Instalar y configurar MySQL en Docker

El tutorial incluye conceptos como conectarse a servidores MySQL, ejecutar clientes MySQL para conectarse a contenedores, etc.

Estas son las características de MySql:

- **Relacional:** sigue el modelo relacional y utiliza SQL para gestionar bases de datos.
- **Código abierto (licencia GNU)**
- **Escalable:** puede manejar aplicaciones desde pequeñas hasta de nivel empresarial.
- **Seguro:** ofrece autenticación de usuarios, gestión de acceso y cifrado.
- **Alto rendimiento:** conocido por su velocidad y eficiencia en el manejo de consultas complejas y grandes volúmenes de datos.
- **Replicación y respaldo:** cuenta con opciones de replicación y respaldo de datos, permitiendo estrategias de recuperación ante desastres.

En cuanto al uso de MySQL dentro de contenedores Docker, bueno, eso es solo una combinación hecha en las nubes. Si ha trabajado con Docker antes, todos sus beneficios se aplican también a los contenedores Docker de MySQL:

- **Aislamiento y coherencia:** la instancia de MySQL se aislará de otro software y dependencias, evitando posibles conflictos.
- **Control de versiones:** Docker le permite versionar toda su pila de software junto con MySQL. Esto significa que puede reproducir su entorno en cualquier momento, facilitando el desarrollo y las pruebas.
- **Escalabilidad y gestión de recursos:** con Docker, puede escalar su aplicación MySQL asignando fácilmente más recursos como memoria o CPU.
- **Gestión de dependencias:** Docker encapsula su instancia de MySQL, permitiéndole administrar diferentes versiones sin molestar nada en su máquina local.

Lo más importante de todo es que su aplicación MySQL funcionará en cualquier lugar, no solo en su computadora.

Entonces, hoy aprenderá los fundamentos de la ejecución de MySQL en conjunto con Docker. ¡Empecemos!

Requisitos previos

Como este artículo se centra en los contenedores MySQL Docker, tiene algunos requisitos previos a seguir:

- **Acceso a línea de comando/terminal:** necesita un entorno local con acceso a terminal.
- **Una instancia de Docker en ejecución:** ya debería tener Docker instalado.
- **Familiaridad básica con Docker:** aunque explicaré todos los comandos utilizados en el artículo, un conocimiento básico de Docker aumentará significativamente los beneficios que puede obtener de este artículo.
- **SQL:** lo mismo ocurre con SQL; no explicaré ningún comando SQL utilizado en este tutorial, ya que nos desviará del tema principal.

Descargando la imagen oficial de MySQL Docker

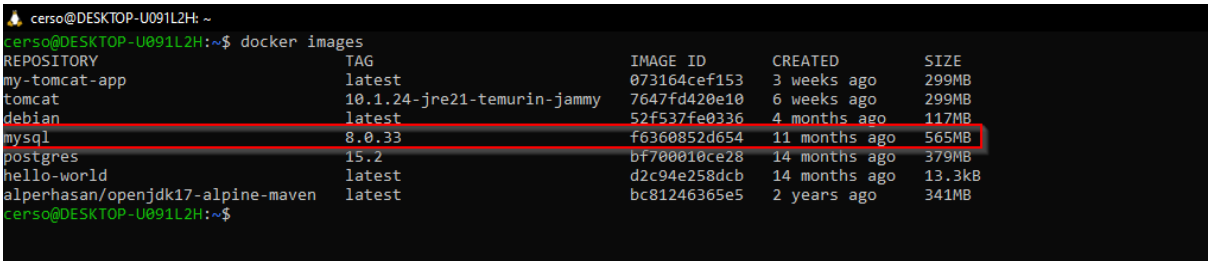
Comenzaremos descargando la imagen oficial de MySQL Docker con el siguiente comando:

$ docker pull mysql:latest

`docker pull` requiere el nombre y la versión de la imagen con `image:version` sintaxis. El uso de la palabra clave `latest` descarga la versión estable más reciente.

Si visita la página oficial de imágenes de MySQL en Docker Hub, podrá ver muchas otras versiones para diferentes propósitos.

En mi caso tengo la version:



Recuerde: las imágenes de Docker son planos para crear contenedores. Así como un plano le permite construir una casa, una imagen de Docker contiene todas las instrucciones y componentes necesarios para crear una instancia en ejecución de una aplicación o servicio.

Si tiene experiencia en **programación orientada a objetos** , piense en las imágenes de Docker como clases. Así como crear una sola clase te permite crear múltiples objetos, las imágenes de Docker te permiten crear múltiples contenedores a partir de ellas.

