Introducción a Docker  
UD 06. Caso práctico 01 - Wordpress + MySQL

short line

Autor: Sergi García Barea

Actualizado Abril 2022

Licencia

**Reconocimiento – NoComercial - CompartirIgual (BY-NC-SA)**: No se permite un uso comercial de la obra original ni de las posibles obras derivadas, la distribución de las cuales se debe hacer con una licencia igual a la que regula la obra original.

Nomenclatura

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

📖 **Importante**

❕ **Atención**

💬 **Interesante**

**1.** [**Introducción**](#_dbh0n1vac4c8) **3**

**2.** [**Fichero “docker-compose.yml” del caso práctico**](#_vyhbfp4t666x) **3**

**3.** [**Paso 1: Poniendo en marcha el sistema**](#_3b05p2f899fh) **4**

**4.** [**Paso 2: Parando el sistema**](#_9p0xnot11ut4) **5**

**5.** [**Paso 3: Re-lanzando el sistema**](#_i551rgm75761) **5**

**6.** [**Bibliografía**](#_72yv1gty1dke) **5**

UD06. Caso práctico 01

# 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 MySQL. Este ejemplo es similar al propuesto como ejemplo durante el contenido de la unidad.

# 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 "mysql", version 5.7*  image: mysql:5.7  *#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 mysql, nombre de base de datos,*  *# usuario con permisos root (necesario para conexiones remotas) y password de ese usuario*  environment:  MYSQL\_ROOT\_PASSWORD: somewordpress  MYSQL\_DATABASE: wordpress  MYSQL\_USER: wordpress  MYSQL\_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.

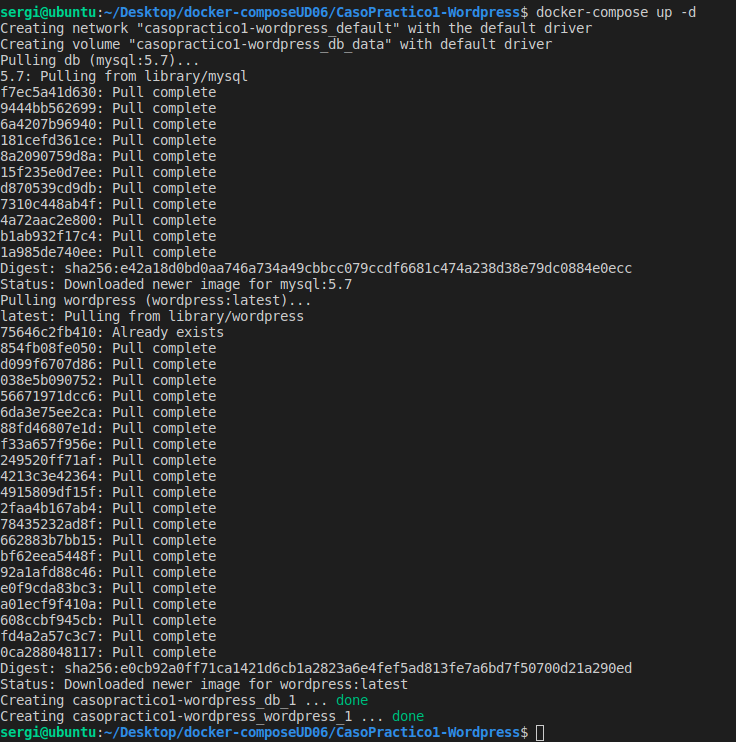
# Paso 1: Poniendo en marcha el sistema

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:

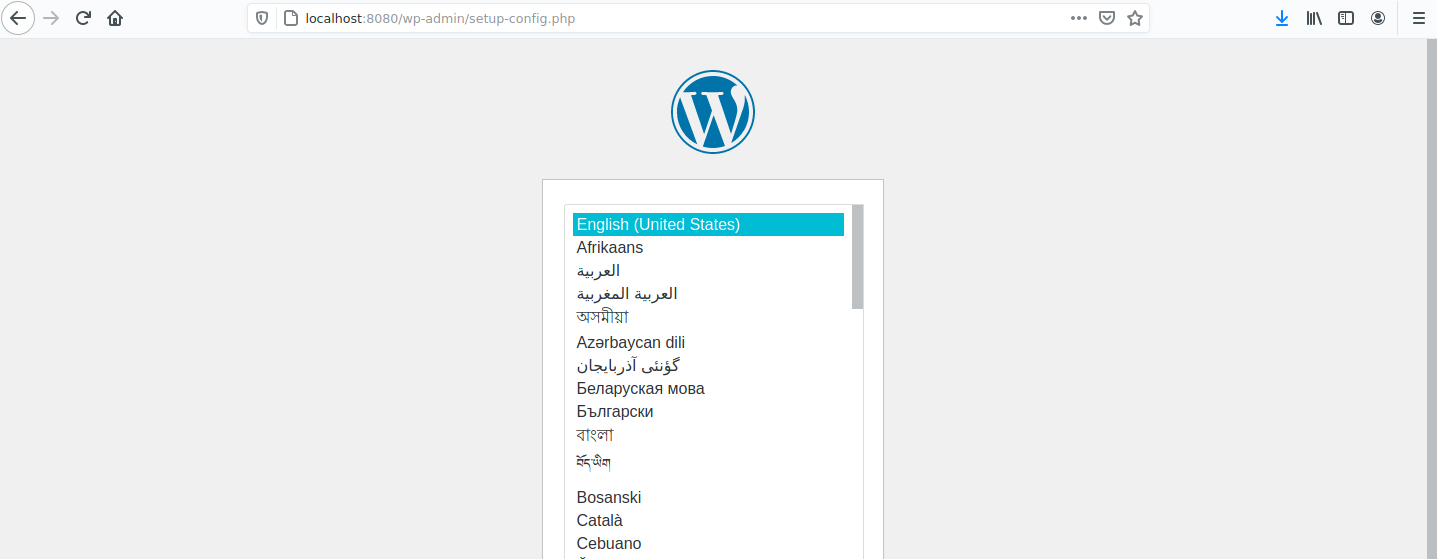
| docker-compose up -d |
| --- |

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.



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

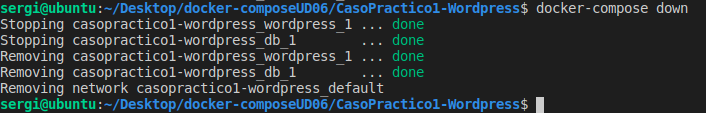


Y simplemente podremos proseguir a poner en marcha nuestro sitio Wordpress.

# Paso 2: Parando el sistema

Parar el sistema es tan sencillo como utilizar el comando

| docker-compose down |
| --- |

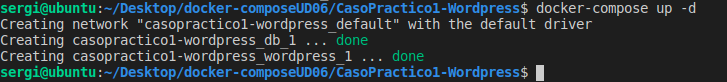


Con ello se pararán y eliminarán los contenedores. No se eliminarán ni las imágenes ni los volúmenes (el sistema Wordpress mantiene la persistencia, al tener mapeados la información de la base de datos a un volumen).

# 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.

# Bibliografía

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

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