Docker Guide for Bedu SS

Docker Version

Hola, si ya instalaste Docker, lo mejor será probar que los comandos funcionen, para eso abre tu terminal y prueba el siguiente comando:

```
docker --version
```

Si todo va bien, entonces deberías tener un resultado como este:

```
$ docker --version

Docker version 20.10.7, build f0df350
```

Docker Image

El nombre de "imagen" puede traer muchísima confusión cuando apenas te vas iniciando en el mundo de Docker, pues realmente no queda muy claro que es lo que es, ¿Es cómo una imagen de gatitos 🐱 JPG, o cómo?

Piensa en una imagen de Docker como un "snapshot". Supongamos que te gustaría clonar exactamente el **estado** de tu computadora con todo el software que tienes instalado, tus configuraciones y archivos y pasarlo a una o más computadoras. Todas las computadoras tendrán exactamente lo mismo (aunque el hardware sea diferente) y todas van a iniciar con ese "estado inicial". A ese estado inicial es lo que se le conoce como **snapshot** o en el mundo de Docker como una **imagen**. Has de cuenta que es como si le tomarás una "fotografía" a tu computadora o mejor dicho como una captura de pantalla, pero en vez de ser solo una imagen gráfica (como JPG) es una representación de todo el estado (archivos, software, configuraciones) de tu computadora hasta ese momento de la captura. Otra forma de verlo es como si "congelaras" todo el estado de tu computadora en una imagen, para revivirlo después 🤯.

Si no queda muy claro todavía que cosa es una imagen de Docker, no te preocupes, tarde o temprano tendrá sentido.

Lo más hermoso de todo esto, por si no te habías dado cuenta aún, es que poder hacer este tipo de imágenes está super cool porque así puedes crear ni lo que te imaginas y compartirlo con más desarrolladores del mundo mundial. Actualmente, existen imágenes de Docker casi para cualquier cosa, servidores web (Apache por ejemplo), servidores de base de datos (Postgres, MongoDB, MariaDB, MySQL, Redis, Sqlite, uff..), proxies (Nginx), aplicaciones (Node.js, PHP, Java, ...), y para todo tipo de propósitos, échale un ojo a Docker Hub.

Todas estas imágenes ya tienen todo el software, archivos y configuraciones (incluido el Sistema Operativo, que por lo regular son distribuciones de Linux) que necesitas para echarlas andar. Dicho esto, entonces podemos traernos un servidor de base de datos ya pre configurado y jugar con él en nuestra computadora local, sip, así de fácil, sin necesidad de instalar ¡NADA!

Descargar imágenes de Docker

Para esta mini-guía vamos a descargar MariaDB que es un clon pero de código abierto de MySQL. El comando que vamos usar es `docker pull` seguido de `<nombre-de-imagen>:<tag>`. El `nombre-de-imagen` lo puedes encontrar en Docker Hub, el `tag` se refiere a la versión de la imagen, aunque es opcional, es recomendable siempre especificar la versión que queremos usar, si se omite el valor por default será `latest`.

Descarguemos MariaDB en su versión `10.7`, ejecuta el siguiente comando:

```
docker pull mariadb:10.7
```

Si todo va bien, el proceso de descarga de la imagen debió inciar y el resultado sería algo como esto:

```
$ docker pull mariadb:10.7
```

```
10.7: Pulling from library/mariadb
a39c84e173f0: Pull complete
19c05479159a: Pull complete
7a3fae4be7ce: Pull complete
c6f314de44c1: Pull complete
37a2529e55ed: Pull complete
0e027baf10a6: Pull complete
bba14cc653d8: Pull complete
134d1cd3553b: Pull complete
b0bc73664746: Pull complete
57ff52a2f89e: Pull complete
Digest: sha256:832c6e488f49720f484f87ee9f2cd4487321b373db07ac77037860bcd97d92bb
Status: Downloaded newer image for mariadb:10.7
docker.io/library/mariadb:10.7
```

Finalmente, juguemos con algunos comandos básicos de Docker relacionados con las imágenes.

