**Docker Commands Sheet**

Docker Commands Cheat Sheet | Buddy



# 🡪Docker comment

# 🡪Basic Docker commands:

**Docker <command> --<a,e,etc[attribute]> <[image name]/[container name]> : <version>**

## Docker List container commands

* docker ps : List-all-the-running-containers
* docker ps –a / -all : List all the Containers (irrespective of the state)
* docker ps –s / docker container ls –s : List all the Running Containers with the File Size
* docker ps –q / docker container ls –q : List the IDs of the Running Containers
* docker ps -a –q / docker ps –aq : List the IDs of all the Containers (irrespective of the state)
* docker ps -f name=un /docker ps -a -f name=ar : Filter container list
* docker ps -a -f status=running : filtyer based on status

## Docker create container commands

* docker create <image\_name> / docker container create <image\_name> : create new container
* docker create --name <container\_name> <image\_name> / docker container create --name <container\_name> <image\_name> : Creating a new Container using Docker Image with some fixed name

## Start a Docker Container

* docker start <container\_id or container\_name>
* docker container start <container\_id or container\_name>

## Stop a running Docker Container

* docker stop <container\_id or container\_name>
* docker container stop <container\_id or container\_name>

## Restart a Docker container

* docker restart <container\_id or container\_name>
* docker container restart <container\_id or container\_name>

## Pause a running Container

* docker pause <container\_id or container\_name>

## Resume/Unpause a paused Container

* docker unpause <container\_id or container\_name>
* docker container unpause <container\_id or container\_name>

## Docker Run command

* docker run <image\_name>
* docker container run <image\_name>

### Attributes:

## Run the container in background daemon mode

* docker run -d <image\_name>

## Run Docker Container with a name using the run command

* docker container run -d --name <container\_name> <image\_name>

## Docker Run command in Foreground and Detached Modes

The Docker container can run in two modes:

🡪 Background or detached or daemon mode

Docker can run in background this will not stop the current terminal and new container can also be used in background

🡪 Foreground mode

Docker runs the container in the foreground mode by default. In this mode, Docker starts the root process in the container in the foreground and attaches the standard input(stdin), output(stdout), and error(stderr) of the process to the terminal session.

In the foreground mode, you cannot execute any other command on the terminal session until the container is running. This is the same as running a [Linux process in the foreground mode](https://buddy.works/docs/on-premises/installation/linux).

## Delete the container on the exited state

* docker run --rm nginx

## Listing Processes running in a Docker Container

* docker top <container\_name or container\_id>
* docker container top <container\_name or container\_id>

## Run the Docker Container in an Interactive Mode

* docker container run -it <image\_name> /bin/bash

## Map ports of a Docker Container

The port mappings allow us to map a port on the container with a different port on the host machine.

* docker container run --name <container\_name> -d -p <host\_post>:<container\_port> <image\_name>

## Rename a Docker Container

* docker rename <old\_name> <new\_name>
* docker container rename <old\_name> <new\_name>

## Get Inside the Running Container (Literally!)

The docker exec command executes the /bin/bash command and starts a bash shell session inside the container as shown in the below screenshot:

* docker exec -it <contaner\_id or container\_name> /bin/bash

## Start a Docker Container and keep it running

* docker run -dt <image\_name>

## Copy a File from a Container to a Host

* docker cp <container\_id or container\_name>:<source\_file\_path> <destination\_path>

## Copy a File from the Host to the Docker Container

* docker cp <location\_of\_file\_on\_host> <container\_id or container\_name>:<file\_desinaion>

## Remove a specific Docker Container

* docker rm <container\_name or container\_id> : remove after container is stopped
* docker rm -f <container\_name or container\_id> : remove without stopping

## Remove a Docker Container after it exits

* docker run --rm <image\_name>

## Delete all the Stopped Containers

* docker container prune

## Delete all the Docker Containers

This command is used to delete all the running as well as the stopped containers:

* docker rm -f $(docker ps -a -q)
* docker container rm -f $(docker ps -a -q)

## Create a Docker image from a Docker Container

* docker container commit <container\_id or container\_name> <new\_image\_name>
* docker commit <container\_id or container\_name> <new\_image\_name>

## Run command inside the Docker Container

* docker exec -it <container\_id or container\_name> <command>
* docker exec -it f40cc2e51d5b ps -afe

