Lab. 01 Redes en Docker

# Despliegue el CMS Wordpress

En este ejemplo vamos a desplegar el CMS Wordpress, el creador de páginas web más popular del mundo.

|  |  |
| --- | --- |
|  | Un sistema de gestión de contenidos o CMS es un programa informático que permite crear un entorno de trabajo para la creación y administración de contenidos, principalmente en páginas web. |
|  | **Ejecutar desde la consola de WLS** |

Aquí los comandos necesarios para montar el entorno de Wordpress sobre dos contenedores conectados a la red, red\_wp.

$ docker network create red\_wp

$ docker run -d --name servidor\_mysql \

--network red\_wp \

-v /opt/mysql\_wp:/var/lib/mysql \

-e MYSQL\_DATABASE=bd\_wp \

-e MYSQL\_USER=user\_wp \

-e MYSQL\_PASSWORD=asdasd \

-e MYSQL\_ROOT\_PASSWORD=asdasd \

mariadb

$ docker run -d --name servidor\_wp \

--network red\_wp \

-v /opt/wordpress:/var/www/html/wp-content \

-e WORDPRESS\_DB\_HOST=servidor\_mysql \

-e WORDPRESS\_DB\_USER=user\_wp \

-e WORDPRESS\_DB\_PASSWORD=asdasd \

-e WORDPRESS\_DB\_NAME=bd\_wp \

-p 80:80

wordpress

El contenedor **servidor\_mysql** ejecuta un script **docker-entrypoint.sh** que es el encargado, a partir de las variables de entorno, configurar la base de datos y termina ejecutando el servidor mariadb.

Al crear la imagen **mariadb** han tenido en cuenta de que tiene que permitir la conexión desde otra máquina, por lo que en la configuración tenemos comentado el parámetro **bind-address.**

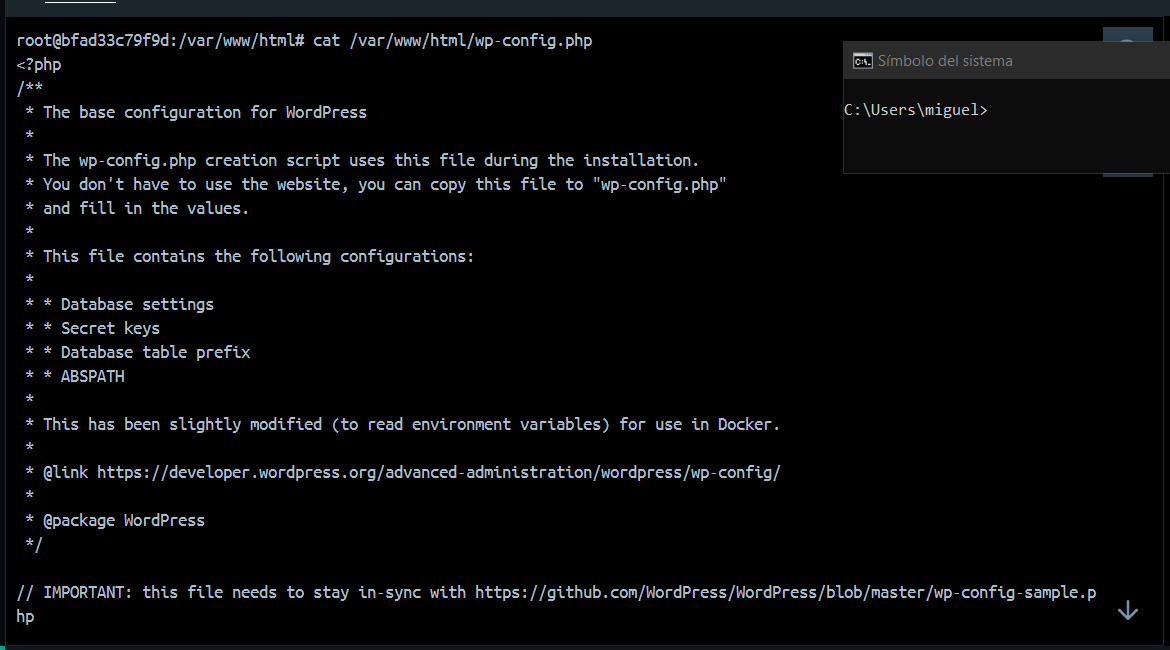
Del mismo modo el contenedor **servidor\_wp** ejecuta un script **docker-entrypoint.sh**, que entre otras cosas, a partir de las variables de entorno, ha creado el fichero **wp-config.php** de wordpress, por lo que durante la instalación no te ha pedido las credenciales de la base de datos.

