



**Natalia Coronado Romero 1ºDAM**

## ÍNDICE

Introducción.....	3
Objetivo de la memoria.....	3
Material utilizado.....	3
Desarrollo.....	4
Actividad 1 - GitHub.....	4
Actividad 2 – Casos Prácticos.....	5
Unidad 3.....	5
Caso Práctico 2.....	5
Caso Práctico 3.....	10
Unidad 5.....	11
Caso Práctico 1.....	11
Unidad 6.....	15
Caso Práctico 1.....	15
Actividad 3 – Kubernetes.....	18
Problemas encontrados y sugerencias.....	19
Conclusión.....	19
Bibliografía/webgrafía.....	19

## Introducción

En este curso intentamos introducir al uso de Docker a través de casos prácticos y selectivos para empezar a tener una base.

## Objetivo de la memoria

El objetivo de esta práctica es aprender las dinámicas de los contenedores y distintos usos que pueden tener de utilidad como su almacenaje, su conexión y distintos servicios.

## Material utilizado

**Marca y Modelo del Procesador:** Intel Pentium CPU G4400 @ 3.30GHz

**Tipo Memoria RAM (memoria y slots):** 4GB RAM, 2400 MHz

**Tipo de dispositivo de almacenamiento**

**capacidad (GiB):** HDD 1TB

**Programas utilizados:** Spectacle

# Desarrollo

## Actividad 1 - GitHub

Se ha utilizado GitHub para poder subir los casos prácticos y todo el trabajo que se vaya realizando. Para eso se ha creado un repositorio que se utilizará para este módulo específico.

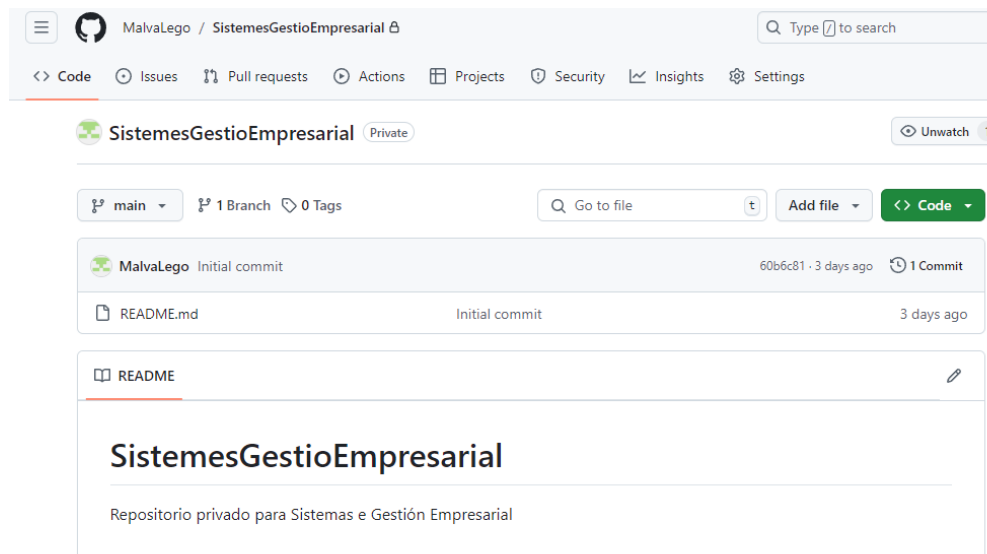


Figura 1: Repositorio de GitHub

Este es el enlace al repositorio:

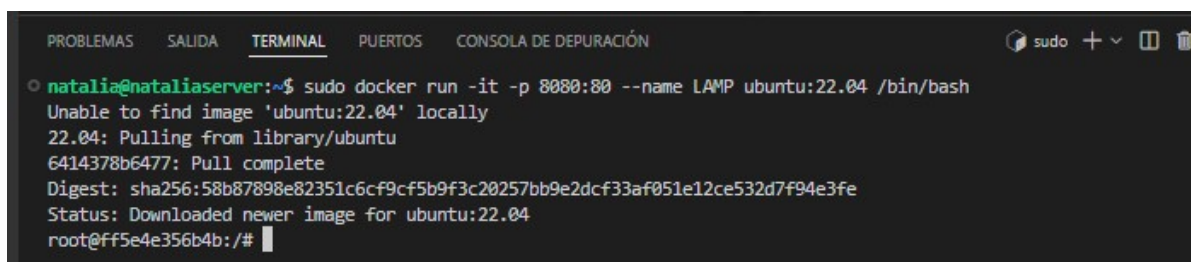
<https://github.com/MalvaLego/SistemasGestioEmpresarial.git>

