

# Contenerización de bases de datos (MySQL)

Fase	Semana	Día	Lección
6 - Bases de Datos	3	4	4-5

# Objetivos

- Comprender la contenerización de bases de datos MySQL.
- Crear y correr contenedor de base de datos desde terminal y archivo docker compose
- Interactuar con un contenedor de base de datos MySQL

# Creación de un contenedor para BD MySQL

## Sin persistencia de datos

# Creación de un contenedor para BD MySQL

Pasos crear un contenedor para una base de datos sin persistencia de datos

1. Descargar la imagen de la base de datos que vamos a utilizar desde DockerHub

- Correr el siguiente comando

```
docker pull mysql
```

- Para obtener información desde la documentación de la imagen ir al sitio [https://hub.docker.com/\\_/mysql](https://hub.docker.com/_/mysql)
- **NOTA:** Elegir versión más reciente en caso de base de datos nueva o asegurarse de elegir la versión que tenga la base de datos actual para evitar problemas de versión.

# Creación de un contenedor para BD MySQL

2. Correr comando para asegurarse que la imagen se descargó correctamente

`docker image ls`

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
mysql	8.0	82ebbd05b8a9	2 months ago	577MB

3. Correr comando para generar contenedor usando la imagen descargada

`docker run --rm --name mysql -e MYSQL_ROOT_PASSWORD=root -p 3306:3306 mysql:8.0`

Explicación del comando

- `docker run`: comando que nos permite crear un contenedor a partir de una imagen Docker.
- `--rm`: ayuda a que terminemos de utilizar el contenedor, éste se elimine y no ocupe espacio en nuestro disco.

# Creación de un contenedor para BD MySQL

- **--name:** se utiliza para asignarle un nombre a nuestro contenedor. En caso de no usar este parámetro, docker asignará uno automáticamente.
- **-e:** se utiliza para pasarle al contenedor una variable de ambiente.
  - **NOTA:** Revisar las variables de ambiente específicas de la imagen en la documentación ya que algunas de ellas son obligatorias. [https://hub.docker.com/\\_/mysql](https://hub.docker.com/_/mysql)
- **-p:** se utiliza para mapear los puertos entre el contenedor y nuestra máquina local.
  - **NOTA:**
    - El primer número indica el puerto en el que estará disponible la comunicación desde nuestra máquina con el contenedor
    - El segundo número indica el puerto por el que el contenedor estará escuchando para cuando nuestra máquina se comuniquen con él.

# Creación de un contenedor para BD MySQL

- **mysql:8.0**: es el nombre de la imagen y la versión que vamos a utilizar para crear el contenedor. Si no se indica, docker buscará la imagen en Docker Hub.

4. Al correr el comando, en la terminal se comenzarán a ver mensajes como estos

```
:19+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.34-1.el8 started.  
:19+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'  
:19+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.34-1.el8 started.  
:19+00:00 [Note] [Entrypoint]: Initializing database files
```

5. Para comprobar que el contenedor está siendo ejecutado desde otra terminal podemos correr comando **docker ps** el cual mostrará los siguiente:

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
4ff3d699f1f9	mysql:8.0	"docker-entrypoint.s..."	6 seconds ago	Up 5 seconds	33060/tcp, 0.0.0.0:3306->3306/tcp	mysql

# Creación de un contenedor para BD MySQL

6. Para detener el contenedor debemos de

- Copiar el id de la columna CONTAINER ID que nos muestra el resultado de correr `docker ps`

```
CONTAINER ID  
4ff3d699f1f9
```

- Correr el comando `docker stop` seguido del id del contenedor

```
docker stop 4ff3d699f1f9
```



# Creación de un contenedor para BD MySQL

## Con persistencia de datos

# Creación de un contenedor para BD (persistencia datos)

Si queremos que los datos del contenedor sean persistentes (se almacenen en disco) tenemos que crear un volumen donde vamos a indicar el directorio de nuestra máquina local donde vamos a almacenar los datos al que va a acceder nuestro contenedor

Para crear un volumen utilizamos el parámetro `-v`.

