DESPLIEGUE DE APLICACIONES WEB

USO DE CONTENEDORES CON DOCKER

AUTOR: Alberto Martínez Pérez

CURSO: 2º CFGS Desarrollo de Aplicaciones Web (DAW)

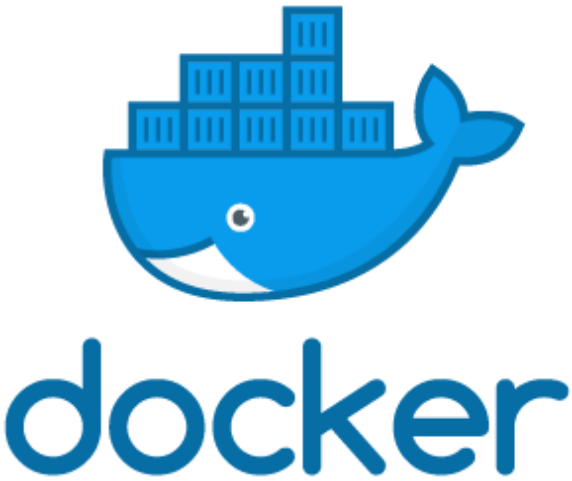
MÓDULO: Despliegue de Aplicaciones Web

FECHA DE ENTREGA: X de marzo de 2024

ÍNDICE

INTRODUCCIÓN

A. ¿QUÉ ES DOCKER?



Docker es una plataforma de código abierto diseñada para facilitar la creación, implementación y ejecución de aplicaciones en contenedores. Los contenedores son entornos ligeros y portátiles que encapsulan una aplicación junto con sus dependencias y configuraciones, permitiendo que se ejecute de manera consistente en diferentes entornos.

Utiliza características de aislamiento de recursos del kernel de Linux como por ejemplo cgroups o los namespaces para permitir que los contenedores se ejecuten dentro de una única instancia de Linux y así evitar la sobrecarga de iniciar y mantener múltiples máquinas virtuales.

B. ¿PARA QUÉ ES ÚTIL DOCKER?

Docker es una herramienta invaluable tanto para administradores de sistemas como para desarrolladores, ya que aborda problemas recurrentes durante el desarrollo y despliegue de aplicaciones en entornos heterogéneos. La diversidad de sistemas operativos, versiones de librerías y configuraciones puede provocar inconsistencias y dificultar la garantía de calidad en el desarrollo de software.

El desafío radica en la variabilidad entre los entornos de desarrollo, pruebas, preproducción y producción. Los desarrolladores pueden preferir diferentes distribuciones de GNU/Linux o incluso sistemas operativos privativos. Además, los sistemas de producción tienden a ser más nuevos y potentes, mientras que los de pruebas y preproducción pueden ser más antiguos. Esto puede resultar en conflictos de versiones de librerías y sistemas operativos, generando problemas de compatibilidad.

Otro obstáculo es la posibilidad de que un desarrollador o un sistema de despliegue tenga que trabajar en múltiples proyectos que requieran

versiones diferentes de librerías. Esto complica aún más la gestión de dependencias y configuraciones.

Docker surge como una solución integral a estos desafíos. Permite crear entornos aislados y reproducibles, asegurando que las aplicaciones se ejecuten de manera consistente en cualquier entorno compatible con Docker. Con la encapsulación de aplicaciones y sus dependencias en contenedores, Docker facilita la creación de entornos idénticos, eliminando los problemas asociados con las variaciones en sistemas operativos y configuraciones.

C. ¿DOCKER ES VIRTUALIZACIÓN?

En GNU/Linux, Docker no utiliza virtualización con un hipervisor. Los procesos que se ejecutan dentro de un contenedor de Docker comparten el mismo kernel que la máquina anfitriona. En lugar de virtualización, Linux utiliza mecanismos de aislamiento para separar estos procesos del resto del sistema, ya sean procesos de la máquina anfitriona o de otros contenedores. Además, Linux puede gestionar los recursos asignados a estos contenedores, como CPU, memoria y red.

