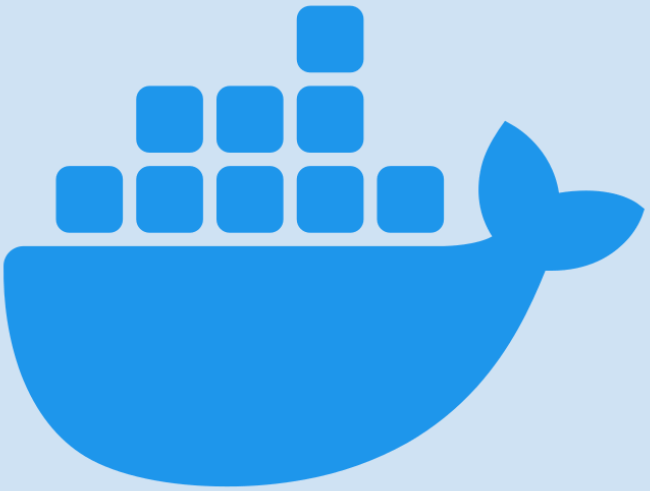
Despliegue de Wordpress con Docker



ÍNDICE

Objetivo:

Desplegar la aplicación web de WordPress utilizando Docker, comprendiendo el proceso de contenerización y el manejo de bases de datos en contenedores separados.
Esta tarea incluirá la creación de una red personalizada en Docker para facilitar la comunicación entre el contenedor de WordPress y el contenedor de la base de datos MariaDB.

Descripción:

WordPress es un sistema de gestión de contenidos (CMS) que permite crear y mantener fácilmente un blog o sitio web.
Para esta tarea, desplegarán WordPress en un contenedor Docker y lo conectas con un contenedor MariaDB en una red Docker personalizada.
Esto te proporcionará una comprensión práctica de cómo los contenedores pueden ser utilizados para desplegar aplicaciones web y cómo gestionar la persistencia de datos en entornos contenerizados.

Requisitos previos:

- Tener Docker instalado en tu sistema (Windows, Linux o macOS).
- Conocimientos básicos de línea de comandos en tu sistema operativo.
- Conocimientos básicos de Docker y conceptos de contenerización.

Instrucciones:

1. **Crear una Red Docker Personalizada.**

2. **Desplegar el Contenedor de la Base de Datos MariaDB.**
   Inicia un contenedor MariaDB configurado para tu entorno WordPress.

3. **Desplegar WordPress.**
   Configura y lanza un contenedor de WordPress que se conecte a tu base de datos MariaDB.

4. **Acceso a WordPress.**
   Completa la instalación de WordPress a través de su interfaz web accediendo a una URL específica.

5. **Documenta todo el proceso de despliegue.**

Explica las decisiones tomadas durante el despliegue, como la configuración de la red Docker personalizada. Reflexiona sobre la importancia de separar la aplicación web y la base de datos en contenedores diferentes.

1. Crear una Red Docker personalizada.

Para empezar, crear una red personalizada en Docker permite que los contenedores (WordPress y MariaDB) se comuniquen entre sí usando el nombre del host. Esto permite aislar la comunicación de otros contenedores del sistema y evita conflictos de nombres o puertos.

Para crear la red de Docker abriremos la terminal o **cmd** y ejecutaremos el siguiente comando: **<docker network create Wordpress-trabajo-despliegue>**. Con eso crearemos nuestra red y para ver si se ha creado correctamente usamos el comando **<docker network ls>**. Intenté ver si podían verse las redes creadas directamente en Docker desktop pero parece que mi versión no lo tiene implementado, sería de utilidad la verdad y te ahorras el tener que usar la consola.

```
C:\Users\vrida>docker network ls
NETWORK ID     NAME                              DRIVER    SCOPE
946faffc1cad   Wordpress-trabajo-despliegue      bridge    local
68aed30db750   bridge                            bridge    local
e702edd42fb5   host                              host      local
732e4010cd8d   none                              null      local
8ca3698d77a6   redis                             bridge    local
```

Por último, si quieres ver los detalles puedes usar el comando **<docker network inspect Wordpress-trabajo-despliegue>**.

```
C:\Users\vrida>docker network inspect Wordpress-trabajo-despliegue
[
    {
        "Name": "Wordpress-trabajo-despliegue",
        "Id": "946faffc1cad715098094e8cdaa598fb70a2bc25b1bfe1fbc49ec0a8fab84336",
        "Created": "2025-10-24T15:32:16.734858876Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv4": true,
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "172.19.0.0/16",
                    "Gateway": "172.19.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
```

2. Desplegar el Contenedor de la Base de Datos en MariaDB

Ahora crearemos el contenedor de la Base de Datos con MariaDB para Wordpress y que esté conectado a nuestra red.