Para más información sobre volúmenes consultar: <https://docs.docker.com/storage/volumes/>

A continuación el comando a correr para ejecutar el contenedor con un volumen

```
docker run --rm --name mysql -e MYSQL_ROOT_PASSWORD=root -p 3306:3306 -v mysql_data:/var/lib/mysql mysql:8.0  
-v:
```

La primer parte (antes de los dos puntos) sería el nombre que tendrá el volumen dentro de nuestra máquina local. Este se creará dentro del sistema de archivos que gestiona Docker.

La segunda parte indica el directorio dentro del contenedor que utiliza MySQL para almacenar las bases de datos.

# Creación de un contenedor para BD (persistencia datos)

Podemos listar el volumen creado corriendo el comando

```
docker volume ls
```

DRIVER	VOLUME NAME
local	mysql_data

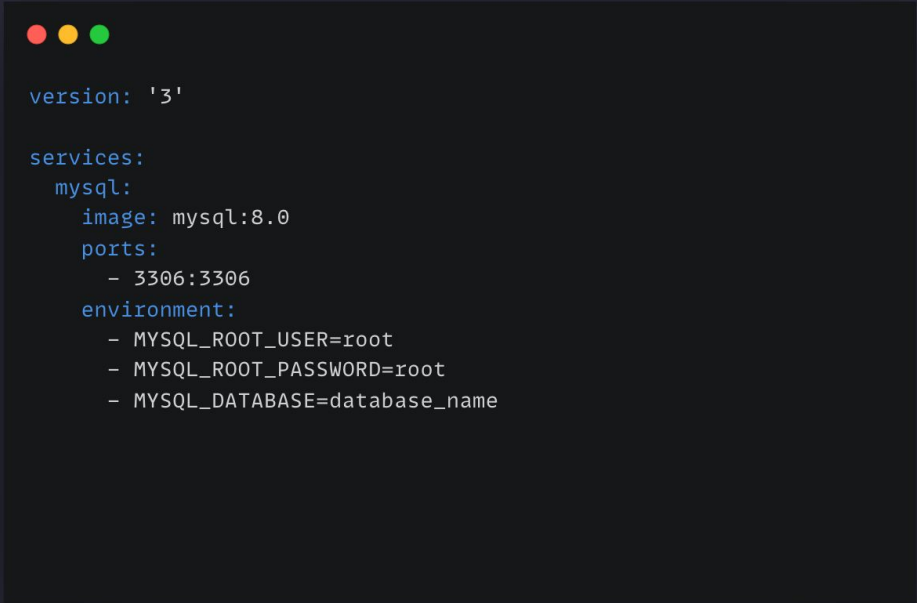
Ahora, al correr el contenedor de esta manera se logra que los datos que se vean afectados (creados, eliminados, actualizados) con la base de datos de este contenedor, se mantendrán incluso si se detiene el contenedor y se vuelve a correr.

# Creación de un contenedor para BD MySQL (Docker Compose)

Sin persistencia de datos

# Creación de un contenedor para BD MySQL (Docker Compose)

Para la creación de un contenedor desde un archivo **docker-compose.yml** necesitamos el siguiente código

A dark-themed terminal window with three colored window control buttons (red, yellow, green) in the top-left corner. It displays a Docker Compose configuration for a MySQL database service.

```
version: '3'

services:
  mysql:
    image: mysql:8.0
    ports:
      - 3306:3306
    environment:
      - MYSQL_ROOT_USER=root
      - MYSQL_ROOT_PASSWORD=root
      - MYSQL_DATABASE=database_name
```

# Creación de un contenedor para BD MySQL (Docker Compose)

El código anterior nos permitirá obtener el mismo comportamiento que obtuvimos al correr el comando de la sección anterior pero desde un archivo docker compose, de esta manera será más sencillo integrar más servicios en el futuro y hacer que se comuniquen entre ellos de manera más sencilla.

