



# Cheatsheet for Docker CLI

## Run a new Container

Start a new Container from an Image

```
docker run IMAGE  
docker run nginx
```

...and assign it a name

```
docker run --name CONTAINER IMAGE  
docker run --name web nginx
```

...and map a port

```
docker run -p HOSTPORT:CONTAINERPORT IMAGE  
docker run -p 8080:80 nginx
```

...and map all ports

```
docker run -P IMAGE  
docker run -P nginx
```

...and start container in background

```
docker run -d IMAGE  
docker run -d nginx
```

...and assign it a hostname

```
docker run --hostname HOSTNAME IMAGE  
docker run --hostname srv nginx
```

...and add a dns entry

```
docker run --add-host HOSTNAME:IP IMAGE
```

...and map a local directory into the container

```
docker run -v HOSTDIR:TARGETDIR IMAGE  
docker run -v ~/.usr/share/nginx/html nginx
```

...but change the entrypoint

```
docker run -it --entrypoint EXECUTABLE IMAGE  
docker run -it --entrypoint bash nginx
```

## Manage Containers

Show a list of running containers

```
docker ps
```

Show a list of all containers

```
docker ps -a
```

Delete a container

```
docker rm CONTAINER  
docker rm web
```

Delete a running container

```
docker rm -f CONTAINER  
docker rm -f web
```

Delete stopped containers

```
docker container prune
```

Stop a running container

```
docker stop CONTAINER  
docker stop web
```

Start a stopped container

```
docker start CONTAINER  
docker start web
```

Copy a file from a container to the host

```
docker cp CONTAINER:SOURCE TARGET  
docker cp web:/index.html index.html
```

Copy a file from the host to a container

```
docker cp TARGET CONTAINER:SOURCE  
docker cp index.html web:/index.html
```

Start a shell inside a running container

```
docker exec -it CONTAINER EXECUTABLE  
docker exec -it web bash
```

Rename a container

```
docker rename OLD_NAME NEW_NAME  
docker rename 096 web
```

Create an image out of container

```
docker commit CONTAINER  
docker commit web
```

## Manage Images

Download an image

```
docker pull IMAGE[:TAG]  
docker pull nginx
```

Upload an image to a repository

```
docker push IMAGE  
docker push myimage:1.0
```

Delete an image

```
docker rmi IMAGE
```

Show a list of all Images

```
docker images
```

Delete dangling images

```
docker image prune
```

Delete all unused images

```
docker image prune -a
```

Build an image from a Dockerfile

```
docker build DIRECTORY  
docker build .
```

Tag an image

```
docker tag IMAGE NEWIMAGE  
docker tag ubuntu ubuntu:18.04
```

Build and tag an image from a Dockerfile

```
docker build -t IMAGE DIRECTORY  
docker build -t myimage .
```

Save an image to .tar file

```
docker save IMAGE > FILE  
docker save nginx > nginx.tar
```

Load an image from a .tar file

```
docker load -i TARFILE  
docker load -i nginx.tar
```

## Info & Stats

Show the logs of a container

```
docker logs CONTAINER  
docker logs web
```

Show stats of running containers

```
docker stats
```

Show processes of container

```
docker top CONTAINER  
docker top web
```

Show installed docker version

```
docker version
```

Get detailed info about an object

```
docker inspect NAME  
docker inspect nginx
```

Show all modified files in container

```
docker diff CONTAINER  
docker diff web
```

Show mapped ports of a container

```
docker port CONTAINER  
docker port web
```

## Container Management CLIs

# Container management commands

command	description
<code>docker create image [ command ]</code> <code>docker run image [ command ]</code>	create the container = <code>create</code> + <code>start</code>
<code>docker start container...</code> <code>docker stop container...</code> <code>docker kill container...</code> <code>docker restart container...</code>	start the container graceful <sup>2</sup> stop kill (SIGKILL) the container = <code>stop</code> + <code>start</code>
<code>docker pause container...</code> <code>docker unpause container...</code>	suspend the container resume the container
<code>docker rm [ -f<sup>3</sup> ] container...</code>	destroy the container

---

<sup>2</sup>send SIGTERM to the main process + SIGKILL 10 seconds later

<sup>3</sup>-f allows removing running containers (= `docker kill` + `docker rm`)

### Inspecting the container

command	description
<code>docker ps</code>	list running containers
<code>docker ps -a</code>	list all containers
<code>docker logs [ -f<sup>6</sup> ] container</code>	show the container output ( <i>stdout+stderr</i> )
<code>docker top container [ ps options ]</code>	list the processes running inside the containers
<code>docker diff container</code>	show the differences with the image (modified files)
<code>docker inspect container...</code>	show low-level infos (in json format)

