IMÁGENES DE DOCKER

docker images → listar imágenes descargadas

La información que se nos muestra:

- **REPOSITORY:** Nombre de la imagen en el repositorio. Por ejemplo: *mysql.*
- **TAG:** Versión de la imagen que hemos descargado.
- **IMAGE ID:** Un identificador que es único para cada imagen.
- **CREATED:** Hace cuánto se creó la imagen.
- **SIZE:** Tamaño de la imagen.

docker pull → actualizar una determina pareja imagen:versión a su última actualización.

Sólo tendré que hacer docker pull con el mismo imagen:version

```
# Suponiendo que ya teníamos previamente la versión descargada. Actualiza la versión mysql:5.7

> docker pull mysql:5.7
```

Me permite bajar todas las versiones de una imagen de una sola vez. Esto puede ser peligroso si una imagen tiene muchas versiones disponibles. Lo conseguiremos con la opción -a o --all-tags

```
# Descargamos todas las versiones de la imagen php. CON MUCHO CUIDADO, NO PROBAR

> docker pull -a php o docker pull --all-tags php
```

Tiene otras opciones que son útiles a nivel de usuario, de momento nos quedaremos con aquella que no me muestra toda la información de las capas.

```
# No muestro la información de las capas al descargarse

> docker pull -q httpd  o docker pull --quiet
```

docker rmi / docker image rm → borrado de imágenes

```
# Borrado de la imagen mysql:8.0.22

> docker rmi mysql:8.0.22

# Borrado de una imagen usando su IMAGE ID

> docker rmi dd7265748b5d
```

```
# Usando la orden docker image rm y el nombre

> docker image rm mysq:8.0.22

# Usando la orden docker image rm y el IMAGE ID

> docker image rm dd7265748b5d

# Borrado de dos imágenes (o varias)  a la vez. Puedes usar nombre e IMAGE ID

> docker rmi mysql:8.0.22 mysql:5.7
```

!!!!NO PODEMOS BORRAR UNA IMAGEN SI YA TENEMOS UN CONTENEDOR QUE ESTÁ USÁNDOLA.
Para poder forzar ese borrado, usamos la opción -f o --force.

```
# Borra la imagen httpd (Apache latest) aunque hubiera contenedores que estuvieran usando esa imagen.

> docker rmi -f httpd
```

docker image prune [options] → **eliminar imágenes más fácilmente**

- **-a o --all** para borrar todas las imágenes que no están siendo usadas por contenedores
- **-f o --force** para que no nos solicite confirmación. Es una operación que puede borrar muchas imágenes de una tacada y debemos ser cuidadosos. Os recomiendo no usar esta opción.
- **--filter** para especificar ciertos filtros a las imágenes.

```
# Borrar todas las imágenes sin usar

> docker image prune -a

# Borrado de la imágenes creadas hace más de una semana  10 días

> docker image prune --filter until="240h"
```

docker image inspect / docker inspect → **obtener información de las imágenes (id y checksum de la imagen, puertos abiertos, arquitectura y SO, tamaño, volúmenes, ENTRYPOINT y capas).**

```
# Dos formas de obtener información de la imagen mysql:8.0.22

> docker image inspect mysql:8.0.22

> docker inspect mysql:8.0.22
```

```
# Mostrar la arquitectura y el sistema
> docker inspect --format '{{.Architecture}} es la arquitectura y el SO es {{.Os}}' mysql:8.0.22
amd64 es la arquitectura y el SO es linux
# Mostrar la lista de puertos expuestos
> docker inspect --format '{{.Config.ExposedPorts}}' mysql:8.0.22
map[3306/tcp:{} 33060/tcp:{}]
```

OTROS COMANDOS:

- **docker image build → construir una imagen desde un fichero Dockerfile (se verá en el apartado 6).**
- **docker image history → mostrar por pantalla la evolución de esa imagen.**
- **docker image save / docker image load (o docker save / docker load) → guardar imágenes en fichero y cargarlas desde fichero (se verá en el apartado 6).**
- **docker image tag (docker tag) → añadir TAGs (versiones) a las distintas imágenes.**