Internamente, un contenedor no es consciente de que está en uno, y funciona como una distribución GNU/Linux independiente. Sin embargo, a diferencia de las virtualizaciones convencionales, no sufre la penalización de rendimiento asociada. Al ejecutar un contenedor, esencialmente estamos lanzando un servicio dentro de una distribución construida a partir de una "receta". Esta receta asegura que el sistema sea coherente, independientemente de la distribución de Linux utilizada, ya sea Ubuntu, Fedora, u otros sistemas compatibles con Docker, incluso aquellos privativos.

Al ejecutar contenedores de GNU/Linux dentro de sistemas privativos, se recurre a la virtualización para proporcionar el entorno necesario. En resumen, Docker ofrece un enfoque eficiente para desarrollar y desplegar aplicaciones, asegurando la consistencia del entorno a pesar de las diferencias en las distribuciones de Linux o incluso en sistemas operativos propietarios compatibles con Docker.

D. CONCEPTOS BÁSICOS

- **Demonio de Docker (Docker Daemon):** Es el proceso principal de Docker. Funciona como un servicio en segundo plano y es responsable de gestionar los objetos de Docker, como imágenes, contenedores, redes y volúmenes. El daemon escucha peticiones a

la API de Docker y puede comunicarse con otros demonios para controlar servicios Docker.

- **Cliente de Docker (Docker Client):** Es la principal herramienta que utilizan los administradores de sistemas para interactuar con el sistema Docker. A través del cliente de Docker, los usuarios pueden enviar comandos al demonio de Docker para realizar diversas operaciones, como crear, iniciar, detener o eliminar contenedores.
- **Registro de Docker (Docker Registry):** Es el repositorio donde se almacenan las imágenes de Docker. Permite descargar imágenes para su reutilización. Docker Hub es el principal registro público de Docker, que contiene una amplia variedad de imágenes listas para usar, como MySQL, WordPress, entre otras.

E. OBJETOS DE DOCKER

- **Imagen (Image):** Es una plantilla de solo lectura que contiene las instrucciones para crear un contenedor Docker. Las imágenes pueden basarse en otras imágenes y proporcionan la base para la creación de contenedores.
- **Contenedor (Container):** Es una instancia ejecutable de una imagen. Puede ser creada, iniciada, detenida, movida o eliminada a través del cliente de Docker o la API. Los contenedores pueden conectarse a redes, sistemas de almacenamiento y se pueden configurar en cuanto a su grado de aislamiento del sistema anfitrión y otros contenedores.
- **Servicios (Services):** Los servicios permiten escalar contenedores a través de múltiples demonios de Docker, trabajando conjuntamente como un enjambre (swarm). Los servicios son útiles para distribuir la carga y mejorar la disponibilidad de las aplicaciones en entornos de producción.

INSTALACIÓN DE DOCKER

Para instalar Docker en un sistema GNU/Linux tenemos que utilizar alguna de las siguientes distros CentOS, Debian, Fedora o Ubuntu. En nuestro caso vamos a realizar esta práctica sobre un sistema Ubuntu en su versión 22.04.

Existen dos versiones de Docker, una libre y otra que no lo es. En esta práctica utilizaremos la primera, la conocida como Docker CE o Docker Community Edition.

Lo primero que debemos hacer es actualizar nuestros repositorios y descargar las actualizaciones más recientes:

sudo apt-get update

sudo apt-get upgrade

```
albertom-cliente@docker:~$ sudo apt-get update
Obj:1 http://security.ubuntu.com/ubuntu jammy-security InRelease
Obj:2 http://es.archive.ubuntu.com/ubuntu jammy InRelease
Obj:3 http://es.archive.ubuntu.com/ubuntu jammy-updates InRelease
Obj:4 http://es.archive.ubuntu.com/ubuntu jammy-backports InRelease
Leyendo lista de paquetes... Hecho
albertom-cliente@docker:~$ sudo apt-get upgrade
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Calculando la actualización... Hecho
```

A. INSTALACIÓN DEL GESTOR DE CLAVES

Una vez que hayamos hecho esto debemos instalar el gestor de claves GPG (GNU Privacy Guard) de Docker utilizando los siguientes comandos:

sudo apt-get install ca-certificates curl

De esta forma instalaremos:

- El paquete ca-certificates que contiene certificaciones de autoridades de certificación (CA) que son utilizadas para verificar la autenticidad de certificados SSL.
- El paquete curl que es una herramienta que permite realizar solicitudes HTTP, HTTPS y FPS desde la línea de comandos. Es útil en la descarga de archivos y contenidos.

```
albertom-cliente@docker:~$ sudo apt-get install ca-certificates curl
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
ca-certificates ya está en su versión más reciente (20230311ubuntu0.22.04.1).
fijado ca-certificates como instalado manualmente.
Los paquetes indicados a continuación se instalaron de forma automática y ya
  libflashrom1 libftdi1-2
Utilice «sudo apt autoremove» para eliminarlos.
Se instalarán los siguientes paquetes NUEVOS:
  curl
0 actualizados, 1 nuevos se instalarán, 0 para eliminar y 18 no actualizados.
Se necesita descargar 194 kB de archivos.
```

sudo install -m 0755 -d /etc/apt/keyrings

Este comando se utiliza para crear un directorio en el sistema de archivos con ciertos permisos.

En este caso estamos creando el directorio (opción -d) /etc/apt/keyrings que tendrá unos permisos 755 (rwxr-xr-x).

```
albertom-cliente@docker:~$ sudo install -m 0755 -d /etc/apt/keyrings
```

sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc

Este comando sirve para descargar un archivo GPG que contiene la clave de firma del repositorio de Docker.

```
albertom-cliente@docker:~$ sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/a
pt/keyrings/docker.asc
albertom-cliente@docker:~$ sudo ls -la /etc/apt/keyrings/
total 12
drwxr-xr-x 2 root root 4096 feb 21 15:57 .
drwxr-xr-x 8 root root 4096 nov  1 12:46 ..
-rw-r--r-- 1 root root 3817 feb 21 15:57 docker.asc
```

sudo chmod a+r /etc/apt/keyrings/docker.asc

El último comando se utiliza para modificar los permisos del archivo docker.asc en el directorio /etc/apt/keyrings/.

```
albertom-cliente@docker:~$ sudo chmod a+r /etc/apt/keyrings/docker.asc
albertom-cliente@docker:~$ sudo ls -la /etc/apt/keyrings/
total 12
drwxr-xr-x 2 root root 4096 feb 21 15:57 .
drwxr-xr-x 8 root root 4096 nov  1 12:46 ..
-rw-r--r-- 1 root root 3817 feb 21 15:57 docker.asc
```

B. AÑADIR EL REPOSITORIO DE DOCKER

Ahora vamos a añadir el repositorio de Docker a los repositorios de nuestro sistema para que pueda ser utilizado:

echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \ $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \ sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

```
albertom-cliente@docker:~$ echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrin
gs/docker.asc] https://download.docker.com/linux/ubuntu $(. /etc/os-release && echo "$VERSION_COD
ENAME") stable" |  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

Y actualizamos de nuevo los repositorios

```
albertom-cliente@docker:~$ sudo apt-get update
Obj:1 http://es.archive.ubuntu.com/ubuntu jammy InRelease
Obj:2 http://es.archive.ubuntu.com/ubuntu jammy-updates InRelease
Obj:3 http://es.archive.ubuntu.com/ubuntu jammy-backports InRelease
Obj:4 http://security.ubuntu.com/ubuntu jammy-security InRelease
Des:5 https://download.docker.com/linux/ubuntu jammy InRelease [48,8 kB]
Des:6 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages [26,7 kB]
Descargados 75,5 kB en 2s (43,3 kB/s)
Leyendo lista de paquetes... Hecho
```

C. INSTALACIÓN DE DOCKER

En este paso instalaremos los paquetes necesarios para Docker con el siguiente comando:

sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

- **docker-ce:** Es el paquete principal de Docker Community Edition (CE), que incluye el daemon de Docker, las herramientas de línea de comandos y la documentación.
- **docker-ce-cli:** Es el paquete que proporciona las herramientas de línea de comandos adicionales para Docker, aparte del daemon.
- **containerd.io:** Es un contenedor de tiempo de ejecución que es parte del proyecto CNCF (Cloud Native Computing Foundation). Docker utiliza containerd para ejecutar y supervisar contenedores en el sistema.
- **docker-buildx-plugin:** Es un complemento para Docker que extiende las capacidades de construcción de imágenes de Docker. Permite construir imágenes de contenedores para diferentes plataformas y arquitecturas.
- **docker-compose-plugin:** Aunque la instalación de plugins de Docker Compose solía ser una práctica común, no es necesario instalar un paquete separado para el uso de Docker Compose. Docker Compose se puede instalar y utilizar directamente sin un paquete adicional.

