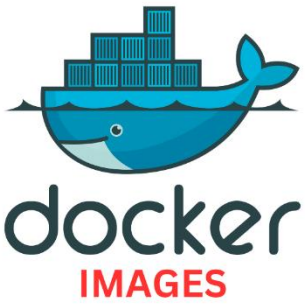
docker
IMAGES

❖ DOCKER IMAGES:

- Docker image is a read-only template with instructions for creating a Docker container.
- An image is based on another image, with some additional customization.
- An image is an executable package that includes everything needed to run an application: code, a runtime, libraries, environment variables, and configuration files.
- There are two important principles of images:
- Images are immutable. Once an image is created, it can't be modified. You can only make a new image or add changes on top of it.
- Container images are composed of layers. Each layer represented a set of file system changes that add, remove, or modify files.
- Finding images
- Docker Hub is the default global marketplace for storing and distributing images. It has over 100,000 images created by developers that you can run locally. You can search for Docker Hub images and run them directly from Docker Desktop.

HOW DOCKER IMAGES WORK:

- A Docker image, or container image, is a standalone, executable file used to create a container. This container image contains all the libraries, dependencies, and files that the container needs to run.
- A Docker image is shareable and portable, so you can deploy the same image in multiple locations at once—much like a software binary file.
- You can store images in registries to keep track of complex software architectures, projects, business segments, and user group access.
- For instance, the public Docker Hub registry contains images such as operating systems, programming language frameworks, databases, and code editors.

SYNTAX: $docker images [OPTIONS] [REPOSITORY [:TAG]]

DOCKER SEARCH:

- It will search Docker Hub for Images:

→ **Displays images with a name containing centos, busybox, ubuntu:**

$docker search centos

$docker search busybox

$docker search ubuntu

→ **images with a name containing 'ubuntu' and at least 3 stars**:

$docker search --filter stars=3 ubuntu

→ **Displays images with a name containing 'ubuntu', at least 3 stars and are official builds:**

$docker search --filter is-official=true --filter stars=3 ubuntu

→ **Display non-truncated description (--no-trunc):**

$docker search --filter=stars=3 --no-trunc ubuntu

DOCKER IMAGE PULL / DOCKER PULL:

- Pull an image from Docker Hub

$docker image pull ubuntu

→ **To see which images are present locally:**

$docker images

$docker image ls

→ **To know the digest of an image, pull the image first. pull the latest ubuntu:22.04 image from Docker Hub:**

$docker pull ubuntu:22.04

→ **Pull a repository with multiple images (-a, --all-tags):**

$docker image pull --all-tags ubuntu

$docker images

$docker image ls --filter reference=ubuntu

NOTE: To Cancel a pull:

Killing the docker pull process, for example by pressing **CTRL-c** while it is running in a terminal, will terminate the pull operation.

$docker pull ubuntu

DOCKER IMAGE HISTORY / DOCKER HISTORY:

- It Shows the history of an image

→ **To see how the docker:latest image was built:**

$docker history docker

$docker image history a9d583973f65

DOCKER IMAGE INSPECT:

- Display detailed information on one or more images

$docker image inspect ubuntu

DOCKER IMAGE TAG / DOCKER TAG:

- Create tag TARGET_IMAGE that refers to SOURCE_IMAGE.

→ **To tag a local image with ID 0e5574283393 as fedora/httpd with the tag version1.0:**

$docker tag 0e5574283393 fedora/httpd:version1.0

→ **To tag a local image httpd as fedora/httpd with the tag version1.0:**

$docker tag httpd fedora/httpd:version1.0

DOCKER IMAGE SAVE / DOCKER SAVE:

- Save one or more images to a tar archive (streamed to STDOUT by default)

→ **Create a backup that can then be used with docker load:**

$docker save ubuntu > ubuntu.tar

$ls -sh ubuntu.tar

$docker save --output ubuntu.tar ubuntu

$ls -sh ubuntu.tar

→ **Save an image to a tar.gz file using gzip:**

$docker save myimage:latest | gzip > myimage_latest.tar.gz

DOCKER IMAGE RM / DOCKER RMI:

- Remove one or more images

→ **List images in local:**

$docker images

$docker rmi ubuntu

$docker images

→ **untags and removes all images that match the specified ID:**

$docker rmi -f fd484f19954f

DOCKER IMAGE PRUNE:

- Remove unused images
- Remove all dangling images. If -a is specified, also remove all images not referenced by any container.

→ **Removes images with the label deprecated:**

$docker image prune --filter="label=deprecated"

→ **Removes images with the label maintainer set to ram:**

$docker image prune --filter="label=maintainer=ram"

→ **This example removes images which have no maintainer label:**

$docker image prune --filter="label!=maintainer"

→ **This example removes images which have a maintainer label not set to john:**

$docker image prune --filter="label!=maintainer=john"

→ **Removing all unused images**

$docker image prune -a