Listar imágenes de Docker

Para listar que imágenes tenemos descargadas ejecuta el siguiente comando:

```
docker image ls
```

El resultado será más o menos como este, dependiendo de cuantas imágenes tengas descargadas:

```
$ docker image ls

REPOSITORY                  TAG                 IMAGE ID       CREATED         SIZE
mariadb                     10.7                45f96c151bda   2 weeks ago
395MB
mongo                       latest              3376bda5d334   3 months ago
648MB
mysql                       5.7                 8cf625070931   4 months ago
448MB
php                         7.4-apache          308df3245e41   4 months ago
375MB
elixir                      1.8.2               bea4745a510c   7 months ago
1.02GB
```

```
influxdb                    1.8                 a13abe0c1420   7 months ago
293MB
redis                       6                   45e311cc081d   7 months ago
99.7MB
postgres                    13                  9a0cac47ad2e   7 months ago
300MB
node                        14.5.0-alpine3.12   f2f2c78e77d4   17 months ago
117MB
postgres                    11.2                cd1b8c8d55f9   2 years ago
221MB
```

Eliminar imágenes de Docker

Para eliminar alguna imagen, ideal si te estas quedando sin espacio y en realidad no la ocupas, ejecuta `docker image rm` seguido de `<nombre-de-imagen>:<version>`.

Ejemplo:

```
docker image rm mariadb:10.7
```

El resultado del comando anterior sería similar al siguiente:

```
$ docker image rm mariadb:10.7

Untagged: mariadb:10.7
Untagged:
mariadb@sha256:832c6e488f49720f484f87ee9f2cd4487321b373db07ac77037860bcd97d92bb
Deleted: sha256:45f96c151bda8d115bb261123273584d85ea3bcf689105b99c308ea5595a1081
Deleted: sha256:86d4beffd850384dbde6e98cf5c8809cec1c054753cbe7157b3f682b2c5e847c
Deleted: sha256:7e79ae4ae49508588bbbb9a9dcbdd60dfbf3470edf9313001bf7b7f8ddd863dd
Deleted: sha256:111d240cfcd9af22211f78337407c5cdeaadfe45bc77897c9661b95f9b359a19
Deleted: sha256:f1068e987a8c272bba5324999a8f4627b1a23d54aeccf617d359b8c65a334378
Deleted: sha256:3ce04925b3ce2a330f585467d9344a60ebcdbae5e0d46d2b65710d92c2ab7b4d
Deleted: sha256:b89344f95e234b5c1ae32e28ab3b2af6ca085af5a057b875e4e77e5e098ccc6e
Deleted: sha256:f5db8fe6d130581675bf40b189fb3cbb490e0d03842c9e692ad1452e0f889df0
Deleted: sha256:b1d96fe89a023045b668462ef31739e975b1828b4a6686116e8584eefc30e58b
Deleted: sha256:67363cdc1af0ab5009d2924a471c725a5439cfc962601b1467419a0ddc2c7a9d
Deleted: sha256:350f36b271dee3d47478fbcd72b98fed5bbcc369632f2d115c3cb62d784edaec
```

Después de borrar una imagen, ¿cómo se te ocurre recuperarla? 🤔

Pide Aiuda

Cuando tengas duda de que otras cositas te permite hacer Docker, prueba correr cualquier comando seguido de `--help`. Por ejemplo:

```
docker image --help
```

Estamos pidiendo ayuda sobre el comando `docker image` y este es el resultado:

```
$ docker image --help

Usage:  docker image COMMAND

Manage images

Commands:
  build       Build an image from a Dockerfile
  history     Show the history of an image
  import      Import the contents from a tarball to create a filesystem image
  inspect     Display detailed information on one or more images
  load        Load an image from a tar archive or STDIN
  ls          List images
  prune       Remove unused images
  pull        Pull an image or a repository from a registry
  push        Push an image or a repository to a registry
  rm          Remove one or more images
  save        Save one or more images to a tar archive (streamed to STDOUT by
default)
  tag         Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE

Run 'docker image COMMAND --help' for more information on a command.
```

Como puedes leer, te explica como es la sintaxis del comando y sus opciones.

Prueba ejecutando `docker image ls --help` y me cuentas.

Docker Container

Una imagen de Docker por sí sola pues es solo eso, una "imagen", ¿cómo se pueden echar a "andar" las imágenes? o sea, ¿cómo puedo empezar a jugar con ella? ¿dónde se le pica para que prenda? La respuesta alumno mío, está en tu corazón ❤️.