Para ello primero ejecutaremos el siguiente comando en la terminal:

<
docker run -d --name mariadb --network Wordpress-trabajo-despliegue -e MYSQL_ROOT_PASSWORD=rootpassword -e MYSQL_DATABASE=wordpress -e MYSQL_USER=wp_user_victor -e MYSQL_PASSWORD=wp_password mariadb:latest
>

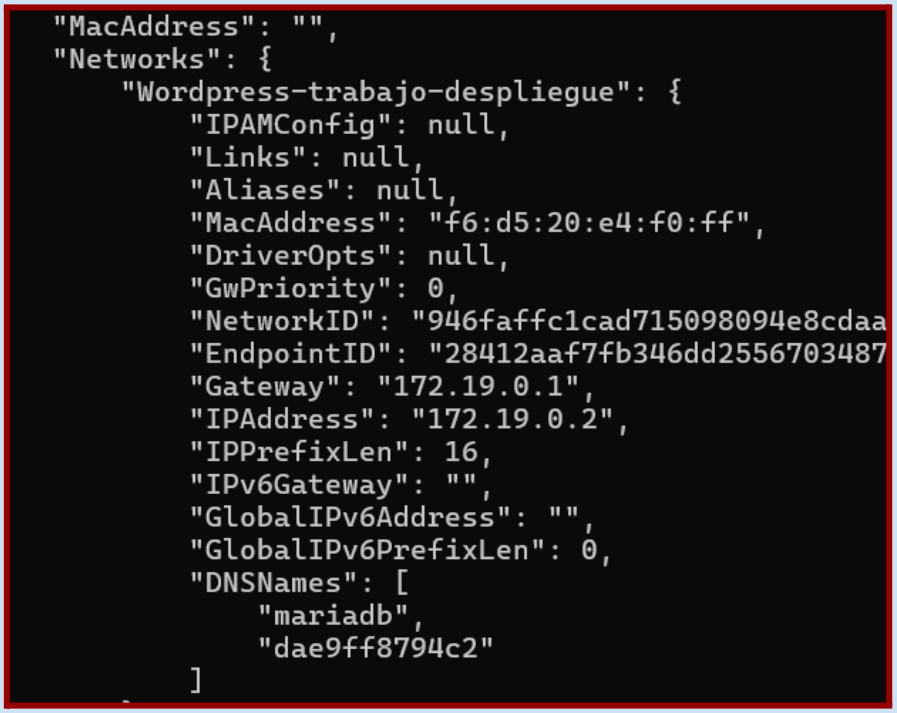
Con este comando se crea una base de datos llamada **“wordpress”**, un usuario llamado **“wp_user_victor”** y se conecta a la red usando el nombre del contenedor **“mariadb”**.

```
C:\Users\vrida>docker run -d --name mariadb --network Wordpress-trabajo-despliegue -e MYSQL_ROOT_PASSWORD=rootpassword -
e MYSQL_DATABASE=wordpress -e MYSQL_USER=wp_user_victor -e MYSQL_PASSWORD=wp_password mariadb:latest
Unable to find image 'mariadb:latest' locally
latest: Pulling from library/mariadb
6499e85d8558: Pull complete
78540072237e1: Pull complete
402f4c7dd065: Pull complete
4b3ffd8ccb52: Pull complete
8ed1b5271813: Pull complete
eece26ecda4b: Pull complete
2295facceabc: Pull complete
4c2e197f6cb0: Pull complete
Digest: sha256:5b6a1eac15b85b981a61afb89aea2a22bf76b5f58809d05f0bcc13ab6ec44cb8
Status: Downloaded newer image for mariadb:latest
dae9ff8794c2f99f0892e67dfe5944a1165ed46f16429100cf45bec4a7366cec

C:\Users\vrida>
```

Una vez completada la descarga de archivos (ha tardado un rato), ya estaría creado el contenedor. Para ver si está corriendo usamos el comando **<docker ps>**.

```
C:\Users\vrida>docker ps
CONTAINER ID   IMAGE           COMMAND                  CREATED              STATUS              PORTS      NAMES
dae9ff8794c2   mariadb:latest   "docker-entrypoint.s…"   About a minute ago   Up About a minute   3306/tcp   mariadb

C:\Users\vrida>
```

En el apartado **STATUS** si aparece **Up** significa que está funcionando correctamente. Ahora, para ver que está corriendo en la red creada anteriormente usaremos el comando **<docker inspect mariadb>**. Si buscamos la sección **"Networks"** nos debería aparecer.



Esto nos deja claro que WordPress podrá conectarse usando el nombre del contenedor **mariadb**.

3. Desplegar WordPress

Ahora debemos configurar y lanzar un contenedor de WordPress que se conecte a la base de datos de mariadb que hemos creado y que use la red personalizada para que se comuniquen entre sí.

