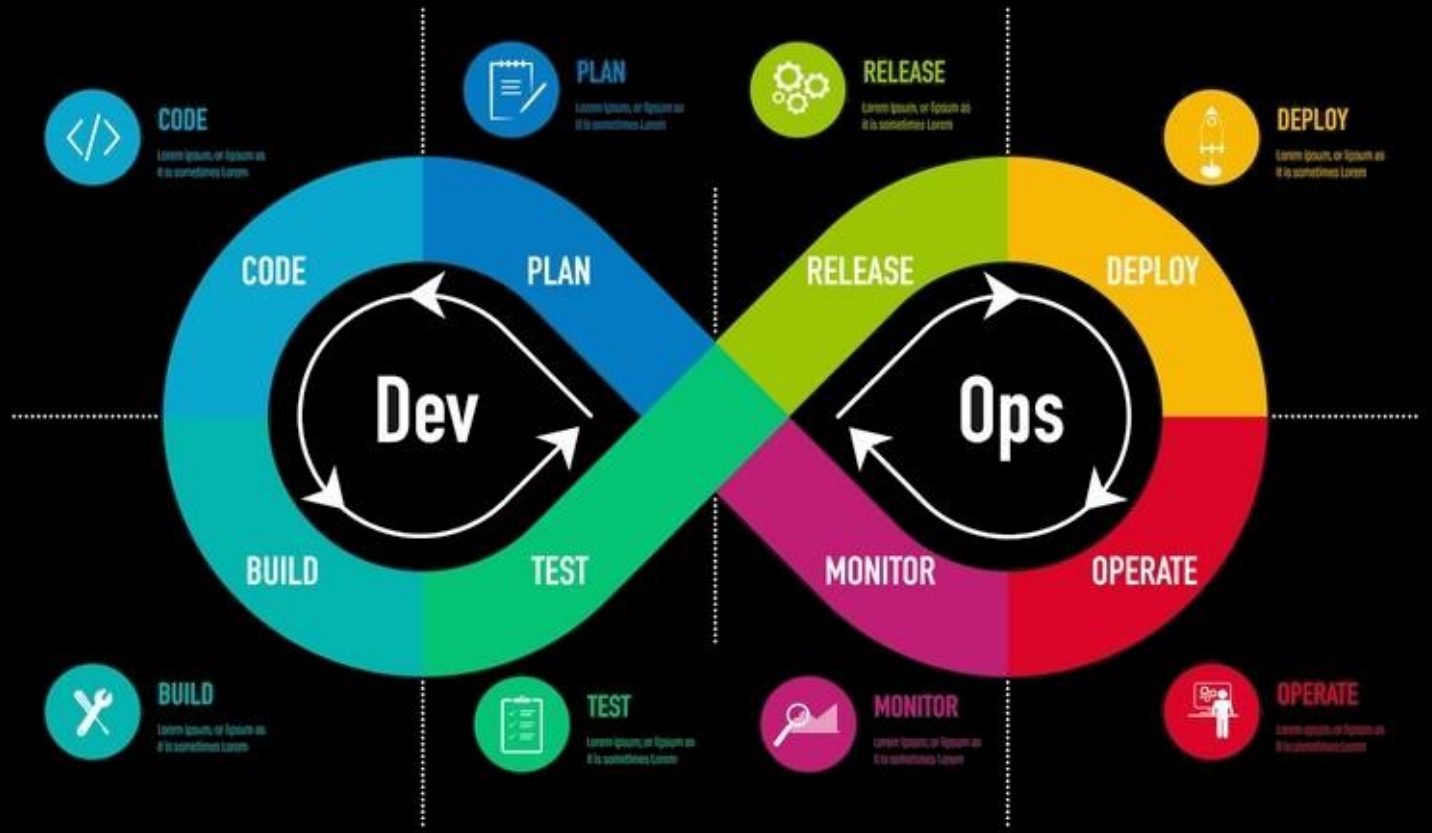
CFGS DAM

IES MONTE NARANCO



UT- 3 DevOps y Cloud Computing

Fundamentos de contenedores

Práctica 2 – Configura imagen MySQL

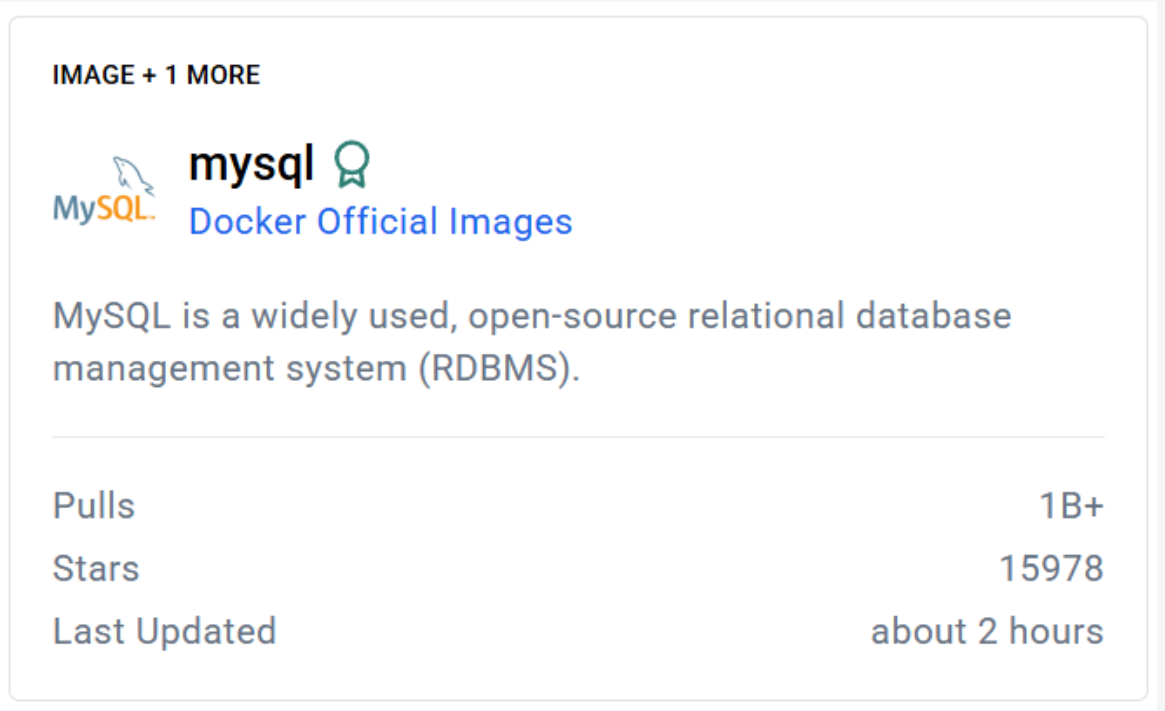
Descargar la imagen de MySQL

```
docker pull mysql
```

Al no poner versión es como si hubiéramos puesto **mysql:latest** como vimos en la práctica anterior.

Consultar información sobre la imagen

- https://hub.docker.com/
- Buscamos “mysql”
- Pinchamos en la imagen de mysql



Crear un contenedor con la imagen

```
docker run --name mi-mysql -e MYSQL_ROOT_PASSWORD=1234 -d mysql
```

- **--name mi-mysql** establecemos que el contenedor se llamará mi-mysql
- **-e MYSQL_ROOT_PASSWORD=1234** el parámetro -e sirve para establecer variables de entorno. En este caso estamos estableciendo el valor de **MYSQL_ROOT_PASSWORD** que es el valor de la contraseña del usuario **root** y le estamos dando el valor 1234
- **-d** indicamos que el contenedor se ejecute en segundo plano (detached)
- **mysql** indicamos que el contenedor usará la imagen mysql y al no poner etiqueta se utilizará la imagen mysql:latest

Crear un contenedor con la imagen

Mostramos un listado de los contenedores:

```
docker ps
```

En este caso no nos ha hecho falta poner **docker ps -a** ya que al ser una base de datos mysql se quedará activo

Utilizar la consola de mysql

```
docker exec -it mi-mysql mysql -u root -p
```

- **docker exec**: Ejecuta un comando en un contenedor en funcionamiento.
- **-it**: es una combinación de dos parámetros:
  - **-i** (--interactive): Mantiene la entrada estándar (STDIN) abierta.
  - **-t** (--tty): Asigna un pseudo-terminal para que la interfaz se vea y funcione correctamente.
- **mi-mysql**: nombre de la imagen en la que se ejecutará el comando, se puede usar el nombre o el **id**

