



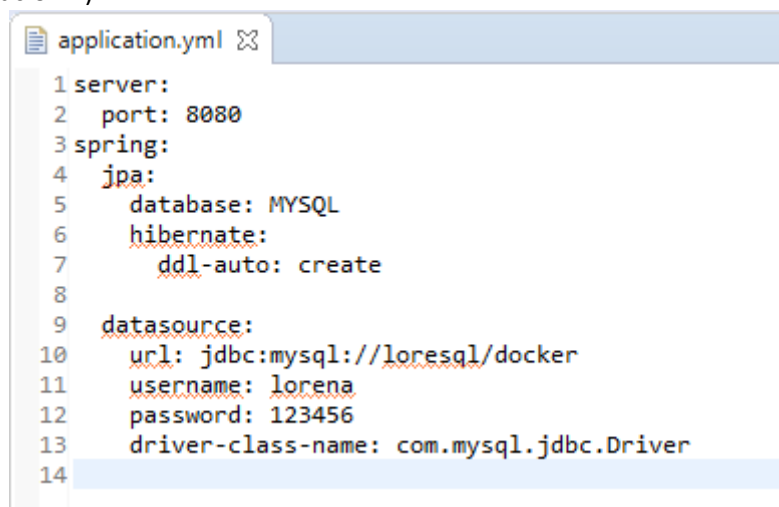
# DOCKER AVANZADO



Lorena Martín Tejera  
4 Diciembre 2017

Los pasos para la realización de la práctica utilizando como sistema operativo Linux han sido los siguientes:

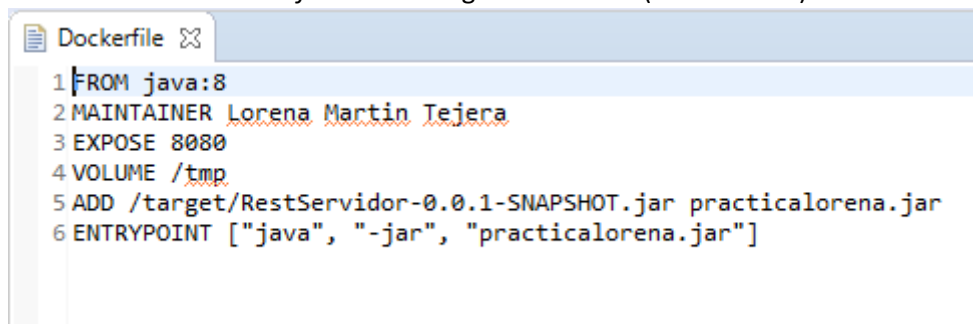
- En primer lugar, se creará el contenedor y la imagen que servirán para el lado cliente mediante los comandos “`docker build -t loresql --label loresql 5.7/`” y “`docker run --name loresql -e MYSQL_ROOT_PASSWORD=123456 -e MYSQL_DATABASE=docker -e MYSQL_USER=Lorena -e MYSQL_PASSWORD=123456 -p 3306:3306 -d loresql:latest`”. Una vez hecho este paso se procede a abrir la consola de mysql para poder realizar las consultas con el comando “`docker exec -it loresql mysql -uroot -p`”.
- Para la parte del servidor, se creará también un contenedor y una imagen. Para realizarlo primero es necesario:
  - Añadir en el pom.xml a continuación del “`artifactId`” la siguiente línea “`<packaging>jar</packaging>`” para poder realizar el jar del servidor.
  - Modificar el “`application.yml`” como se muestra en la siguiente imagen (Ilustración 1).

