

## Contenedores Docker. Conectando aplicaciones

1. Vamos a emplear el parametro –link de docker para enlazar distintos contenedores. En este caso enlazaremos phpmyadmin con mysql. Para ello:
  - a) Descargamos con docker pull las imágenes de phpmyadmin y de mysql
  - b) Creamos el contenedor de mysql con:

```
docker run --name bbdd -e MYSQL_ROOT_PASSWORD=naranco -d mysql
```

Le estamos asignando el nombre “bbdd” al nuevo contenedor y estamos estableciendo un valor de entorno de para la contraseña de root (“naranco” en este caso)

The screenshot shows the Docker desktop interface. At the top, there's a list of containers: 'bbdd' (running, image e6cc0ca4dd14, mysql), 'tomcat\_nginx' (idle, image 77a468d, -), and 'act7-ejer4' (idle, image 97b5c45, -). Below this is a 'Terminal' window with the following content:

```
4dc3354: Pull complete
77a468d: Pull complete
7925374: Pull complete
97b5c45: Download complete
55c3a16: Download complete
t: sha256:c9f0c66a87356518d4b30dbc065eb4567e4a04aff4d0ff194dea0973e5985b57
s: Downloaded newer image for mysql:latest
r.io/library/mysql:latest
Users\fabri> docker run --name bbdd -e MYSQL_ROOT_PASSWORD=naranco -d mysql
e6cc0ca4dd14cfb6e6d17ccfc1d161d2f4954ec2a4c453e71def2b72401355ee
[User\fabri>
```

- c) Creamos ahora el contenedor para phpmyadmin con:

```
docker run --name pma -d --link bbdd:db -p 8080:80 phpmyadmin
```

Le estamos poniendo el nombre “pma” al nuevo contenedor, mapeando el puerto 80 del contenedor con el puerto 8080 de la maquina anfitrión y con la opción –link estamos conectándolo con el contenedor “bbdd” y con el valor de alias “db”

The screenshot shows the Docker desktop interface. The list of containers now includes 'pma' (running, image ea299efda8cb, phpmyadm 8080:80). The 'Terminal' window shows the following command and its execution:

```
9fe447925374: Pull complete
99d7f97b5c45: Download complete
56c6165c3a16: Download complete
Digest: sha256:c9f0c66a87356518d4b30dbc065eb4567e4a04aff4d0ff194dea0973e5985b57
Status: Downloaded newer image for mysql:latest
r.io/library/mysql:latest
PS C:\Users\fabri> docker run --name pma -d --link bbdd:db -p 8080:80 phpmyadmin
e299efda8cb1d70dcdd019a63ccc4af7137ea8cc46c58cdc3519fc0d7c353e
PS C:\Users\fabri>
```

- d) Comprobamos con docker ps que los dos contenedores estan funcionando y finalmente accedemos a la interfaz de phpmyadmin desde un navegador cualquiera con <http://localhost:8080>. Nos loguearemos con el usuario root y la password asignada (naranco)

phpMyAdmin  
Bienvenido a phpMyAdmin

Idioma (Language)  
Español - Spanish

Iniciar sesión  
Usuario: root  
Contraseña: .....  
Iniciar sesión

¿Deseas que el administrador de contraseñas guarde la contraseña de "localhost:8080"?  
Guardar Nunca

phpMyAdmin  
Reciente Favoritas

Nueva information\_schema mysql performance\_schema sys

Configuraciones generales  
Cambio de contraseña  
Cotejamiento de la conexión al servidor: utf8mb4\_unicode\_ci  
Más configuraciones

Configuraciones de apariencia  
Idioma (Language) Español - Spanish  
Tema pmahomme Ver todo

Servidor de base de datos  
Servidor: db via TCP/IP  
Tipo de servidor: MySQL  
Conexión del servidor: No se está utilizando SSL  
Versión del servidor: 9.5.0 - MySQL Community Server - GPL  
Versión del protocolo: 10  
Usuario: root@172.17.0.3  
Conjunto de caracteres del servidor: UTF-8 Unicode (utf8mb4)

Servidor web  
Apache/2.4.66 (Debian)  
Versión del cliente de base de datos: libmysql - mysqld 8.3.29  
extensión PHP: mysqli curl mbstring sodium  
Versión de PHP: 8.3.29

- Vamos a descargar las imágenes de mariadb y wordpress y lanzar los contenedores, pero en este caso vamos a crear previamente una red particular (dentro de la red brigde) para conectar las dos aplicaciones. Para ello:

- Creamos la red privada con:

```
docker network create red_wp
```

Comprobaremos con docker inspect red\_wp la nueva dirección de red para los contenedores de esta red

- Creamos ahora el contenedor de mariadb con:

```
docker run -d --name servidor_mysql \
--network red_wp \
-v /opt/mysql_wp:/var/lib/mysql \
-e MYSQL_DATABASE=bd_wp \
-e MYSQL_USER=user_wp \
-e MYSQL_PASSWORD=asdasd \
-e MYSQL_ROOT_PASSWORD=asdasd \
mariadb
```

```
docker run -d --name servidor_mysql --network red_wp -v C:/Users/fabri/Daw-2B/DAW-Fabricio-Garcia-Angeles/DAW-25-26-FabricioGarciaAngeles/Docker/ACT6:/var/lib/mysql -e MYSQL_DATABASE=bd_wp -e MYSQL_USER=user_wp -e MYSQL_PASSWORD=asdasd -e MYSQL_ROOT_PASSWORD=asdasd mar
```

donde –network indica la red privada que empleamos. La opción -v debe ajustarse con una ruta real de nuestro sistema de archivos y los demás parámetros permite crear una base de datos, crear un usuario con contraseña y ajustar la contraseña del root

- c) Creamos ahora el contenedor de wordpress con:

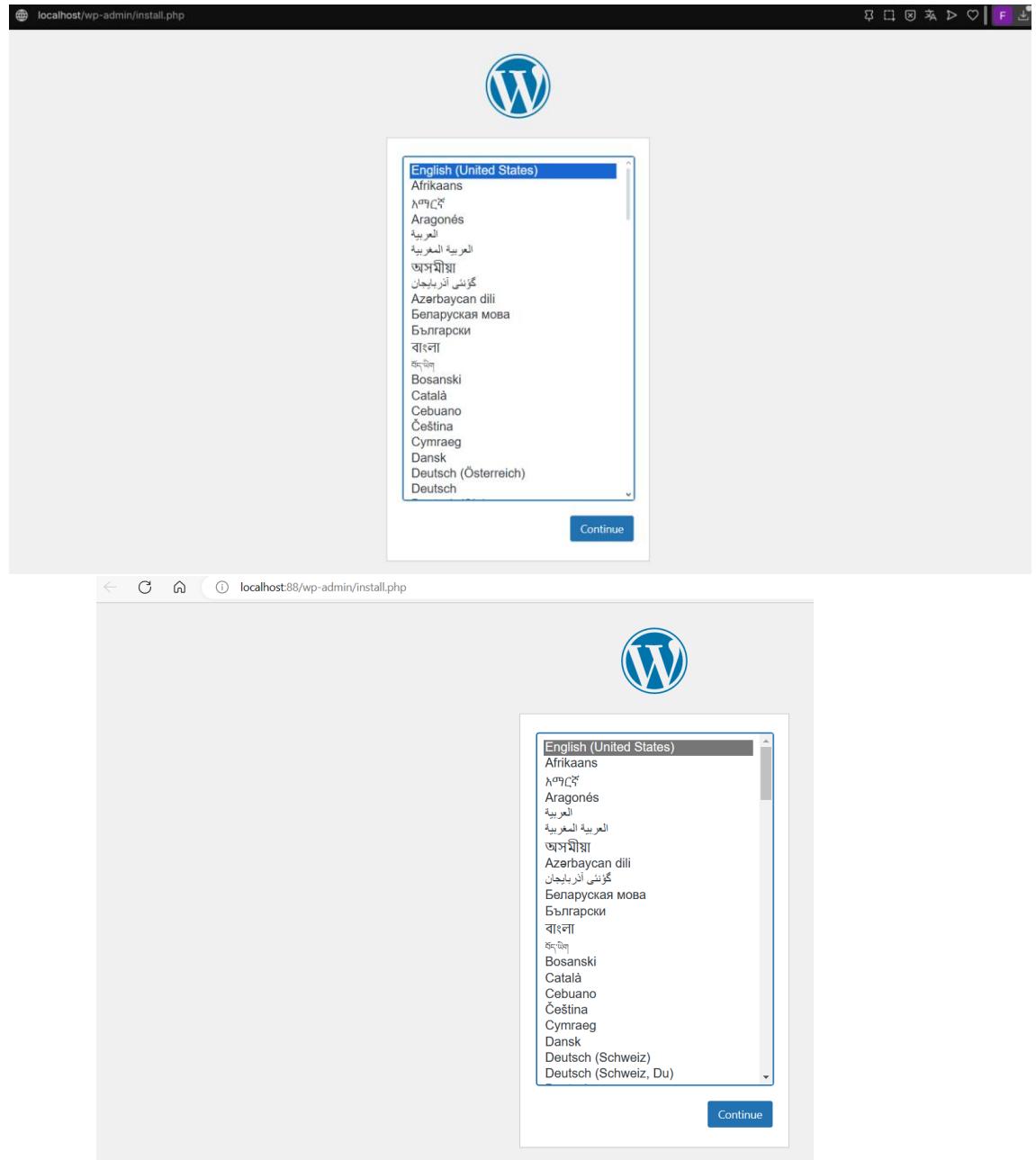
```
docker run -d --name servidor_wp \
--network red_wp \
-v /opt/wordpress:/var/www/html/wp-content \
-e WORDPRESS_DB_HOST=servidor_mysql \
-e WORDPRESS_DB_USER=user_wp \
-e WORDPRESS_DB_PASSWORD=asdasd \
-e WORDPRESS_DB_NAME=bd_wp \
-p 80:80 \
wordpress
```

```
Run 'docker run --help' for more information
PS C:\Users\fabri> docker run -d --name servidor_wp --network red_wp -v C:/Users/fabri/Daw-2B/DAW-Fabricio-Garcia-Angeles/Docker/ACT6_2:/var/www/html/wp-content -e WORDPRESS_DB_HOST=servidor_mysql -e WORDPRESS_DB_USER=user_wp -e WORDPRESS_DB_PASSWORD=asdasd -e WORDPRESS_DB_NAME=bd_wp -p 80:80 wordpress
e0f31ac4f350f2aac90cbc2a67433cc186f440cc730b2c4a0baa60dfccce2f1c
```

```
docker run -d --name servidor_wp --network red_wp -v C:/Users/fabri/Daw-2B/DAW-Fabricio-Garcia-Angeles/DAW-25-26-FabricioGarciaAngeles/Docker/ACT6_2:/var/www/html/wp-content -e WORDPRESS_DB_HOST=servidor_mysql -e WORDPRESS_DB_USER=user_wp -e WORDPRESS_DB_PASSWORD=asdasd -e WORDPRESS_DB_NAME=bd_wp -p 80:80
wordpress
```

De nuevo indicamos la red a usar y con la opción -v (que debemos ajustar) creamos un volumen para los datos de wordpress. Los demás valores permiten indicar el HOST (contenedor) donde está mariadb y los valores de usuario, contraseña y base de datos a emplear. En este caso se está mapeando el puerto 80 del contenedor con el puerto 80 de la máquina (esto puede ajustarse si es necesario)

- d) Acceder desde un navegador de la máquina a la dirección <http://localhost> y completar la instalación de Wordpress para verificar que todo funciona bien



3. En este caso vamos a usar una aplicación de visualización de temperaturas que emplea dos componentes: un backend con una API REST que permite la consulta de temperaturas en España y un frontend escrito en Python que permite hacer el acceso a municipios. Para ello:

- a) Creamos una red bridge privada para los dos contenedores con

```
docker create network red_temperaturas
```

- b) Lanzamos el container del backend con:

```
docker run -d --name temperaturas-backend --network red_temperaturas
avelinopef/temperaturas_back
```

c) Creamos ahora el contenedor del frontend con:

```
docker run -d -p 80:3000 --name temperaturas-frontend --network red_temperaturas avelinopef/temperaturas_front
```