Ejecutar y administrar un contenedor de servidor MySQL

Ahora, creamos nuestro primer contenedor a partir de la imagen `mysql` . Aquí está el comando que usaremos:

$ docker run --name test-mysql -e MYSQL_ROOT_PASSWORD=strong_password -d mysql

- `run`: crea un nuevo contenedor o inicia uno existente
- `--name CONTAINER_NAME`: le da un nombre al contenedor. El nombre debe ser legible y breve. En nuestro caso el nombre es `test-mysql`.
- `-e ENV_VARIABLE=value`: la etiqueta `-e` crea una variable de entorno a la que se podrá acceder dentro del contenedor. Es fundamental configurarla `MYSQL_ROOT_PASSWORD` para que podamos ejecutar comandos SQL más tarde desde el contenedor. Asegúrese de guardar su contraseña segura en un lugar seguro.
- `-d`: abreviatura de separado, la `-d` etiqueta hace que el contenedor se ejecute en segundo plano. Si elimina esta etiqueta, el comando seguirá imprimiendo registros hasta que el contenedor se detenga.
- `image_name`: el argumento final es el nombre de la imagen a partir del cual se construirá el contenedor. En este caso nuestra imagen es `mysql`.

Si el comando devuelve una larga cadena (el ID del contenedor), significa que el contenedor se ha iniciado. Podes comprobar su estado con `docker ps`:

```
cerso@DESKTOP-U091L2H: ~
cerso@DESKTOP-U091L2H:~$ docker run --name test-mysql -e MYSQL_ROOT_PASSWORD=strong_password -d mysql:8.0.33
07052e0a065087ec13bbd3a61614a82a7c8070cbe598b45a2bc82a81b2cec011
cerso@DESKTOP-U091L2H:~$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS                 NAMES
07052e0a0650   mysql:8.0.33   "docker-entrypoint.s…"   50 seconds ago   Up 50 seconds   3306/tcp, 33060/tcp   test-mysql
cerso@DESKTOP-U091L2H:~$
```

Recuerde: un contenedor Docker es un potente emulador de un sistema operativo. Además de ejecutar MySQL, puedes realizar cualquier tarea que normalmente harías con la terminal de tu computadora desde el contenedor.

Para acceder a la terminal dentro de su contenedor, puede usar el siguiente comando:

$ docker exec -it container_name bash

Esto iniciará una sesión de bash.

```
cerso@DESKTOP-U091L2H:~$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS         PORTS                 NAMES
07052e0a0650   mysql:8.0.33   "docker-entrypoint.s…"   4 minutes ago   Up 4 minutes   3306/tcp, 33060/tcp   test-mysql
cerso@DESKTOP-U091L2H:~$ docker exec -it  test-mysql bash
bash-4.4# mysql -u root -pstrong_password
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.33 MySQL Community Server - GPL

Copyright (c) 2000, 2023, 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> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.00 sec)

mysql>
```

También es posible conectarse al servidor MySQL fuera del contenedor. Por ejemplo, para conectarse desde su máquina host, puede instalar el cliente MySQL manualmente en su sistema.

Detener el contenedor y eliminarlo

$ docker stop test-mysql

test-mysql

$ docker rm test-mysql

A continuación, reiniciamos el contenedor asignando un puerto del contenedor a un puerto en nuestra máquina local:

$ docker run -d --name test-mysql -e MYSQL_ROOT_PASSWORD=strong_password -p 3307:3306 mysql

Este comando hace lo siguiente:

- `-p 3307:3306`: Asigna el puerto 3306 del contenedor (el puerto predeterminado para MySQL) a su puerto local 3307. Esto significa que cualquier tráfico enviado a su puerto local 3307 se reenviará al puerto 3306 del contenedor y se podrá acceder a su servidor MySQL en ese puerto.

- `-d`: Ejecuta el contenedor nuevamente en modo desconectado.
- `--name test-mysql`: Reutiliza el mismo nombre de contenedor "test-mysql".

- `-e MYSQL_ROOT_PASSWORD=strong_password`: establece nuevamente la contraseña de root para el servidor MySQL.

