**Unidad didáctica nº 0- Actividad de Desarrollo-**

**"Dockers "**

**Servicios en Red e Internet – 2º ASIR**

**I.E.S. Miguel Herrero**

**Curso 2021-22**

**Objetivos:** Dockers

**Duración:** 4 h

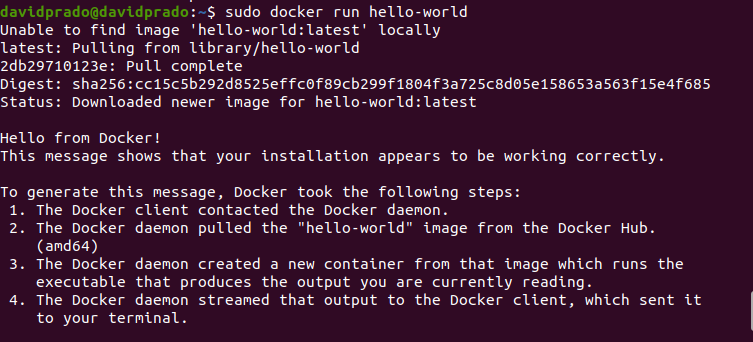
**Recursos:**

Curso “Introducción a Docker: https://iesgn.github.io/curso\_docker\_2021/

**1ª PARTE**

1.- Instala docker en una máquina y busca cómo configurarlo para que se pueda usar con un usuario sin privilegios.

2.- Ejecuta un contenedor a partir de la imagen hello-word. Comprueba que nos devuelve la salida adecuada. Comprueba que no se está ejecutando. Lista los contenedores que están parado. Borra el contenedor.



3. Vamos a probar la imagen **ubuntu**.

a) Crea un contenedor interactivo desde una imagen ubuntu. Instala el paquete nano.

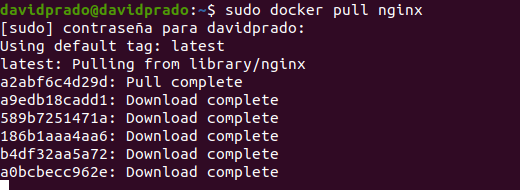
Prueba a editar un fichero llamado prueba.txt.

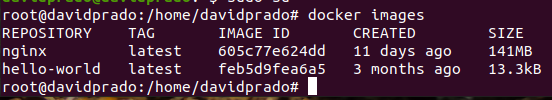
b) Para ahora el contenedor.

c) Vuelve a arrancarlo. ¿Tiene el paquete instalado?

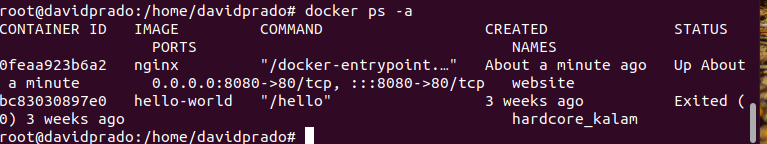
d) Crea un nuevo contenedor interactivo desde la misma imagen. ¿Tiene nano instalado? ¿Por qué?

4. Crea un contenedor demonio con un servidor **nginx**, usando la imagen oficial de **nginx**. Accede al navegador web y comprueba que el servidor web esta funcionando.













5. Crea un contenedor con la aplicación **mysql**, mirando la documentación en docker Hub, para personalizar el nombre de la base de datos, así como el nombre del usuario y su contraseña.

6.- Vamos a entregar el ejercicio 4 con algunas modificaciones. Crearemos un contenedor demonio a partir de la imagen nginx, el contenedor se debe llamar servidor\_web y se debe acceder a él utilizando el puerto 8181 del ordenador donde tengas instalado docker. Realiza lo siguiente:

Crea el contenedor y muestra que el contenedor está funcionando.

Accede al servidor web utilizando un navegador web (recuerda que tienes que acceder a la ip del ordenador donde tengas instalado docker)

Muestra las imágenes que tienes en tu registro local.

Elimina el contenedor (recuerda que antes debe estar parado el contenedor).

7.- Realiza las preguntas 4 y 6 con un servidor web apache usando la imagen oficial de **httpd**.