En este caso se puede mapear el puerto 3000 del servicio con otro puerto diferente del 80 si es necesario

```
2 docker network create red_temperaturas  
3 docker run -d --name temperaturas-backend --network red_temperaturas avelinopef/temperaturas_back  
4 docker run -d --name temperaturas-backend-U6 --network red_temperaturas avelinopef/temperaturas_back  
5 docker run -d -p 80:3000 --name temperaturas-frontend-U6 --network red_temperaturas avelinopef/temperaturas_front
```

d) Con un navegador accedemos a <http://localhost> y verificamos que se puede acceder a la aplicación



4. Vamos ahora a lanzar un contenedor de tomcat. Para ello:

a) `docker run -d --name aplicacionjava -v c:\users\avelino\tomcat:/usr/local/tomcat/webapps/ tomcat:9.0`

En este caso debe ajustarse la ruta del volumen para emplear una carpeta de nuestro sistema de archivos

```
PS C:\Users\fabri> docker run -d --name aplicacionjava -v C:\Users\fabri\DAW-2B\DAW-Fabricio-Garcia-Angeles\Dock er\ATC6_3:/usr/local/tomcat/webapps/ tomcat:9.0  
9b3c12c2ec244eabd113cc0037e4f5a1578343b2567793412001ac9f5fedd98  
PS C:\Users\fabri>
```

b) copiaremos en nuestra carpeta del volumen el fichero sample.war que se entrega

DAW-25-26-FabricioGarciaAngeles > Docker > ATC6_3 > sample >					Buscar en sample
		↑ Ordenar	Ver	...	
Nombre		Fecha de modificación	Tipo	Tamaño	
📁 images		16/01/2026 21:12	Carpeta de archivos		
📁 META-INF		16/01/2026 21:12	Carpeta de archivos		
📁 WEB-INF		16/01/2026 21:12	Carpeta de archivos		
⚙ hello.jsp		30/07/2007 18:26	Archivo de origen Java ...	1 KB	
🌐 index.html		30/07/2007 18:26	Opera GX Web Document	1 KB	
📄 sample.war		16/01/2026 20:43	Archivo WAR	5 KB	



## Sample "Hello, World" Application

This is the home page for a sample application used to illustrate the source directory organization of a web application utilizing the MVC pattern. To prove that they work, you can execute either of the following links:

- To a JSP page.
  - To a servlet.

Ahora mismo funciona en localhost sample

- c) A continuación, vamos a emplear el contenedor de nginx para hacer que las peticiones a <http://localhost> se redirijan a [http://IP\\_aplicacionjava:8080/sample/](http://IP_aplicacionjava:8080/sample/)
  - d) Finalmente probaremos desde un navegador que al cargar la URL <http://localhost> se nos redirige a la aplicación java sample



# Sample "Hello, World" Application

This is the home page for a sample application used to illustrate the source direc

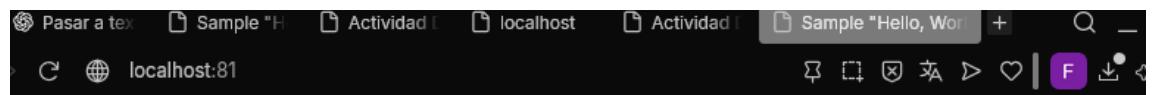
To prove that they work, you can execute either of the following links:

- To a JSP page.
  - To a servlet.

```
+ FullyQualifiedErrorId : RedirectionNotSupportedException

$ C:\Users\fabri> docker exec -it redireccion-Java bash
$ 
$ Error response from daemon: container d42b52b7423b6da65d9b0ca03878184578cc20acf09deb94b644605c3269d45b is not running
$ C:\Users\fabri> docker exec -it redireccion-Java bash
$ 
$ root@d42b52b7423b:/# chmod -R 755 /usr/share/nginx/html
$ root@d42b52b7423b:/# 
```

Hay que cambiar desde la consola del nginx los permisos



## Sample "Hello, World" Application

This is the home page for a sample application used to illustrate the source directory organization of a web application utilizing the principles outlined in the Application Developer's Guide.

ove that they work, you can execute either of the following links:

To a [JSP page](#).

To a [servlet](#).