A screenshot of a code editor showing the content of a file named 'application.yml'. The file contains a YAML configuration for a server. The configuration includes a 'server' section with a 'port' of 8080, a 'spring' section with a 'jpa' subsection containing 'database: MYSQL', 'hibernate' subsection with 'ddl-auto: create', and a 'datasource' subsection with 'url: jdbc:mysql://loresql/docker', 'username: lorena', 'password: 123456', and 'driver-class-name: com.mysql.jdbc.Driver'. The lines are numbered 1 through 14.

```
1 server:
2   port: 8080
3 spring:
4   jpa:
5     database: MYSQL
6     hibernate:
7       ddl-auto: create
8
9   datasource:
10    url: jdbc:mysql://loresql/docker
11    username: lorena
12    password: 123456
13    driver-class-name: com.mysql.jdbc.Driver
14
```

Ilustración 1: application.yml del servidor

- Crear el archivo “`Dockerfile`” con los siguientes datos (Ilustración 2).

A screenshot of a code editor showing the content of a file named 'Dockerfile'. The file contains a Dockerfile configuration for a Java application. The configuration includes a 'FROM' statement with 'java:8', a 'MAINTAINER' statement with 'Lorena Martin Tejera', an 'EXPOSE' statement with '8080', a 'VOLUME' statement with '/tmp', an 'ADD' statement with '/target/RestServidor-0.0.1-SNAPSHOT.jar' and 'practicalorena.jar', and an 'ENTRYPOINT' statement with '["java", "-jar", "practicalorena.jar"]'. The lines are numbered 1 through 6.

```
1 FROM java:8
2 MAINTAINER Lorena Martin Tejera
3 EXPOSE 8080
4 VOLUME /tmp
5 ADD /target/RestServidor-0.0.1-SNAPSHOT.jar practicalorena.jar
6 ENTRYPOINT ["java", "-jar", "practicalorena.jar"]
```

Ilustración 2: Dockerfile del servidor

- Por último, hacer en eclipse mediante Maven Build → “`clean install verify package`” para crear el jar.

- Una vez hecho los pasos anteriores, lo siguiente que hay que hacer es abrir otra terminal y moverse hasta la ubicación del “*dockerfile*” del servidor y realizar los comandos “*docker build -t practicalorena .*” y “*docker run --name practicalorena -p 8080:8080 --link loresql -d practicalorena*” para poder crear la imagen y el contenedor de la parte servidor de nuestra aplicación (Ilustración 3).

```

root@sephit: ~/Programas/WSEclipse/Lorena/EjercicioTema4_2RestServidor
File Edit View Search Terminal Help
EjercicioTema3_6RestCliente EjercicioTema4_2RestCliente
EjercicioTema3_6RestServidor EjercicioTema4_2RestServidor
root@sephit:~/Programas/WSEclipse/Lorena# cd EjercicioTema4_2Rest
bash: cd: EjercicioTema4_2Rest: No such file or directory
root@sephit:~/Programas/WSEclipse/Lorena# cd EjercicioTema4_2RestServidor
root@sephit:~/Programas/WSEclipse/Lorena/EjercicioTema4_2RestServidor# ls
Dockerfile pom.xml src target
root@sephit:~/Programas/WSEclipse/Lorena/EjercicioTema4_2RestServidor# docker build -t practicalorena .
Sending build context to Docker daemon 28.16MB
Step 1/6 : FROM java:8
----> d23bdf5b1b1b
Step 2/6 : MAINTAINER Lorena Martin Tejera
----> Running in bdc696c5e3fc
----> 4ecede55e72
Removing intermediate container bdc696c5e3fc
Step 3/6 : EXPOSE 8080
----> Running in 9aac95dfa7bc
----> 15de73816e98
Removing intermediate container 9aac95dfa7bc
Step 4/6 : VOLUME /tmp
----> Running in 03cd0c0b98ac
----> 90c8833d1383
Removing intermediate container 03cd0c0b98ac
Step 5/6 : ADD /target/RestServidor-0.0.1-SNAPSHOT.jar practicalorena.jar
----> 48f4ad79d52
Removing intermediate container d4e91671ee6c
Step 6/6 : ENTRYPOINT java -jar practicalorena.jar
----> Running in 95fcc79bc7e9
----> 939e669a67dc
Removing intermediate container 95fcc79bc7e9
Successfully built 939e669a67dc
Successfully tagged practicalorena:latest
root@sephit:~/Programas/WSEclipse/Lorena/EjercicioTema4_2RestServidor# docker run --name practicalorena -p 8080:8080 --link loresql -d practicalorena
root@sephit:~/Programas/WSEclipse/Lorena/EjercicioTema4_2RestServidor#

```

Ilustración 3: crear el contenedor y la imagen del servidor

- Para probar que todo es correcto, se arranca el cliente en eclipse (Ilustración 4).

