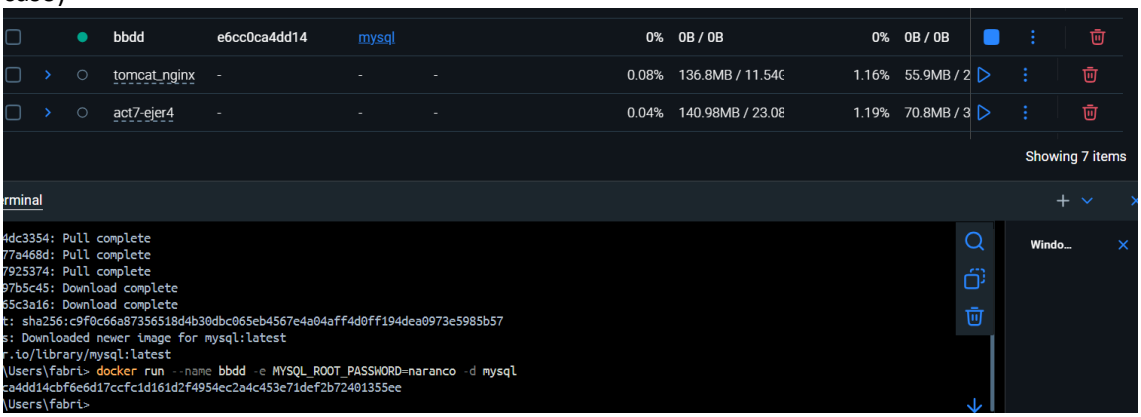
Contenedores Docker. Conectando aplicaciones

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
   ```

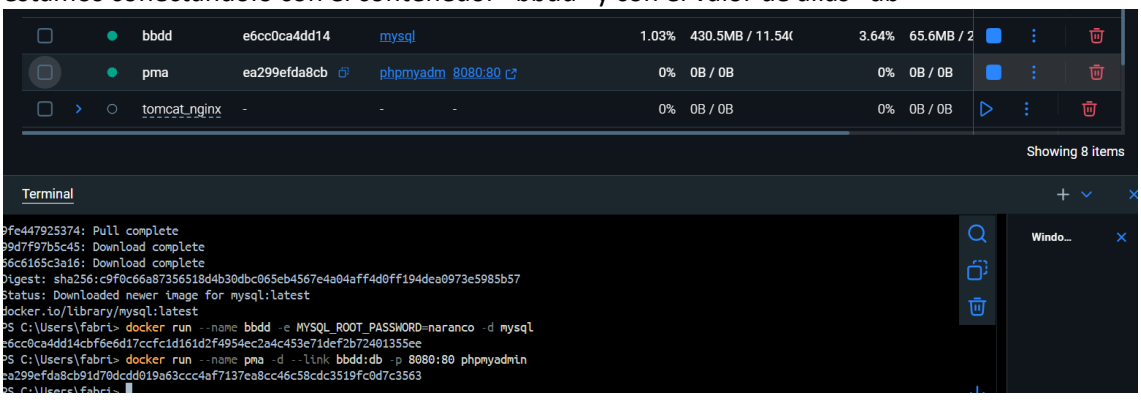
   Le estamos asignando el nombre “bbdd” al nuevo contenedor y estamos estableciendo un valor de entorno de para la contraseña de root (“naranco” en este caso)



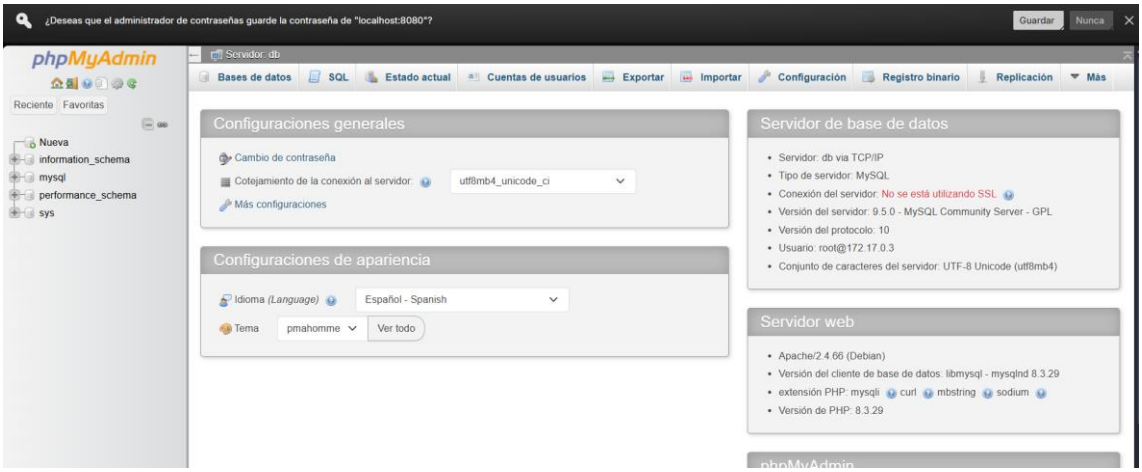
   c) Creamos ahora el contenedor para phpmyadmin con:

   ```
   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”