Contenedores de Docker, ¿qué es un contenedor? Has de cuenta que un contenedor es como una mini maquinita virtual que va a compartir recursos con tu computadora localhost (memoria, CPU, espacio de almacenamiento, red, etc) y se va a encargar de "correr" la imagen de la que hemos estado hablando ahí dentro. Sip, es como otra computadora chiquita dentro de tu computadora local con sentimientos y todo.

Estas maquinitas, o contenedores para los cuates, se pueden crear, ejecutar, pausar, parar, eliminar, etc, etc, etc. Cuando creas y ejecutas un contenedor basado en una imagen has de cuenta que es como "pucharle" al botón start y ¿adivina qué? se van a iniciar todos los procesos en esa imagen justamente con ese "estado inicial" para que empieces fresco.

Todos los cambios que le hagas a los archivos (crear o borrar cosas), al software (instalar o desinstalar cosas), o configuraciones (red, seguridad, permisos, etc, etc, etc) dentro del contenedor, vivirán siempre y cuando viva el contenedor.

Los contenedores son efímeros, o sea, son desechables 🥺. Los contenedores están pensados para correr como punto de partida una imagen y de ahí empezar desde cero para correr servidores, aplicaciones, procesos complejos, etc, etc, etc, una vez que su tarea se cumple, pueden detenerse o quizá se mantengan en ejecución (como es el caso de servidores de bases de datos que deben estar todo el tiempo disponibles). Los contenedores no deben almacenar información que se desee preservar, son solo el medio. Pero entonces si un contenedor es efímero y no debe almacenar información porque está se perderá si el contenedor se borra, ¿cómo le hacemos para guardar los datos de la base de datos? 🤔

Docker Volume

Sip, adivinaste bien, además de las imágenes y los contenedores, Docker provee de otra utilidad para poder preservar los datos aunque un contenedor se borre, los volúmenes.

Los volúmenes son como unidades de almacenamiento virtuales pero lo más padre es que se pueden crear varios con propósitos diferentes para un solo contenedor. En el caso de nuestro servidor de MariaDB los datos se guardarán en algún lado, ¿cierto? ¿Y si supiéramos en que carpeta guardan estos datos (lo sabemos, es `/var/lib/mysql`) y pudiéramos de alguna manera construir un "puente virtual", (algo así como los symlinks) entre esta carpeta dentro del contenedor con una carpeta en nuestro local para preservarlos?

Crear un volumen de Docker

Yep, lo volviste a hacer 😉. Vamos a crear un volumen de Docker para que nuestra información se preserve a pesar de que borremos los contenedores de Docker. Un mismo volumen se puede usar a lo largo del tiempo con diferentes contenedores, pero no al mismo tiempo (aunque de esto no estoy muy seguro, pero mejor no lo hagas). Ejecuta el siguiente comando:

```
docker volume create --name=bedu_mariadb
```

Si todo salió bien, deberías tener un resultado como el siguiente:

```
$ docker volume create --name=bedu_mariadb

bedu_mariadb
```

Listar volúmenes de Docker

Al igual que las imágenes, docker también provee de un comando para ver cuantos volúmenes tienes:

```
docker volume ls
```

Si ejecutas el comando anterior, tendrás un resultado como este:

```
$ docker volume ls

DRIVER    VOLUME NAME
local     b71c86145ee675edee8da05842505c0f2bae9c3d2f5e3dc898b9affdd1dd8325
local     bedu_mariadb
local     c188305ecfd12b5f06613f948f8f5551120c10d65b17a86eea2a123cf16b6306
```

No te preocupes de aquellos volúmenes con nombre `!#$%&/•` 😂 esos son creados automáticamente por Docker y deberás eliminarlos cuando sepas qué es lo que estás haciendo, por ahora, déjalos ahí si te aparecen.

Borrar volúmenes de Docker

Antes de continuar, se consciente que un volumen borrado no se puede recuperar (no que yo sepa) y todos los datos guardados allí dentro que no hayas respaldado antes se perderán. Así que cuidado ⚠.