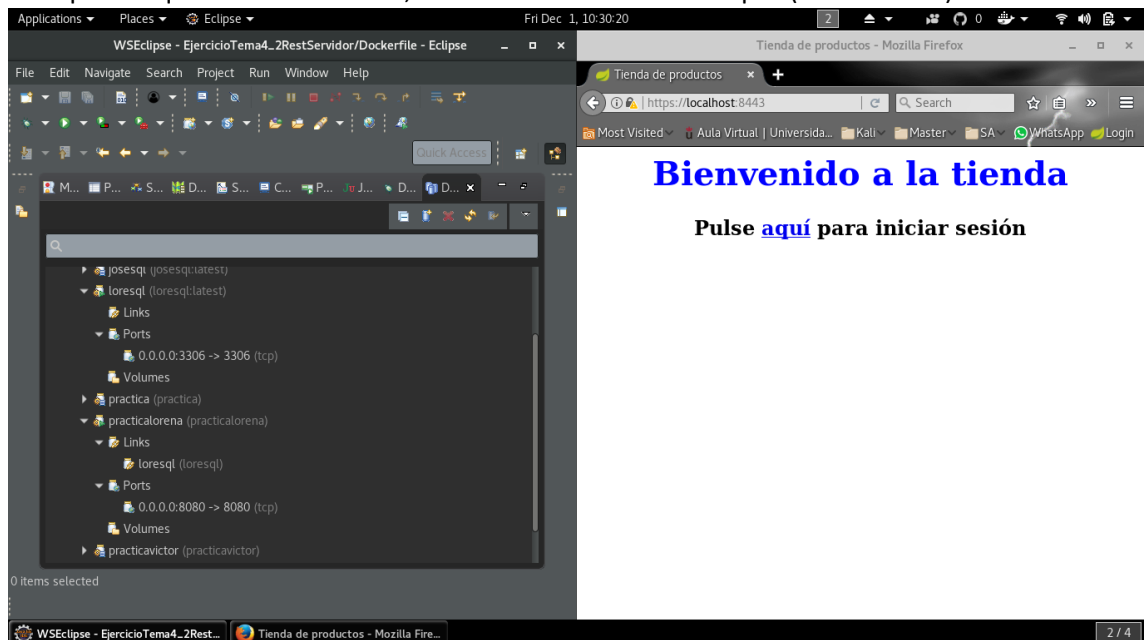


Ilustración 4: servidor arrancado mediante Docker y cliente mediante eclipse

**NOTA:** Como en la práctica anterior, es necesario modificar el “*application.yml*” para que pueda crear los usuarios una única vez sin repeticiones.

- Una vez arrancado el cliente, se introduce en el navegador “localhost:8080/listaProductos” para comprobar que la lista de productos desde el servidor está vacía inicialmente (Ilustración 5).

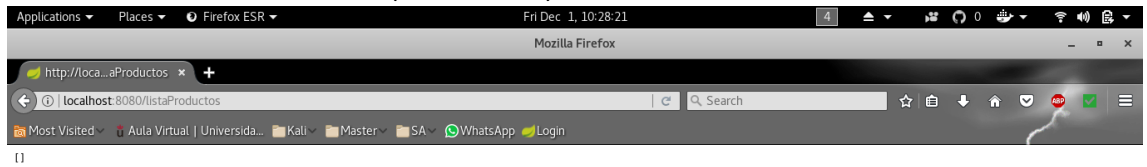


Ilustración 5: lista de productos almacenada en el servidor

- Se crea un producto desde el lado cliente de la aplicación y mediante la consola de mysql hablada en el primer paso, se comprueba que los productos se crean de manera correcta al igual que los usuarios (Ilustración 6).