Comandos importantes correr, detener el contenedor usando docker compose

- `docker compose up`: inicializa los servicios (contenedores)
- `docker compose down`: detiene los servicios
- `docker-compose ps`: lista los servicios (en caso de que estén corriendo)

# Creación de un contenedor para BD MySQL (Docker Compose)

## Con persistencia de datos

# Creación de un contenedor para BD MySQL (Docker Compose)

Para la creación de un contenedor usando volúmenes para la persistencia de los datos desde un archivo `docker-compose.yml` necesitamos el siguiente código

```
version: '3'

services:
  mysql:
    image: mysql:8.0
    ports:
      - 3306:3306
    environment:
      - MYSQL_ROOT_USER=root
      - MYSQL_ROOT_PASSWORD=root
      - MYSQL_DATABASE=database_name
    volumes:
      - mysql_data:/var/lib/mysql

volumes:
  mysql_data:
```



# Creación de un contenedor para BD MySQL (Docker Compose)

El código anterior nos permitirá obtener el mismo comportamiento corriendo un contenedor/servicio de mysql pero esta vez agregando la sección para los volúmenes

Comandos importantes correr, detener el contenedor usando docker compose

- `docker-compose down -v`: comando para detener los servicios incluidos en el archivo `docker-compose.yml` y además elimina el/los volumen/volúmenes generados.

# Interacciones con bases de datos en contenedores

# Interactuar con MySQL desde contenedor

Para poder interactuar con MySQL (crear bases de datos, tablas, ejecutar queries) dentro de nuestro contenedor necesitamos:

## 1. Correr el contenedor



```
docker run --rm --name mysql -e MYSQL_ROOT_PASSWORD=root -p  
3306:3306 mysql:8.0
```

## 2. Interactuar con el contenedor y correr **mysql** dentro del mismo (importante las credenciales)



```
docker exec -it CONTAINER_ID /usr/bin/mysql -u root --password=root
```

# Conectar a Base de Datos desde DBMS

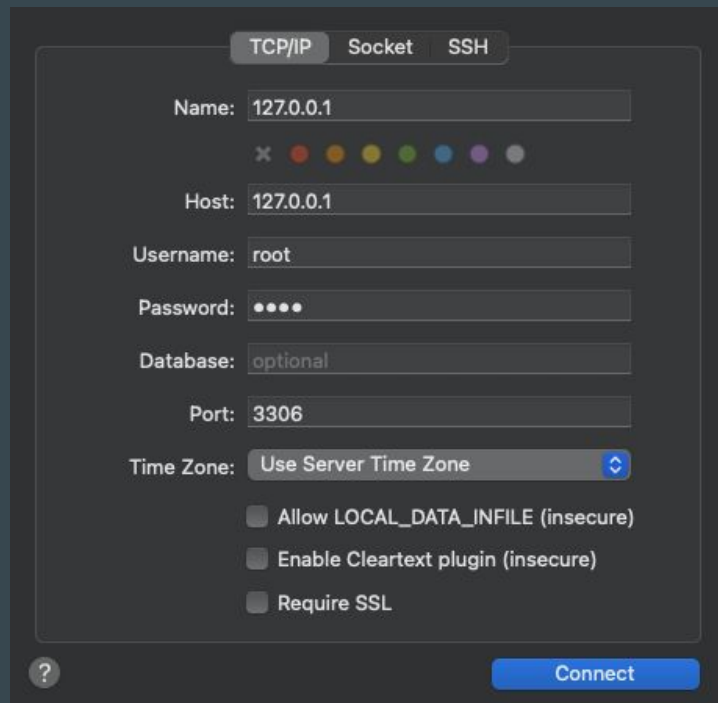
Una vez que el contenedor de MySQL está en ejecución, podemos conectarnos con cualquier cliente de base de datos como: MySQL Workbench, PHPMyAdmin, SequelAce,

...