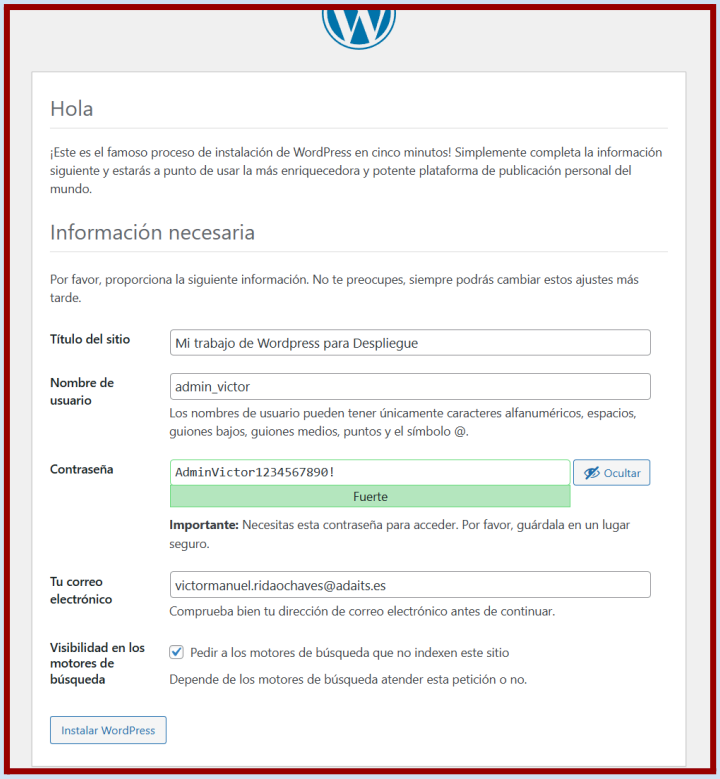
Para ello, ejecutaremos en nuestro CMD el siguiente comando:

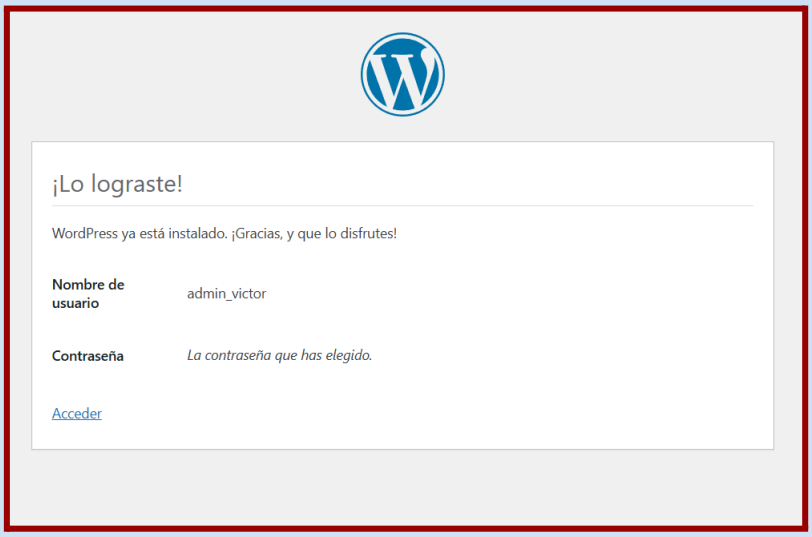
<
docker run -d --name wordpress-app --network Wordpress-trabajo-despliegue -p 8080:80 -e WORDPRESS_DB_HOST=mariadb:3306 -e WORDPRESS_DB_USER=wp_user_victor -e WORDPRESS_DB_PASSWORD=wp_password -e WORDPRESS_DB_NAME=wordpress wordpress:latest
>

Luego comprobaremos que el contenedor funciona con el comando **<docker ps>**.

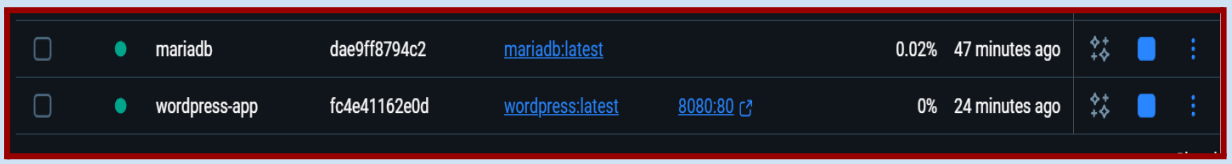
```
C:\Users\vrida>docker ps
CONTAINER ID   IMAGE              COMMAND                  CREATED          STATUS          PORTS
                NAMES
fc4e41162e0d   wordpress:latest   "docker-entrypoint.s…"   24 seconds ago   Up 22 seconds   0.0.0.0:8080->80/tcp, [::]:8
080->80/tcp     wordpress-app
dae9ff8794c2   mariadb:latest     "docker-entrypoint.s…"   23 minutes ago   Up 23 minutes   3306/tcp
                mariadb
```

Ahora vamos al navegador y copiamos esta dirección: **http://localhost:8080**
En la pantalla de WordPress seleccionamos primero el idioma y luego rellenamos los campos con nuestros datos. Una vez hecho, ya estará conectado a mariadb.