## Set Environment Variable in a Docker Container

* docker run --env ENV\_VAR1=value1 --env ENV\_VAR1=value2 --name <container\_name> <image\_name>

## Set Environment Variable in a Docker Container using a File

* docker run --env-file <path\_to\_the\_file> --name <container\_name> <image\_name>
* docker run -dt --env-file file1.txt --name centos-container-1 centos

# 🡪Docker Image Commands

## List all the Docker Images

* docker images

## List the Docker Image Ids

* docker images -q

## List all the Docker Images (including dangling images)

* docker images -a

## Build a Docker Image

* docker build -t <image\_name> <contenxt\_dir>

## Build Docker Images with a different tag

* docker build . -t <image\_name>:<tag or version>
* docker build . -t centos\_buddy:1.8

## Build a Docker Image using a custom named Dockerfile

* docker build -f <custom\_docker\_file\_name> -t <image\_name> .
* docker build -f custom\_docker\_file -t centos\_custom .

## Build a Docker Image from a Dockerfile that is not in the Current Directory

* docker build -f </path/to/dockerfilename> -t <image\_name> .

## Show History of a Docker Image

* docker history <imagename or imageid>

## Rename an existing Docker Image

* docker tag <imagename> <newname>:<version>

## Remove Docker images

* docker rmi <image\_name or image\_id>

## Force delete a Docker Image

* docker rmi -f <image\_name or image\_id>

## Unused Docker Images

* Unused Docker images are not used by any containers.
* The images that are displayed when we do docker ps -a are used by some of the existing containers.
* So, the unused images are:
* (All images from docker images -a) - (all images from docker ps -a)

## Dangling Docker Images

* When we build a Docker image using Dockerfile, Docker creates an image with the given name.
* Here's a simple example:
* docker build . -t imagename
* Docker will create an image from the Dockerfile in the current directory with the name imagename.
* If we do some changes in the Dockerfile and rebuild the image again with the same name, Docker will update the name of the previous image to <none> and tag it <none>.
* These images with the name <none> and tag <none> are called dangling images.

## List Dangling Docker Images

* docker images -f dangling=true

## Remove all the Dangling Docker Images

* docker image prune
* If the dangling images are referenced by containers (either running or not running), Docker will not prune these dangling images.
* To remove dangling images, we've to make sure that they are not referenced by any container.
* We can first run docker container prune to remove all the stopped containers and the docker images command will now remove the dangling images that were referenced by these stopped containers.
* We can also use the below command to remove the dangling images:
* docker rmi $(docker images -f dangling=true -q)
* docker images -f dangling=true -q would return the Ids of all the dangling images.

## Remove all the Dangling and Unused Docker Images

* docker image prune -a

# 🡪Login to Docker

* To login to Docker hub, we can use the below command:
* docker login

## Push a Docker Image to the Docker Registry

* docker push repository\_name/imagename:tag

## Download a Docker Image from the registry

* docker pull imagename:tag

# 🡪Docker logs

## Get Logs of the Docker container

* docker container logs <container\_id or container\_name>

## Monitor the Docker Container Logs

* docker container logs -f <container\_id or container\_name>

## Get the last 2 lines of the Container Logs

* docker container logs --tail 2 <container\_id or container\_name>

# 🡪Docker Network Commands

## List all the Networks

* docker network ls

## Create a Network

* docker network create --driver <driver-name> <bridge-name>
* driver-name can be either bridge or overlay
* bridge would be used by default if --driver option is not provided.
* docker network create --driver bridge new-network
* The network called new-network is created successfully!

## Connect a Docker Container to a Network:

* docker network connect <network\_id or network\_name> <container\_id or container\_name>

## Connect a Docker Container to a Network on Start

* docker run -d --network=<network\_name or id> <contaienr\_name>

## Disconnect a Docker container from a Network:

* docker network disconnect <network\_name\_or\_id> <container\_name\_or\_id>

## Remove a Network

* docker network rm <network\_id or network\_name>

## Show Information about one or more Networks

* docker network inspect <network\_id or network\_name>

## Get the IP Address of the running Docker Container

* sudo docker inspect -f '{{range.NetworkSettings.Networks}}{{.IPAddress}}{{end}}' <container\_name or container\_id>

# 🡪Docker Volumes

## Create Docker Volume

* docker volume create --name volume-name

## List Docker Volumes

* docker volume ls

## Mounting Docker Volume using the -v Flag

* We can mount the volume inside the Docker container once it is created using the below command:
* docker run -it --name <container-name> -v <volume-name>:<path-in-container-where-volume-is-mounted> <image-name>
* We are creating a new container with the container name <container-name> using the image <image-name> and then mount the volume volume-name inside the container at path-in-container-where-volume-is-mounted.

