

# docker (altran)

[devops](#), [cursos](#)

Ponente: **Daniel Sanchez**

## próximos cursos

- vagrant = levantar MV en entorno de desarrollo
- ansible = configurar una de esas máquinas
- jenkins = continuous integration
- kubernetes = orquestador de contenedores : «hypervisor»
- openshift = redhat + kubernetes

## resumen comandos

### informativos

- `docker info`
- `docker ps`
  - `-a`
  - `-q`
- `docker search`
- `docker inspect <id>` : vuela JSON con información
  - `docker inspect <id> | grep IPAddress`

### contenedores

- `docker ps`
  - `-a`
  - `-q`
- `docker run <imagen> [<comando>]`
  - `-it` : tty interactive
  - `-rm` : al parar el contenedor, elimina el contenedor (para no ir dejando un rastro de contenedores)
  - `-d` : detach (el proceso no se queda «colgado» ejecutando el contenedor)... lanzar contenedor en background
  - `-p 80:80` : mapea el puerto host:contenedor, en todos los interfaces
  - `-p 127.0.0.1:80:80` : el contenedor solo será accesible desde 127.0.0.1
  - `-P` : mapea el puerto en el que está escuchando el contenedor a un puerto aleatorio del equipo
  - `--name <nombre_contenedor>`
- `docker stop <container_id>`
- `docker exec [-it] <container_id> [<comando>]`
  - `bash` o `sh` serían comandos válidos si están instalados en el contenedor
- `docker run -it <container_id>` : al salir del contenedor pasa a **STOPPED**
- `docker rm <container_id>`
  - no se puede eliminar containers en ejecución
  - `docker rm $(docker ps -aq)` : elimina todos los contenedores
- `docker commit <id_contenedor> [REPOSITORY[:TAG]]` : crea imagen de un contenedor

## imágenes

- `docker pull debian[:<tag>]`
- `docker images`
  - `-f «dangling=true»`: filtra lista imágenes no tageadas
  - `-filter «label=<clave>»`: filtra por labels (a nivel de imagen, se ven siempre)
  - `-filter ...?`
  - `-format «{{.ID}}:{{.Repository}}»`: formato de salida (escrito en Go, plantilla)
- `docker rmi <imagen_id>`: borrar una imagen
- `docker tag <imagen_id> <nuevo_nombre>` asignar un nombre a una imagen sin tagear, copiar si ya estaba tageada
- `docker save -o <destino> <imagen>:<tag>` guarda en <destino> una copia «física» de la imagen
  - NO recomendado!
- `docker load -i <imagen_disco>`: importa la imagen
- `docker history <imagen>`: muestra las capas de una imagen y el tamaño de cada una

## Dockerfile

### restaurar

#### stage

- `git reset` : elimina **stage** y lo pasa a **working**
  - `git reset <file>`

```
git reset --hard: elimina todas las modificaciones del stage y working (menos untracked)
```

#### working area

```
git checkout -- <file>: Elimina los cambios en el fichero del working copy
```

```
git checkout <folder>: Elimina los cambios en el directorio del working copy
```

```
git checkout .: Elimina los cambios en TODO el working copy
```

#### fichero untracked

```
git clean -f: Elimina los ficheros untracked
```

```
git clean -fd: Elimina los ficheros untracked de raiz y directorios
```

```
git clean -id: clean interactivo
```

## Curso

- [introducción](#)
- [instalación](#)
- [trabajando con contenedores](#)
- [imágenes](#)
- [Dockerfile](#)
- [volumes](#)
- [alpine](#)
- [multi-stage](#)
- [dockerhub](#)
- [onbuild](#)
- [build](#)
- [exec](#)
- [run](#)
- [update](#)
- [logs](#)
- [attach](#)
- [rename](#)
- [tag](#)
- [cp](#)
- [diff](#)
- [top](#)
- [stats](#)
- [export & import](#)
- [volumes](#)
- [network](#)
- [dind \(docker inside docker\)](#)
- [portainer](#)
- [docker-compose](#)

## ejemplos

- [compartir archivos: nginx-autoindex](#) → `docker run --rm -p 80:80 -v $PWD:/usr/share/nginx/html jrelva/nginx-autoindex`

From:

<https://miguelangel.torresegea.es/wiki/> - **miguel angel torres egea**

Permanent link:

<https://miguelangel.torresegea.es/wiki/info:cursos:altran:docker?rev=1531846347>

Last update: **17/07/2018 09:52**