   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)



2. 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:

   a) 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

   b) 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:

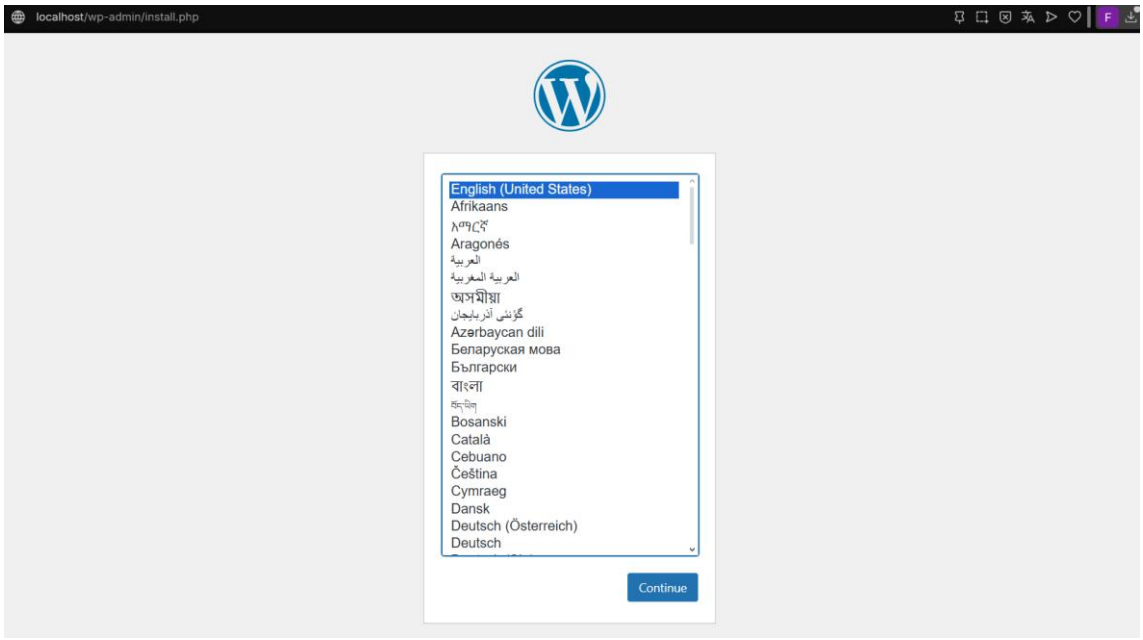
```
docker run -d --name servidor_wp \
              --network red_wp \
              -v /opt/wordpress:/var/www/html/wp-content \
              -e WORDPRESS_DB_HOST=servidor_mysql \
              -e WORDPRESS_DB_USER=user_wp \
              -e WORDPRESS_DB_PASSWORD=asdasd \
              -e WORDPRESS_DB_NAME=bd_wp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

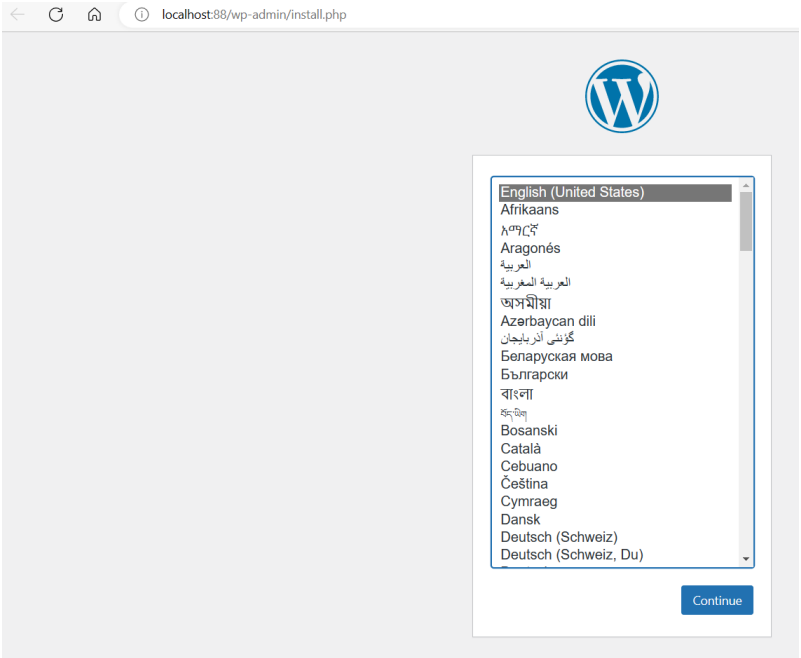
```
Run 'docker run --help' for more information
PS C:\Users\fabri> docker run -d --name servidor_wp --network red_wp -v C:/Users/fabri/Daw-2B/DAW-Fabricio-Garcia-Angeles/DAW-25-26-FabricioGarciaAngeles/Docker/ACT6_2:/var/www/html/wp-content -e WORDPRESS_DB_HOST=servidor_mysql -e WORDPRESS_DB_USER=user_wp -e WORDPRESS_DB_PASSWORD=asdasd -e WORDPRESS_DB_NAME=bd_wp -p 80:80 wordpress
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. Loa 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 maquina (esto puede ajustarse si es necesario)

d) Acceder desde un navegador de la maquina 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 componenetes 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 brigde privada para los dos contenedores con

   ```
   docker create network red_temperaturas
   ```

   b) Lanzamos el conedor del backend con:

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

c) Creamos ahora el contendor 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

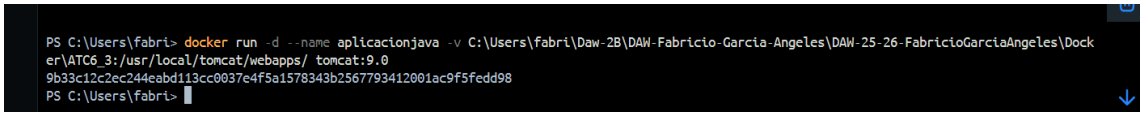
```
2 docker network create red_temperaturas
3 docker run -d --name temperaturas-backend --network red_temperaturas avelinopef/temperaturas_back
4 docker run -d --name temperaturas-backend-U6 --network red_temperaturas avelinopef/temperaturas_back
5 docker run -d -p 80:3000 --name temperaturas-frontend-U6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

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

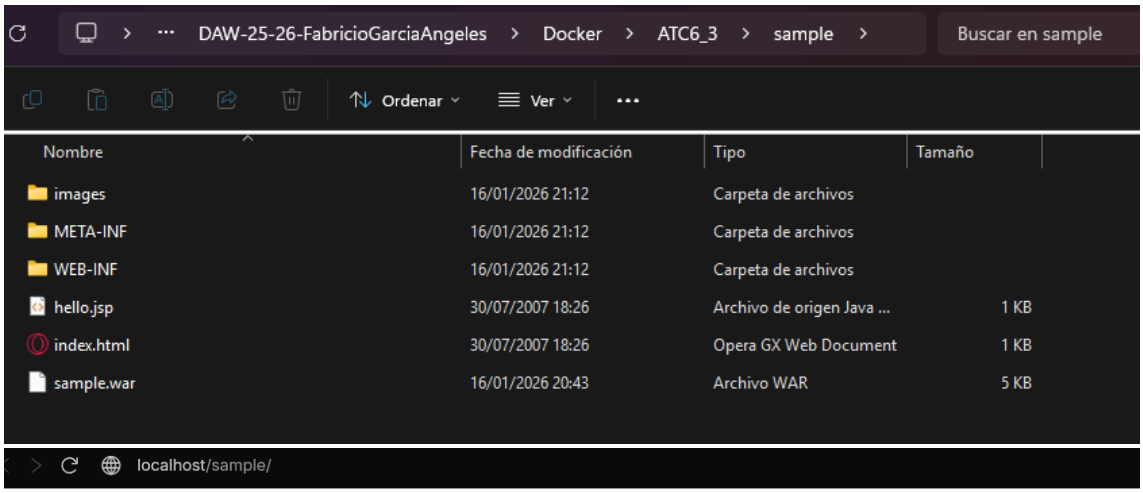
4. Vamos ahora a lanzar un contenedor de tomcat. Para ello:

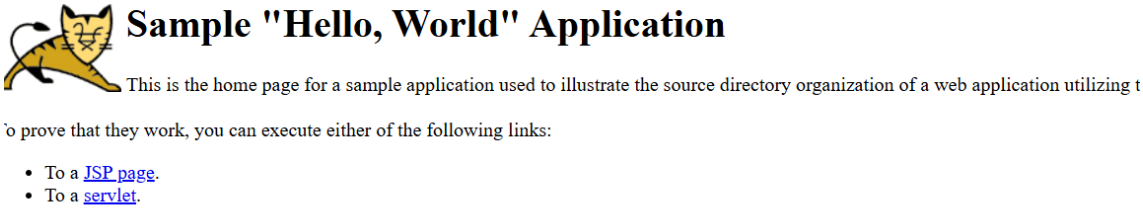
a) `docker run -d --name aplicacionjava -v c:\users\avelino\tomcat:/usr/local/tomcat/webapps/ tomcat:9.0`

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



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

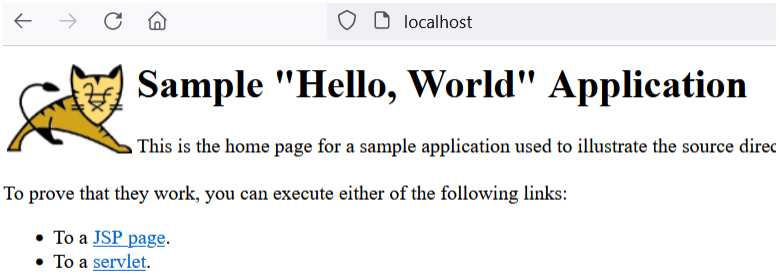


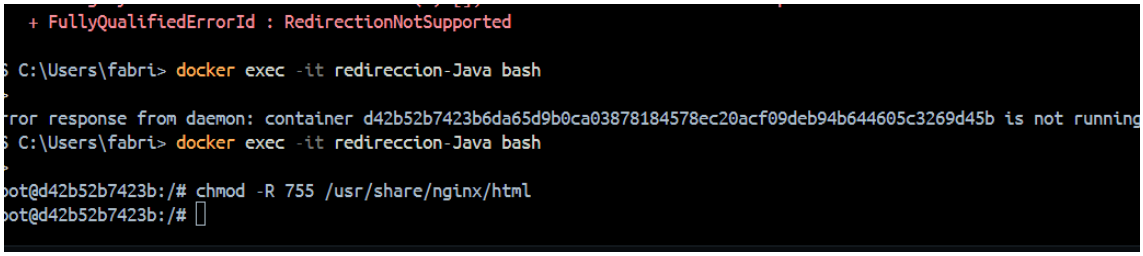


Ahora mismo funciona en localhost sample

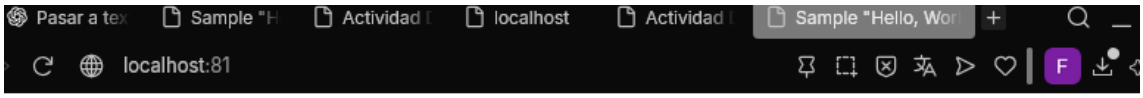
c) A continuación, vamos a emplear el contenedor de nginx para hacer que las peticiones a http://localhost se redirijan a http://IP_aplicacionjava:8080/sample/

d) Finalmente probaremos desde un navegador que al cargar la URL http://localhost se nos redirige a la aplicación java sample





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.