```
albertom-cliente@docker:~$ sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buil
dx-plugin docker-compose-plugin
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Los paquetes indicados a continuación se instalaron de forma automática y ya no son necesarios.
  libflashrom1 libftdi1-2
Utilice «sudo apt autoremove» para eliminarlos.
Se instalarán los siguientes paquetes adicionales:
  docker-ce-rootless-extras git git-man liberror-perl libslirp0 pigz slirp4netns
Paquetes sugeridos:
  aufs-tools cgroupfs-mount | cgroup-lite git-daemon-run | git-daemon-sysvinit git-doc
  git-email git-gui gitk gitweb git-cvs git-mediawiki git-svn
Se instalarán los siguientes paquetes NUEVOS:
  containerd.io docker-buildx-plugin docker-ce docker-ce-cli docker-ce-rootless-extras
  docker-compose-plugin git git-man liberror-perl libslirp0 pigz slirp4netns
```

D. COMPROBACIÓN DE FUNCIONAMIENTO CORRECTO

Para comprobar el funcionamiento vamos a ejecutar la siguiente línea de código:

sudo docker run hello-world

```
albertom-cliente@docker:~$ sudo docker run hello-world
```

Esto produce el siguiente resultado:

```
Unable to find image 'hello-world:latest' locally
```

Que nos informa de que no ha sido capaz de encantrar la imagen hello-world.

Como es la primera vez hemos ejecutado esta imagen, al no encontrarla, procederá a buscar la imagen en el registro público de Docker. En el caso de contar con conexión a Internet, la imagen pública se descargará y

automáticamente creará un contenedor (siempre y cuando la encuentre en el registro público):

```
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:d000bc569937abbe195e20322a0bde6b2922d805332fd6d8a68b19f524b7d21d
Status: Downloaded newer image for hello-world:latest
```

Por último, tanto si se ha descargado la imagen o si ya estaba descargada, el contenedor se ejecutará de forma automática:

```
Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

PREPARANDO EL ENTORNO DE PRÁCTICAS

Vamos a configurar nuestro usuario para usar el cliente de Docker sin que sea necesario poner siempre sudo al inicio de cada orden CLI. Para ello utilizamos el siguiente comando que añadirá al usuario al grupo de Docker:

sudo usermod -aG docker nombre_usuario

```
albertom-cliente@docker:~$ sudo usermod -aG docker albertom-cliente
```

Para que estos cambios tengan efecto debemos cerrar la sesión y volverla a iniciar, de esta forma los permisos se actualizarán.

En esta práctica vamos a utilizar las imágenes de WordPress y MariaDB, para traerlas de los registros públicos podemos usar los siguientes comandos:

docker pull wordpress:latest

```
albertom-cliente@docker:~$ docker pull wordpress:latest
latest: Pulling from library/wordpress
e1caac4eb9d2: Pull complete
8c386db9cb1d: Pull complete
bef1b237c949: Extracting  98.04MB/104.4MB
56c66cb68b0f: Download complete
9c790c1c009d: Download complete
e055748d0b38: Download complete
5a9d72b3b895: Download complete
98b90bb43484: Download complete
b0a0159e983e: Download complete
4a03c0d0f683: Download complete
5cb1486f0b5a: Download complete
370828abc98a: Download complete
a789e0a12acd: Download complete
30b6bf5f6eeb: Download complete
cc0d5481a137: Download complete
f8966f941a57: Download complete
83483f29ea4d: Download complete
371dfab62a96: Download complete
3f4af3e34785: Download complete
ad3725d6d7c1: Download complete
82c78f0e7ebc: Download complete
```

docker pull mariadb:latest

```
albertom-cliente@docker:~$ docker pull mariadb:latest
latest: Pulling from library/mariadb
01007420e9b0: Pull complete
caa624584535: Pull complete
86559e881185: Pull complete
1741a079f7cc: Pull complete
5c03673e89e0: Pull complete
e66164620a33: Pull complete
08e660f565e9: Pull complete
374d99d41aee: Pull complete
Digest: sha256:c33b9fe0c4c10e2a94b62ba952e006c185dfece517c7e3303eff8473f2f523f3
Status: Downloaded newer image for mariadb:latest
docker.io/library/mariadb:latest
```

