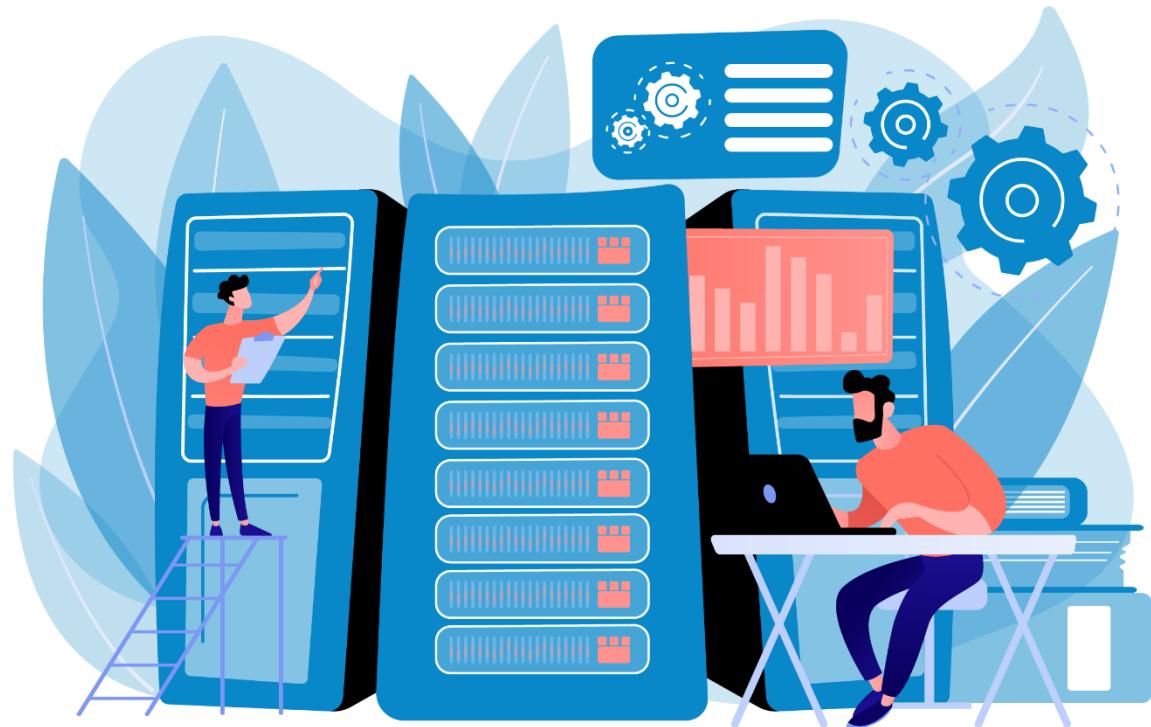


IES MONTE NARANCO

PRÁCTICAS

DEVOPS Y CLOUD COMPUTING



UT - 3

Práctica 1

Índice

1	<i>Creación del primer contenedor</i>	4
1.1	Creación del contenedor	4
	Mostrar los contenedores que se están ejecutando	5
	Mostrar todos los contenedores	5
	Ejecutar un contenedor en segundo plano	6
2	<i>Gestión de contenedores</i>	7
	Eliminar los contenedores parados	7
	Detener un contenedor en ejecución	8
	Eliminar un contenedor detenido	9
	Eliminar un contenedor forzando su detención	9
	Eliminar todos los contenedores parados	10
	Iniciar un contenedor parado	11
3	<i>Gestión de imágenes</i>	12
	Listar las imágenes	12
	Descargar una imagen	12
	Borrar una imagen	16
	Borrar las imágenes que no se están utilizando	16

1 Creación del primer contenedor

COMANDOS QUE SE USARÁN:

docker run ejecuta un contenedor

docker run -d ejecuta un contenedor en segundo plano

docker ps muestra los contenedores que se están ejecutando

docker ps -a muestra todos los contenedores incluso los parados

1.1 Creación del contenedor

1. Desde una ventana de comandos de msdos o de powershell ejecutamos el siguiente comando:

```
docker run alpine echo "¡Hola desde un contenedor!"
```

¿Qué hace este comando?