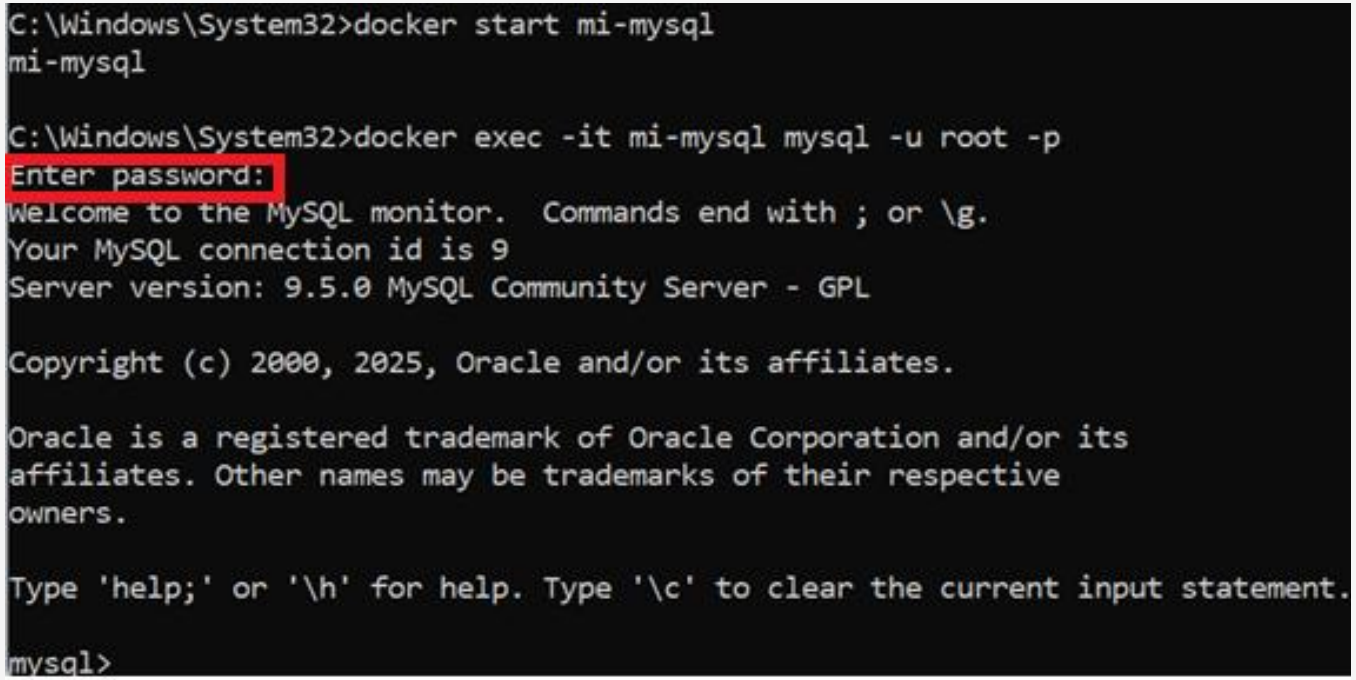
Utilizar la consola de mysql

```
docker exec -it mi-mysql mysql -u root -p
```

- **mysql**: comando que se va a ejecutar dentro del contenedor
- **-u root**: con este parámetro indicaremos el usuario con el que vamos a conectar a la consola de mysql, en este caso estamos indicando que conectaremos con el usuario **root**
- **-p**: con este parámetro indicamos que se pedirá la contraseña del usuario de forma interactiva (si quitamos este parámetro se producirá un error de credenciales al intentar conectarse a la consola mysql)