Por último, también debemos instalar una herramienta más de Docker, Docker Compose, que se utiliza para definir y ejecutar aplicaciones de Docker en varios contenedores.

En esta herramienta utilizaremos un archivo YAML para conficar los servicios de la aplicación y, de esa forma, con un único comando, crear e iniciar todos los servicios.

sudo apt-get install docker-compose

```
albertom-cliente@docker:~$ sudo apt-get install docker-compose
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Los paquetes indicados a continuación se instalaron de forma automá
  libflashrom1 libftdi1-2
Utilice «sudo apt autoremove» para eliminarlos.
Se instalarán los siguientes paquetes adicionales:
  python3-attr python3-distutils python3-docker python3-dockerpty p
  python3-jsonschema python3-pyrsistent python3-setuptools python3-
Paquetes sugeridos:
  python-attr-doc python-jsonschema-doc python-setuptools-doc
```

IMÁGENES

Las imágenes en el contexto de Docker son plantillas de sólo lectura que se crean incorporando los requisitos necesarios para cumplir el objetivo para el cual fueron creadas.

Crearlas de cero supone un trabajo demasiado grande, por lo que lo normal es partir de una creada y almacenada en el registro oficial (http://hub.docker.com).

En esta práctica imaginaremos que queremos crear una web con WordPress, para ello lo primero que haremos será iniciar la imagen de WordPress que hemos descargado en el paso anterior, con el siguiente comando:

docker run -p 8080:80 wordpress

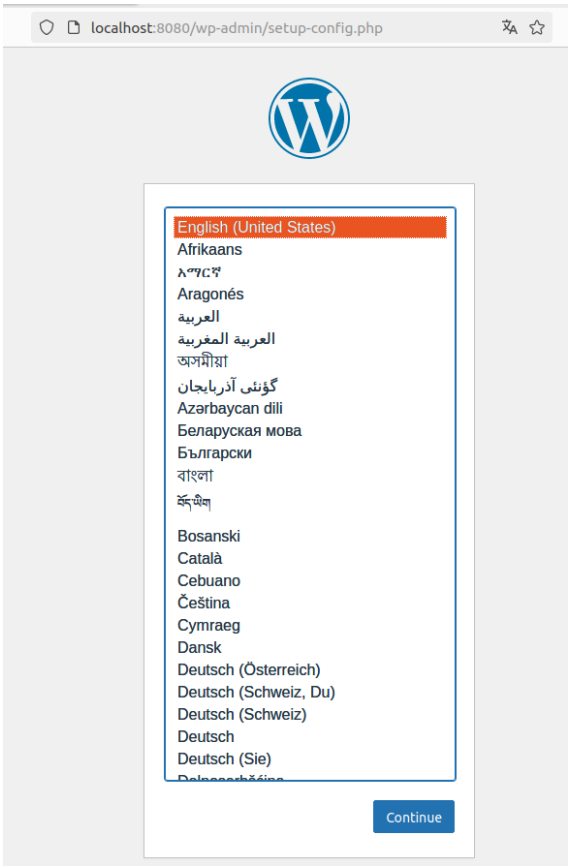
La opción -p seguida de los puertos 8080:80 sirve para mapear el puerto 8080 del sistema anfitrión al puerto 80 del contenedor, es decir, se podrá acceder a la aplicación de WordPress desde el puerto 8080.

```
albertom-cliente@docker:~$ docker run -p 8080:80 wordpress
```