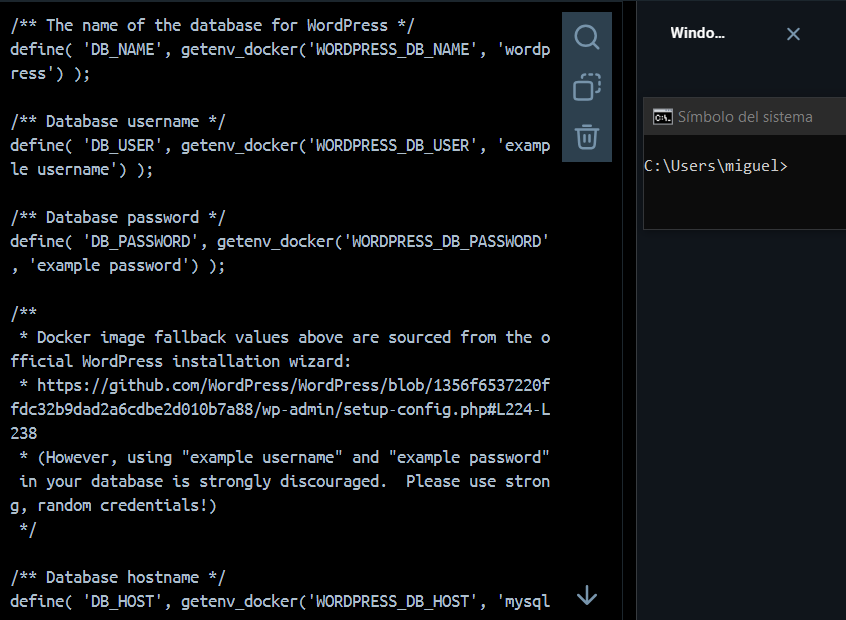
Si te das cuenta la variable de entorno **WORDPRESS\_DB\_HOST** la hemos inicializado al nombre del servidor de base de datos. Como están conectada a la misma red definida por el usuario, el contenedor wordpress al intentar acceder al nombre **servidor\_mysql** estará accediendo al contenedor de la base de datos.

Al servicio al que vamos a acceder desde el exterior es al servidor web, es por lo que hemos mapeado los puertos con la opción **-p**. Sin embargo, en el contenedor de la base de datos no es necesario mapear los puertos porque no vamos a acceder a ella desde el exterior. Sin embargo, el contenedor **servidor\_wp** puede acceder al puerto 3306 del **servidor\_mysql** sin problemas ya que están conectados a la misma red.

1. Ejecuta una instrucción docker para visualizar el contenido del fichero `wp-config.php` y verifica que los parámetros de conexión a la base de datos son los mismo que los indicados en las variables de entorno.

