

¿Qué es docker?	1
Contenedor e imágenes.....	3
Instalación de Docker-CE	5
Docker y Busybox.....	7
El comando «docker ps»	9
Ejecución interactiva	10
Limpiando espacio.....	10
Servidor MariaDB con Docker.....	12
Descarga de la imagen	12
Creación de un contenedor	13
Conexión desde un cliente.....	16

¿Qué es docker?



Según la Wikipedia:

Docker es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de virtualización de aplicaciones en múltiples sistemas operativos. Docker hace uso de las características de aislamiento de recursos del núcleo de Linux, tales como los cgroups y los namespaces que permiten que contenedores independientes se ejecuten dentro de una misma instancia de Linux, evitando la sobrecarga de iniciar y mantener máquinas virtuales.

Una definición más práctica nos la da la firma analista 451 Research:

Docker es una herramienta que puede empaquetar una aplicación y sus dependencias en un contenedor virtual que puede ejecutarse en cualquier servidor Linux. Esto aporta flexibilidad y portabilidad a las aplicaciones, ya que éstas se pueden ejecutar en instalaciones físicas, la nube pública, nubes privadas, etc.

Básicamente, lo que consigue Docker es ofrecer contenedores de aplicaciones que aprovechen las capacidades de virtualización del kernel de Linux para poder ejecutar procesos y servicios de forma aislada.

Se trata de un concepto parecido al de máquina virtual, pero que no requiere de un sistema operativo, sino que aprovecha el kernel del propio Linux y sus capacidades de aislar recursos, tales como la CPU, la memoria, la red o la entrada/salida. Así pues, podemos tener varios contenedores compartiendo el mismo kernel de Linux, pero cada uno con restricciones de acceso a determinados recursos.

Las principales ventajas del uso de contenedores son:

- La flexibilidad, ya que incluso las aplicaciones más complejas pueden incluirse en los contenedores,
- La poca carga que suponen para el sistema, al compartir el mismo kernel que el anfitrión,
- La posibilidad de desplegar actualizaciones en caliente,
- La portabilidad, ya que se pueden desarrollar localmente, desplegar en la nube y lanzarlos en cualquier sitio,
- La escalabilidad, ya que permite incrementar automáticamente réplicas de los contenedores,
- Los servicios en contenedores pueden apilarse on the fly

Contenedor e imágenes

A lo largo del documento hablaremos sobre imágenes y contenedores, por lo que conviene aclarar estos conceptos.

Una **imagen** es un paquete ejecutable que incluye todo lo necesario para ejecutar una aplicación: el código, el entorno de ejecución, librerías, variables de entorno y archivos de configuración.

Un **contenedor**, por su parte, es una instancia de una imagen en ejecución: lo que se crea cuando ponemos en marcha una imagen. Podríamos decir que un contenedor es a una imagen lo que un proceso en un programa (proceso=programa en ejecución -> contenedor=imagen en ejecución).

1. Preparación del entorno

Fuente: <https://docs.docker.com/install/linux/docker-ce/ubuntu/>

El proyecto Docker dispone de sus propios repositorios de software. En este apartado vamos a ver cómo descargarnos la versión Community de Docker, orientada a desarrolladores y equipos pequeños que comienzan con Docker. La alternativa empresarial sería Docker Enterprise Edition (EE).

En este apartado vamos a instalar las herramientas necesarias para poder descargar e instala Docker CE en nuestro equipo (o máquina virtual).

En primer lugar, actualizamos la caché de paquetes del ordenador:

sudo apt-get update

```
sudo apt-get update
```

E instalamos los siguientes paquetes:

sudo apt-get install apt-transport-https ca-certificates curl software-properties-common

```
sudo apt-get install apt-transport-https ca-certificates curl software-properties-common
```

Con esto, descargamos la clave GPG del sitio de Docker (con el comando curl) y la incorporamos al sistema (con apt-key add -), para que nuestro sistema confíe en el sitio para la descarga de

software (fijese que hay una tubería |):

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

Para comprobar que la clave se ha instalado correctamente:

sudo apt-key fingerprint 0EBFCD88

```
manu@ideamanu:~$ sudo apt-key fingerprint 0EBFCD88
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).
pub   rsa4096 2017-02-22 [SCEA]
      9DC8 5822 9FC7 DD38 854A  E2D8 8D81 803C 0EBF CD88
uid   [desconocida] Docker Release (CE deb) <docker@docker.com>
sub   rsa4096 2017-02-22 [S]
```

Si todo es correcto y han aparecido bien las claves de arriba, podemos continuar añadiendo el repositorio a los orígenes de software de nuestro sistema (lo que tenemos en /etc/apt/sources.list*):

sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu \$(lsb_release -cs) stable"

```
manu@ideamanu:~$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
Repositorio: «deb [arch=amd64] https://download.docker.com/linux/ubuntu noble stable»
Descripción:
Archive for codename: noble components: stable
Más información: https://download.docker.com/linux/ubuntu
Añadiendo repositorio.
Oprima [INTRO] para continuar o Ctrl+c para cancelar.
Se encontró la entrada deb existente en /etc/apt/sources.list.d/docker.list
Actualizando la entrada existente en lugar de usar /etc/apt/sources.list.d/archive_uri-https_download_docker_com_linux_ubuntu-noble.list
Añadiendo la entrada desactivada deb-src a /etc/apt/sources.list.d/docker.list
Obj:1 http://security.ubuntu.com/ubuntu noble-security InRelease
Obj:2 https://download.docker.com/linux/ubuntu noble InRelease
Obj:3 https://packages.microsoft.com/repos/code stable InRelease
Obj:4 https://dl.google.com/linux/chrome/deb stable InRelease
Obj:5 http://es.archive.ubuntu.com/ubuntu noble InRelease
Obj:6 http://es.archive.ubuntu.com/ubuntu noble-updates InRelease
Obj:7 http://es.archive.ubuntu.com/ubuntu noble-backports InRelease
Leyendo lista de paquetes... Hecho
```

Y ahora, ya instalamos el paquete docker-ce.

Instalación de Docker-CE

Con los repositorios ya configurados, sólo debemos realizar la instalación con apt-get:

sudo apt-get install docker-ce

```
sudo apt-get install docker-ce
```

Para comprobar que todo ha funcionado bien, vamos a lanzar un contenedor con la imagen hello-world:

sudo docker run hello-world

```
manu@ideamanu:~$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
17eec7bbc9d7: Pull complete
Digest: sha256:54e66cc1dd1fcb1c3c58bd8017914dbed8701e2d8c74d9262e26bd9cc1642d31
Status: Downloaded newer image for hello-world:latest

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/
```

Como vemos, nos indica que no encuentra la imagen «hello-world:latest», por lo que la descarga de la librería (pull). Una vez descargada ya nos muestra el mensaje que comienza con Hello from Docker, correspondiente a esta imagen. Si volvemos a lanzar el comando, comprobaremos cómo ya no realiza la descarga, sino que ejecuta directamente la imagen Hello World.

Docker y servicios

El demonio (daemon) de Docker es el servicio que gestiona la creación, ejecución y distribución de contenedores. El comando docker que hemos visto anteriormente, es el cliente de docker, que permite al usuario interactuar con el sistema, aunque también existen otras aplicaciones clientes.

Si queremos que el servicio de Docker se inicie al arrancar el sistema (systemd), haremos:

sudo systemctl enable docker

Y si queremos quitarlo del inicio:

sudo systemctl disable docker

Podemos encontrar más información sobre todo lo que podemos hacer después de haber instalado Docker en: <https://docs.docker.com/install/linux/linux-postinstall/#configure-docker-to-start-on-boot>, así como documentación para la configuración del servicio de Docker para que escuche determinados puertos mediante el archivo `/etc/docker/daemon.json`: <https://docs.docker.com/install/linux/linux-postinstall/#configure-where-the-docker-daemon-listens-for-connections>

Docker y Busybox

Para descargar el busybox hacemos uso del comando de Docker `docker pull` (fijese que somos el usuario administrador):

docker pull busybox

```
manu@ideamanu:~$ sudo docker pull busybox
Using default tag: latest
latest: Pulling from library/busybox
80bfbb8a41a2: Pull complete
Digest: sha256:d82f458899c9696cb26a7c02d5568f81c8c8223f8661bb2a7988b269c8b9051e
Status: Downloaded newer image for busybox:latest
docker.io/library/busybox:latest
manu@ideamanu:~$
```

Con esto debemos descargar la imagen de busybox pe a Docker desde el Docker Hub (<https://hub.docker.com/explore/>).