### Interacting with the container

command	description
<code>docker attach container</code>	attach to a running container (stdin/stdout/stderr)
<code>docker cp container:path hostpath </code> <code>docker cp hostpath - container:path</code>	copy files from the container copy files into the container
<code>docker export container</code>	export the content of the container (tar archive)
<code>docker exec container args...</code>	run a command in an existing container ( <b>useful</b> for debugging)
<code>docker wait container</code>	wait until the container terminates and return the exit code
<code>docker commit container image</code>	commit a new docker image (snapshot of the container)

# Image Management Commands

# Image management commands

command	description
<code>docker images</code> <code>docker history image</code> <code>docker inspect image...</code>	list all local images show the image history (list of ancestors) show low-level infos (in json format)
<code>docker tag image tag</code>	tag an image
<code>docker commit container image</code> <code>docker import url - [tag]</code>	create an image (from a container) create an image (from a tarball)
<code>docker rmi image...</code>	delete images

## Image transfer commands

### Using the registry API

<code>docker pull repo[:tag]...</code>	pull an image/repo from a registry
<code>docker push repo[:tag]...</code>	push an image/repo from a registry
<code>docker search text</code>	search an image on the official registry
<code>docker login ...</code>	login to a registry
<code>docker logout ...</code>	logout from a registry

### Manual transfer

<code>docker save repo[:tag]...</code>	export an image/repo as a tarball
<code>docker load</code>	load images from a tarball
<code>docker-ssh<sup>10</sup> ...</code>	proposed script to transfer images between two daemons over ssh



## Builder Main Commands

## Builder main commands

command	description
<b>FROM</b> <i>image scratch</i>	base image for the build
<b>MAINTAINER</b> <i>email</i>	name of the maintainer (metadata)
<b>COPY</b> <i>path dst</i>	copy <i>path</i> from the context into the container at location <i>dst</i>
<b>ADD</b> <i>src dst</i>	same as <b>COPY</b> but untar archives and accepts http urls
<b>RUN</b> <i>args. . .</i>	run an arbitrary command inside the container
<b>USER</b> <i>name</i>	set the default username
<b>WORKDIR</b> <i>path</i>	set the default working directory
<b>CMD</b> <i>args. . .</i>	set the default command
<b>ENV</b> <i>name value</i>	set an environment variable

# The Docker CLI

## Manage images

`docker build`

```
docker build [options] .  
  -t "app/container_name"    # name
```

Create an `image` from a Dockerfile.

`docker run`

```
docker run [options] IMAGE  
  # see `docker create` for options
```

Run a command in an `image`.

## Manage containers

`docker create`

```
docker create [options] IMAGE
-a, --attach                # attach stdout/err
-i, --interactive           # attach stdin (interactive)
-t, --tty                   # pseudo-tty
    --name NAME             # name your image
-p, --publish 5000:5000     # port map
    --expose 5432           # expose a port to linked containers
-P, --publish-all         # publish all ports
    --link container:alias # linking
-v, --volume `pwd`:/app     # mount (absolute paths needed)
-e, --env NAME=hello       # env vars
```

## Example

```
$ docker create --name app_redis_1 \
  --expose 6379 \
  redis:3.0.2
```

Create a `container` from an `image`.

`docker exec`

```
docker exec [options] CONTAINER COMMAND
  -d, --detach      # run in background
  -i, --interactive # stdin
  -t, --tty         # interactive
```

## Example

```
$ docker exec app_web_1 tail logs/development.log
$ docker exec -t -i app_web_1 rails c
```

Run commands in a `container`.

`docker start`

```
docker start [options] CONTAINER
  -a, --attach      # attach stdout/err
  -i, --interactive # attach stdin

docker stop [options] CONTAINER
```

Start/stop a `container`.

`docker ps`

```
$ docker ps
$ docker ps -a
$ docker kill $ID
```

Manage `container`s using `ps/kill`.

# Images

`docker images`

```
$ docker images
REPOSITORY    TAG       ID
ubuntu        12.10     b750fe78269d
me/myapp       latest    7b2431a8d968
```

```
$ docker images -a  # also show intermediate
```

Manages `images`.

`docker rmi`

```
docker rmi b750fe78269d
```

Deletes `images`.

