
Despliegue de WordPress con Docker y Docker Compose

Implantación de Aplicaciones Web

Curso 2025/2026

Índice general

1	Despliegue de WordPress usando contenedores Docker y Docker Compose	1
1.1	Tareas a realizar	1
1.2	Requisitos de la instancia EC2 en AWS	1
1.3	Requisitos del archivo docker-compose.yml	1
1.3.1	Imágenes	1
1.3.2	Networks	2
1.3.3	Docker restart policies	3
1.3.4	Variables de entorno	4
1.3.5	Orden en el que se inician los servicios	4
1.4	Entregables	4
2	Referencias	5
3	Licencia	6

1 Despliegue de WordPress usando contenedores Docker y Docker Compose

En esta práctica tendremos que realizar la implantación de un sitio [WordPress](#) en [Amazon Web Services \(AWS\)](#) haciendo uso de contenedores [Docker](#) y la herramienta [Docker Compose](#).

1.1. Tareas a realizar

A continuación se describen **muy brevemente** algunas de las tareas que tendrá que realizar.

1. Crear una máquina virtual [Amazon EC2](#).
2. Instalar y configurar [Docker](#) y [Docker compose](#) en la máquina virtual.
3. Crear un archivo **docker-compose.yml** para poder desplegar los servicios de **WordPress, MySQL, phpMyAdmin** y **HTTPS-portal**.
4. Buscar cuál es la dirección IP pública de su instancia en AWS y comprobar que puede acceder a los servicios de **WordPress** y **phpMyAdmin** desde una navegador web.
5. Una vez que tenga el archivo `docker-compose.yml` configurado de forma correcta, tendrá que añadir el servicio `https-portal` para poder acceder al servicio de **WordPress** a través de un nombre de dominio por HTTPS.

1.2. Requisitos de la instancia EC2 en AWS

Se recomienda que la instancia EC2 de AWS tenga como mínimo las siguientes características:

- **Tipo de instancia:** t2.small
- **Almacenamiento:** 20 GB

1.3. Requisitos del archivo `docker-compose.yml`

1.3.1. Imágenes

Deberá utilizar las imágenes de [Docker Hub](#) que se indican para cada uno de los servicios:

- Servicio: `wordpress`: [bitnami/wordpress](#)
- Servicio: `mysql`: [mysql](#)
- Servicio: `phpmyadmin`: [phpmyadmin/phpmyadmin](#)
- Servicio: `https-portal`: [steveltm/https-portal](#)

Imagen de WordPress de Bitnami

En [Docker Hub](#) puede encontrar más información sobre las variables de entorno que se pueden utilizar con la imagen [bitnami/wordpress](#).

Otra forma de conocer cuáles son las variables de entorno y los valores que tienen asignados por defecto es crear un contenedor con la imagen de [bitnami/wordpress](#) y examinar el contenido de las variables de entorno.

Paso 1. Crear un contenedor en modo interactivo con la imagen de [bitnami/wordpress](#).

```
1 docker run -it bitnami/wordpress /bin/sh
```

Paso 2. Una vez dentro del contenedor puede consultar el valor de las variables de entorno.

```
1 env
```

En nuestro caso nos interesan las siguientes variables de entorno.

```
1 WORDPRESS_DATABASE_HOST=mariadb
2 WORDPRESS_DATABASE_USER=bn_wordpress
3 WORDPRESS_PASSWORD=bitnami
4 WORDPRESS_DATABASE_NAME=bitnami_wordpress
5 WORDPRESS_BLOG_NAME="User's blog"
6 WORDPRESS_USERNAME=user
7 WORDPRESS_PASSWORD=bitnami
8 WORDPRESS_EMAIL=user@example.com
```

Estas son las variables de entorno que deberá utilizar en su archivo `docker-compose.yml` y configurar `.env`.

1.3.2. Networks

Los servicios definidos en el archivo `docker-compose.yml` deberán usar dos redes:

- `frontend-network`
- `backend-network`

En la red `frontend-network` estarán los servicios:

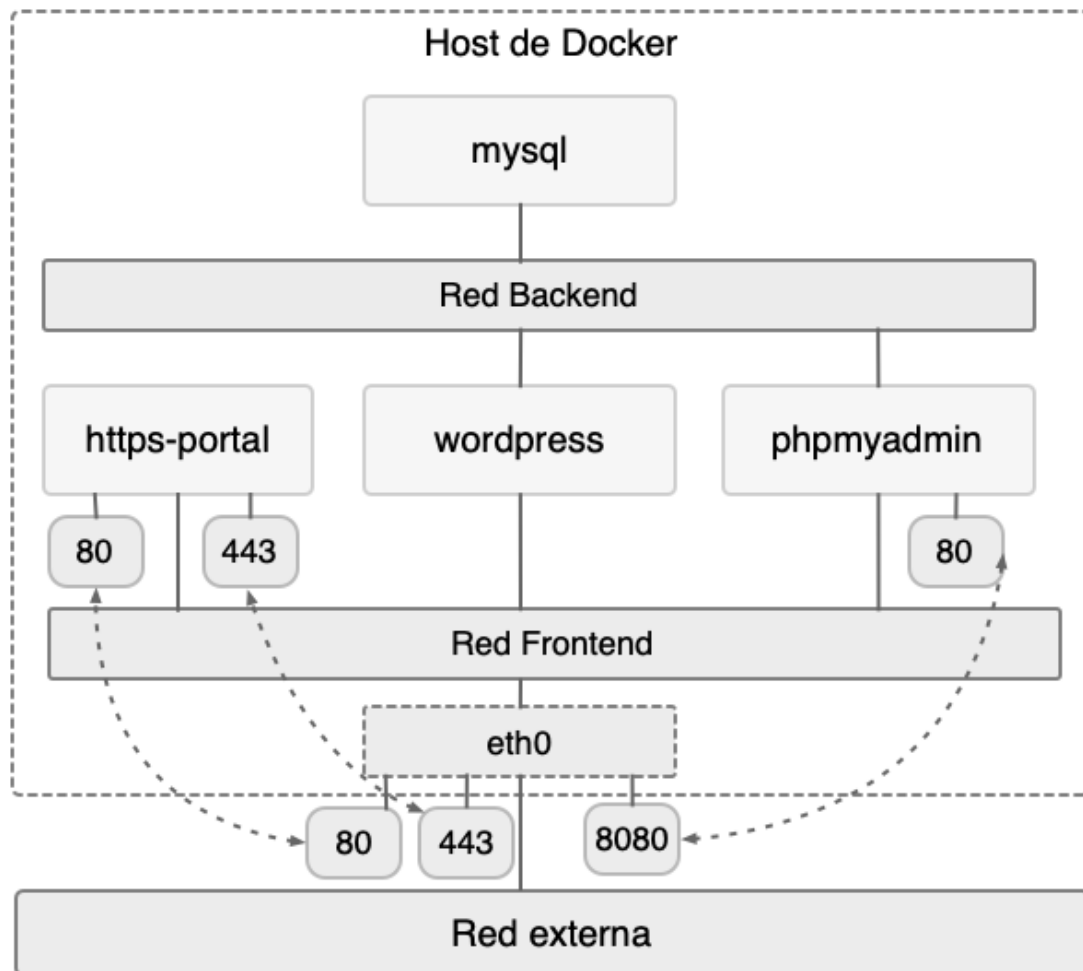
- **wordpress**
- **phpmyadmin**
- **https-portal**

Y en la red `backend-network` sólo estará el servicio:

- **mysql**

Sólo los servicios que están en la red *frontend-network* expondrán sus puertos en el host. Por lo tanto, el servicio de **mysql** no deberá estar accesible desde el host.

A continuación se muestra un diagrama con las redes y los servicios que tiene que crear:



1.3.3. Docker restart policies

Deberá utilizar alguna política de reinicio para que los contenedores se reinicien cada vez que se detengan de forma inesperada.

Se recomienda consultar la [documentación oficial de la opción restart](#).

1.3.4. Variables de entorno

Deberá hacer uso de un archivo **.env** para almacenar todas las variables de entorno que necesite en el archivo **docker-compose.yml**.

En la [documentación oficial](#) puede encontrar más información sobre cómo hacer uso de variables de entorno en el archivo **docker-compose.yml**.

1.3.5. Orden en el que se inician los servicios

Deberá indicar el orden en el que se deben iniciar los servicios con la opción `depends_on`. Se recomienda la lectura del artículo [Control startup and shutdown order in Compose](#)

Para garantizar que el servicio de **MySQL** está listo para aceptar conexiones, deberá utilizar la opción `healthcheck` del archivo **docker-compose.yml**. Se recomienda la lectura del artículo [Healthcheck. Compose file version 3 reference](#).

1.4. Entregables

En esta práctica habrá que entregar un **documento técnico** con la descripción de los pasos que se han llevado a cabo durante todo el proceso.

El documento debe incluir **como mínimo** lo siguientes contenidos:

- URL del repositorio de GitHub donde se ha alojado el documento técnico escrito en [Markdown](#).
- Descripción de la configuración del archivo `docker-compose.yml` que se ha utilizado en esta práctica.
- Descripción de las acciones que ha realizado durante durante la puesta en producción
- URL del sitio web con HTTPS habilitado.

2 Referencias

- [Quickstart: Compose and WordPress.](#)
- [Curso de introducción a Docker.](#)
- [Docker](#)
- [Docker Compose](#)
- [Control startup and shutdown order in Compose](#)
- [Healthcheck. Compose file version 3 reference](#)

3 Licencia

Esta página forma parte del curso Implantación de Aplicaciones Web de José Juan Sánchez Hernández y su contenido se distribuye bajo una licencia Creative Commons Reconocimiento-NoComercial-CompartirIgual 4.0 Internacional.