Para ver las imágenes de Docker que tenemos instaladas, podemos realizar:

docker images

```
manu@ideamanu:~$ sudo docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
hello-world   latest    1b44b5a3e06a   5 weeks ago   10.1kB
busybox       latest    0ed463b26dae   11 months ago 4.43MB
manu@ideamanu:~$
```

Como se puede comprobar, tenemos la imagen de busybox y hello-world.

Vamos ahora a lanzar el Busybox. Para ello haremos:

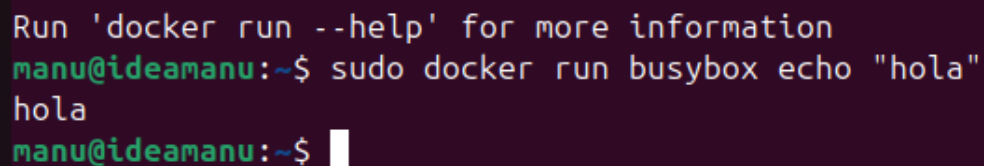
docker run busybox

Con esto, si no tenemos descargada la imagen, lo primero que hará es descargarla. Si ya la tenemos descargada, omitirá este paso. Con la imagen de busybox en el sistema, Docker la busca, la carga

en un contenedor y ejecuta en el busybox las órdenes que le pasamos. Como en este caso no le hemos pasado ningún orden, aparentemente no hará nada.

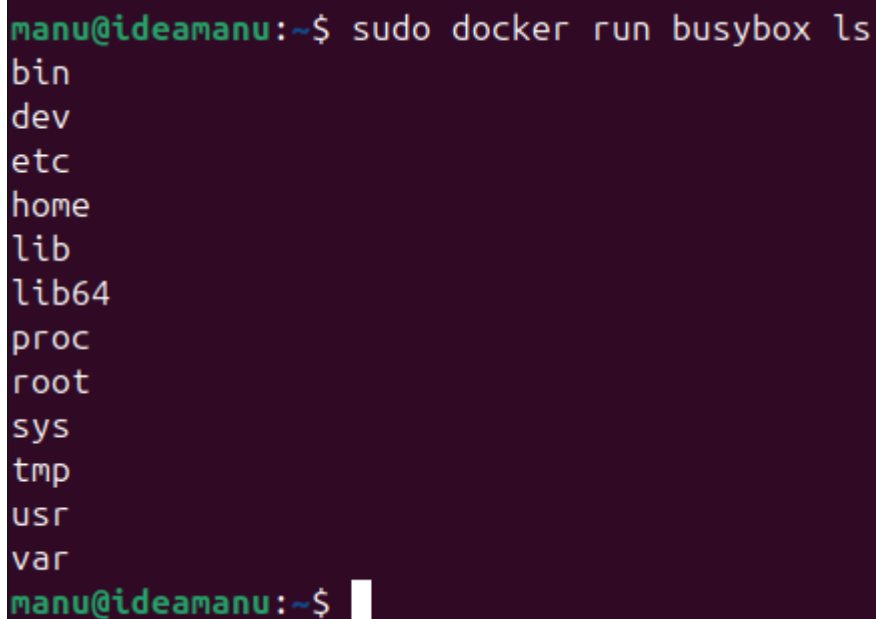
Así pues, para ejecutar algo dentro del docker deberemos pasarle como parámetro:

`docker run busybox echo "hola"`

A terminal window with a dark purple background. The prompt is 'manu@ideamanu:~\$'. The command 'sudo docker run busybox echo "hola"' is entered. The output 'hola' is displayed on the next line. The prompt is then 'manu@ideamanu:~\$' with a cursor.

```
Run 'docker run --help' for more information
manu@ideamanu:~$ sudo docker run busybox echo "hola"
hola
manu@ideamanu:~$
```

docker run busybox ls

A terminal window with a dark purple background. The prompt is 'manu@ideamanu:~\$'. The command 'sudo docker run busybox ls' is entered. The output lists the contents of the root directory: 'bin', 'dev', 'etc', 'home', 'lib', 'lib64', 'proc', 'root', 'sys', 'tmp', 'usr', 'var'. The prompt is then 'manu@ideamanu:~\$' with a cursor.

```
manu@ideamanu:~$ sudo docker run busybox ls
bin
dev
etc
home
lib
lib64
proc
root
sys
tmp
usr
var
manu@ideamanu:~$
```

docker run busybox cat /etc/passwd


```
manu@ideamanu:~$ sudo docker run busybox cat /etc/passwd
root:x:0:0:root:/root:/bin/sh
daemon:x:1:1:daemon:/usr/sbin:/bin/false
bin:x:2:2:bin:/bin:/bin/false
sys:x:3:3:sys:/dev:/bin/false
sync:x:4:100:sync:/bin:/bin/sync
mail:x:8:8:mail:/var/spool/mail:/bin/false
www-data:x:33:33:www-data:/var/www:/bin/false
operator:x:37:37:Operator:/var:/bin/false
nobody:x:65534:65534:nobody:/home:/bin/false
manu@ideamanu:~$
```

