3. Gestión de imágenes docker

Hasta ahora hemos **descargado** las imágenes o bien a través de la orden ***docker pull*** o bien al ejecutar ***docker run*** cuando la imagen de base para el container no existía aún en nuestro sistema. Con este sistema llegará un momento en el que tras ir acumulando imágenes querremos deshacernos de algunas de ellas por el motivo que sea.

En este capítulo veremos cómo realizar ese **borrado**, profundizaremos en las **posibilidades que tenemos al descargarnos** una imagen y descubriremos **cómo obtener información detallada** sobre las mismas.

3.1 Descarga de imágenes

Descarga de imágenes Docker

En lo que llevamos de curso hemos visto dos formas para poder descargarme una imagen docker:

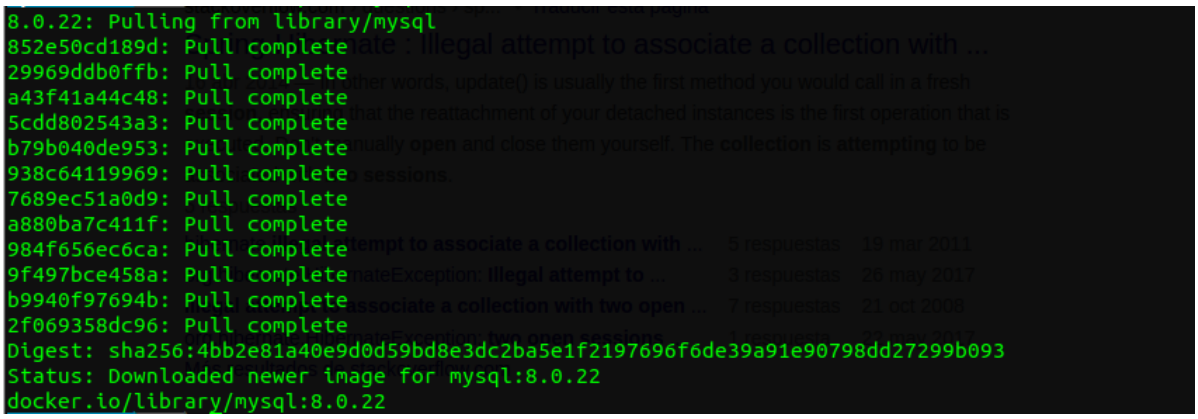
- Usando la el comando ***docker pull*** indicando el nombre de la imagen y la versión de la misma (**TAG**). Si no indicamos nada se descarga la última versión (latest). Por ejemplo:

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

> docker pull mysql:8.0.22
```

- Al hacer ***docker run*** indicando, para la ejecución del contenedor, una 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. Por ejemplo:

```
# 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
```

En cualquiera de los dos casos empieza un proceso de descarga que nos mostrará una serie de líneas, algo parecido a la siguiente imagen:



Juan Diego Pérez. *Resultado: docker pull mysql:8.0.22* (Dominio público)

Cada una de las líneas que se nos muestra es una de las **capas** que **conforman la imagen docker**. Se parte de una imagen **inicial** o **imagen base** y cada instrucción que realizamos para construir una nueva imagen con los elementos que queramos genera una nueva capa que en realidad, son **imágenes intermedias**. Eso nos permite además que las **capas que son compartidas** por varias imágenes no tengan que bajarse dos veces ya que las tendremos ya disponibles en nuestro sistema.

De todas esto hablaremos con más calma en el módulo 6 de este mismo curso: ***"Construyendo mis propios contenedores".***

Mostrar imágenes descargadas

Cada vez que descarguemos una imagen podemos mostrar por pantalla una lista de las imágenes que tenemos en nuestro sistema usando la orden:

```
# Listas imágenes descargas

> docker images
```

La salida a ese comando será algo similar a la siguiente:

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
jperjim398/tomcatcurso2021	latest	a8934a899649	3 days ago	707MB
mysql	8.0.22	dd7265748b5d	5 days ago	545MB
tomcat	9.0.39-jdk11	2703bbe9e9d4	2 weeks ago	648MB
php	7.3-apache	eb304dbcbf28	4 weeks ago	410MB
tomcat	9.0.39-jdk14	9cfb02577f65	6 weeks ago	688MB
bitnami/mariadb	10.2-debian-10	36623ca70d04	6 weeks ago	315MB
jenkins/jenkins	lts	f669140ba6ec	7 weeks ago	711MB
bitnami/minideb	latest	48efba5935f1	8 weeks ago	67.5MB
mariadb	latest	41fa9265d4df	2 months ago	406MB
httpd	latest	417af7dc28bc	2 months ago	138MB
tomcat	9.0-jdk8	fb5729da022d	2 months ago	531MB
tomcat	9.0-jdk11	f796d3d2c195	2 months ago	647MB
tomcat	9.0.38	f796d3d2c195	2 months ago	647MB
tomcat	latest	f796d3d2c195	2 months ago	647MB
php	7.2-apache	569bc7d05428	2 months ago	410MB
mysql	5.7	ef08065b0a30	2 months ago	448MB
mysql	latest	e1d7dc9731da	2 months ago	544MB
debian	latest	f6dcff9b59af	2 months ago	114MB
bitnami/laravel	7-debian-10	1a07111b28fe	4 months ago	778MB
bitnami/mariadb	10.1-debian-10	47e19423b0c1	4 months ago	304MB
bitnami/laravel	6-debian-10	e80e4e6a3218	4 months ago	771MB
sonarqube	latest	4071b5715d76	4 months ago	461MB
danielkraic/asciiquarium	latest	7bab964067d2	21 months ago	309MB
php	5.5-apache	ea0a3d41ce6c	4 years ago	390MB
springio/gs-spring-boot-docker	latest	9065659c6e53	50 years ago	143MB

Juan Diego Pérez Jiménez. *Mostrar imágenes descargadas* (Dominio público)

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

Pese a que hemos visto que existen dos formas de realizar la descarga de las imágenes yo os recomiendo usar docker pull por las siguientes razones:

- 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
```

