Enlazar contenedores (Wordpress y MariaDB)

En esta parte de la práctica crearemos un contenedor de una imagen de Wordpress (Gestor de contenido) y una imagen de MariaDB (Base de datos) y los enlazaremos para tener nuestro Wordpress bien configurado con una base de datos. Para enlazarlos los uniremos a la misma red.

Contenido de ayuda:

- Ver documentación en la página Web de Docker Hub para ver los parámetros necesarios para crear contenedor de base de datos.

1. Creamos una red o usamos una ya creada de las actividades anteriores.
2. Descargamos las imágenes de wordpress y mariadb.

$docker pull wordpress

```
[root@localhost ~]# docker pull wordpress
Using default tag: latest
latest: Pulling from library/wordpress
b0a0cf830b12: Pull complete
c93478d47932: Pull complete
e74cc574d0d2: Pull complete
e4782e138a90: Pull complete
cfeec87621ae: Pull complete
c1badcd002c0: Pull complete
e0d463a60cb6: Pull complete
d1ad50b335f5: Pull complete
98c64971444a: Pull complete
ba7d8962b7e1: Pull complete
e620d8fafd56: Pull complete
2038239aa3e1: Pull complete
df99eaff6870: Pull complete
775ef72901a7: Pull complete
c7acd866c76c: Pull complete
666e528f269a: Pull complete
0312bea7a451: Pull complete
8477b729f124: Pull complete
46e06c94867e: Pull complete
6baa1fda38e7: Pull complete
59fe2989bce1: Pull complete
Digest: sha256:022c7d4c14b8af7ddd7c1a7c3b8ea633d2db42219913eb1e79ef89484ae26375
Status: Downloaded newer image for wordpress:latest
docker.io/library/wordpress:latest
[root@localhost ~]#
```

$docker pull mariadb

```
[root@localhost ~]# docker pull mariadb
Using default tag: latest
latest: Pulling from library/mariadb
e311a697a403: Pull complete
bf1ba319970c: Pull complete
cc8c04c19869: Pull complete
83faec168fdb: Pull complete
6f7b4e5babb3: Pull complete
21f06199db3e: Pull complete
0ba9e4e742cf: Pull complete
424da8c2bb4f: Pull complete
Digest: sha256:416016f57f522c215d2908866e56e13e6474f071cd65112c216ba19e68f219ca
Status: Downloaded newer image for mariadb:latest
docker.io/library/mariadb:latest
[root@localhost ~]# _
```

3. Empezamos creando un contenedor de base de datos (MariaDB). Crearemos nuestra base de datos llamada (wordpress), el usuario de la base de datos (usuario_wordpress), contraseña para el usuario (password1). También le tenemos que dar una contraseña al usuario root.

$docker run -d --name Nombre_Contenedor --network Nombre_Red -e MARIADB_DATABASE=wordpress -e MARIADB_USER=Usuario_Wordpress -e MARIADB_PASSWORD=Contraseña -e MARIADB_ROOT_PASSWORD=Contraseña_Root mariadb

```
[root@localhost ~]# docker run -d --name postgres --network red2 -e MARIADB_DATABASE=wordpress -e MA
RIADB_USER=Usuario_Wordpress -e MARIADB_PASSWORD=Contraseña -e MARIADB_ROOT_PASSWORD=Contraseña_Root
 mariadb
6c031dae8f0142b46aba7ec5b33c3a8419727f4d4c70cf1fed35bca5c46c5ec0
[root@localhost ~]#
```

4. Con el contenedor de base de datos ya creado procedemos acceder al mismo para comprobar que se ha creado todo correctamente.

$docker exec -it mariadb1 bash

```
[root@localhost ~]# docker exec -it postgres bash
;root@6c031dae8f01: /root@6c031dae8f01:/# _
```

5. Nos conectamos a la base de datos de MariaDB para ello se utiliza el siguiente comando dentro del contenedor:

$mariadb -u Usuario_Wordpress -p

```
;root@6c031dae8f01: /root@6c031dae8f01:/# mariadb -u Usuario_Wordpress -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 3
Server version: 11.3.2-MariaDB-1:11.3.2+maria~ubu2204 mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> _
```

```
[iesroqueamagro@Chromecast ~]$ docker exec -it mariadb1 bash
root@7bc8d7113a57:/# mariadb -u usuario_wordpress -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 5
Server version: 10.11.2-MariaDB-1:10.11.2+maria~ubu2204 mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

Usamos el siguiente comando para ver que efectivamente se ha creado nuestra base de datos para Wordpress:

```
$show databases;
```

```
MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| wordpress          |
+--------------------+
2 rows in set (0.000 sec)

MariaDB [(none)]> _
```

Con lo anterior comprobamos que el usuario accede correctamente a la base de datos y que la base de Wordpress se ha creado correctamente.

Para salir de la base de datos usamos el siguiente comando:

```
$quit
```

```
MariaDB [(none)]> quit
Bye
;root@6c031dae8f01: /root@6c031dae8f01:/# _
```

Llegados a este punto ya tenemos la base de datos preparada para tener almacenada toda la información que necesitará Wordpress para funcionar. Pasamos ahora a crear nuestro contenedor de wordpress.

6. Creamos el contenedor de wordpress, asociado a la misma red y elegimos el puerto

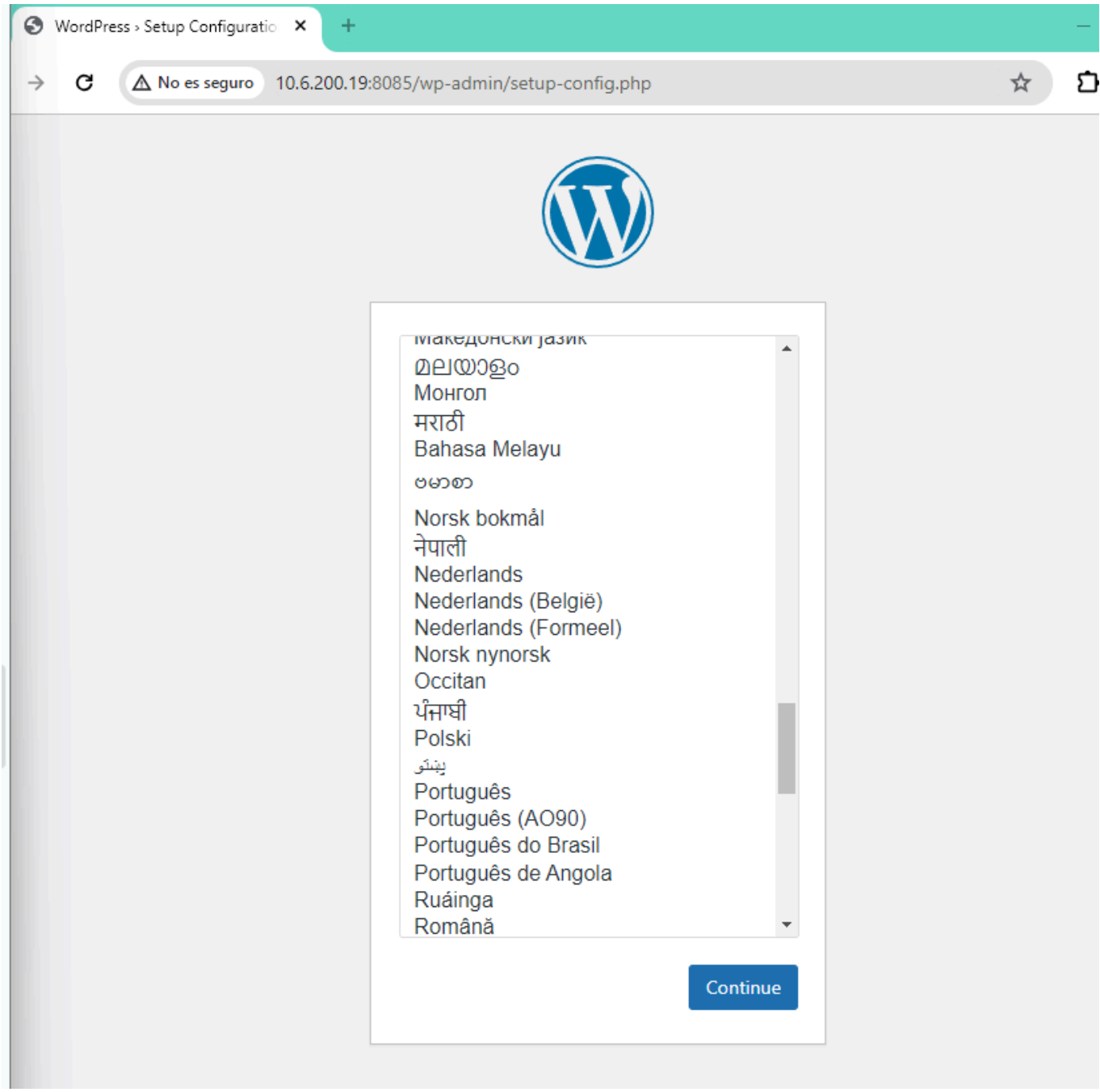
8080 de la máquina host para escuchar. Se podría pasar todos los datos necesarios por variables para que wordpress se conecte directamente a la base de datos, pero en este caso optamos por hacerlo de manera gráfica al conectarnos por el puerto 8080.

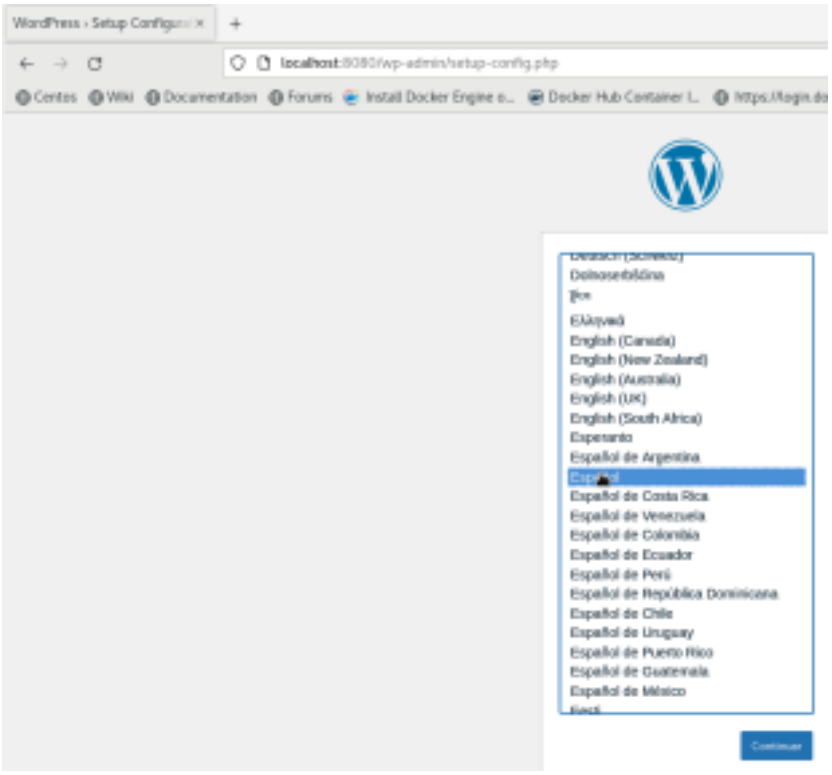
$ docker run --name Nombre_Wordpress --network Nombre_Red -d -p 8080:80 wordpress