## Also see

- [Getting Started](#) (*docker.io*)

# Dockerfile

## Inheritance

```
FROM ruby:2.2.2
```

## Variables

```
ENV APP_HOME /myapp  
RUN mkdir $APP_HOME
```

## Initialization

```
RUN bundle install
```

```
WORKDIR /myapp
```

```
VOLUME ["/data"]  
# Specification for mount point
```

```
ADD file.xyz /file.xyz  
COPY --chown=user:group host_file.xyz /path/container_file.xyz
```

## Onbuild

```
ONBUILD RUN bundle install  
# when used with another file
```

## Commands

```
EXPOSE 5900  
CMD ["bundle", "exec", "rails", "server"]
```

## Entrypoint

```
ENTRYPOINT ["executable", "param1", "param2"]  
ENTRYPOINT command param1 param2
```

Configures a container that will run as an executable.

```
ENTRYPOINT exec top -b
```

This will use shell processing to substitute shell variables, and will ignore any `CMD` or `docker run` command line arguments.

## Metadata

```
LABEL version="1.0"
```

```
LABEL "com.example.vendor"="ACME Incorporated"  
LABEL com.example.label-with-value="foo"
```

```
LABEL description="This text illustrates \  
that label-values can span multiple lines."
```



## See also

- <https://docs.docker.com/engine/reference/builder/>

# docker-compose

## Basic example

```
# docker-compose.yml
version: '2'

services:
  web:
    build: .
    # build from Dockerfile
    context: ./Path
    dockerfile: Dockerfile
    ports:
      - "5000:5000"
    volumes:
      - ./code
  redis:
    image: redis
```

## Commands

```
docker-compose start
docker-compose stop
```

```
docker-compose pause
docker-compose unpause
```

```
docker-compose ps
docker-compose up
docker-compose down
```

## Reference

### Building

```
web:
  # build from Dockerfile
  build: .
```

```
# build from custom Dockerfile
build:
  context: ./dir
  dockerfile: Dockerfile.dev
```

```
# build from image
image: ubuntu
image: ubuntu:14.04
image: tutum/influxdb
image: example-registry:4000/postgresql
image: a4bc65fd
```

## Ports

```
ports:
  - "3000"
  - "8000:80" # guest:host
```

```
# expose ports to linked services (not to host)
expose: ["3000"]
```

## Commands

```
# command to execute
command: bundle exec thin -p 3000
command: [bundle, exec, thin, -p, 3000]
```

```
# override the entrypoint
entrypoint: /app/start.sh
entrypoint: [php, -d, vendor/bin/phpunit]
```

## Environment variables

```
# environment vars
environment:
  RACK_ENV: development
environment:
  - RACK_ENV=development
```

```
# environment vars from file
env_file: .env
env_file: [.env, .development.env]
```

## Dependencies

```
# makes the `db` service available as the hostname `database`
# (implies depends_on)
links:
  - db:database
  - redis
```

```
# make sure `db` is alive before starting
depends_on:
  - db
```

## Other options

```
# make this service extend another
extends:
  file: common.yml # optional
  service: webapp
```

```
volumes:
  - /var/lib/mysql
  - ./_data:/var/lib/mysql
```

# Advanced features

## Labels

```
services:
  web:
    labels:
      com.example.description: "Accounting web app"
```

## DNS servers

```
services:
  web:
    dns: 8.8.8.8
    dns:
      - 8.8.8.8
      - 8.8.4.4
```

## Devices

```
services:
  web:
    devices:
      - "/dev/ttyUSB0:/dev/ttyUSB0"
```

## External links

```
services:
  web:
    external_links:
      - redis_1
      - project_db_1:mysql
```

## Hosts

```
services:
  web:
    extra_hosts:
      - "somehost:192.168.1.100"
```

## services

To view list of all the services running in swarm

```
docker service ls
```

To see all running services

```
docker stack services stack_name
```

to see all services logs

```
docker service logs stack_name service_name
```

To scale services quickly across qualified node

```
docker service scale stack_name_service_name=replicas
```

## clean up

To clean or prune unused (dangling) images

```
docker image prune
```

To remove all images which are not in use containers , add - a

```
docker image prune -a
```

To Purge your entire system

```
docker system prune
```

To leave swarm

```
docker swarm leave
```

To remove swarm ( deletes all volume data and database info)

```
docker stack rm stack_name
```

To kill all running containers

```
docker kill $(docekr ps -q )
```

## Contributor -

[Sangam biradar](#) - Docker Community Leader

---

## Practice Docker Tutorial

[free Ubuntu VM](#)