**2ª PARTE**

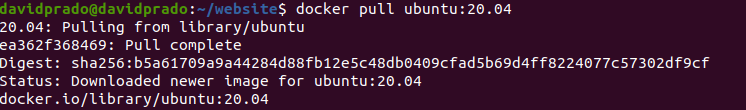
1.- Descarga las siguientes imágenes:

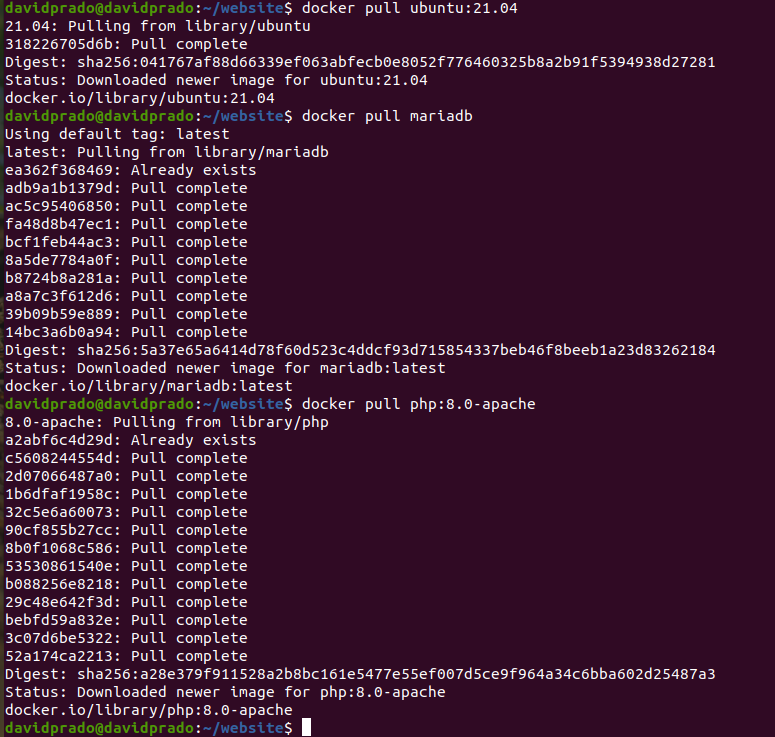
a) ubuntu:20.04

b) ubuntu:21.04

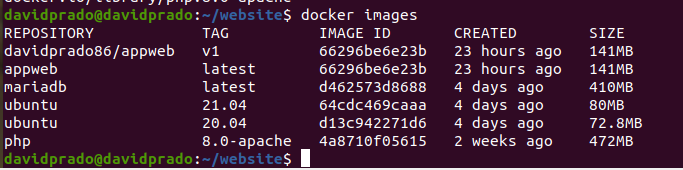
c) mariadb

d) php:8.0-apache

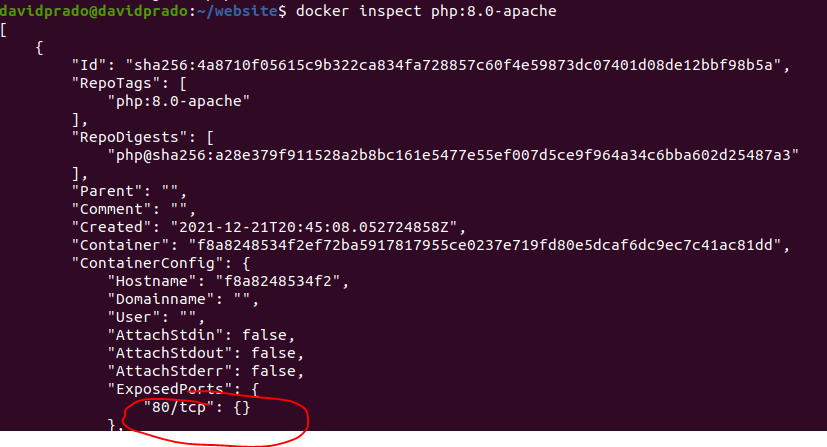




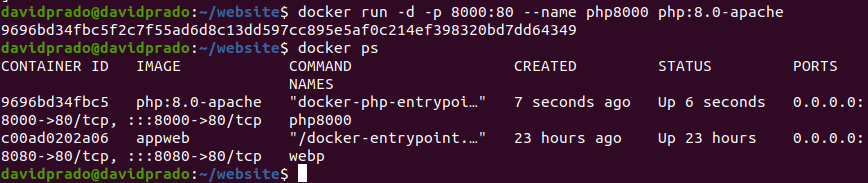
2. Muestras las imágenes que has descargado



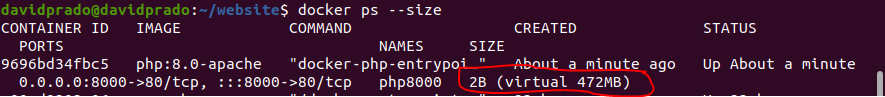
3. Haz un inspect para ver qué puerto expone la imagen php:8.0-apache



4. Crea un contenedor demonio con la imagen php:8.0-apache. Expón el puerto 8000 de tu máquina al puerto 80 del contenedor.



5.- Comprueba el tamaño del contenedor en el disco duro.



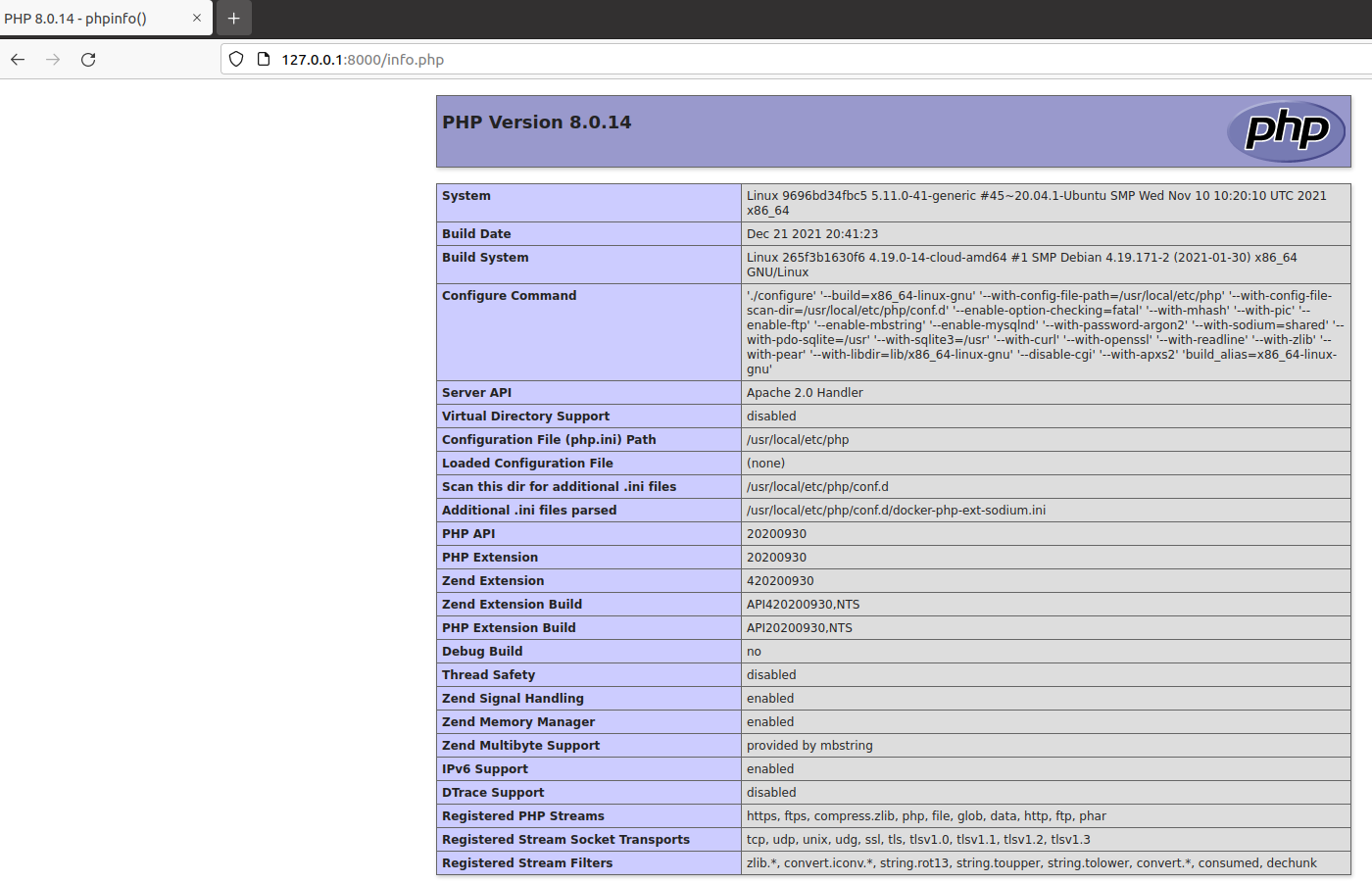
5. Crea un fichero info.php con la función phpinfo(); y ponlo en la carpeta /*var/www/html*