```
[root@localhost ~]# docker run --name wordpress --network red2 -d -p 8085:80 wordpress
c37b71124a125000dc2e836beb3c91aaada6ed95c637d8074db1c1872b9dae5b
[root@localhost ~]#
```

7. Comprobar e inspeccionar que los contenedores están en marcha y en la misma red.

8. Nos conectamos a un navegador por el puerto 8080 y nos debe aparecer el proceso de instalación de Wordpress:


WordPress › Setup Configuratio
No es seguro
10.6.200.19:8085/wp-admin/setup-config.php
Македонски јазик
മലയാളം
Монгол
मराठी
Bahasa Melayu
ဗမာစာ
Norsk bokmål
नेपाली
Nederlands
Nederlands (België)
Nederlands (Formeel)
Norsk nynorsk
Occitan
ਪੰਜਾਬੀ
Polski
پښتو
Português
Português (AO90)
Português do Brasil
Português de Angola
Ruáinga
Română
Continue


WordPress › Setup Configur
localhost:8080/wp-admin/setup-config.php
Centos
Wiki
Documentation
Forums
Install Docker Engine o...
Docker Hub Container I...
Ελληνικά
English (Canada)
English (New Zealand)
English (Australia)
English (UK)
English (South Africa)
Esperanto
Español de Argentina
Español de Costa Rica
Español de Venezuela
Español de Colombia
Español de Ecuador
Español de Perú
Español de República Dominicana
Español de Chile
Español de Uruguay
Español de Puerto Rico
Español de Guatemala
Español de México


A continuación tendrás que introducir los detalles de tu conexión con la base de datos. Si no estás seguro de ellos, contacta con tu proveedor de alojamiento.
Nombre de la base de datos
wordpress
El nombre de la base de datos que quieres usar con WordPress.
Nombre de usuario
usuario_wordpress
El nombre de usuario de tu base de datos.
Contraseña
password1
La contraseña de tu base de datos.
Servidor de la base de datos
mariadb1
Si localhost no funciona, deberías poder obtener esta información de tu proveedor de alojamiento web.
Prefijo de tabla
wp_
Si quieres ejecutar varias instalaciones de WordPress en una sola base de datos cambia esto.
Enviar

Ya tenemos nuestros dos contenedores enlazados y funcionando entre sí.

Volumenes en Docker

En Docker, un volumen es un mecanismo que permite a los contenedores de Docker almacenar y acceder a datos de forma persistente. Los volúmenes en Docker son utilizados para compartir y persistir datos entre contenedores o entre el sistema operativo anfitrión y los contenedores.

Estos volúmenes son directorios montados en el sistema de archivos del host o en otro contenedor, y están diseñados para ser independientes del ciclo de vida de los contenedores. Esto significa que los datos almacenados en un volumen persistirán incluso si se elimina el contenedor que lo creó. Los volúmenes también son utilizados para compartir datos entre contenedores, lo que permite la comunicación y colaboración entre distintos contenedores en un entorno de Docker.

Se utilizan para manejar datos que deben ser persistentes, como bases de datos, archivos de configuración, registros, y otros datos que necesitan sobrevivir al ciclo de vida de los contenedores individuales. Los volúmenes son una herramienta poderosa en Docker para separar la lógica de la aplicación de los datos, permitiendo una mayor flexibilidad y portabilidad en el despliegue de aplicaciones en contenedores Docker.

Comando de ayuda:

```
$ docker volume --help
```

Antes de crear nuestros propios volúmenes pasaremos a usar una imagen que al crear un contenedor de la misma crea un volumen por defecto. La imagen utilizada es Owncloud,

servicio que nos permite disponer de nuestro almacenamiento privado en la nube. Este servicio escucha por el puerto 8080.
Recuerda inspeccionar las imágenes para conocer información importante de dicha imagen.

1. Descargamos Owncloud/Server y comprobamos que la imagen se ha descargado correctamente.

```
[root@localhost ~]# docker pull owncloud/server
Using default tag: latest
latest: Pulling from owncloud/server
4477f8fe99eb: Pull complete
eb038e86b6fa: Pull complete
d037ab521f57: Pull complete
dad61813398c: Pull complete
6237f2d95335: Pull complete
bcc0150694fb: Pull complete
4f4fb700ef54: Pull complete
4d6c811c3b79: Pull complete
f45d1f6112b1: Pull complete
71ee7589ecc3: Pull complete
d6013244ffc2: Pull complete
d0a869b6a52e: Pull complete
9c6da59d048f: Pull complete
Digest: sha256:80b74ea8bf968912a5816f97f33b7cb58e3bb76cfafdea4b0c15b4f78e1742b5
Status: Downloaded newer image for owncloud/server:latest
docker.io/owncloud/server:latest
[root@localhost ~]#
```

2. Vamos ahora a comprobar los volúmenes que tenemos en nuestro Host con el siguiente comando:

```
[root@localhost ~]# docker volume ls
DRIVER    VOLUME NAME
local     2d6f277b049f14f6add06a5a4cbcaea0708f46de4e7a458316b59e690873f688
local     70e37fbee969c45d4c41824d9d57474d3f852157058aed1e78d80a5820585659
local     64309eb155883a15f0fb250c2cf6fff9ea7705560853774a1ef2e93f800716ac
local     a51db9fdec2dd39a72d93626ced4e330d4ad745aa4a7c38d9adb4ffcb64f7505
local     aee3ef861564f4803f8ccced1f656346884b1711c9abc7b28b7640ceb9764f95
local     de3b2575064e03e51a1e901158377e59be72477c9efa6a531e1c9d6d4ebb0d3d
[root@localhost ~]# _
```

$ docker volume ls

   a. Para eliminar los volúmenes que no se están utilizando por ningún contenedor se usa el siguiente comando:

$ docker volume prune