Como podemos ver, hemos lanzado tres órdenes diferentes sobre el Busybox: hemos escrito hola, hemos listado el sistema de archivos, y hemos consultado el archivo `/etc/passwd`. Nótese que ni el sistema de archivos que hemos mostrado ni el archivo `passwd` se corresponden con la estructura de archivos del sistema o el archivo `passwd` de nuestro sistema. Estamos accediendo al sistema de archivos y el archivo `passwd` del propio Busybox. De hecho, si

hacemos un `ps aux`, ver que no hay otro proceso en el sistema:

docker run busybox ps aux

```
manu@ideamanu:~$ sudo docker run busybox ps aux
PID    USER     TIME   COMMAND
   1  root         0:00   ps aux
manu@ideamanu:~$
```

El comando «docker ps»

El mandato `docker ps` sirve para ver los contenedores que se están ejecutando en un momento dado. Si hacemos en un terminal:

docker run busybox sleep 10

Y desde otro:

sudo docker ps

```
manu@ideamanu:~$ sudo docker ps
[sudo] contraseña para manu:
Lo siento, pruebe otra vez.
[sudo] contraseña para manu:
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
299f87e2f590   busybox   "sleep 10"  5 seconds ago  Up 4 seconds           serene_booth
```

Vemos que tenemos el comando sleep 10 funcionando sobre la imagen de busybox en el contenedor 299f87e2f590

Ejecución interactiva

Si queremos lanzar más de una orden por contenedor, podemos hacer uso del parámetro -it (flag interactive):

docker run -it busybox sh

```
manu@ideamanu:~$ sudo docker run -it busybox sh
/ # ls
bin    dev    etc    home   lib    lib64  proc   root   sys    tmp    usr    var
/ # users

/ # cat /etc/passwd
root:x:0:0:root:/root:/bin/sh
daemon:x:1:1:daemon:/usr/sbin:/bin/false
bin:x:2:2:bin:/bin:/bin/false
sys:x:3:3:sys:/dev:/bin/false
sync:x:4:100:sync:/bin:/bin/sync
mail:x:8:8:mail:/var/spool/mail:/bin/false
www-data:x:33:33:www-data:/var/www:/bin/false
operator:x:37:37:Operator:/var:/bin/false
nobody:x:65534:65534:nobody:/home:/bin/false
/ # exit
manu@ideamanu:~$
```

Limpiando espacio

Con la opción `-a` de `docker ps` podemos obtener todos los contenedores que se han creado en la sesión actual:

`docker ps -a`

```
manu@ideamanu:~$ sudo docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
af02c9c58490	busybox	"sh"	3 minutes ago	Exited (0) 2 minutes ago		unruffled_leavitt
299f87e2f590	busybox	"sleep 10"	18 minutes ago	Exited (0) 18 minutes ago		serene_booth
7b4f49b5090a	busybox	"sleep 10"	18 minutes ago	Exited (0) 18 minutes ago		peaceful_montalcini
3ceb35c1895d	busybox	"sleep 10"	18 minutes ago	Exited (0) 18 minutes ago		wizardly_curran
a8257e82379f	busybox	"sleep 10"	20 minutes ago	Exited (0) 20 minutes ago		sweet_mclaren
db4dd1f4843d	busybox	"ps aux"	22 minutes ago	Exited (0) 22 minutes ago		gracious_almeida
9d7782e89440	busybox	"cat /etc/passwd"	24 minutes ago	Exited (0) 24 minutes ago		affectionate_babbage
7b20af293fbc	busybox	"ls"	26 minutes ago	Exited (0) 26 minutes ago		elated_haibt
d63e4afd1ac4	busybox	"echo hola"	26 minutes ago	Exited (0) 26 minutes ago		sharp_wright
2ef815e4ed4d	hello-world	"/hello"	41 minutes ago	Exited (0) 41 minutes ago		jolly_villani
e28dc85e348c	hello-world	"/hello"	21 hours ago	Exited (0) 21 hours ago		nervous_ardinghelli

