**Docker Client and Docker Server:**

when you install Docker Desktop on your machine, it includes the Docker client as part of the installation. Docker Desktop is a comprehensive software package that provides both the Docker client and the Docker server (also known as the Docker Engine) in a single package. This makes it easy for you to work with Docker containers and manage containerized applications on your local development environment.

Here's what each component does:

1. **Docker Client:** The Docker client is the command-line tool that you use to interact with Docker. It allows you to build, run, manage, and interact with containers. When you run Docker commands in your terminal, you are using the Docker client.
2. **Docker Server (Docker Engine):** The Docker server, often referred to as the Docker Engine, is the core component that manages containers on your system. It is responsible for creating, running, and stopping containers based on the commands you issue through the Docker client.

Docker Desktop bundles both of these components together, making it easy for developers to work with Docker containers on macOS and Windows operating systems. Once Docker Desktop is installed, you can use the Docker client to interact with containers and images on your local system.

Keep in mind that if you later decide to work with Docker on a Linux system, you will need to install the Docker client and Docker server separately, as Linux distributions typically provide Docker as separate packages that you can install via package managers like **apt**, **yum**, or **dnf**.

Top of Form

**Install the Docker client**

To install   
To install the Docker client (also known as the Docker CLI) on your system without installing the full Docker Desktop or Docker Engine, you can follow these instructions. The Docker CLI is available for various operating systems, including Linux, macOS, and Windows.

### Installing Docker CLI on Linux:

1. **Ubuntu/Debian:**

You can install the Docker CLI on Ubuntu and Debian-based systems using the following commands:

bashCopy code

sudo apt update sudo apt install docker.io

1. **CentOS/RHEL:**

For CentOS and Red Hat Enterprise Linux (RHEL), you can use the following commands:

bashCopy code

sudo yum install docker

1. **Fedora:**

If you're using Fedora, you can use **dnf**:

bashCopy code

sudo dnf install docker

### Installing Docker CLI on macOS:

On macOS, you can install the Docker CLI using Homebrew, which is a popular package manager for macOS:

1. **Install Homebrew (if not already installed):**

If you don't have Homebrew installed, you can install it by running this command:

bashCopy code

/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install.sh)"

1. **Install Docker CLI:**

Once Homebrew is installed, you can install the Docker CLI with this command:

bashCopy code

brew install --cask docker

### Installing Docker CLI on Windows:

On Windows, you can install the Docker CLI as part of the Docker Desktop installation. However, if you want to install the Docker CLI separately, you can use the Windows Subsystem for Linux (WSL) and follow these steps:

1. **Install WSL:**

If you don't already have WSL set up on your Windows machine, you can follow the official Microsoft documentation to install WSL: [Install Windows Subsystem for Linux (WSL)](https://docs.microsoft.com/en-us/windows/wsl/install)

1. **Install Docker CLI in WSL:**

Once WSL is set up, you can install the Docker CLI within your WSL distribution of choice (e.g., Ubuntu) by running the following commands inside the WSL terminal:

bashCopy code

sudo apt update sudo apt install docker.io

This will install the Docker CLI within your WSL environment.

After you've installed the Docker CLI on your system, you can start using it to interact with Docker containers and images by running Docker commands in your terminal or command prompt. Remember that the Docker CLI communicates with a Docker server, so ensure that you have access to a Docker server (either locally or remotely) to execute Docker commands successfully.

**Docker Registry**

The Docker registry is a central repository that stores and manages Docker images.

## Docker Commands Cheat Sheet

Now that you know how Docker functions, let’s look at some of the most popular Docker command examples.

### 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 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**](https://www.hostinger.in/tutorials/docker-start-a-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 (command) (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 |