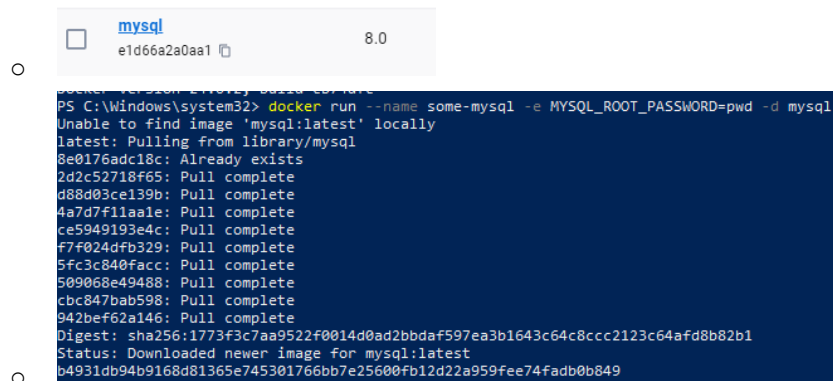


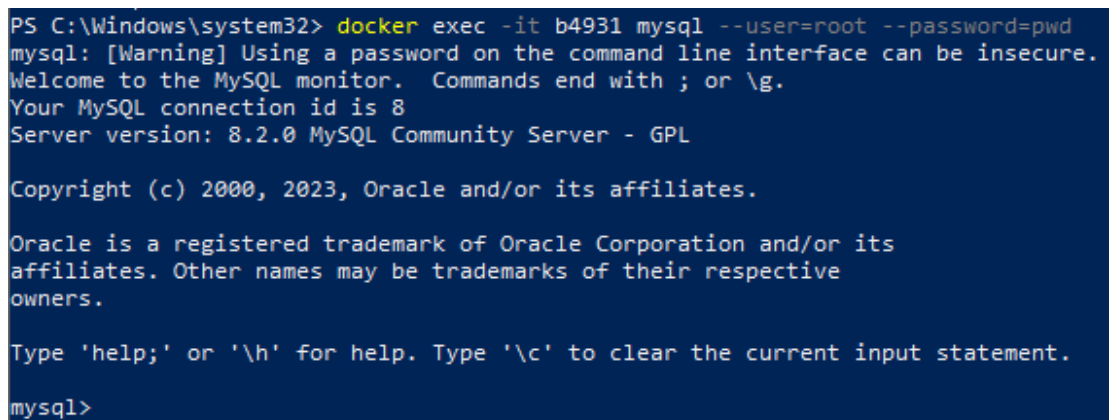
BBDD Y VOLÚMENES

- Vamos a docker hub



The screenshot shows the Docker Hub interface for the 'mysql' image, version 8.0, with ID e1d66a2a0aa1. Below it, a terminal window displays the command `docker run --name some-mysql -e MYSQL_ROOT_PASSWORD=pwd -d mysql` and its output, which includes pulling the latest image from the library and showing the image's SHA256 digest.

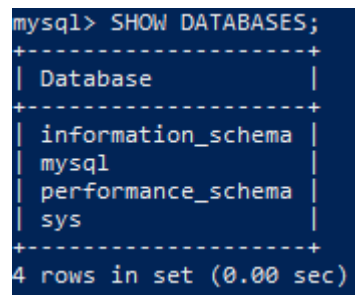
- Ejecutamos



The terminal window shows the command `docker exec -it b4931 mysql --user=root --password=pwd` being executed. The output displays the MySQL welcome message, including the version (8.2.0) and the GPL license. The prompt `mysql>` is shown at the bottom.

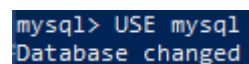
En este punto habéis entrado en mysql

- Comprobad bases de datos



The screenshot shows the `mysql> SHOW DATABASES;` command being executed. The output lists the databases: `information_schema`, `mysql`, `performance_schema`, and `sys`. The prompt `mysql>` is shown at the bottom.

- usemos la base de datos mysql



The screenshot shows the `mysql> USE mysql` command being executed. The output is `Database changed`, indicating that the current database has been switched to 'mysql'.

