

Creación de una base de datos MySql con  
Docker.

# Docker

MySql

Rubén Morante González

---

## Tabla de contenido

Introducción .....	<b>¡Error! Marcador no definido.</b>
Desarrollo .....	<b>¡Error! Marcador no definido.</b>
2.1. Entorno utilizado.....	3
2.2. Desarrollo de la práctica.....	3
2.2.1. Añadir base de datos MySql .....	3
2.2.2. Crear la imagen Docker .....	4
2.2.3. Comandos sobre el contenedor .....	5
Conclusiones .....	<b>¡Error! Marcador no definido.</b>

## Tabla de ilustraciones

Ilustración 01 Docker, Wikipedia.....	2
Ilustración 2 docker run .....	4
Ilustración 3 use dbpractica .....	5

# 1. INTRODUCCIÓN

Docker es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores software, proporcionando una capa adicional de abstracción y automatización de Virtualización a nivel de sistema operativo en Linux. Docker utiliza características de aislamiento de recursos del kernel Linux, tales como cgroups y espacios de nombres (namespaces) para permitir que “contenedores” independientes se ejecuten dentro de una sola instancia de Linux, evitando la sobrecarga de iniciar y mantener máquinas virtuales. [1]

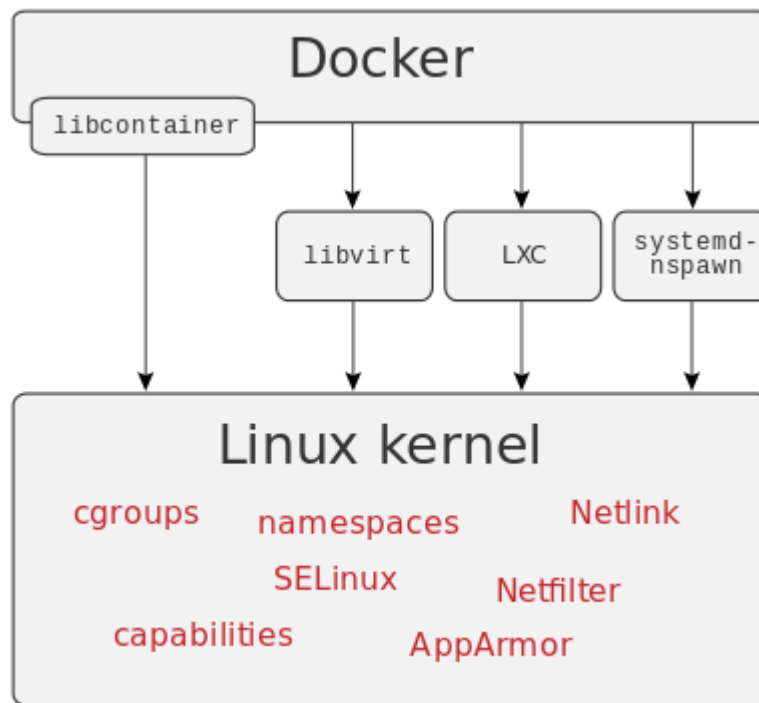


Ilustración 11 Docker, Wikipedia

Docker nos ayuda a no malgastar nuestro tiempo configurando el entorno, y las dependencias del sistema, porque lo vamos a poder desplegar fácilmente, algo muy útil tanto para grandes empresas como para las pequeñas startups.

## 2. DESARROLLO

En este apartado se explica cómo se ha llevado a cabo el desarrollo de la práctica de Docker.

### 2.1. Entorno utilizado

- Asus con Windows 10, 8 Gb de RAM
- Asus con Kali Linux, 4 Gb de RAM
- Spring Tool Suite para el desarrollo del código Java
- Docker

### 2.2. Desarrollo de la práctica

#### 2.2.1. Añadir base de datos MySQL

Utilizando un proyecto anterior cliente servidor con H2, cambiamos la base de datos a MySQL, cambiando las dependencias y el archivo application.yml.

De forma que para la base de datos tendremos las siguientes dependencias:

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
</dependency>
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-entitymanager</artifactId>
</dependency>
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
</dependency>
```

Y el application.yml queda de la siguiente forma:

#### **Cliente**

```
server:
  port: 8443
  ssl:
    key-store: keystore.jks
    key-store-password: password
    key-password: 123456

spring:
  jpa:
    database: MYSQL
    hibernate:
      ddl-auto: create

datasource:
  url: jdbc:mysql://localhost/bdpractica
  username: ruben
  password: 123456
  driver-class-name: com.mysql.jdbc.Driver
```

#### **Servidor**

```
server:
  port: 8080

spring:
  jpa:
    database: MYSQL
    hibernate:
      ddl-auto: validate

datasource:
  url: jdbc:mysql://localhost/bdpractica
  username: ruben
  password: 123456
  driver-class-name: com.mysql.jdbc.Driver
```

## 2.2.2. Crear la imagen Docker

Descargamos las versiones de MySql del siguiente enlace:

<https://github.com/docker-library/mysql>

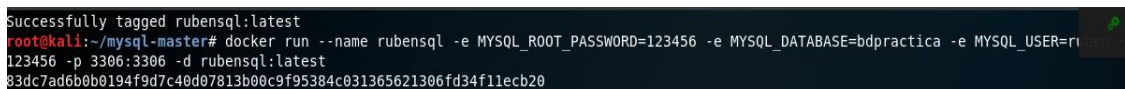
De forma que ahora tenemos la siguiente carpeta, *mysql-master*. En ella se encuentran distintas versiones de MySql, siendo la 5.7 nuestro objeto de estudio.

Lo primero a realizar es crear el contenedor de Docker con nombre 'rubensql' en la carpeta de *mysql-master*, se deben tener permisos de administrador:

```
docker build -t rubensql --label rubensql 5.7/
```

Una vez que tenemos el contenedor de Docker, creamos y metemos una imagen de la base de datos MySql:

```
docker run --name rubensql -e MYSQL_ROOT_PASSWORD=123456 -e
MYSQL_DATABASE=bdpractica -e MYSQL_USER=ruben -e MYSQL_PASSWORD=123456 -p
3306:3306 -d rubensql:latest
```



```
Successfully tagged rubensql:latest
root@kali:~/mysql-master# docker run --name rubensql -e MYSQL_ROOT_PASSWORD=123456 -e MYSQL_DATABASE=bdpractica -e MYSQL_USER=ruben -e MYSQL_PASSWORD=123456 -p 3306:3306 -d rubensql:latest
83dc7ad6b0b0194f9d7c40d07813b00c9f95384c031365621306fd34f11ecb20
```

Ilustración 2 docker run

Ahora solo falta ejecutar la aplicación para que se rellenen las tablas usando el método `initDatabase` en la clase `DatabaseLoader.java` en el cliente. Una vez ejecutado el programa, hay que

comentar el método `initDatabase` y cambiar en el `application.yml` del cliente *ddl-auto: create* por *ddl-auto: validate*. De esta forma solo se rellenan las tablas por defecto una vez, de forma que las siguientes ejecuciones trabajen con una base de datos ya creada.

### 2.2.3. Comandos sobre el contenedor

Con estos pasos ya hemos creado el contenedor con la imagen dentro. Ahora si queremos ver las tablas de la base de datos y el contenido de las mismas podemos abrir una consola de MySQL utilizando la siguiente instrucción:

```
docker exec -it rubensql mysql -uroot -p
```

Si por alguna razón el contenedor no está arrancado, seguiríamos los siguientes pasos:

*Docker images*

Localizaríamos el contenedor que queremos levantar, y en base a su ID llamamos al comando `start` para ponerlo en funcionamiento. Con escribir los primeros números es suficiente. (sustituir ID por el id real del contenedor):

```
docker start ID
```

Podemos verificar que el contenedor está activo utilizando el comando `'ps'`:

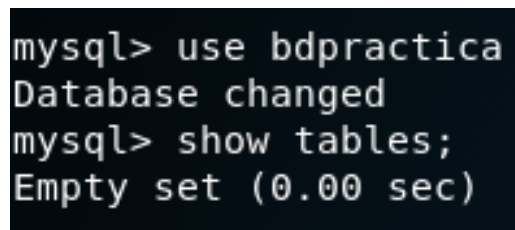
*Docker ps -a*

para borrar un contenedor se usa el comando `'rm'`:

*Docker rm ID*

Ya se dijo anteriormente una forma de levantar la consola de MySQL, una vez en esta podemos hacer consultas. Lo primero será seleccionar la base de datos que vamos a usar, en este caso `'bdpractica'`:

```
use bdpractica;
```



```
mysql> use bdpractica
Database changed
mysql> show tables;
Empty set (0.00 sec)
```

Ilustración 3 use dbpractica

Ahora ya podemos hacer consultas. Un ejemplo de algunas consultas que se pueden hacer:

- *show tables;* (Muestra una lista con todas las tablas de la base de datos)
- *select \* from bdpractica.product;* (Muestra la tabla de productos)
- *select \* from bdpractica.user;* (Muestra la tabla de usuarios)

### 3. CONCLUSIÓN

Docker es una herramienta potente que facilita el uso de aplicaciones en distintos sistemas operativos, pero a la hora de trabajar con Docker hay que tener en cuenta ciertos detalles.

Uno de ellos es que, si se trabaja con Windows, Docker reservará 2Gb de RAM para su uso. Si se quiere tener abierto, por ejemplo, el IDE Spring Tools Suite, Docker y MySQL, se necesitarán más de 4 Gb de RAM, por lo que recomiendo operar con una máquina que disponga 8Gb como mínimo sobre Windows.

## ANEXO 1: REFERENCIAS

[1] Docker Software, Wikipedia, [https://es.wikipedia.org/wiki/Docker\\_\(software\)](https://es.wikipedia.org/wiki/Docker_(software))