**

**

**

6. Accede al servidor a través del navegador



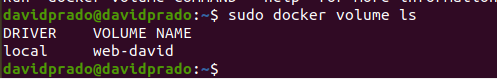
7. Borra la imagen ubuntu:21.04

**3ª PARTE**

**1.- En este primer ejercicio vamos a trabajar con volúmenes.**

a) Crea un volumen con la orden ***docker volume*** que se llame web-TUNOMBRE





b) Crea un contenedor llamado c1 desde la imagen php:8.0-apache donde montes en el directorio /var/www/html (que sabemos que es el DocumentRoot del servidor que nos ofrece esa imagen) el volumen creado antes. Ademas, expón el puerto 8000 al 80 del contenedor.



c) Crea un archivo index.php en el que utilices algo de código PHP y mételo a la carpeta /var/www/html del contenedor. Por ejemplo:

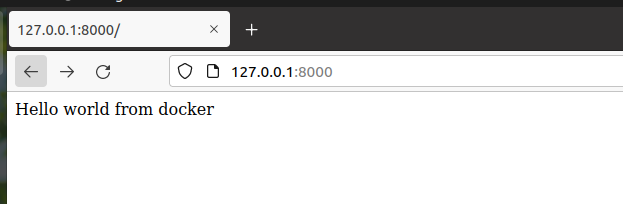
*<?php echo “Hello world from Docker”;?>*

**

**

**

d) Prueba a entrar a través del navegador al servidor

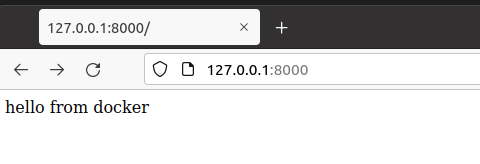


e) Borra el contenedor c1

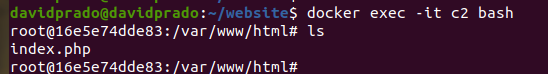


f) Crea un nuevo contenedor c2 y monta el mismo volumen





g) Accede de nuevo desde el navegador. ¿Sigue existiendo el archivo creado antes? **si**



**2. Ahora trabajaremos con bind mounts.**

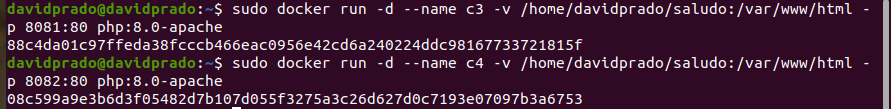
a) Crea un directorio en tu equipo llamado ***saludo*** y dentro de ella crea un fichero llamado index.html con el siguiente contenido (deberás sustituir XXX por tu nombre y apellidos):

*<h1>Hola soy XXXX</h1>*

**

**

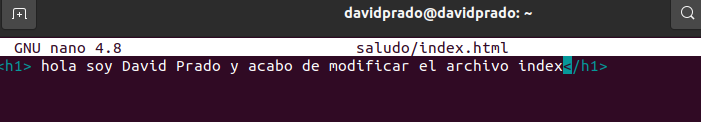
b) Crea dos contenedores llamado c3 y c4 desde la imagen php:8.0-apache donde montes, bind mount, en el directorio /var/www/html el directorio creado en el punto anterior. Uno de ellos vamos a acceder con el puerto 8181 y el otro con el 8282



c) Accede a ambos servidores web a través del navegador.

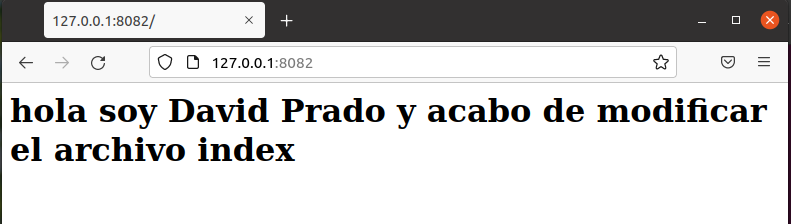


d) Modifica el contenido del fichero ~/saludo/index.html

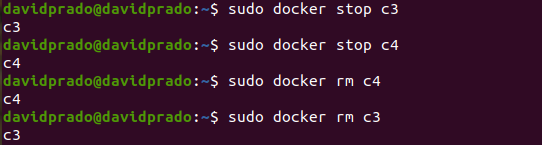


e) Prueba a refrescar el navegador. ¿Se han modificado los datos?