## Mounting Docker Volume using the --mount Flag

* docker run -it --name <container-name> --mount soure=<volume-name>, destination=<path-in-container-where-volume-is-mounted> <image-name>

## Get Details about a Docker Volume

* docker volume inspect <volume-name>

## Remove a Docker Volume

* To remove the volume, we first have to remove the containers using that volume and then only we can remove the volume.
* To remove the volume we can use the below command:
* docker volume rm <volume-name>

## Volume Mount using bind-mount

* To mount any specific host directory inside the container, we have to use the below Docker run command:
* docker run -it -v /path/on/host:/path/in/contianer/where/volume/has/to/be/mounted <image-name>

## Creating Bind Mount Volume using the --mount flag

* docker run -it --name <container\_name> --mount type=bind,source=/path/on/host/,target=/path/on/container first-image
* If a directory in a container has some content and you mount the volume with type bind onto that directory, the existing content of that directory would be lost and you get an empty directory.

# 🡪Docker System-wide Commands

## Docker Info

* docker info

## Docker Stats of the running Container

* docker stats

## Docker Stats of all the Containers

* docker stats --all

## Show the Docker Version

* docker version

## Get Detailed Info about an Object (Container, Image, Volume, etc)

* docker inspect <name or id>

## Get the Summary of Docker Usage

* docker system df

1. The total size of all the containers
2. The total size of all the images
3. The total size of the volumes
4. Cache

## Clean your Docker system

* docker system prune

1. All the stopped containers
2. All the networks not used by at least one container
3. All the dangling images
4. All the dangling build cache

# 🡪docker –help

* Usage: docker [OPTIONS] COMMAND
  + A self-sufficient runtime for containers
* Options:
* Management Commands:

|  |  |
| --- | --- |
| builder | Manage builds |
| config | Manage Docker configs |
| container | Manage containers |
| context | Manage contexts |
| image | Manage images |
| network | Manage networks |
| node | Manage Swarmnodes |
| plugin | Manage plugins |
| secret | Manage Docker secrets |
| service | Manage services |
| stack | Manage Docker stacks |
| swarm | Manage Swarm |
| system | Manage Docker |
| trust | Manage truston Docker images |
| volume | Manage volumes |

* Commands:

attach Attach local standard input, output, and error streams to a running container

build Build an image from a Dockerfile

commit Create a new image from a container's changes

cp Copy files/folders between a container and the local filesystem

create Create a new container

diff Inspect changes to files or directories on a container's filesystem

events Get real time events from the server

exec Run a command in a running container

export Export a container's filesystem as a tar archive

history Show the history of an image

images List images

import Import the contents from a tarball to create a filesystem image

info Display system-wide information

inspect Return low-level information on Docker objects

kill Kill one or more running containers

load Load an image from a tar archive or STDIN

login Log in to a Docker registry

logout Log out from a Docker registry

logs Fetch the logs of a container

pause Pause all processes within one or more containers

port List port mappings or a specific mapping for the container

ps List containers

pull Pull an image or a repository from a registry

push Push an image or a repository to a registry

rename Rename a container

restart Restart one or more containers

rm Remove one or more containers

rmi Remove one or more images

run Run a command in a new container

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

search Search the Docker Hub for images

start Start one or more stopped containers

stats Display a live stream of container(s) resource usage statistics

stop Stop one or more running containers

tag Create a tag TARGET\_IMAGE that refers to SOURCE\_IMAGE

top Display the running processes of a container

unpause Unpause all processes within one or more containers

update Update configuration of one or more containers

version Show the Docker version information

wait Block until one or more containers stop, then print their exit codes

* Run 'docker COMMAND --help' for more information on a command.

# 🡪Conclusion

* This is an extensive guide on using Docker.
* We learned many useful commands related to Docker containers, images, networks and volumes in this article.

***Thank You***