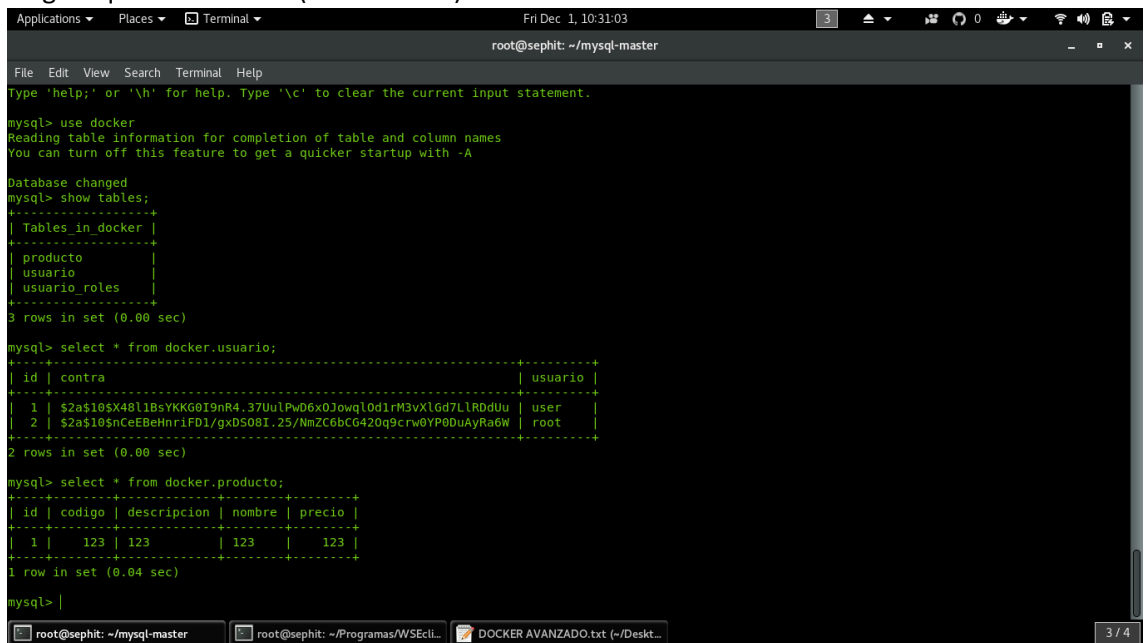


Ilustración 6: consola mysql

- Para subir la imagen del servidor y de la base de datos a Docker Hub, lo primero que hay que hacer es crear una cuenta y el repositorio donde se almacenará ambas imágenes y a continuación, se deben realizar los siguientes comandos en la terminal:
  - `"docker login"` para iniciar sesión en la cuenta de Docker Hub (a continuación, pedirá el nombre de usuario y la contraseña).
  - `"docker tag practicalorena evaurjc2017/lorena:servidor"` para almacenar la imagen del servidor y `"docker tag loresql evaurjc2017/lorena:basedatos"` para almacenar la imagen de la base de datos en el repositorio especificado.
  - `"docker push evaurjc2017/lorena"` para subir a Docker Hub las imágenes.

Una vez hecho los pasos anteriores, aparecen las imágenes en Docker Hub (<https://hub.docker.com/r/evaurjc2017/lorena/>) (Ilustración 7).

PUBLIC REPOSITORY

**evaurjc2017/lorena** ☆

Last pushed: a minute ago

---

Repo Info   Tags   Collaborators   Webhooks   Settings

Tag Name	Compressed Size	Last Updated
servidor	269 MB	a minute ago
basedatos	144 MB	2 minutes ago

Ilustración 7: repositorio de Docker Hub