Los datos de conexión serían:

- Host: 127.0.0.1
- Puerto: 3306
- Usuario: root
- Password: root


**NOTA:** Usuario y password utilizan los default de la imagen, se aconseja crear un usuario distinto y cambiar el password del usuario root.



The image shows a dark-themed window for connecting to a database. At the top, there are three tabs: 'TCP/IP' (selected), 'Socket', and 'SSH'. Below the tabs, there are several input fields and checkboxes. The 'Name' field contains '127.0.0.1'. Below it is a row of seven colored circles (red, yellow, green, blue, purple, grey) with a small 'x' icon to the left. The 'Host' field also contains '127.0.0.1'. The 'Username' field contains 'root'. The 'Password' field contains four dots. The 'Database' field contains 'optional'. The 'Port' field contains '3306'. The 'Time Zone' field is a dropdown menu showing 'Use Server Time Zone' with a blue arrow icon. Below these fields are three checkboxes: 'Allow LOCAL\_DATA\_INFILE (insecure)' (unchecked), 'Enable Cleartext plugin (insecure)' (unchecked), and 'Require SSL' (unchecked). At the bottom right, there is a blue 'Connect' button. At the bottom left, there is a small circular icon with a question mark.

# Exportar Base de Datos desde contenedor

Para poder exportar (o hacer dump) de una base de datos de un contenedor existente podemos correr el siguiente comando



```
docker exec CONTAINER_ID /usr/bin/mysqldump -u root --password=root  
DATABASE_NAME > backup.sql
```

Lo que hace el comando es ejecutar el comando **mysqldump** dentro del contenedor que nos ayudará a exportar la base de datos indicada a un archivo llamado **backup.sql** que se almacenará en el directorio donde estemos corriendo el comando.

# Importar Base de Datos a contenedor

Para poder importar una base de datos (dentro de una archivo sql) primero es necesario crear una base de datos en blanco en nuestro contenedor.

Para crearla podemos “ingresar” al contenedor



```
docker exec -it CONTAINER_ID /usr/bin/mysql -u root --password=root
```

Y ejecutamos el siguiente comando para crearla




```
CREATE DATABASE DATABASE_NAME;
```

# Importar Base de Datos a contenedor

Luego de que hayamos creado la base de datos, podemos correr el siguiente comando.

**NOTA:** **DATABASE\_NAME** debe de ser el mismo nombre usado en el paso anterior

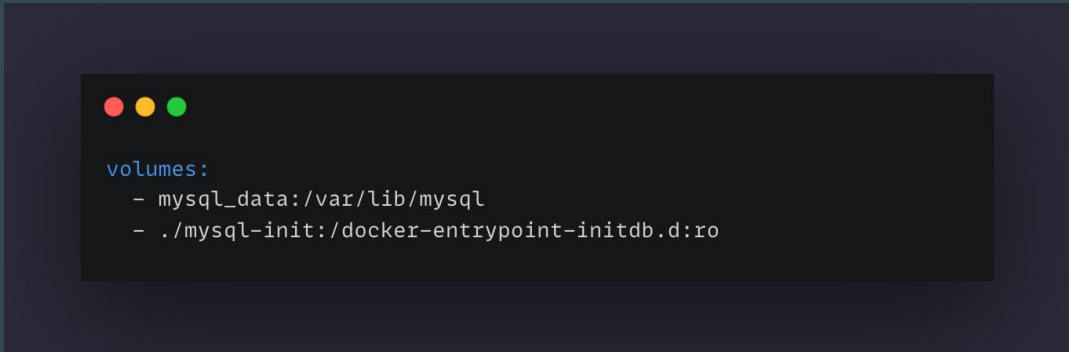


```
docker exec -i CONTAINER_ID /usr/bin/mysql -u root --password=root  
DATABASE_NAME < backup.sql
```

Este comando tomará los datos del archivo **backup.sql** y los almacenará dentro del contenedor en la base de datos indicada.

