Installation

For mac and Windows:

<https://www.docker.com/products/docker-desktop>

# CONTAINERS

## Run a container

Running a container

#### Usage

Docker container run <image-name>

#### Examples

* Docker **container** run nginx

## Listing containers

Gives info about the containers

* List
  + Docker container ls
  + Docker container ls -a
* Process status
  + Docker container ps
  + Docker container ps -a

## Remove containers

Removing unused container

#### Usage:

Docker **container** rm <container-name or container-id>

#### Examples

Docker **container** rmmy-container

## Image with tags

Tags, versions, and types of images

#### Usage

Docker container run <image:tag>

#### Examples

* Docker **container** run **nginx**:1.16
* Docker **container** run **nginx**:alpine
* Docker **container** run **nginx**:stable

## Naming an image

Naming image to ease operations

#### Usage

Docker container run --name <give-a-name> <image-name >

#### Examples

* Docker **container** run --name test **nginx**

## Interactive terminal

Start the container and run a command

#### Usage

Docker container run -it <image > <command>

#### Examples

* Docker **container** run -it **nginx** bash
* Docker **container** run -it **nginx** echo "text"

## Daemon Option

Runs an image in background

#### Usage:

Docker **container** run -d <image>

## Giving a port

Opens a port from container and pairs with host port

#### Usage:

Docker container run -p <host-port>:<container-port> <image >

#### Examples

* Docker **container** run -p 8080:80 **nginx**
* Docker **container** run -p 3306:3306 **mysql**

## Inspect

Getting info from the containers, volumes etc

#### Usage:

Docker container inspect <container-id or container-name>

Docker volume inspect <volume-id or volume -name>

## Logs

Getting logs from the container

#### Usage:

Docker **container** logs **mysql**

## Environment variables

Giving environment variables to containers

#### Usage:

Docker **container** run -e <variable-to-pass> <image>

#### Examples

Docker **container** run -v my-data:/var/lib/mysql -e MYSQL\_ALLOW\_EMPTY\_PASSWORD=true --name mydb -d **mysql**

## Container

Docker container run

|  |  |
| --- | --- |
| --rm | remove container automatically after it exits |
| -it | connect the container to terminal |
| --name web | name the container |
| -p 5000:80 | expose port 5000 externally and map to port 80 |
| -v ~/dev:/code | create a host mapped volume inside the container |
| --network=mynet | Runs the container on a specific network |
| alpine:3.4 | the image from which the container is instantiated |
| /bin/sh | the command to run inside the container |

# VOLUMES

Used for persistent data

## Running container with volume

#### Usage:

Bind mount: Docker container run -v <host-path:container-path>

Named Volume: Docker container run -v <volume-name:container-path>

Unnamed Volume: Docker container run -v <container-path>

#### Examples

Docker **container** run -v ${PWD}/db:/var/lib/mysql **mysql**

Docker **container** run -v my-data:/var/lib/mysql **mysql**

Docker **container** run -v /var/lib/mysql **mysql**

## Creating a volume

#### Usage:

Docker volume create <options> <volume>

#### Examples:

Docker **volume** create db\_data

Docker **volume** create --d local --opt type=tmpfs --opt device=tmpfs --opt o=size=100m,uid=1000 dt

# NETWORK

Containers created on and communicate with default network “bridge”

To containers to communicate they need to be in the same network

Docker **network** ls

Docker **network** create my-network

Docker **network** inspect my-network

Running a container in a network

Docker **container** run -d --name nginx\_srv --network my-network **nginx**

Docker **network** (dis)connect my-network nginx\_srv

Giving a container alias to use in network

Docker **container** run -d --net my-network --net-alias search **elasticsearch**

# Images

Read-only template with instructions for creating a Docker container

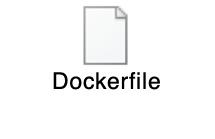
## Creating an image

Creates an image from a file (without extension)

Usage:

Check <https://docs.docker.com/develop/develop-images/dockerfile_best-practices/>