## Actividad 2 – Casos Prácticos

### Unidad 3

#### Caso Práctico 2

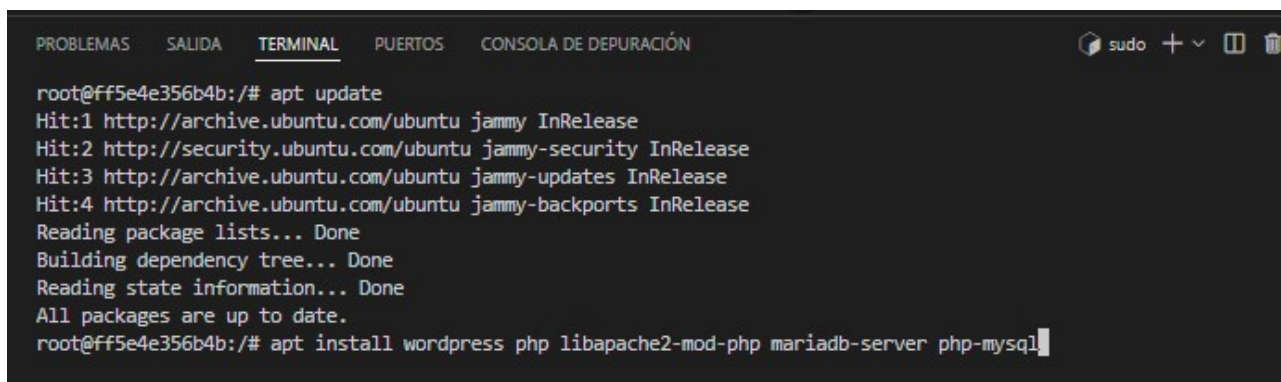
Para esta práctica instalaremos LAMP + Wordpress en un contenedor. Para este contenedor pondremos como imagen oficial la de ubuntu y lo llamaremos LAMP, en el que indicaremos que utilizará el puerto 8080.



```
PROBLEMAS  SALIDA  TERMINAL  PUERTOS  CONSOLA DE DEPURACIÓN  sudo + - [ ] [X]
natalia@nataliaserver:~$ sudo docker run -it -p 8080:80 --name LAMP ubuntu:22.04 /bin/bash
Unable to find image 'ubuntu:22.04' locally
22.04: Pulling from library/ubuntu
6414378b6477: Pull complete
Digest: sha256:58b87898e82351c6cf9cf5b9f3c20257bb9e2dcf33af051e12ce532d7f94e3fe
Status: Downloaded newer image for ubuntu:22.04
root@ff5e4e356b4b:/#
```

Figura 2: Creación del contenedor LAMP

A continuación instalaremos los paquetes necesarios para la instalación de LAMP y wordpress y los actualizaremos.



```
PROBLEMAS  SALIDA  TERMINAL  PUERTOS  CONSOLA DE DEPURACIÓN  sudo + - [ ] [X]
root@ff5e4e356b4b:/# apt update
Hit:1 http://archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:3 http://archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:4 http://archive.ubuntu.com/ubuntu jammy-backports InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
root@ff5e4e356b4b:/# apt install wordpress php libapache2-mod-php mariadb-server php-mysql
```

Figura 3: Instalación de paquetes LAMP+ Wordpress

Para comprobar que todo está bien activamos el servicio Apache y lo probaremos con el local del puerto asignado anteriormente.

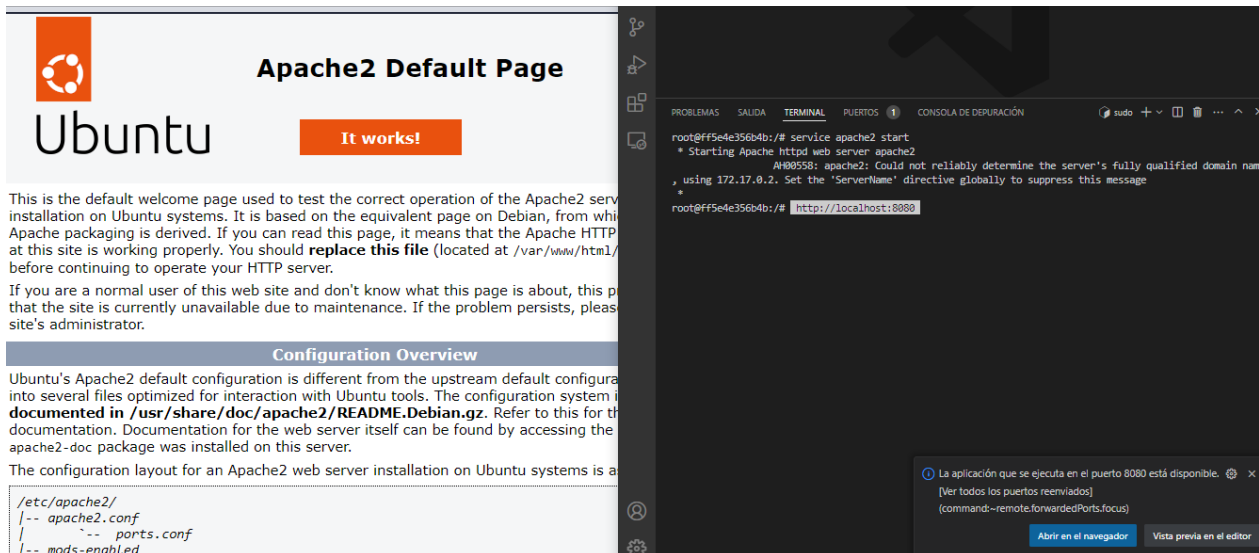


Figura 4: Comprobación local del servicio Apache2

Ya sabemos que el servicio funciona correctamente pero habrá que configurarlo para que haga algo. Para ello crearemos el archivo `/etc/apache2/sites-available/wordpress.conf` con el editor de texto que más os guste, en mi caso he utilizado nano, para configurar el sitio del acceso a Wordpress. Este sería su contenido:

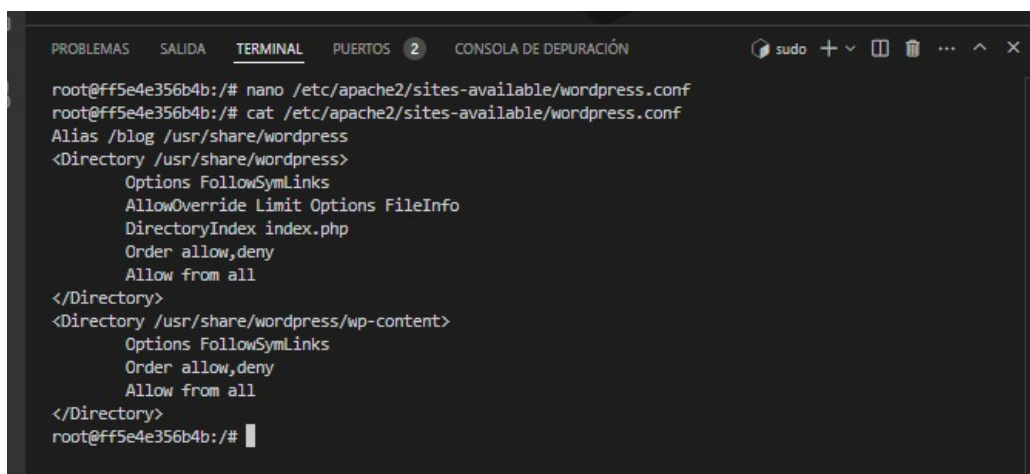
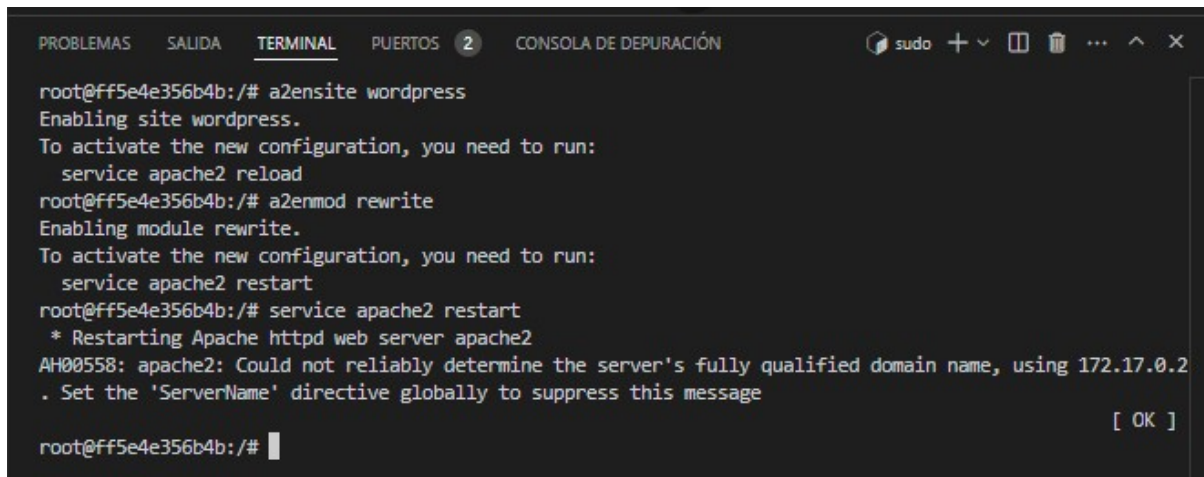


Figura 5: Configuración del archivo wordpress.conf para su acceso



Una vez creado el fichero cargamos el sitio y recargamos el servicio Apache2 con estos comandos:



```
root@ff5e4e356b4b:/# a2ensite wordpress
Enabling site wordpress.
To activate the new configuration, you need to run:
  service apache2 reload
root@ff5e4e356b4b:/# a2enmod rewrite
Enabling module rewrite.
To activate the new configuration, you need to run:
  service apache2 restart
root@ff5e4e356b4b:/# service apache2 restart
* Restarting Apache httpd web server apache2
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.17.0.2
. Set the 'ServerName' directive globally to suppress this message

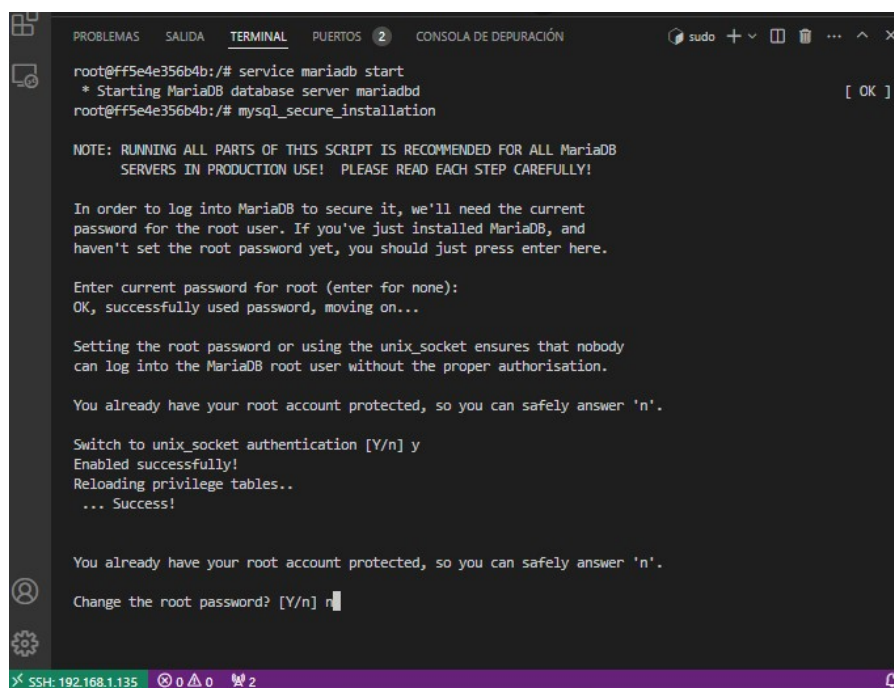
[ OK ]

root@ff5e4e356b4b:/#
```

Figura 6: Reinicio del servicio Apache2 para guardar configuración

Ya tenemos todo listo pero solo hemos configurado el servicio de Apache2, aún nos falta el servicio para la base de datos. Para ello utilizaremos el servicio llamado mariadb, por lo que vamos a ponernos a ello.