```
manu@ideamanu:~$
```

Como vemos, la columna status indica que los contenedores han terminado. Cuando queremos eliminarlos, podríamos hacer:

`docker rm e28dc85e348c, 2ef815e4ed4d ...`

Este mecanismo es algo tedioso, por lo que vamos a hacerlo más sencillo.

Con la siguiente orden, podemos obtener los ids de los contenedores que ya han terminado:

`sudo docker ps -a -q -f status=exited`

```
manu@ideamanu:~$ sudo docker ps -a -q -f status=exited
af02c9c58490
299f87e2f590
7b4f49b5090a
3ceb35c1895d
a8257e82379f
db4dd1f4843d
9d7782e89440
7b20af293fbc
d63e4afd1ac4
2ef815e4ed4d
e28dc85e348c
manu@ideamanu:~$
```

Por lo que si combinamos esta orden con `docker rm`:

sudo docker rm \$(sudo docker ps -a -q -f status=exited)

```
manu@ideamanu:~$ sudo docker rm $(sudo docker ps -a -q -f status=exited)
af02c9c58490
299f87e2f590
7b4f49b5090a
3ceb35c1895d
a8257e82379f
db4dd1f4843d
9d7782e89440
7b20af293fbc
d63e4afd1ac4
2ef815e4ed4d
e28dc85e348c
manu@ideamanu:~$
```

Servidor MariaDB con Docker

Como hemos comentado anteriormente, Docker Hub (<https://hub.docker.com/explore/>) ofrece una gran cantidad de imágenes. Este sitio puede entenderse como una especie de Github para imágenes de Docker. Las imágenes se pueden clasificar de diversas formas atendiendo a distintos criterios.

Atendiendo al proceso de creación de la imagen, podemos distinguir:

- Imágenes de base: Aquellas que han sido creadas de cero, generalmente a partir de sistemas operativos como Ubuntu, Busybox o Debian, o bien
- Imágenes hijas: Aquellas que están construidas sobre una imagen base, con funcionalidad adicional.

Por otra parte, atendiendo a quien ha creado las imágenes, distinguimos:

- Imágenes oficiales: Son aquellas mantenidas por Docker, y que generalmente son imágenes de base. Generalmente el nombre es una cadena, como busybox y hello-world.
- Imágenes de usuario: Son aquellas creadas y compartidas por los usuarios. Por lo general, son imágenes basadas en las imágenes de base con funcionalidad adicional. Normalmente, se llaman con usuario/imagen.

Para disponer de un servidor de Subversion, haremos uso de la imagen mamohr/subversion-edge (<https://hub.docker.com/r/mamohr/subversion-edge>), basada en el servidor de subversion Subversion Edge de CollabNet, que dispone de interfaz web para su gestión.

Descarga de la imagen

Vamos a trabajar con la imagen oficial de SGBD MySQL (https://hub.docker.com/_/mysql). Para descargar la imagen más reciente del SGBD, haremos:

docker pull mysql

```
manu@ideamanu:~$ sudo docker pull mysql
Using default tag: latest
latest: Pulling from library/mysql
500d7b2546c4: Pull complete
01859c60b4e2: Pull complete
87565b56b57e: Pull complete
9b2f3769f0be: Pull complete
7d70b564625b: Pull complete
1d289f7d1ed9: Pull complete
d210a5b69bfe: Pull complete
95f5cac1a9e9: Pull complete
98045e6cd572: Pull complete
1421f5b704d5: Pull complete
Digest: sha256:94254b456a6db9b56c83525a86bff4c7f1e52335f934cbcd686fe1ce763116a0
Status: Downloaded newer image for mysql:latest
docker.io/library/mysql:latest
manu@ideamanu:~$
```

Creación de un contenedor

Una vez descargada la imagen, podemos lanzarla con un simple `docker run mysql`, pero no tendríamos el servicio disponible desde nuestro equipo, y sus datos desaparecerían cuando borramos el contenedor.

Para darle persistencia, deberemos enlazar la carpeta donde guarda MySQL toda la información con una carpeta de nuestro equipo, conocida en terminología de Docker como volumen.

Para ello, crearemos una carpeta, por ejemplo, en `/srv/mysql-data`:

sudo mkdir /srv/mariadb-data

Y ahora lanzaremos un contenedor con la siguiente orden (las `\` son para indicar cambio de línea en bash, pero no deben ponerse si lo escribimos todo a una línea):

