Introducción a Docker y Kubernetes

# UD 03. Caso práctico 02 - Instalando LAMP + Wordpress en un contenedor

Autor: Sergi García Barea

Actualizado Enero 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**

# ÍNDICE

# UD03. CASO PRÁCTICO 02

## 1. INTRODUCCIÓN

En este caso práctico, vamos a utilizar como base de contenedor la imagen oficial de "***ubuntu***". En ese contenedor instalaremos Apache, PHP y MySQL (LAMP), junto al CMS Wordpress.

## 2. PREPARANDO EL CONTENEDOR

Crearemos un contenedor con la imagen base ***"ubuntu"*** *(esta vez, indicando una versión, la que tiene la etiqueta "22.04")*, al que llamaremos LAMP y en el que expondremos su puesto 80 dentro del puerto 8080 de nuestro sistema. Además, al crearlo, dejaremos lista una "shell" para instalar los programas pertinentes.

Esto podemos hacerlo con la orden:

```
docker run -d -p 8080:80 --name LAMP ubuntu:22.04 tail -f /dev/null
```

Podemos acceder al contenedor una vez creado con:

```
docker exec -it LAMP  bash
```

Si todo sale bien nos encontraremos en la "shell" del contenedor, de forma similar a:

```
alumno@alumno-virtualbox:~/Desktop$ docker run -d -p 8080:80 --name LAMP ubuntu:22.04 tail -f /dev/null

e6129c4bc8b892edac052f6b5bfaee0a6889df05f449f5843fc84d540cf47829
alumno@alumno-virtualbox:~/Desktop$ docker exec -it LAMP bash
root@e6129c4bc8b8:/#
```

## 3. INSTALANDO LAMP Y WORDPRESS

A continuación, instalaremos todo lo necesario para disponer de Wordpress (incluyendo LAMP) siguiendo las instrucciones de instalación en Ubuntu. Podéis ver como instalar LAMP en:

https://www.itzgeek.com/how-tos/linux/ubuntu-how-tos/install-lamp-stack-apache-mariadb-php-on-ubuntu-22-04.html?utm_content=cmp-true

A continuación, detallamos los pasos a seguir.

### 3.1 Actualizando repositorio e instalando LAMP + Wordpress

En primer lugar, actualizamos la lista de paquetes del repositorio:

```
apt update
```

Tras ello, instalamos los paquetes necesarios para instalar LAMP + Wordpress

```
apt install wordpress php libapache2-mod-php mariadb-server php-mysql
```

Tras ello podemos lanzar el servicio Apache con el comando:

```
service apache2 start
```

Con esto ya tenemos todo el software instalado. Podemos hacer ya una pequeña prueba conectando desde nuestra máquina http://localhost:8080

### 3.2 Preparando Apache para usar Wordpress

Antes que nada, podemos instalar un editor de texto en modo consola que sea de vuestro agrado.

Por ejemplo, con este comando podemos instalar “nano”:

```
apt install nano
```

Una vez instalado vuestro editor favorito, editamos la configuración de Apache para trabajar con Wordpress. Deberemos crear el fichero de configuración del sitio en Apache ***“/etc/apache2/sites-available/wordpress.conf”*** que configurará el acceso al sitio Wordpress. Dicho fichero deberá tener el siguiente contenido:

```
Alias /blog /usr/share/wordpress
<Directory /usr/share/wordpress>
    Options FollowSymLinks
    AllowOverride Limit Options FileInfo
    DirectoryIndex index.php
    Order allow,deny
    Allow from all
</Directory>
<Directory /usr/share/wordpress/wp-content>
    Options FollowSymLinks
    Order allow,deny
    Allow from all
</Directory>
```

Creado el fichero, deberemos cargar el sitio, habilitar “URL rewriting” y recargar Apache con los siguientes comandos:

```
a2ensite wordpress
```

```
a2enmod rewrite
```

```
service apache2 restart
```

Con esto ya podemos probar con la URL http://localhost:8080/blog, aunque veremos un error porque aún no hemos configurado el fichero ***“/etc/wordpress/config-localhost.php”.***

### 3.3 Preparando MariaDB Server

En primer lugar, deberemos poner en marcha el servicio con el comando

```
service mariadb start
```

Tras ello, deberemos ejecutar el comando para generar un password de root de MySQL Server de forma segura (deberemos recordarlo) y otras opciones. El comando a ejecutar es:

```
mysql_secure_installation
```

Más información en https://mariadb.com/kb/en/mysql_secure_installation/

Tras ello, accederemos a la base de datos con el cliente MySQL de la siguiente forma

```
mysql -u root -p
```

Tras indicar la contraseña de "root", podremos escribir comandos para MySQL. Escribiremos los siguientes comandos. En primer lugar, creamos la base de datos ***"wordpress"***:

```
CREATE DATABASE wordpress;
```

Tras ellos, creamos el usuario "wordpress" (con contraseña "`MiPass-2023`") y le damos permisos totales en la base de datos "wordpress".

```
CREATE USER 'wordpress'@'%' IDENTIFIED BY 'MiPass-2023';
```

```
GRANT ALL PRIVILEGES ON wordpress.* TO 'wordpress'@'%' WITH GRANT
OPTION;
```

Finalmente, propagamos los privilegios establecidos para que ya estén operativos en el servidor.

```
FLUSH PRIVILEGES;
```

### 3.4 Configurando Wordpress

Usando un editor de texto de consola, vamos a crear (si no existe) o editar el fichero de configuración de Wordpress, ***"/etc/wordpress/config-localhost.php"***, quedando de la forma que quede como ahora indicamos:

```
<?php
define('DB_NAME', 'wordpress');
define('DB_USER', 'wordpress');
define('DB_PASSWORD', 'MiPass-2023');
define('DB_HOST', 'localhost');
define('DB_COLLATE', 'utf8_general_ci');
define('WP_CONTENT_DIR', '/usr/share/wordpress/wp-content');
?>
```

### 3.5 Configurando nuestro sitio Wordpress desde el navegador

Si todo ha ido bien, accediendo a la URL http://localhost:8080/blog podremos configurar nuestro sitio Wordpress, indicando nombre de sitio, usuario y email.