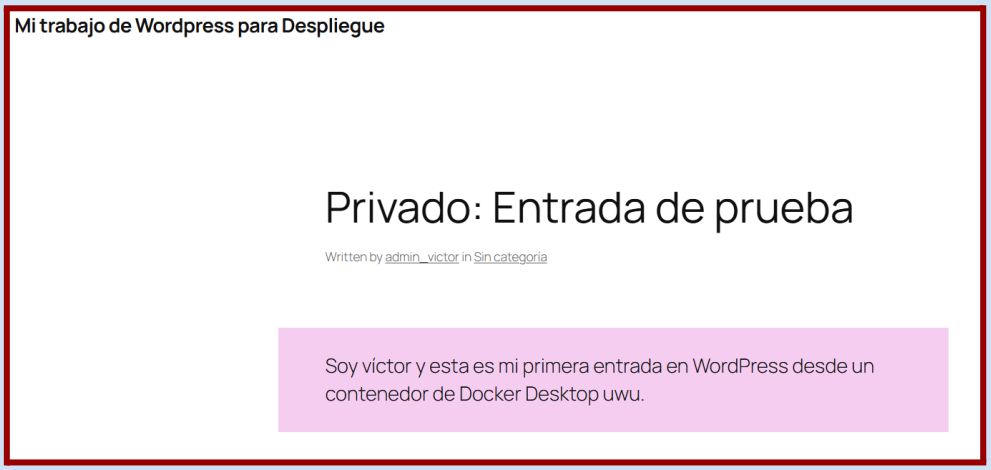
Por último, si nos vamos a Docker desktop, en los contenedores veremos que están funcionando.



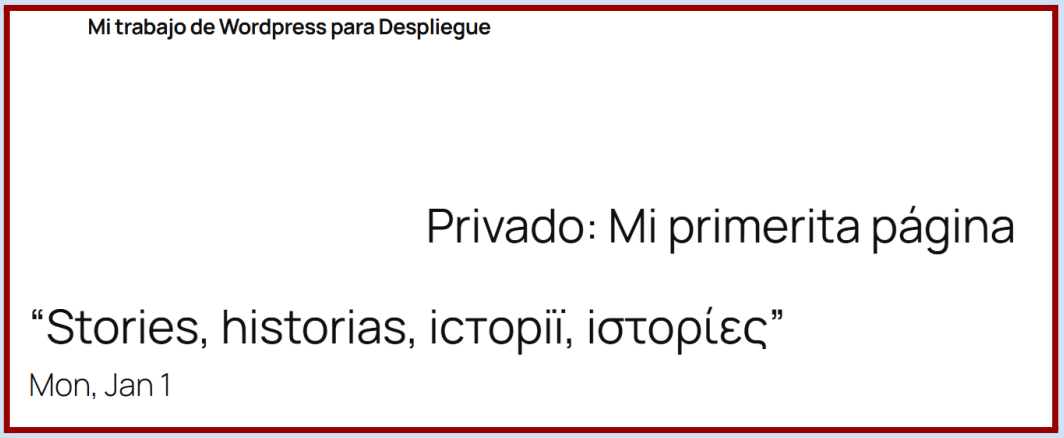
4. Acceso a WordPress

Una vez en la página de inicio de WordPress realizaremos algunas pruebas para comprobar que funciona correctamente en el container.

Primero crearemos una entrada de prueba, para eso en Entradas añadiremos una nueva, escribimos un título y un contenido, la publicamos y verificamos que aparece en el sitio.



Ahora crearemos una página de prueba, para ello, desde el menú de inicio iremos a Páginas, añadiremos una nueva y de nuevo añadimos contenido y publicamos.



5. Reflexión final

El hecho de separar tanto la aplicación como la base de datos en contenedores diferentes es conveniente ya que si alguno de los dos llega a fallar por el motivo que sea el otro sigue funcionando lo cual acaba ahorrando bastante tiempo.
También por seguridad ya que dentro de lo malo, si acceden es mejor que sea solo a una parte del sistema que a todo el entorno.
Cambiar la base de datos es más sencillo ya que no hay que tocar la aplicación.
Y por último que, al ser contenedores separados, si se quieren mover o desplegar por separado se podrá hacer sin que uno afecte al otro.