```
sudo docker run --name mysql-srv \
```

```
-p 3308:3306 \  
-v /srv/mysql:/var/lib/mysql \  
-e MYSQL_ROOT_PASSWORD="root" \  
-d mysql
```

```
manu@ideamanu:~$ sudo docker run --name mysql-srv \  
-p 3308:3306 \  
-v /srv/mysql:/var/lib/mysql \  
-e MYSQL_ROOT_PASSWORD="root" \  
-d mysql  
73a6f84b416253f8748eede5591331a03ad648e83bf9eb159ab69020e554415a  
manu@ideamanu:~$
```

Las opciones que hemos utilizado han sido:

- `--name mysql-srv`: Le damos un nombre (mysql-srv) al contenedor, para cuando debamos detenerlo o eliminarlo referirnos a él de forma más sencilla.
- `-p 3308:3306`: Aquí hacemos lo que se conoce como exposición de puertos, es decir, exponemos los puertos para los que trabaja el contenedor por defecto a través de los puertos de nuestro equipo. En ese caso, el puerto por defecto de MySQL (3306) del contenedor, estará visible en nuestra máquina a través del puerto 3308.
- `-v /srv/mysql:/var/lib/mysql`: Enlazamos el volumen que acabamos de crear (la carpeta /srv/mysql), con la carpeta /var/lib/mysql, que es donde el servidor de MySQL almacena los datos. `-e MYSQL_ROOT_PASSWORD="root"`: Con `-e` establecemos variables de entorno. En ese caso, estamos estableciendo el valor de la variable `MYSQL_ROOT_PASSWORD` (es decir, la contraseña de root) como root.
- `-d`: Indica que vamos a lanzar el contenedor en background, sin que nos muestre todos los mensajes de log por pantalla.

Y, por último, hemos indicado el nombre de la imagen **mysql** a lanzar. Con esto, si hacemos un **docker ps**, obtendremos:

```
manu@ideamanu:~$ docker ps  
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS                               NAMES  
73a6f84b4162   mysql    "docker-entrypoint.s..." 4 minutes ago  Up 4 minutes  33060/tcp, 0.0.0.0:3308->3306/tcp, [::]:3308->3306/tcp  mysql-srv  
manu@ideamanu:~$
```

Con esto ya tendremos un contenedor con un servidor de MySQL funcionando en estos momentos, y que podremos detener y arrancar cuando lo necesitemos.

Poniendo en marcha un contenedor ya creado

Una vez lanzado el comando `docker run` anterior, creamos el contenedor a partir de la imagen, y podemos detenerlo con `docker stop`. De todas formas, como hemos visto, con esto detenemos el contenedor, pero no lo eliminamos. De esta forma, si intentamos volver a crear de nuevo el contenedor con `docker run`, obtendremos el siguiente error:

```
sudo docker run --name mysql-srv -p 3308:3306 -v /srv/mysql:/var/lib/mysql -e MYSQL_ROOT_PASSWORD="root" -d mysql
```

Esto nos está indicando que el nombre de contenedor `mysql-srv` ya está en uso por otro contenedor. Si hacemos un `docker ps -a`, veremos que tenemos este contenedor creado:

```
docker ps -a
```

Ahora tenemos dos posibilidades, eliminar este contenedor (con `docker rm`) o seguir ejecutándolo.

Esta última opción tiene la ventaja de que, dado que no hemos eliminado el contenedor, la información que hemos guardado, aunque no hemos utilizado ningún volumen, seguiría estando disponible. Así, para poner en marcha de nuevo este contenedor ya creado, haremos:

```
docker start mysql-srv
```

Así pues, ya modo de conclusión, podemos establecer las siguientes diferencias entre `docker run` y `docker start`:

- `docker run`: Crea un nuevo contenedor a partir de una imagen y ejecuta los comandos que indicamos.

- `docker start`: Inicia un contenedor parado, manteniendo éste tal y como estaba en el momento de detenerlo, por lo que mantiene la información que éste estuviera gestionando sin necesidad de utilizar volúmenes.

Conexión desde un cliente

Para conectarnos desde la terminal, deberemos indicarle la IP local 127.0.0.1 (no nos sirve el localhost por defecto) y el puerto por el que nos vamos a conectar (3308):

`sudo mysql -u root --host=127.0.0.1 --port=3308 -p`

```
manu@ideamanu:~$ sudo mysql -u root --host=127.0.0.1 --port=3308 -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 9.4.0 MySQL Community Server - GPL

Copyright (c) 2000, 2025, Oracle and/or its affiliates.