### 3.6 Configurando los servicios para que arranquen al iniciar el contenedor

En este punto vamos a explicar cómo conseguir que nuestro contenedor Docker pueda lanzar los servicios al iniciarse. Generalmente, por el bajo coste de poner en marcha un contenedor, los sistemas Docker están pensados para ejecutar mono-servicios.

Los contenedores multiservicio son la excepción (este caso práctico ha sido presentado con un fin didáctico y no de uso real). Más adelante, en el curso, veremos cómo realizar esta misma configuración utilizando servicios alojados en distintos contenedores.

Al situarnos en un caso de contenedor multiservicio, seguimos las premisas que nos indica la documentación de Docker https://docs.docker.com/config/containers/multi-service_container/ que propone que si necesitamos lanzar varios servicios, lancemos como comando la ejecución de un script que lance los servicios que necesitemos.

Como en este caso práctico nuestro contenedor está configurado para ejecutar al iniciarse la shell ***"bash"***, podemos editar el fichero de configuración ***".bashrc"*** con el fin de que al iniciar la shell, se lancen los servicios Apache y MySQL.

Podemos hacerlo siguiendo los siguientes pasos dentro del contenedor:

- Accedemos a la carpeta de nuestro usuario, "***/root***" escribiendo "***cd***", "***cd ~***" o "***cd /root***".
- Ahí, con un editor de texto, modificamos el fichero "***.bashrc***" y al final del mismo añadimos las líneas siguientes.

```
service apache2 start
service mariadb start
```

**Extra:** Los contenedores de Docker, al contrario que un sistema Linux real, no inician por defecto lanzando un proceso que lanza a otros procesos (sustitutos del antiguo proceso "Init"), como Systemd https://es.wikipedia.org/wiki/Systemd o Upstart https://es.wikipedia.org/wiki/Upstart. Por lo cual, aunque lo configuremos, al no iniciarse junto con el contendor, no arrancara los procesos.

Si en algún contexto (aunque no es lo más habitual), quisiéramos utilizar un proceso de estas características o incluso lanzar Systemd o Upstart, se plantean las siguientes soluciones.

- Instalando los sistemas en nuestro contenedor y crear el contenedor Docker en modo privilegiado. **OJO, problemas de seguridad. No recomendado.** Más información:
  - https://www.trendmicro.com/en_us/research/19/l/why-running-a-privileged-container-in-docker-is-a-bad-idea.html
- **Una alternativa sin contenedores privilegiados**, es utilizar una característica reciente añadida a Docker que permite lanzar su propio proceso de inicio en un contenedor no privilegiado con la opción ***"--init"***, del comando ***"docker run"***, e indicar ahí el proceso de SystemD o Upstart. Si no se indica nada, usará un proceso de inicio propio de Docker llamado "***tini***". Más información:
  - https://docs.docker.com/engine/reference/run/#specify-an-init-process
  - https://developers.redhat.com/blog/2016/09/13/running-systemd-in-a-non-privileged-container/

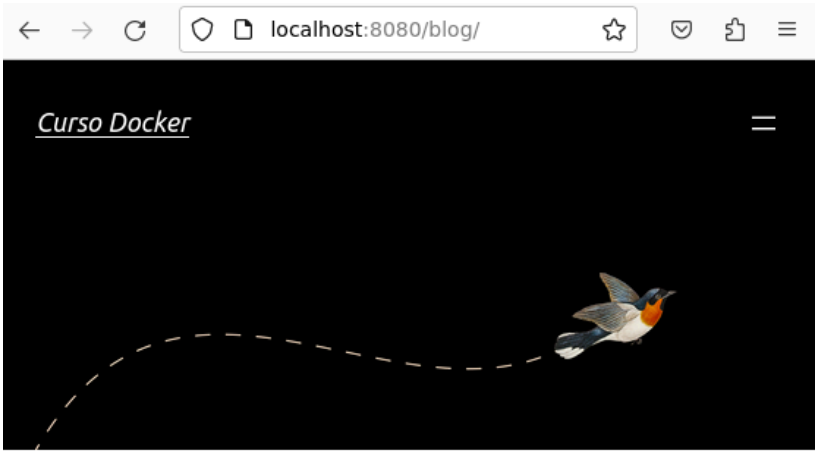
### 3.7 Comprobando que todo ha funcionado correctamente

Por último, para comprobar que todo ha funcionado correctamente, probaremos a parar el contenedor, ponerlo de nuevo en marcha y comprobar que podemos acceder a nuestro Wordpress en http://localhost:8080/blog. Para parar y lanzar el contenedor utilizaremos los comandos:

```
docker stop LAMP
```

```
docker start LAMP
```

Podremos observar algo similar a esto (puede variar la imagen):




## 4. BIBLIOGRAFÍA

[1] Install and configure LAMP on Ubuntu
https://www.itzgeek.com/how-tos/linux/ubuntu-how-tos/install-lamp-stack-apache-mariadb-php-on-ubuntu-22-04.html?utm_content=cmp-true