Utilizar la consola de mysql

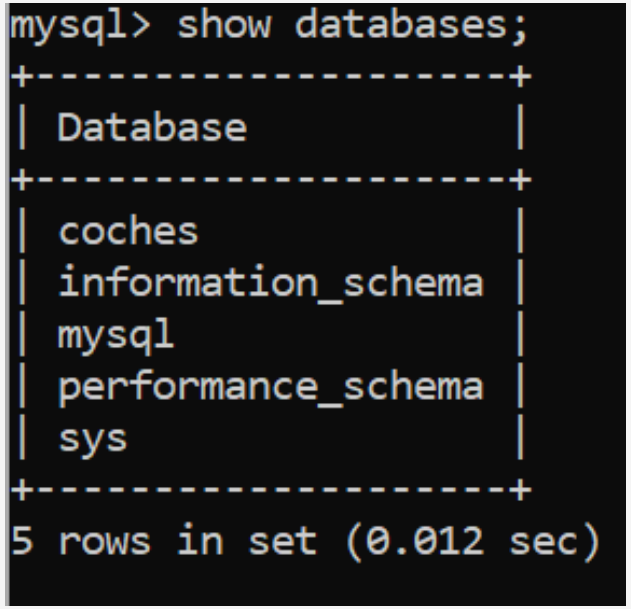
Como vemos en la imagen se solicita la contraseña del usuario root y una vez que la introducimos se inicia la sesión en la consola mysql.



Utilizar la consola de mysql

Ejecutamos un comando de prueba para comprobar que todo funciona correctamente:

```
show databases;
```



Conectar a mysql desde un cliente externo

- Para poder conectar al servidor mysql desde un servidor externo debemos mapear el puerto de nuestro equipo con el del contenedor.
- Esto se hace utilizando la opción **-p** (o **--publish**) al ejecutar el comando **docker** run para crear el contenedor.

```
-p [PUERTO_HOST]:[PUERTO_CONTENEDOR]
```

- [PUERTO_HOST]: es el puerto de nuestra máquina (el que abrimos en nuestro navegador o cliente) para acceder al servicio.
- [PUERTO_CONTENEDOR]: El puerto interno en el que el servicio (como un servidor web o una base de datos) está realmente escuchando dentro del contenedor.

Conectar a mysql desde un cliente externo

```
docker run --name mi-mysql -e MYSQL_ROOT_PASSWORD=1234 -d -p 3306:3306 mysql
```

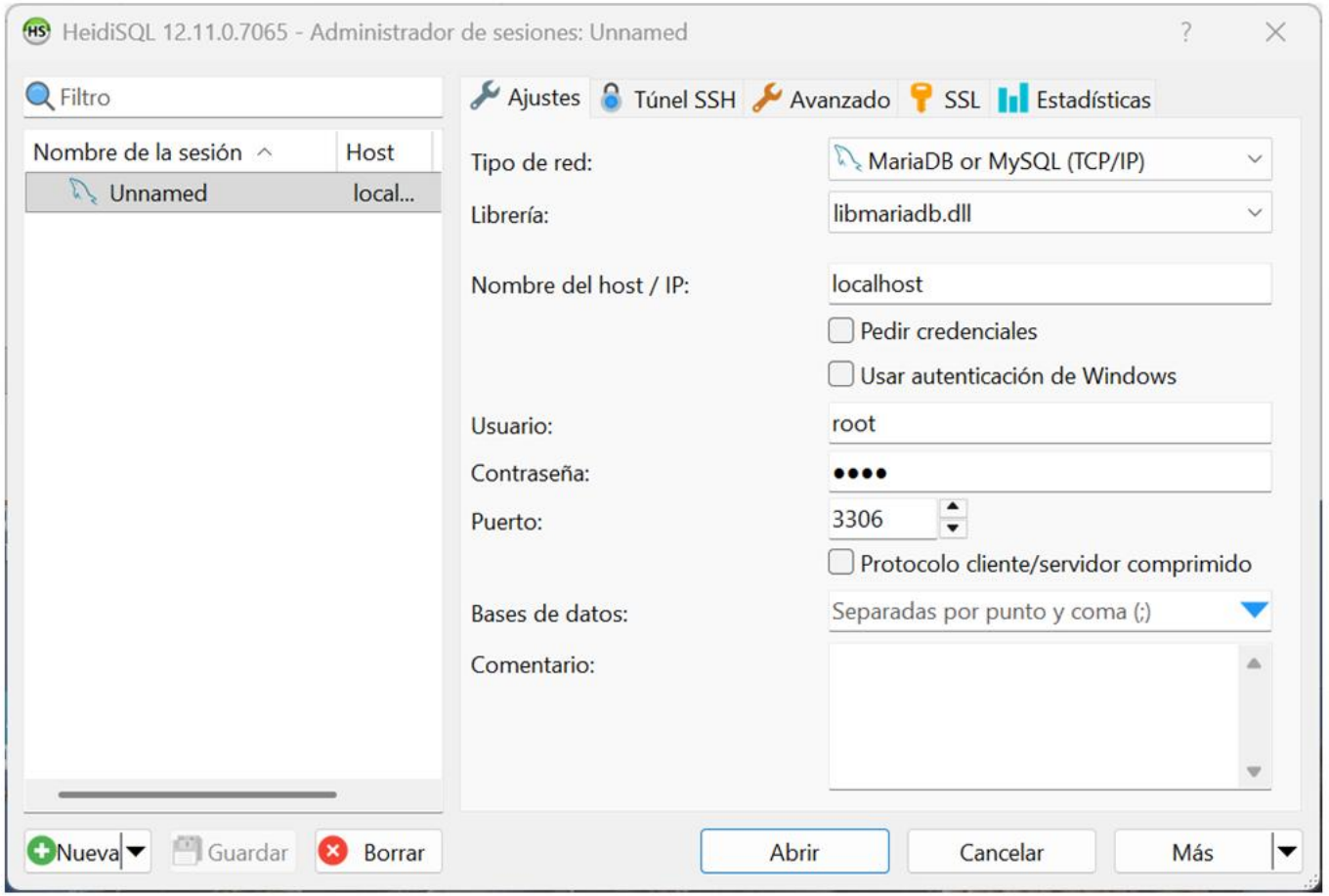
Si listamos los contenedores que se están ejecutando:

```
docker ps
```

```
Windows\System32>docker ps
TAINER ID   IMAGE     COMMAND                  CREATED              STATUS              PORTS
33c24b69a   mysql     "docker-entrypoint.s…"   About a minute ago   Up About a minute   0.0.0.0:3306->33060/tcp, [::]:3306->33060/tcp
```