3.2 Borrado de imágenes

Conforme vamos avanzando en el uso de Docker iremos **acumulando imágenes** en nuestro sistema. Estas imágenes, bien es cierto, no ocupan tanto espacio como una máquina virtual pero si hemos descargado varias decenas o centenas de las mismas (basta un par docker pull -a para eso) al

final nos encontraremos con que **podemos llegar a ocupar una cantidad considerable de espacio en disco** si no tenemos cierto control sobre las mismas.

En este caso, para una mejor gestión, podemos empezar a borrar imágenes de la siguiente forma:

```
# 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
```

Pero este borrado de imágenes, ¿va a ser siempre efectivo?. **NO.**

NO PODEMOS BORRAR UNA IMAGEN SI YA TENEMOS UN CONTENEDOR QUE ESTÁ USÁNDOLA.

No obstante, si lo intentamos no va a suceder nada, simplemente se nos mostrará un mensaje de error como el siguiente:

```
Error response from daemon: conflict: unable to remove repository reference "XXXX" (must force) - container 8d12ffafaaec is using its referenced image 417af7dc28bc
```

Si aun 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"
```

3.3 Obteniendo información de las imágenes

Una vez tenemos ya las imágenes descargadas es muy interesante conocerlas al máximo para poder utilizarlas. Para ello tenemos **dos fuentes principales**:

- La **página de la imagen en DockerHub** que suele recoger sobre todo información relativa a aspectos como:
  - Una descripción de la aplicación o servicio que contiene la imagen.
  - Una lista de versiones TAGs disponibles.
  - Variables de entorno interesantes.
  - Cómo ejecutar la imagen.
- La salida de las órdenes **docker image inspect / docker inspect** que nos da ya una información más detallada sobre las características, con todos los metadatos de la misma.

Veamos un ejemplo 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
```

Obtendremos una salida similar a la siguiente:

```
{
    "Id": "sha256:dd7265748b5dc3211208fb9aa232cef8d3fefd5d9a2a80d87407b8ea649e571c",
    "RepoTags": [
        "mysql:8.0.22"
    ],
    "RepoDigests": [
        "mysql@sha256:4bb2e81a40e9d0d59bd8e3dc2ba5e1f2197696f6de39a91e90798dd27299b093"
    ],
    "Parent": "",
    "Comment": "",
    "Created": "2020-11-21T01:22:38.231041166Z",
    "Container": "bb66c0100e94495bbfc0cd4cd92bcd8d338b96933b20656214f68fd22579b671",
    "ContainerConfig": {
        "Hostname": "bb66c0100e94",
        "Domainname": "",
        "User": "",
        "AttachStdin": false,
        "AttachStdout": false,
        "AttachStderr": false,
        "ExposedPorts": {
            "3306/tcp": {},
            "33060/tcp": {}
        },
        "Tty": false,
        "OpenStdin": false,
        "StdinOnce": false,
        "Env": [
            "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin",
            "GOSU_VERSION=1.12",
            "MYSQL_MAJOR=8.0",
            "MYSQL_VERSION=8.0.22-1debian10"
        ],
        "Cmd": [
            "/bin/sh",
            "-c",
            "#(nop) ",
            "CMD [\"mysqld\"]"
        ],
        "Image": "sha256:5f5de850f91e81a1e04812b200a061d8f380294154b0939a25cb4e8fff7ae5dc",
        "Volumes": {
            "/var/lib/mysql": {}
        },
        "WorkingDir": "",
        "Entrypoint": [
            "docker-entrypoint.sh"
        ],
        "OnBuild": null,
        "Labels": {}
    },
    "DockerVersion": "19.03.12",
    "Author": "",
```


Juan Diego Pérez Jiménez. *Salida del comando docker inspect mysql:8.0.22* (Dominio público)

Esta imagen es una imagen parcial, porque se nos muestra mucha información, está en formato JSON (JavaScript Object Notation) y nos da datos sobre aspectos como:

- 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....

Adicionalmente podemos formatear la salida usando Go Templates y el flag --format/-f. Una descripción detallada queda fuera de los objetivos de este curso pero vamos a poner varios ejemplos:

3.4 Más comandos relacionados con imágenes

Además de los comandos que hemos visto en los apartados anteriores la orden ***docker image*** tiene una gran variedad de **subcomandos**, que si bien no son necesarios para poder empezar con docker si que es bueno conocer que existen, os recomiendo los siguientes:

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