##### Agrega un pantallazo





Monto el entorno de wordpres:

docker run -d --name servidor\_mysql --network red\_wp -v /opt/mysql\_wp:/var/lib/mysql -e MYSQL\_DATABASE=bd\_wp -e MYSQL\_USER=user\_wp -e MYSQL\_PASSWORD=asdasd -e MYSQL\_ROOT\_PASSWORD=asdasd mariadb

docker run -d --name servidor\_wp --network red\_wp -v /opt/wordpress:/var/www/html/wp-content -e WORDPRESS\_DB\_HOST=servidor\_mysql -e WORDPRESS\_DB\_USER=user\_wp -e WORDPRESS\_DB\_PASSWORD=asdasd -e WORDPRESS\_DB\_NAME=bd\_wp -p 8080:80 wordpress

Para activar la red red\_wp hago el siguiente comando: docker network create red\_wp

Después verifico que todo ha ido bien: docker network ls

Después conecto los contenedores a la red:

docker network connect red\_wp servidor\_wp

docker network connect red\_wp servidor\_mysql

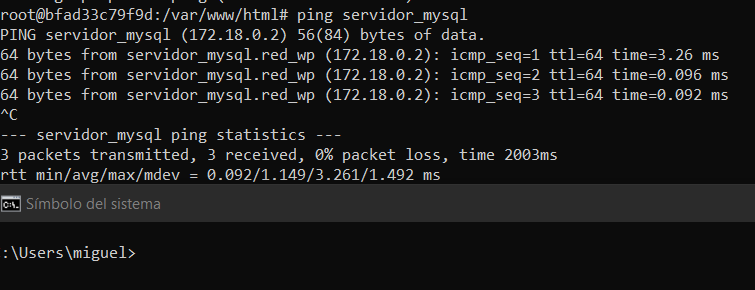
Reinicio los ordenadores:

docker restart servidor\_mysql

docker restart servidor\_wp

1. Ejecuta una instrucción docker para comprobar que desde wl `servidor\_wp` podemos hacer ping usando el nombre `servidor\_mysql`. (Tendrás que instalar el paquete `iputils-ping` en el contenedor).

##### Agrega un pantallazo



Accedo al contenedor servidor\_wp: docker exec -it servidor\_wp bash

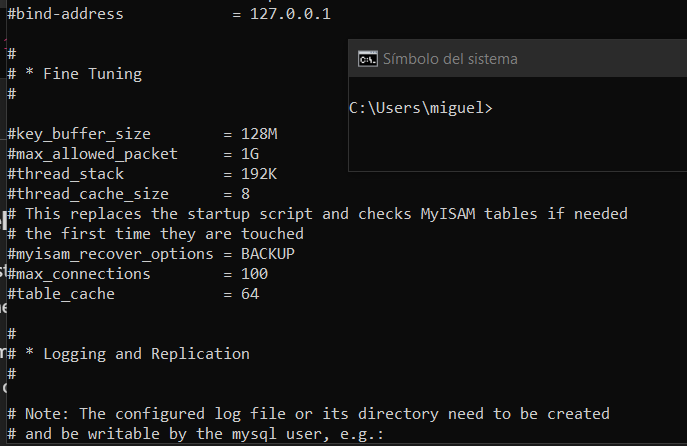
Instalo iputils-ping para hecer ping: apt update && apt install -y iputils-ping

Hago ping: ping servidor\_mysql

Salgo:exit

1. Visualiza el fichero `/etc/mysql/mariadb.conf.d/50-server.cnf` del contenedor con la base de datos y comprueba cómo está configurado el parámetro `bind-address`.