- miremos las tablas de esta base de datos

```
mysql> SHOW TABLES;
+-----+
| Tables_in_mysql |
+-----+
| columns_priv     |
| component        |
| db               |
| default_roles    |
| engine_cost      |
| func             |
| general_log      |
| global_grants    |
| gtid_executed    |
| help_category    |
| help_keyword     |
| help_relation    |
| help_topic       |
| innodb_index_stats |
| innodb_table_stats |
| ndb_binlog_index |
| password_history  |
| plugin           |
| proc_priv        |
| proxies_priv     |
| replication_asynchronous_connection_failover |
| replication_asynchronous_connection_failover_managed |
| replication_group_configuration_version |
| replication_group_member_actions |
| role_edges       |
| server_cost      |
| servers          |
| slave_master_info |
| slave_relay_log_info |
| slave_worker_info |
| slow_log         |
| tables_priv      |
| time_zone        |
| time_zone_leap_second |
| time_zone_name   |
| time_zone_transition |
| time_zone_transition_type |
| user             |
+-----+
```

- Ejecutemos la consulta

[illegible]

- Comprobad que tenéis el usuario root que hemos creado con su contraseña

```
mysql> SELECT user, host, authentication_string FROM mysql.user WHERE user = 'root';
+-----+-----+-----+
| user | host      | authentication_string |
+-----+-----+-----+
|xq/j9~p|_B##8XAs7y2w8CchVnsuIJMXr2ip5i6DK41eL8Zm9m/kDg2 |
| root | localhost | $A$005$KJSC7"AF4e8dC0a|6'G9UuaxdsMtUt2Fw17t0rH0NfLZsbgIB1R2wkUYX8n.1PA |
+-----+-----+-----+
2 rows in set (0.00 sec)
```

- paremos el contenedor

```
mysql> SHUTDOWN;
Query OK, 0 rows affected (0.00 sec)

mysql>
PS C:\Windows\system32>
```

Volúmenes

```
PS C:\Windows\system32> docker volume ls
DRIVER      VOLUME NAME
local       48cc7a61502d7e7d5adda7b610e087dcbec850241cd7e3a043b1e8c097915ef2
local       mariadb_data
local       mvcsencillo_mysql_data
```

El volumen ha sido creado al crear mysql y persiste aunque no esté corriendo un contenedor.

Si vamos a dockerhub y vemos el dockerfile de mysql podremos ver una línea que pone

VOLUME /var/lib/mysql

- Destruyamos mysql y veamos si el volumen persiste

CAPTURA2 CON EL RESULTADO DE docker ps -a para comprobar que no hay proceso de mysql creado

```
PS C:\Windows\system32> docker ps -a
CONTAINER ID   IMAGE                COMMAND                  CREATED        STATUS        PORTS                    NAMES
0eaff1ac2e52   phpmyadmin:5        "/docker-entrypoint..." 3 hours ago    Exited (0) 27 minutes ago           mvcsencillo-phpmyadmin-1
3dd726aefcaa   mysql:8.0           "/docker-entrypoint..." 3 hours ago    Exited (0) 27 minutes ago           mvcsencillo-mysql-1
647b0903e0a4   mvcsencillo-apache  "/usr/sbin/apache2ct..." 3 hours ago    Exited (137) 26 minutes ago         mvcsencillo-apache-1
6422e0621144   roberto:1.0         "/docker-entrypoint..." 6 days ago     Exited (0) 5 days ago               tar_dockerfile
3c7590b843d0   roberto:1.0         "/docker-entrypoint..." 7 days ago     Exited (0) 6 days ago               roberto_dockerfile
2a136f5dec0f   nginx.dockerfile.roberto:1.0 "/docker-entrypoint..." 7 days ago     Exited (0) 7 days ago               mi-contenedor-nginx
20e1194590fc   ow.servidor-miservicio_php "docker-php-entrypoi..." 2 weeks ago    Exited (0) 7 days ago               miphp
```

Aunque hayamos eliminado mysql el volumen que creó al crear mysql persiste. Comprobado con `docker volumen ls`