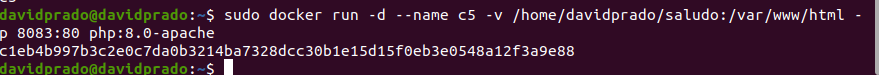




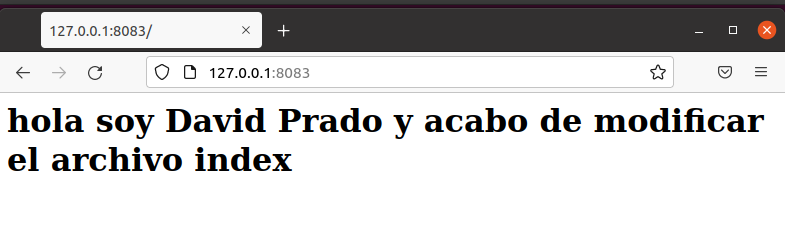
f) Borra los contenedores c3 y c4.



g) Crea un nuevo contenedor c5 y monta el mismo directorio de antes. Vamos a acceder en el puerto 8383.



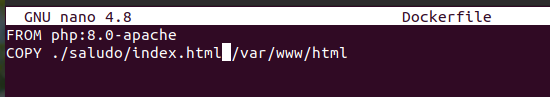
h) Prueba de nuevo a acceder al contenedor desde el navegador. ¿Se sigue viendo el mismo contenido?



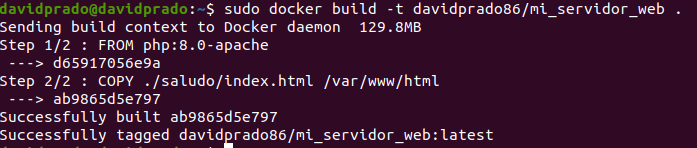
**4ª PARTE**

**1.- Creación de imágenes a partir de un Dockerfile:**

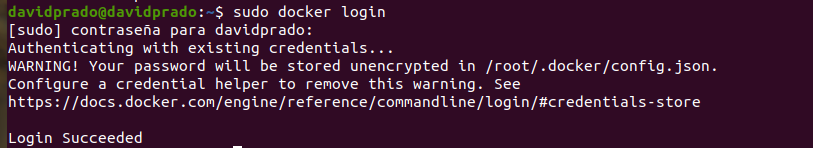
i) Crea un fichero Dockerfile que permita crear una imagen con un servidor web sirviendo el sitio anterior: ***saludo***. Puedes usar la imagen php:8.0-apache .

