# Build an image

When we build an image we need to either run the ‘docker build’ command from a folder which contains a Dockerfile and all the other files needed or use this command:

* Docker build -f docker\_file\_path -t image\_name build-context-directory-path

Note:

* Build context directory is a directory which contains the Dockerfile and all the other files which will be used in the Dockerfile (which we will copy into an image using the COPY or ADD instructions).

Where:

* **docker\_file\_path** - A path to the Dockerfile which will be used for building an image. We don’t need to provide it if the Dockerfile is called ‘Dockerfile’ and it is located in the build context directory.
* **build-context-directory-path** – A path to the build context directory.

When we run for example ‘COPY’ instruction in the Dockerfile then we need to provide there

# Get access to a container’s terminal

## Get access to a running container’s terminal

Docker exec runs a new command inside of a running container. For example if we run:

* Docker exec -it <container\_name\_or\_id> /bin/bash

In simple words it gives us access to the container’s terminal.

To be more precise it starts a new bash session inside of a container and allocates a pseudo terminal which we can use from our computer’s terminal.

## Run a container and get immediately access to its terminal

docker run -it <image-name> /bin/bash

# Remove containers

Remove one container: docker rm <container-ID>

Remove all containers: docker rm $(docker ps -aq)

# Remove images

Remove one image: docker rmi <image-name-or-ID>

Remove all the images: docker rmi $(docker image ls -aq)

Remove dangling images (untagged build cache layers):

* docker image prune

Remove all unused images (untagged build cache layers):

* docker image prune -a

Remove dangling build cache:

* docker builder prune

Remove all build cache:

* docker builder prune --all

# Remove volumes

docker volume rm $(docker volume ls -q)

# Remove netwoks (except for the default one)

docker network rm $(docker network ls -q)

# Remove build cache

docker builder prune -a

# Remove everything

docker system prune --all --volumes