGE
docker run -it --entrypoint bash nginx
```

Manage Containers

Show a list of running containers
```
docker ps
```

Show a list of all containers
```
docker ps -a
```

Delete a container
```
docker rm CONTAINER
docker rm web
```

Delete a running container
```
docker rm -f CONTAINER
docker rm -f web
```

Delete stopped containers
```
docker container prune
```

Stop a running container
```
docker stop CONTAINER
docker stop web
```

Start a stopped container
```
docker start CONTAINER
docker start web
```

Copy a file from a container to the host
```
docker cp CONTAINER:SOURCE TARGET
docker cp web:/index.html index.html
```

Copy a file from the host to a container
```
docker cp TARGET CONTAINER:SOURCE
docker cp index.html web:/index.html
```

Start a shell inside a running container
```
docker exec -it CONTAINER EXECUTABLE
docker exec -it web bash
```

Rename a container
```
docker rename OLD_NAME NEW_NAME
docker rename 096 web
```

Create an image out of container
```
docker commit CONTAINER
docker commit web
```

Manage Images

Download an image
```
docker pull IMAGE[:TAG]
docker pull nginx
```

Upload an image to a repository
```
docker push IMAGE
docker push myimage:1.0
```

Delete an image
```
docker rmi IMAGE
```

Show a list of all Images
```
docker images
```

Delete dangling images
```
docker image prune
```

Delete all unused images
```
docker image prune -a
```

Build an image from a Dockerfile
```
docker build DIRECTORY
docker build .
```

Tag an image
```
docker tag IMAGE NEWIMAGE
docker tag ubuntu ubuntu:18.04
```

Build and tag an image from a Dockerfile
```
docker build -t IMAGE DIRECTORY
docker build -t myimage .
```

Save an image to .tar file
```
docker save IMAGE > FILE
docker save nginx > nginx.tar
```

Load an image from a .tar file
```
docker load -i TARFILE
docker load -i nginx.tar
```

Info & Stats

Show the logs of a container
```
docker logs CONTAINER
docker logs web
```

Show stats of running containers
```
docker stats
```

Show processes of container
```
docker top CONTAINER
docker top web
```

Show installed docker version
```
docker version
```

Get detailed info about an object
```
docker inspect NAME
docker inspect nginx
```

Show all modified files in container
```
docker diff CONTAINER
docker diff web
```

Show mapped ports of a container
```
docker port CONTAINER
docker port web
```

https://phoenixnap.com/kb/docker-commands-cheat-sheet
https://dockerlabs.collabnix.com/docker/cheatsheet/
https://dockerlabs.collabnix.com/intermediate/docker-compose/compose-cheatsheet.html

Container Management

`docker ps`	List the running containers.	`docker logs -f --until=[interval] [container]`	Retreive logs before a specific point in time.
`docker ps -a`	List all the containers, both running and stopped.	`docker events [container]`	View real time events for a container.
`docker create [image]`	Create a container without starting it.	`docker update [container]`	Update the configuration of a container.
`docker create -it [image]`	Create an interactive container with pseudo-TTY.	`docker port [container]`	Show port mapping for a container.
`docker rename [container] [new-name]`	Rename a container.	`docker top [container]`	Show running processes in a container.
`docker rm [container]`	Remove a stopped container.	`docker stats [container]`	Show live resource usage statistics for a container.
`docker rm -f [container]`	Force remove a container, even if it is running.	`docker diff [container]`	Show changes to files or directories on the filesystem.
`docker logs [container]`	View logs for a running container.	`docker cp [file-path] CONTAINER:[path]`	Copy a local file to a directory in a container.

Running a Container

`docker run [image] [command]`	Run a command in a container based on an image.	`docker restart [container]`	Stop a container and start it again.
`docker run --name [container-name] [image]`	Create, start, and name a container.	`docker pause [container]`	Pause processes in a running container.
`docker run -p [host]:[container-port] [image]`	Map a host port to a container port.	`docker unpause [container]`	Unpause processes in a running container.
`docker run --rm [image]`	Run a container and remove it after it stops.	`docker wait [container]`	Block input until the container stops.
`docker run -d [image]`	Run a detached (background) container.	`docker kill [container]`	Send a SIGKILL signal to stop a container.
`docker run -it [image]`	Run an interactive process, e.g., a shell, in a container.	`docker attach [container]`	Attach local standard input, output and error.
`docker start [container]`	Start a container.	`docker exec -it [container] [shell]`	Run a shell inside a running container.
`docker stop [container]`	Stop a container.		

Image Management

`docker build [dockerfile-path]`	Create an image from a Dockerfile.	`docker tag [image] [image]:[tag]`	Tag an image.
`docker build .`	Build an image using the files from the current path.	`docker images`	Show all locally stored top level images.
`docker build -t [name]:[tag] [location]`	Create an image from a Dockerfile and tag it.	`docker history [image]`	Show history for an image.
`docker build -f [file]`	Specify a file to build from.	`docker rmi [image]`	Remove an image.
`docker pull [image]`	Pull an image from a registry.	`docker load --image [tar-file]`	Load an image from a tar archive file.
`docker push [image]`	Push an image to a registry.	`docker save [image] > [tar-file]`	Save an image to a tar archive file.
`docker import [url/file]`	Create an image from a tarball.	`docker search [query]`	Search Docker Hub for images.
`docker commit [container] [new-image]`	Create an image from a container.	`docker image prune`	Remove unused images.

Networking

`docker network ls`	View available networks.
`docker network rm [network]`	Remove a network.
`docker network inspect [network]`	Show information about a network.
`docker network connect [network] [container]`	Connect a container to a network.
`docker network disconnect [network] [container]`	Disconnect a container from a network.

General Management

`docker login`	Log in to a Docker registry.
`docker logout`	Log out of a Docker registry.
`docker inspect [object]`	Show low-level information about an object.
`docker version`	Show the version of the local Docker installation.
`docker info`	Display information about the system.
`docker system prune`	Remove unused images, containers, and networks.

Plugin Management

`docker plugin enable [plugin]`	Enable a Docker plugin.
`docker plugin disable [plugin]`	Disable a Docker plugin.
`docker plugin create [plugin] [path-to-data]`	Create a plugin from config.json and rootfs.
`docker plugin inspect [plugin]`	View details about a plugin.
`docker plugin rm [plugin]`	Remove a plugin.