#### Examples:



Create a docker file and execute command:

* Build the image from the docker file
  + Docker build -t my-sample-image:1.0
* Tag the image
  + Docker image tag my-sample-image:1.0 abdullahahci/ my-sample-image:1.0
* Push it to docker hub
  + Docker image push abdullahahci/my-sample-image:1.0

Each command creates a new layer

Combine commands with && or ; in single line

Use commands that can change at the bottom lines

## Stages

Divide dockerfile into stages to benefit from image size and creation time

## More about Image

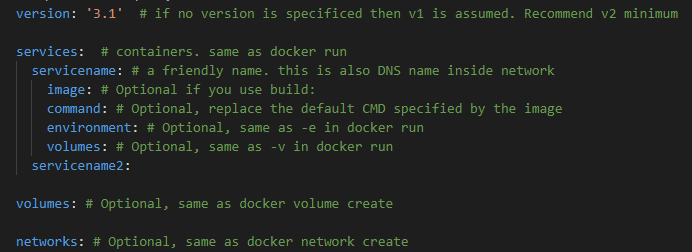
To get info about images, use these commands

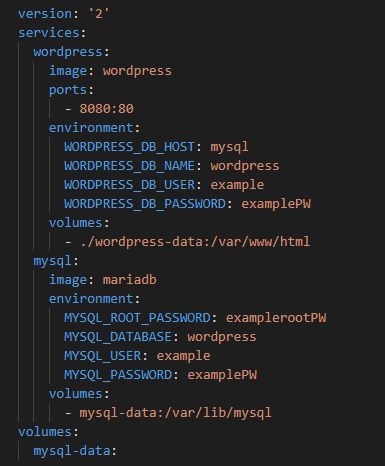
Docker **image** history **nginx**

Docker **image** inspect **nginx**

# DOCKER COMPOSE

YAML formatted





<https://docs.docker.com/compose/compose-file/compose-versioning/>

# DOCKER SWARM

Container orchestrator

## Init

Initializes the swarm mode

Docker **swarm** init

## Create Service

Creates a service, a set of containers

Docker **service** create **alpine** ping 8.8.8.8

Docker **service** create --name my\_service --replicas 3 **alpine** ping 8.8.8.8

## Service info

Get info about services

* List Services
  + Docker service ls
* Get service info
  + Docker service ps my\_service

## Scaling service

Scales up or down a service, changes replica count

Docker **service** update my\_service --replicas 5

## Removing Service

Removes the service and containers under the service

Docker **service** rm my\_service

## Networking

### Overlay

Creates an overlay network to above nodes to help them talk each other (default:ingres)

Docker **network** create --driver overlay my\_network

Docker **service** create --name myd b --network my\_network -e POSTGRES\_PASSWORD=pass **postgres**

## Routing Mesh

Routes incoming requests to nodes, load balances

Docker service create --name search --replicas 3 -p 9200:9200 elasticsearch:5.6.16-alpine

Docker service create -e ES\_JAVA\_OPTS="-Xms512m -Xmx512m" --name search2 --replicas 3 -p 9200:9200 elasticsearch:5.6.16-alpine

## Healthchecks

Docker engine exects command (default:every 30 seconds) and expects 0 if it is ok, 1 otherwise

3 states: starting, healthy, unhealthy

Docker **container** run --health-cmd=”curl -f localhost:9200/\_cluster/health || false” \

--health-interval=5s --health-retries=3 --health-timeout=2s \

--health-start-period=15s **elasticsearch**

Docker **container** run --name p1 -d **postgres**

Docker **container** run --name p2 -d --health-cmd=”pg\_isready -U postgres || exit 1” **postgres**

Run “Docker **container** ls “ to see difference

# DOCKER REGISTRY

To create another registry than dockerhub

Docker container run -d -p 5000:5000 –name registry registry

Docker tag hello-world 127.0.0.1: 5000/hello-world

Docker push 127.0.0.1: 5000/hello-world