Suponiendo que quieres hacer borrón y cuenta nueva para empezar fresco con tu servidor MariaDB, entonces será necesario crear un nuevo volumen para tu contenedor o borrar el que ya tienes y volverlo a crear. Para borrar un contenedor ejecuta:

```
docker volume rm bedu_mariadb
```

Si el comando se ejecuta correctamente, obtendrás el siguiente resultado:

```
$ docker volume rm bedu_mariadb
```

```
bedu_mariadb
```

Si vuelves a listar tus volúmenes, ya no debería aparecer.

Te recomiendo que lo tengas listo, para la siguiente sección. Anda y vuélvelo a crear, porfis 🙏.

Run MariaDB, run

Una vez que ya sabes la relación que existe entre una imagen, un volumen y un contenedor, es momento de ¡jugar!

Crear y correr un contenedor de Docker

Vamos a crear y correr un container de la imagen de `mariadb:10.7` que utilizará nuestro volumen de `bedu_mariadb` de la sección anterior mapeado al directorio `/var/lib/mysql` dentro del contenedor. También vamos a tener que proveer la variable de entorno `MYSQL_ROOT_PASSWORD` que será la contraseña que se le va a asignar al usuario **root** para poderse conectar a la base de datos, en este caso, le asignaremos `secret` pero puedes cambiarla por lo que tu quieras. A continuación, tenemos que mapear el puerto `3306` del contenedor a puerto `3306` de nuestro localhost, de lo contrario no podremos conectarnos al servidor de base de datos con herramientas como DBeaver. Por último le apodaremos `mariadb_server` para que tenga un nombre.

El comando está un poco largo, pero espero que se entienda. Ejecuta lo siguiente:

```
docker run -v mariadb_data:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=secret -p
0.0.0.0:3306:3306 --name mariadb_server -d mariadb:10.7
```

Si todo salió bien, deberías tener el siguiente resultado:

```
$ docker run -v mariadb_data:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=secret -p
0.0.0.0:3306:3306 --name mariadb_server -d mariadb:10.7
```

```
4fb598e987dc2d122d3f48c2701046332ae52dc3ba8e856743bf6019f9a0a3b4
```

Ese número largo, es el **ID** de tu contenedor, no pasa nada si no te lo aprendes, para eso le pusimos nombre 😉.

Listar los contenedores en ejecución

Seguro te preguntarás y ¿ahora qué? ¿qué pasó? ¿lo hice bien? ¿está vivo? ¿cuál es el sentido de la existencia?

Se supone que tu contenedor está corriendo en el "background" eso fue lo que le indicamos con la opción `-d` , para tener visibilidad de si está corriendo o no, ejecuta el siguiente comando:

```
docker ps
```

El resultado será más o menos así:

```
$ docker ps

CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS
  PORTS                    NAMES
4fb598e987dc   mariadb:10.7   "docker-entrypoint.s…"   5 minutes ago   Up 5 minutes
  0.0.0.0:3306->3306/tcp   mariadb_server
```

Está un poco amontonado pero si ves esto, significa que tu contenedor ¡está vivo! puedes verificarlo checando la columna de `STATUS` que te dirá cuanto tiempo tiene corriendo.

Conexión al servidor de MariaDB con DBeaver.

Ahora ve a DBeaver y conéctate al servidor de base de datos creando una nueva conexión.

***Nota**: Yo tengo DBeaver instalado en Inglés y así daré las instrucciones. Si tienes problema en encontrar las cosas, no dudes en pedir ayuda.*

Del menú selecciona "Database" y después "New Database Connection". Busca "MariaDB" de entre la lista de opciones, una vez que lo encuentres selecciónalo y da click en "Next".

Vas a necesitar estos datos para la siguiente ventana:

Server Host: localhost
Port: 3306
Database: (en blanco por ahora)
Username: root
Password: secret (si lo cambiaste, escribe la nueva)

***NOTA**: Al momento de crear o probar una nueva conexión de MariaDB probablemente te pida que descargues un controlador, dale permiso de hacerlo.*

Antes de dar click en "Finish" prueba tu configuración con el botón de "Test Connection ...". Si te sale una ventana emergente diciendo "Connected", ¡ya la hiciste! Ahora sí, Finished it!

Listo, hasta este punto ya tienes todo lo necesario para la clase. Presúmeles a tus amigos que ya sabes echar a andar un servidor de MariaDB con Docker 😎.