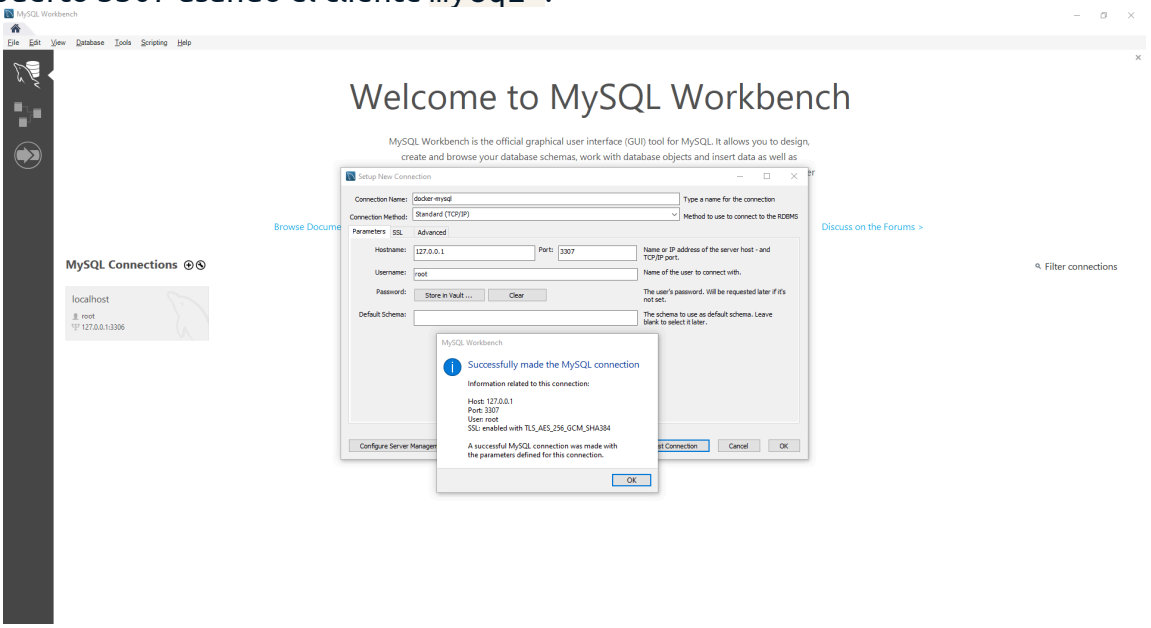
- `mysql`: Especifica la imagen de Docker que se ejecutará, que es la imagen oficial de MySQL.

Después de que la terminal genere una nueva ID para el contenedor, podemos verificar las asignaciones de puertos:

$ docker port test-mysql

3306/tcp -> 0.0.0.0:3307

¡Fue un exito! Ahora, desde su máquina local, puede conectarse al servidor en el puerto 3307 usando el cliente `mysql` :



Cómo conservar los datos almacenados en el contenedor Docker de MySQL

Los datos persistentes almacenados en sus contenedores MySQL son cruciales por muchas razones:

- **Persistencia de datos** : cuando detiene o elimina un contenedor, todos los datos se pierden, incluida su base de datos. Desacoplar los datos del contenedor los hace siempre accesibles.
- **Compartir datos entre contenedores** : separar los datos del contenedor permite que varios contenedores tengan acceso a ellos. De esta manera, puede evitar la duplicación de datos y simplificar la sincronización entre proyectos que utilizan los mismos datos.
- **Portabilidad y respaldo** : los datos persistentes se pueden respaldar y compartir fácilmente de forma independiente, lo que proporciona una forma confiable de recuperarse de la pérdida de datos o la eliminación accidental.
- **Rendimiento y escalabilidad mejorados** : al almacenar los datos a los que se accede con frecuencia en un almacenamiento persistente como SSD, puede mejorar el rendimiento de su aplicación en comparación con confiar en la capa de escritura del contenedor, que normalmente es más lenta.

El proceso es el que ya hemos visto: crearemos un volumen y lo montaremos en el lugar donde se almacenan los datos en nuestro contenedor. Aquí están los pasos:

1. Crea un volumen:

$ docker volume create test-mysql-data

El comando `volume create` crea un almacenamiento dedicado en su sistema de archivos local para el volumen. Una vez montado el volumen, todos los datos del contenedor se vincularán a él.

2. Reinicie el contenedor con el volumen montado:

$ docker stop test-mysql; docker rm test-mysql

$ docker run \

--name test-mysql \

-v test-mysql-data:/var/lib/mysql \

```
  -e MYSQL_ROOT_PASSWORD=strong_password \
  -d mysql
```

Esta vez, la sintaxis tiene este formato: `-v volume_name:directory_in_container`. Todos los volúmenes creados deben montarse en el directorio `/var/lib/mysql` como se especifica en los documentos de imágenes de MySQL.

Entonces, ahora, cualquier base de datos o tabla creada dentro `test-mysql` persistirá localmente, incluso después de que se detenga o elimine el contenedor.

Conclusión

Este artículo cubrió aspectos esenciales de la ejecución y administración de bases de datos MySQL dentro de contenedores Docker. Hemos aprendido cómo descargar y configurar imágenes de MySQL, iniciar servidores MySQL dentro de contenedores, cómo modificar esos contenedores y agregar volúmenes para una configuración personalizada y persistencia de datos.