# Importar Base de Datos a contenedor (Docker Compose)

Para importar un “dump” de una base de datos o script (archivos `.sql` o `.sh`) que queramos correr al iniciar nuestro contenedor por primer vez (instancia fresca) podemos agregarlos dentro de una carpeta (`mysql-init`) el siguiente comando dentro del bloque de `volumes` en nuestro archivo `docker-compose.yml`

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top left corner. It displays the 'volumes' section of a Docker Compose file.

```
volumes:
- mysql_data:/var/lib/mysql
- ./mysql-init:/docker-entrypoint-initdb.d:ro
```

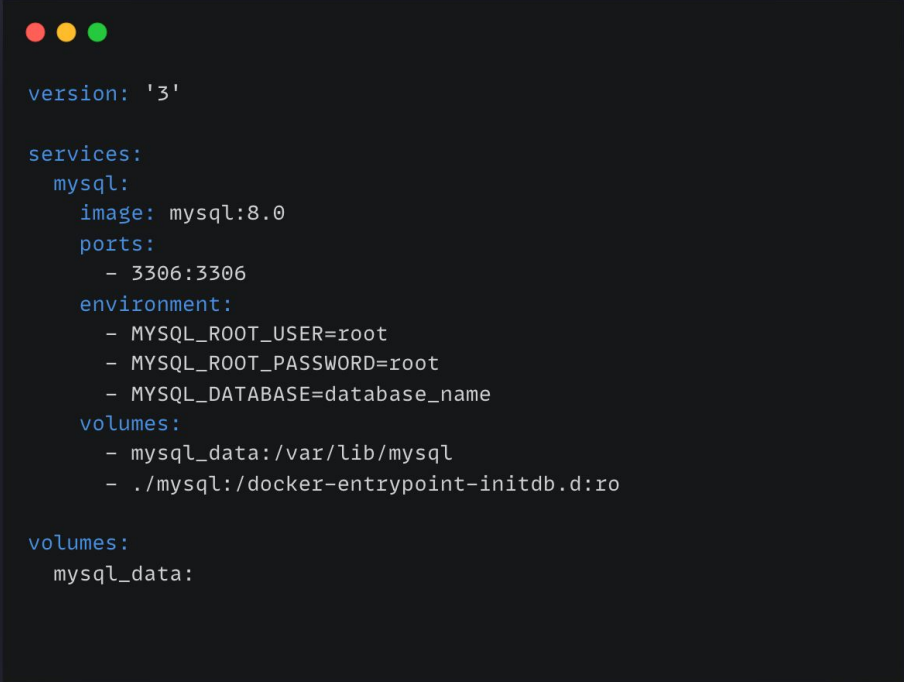
La línea `./mysql-init:/docker-entrypoint-initdb.d:ro` nos indica que en la carpeta `mysql-init` (que se encuentra dentro del mismo directorio padre del `docker-compose.yml`), se encuentra los archivos que queremos correr dentro del contenedor (como el `backup.sql`). La parte de `docker-entrypoint-initdb.d:ro` del comando es quién le indica al contenedor que los debe de correr y que esos archivos deben de ser de lectura.

Leer sección `Initializing a fresh instance` en la documentación de la imagen: [https://hub.docker.com/\\_/mysql](https://hub.docker.com/_/mysql)



# Importar Base de Datos a contenedor (Docker Compose)

Así se vería el archivo final



```
version: '3'

services:
  mysql:
    image: mysql:8.0
    ports:
      - 3306:3306
    environment:
      - MYSQL_ROOT_USER=root
      - MYSQL_ROOT_PASSWORD=root
      - MYSQL_DATABASE=database_name
    volumes:
      - mysql_data:/var/lib/mysql
      - ./mysql:/docker-entrypoint-initdb.d:ro

volumes:
  mysql_data:
```