Continua leyendo para conocer otros comandos útiles relacionados con los contenedores.

Detener un contenedor de Docker

A veces quisieras detener los contenedores en vez de borrarlos porque tu computadora se está quedando sin recursos o porque simplemente ya terminaste tu chamba. Detener un contenedor es como si apagaras esa mini computadora pero dejándola vivir. Ejecuta:

```
docker stop mariadb_server
```

Si todo salió bien, verás una respuesta como esta:

```
$ docker stop mariadb_server
mariadb_server
```

Si listas nuevamente los contenedores en ejecución:

```
$ docker ps

CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
```

Notarás que tu contenedor ¡ya no está! 😱 Tranqui, está apagado. Ejecuta el siguiente comando:

```
docker ps -a
```

El resultado será más o menos así:

```
$ docker ps -a
CONTAINER ID   IMAGE                          COMMAND                  CREATED
     STATUS                      PORTS                                    NAMES
4fb598e987dc   mariadb:10.7                   "docker-entrypoint.s…"   28 minutes
ago   Exited (0) 2 minutes ago
mariadb_server
```

Perdón si está medio amontonado, pero lo que nos interesa es ver que efectivamente el container sigue apareciendo en la lista y que en la columna `STATUS` verás que dice `Exited (0) <n> minutes ago` o algo parecido.

Iniciar un contenedor de Docker

Cuando un contenedor está apagado o sea detenido,
- ¿Cómo lo volvemos a correr? ¿Con el comando `run` otra vez?

- ¡NO, cuidado! el comando `run` siempre te va a crear un nuevo y brillante contenedor desde cero.
- ¿Tons?
- ¿Qué es lo opuesto de `stop`?
- Pues `run` ¿qué no?
- El otro opuesto 🤦🏻‍♂️
- 🤔
- ⏳
- ¡Ah, ya¡ El `start` 😋
- ¡Eso! Ahora ve e inicia tu container. Ejecuta:

```
docker start mariadb_server
```

Si todo sale bien, obtendrás la siguiente salida:

```
$ docker start mariadb_server
mariadb_server
```

Lista nuevamente los contenedores en ejecución, ahí deberá aparecer.

Borrar un contenedor de Docker

Hemos llegado a la última sección de este mini tutorial express de Docker... lo sé, lo sé, no llores, mira ven 🤗.

Cuando desees eliminar un contenedor de Docker, recuerda que todas sus aventuras y sus memorias pasarán al olvido también, a no ser que lo hayas conectado con un volumen que preservará los datos que hayas elegido por la eternidad. Recuerda que los contenedores su único propósito en el mundo es nacer, crecer, cumplir con su tarea y morir... 😭. Ya pues. ¡Vamos a borrarlo! 😈 Dale:

```
docker rm mariadb_server
```

Jajaja, no se deja morir tan fácil, ¿verdad?

```
$ docker rm mariadb_server

Error response from daemon: You cannot remove a running container
4fb598e987dc2d122d3f48c2701046332ae52dc3ba8e856743bf6019f9a0a3b4. Stop the
container before attempting removal or force remove
```

Para borrar un contenedor, primero necesitamos detenerlo con el comando `stop`, después de hacerlo, ya puedes ir a borrarlo. Vamos, inténtalo nuevamente 😉.

Ahora sí, este es el fin.

Notas finales

Espero que hayas disfrutado esta mini guía y que le hayas entendido un poco o más o menos a esta cosa llamada Docker. La verdad a mí me tomó mucho tiempo y cursos agarrarle la onda, si no entendiste nada de nada, no te preocupes, a todos nos pasa, una vez que entiendes Docker, no quieres que se vaya de tu vida nunca más. ¡El futuro es hoy, viejo 😎!

Así como en el último comando nos topamos con un error, es probable que te topes con muchos más, normalmente los errores te dirán que estas tratando de hacer y porque no puedes hacerlo, lee detenidamente cual es el problema e intenta resolver con lo que sabes y si no puedes, siempre puedes consultar en la red 😚 o preguntarle a tu colega de confianza.

Gracias por leer, me divertí mucho haciendo este documento.

Chido ✌🏼

Isaac R.