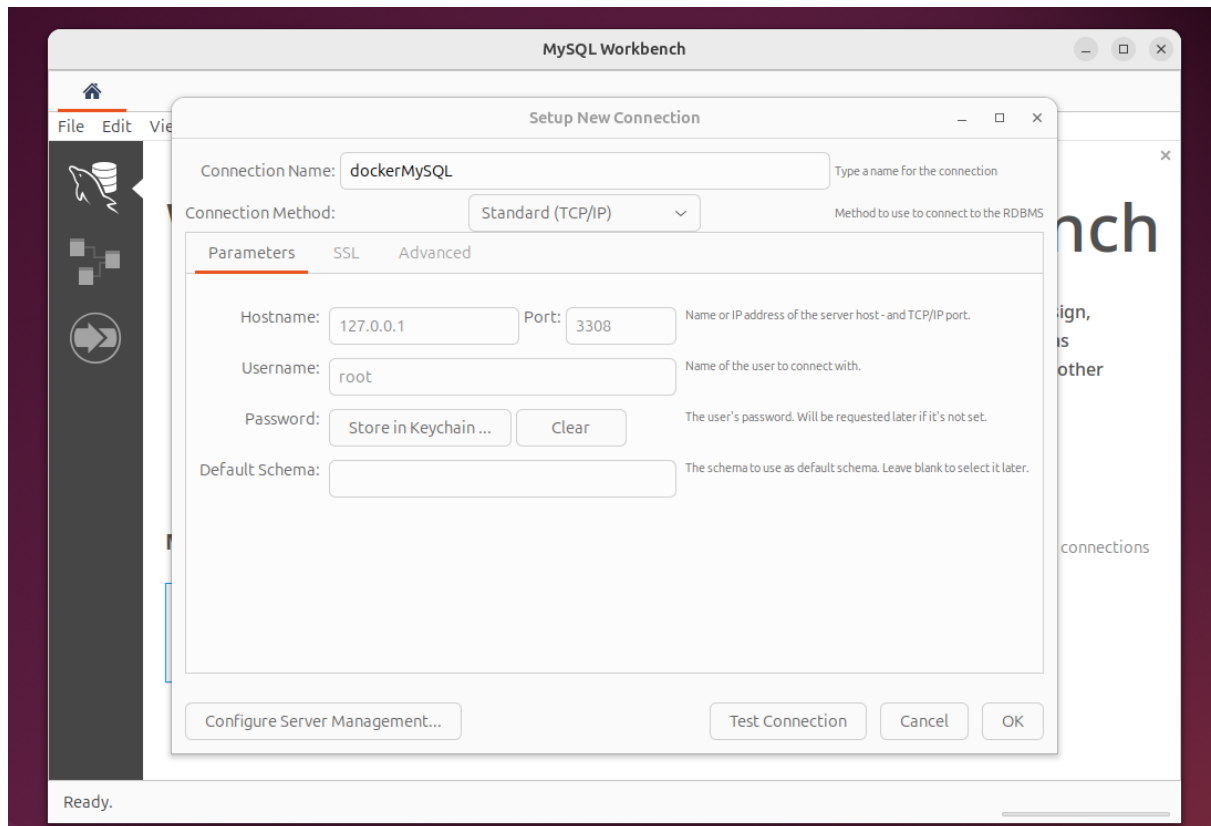
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Como vemos, nos pide el password de root (ya que hemos indicado la opción -p), y una vez introducido, nos muestra el prompt de MySQL.

También podemos conectarnos desde otro cliente que nos dé más juego, como MySQL Workbench. Si no lo tenemos instalado, podemos hacerlo con `sudo apt-get install mysql-workbench`. Con esta herramienta, sólo deberemos crear una nueva conexión con los parámetros con los que hemos configurado el servidor. Para ello, desde la ventana principal, haremos clic en el símbolo + que tenemos ubicado junto a MySQL Connections y configurar la nueva conexión con los siguientes parámetros:



Hecho esto, podemos testear la conexión para ver si está todo correcto y conectarnos al servidor.