

List of Docker Commands

Docker Container Commands

In this section you will find the most important commands related to the lifecycle of Docker containers.

Create a container (without starting it):

```
docker create [IMAGE]
```

Rename an existing container:

```
docker rename [CONTAINER_NAME] [NEW_CONTAINER_NAME]
```

Run a command in a new container:

```
docker run [IMAGE] [COMMAND]
```

`docker run --rm [IMAGE]` – removes a container after it exits.

`docker run -td [IMAGE]` – starts a container and keeps it running.

`docker run -it [IMAGE]` – starts a container, allocates a pseudo-TTY connected to the container's stdin, and creates an interactive bash shell in the container.

`docker run -it-rm [IMAGE]` – creates, starts, and runs a command inside the container. Once it executes the command, the container is removed.

Delete a container (if it is not running):

```
docker rm [CONTAINER]
```

Update the configuration of one or more containers:

```
docker update [CONTAINER]
```

Visit our detailed guide on [how to update Docker container and image to the latest version](#) for more information.

Starting and Stopping Containers

The following commands show you how to [start and stop](#) processes in a particular container.

Start a container:

```
docker start [CONTAINER]
```

Stop a running container:

```
docker stop [CONTAINER]
```

Stop a running container and start it up again:

```
docker restart [CONTAINER]
```

Pause processes in a running container:

```
docker pause [CONTAINER]
```

Unpause processes in a running container:

```
docker unpause [CONTAINER]
```

Block a container until others stop (after which it prints their exit codes):

```
docker wait [CONTAINER]
```

Kill a container by sending a SIGKILL to a running container:

```
docker kill [CONTAINER]
```

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

```
docker attach [CONTAINER]
```



Note: If you are still unsure of how Docker images and containers differ, you may want to check out the article on [Images vs Containers](#).

Docker Image Commands

Below you will find all the necessary commands for working with [Docker images](#).

Create an image from a Dockerfile:

```
docker build [URL]
```

`docker build -t` – builds an image from a Dockerfile in the current directory and tags the image

Pull an image from a registry:

```
docker pull [IMAGE]
```

Push an image to a registry:

```
docker push [IMAGE]
```

Create an image from a tarball:

```
docker import [URL/FILE]
```

Create an image from a container:

```
docker commit [CONTAINER] [NEW_IMAGE_NAME]
```

Remove an image:

```
docker rmi [IMAGE]
```

Load an image from a tar archive or stdin:

```
docker load [TAR_FILE/STDIN_FILE]
```

Save an image to a tar archive, streamed to STDOUT with all parent layers, tags, and versions:

```
docker save [IMAGE] > [TAR_FILE]
```

Docker Commands for Container and Image Information

Once you set up your containers, you will need to know how to get all the important information for managing them. The following commands will provide details on images and containers on your system.

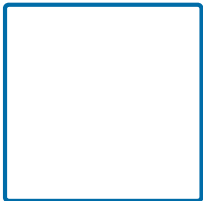
List running containers:

```
docker ps
```

`docker ps -a` – lists both running containers and ones that have stopped

List the logs from a running container:

```
docker logs [CONTAINER]
```



Note: Refer to our post [Docker Container Logs: Commands & Best Practices](#) for a more in-depth look at managing Docker container logs.

List low-level information on Docker objects:

```
docker inspect [OBJECT_NAME/ID]
```

List real-time events from a container:

```
docker events [CONTAINER]
```

Show port (or specific) mapping for a container:

```
docker port [CONTAINER]
```

Show running processes in a container:

```
docker top [CONTAINER]
```

Show live resource usage statistics of containers:

```
docker stats [CONTAINER]
```

Show changes to files (or directories) on a filesystem:

```
docker diff [CONTAINER]
```

List all images that are locally stored with the docker engine:

```
docker image ls
```

Show the history of an image:

```
docker history [IMAGE]
```

Networks

One of the most valuable features of Docker software is the ability to [connect containers to each other](#) and to other non-Docker workloads. This section covers network-related commands.

List networks:

```
docker network ls
```

Remove one or more networks:

```
docker network rm [NETWORK]
```

Show information on one or more networks:

```
docker network inspect [NETWORK]
```

Connects a container to a network:

```
docker network connect [NETWORK] [CONTAINER]
```

Disconnect a container from a network:

```
docker network disconnect [NETWORK] [CONTAINER]
```