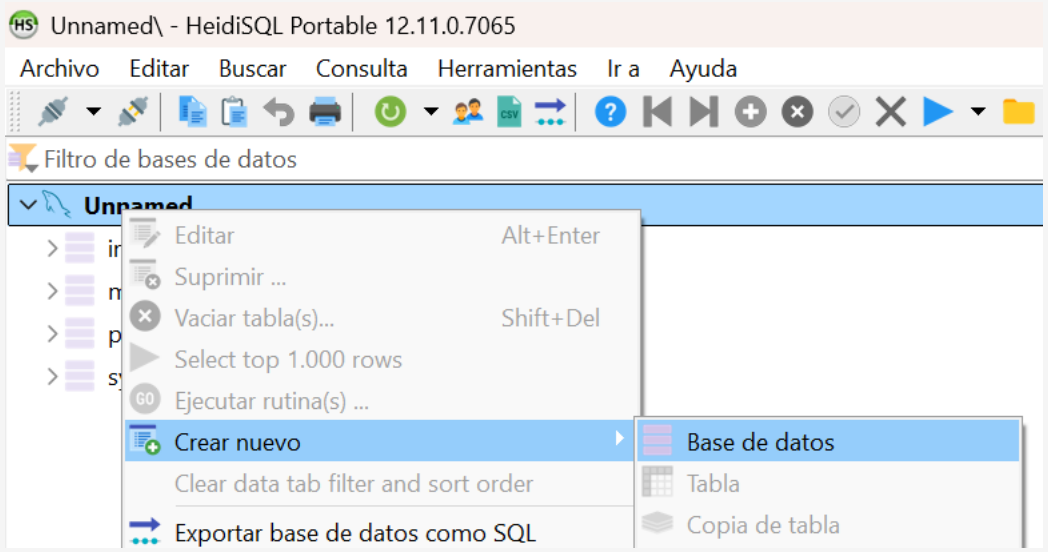
Conectar a mysql desde un cliente externo

