Introducción a Docker

# UD 06. Caso práctico 01 - Wordpress + MariaDB






Autor: Sergi García Barea

Actualizado Febrero 2025

## Licencia



## Nomenclatura

A lo largo de este tema se utilizarán distintos símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:

**📖 Importante**

**❕ Atención**

**💬 Interesante**

# UD06. CASO PRÁCTICO 01

## 1. INTRODUCCIÓN

En este caso práctico vamos a poner en marcha el popular CMS Wordpress. Para ello usaremos un fichero "***docker-compose.yml***" comentado que nos pondrá en marcha dos contenedores: el primero utilizando "***Apache + PHP***", junto con una versión instalada de **Wordpress**, mientras que el segundo contendrá un servidor de bases de datos MariaDB. Este ejemplo es similar al propuesto como ejemplo durante el contenido de la unidad.

## 2. FICHERO "DOCKER-COMPOSE.YML" DEL CASO PRÁCTICO

El contenido del fichero "***docker-compose.yml***" que incluimos comentado, es el siguiente:

```
#Versión del fichero docker-compose 3.9. No obligatorio desde la versión de docker-compose 1.27.0
version: "3.9"

#Indicamos los servicios a lanzar
services:
  #Plantill del servicio "db"
  db:
        #Se basa en la imagen "mariadb", version 10.11.2
        image: mariadb:10.11.2
        #Mapea en el volumen "db_data" el directorio "/var/lib/mysql", lo que da persistencia al contenido
de
        #Wordpress almacenado en la base de datos
        volumes:
        - db_data:/var/lib/mysql
        #Indica que siempre que el servicio finalice, se reiniciará
        restart: always
        #Define un conjunto de variables de entorno para estos contenedores,
        #indicando password de root de mariadb, nombre de base de datos,
        # usuario con permisos root (necesario para conexiones remotas) y password de ese usuario
        environment:
        MARIADB_ROOT_PASSWORD: somewordpress
        MARIADB_DATABASE: wordpress
        MARIADB_USER: wordpress
        MARIADB_PASSWORD: wordpress
  #Plantilla del servicio "wordpress"
  wordpress:
        #Indicamos que para lanzar este servicio, debe estar en marcha "db"
        depends_on:
        - db
        #Indicamos que basa en la imagen "wordpress", version "latest"
        image: wordpress:latest
        #Indicamos que el puerto 80 del contenedor se mapea con el puerto 8000 del anfitrion
        ports:
        - "8000:80"
        #Indica que siempre que el servicio finalice, se reiniciará
        restart: always
        #Definimos "variables de entorno". Definimos donde conectarnos a la base de datos,
        #usuario de la base de datos, password de la base de datos y nombre de la base de datos
        environment:
        WORDPRESS_DB_HOST: db:3306
        WORDPRESS_DB_USER: wordpress
        WORDPRESS_DB_PASSWORD: wordpress
        WORDPRESS_DB_NAME: wordpress
        #Indicamos los volúmenes creados y compartidos a lo largo del fichero docker-compose.yml
volumes:
  db_data:
```

Este fichero se ha explicado en detalle de forma didáctica durante la unidad, pero aquí repasamos las claves más importantes:

- Indicamos que los servidores de bases de datos ("***db***")
  - Enlacen su información a un volumen, dotándolo de persistencia.
  - Definan una serie de variables de entorno definiendo usuarios, contraseñas y bases de datos a usar.
- Indicamos que los servidores con Apache + PHP + Wordpress ("***wordpress***"):
  - Para iniciarse, debe iniciarse antes un servicio "***db***".
  - Establece variables de entorno definiendo valores para la conexión de base de datos de Wordpress.
  - Enlaza puerto 80 del contenedor a puerto 8000 del anfitrión.

## 3. PASO 1: PONIENDO EN MARCHA EL SISTEMA

Para poner en marcha este sistema, simplemente nos situamos en el directorio donde tengamos el fichero "***docker-compose.yml***" de este caso práctico y escribimos:

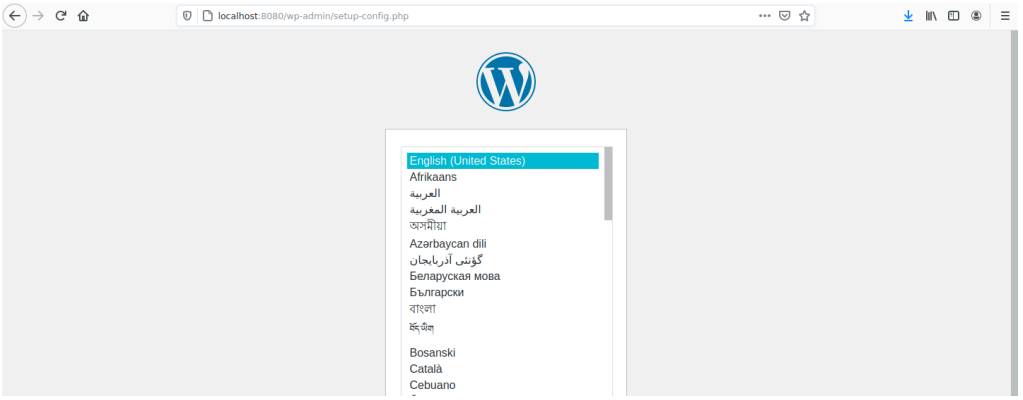
```
docker compose up -d
```

La opción "***-d***" indica que "***Docker Compose***" se ejecute en segundo plano.

La opción "***up***", descarga y construye imágenes (si no estaban ya). Tras ello lanza los contenedores asociados, siguiendo orden de dependencia.

```
sergi@casa:~/Desktop/miwordpress$ docker compose up -d
[+] Running 31/2
 ⠿ db Pulled
 ⠿ wordpress Pulled
[+] Running 3/3
 ⠿ Network miwordpress_default        Created
 ⠿ Container miwordpress-db-1         Started
 ⠿ Container miwordpress-wordpress-1  Started
```

Tras ello, podemos probar que todo es correcto accediendo a http://localhost:8000 donde veremos algo similar a:

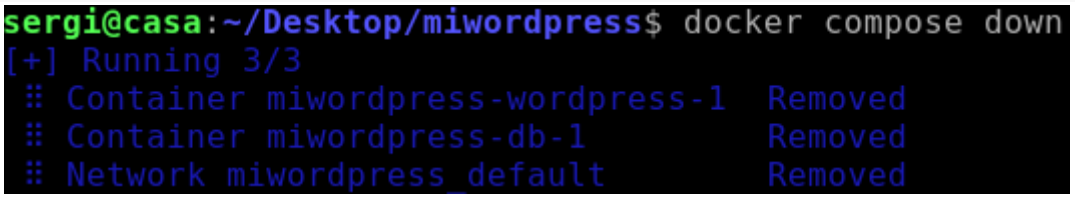



Y simplemente podremos proseguir a poner en marcha nuestro sitio Wordpress.

## 4. PASO 2: PARANDO EL SISTEMA

Parar el sistema es tan sencillo como utilizar el comando

```
docker compose down
```

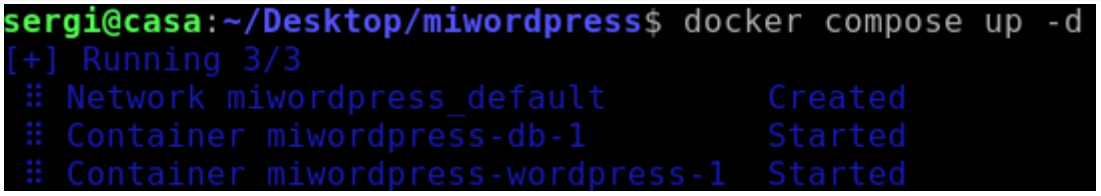



Con ello se pararán y eliminarán los contenedores. No se eliminarán ni las imágenes ni los volúmenes (el sistema Wordpress mantiene la persistencia, al tener mapeados la información de la base de datos a un volumen).

## 5. PASO 3: RE-LANZANDO EL SISTEMA

Relanzar el sistema es tan sencillo como volver a lanzar el comando

```
docker compose up -d
```




Observamos que el sistema aprovecha las imágenes ya creadas para acelerar el proceso de puesta en marcha, simplemente creando y lanzando los contenedores.

## 6. BIBLIOGRAFÍA

[1] Docker Docs https://docs.docker.com/

[2] Docker Compose Docs https://docs.docker.com/compose/