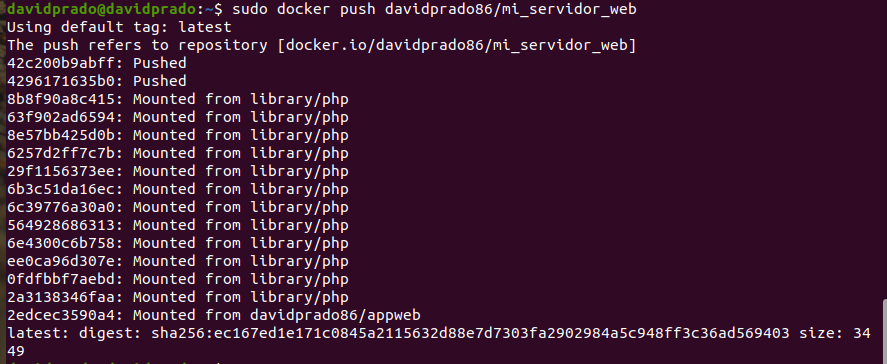


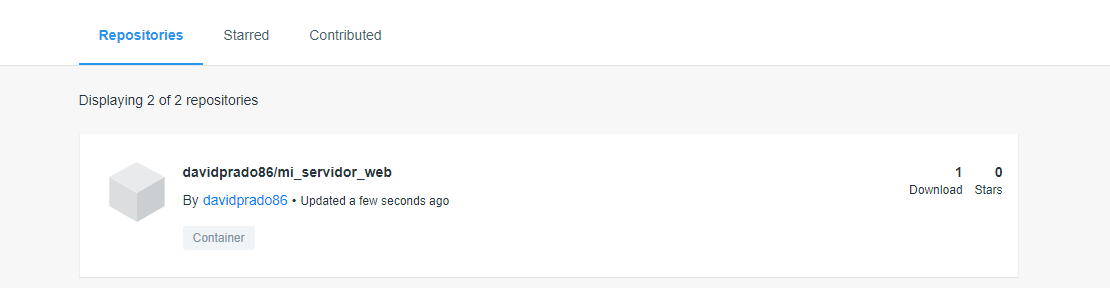
j) Ejecuta el comando dockDockerfileer que crea la nueva imagen. La imagen se debe llamar ***<tu\_usuario\_docker\_hub>/mi\_servidor\_web.***

******

k) Conéctate a Docker Hub y sube la imagen que acabas de crear.







l) Descarga la imagen en otro ordenador donde tengas docker instalado, y crea un contenedor llamado c6 a partir de ella. Vamos a acceder en el puerto 8484 (Si no tienes otro ordenador con docker instalado, borra la imagen en tu ordenador y bájala de Docker Hub).

**5ª PARTE**

Vamos a crear dos redes de ese tipo (BRIDGE) con los siguientes datos:

**Red1**

Nombre: red1

Dirección de red: 172.28.0.0

Máscara de red: 255.255.0.0

Gateway: 172.28.0.1

**Red2**

Nombre: red2

Es resto de los datos será proporcionados automáticamente por Docker.

1.- Poner en ejecución un contenedor de la imagen ubuntu:20.04 que tenga como hostname host1, como IP 172.28.0.10 y que esté conectado a la red1. Lo llamaremos u1.

Entrar en ese contenedor e instalar la aplicación ping (apt update && apt install inetutils-ping).

2.- Poner en ejecución un contenedor de la imagen ubuntu:20.04 que tenga como hostname host2 y que esté conectado a la red2. En este caso será docker el que le de una IP correspondiente a esa red. Lo llamaremos u2.

Entrar en ese contenedor e instalar la aplicación ping (apt update && apt install inetutils-ping).

Muestra la configuración de red del contenedor u1.

Muestra la configuración de red del contenedor u2.

Muestra desde cualquiera de los dos contenedores se pueda ver que no podemos hacer ping al otro ni por ip ni por nombre.

Muestra que si conectamos el contenedor u1 a la red2 (con docker network connect), desde el contenedor u1, tenemos acceso al contenedor u2 mediante ping, tanto por nombre como por ip.

**6ª PARTE**

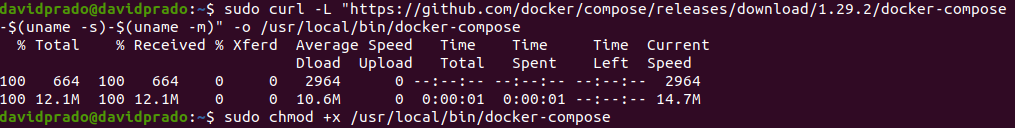
**Recursos:**

**Github**-><https://github.com/jmlcas/joomla>

**Instalando joomla con docker. The easy way.->**https://www.youtube.com/watch?v=AQ4f1Jz8228

**Creating a Joomla Docker container.->**https://www.hamrodev.com/en/app-development/joomla-docker-tutorial

1. Instala docker-compose.



2. Vamos a crear una instalación de joomla. Para ello tendremos dos servicios: joomla y basedatos.

a) El primero por el puerto 8888. El segundo por el puerto 3306 (si ya tienes algún servicio en el puerto 3306 usa el 33060).

b) El contenedor del primero se llamará joomla-tunombre. El contenedor del segundo bd\_joomla

c) Usa bind mounts. En joomla que apunte a /var/www/html*.* En basedatos a /var/lib/mysql

3. Levanta el escenario con docker-compose

4. Muestra los contenedores en ejecución con docker-compose

5. Entra al bash de joomla y muestra los ficheros y carpetas de /var/www/html

6. Para y borra los contenedores (incluyendo volúmenes) con docker-compose