**Docker Cheat Sheet: All the Most Essential Commands in One Place + Downloadable PDF**

**Docker Architecture**

Docker architecture consists of five main components: server, client, container, image, and registry.

**Docker Server**

A Docker server or Docker daemon is a program that runs in the background of your computer and manages Docker containers and images. When you use the Docker command line interface

(CLI) to create, run, or manage containers, you interact with the Docker daemon.

The Docker daemon is an essential platform component that ensures containers can be started and stopped automatically when the system boots up.

**Docker Client**

The Docker client lets users interact with the Docker daemon with its command-line interface (CLI). In simple terms, it’s the main part of the Docker architecture for creating, managing, and [**running container applications**](https://www.hostinger.in/tutorials/docker-start-a-container/).

When you use the Docker CLI to pass a command, the Docker client sends the command to the Docker daemon running on your computer, which then carries out the requested operation. The [**Docker client can be installed**](https://www.hostinger.in/tutorials/how-to-install-docker-on-ubuntu) on any machine that needs to interact with the Docker daemon, including your local machine, a remote server, or a [**virtual server**](https://www.hostinger.in/vps-hosting).

**Docker Container**

A Docker container is a package that contains all the required prerequisites to run an application.

Containers are designed to be highly portable, meaning that they can be easily moved from one environment to another, such as from a developer’s laptop to a testing environment or from a testing environment to a production environment.

**Docker Image**

A Docker image is a preconfigured template that specifies what should be included in a Docker container. Usually, images are downloaded from websites like [**Docker Hub**](https://hub.docker.com/). However, it’s also possible to create a custom image with the help of Dockerfile.

**Docker Registry**

The Docker registry is a central repository that stores and manages Docker images. It is a server-based system that lets users store and share Docker images with others, making it easy to distribute and deploy applications. The most notable Docker

registry is [**Docker Hub**](https://hub.docker.com/)

**Build Commands**

Docker uses the build command for building images from a Docker file. Some of the most common commands include:

|  |  |
| --- | --- |
| **Command** | **Explanation** |
| docker build | Builds an image from a Dockerfile located in the current directory |
| docker build https://github.com/docker/rootfs.git#container:docker | Builds an image from a remote GIT repository |
| docker build -t imagename/tag | Builds and tags an image for easier tracking |
| docker build https://yourserver/file.tar.gz | Builds an image from a remote tar archive |
| docker build -t image:1.0 -<<EOFFROM busyboxRUN echo "hello world"EOF | Builds an image via a Dockerfile that is passed through STDIN |

**Clean Up Commands**

To keep your system clean and save disk space, it’s a great idea to clean up unused images, containers, and volumes. Check the commands below for more details:

|  |  |
| --- | --- |
| **Command** | **Explanation** |
| docker image prune | Clears an unused image |
| docker image prune -a | Clears all images that are not being used by containers |
| docker system prune | Removes all stopped containers, all networks not used by containers, all dangling images, and all build cache |
| docker image rm image | Removes an image |
| docker rm container | Removes a running container |
| docker kill $ (docker ps -q) | Stops all running containers |
| docker swarm leave | Leaves a swarm |
| docker stack rm stackname | Removes a swarm |
| docker volume rm $(docker volume ls -f dangling=**true** -q) | Removes all dangling volumes |
| docker rm $(docker ps -a -q) | Removes all stopped containers |
| docker kill $ (docker ps -q) | Stops all running containers |

**Container Interaction Commands**

Interact with your Docker container with the following common commands:

|  |  |
| --- | --- |
| **Command** | **Explanation** |
| docker start container | Starts a new container |
| docker stop container | Stops a container |
| docker pause container | Pauses a container |
| docker unpause container | Unpauses a container |
| docker restart container | Restarts a container |
| docker wait container | Blocks a container |
| docker export container | Exports container contents to a tar archive |
| docker attach container | Attaches to a running container |
| docker wait container | Waits until the container is terminated and shows the exit code |
| docker commit -m “commit message” -a “author” container username/image\_name: tag | Saves a running container as an image |
| docker logs -ft container | Follows container logs |
| docker exec -ti container script.sh | Runs a command in a container |
| docker commit container image | Creates a new image from a container |
| docker create image | Creates a new container from an image |

**Container Inspection Commands**

Sometimes, you need to inspect your containers for quality assurance or troubleshooting purposes. These commands help you get an overview of what different containers are doing:

|  |  |
| --- | --- |
| **Command** | **Explanation** |
| docker ps | Lists all running containers |
| docker -ps -a | Lists all containers |
| docker diff container | Inspects changes to directories and files in the container filesystem |
| docker top container | Shows all running processes in an existing container |
| docker inspect container | Displays low-level information about a container |
| docker logs container | Gathers the logs for a container |
| docker stats container | Shows container resource usage statistics |

**Manage Images Commands**

Some of the most common image management commands include:

|  |  |
| --- | --- |
| **Command** | **Explanation** |
| docker image ls | Lists images |
| docker image rm mysql | Removes an image |
| docker tag image tag | Tags an image |
| docker history image | Displays the image history |
| docker inspect image | Displays low-level information about an image |

**Run Commands**

Docker uses the run command to create containers from provided images. The default syntax for this command looks like this:

docker run [options] image

         [arg...]

After the default syntax, use one of the following flags:

|  |  |
| --- | --- |
| **Flag** | **Explanation** |
| --detach , -d | Runs a container in the background and prints the container ID |
| --env , -e | Sets environment variables |
| --hostname , -h | Sets a hostname to a container |
| --label , -l | Creates a meta data label for a container |
| --name | Assigns a name to a container |
| --network | Connects a container to a network |
| --rm | Removes container when it stops |
| --read-only | Sets the container filesystem as read-only |
| --workdir , -w | Sets a working directory in a container |

**Registry Commands**

If you need to interact with Docker Hub, use the following commands:

|  |  |
| --- | --- |
| **Command** | **Explanation** |
| docker login | Logs in to a registry |
| docker logout | Logs out from a registry |
| docker pull mysql | Pulls an image from a registry |
| docker push repo/ rhel-httpd:latest | Pushes an image to a registry |
| docker search term | Searches Docker Hub for images with the specified term |

**Service Commands**

Manage all Docker services with these basic commands:

|  |  |
| --- | --- |
| **Command** | **Explanation** |
| docker service ls | Lists all services running in a swarm |
| docker stack services stackname | Lists all running services |
| docker service ps servicename | Lists the tasks of a service |
| docker service update servicename | Updates a service |
| docker service create image | Creates a new service |
| docker service scale servicename=10 | Scales one or more replicated services |
| docker service logs stackname servicename | Lists all service logs |

**Network Commands**

If you need to interact with the Docker network, use one of the following commands:

|  |  |
| --- | --- |
| **Command** | **Explanation** |
| docker network create networkname | Creates a new network |
| docker network rm networkname | Removes a specified network |
| docker network ls | Lists all networks |
| docker network connect networkname container | Connects a container to a network |
| docker network disconnect networkname container | Disconnects a container from a network |
| docker network inspect networkname | Displays detailed information about a network |