IMAGES

docker pull	descargarme una imagen docker Indicando el nombre de la imagen y la versión de la misma (TAG).

```
# mysql - Es el nombre de la imagen 8.0.22 es la versión o TAG

> docker pull mysql:8.0.22
```

docker run	Indicandouna imagen base que no hayamos descargado previamente. En ese caso se descargará la imagen y posteriormente empezará a ejecutarse el contenedor si todos los parámetros están bien.

```
# Supondremos que es la PRIMERA VEZ que vamos a usar esa imagen y no la hemos descargado

> docker run -it -d --name mysql8 -p 3306:3306 mysql:8.0.22
```

docker images	Listas imágenes descargas. La información que se nos muestra se organiza en forma tabular y nos proporciona los siguientes datos: ● REPOSITORY: Nombre de la imagen en el repositorio. Por ejemplo: mysql. ● TAG: Versión de la imagen que hemos descargado. Por ejemplo: Para la imagen mysql tengo 3 versiones descargadas (5.7, latest que significa que era la última en el momento de descargarse y 8.0.22). ● IMAGE ID: Un identificador que es único para cada imagen. Siempre podemos usar este ID en vez del nombre. ● CREATED: Hace cuánto se creo la imagen. ● SIZE: Tamaño de la imagen.

docker pull	Me permite actualizar una determina pareja imagen:versión a su última actualización. Sólo tendré que hacer docker pull con el mismo imagen:versión

```
# 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 todas la información de las capas.

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

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

Borrado:

```
# 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.

Si aún así queremos borrarla podemos forzar ese borrado, lo cuál afectará,evidentemente, a los contenedores que tuviéramos referenciando esa imagen. Eso lo conseguimos añadiendo la opción **-f o --force.** Por ejemplo:

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

> docker rmi -f httpd
```

Este proceso de borrado, sobre todo si tenemos muchas imágenes, puede ser un proceso engorroso. **Para facilitar** esto disponemos de la orden **docker image prune que tiene tres opciones básicas:**

- **-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.

Para demostrar su funcionamiento vamos a poner varios ejemplos:

```
# 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"
```

```
 # 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:{}]
```

podemos formatear la salida usando Go Templates y el **flag --format/-f.**

```
# 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
```

- El **id y el checksum** de la imagen.
- Los **puertos** abiertos.
- La **arquitectura** y el **sistema operativo** de la imagen.
- El **tamaño de la imagen**.
- **Los volúmenes**.
- El **ENTRYPOINT** que es lo que se ejecuta al hacer docker run.
- Las **capas**.
- Y muchas más cosas....

docker image inspect / docker inspect	nos da ya una información más detallada sobre las características, con todos los metadatos de la misma.

```
# 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
```

docker image tag (docker tag)	para añadir TAGs (versiones) a las distintas imágenes.

docker image save / docker image load (o docker save / docker load)	para guardar imágenes en fichero y cargarlas desde fichero
docker image history	para que se nos muestre por pantalla la evolución de esa imagen.
docker image build	para construir una imagen desde un fichero Dockerfile