Primero arranquemos el servicio como siempre utilizando el start y procedemos a seguir los pasos contestando las preguntas necesarias.



```
root@ff5e4e356b4b:/# service mariadb start
* Starting MariaDB database server mariadbd
root@ff5e4e356b4b:/# mysql_secure_installation

NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MariaDB
SERVERS IN PRODUCTION USE! PLEASE READ EACH STEP CAREFULLY!

In order to log into MariaDB to secure it, we'll need the current
password for the root user. If you've just installed MariaDB, and
haven't set the root password yet, you should just press enter here.

Enter current password for root (enter for none):
OK, successfully used password, moving on...

Setting the root password or using the unix_socket ensures that nobody
can log into the MariaDB root user without the proper authorisation.

You already have your root account protected, so you can safely answer 'n'.

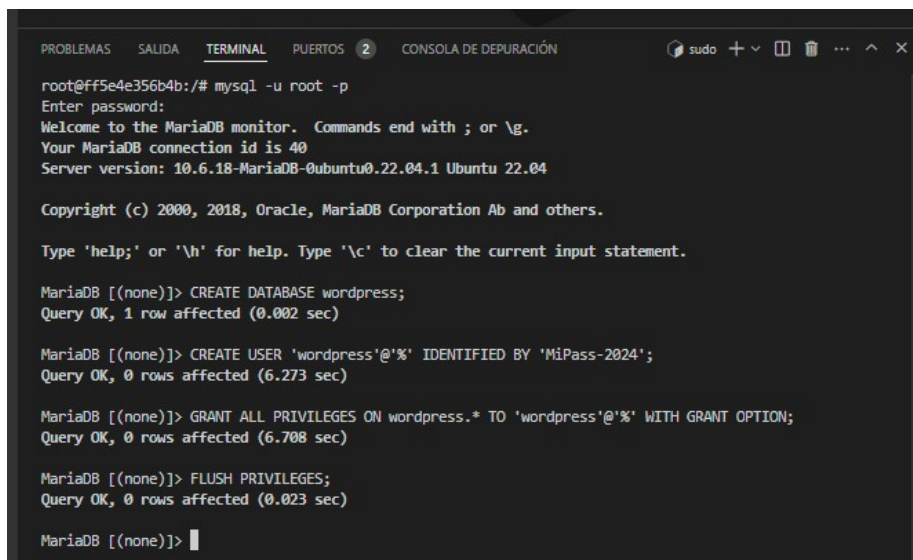
Switch to unix_socket authentication [Y/n] y
Enabled successfully!
Reloading privilege tables..
... Success!

You already have your root account protected, so you can safely answer 'n'.

Change the root password? [Y/n] n
```

Figura 7: Arranque del servicio MariaDB

Una vez hecho todo accederemos a la base de datos con el cliente y crearemos la base de datos llamada wordpress necesaria para el sitio. Es necesario crear un usuario para poder trabajar con él por lo que lo creamos junto a su contraseña y le ponemos todos los permisos en la base e datos.



```
PROBLEMAS SALIDA TERMINAL PUERTOS 2 CONSOLA DE DEPURACIÓN
root@ff5e4e356b4b:/# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 40
Server version: 10.6.18-MariaDB-0ubuntu0.22.04.1 Ubuntu 22.04

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)]> CREATE DATABASE wordpress;
Query OK, 1 row affected (0.002 sec)

MariaDB [(none)]> CREATE USER 'wordpress'@'%' IDENTIFIED BY 'MiPass-2024';
Query OK, 0 rows affected (6.273 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON wordpress.* TO 'wordpress'@'%' WITH GRANT OPTION;
Query OK, 0 rows affected (6.708 sec)

MariaDB [(none)]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.023 sec)

MariaDB [(none)]> |
```

Figura 8: Creación de base de datos y usuario

Ya tenemos todo creado y operativo pero, igual que el otro servicio, se necesita editar un fichero de configuración con todo lo necesario. Editaremos el fichero `/etc/wordpress/config-localhost.php` y al poner los datos siguientes, si entramos al local del puerto asignado ya se podrá ver en el navegador web la instalación para wordpress con interfaz gráfica:

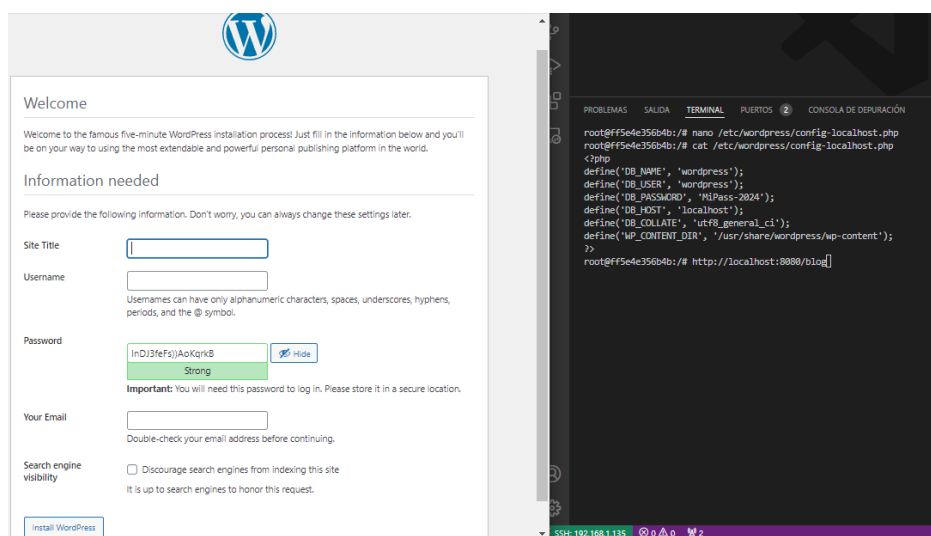


Figura 9: Configuración del archivo `config-localhost.php` y su comprobación al sitio



Editaremos el archivo `/root/.bashrc` ya que este es el que se ejecuta al iniciarse la shell y añadiremos dos líneas de código para que se inicien los servicios.

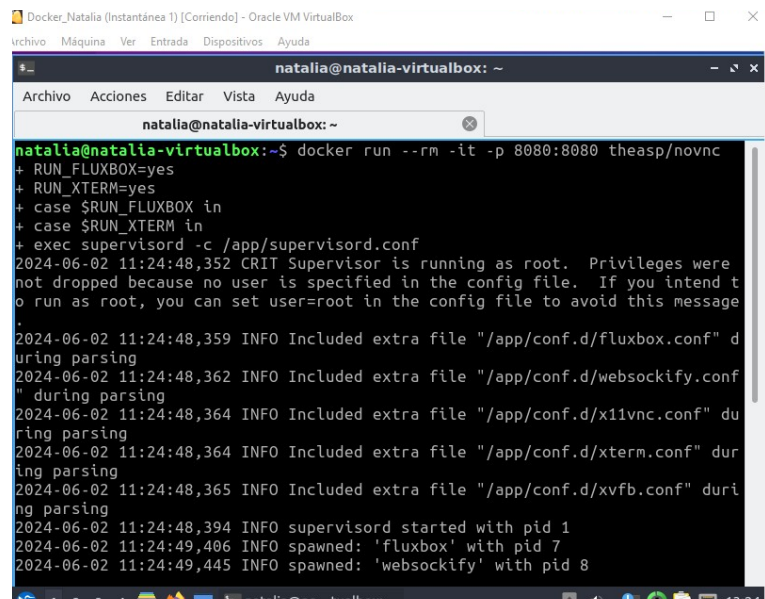
Figura 10: Edición del fichero `.bashrc` para ejecutar al iniciar

The image shows a split-screen view. On the left, a web browser displays a WordPress site with the title 'TITULAZO DEL SITIO' and the subtitle 'Just another WordPress site'. Below the title is a large heading 'Hello world!' followed by a paragraph: 'Welcome to WordPress. This is your first post. Edit or delete it, then start writing!'. At the bottom, it says 'Published September 28, 2024' and 'Categorized as Uncategorized'. On the right, a terminal window shows a series of commands and their outputs. The commands include navigating to a directory, running 'docker start LAMP', 'docker stop LAMP', and 'docker restart LAMP'. The output shows that the 'LAMP' container is already running, and the restart command is successful. The terminal prompt is 'root@ff5e4e356b4b:~#'. The terminal window also shows a list of Docker containers with columns for NAME, ID, IMAGE, STATUS, and CREATED.

Figura 11: Comprobación final del sitio al parar y arrancar

### Caso Práctico 3

En esta práctica pondremos en marcha un servicio VNC (servicio de administración remota) junto con un cliente NoVNC (Cliente para VNC en HTML5 y JavaScript) servido vía web. (Este caso está hecho en ubuntu con interfaz gráfica por complicaciones)



```
Docker_Natalia (Instantánea 1) [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda

natalia@natalia-virtualbox: ~
natalia@natalia-virtualbox: ~
natalia@natalia-virtualbox:~$ docker run --rm -it -p 8080:8080 theasp/novnc
+ RUN_FLUXBOX=yes
+ RUN_XTERM=yes
+ case $RUN_FLUXBOX in
+ case $RUN_XTERM in
+ exec supervisord -c /app/supervisord.conf
2024-06-02 11:24:48,352 CRIT Supervisor is running as root. Privileges were
not dropped because no user is specified in the config file. If you intend t
o run as root, you can set user=root in the config file to avoid this message
.
2024-06-02 11:24:48,359 INFO Included extra file "/app/conf.d/fluxbox.conf" d
uring parsing
2024-06-02 11:24:48,362 INFO Included extra file "/app/conf.d/websockify.conf"
during parsing
2024-06-02 11:24:48,364 INFO Included extra file "/app/conf.d/x11vnc.conf" du
ring parsing
2024-06-02 11:24:48,364 INFO Included extra file "/app/conf.d/xterm.conf" dur
ing parsing
2024-06-02 11:24:48,365 INFO Included extra file "/app/conf.d/xvfb.conf" duri
ng parsing
2024-06-02 11:24:48,394 INFO supervisord started with pid 1
2024-06-02 11:24:49,406 INFO spawned: 'fluxbox' with pid 7
2024-06-02 11:24:49,445 INFO spawned: 'websockify' with pid 8
```

Figura 12: Creación de contenedor con NOVNC

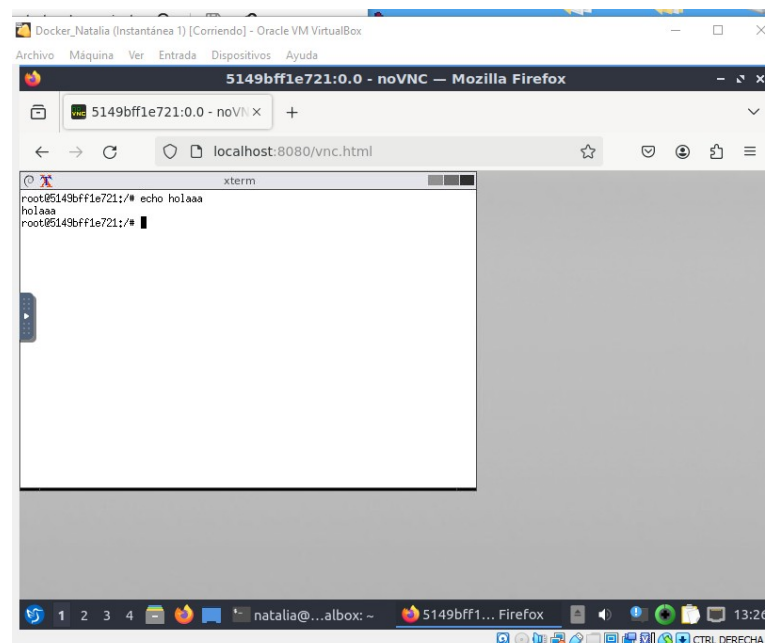


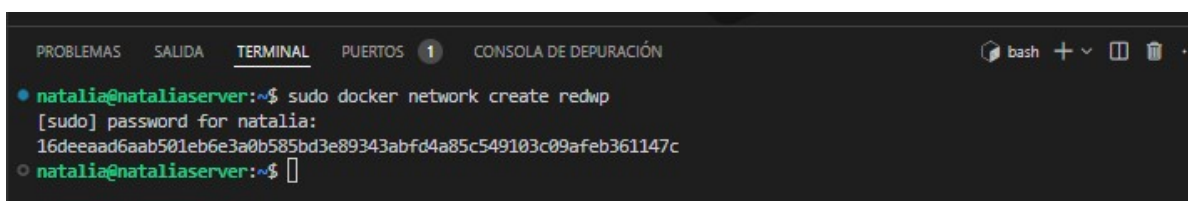
Figura 13: Comprobación de NOVNC

## Unidad 5

### Caso Práctico 1

Para este práctico usaremos una red dónde se conectarán dos contenedores el primero utilizando Apache + PHP, el segundo contendrá un servidor de bases de datos MariaDB. Además, realizaremos un ejemplo de una migración de versión del contenedor MariaDB.


Primero crearemos la red necesaria para esta tarea que llamaremos redwp.



```
PROBLEMAS  SALIDA  TERMINAL  PUERTOS  1  CONSOLA DE DEPURACIÓN
natalia@nataliaserver:~$ sudo docker network create redwp
[sudo] password for natalia:
16deeaad6aab501eb6e3a0b585bd3e89343abfd4a85c549103c09afeb361147c
natalia@nataliaserver:~$
```

Figura 14: Creación de una red

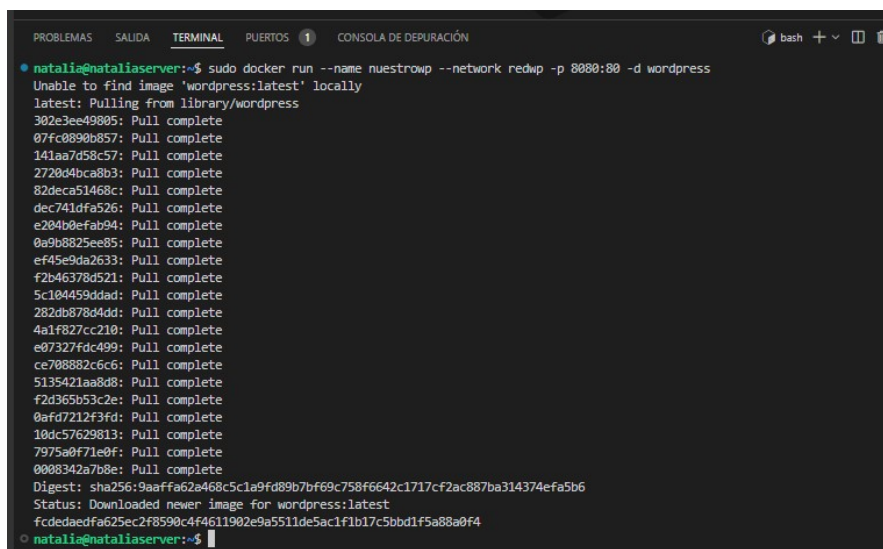
A continuación creamos un contenedor que contendrá la red de bases de datos del servicio mariadb dentro de la red que acabamos de hacer. Este contenedor se llamará nuestromariadb y le pondremos en el mismo comando el usuario y contraseña junto a la base de datos. Utilizaremos la versión de mariadb 10.6 para poder cambiar luego.



```
PROBLEMAS  SALIDA  TERMINAL  PUERTOS  1  CONSOLA DE DEPURACIÓN
natalia@nataliaserver:~$ sudo docker run --name nuestromariadb --network redwp -v /home/natalia/mariadbdata:/var/lib/mysql -e MARIADB
_ROOT_PASSWORD=cefireeroot -e MARIADB_USER=cefireuser -e MARIADB_PASSWORD=cefirepass -e MARIADB_DATABASE=cefiredb -d mariadb:10.6
Unable to find image 'mariadb:10.6' locally
10.6: Pulling from library/mariadb
602d8ad51b81: Pull complete
906bf2208a98: Pull complete
06102c8e12da: Pull complete
79322a8d69e6: Pull complete
9cd48d1e81c5: Pull complete
eb82348957f6: Pull complete
b1dd5e075a7e: Pull complete
712710a7d704: Pull complete
Digest: sha256:1ce48e9be4be695e4e06b775aed83a4805e2fbb8e1219292c4e8bd1483a4f07
Status: Downloaded newer image for mariadb:10.6
501529b38ec072471cf380927f33528919c9f9ab5a74e1d798ca8286f89e63ee
natalia@nataliaserver:~$
```

Figura 15: Creación de un contenedor con bases de datos en la red

Crear el otro contenedor es más fácil ya que se creará de la misma manera que en la actividad anterior solo que indicando que es en la red redwp



```
PROBLEMAS  SALIDA  TERMINAL  PUERTOS  CONSOLA DE DEPURACIÓN
natalia@nataliaserver:~$ sudo docker run --name nuestrowp --network redwp -p 8080:80 -d wordpress
Unable to find image 'wordpress:latest' locally
latest: Pulling from library/wordpress
302e3ee49805: Pull complete
07fc0890b857: Pull complete
141aa7d58c57: Pull complete
2720d4bca8b3: Pull complete
82deca51468c: Pull complete
dec741dfa526: Pull complete
e204b0defab94: Pull complete
0a9b8825ee85: Pull complete
ef45e9da2633: Pull complete
f2b46378d521: Pull complete
5c104459ddad: Pull complete
282db878d4dd: Pull complete
4a1f827cc210: Pull complete
e07327fdc499: Pull complete
ce708882c6c6: Pull complete
5135421aa8d8: Pull complete
f2d365b53c2e: Pull complete
0afd7212f3fd: Pull complete
10dc57629813: Pull complete
7975a0f71e0f: Pull complete
0008342a7b8e: Pull complete
Digest: sha256:9aaffa62a468c5c1a9fd89b7bf69c758f6642c1717cf2ac887ba314374efa5b6
Status: Downloaded newer image for wordpress:latest
fcddaedfa625ec2f8590c4f4611902e9a5511de5ac1f1b17c5bbd1f5a88a0f4
natalia@nataliaserver:~$
```

Figura 16: Creación del segundo contenedor Apache + PHP

Al ir al sitio local del puerto vemos que se puede hacer la instalación del wordpress. Seguimos los pasos con los datos puestos anteriormente en el comando del contenedor y ya lo tendremos instalado.