Si el comando ha funcionado de forma correcta veremos como se inicia el contenedor:

```
WordPress not found in /var/www/html - copying now...
Complete! WordPress has been successfully copied to /var/www/html
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 17
2.17.0.2. Set the 'ServerName' directive globally to suppress this message
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 17
2.17.0.2. Set the 'ServerName' directive globally to suppress this message
[Wed Feb 21 15:39:59.282187 2024] [mpm_prefork:notice] [pid 1] AH00163: Apache/2.4.57 (Debian) PH
P/8.2.16 configured -- resuming normal operations
[Wed Feb 21 15:39:59.282358 2024] [core:notice] [pid 1] AH00094: Command line: 'apache2 -D FOREGR
OUND'
```

Si ahora abrimos un navegador y accedemos a http://localhost:8080, accederemos al sitio web de WordPress:



Para paralizar la ejecución del contenedor, lo hacemos igual que cuando queremos paralizar cualquier servicio lanzado desde consola, pulsando Ctrl + C.

```
^C[Wed Feb 21 15:43:45.819339 2024] [mpm_prefork:notice] [pid 1] AH00169: caught SIGTERM, shuttin
g down
```

A. GESTIÓN DE IMÁGENES

Las imágenes que se descargas se identifican por varias cosas, por ejemplo, por el nombre, por la versión, etc. Con esto podemos conseguir tener distintas versiones de una misma imagen.

Por ejemplo, vamos a descargar la versión que utiliza PHP 7.1, para ello debemos descargar la imagen haciendo referencia a esta versión, es decir, en lugar de escribir :latest como en los anteriores ejemplos, escribiremos :php7.1.

docker pull wordpress:php7.1

```
albertom-cliente@docker:~$ docker pull wordpress:php7.1
php7.1: Pulling from library/wordpress
000eee12ec04: Download complete
8ae4f9fcfeea: Download complete
60f22fbbd07a: Downloading   36.5MB/76.65MB
ccc7a63ad75f: Download complete
a2427b8dd6e7: Download complete
91cac3b30184: Waiting
d6e40015fc10: Waiting
54695fdb10a7: Waiting
500ca11be45f: Waiting
86b2805859cf: Waiting
c61685fa4f4f: Waiting
0bf989f9dbbb: Waiting
01848ea209b5: Waiting
153c27b567fb: Waiting
28cf076c939b: Waiting
33b4fcade3a9: Waiting
652f3e96b616: Waiting
d5526eb84f10: Waiting
f429d00f4645: Waiting
192770a70540: Waiting
```

Si queremos ver el listado de imágenes disponibles debemos usar el comando:

docker images

```
albertom-cliente@docker:~$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED         SIZE
mariadb       latest    d344f866f140   18 hours ago    405MB
wordpress     latest    2fc2a7b04129   3 weeks ago     739MB
hello-world   latest    d2c94e258dcb   9 months ago    13.3kB
wordpress     php7.1    04d298cdf70f   4 years ago     530MB
```

Cuando queramos dejar de usar una imagen usaremos el comando:

docker rmi nombre_imagen

```
albertom-cliente@docker:~$ docker rmi wordpress:php7.1
Untagged: wordpress:php7.1
Untagged: wordpress@sha256:a83b97d11bfb719a1e3a441ede3e3efc2396cf419a8581c24b37d61d5bf7c9ee
Deleted: sha256:04d298cdf70fa800f02ff3a12b5fde18c97198e683dbb5b1f56263ffe1773862
Deleted: sha256:fe0b6dc5aa1c5ac46c1e2a33a7a533f5a8a8d7feb58037ab1dfc14d52e19b259
Deleted: sha256:544cdfda028a424073d5e29d3de7a292cd3232ea5beccb03c86114e03e6ef65e
```

```
albertom-cliente@docker:~$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED         SIZE
mariadb       latest    d344f866f140   18 hours ago    405MB
wordpress     latest    2fc2a7b04129   3 weeks ago     739MB
hello-world   latest    d2c94e258dcb   9 months ago    13.3kB
```

Hay que tener en cuenta que, si la imagen está en uso por algún contenedor, no nos dejará eliminarla:

```
albertom-cliente@docker:~$ docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/

albertom-cliente@docker:~$ docker rmi hello-world:latest
Error response from daemon: conflict: unable to remove repository reference "hello-world:latest"
(must force) - container 79c8d863d393 is using its referenced image d2c94e258dcb
```