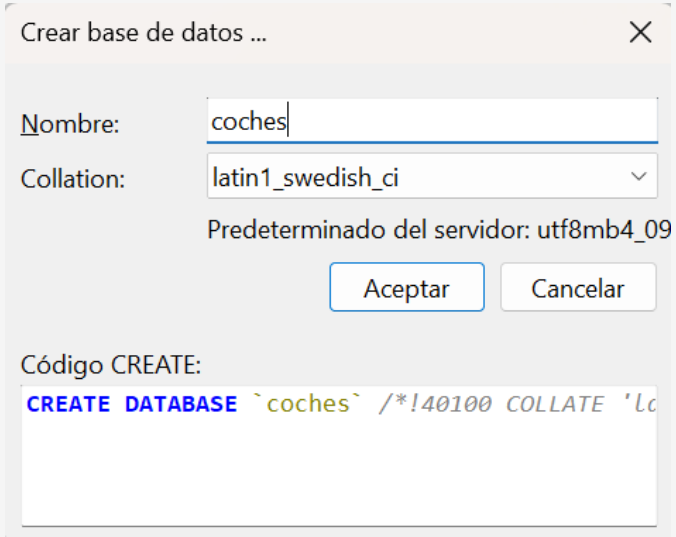
Ahora podemos conectarnos con un cliente como por ejemplo HeidiSQL:



Conectar a mysql desde un cliente externo

- Creamos una base de datos





Conectar a mysql desde un cliente externo

Paramos el contenedor:

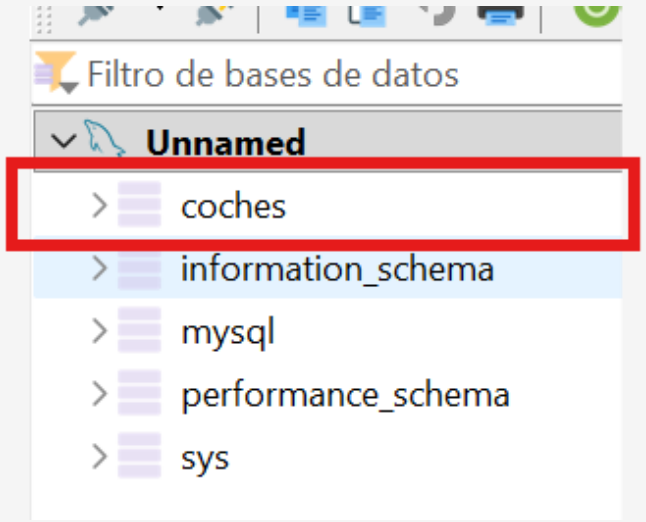
```
docker stop mi-mysql
```

Arrancamos de nuevo el contenedor:

```
docker stop mi-mysql
```

Conectar a mysql desde un cliente externo

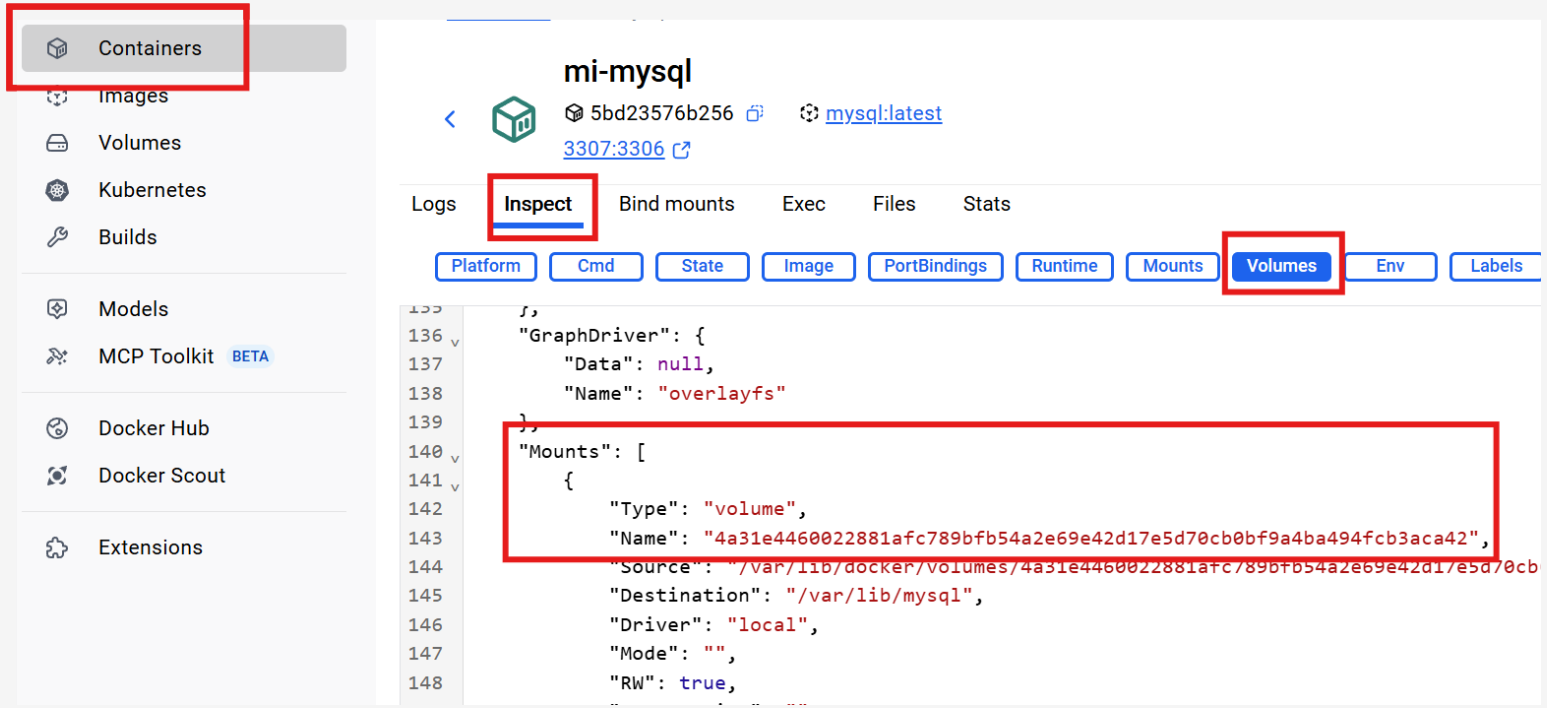
- Si conectamos de nuevo vemos que la base de datos se ha conservado:



- ¿Por qué se conserva si vimos que Docker no guarda los datos entre ejecuciones salvo que usemos algún mecanismo de persistencia de datos como un **volumen** o un **bind mount**?

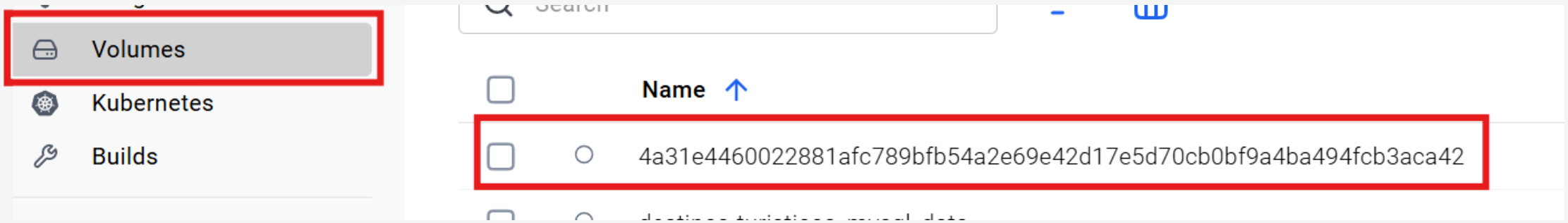
Conectar a mysql desde un cliente externo

- La respuesta es porque la imagen de mysql ya está configurada por defecto para crear un volumen al iniciarse aunque nosotros no se lo asignemos.
- Si vamos a **Containers** -> **Inspect** -> **Volumes** y buscamos **Mounts**



Conectar a mysql desde un cliente externo

- Vemos que el contenedor ha montado el volumen 4a31e44…



Ejercicio

- Busca en **dockerHub** una **imagen** de **Jenkins**
- Consulta la información sobre su uso y descárgala y configúrala para poder usar Jenkins desde un contenedor.
- Documenta todo el proceso en un documento de Word (el objetivo es que puedas llegar a crear una tarea en Jenkins)