##### Agrega un pantallazo



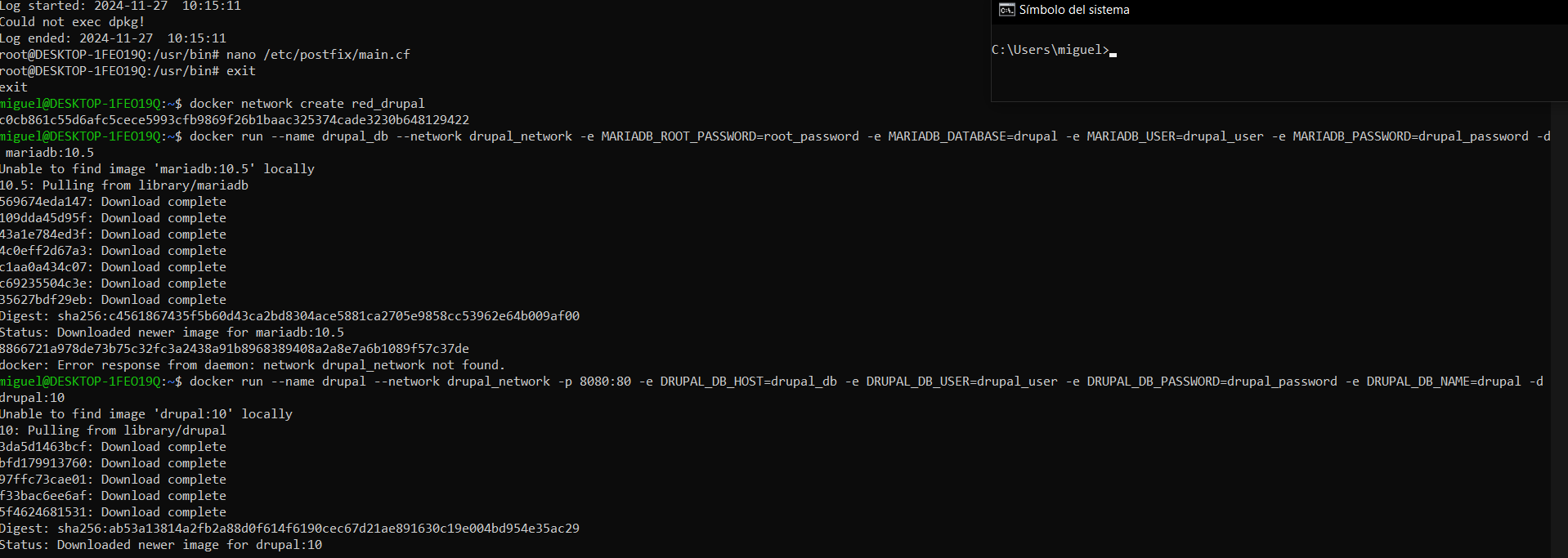
Accedo al contenedor servidor\_mysql: docker exec -it servidor\_mysql bash

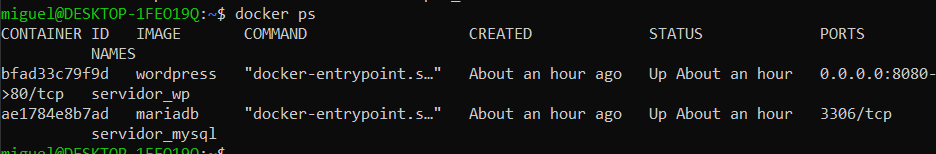
Visualizo la configuración: cat /etc/mysql/mariadb.conf.d/50-server.cnf

Bind-addres = 127.0.0.1 significa que el servidor de MySQL acepta solo las conexiones locales en el mismo servidor

1. Instala Drupal (CMS PHP) siguiendo la documentación de Docker Hub de la aplicación seleccionada.

##### Agrega un pantallazo de los comandos ejecutados





Creo una red en docker: docker network create red\_drupal

Creo los contenedores y los ejecuto:

docker run --name drupal\_db --network drupal\_network -e MARIADB\_ROOT\_PASSWORD=root\_password -e MARIADB\_DATABASE=drupal -e MARIADB\_USER=drupal\_user -e MARIADB\_PASSWORD=drupal\_password -d mariadb:10.5

docker run --name drupal --network drupal\_network -p 8080:80 -e DRUPAL\_DB\_HOST=drupal\_db -e DRUPAL\_DB\_USER=drupal\_user -e DRUPAL\_DB\_PASSWORD=drupal\_password -e DRUPAL\_DB\_NAME=drupal -d drupal:10

miro que los contenedores estan en ejecución: docker ps