```
[root@localhost ~]# docker volume prune
WARNING! This will remove anonymous local volumes not used by at least one container.
Are you sure you want to continue? [y/N] y
Deleted Volumes:
aee3ef861564f4803f8ccced1f656346884b1711c9abc7b28b7640ceb9764f95
a51db9fdec2dd39a72d93626ced4e330d4ad745aa4a7c38d9adb4ffcb64f7505

Total reclaimed space: 0B
[root@localhost ~]#
```

b. Para eliminarlos manualmente, se puede usar el siguiente comando:

$ docker volume rm ID_VOLUME

3. Listamos los volúmenes que tenemos actualmente.

```
[root@localhost ~]# docker volume ls
DRIVER    VOLUME NAME
local     2d6f277b049f14f6add06a5a4cbcaea0708f46de4e7a458316b59e690873f688
local     70e37fbee969c45d4c41824d9d57474d3f852157058aed1e78d80a5820585659
local     64309eb155883a15f0fb250c2cf6fff9ea7705560853774a1ef2e93f800716ac
local     de3b2575064e03e51a1e901158377e59be72477c9efa6a531e1c9d6d4ebb0d3d
[root@localhost ~]# _
```

4. Creamos un contenedor de owncloud (puerto por el que escucha el servicio, 8080).

```
[root@localhost ~]# docker run --name owncloud --network red2 -d -p 8080 owncloud
d865385ce24337f62adcb3234501266fc8962013343dabd9e5ce101cb2ba89fd
```

5. Listamos los volúmenes y debería aparecer un volumen nuevo.

a. Los volúmenes Docker por defecto se almacenan en /var/lib/docker/volumes. Accedemos a ese directorio con usuario administrador y vemos que tiene que aparecer una carpeta con el mismo nombre que el volumen creado por defecto por el contenedor de owncloud creado.

```
[root@localhost ~]# sudo ls /var/lib/docker/volumes
0b33ec0d59e94e376780e4461381cc68981d8683351287e8d7cb3fa68252f978
2d6f277b049f14f6add06a5a4cbcaea0708f46de4e7a458316b59e690873f688
64309eb155883a15f0fb250c2cf6fff9ea7705560853774a1ef2e93f800716ac
70e37fbee969c45d4c41824d9d57474d3f852157058aed1e78d80a5820585659
a8818e10682d942b3cc38b1457c7af2b93551cbe6fbe2e5a4ea3345c47fcf149
backingFsBlockDev
de3b2575064e03e51a1e901158377e59be72477c9efa6a531e1c9d6d4ebb0d3d
metadata.db
[root@localhost ~]#
```

6. Ahora accedemos a nuestro contenedor desde un navegador y nos logueamos con el usuario admin con contraseña admin y creamos un fichero.

    a. Si ahora accedemos a la siguiente ruta en nuestro equipo Host podemos ver el archivo creado.

```
$ cd /var/lib/docker/volumes/Nombre_Volumen_Owncloud/_data/files/admin/files
```

Vemos que los archivos se localizan en el servidor Host y son accedidos desde el contenedor creado.

Podemos cerrar el contenedor creado para esta actividad.

En los siguientes pasos crearemos nuestros propios volúmenes.

Con la opción "-v Directorio" al crear un contenedor podemos crear un volumen y asignarlo al contenedor.

7. Creamos un contenedor de Ubuntu y le asignamos un volumen:

```
$ docker run --name ubuntu1 -d -it -v /datos ubuntu
```

8. Accedemos al contenedor y creamos un fichero en "/datos":
9. Ahora si accedemos al volumen que se nos creó al iniciar el contenedor podemos ver que está el fichero recién creado dentro del contenedor:

```
$ cd /var/lib/docker/volumes/Nombre_Volumen /_data/
```

10. Si ahora creamos un fichero en la dirección del punto anterior podemos ver dicho fichero en el contenedor: