

Docker

- [¿Qué es docker?](#)
- [Instalar Docker](#)
- [Instalar redis-docker](#)
- [Docker - Comandos básicos](#)
- [Docker - Instalar MariaDB y phpmyadmin](#)
- [Usar docker sin sudo en linux](#)
- [Ejemplos de Dockerfile](#)
- [docker-machine docker-swarm](#)
- [Traefik - v2](#)

¿Qué es docker?

<https://www.youtube.com/embed/hQgvt-s-AHQ>

¿Qué es docker?

La idea detrás de Docker es crear contenedores ligeros y portables para que las aplicaciones de software puedan ejecutarse en cualquier máquina con Docker instalado, independientemente del sistema operativo que la máquina tenga por debajo.

<https://www.javiergarzas.com/2015/07/que-es-docker-sencillo.html>

Instalar Docker

Linux - Ubuntu, Linux Mint, . . .

```
sudo apt install docker.io
```

Mac OS

<https://hub.docker.com/editions/community/docker-ce-desktop-mac>

Windows

<https://hub.docker.com/editions/community/docker-ce-desktop-windows>

Docker compose

<https://docs.docker.com/compose/install/>

Instalar redis-docker

```
docker run --name redis -p 6379:6379 -d redis
```

```
# dentro del contenedor configurar password
```

```
#para entrar al contenedor
```

```
docker exec -ti redis /bin/bash
```

```
# luego
```

```
redis-cli
```

```
CONFIG SET requirepass "yourpassword"
```

```
AUTH yourpassword
```

Docker - Comandos básicos

Para crear imagen a partir del Dockerfile del directorio actual

```
docker build -t nombre_imagen .
```

Este comando creará un contenedor a partir de la imagen pasada por parámetro

-p para especificar por cual puerto será accesible desde localhost, -p [puerto_localhost]:[puerto_contenedor]

```
docker run -p 9080:9080 --name nombre_contenedor -d nombre_imagen
```

Lista los contenedores activos

```
docker ps
```

Lista los contenedores activos e inactivos

```
docker ps -a
```

Iniciar contenedor

```
docker start [nombre_contenedor | id_contenedor]
```

Reiniciar contenedor

```
docker restart [nombre_contenedor | id_contenedor]
```

Detener contenedor

```
docker stop [nombre_contenedor | id_contenedor]
```

Remover contenedor

```
docker rm [nombre_contenedor | id_contenedor]
```

Exportar imagen docker a un .tar

```
docker save imagen > imagen.tar
```

Para iniciar todos los contenedores y configuraciones especificadas en el docker-compose.yml

```
docker-compose up -d
```

Entrar a la linea de comando de un contenedor especifico

```
docker exec -ti container_id /bin/bash
```

```
docker run -i -t --rm image bash
```

```
docker inspect -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' mariadbtest
```

```
docker commit upbeat_panini testredis:example
```

```
#reiniciar todos los contenedores
```

```
docker restart $(docker ps -a -q)
```

```
#--- Docker compose
```

```
docker-compose down --rmi all
```

```
# --no-deps - Don't start linked services. --build - Build images before starting containers.
```

```
docker-compose up -d --no-deps --build <service_name>
```

```
# to remove all the images which are not used by existing containers
```

```
docker system prune -a
```

Docker - Instalar MariaDB y phpmyadmin

```
# Crear network (red), para poder conectar ambos contenedores
docker network create mysql-network

# Correr mariadb
docker run -p 3306:3306 --name mysql --net=mysql-network -e MYSQL_ROOT_PASSWORD=secret -d mariadb
docker run -p 3306:3306 --name mysql --net=mysql-network -e MYSQL_ROOT_PASSWORD=secret -d mysql:5.7

# Correr phpmyadmin
docker run --name phpmyadmin \
    --net=mysql-network \
    -e MYSQL_ROOT_PASSWORD=secret \
    -e PMA_HOST="mysql" \
    -e PMA_PORT=3306 \
    -p 8080:80 \
    -d phpmyadmin/phpmyadmin
```

Después de haber corrido los comandos anteriores, la base de datos MariaDB estará disponible en el puerto 3306 del localhost, y phpmyadmin estará corriendo en el puerto 8080 del localhost

Usar docker sin sudo en linux

```
# Crear grupo docker
```

```
sudo groupadd docker
```

```
# Agregar usuario actual al grupo docker
```

```
sudo usermod -aG docker $USER
```

```
# Activar nuevos cambios
```

```
newgrp docker
```

```
# Ya puedes usar docker sin sudo!
```


Ejemplos de Dockerfile

Para deploy de node.js app

```
FROM node:10

# Create app directory
WORKDIR /usr/src/app

# Install app dependencies
# A wildcard is used to ensure both package.json AND package-lock.json are copied
# where available (npm@5+)
COPY package*.json ./

RUN npm install
# If you are building your code for production
# RUN npm ci --only=production

# Bundle app source
COPY . .

EXPOSE 8080

CMD [ "npm", "start" ]
```

Para deploy de django app, las dependencias deben estar en el Pipfile

```
# Pull base image
FROM python:3.7-slim

# Set environment variables
ENV PYTHONUNBUFFERED 1
ENV PYTHONPATH /code

# Set work directory
WORKDIR /code
```

Install dependencies

RUN pip install pipenv

COPY Pipfile Pipfile.lock /code/

RUN pipenv install --system

Copy project

COPY . /code/

CMD ["python", "manage.py", "runserver" ,"0.0.0.0:8000"]

docker-machine docker-swarm

```
#mac os install
```

```
brew install docker-machine
```

```
docker-machine ls
```

```
docker-machine create --driver digitalocean --digitalocean-access-token 5a07a17... docker-app-machine
```

```
eval $(docker-machine env docker-app-machine)
```

```
#connect by ssh to host
```

```
docker-machine ssh swarm-node
```

```
docker swarm init
```

```
docker swarm join
```

```
docker swarm leave
```

Traefik - v2

examples - tutorial: <https://github.com/DoTheEvo/Traefik-v2-examples>