- Volvemos a crear pero así

```
docker run --name some-mysql -e MYSQL_ROOT_PASSWORD=pwd -v db:/var/lib/mysql mysql
```

Estará esperando a que introduzcamos un comando

- Ahora creamos un volumen etiquetado como db
- Comprobamos volúmenes

```
PS C:\Windows\system32> docker volume create --name db
db
PS C:\Windows\system32> docker volume ls
DRIVER      VOLUME NAME
local       db
local       mariadb_data
local       mvcsencillo_mysql_data
```

- Hacemos un contenedor nuevo y lo llamamos mysql12
- Comprobamos que tenemos 2 contenedores activos

```
PS C:\Windows\system32> docker run --name some-mysql12 -e MYSQL_ROOT_PASSWORD=pwd -d mysql
50e4ef8c2c0989cdc08f2359b74477d9ff44c424b649a72a437f2fa864132ca1
PS C:\Windows\system32> docker volume ls
DRIVER      VOLUME NAME
local       9961d1ed2b66c0d7c753e0f46de688f96b81cb488e67bc77ef197a35990f0fd6
local       db
local       mariadb_data
local       mvcsencillo_mysql_data
```

- Entramos en el primero creado donde tenemos el volumen db
- Creamos una base de datos llamada animales







```
PS C:\Windows\system32> docker exec -it ab17346455499acf0768ef7951583f2df1d1858cb1bbf643ead37e33cbd12822 mysql --user=root --password=pwd
mysql> CREATE DATABASE animales;
Query OK, 1 row affected (0.02 sec)
```

```
mysql> CREATE DATABASE animales;
Query OK, 1 row affected (0.02 sec)

mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| animales |
| information_schema |
| mysql |
| performance_schema |
| sys |
+-----+
5 rows in set (0.01 sec)
```

- Eliminemos todos los contenedores de mysql (opción -f para forzar la eliminación mientras están ejecutando)

```
PS C:\Windows\system32> docker rm -f ab17346455499acf0768ef7951583f2df1d1858cb1bbf643ead37e33cbd12822
ab17346455499acf0768ef7951583f2df1d1858cb1bbf643ead37e33cbd12822
PS C:\Windows\system32> docker rm -f 50e4ef8c2c0989cdc08f2359b74477d9ff44c424b649a72a437f2fa864132ca1
50e4ef8c2c0989cdc08f2359b74477d9ff44c424b649a72a437f2fa864132ca1
```

<input type="checkbox"/>		tar_dockerfile 6422e0621144	roberto:1.0	Exited	0%	8080:80		6 days ago
<input type="checkbox"/>		roberto_dockerfile 3c7598b843d0	roberto:1.0	Exited	0%	8080:80		7 days ago
<input type="checkbox"/>		mi-contenedor-nginx 2e13bf5dec0f	nginx_docker	Exited	0%	80:80		7 days ago

- Si volvemos a lanzar un contenedor mysql y lo hacemos en un volumen ya existente, lo hará en él en vez de crear uno nuevo

```
docker run --name some-mysql -e MYSQL_ROOT_PASSWORD=pwd -d -v db:/var/lib/mysql mysql
```

- Comprobemos que no ha creado un volumen nuevo

```
PS C:\Windows\system32> docker run --name some-mysql -e MYSQL_ROOT_PASSWORD=pwd -d -v db:/var/lib/mysql mysql
e95078674656527fb374ea5d740b9f2fa53632982ffa72342787726933c3cf79
PS C:\Windows\system32> docker volume ls
DRIVER      VOLUME NAME
local       9961d1ed2b66c0d7c753e0f46de688f96b81cb488e67bc77ef197a35990f0fd6
local       db
local       mariadb_data
local       mvcsencillo_mysql_data
```

- Metámonos en el contenedor creado de mysql y comprobemos que tenemos la bbdd animales que habíamos creado, que en realidad está en el volumen.

```
PS C:\Windows\system32> docker exec -it e95078674656527fb374ea5d740b9f2fa53632982ffa72342787726933c3cf79 mysql --user=root --password=pwd
```

```
mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| animales |
| information_schema |
| mysql |
| performance_schema |
| sys |
+-----+
5 rows in set (0.01 sec)
```