- **docker run**: le indicamos a Docker que cree y ejecute un contenedor.
- **alpine**: es el nombre de la **imagen** que le indicamos a Docker que use. Docker la buscará y si no la encuentra localmente la descargará.
- **Echo "¡Hola desde un contenedor!"**: es el **comando** que se ejecutará dentro del contenedor.

El resultado obtenido es que se ejecuta el contenedor y se muestra el mensaje ¡Hola desde un contenedor!:

```
C:\Users\Usuario>docker run alpine echo "¡Hola desde un contenedor!"
Unable to find image 'alpine:latest' locally
latest: Pulling from library/alpine
2d35ebdb57d9: Pull complete
Digest: sha256:4b7ce07002c69e8f3d704a9c5d6fd3053be500b7f1c69fc0d80990c2ad8dd412
Status: Downloaded newer image for alpine:latest
¡Hola desde un contenedor!
```

Como vemos en la imagen la primera línea después de ejecutar el comando indica que no se encontró localmente la imagen alpine localmente:

"Unable to find image 'alpine:latest' locally"

Si ejecutamos nuevamente el mismo comando ya no será necesario descargar la imagen ya que se acaba de descargar:

```
C:\Users\Usuario>docker run alpine echo "¡Hola desde un contenedor!"  
¡Hola desde un contenedor!  
C:\Users\Usuario>
```

Mostrar los contenedores que se están ejecutando

2. A continuación, vamos a mostrar los contenedores que se están ejecutando:

```
docker ps
```

```
C:\Users\Usuario>docker ps  
CONTAINER ID        IMAGE           COMMAND       CREATED          STATUS          PORTS          NAMES  
C:\Users\Usuario>
```

No se muestra nada porque los dos contenedores que ejecutamos se detuvieron al terminar el comando echo que indicamos que se ejecutara.

Mostrar todos los contenedores

3. Ahora mostramos todos los contenedores incluso los parados:

```
docker ps -a
```

```
C:\Users\Usuario>docker ps -a  
CONTAINER ID        IMAGE           COMMAND       CREATED          STATUS          PORTS          NAMES  
4d257e877fd0        alpine          "echo '¡Hola desde u...'"  4 seconds ago   Exited (0) 3 seconds ago  
2c6a2610e63e        alpine          "echo '¡Hola desde u...'"  7 seconds ago   Exited (0) 6 seconds ago  
C:\Users\Usuario>
```

Como vemos se muestran los dos contenedores que acabamos de ejecutar, aunque estén detenidos (estado "Exited").

Ejecutar un contenedor en segundo plano

4. Vamos a ejecutar un contenedor que se mantenga en ejecución.

Para ejecutar un contenedor y **evitar que se cierre inmediatamente**, necesitamos proporcionarle un **proceso principal** que se mantenga en ejecución como un servidor web o un servidor mysql.

Para no instalar todavía una imagen compleja vamos a utilizar el siguiente comando:

tail -f /dev/null: este comando se ejecuta indefinidamente esperando nueva información en /dev/null (que nunca llegará), manteniendo el contenedor en estado **Up**.

Además, para evitar que el proceso se quede bloqueando la entrada usaremos el parámetro **-d** (detached) que hace que el contenedor se ejecute en segundo plano.

```
docker run -d alpine tail -f /dev/null
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
49724fb8ec8f	alpine	"tail -f /dev/null"	2 seconds ago	Up 2 seconds		upbeat_lederberg
4d257e877fd0	alpine	"echo '¡Hola desde u...'"	6 minutes ago	Exited (0) 6 minutes ago		magical_clarke
2c6a2610e63e	alpine	"echo '¡Hola desde u...'"	6 minutes ago	Exited (0) 6 minutes ago		amazing_nobel

Al listar los contenedores vemos que el estado del que acabamos de crear es **Up** es decir está en ejecución.

2 Gestión de contenedores

COMANDOS QUE SE USARÁN:

docker ps -a	muestra todos los contenedores
docker run --name	ejecutar un contenedor asignándole un nombre
docker rm id_contenedor	borra el contenedor de la id indicada
docker stop	detiene un contenedor
docker start	inicia un contenedor detenido
docker rm	elimina un contenedor detenido
docker rm -f	fuerza la eliminación de un contenedor
docker container prune	elimina todos los contenedores detenidos

Eliminar los contenedores parados

1. Vamos a eliminar los contenedores parados. Para eliminar un contenedor usaremos el comando **docker rm** poniendo a continuación el **id** o el **nombre** del contenedor que queramos eliminar. Tanto el id como el nombre aparecen en las columnas de la información que se muestra al ejecutar el comando **docker ps -a**.

Borramos el primero de los contenedores detenidos usando su id:

```
C:\Users\Usuario>docker ps -a
CONTAINER ID   IMAGE     COMMAND           CREATED          STATUS          PORTS      NAMES
49724fb8ec8f   alpine    "tail -f /dev/null"   2 seconds ago   Up 2 seconds
4d257e877fd0   alpine    "echo '¡Hola desde u...'"  6 minutes ago   Exited (0) 6 minutes ago
2c6a2610e63e   alpine    "echo '¡Hola desde u...'"  6 minutes ago   Exited (0) 6 minutes ago
upbeat_lederberg
magical_clarke
amazing_nobel
```

```
docker rm 4d257e877fd0
```

```
C:\Users\Usuario>docker rm 4d257e877fd0  
4d257e877fd0  
  
C:\Users\Usuario>docker ps -a  
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES  
49724fb8ec8f alpine "tail -f /dev/null" 4 hours ago Up 4 hours  
2c6a2610e63e alpine "echo '¡Hola desde u...'" 4 hours ago Exited (0) 4 hours ago  
  
C:\Users\Usuario>
```

Vemos que el contenedor con el id especificado ha sido borrado.

2. Borramos el segundo contenedor inactivo usando su nombre. El nombre lo creó automáticamente Docker aunque podríamos haberlo indicado al crear el contenedor.

```
docker rm amazing_nobel
```

```
C:\Users\Usuario>docker rm amazing_nobel  
amazing_nobel  
  
C:\Users\Usuario>docker ps -a  
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES  
49724fb8ec8f alpine "tail -f /dev/null" 4 hours ago Up 4 hours  
  
C:\Users\Usuario>
```

3. Vamos a intentar borrar el contenedor activo igual que los anteriores:

```
docker rm 49724fb8ec8f
```

```
C:\Users\Usuario>docker rm 49724fb8ec8f  
Error response from daemon: cannot remove container "49724fb8ec8f": container is running:  
stop the container before removing or force remove
```

Se produce un error que nos indica que no se puede eliminar el contenedor al estar activo y que tenemos que detenerlo antes de borrarlo.

Detener un contenedor en ejecución

4. Detenemos el contenedor:

```
docker stop 49724fb8ec8f
```

Tras unos segundos termina la ejecución del comando de detención y si listamos los contenedores vemos que el estado ha pasado de **Up** a **Exited**

```
C:\Users\Usuario>docker stop 49724fb8ec8f  
49724fb8ec8f

C:\Users\Usuario>docker ps -a
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS
          PORTS      NAMES
49724fb8ec8f   alpine     "tail -f /dev/null"   6 hours ago   Exited (137) 48 seconds ago
                  upbeat_lederberg
```

Eliminar un contenedor detenido

5. Volvemos a ejecutar el comando para eliminar el contenedor

```
docker rm 49724fb8ec8f
```

Y vemos que ahora sí se ha borrado correctamente:

```
C:\Users\Usuario>docker rm 49724fb8ec8f  
49724fb8ec8f

C:\Users\Usuario>docker ps -a
CONTAINER ID   IMAGE      COMMAND      CREATED        STATUS        PORTS      NAMES
C:\Users\Usuario>
```

6. Vamos a crear de nuevo el contenedor para que se quede en estado activo y en vez de dejar que docker le asigne un nombre vamos a dárselo nosotros:

```
docker run --name mi_contenedor -d alpine tail -f /dev/null
```

Con el parámetro --name le hemos asignado al contenedor el nombre "mi_contenedor" como podemos ver en la siguiente imagen:

```
C:\Users\Usuario>docker run --name mi_contenedor -d alpine tail -f /dev/null
968f8358a68c5f2aa16043718fd79142deaddca30de5a4dade39e0c3f9bb5b0b

C:\Users\Usuario>docker ps -a
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS      NAMES
968f8358a68c   alpine     "tail -f /dev/null"   2 seconds ago   Up 2 seconds
C:\Users\Usuario>
```

Eliminar un contenedor forzando su detención

7. Ahora vamos a borrar el contenedor forzando a que se realice la operación sin que esté detenido. De esta forma nos ahorraremos el paso previo de detenerlo:

```
docker rm -f mi_contenedor
```

```
C:\Users\Usuario>docker rm -f mi_contenedor  
mi_contenedor  
  
C:\Users\Usuario>docker ps -a  
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES  
  
C:\Users\Usuario>
```

8. Vamos a crear dos contenedores que se cierren:

```
docker run alpine echo "¡Hola desde un contenedor!"  
docker run alpine echo "¡Adios desde un contenedor!"
```

```
C:\Users\Usuario>docker run alpine echo "¡Hola desde un contenedor!"  
"¡Hola desde un contenedor!"  
  
C:\Users\Usuario>docker run alpine echo "¡Adios desde un contenedor!"  
"¡Adios desde un contenedor!"  
  
C:\Users\Usuario>docker ps -a  
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES  
8ebd6fc1b60a alpine "echo "¡Adios desde ..." 3 seconds ago Exited (0) 3 seconds ago intelligent_heisenberg  
00ab124e0156 alpine "echo "¡Hola desde u..." 10 seconds ago Exited (0) 9 seconds ago naughty_margulis
```

Eliminar todos los contenedores parados

9. Ahora vamos a ejecutar un comando para borrar todos los contenedores parados:

```
docker container prune
```

Antes de realizar el borrado docker nos pide que confirmemos que queremos borrar todos los contenedores parados:

```
C:\Users\Usuario>docker container prune  
WARNING! This will remove all stopped containers.  
Are you sure you want to continue? [y/N] y  
Deleted Containers:  
8ebd6fc1b60ababda943181b1eb5e4cd1d3700ca389313dc883087b13d7acea6  
00ab124e015623c66cc1fc7588291c54942e0d88e6ef7d97e52ace6a0bf6ae26  
  
Total reclaimed space: 8.192kB  
  
C:\Users\Usuario>docker ps -a  
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES  
  
C:\Users\Usuario>
```

Iniciar un contenedor parado

10. Vamos a crear un contenedor que se mantenga en ejecución como vimos anteriormente. Le ponemos al contenedor el nombre “**prueba**”:

```
docker run --name prueba -d alpine tail -f /dev/null
```

11. A continuación, lo detenemos como vimos anteriormente:

`docker stop prueba`

```
C:\Windows\System32>docker stop prueba
prueba

C:\Windows\System32>docker ps -a
CONTAINER ID        IMAGE       COMMAND                  CREATED             STATUS              PORTS     NAMES
efc0feb60d5a        alpine      "tail -f /dev/null"   About a minute ago   Exited (137) 4 seconds ago   prueba
```

Vemos que ha quedado parado en estado Exited.

12. Lo arrancamos de nuevo:

```
docker start prueba
```

```
C:\Windows\System32>docker start prueba  
prueba  
  
C:\Windows\System32>docker ps -a  
CONTAINER ID        IMAGE       COMMAND                  CREATED             STATUS              PORTS     NAMES  
efc0feb60d5a        alpine      "tail -f /dev/null"   2 minutes ago      Up 1 second          prueba
```

3 Gestión de imágenes

COMANDOS QUE SE USARÁN:

docker images	lista las imágenes descargadas
docker pull	descargar una imagen
docker rmi nombre_imagen	elimina una imagen
docker image prune	borra todas las imágenes
docker image prune -a	borra las imágenes no utilizadas

Listar las imágenes

1. Vamos a listar las imágenes existentes:

```
docker images
```

```
C:\Windows\System32>docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
alpine          latest   4b7ce07002c6  3 weeks ago  12.8MB

C:\Windows\System32>
```

Vemos que en este momento sólo tenemos una imagen descargada.

Descargar una imagen

El repositorio oficial de imágenes docker es **Docker Hub**. Podemos buscar imágenes desde la página web oficial <https://hub.docker.com/> y también desde la aplicación Docker Desktop que tiene un apartado para Docker Hub:

The screenshot shows the Docker Desktop interface. On the left, there's a sidebar with various options like Ask Gordon (BETA), Containers, Images, Volumes, Kubernetes, Builds, Models, MCP Toolkit (BETA), Docker Hub (which is selected and highlighted in grey), Docker Scout, and Extensions. The main area is titled "Docker Hub / Search" and has a search bar with "mysql". Below the search bar, it says "Search results for ‘mysql’" and "Give feedback". The results section shows "1 - 24 of 2500 results". There are two visible results cards: one for the official MySQL repository and one for Bitnami. The MySQL card shows "1B+" downloads and "10K+" stars. The Bitnami card shows "100M+" downloads.

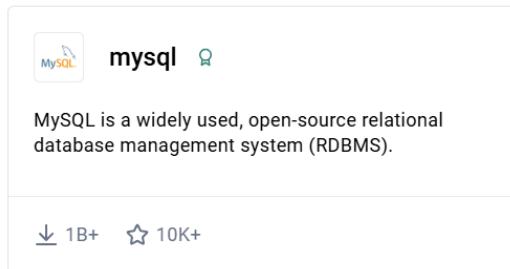
Si buscamos por ejemplo mysql como se muestra en la imagen de arriba aparecen diferentes imágenes:

This screenshot shows the Docker Hub search results for "mysql" with "1 - 24 of 2500 results". The results are displayed in a grid of cards:

- mysql** (Official MySQL repository)
MySQL is a widely used, open-source relational database management system (RDBMS).
Downloads: 1B+, Stars: 10K+
- bitnami/mysql** (Bitnami Secure Image for MySQL)
Bitnami Secure Image for mysql
Downloads: 100M+, Stars: 144
- circleci/mysql** (circleci/mysql repository)
MySQL is a widely used, open-source relational database management system (RDBMS).
Downloads: 100M+, Stars: 32
- bitnamicharts/mysql** (Bitnami Helm chart for MySQL)
Bitnami Helm chart for MySQL.
Downloads: 5M+, Stars: 0
- cimg/mysql** (cimg/mysql repository)
Downloads: 50M+, Stars: 3
- ubuntu/mysql** (Ubuntu MySQL repository)
MySQL open source fast, stable, multi-thread SQL database. Long-term tracks maintained by Canonical.
Downloads: 1M+, Stars: 71
- google/mysql** (Google Compute Engine MySQL repository)
MySQL server for Google Compute Engine
Downloads: 100K+, Stars: 26
- linuxserver/mysql** (LinuxServer.io MySQL repository)
A MySQL container, brought to you by LinuxServer.io.
Downloads: 5M+, Stars: 48
- elestio/mysql** (Elestio MySQL repository)
Mysql, verified and packaged by Elestio
Downloads: 0, Stars: 0
- docksal/mysql** (docksal/mysql repository)
MySQL service images for Docksal -
Downloads: 0, Stars: 0
- alpine/mysql** (alpine/mysql repository)
mysql client
Downloads: 0, Stars: 0
- eclipse/mysql** (Eclipse MySQL repository)
Mysql 5.7, curl, rsync
Downloads: 0, Stars: 0

Podemos descargar la imagen que nos interese directamente desde docker desktop pinchando en el recuadro de la imagen que queramos descargar y luego pulsando en el botón pull.

Por ejemplo, si pinchamos en la primera imagen de mysql:



Vemos que tenemos el botón **Pull** para descargar la imagen:

A screenshot of the Docker Hub MySQL image page. The 'Pull' button is highlighted with a red box. Above the button, there is a dropdown menu set to 'latest'. Below the button is a 'Run' button. The page also shows the MySQL logo, a brief description, and statistics (10K+ stars, 1B+ size).

Quick reference

- Maintained by:
[the Docker Community and the MySQL Team](#)
- Where to get help:

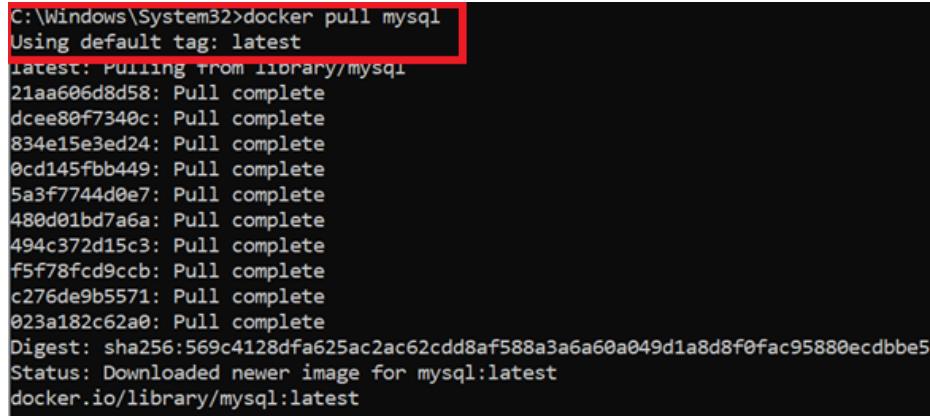
A la izquierda del botón **Pull** vemos que tenemos un cuadro desplegable que pone **latest** que nos permite seleccionar la versión de la imagen:

A screenshot of the Docker Hub MySQL image page. The 'Tags' dropdown menu is open, showing various tags including 'latest', 'oraclelinux9', 'oracle', 'lts-oraclelinux9', 'lts-oracle', 'lts', 'innovation-oraclelinux9', 'innovation-oracle', 'innovation', and '9.5.0-oraclelinux9'. The 'latest' tag is currently selected. The page also includes the MySQL logo, a brief description, and statistics (10K+ stars, 1B+ size), along with 'Overview' and 'Tags' navigation links.

2. Vamos a descargar una imagen de mysql:

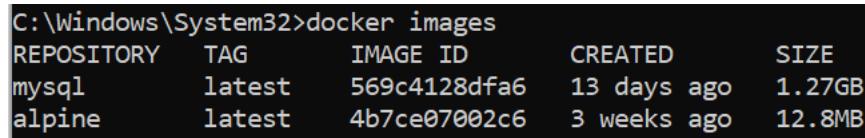
```
docker pull mysql
```

Al no indicar ninguna versión por defecto se utiliza la etiqueta **latest**, es decir se descargará la versión más actual.



```
C:\Windows\System32>docker pull mysql
Using default tag: latest
latest: Pulling from library/mysql
21aa606d8d58: Pull complete
dceef80f7340c: Pull complete
834e15e3ed24: Pull complete
0cd145fb449: Pull complete
5a3f7744d0e7: Pull complete
480d01bd7a6a: Pull complete
494c372d15c3: Pull complete
f5f78fcdb9ccb: Pull complete
c276de9b5571: Pull complete
023a182c62a0: Pull complete
Digest: sha256:569c4128dfa625ac2ac62cdd8af588a3a6a60a049d1a8d8f0fac95880ecdbbe5
Status: Downloaded newer image for mysql:latest
docker.io/library/mysql:latest
```

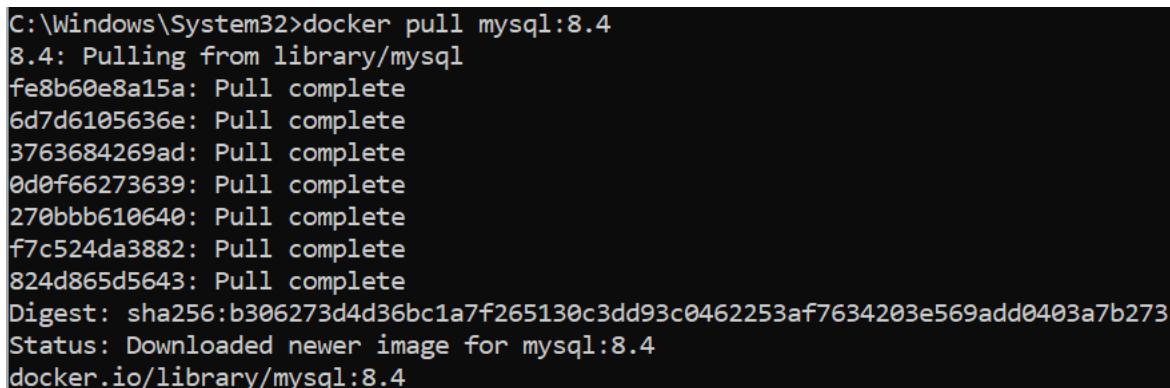
3. Si listamos las imágenes vemos que ahora aparece la nueva imagen descargada:



REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
mysql	latest	569c4128dfa6	13 days ago	1.27GB
alpine	latest	4b7ce07002c6	3 weeks ago	12.8MB

4. Si queremos descargar una versión concreta le añadiremos al nombre de la imagen el carácter de dos puntos ":" y a continuación el nombre de la versión. Por ejemplo para descargar la versión 8.4 de mysql en vez de la última el comando sería:

```
docker pull mysql:8.4
```



```
C:\Windows\System32>docker pull mysql:8.4
8.4: Pulling from library/mysql
fe8b60e8a15a: Pull complete
6d7d6105636e: Pull complete
3763684269ad: Pull complete
0d0f66273639: Pull complete
270bbb610640: Pull complete
f7c524da3882: Pull complete
824d865d5643: Pull complete
Digest: sha256:b306273d4d36bc1a7f265130c3dd93c0462253af7634203e569add0403a7b273
Status: Downloaded newer image for mysql:8.4
docker.io/library/mysql:8.4
```

5. Si listamos nuevamente las imágenes vemos que se ha añadido la imagen de mysql 8.4:

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
mysql	latest	569c4128dfa6	13 days ago	1.27GB
mysql	8.4	b306273d4d36	13 days ago	1.08GB
alpine	latest	4b7ce07002c6	3 weeks ago	12.8MB

Borrar una imagen

Podemos borrar una imagen con el comando **docker rmi nombre_o_id_imagen**.

6. Vamos a borrar la imagen de mysql 8.4

```
docker rmi b306273d4d36
```

```
C:\Windows\System32>docker rmi b306273d4d36
Untagged: mysql:8.4
Deleted: sha256:b306273d4d36bc1a7f265130c3dd93c0462253af7634203e569add0403a7b273
```

7. Ahora borramos la versión de mysql latest usando su nombre que será el nombre de la imagen añadiendo la versión:

```
docker rmi mysql:latest
```

```
C:\Windows\System32>docker rmi mysql:latest
Untagged: mysql:latest
Deleted: sha256:569c4128dfa625ac2ac62cdd8af588a3a6a60a049d1a8d8f0fac95880ecdbbe5
```

Borrar las imágenes que no se están utilizando

- **docker image prune**: este comando elimina únicamente las **imágenes dangling**. Estas son imágenes que se han descargado o construido y que ya **no tienen una etiqueta** (tag) ni son referenciadas por **ningún contenedor**. Por lo general, son capas intermedias o versiones antiguas que quedan huérfanas después de una nueva construcción.
- **docker image prune -a**: elimina todas las imágenes no utilizadas. Esto incluye tanto las imágenes dangling como todas las imágenes que no están asociadas a ningún contenedor en ejecución (o detenido). Esta es una limpieza mucho más agresiva.

8. Si hacemos un listado de las imágenes que nos quedan:

```
docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
alpine	latest	4b7ce07002c6	3 weeks ago	12.8MB

Vemos que actualmente sólo tenemos la imagen alpine.

9. Vamos a comprobar si tiene algún contenedor asociado:

```
docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
C:\Windows\System32>						

10. Ejecutamos el comando de borrar las imágenes **dangling**:

```
docker image prune
```

Nos pide confirmación así que pulsamos en “y” para indicar que si queremos hacer el borrado.

```
C:\Windows\System32>docker image prune
WARNING! This will remove all dangling images.
Are you sure you want to continue? [y/N] y
Total reclaimed space: 0B
```

Vemos que el total de espacio recuperado es 0B y si hacemos un listado de las imágenes:

```
docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
alpine	latest	4b7ce07002c6	3 weeks ago	12.8MB

Vemos que no se ha borrado la imagen, ya que, aunque no tenía un contenedor asociado si que tiene la etiqueta “**latest**” por lo que no es una imagen **dangling**.

11. Ejecutamos el comando que borra todas las imágenes no utilizadas:

```
docker image prune -a
```

```
C:\Windows\System32>docker image prune -a
WARNING! This will remove all images without at least one container associated to them.
Are you sure you want to continue? [y/N] y
Deleted Images:
untagged: alpine:latest
deleted: sha256:4b7ce07002c69e8f3d704a9c5d6fd3053be500b7f1c69fc0d80990c2ad8dd412
deleted: sha256:85f2b723e106c34644cd5851d7e81ee87da98ac54672b29947c052a45d31dc2f

Total reclaimed space: 3.813MB
```

Vemos que tras confirmar que si queremos borrar las imágenes, ahora si que se produce el borrado de la imagen alpine:

```
docker images
```

```
C:\Windows\System32>docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
```