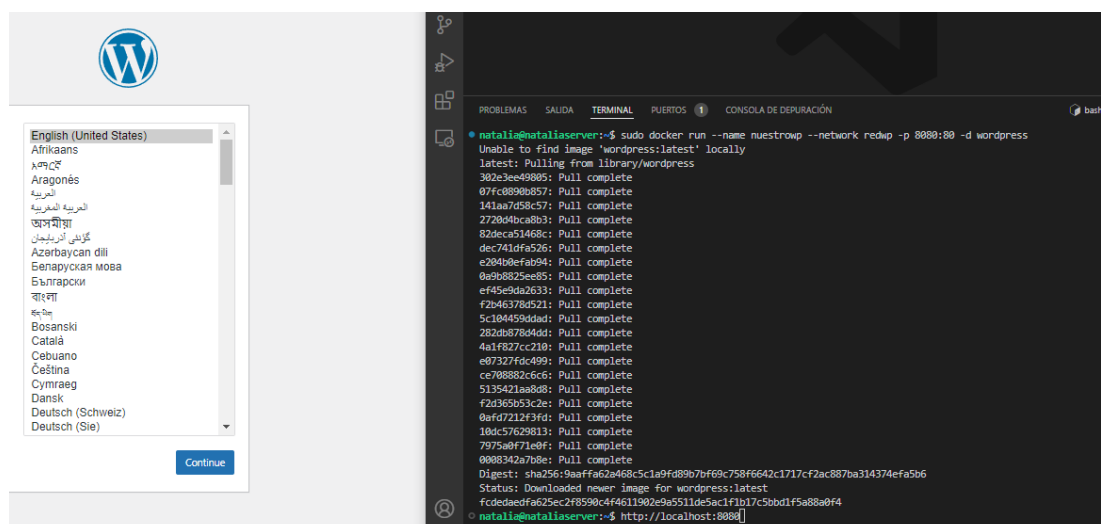


Figura 17: Comprobación en el sitio

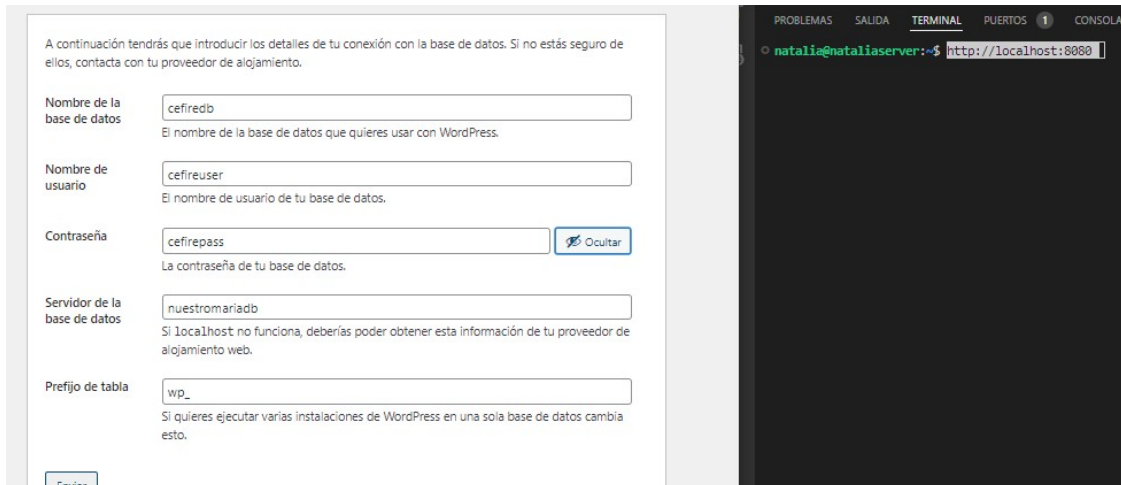


Figura 18: Instalación con los datos puestos en el comando del contenedor

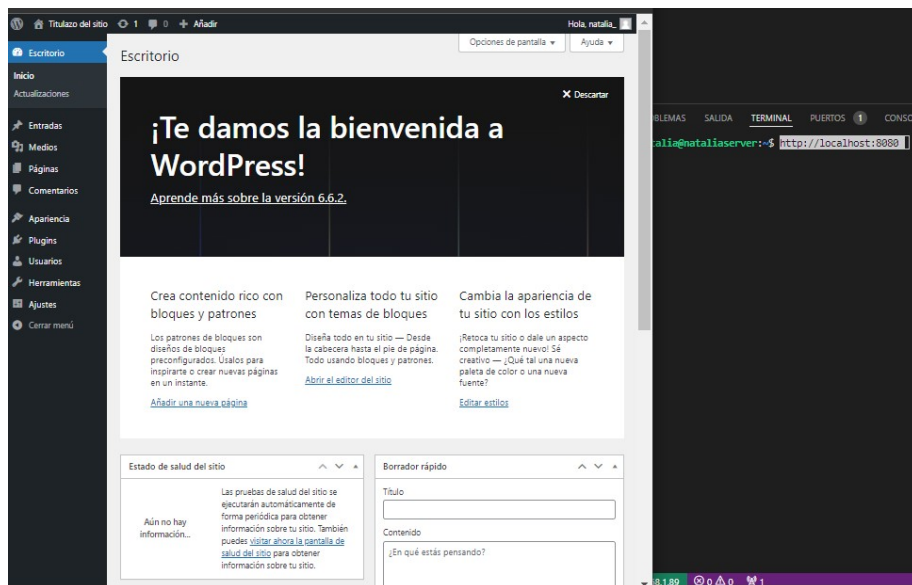


Figura 19: Comprobación del sitio del wordpress totalmente funcional e instalado

Por último vamos a cambiar la versión de mariadb 10.6 a la 10.7, esto es muy útil por si en algún momento necesitamos actualizarlo en caso de necesitamos una nueva.

Paramos el servicio y lo borramos, y utilizando el mismo comando que hicimos para crearlo lo ponemos otra vez pero cambiando la versión.

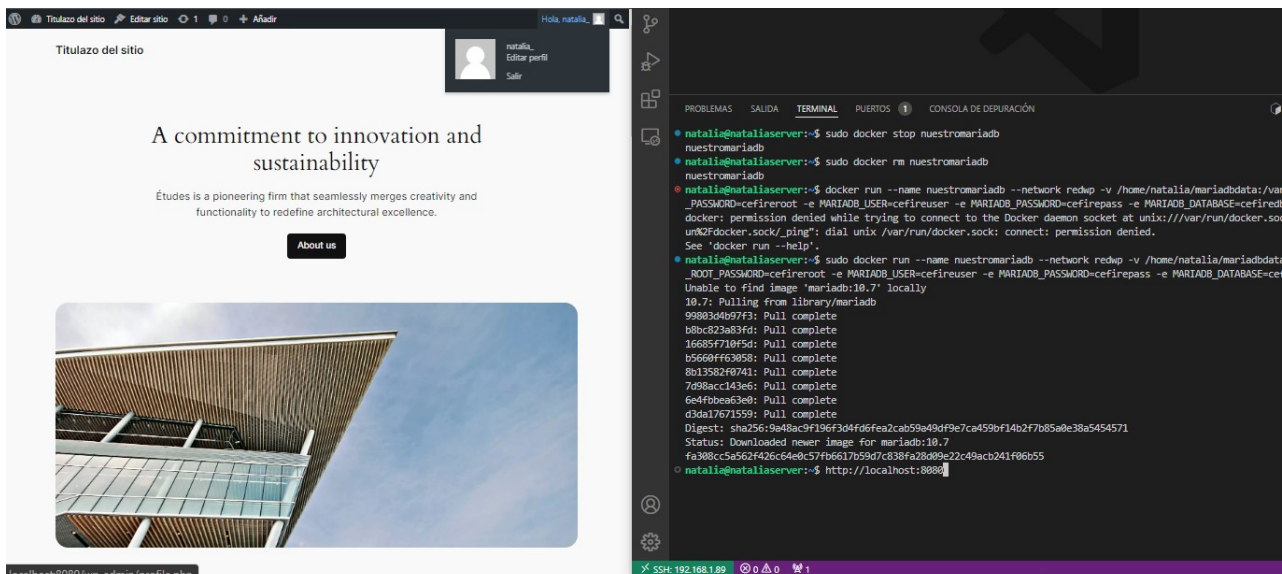


Figura 20: Cambio de versión del servicio MariaDB eliminándolo y creándolo otra vez

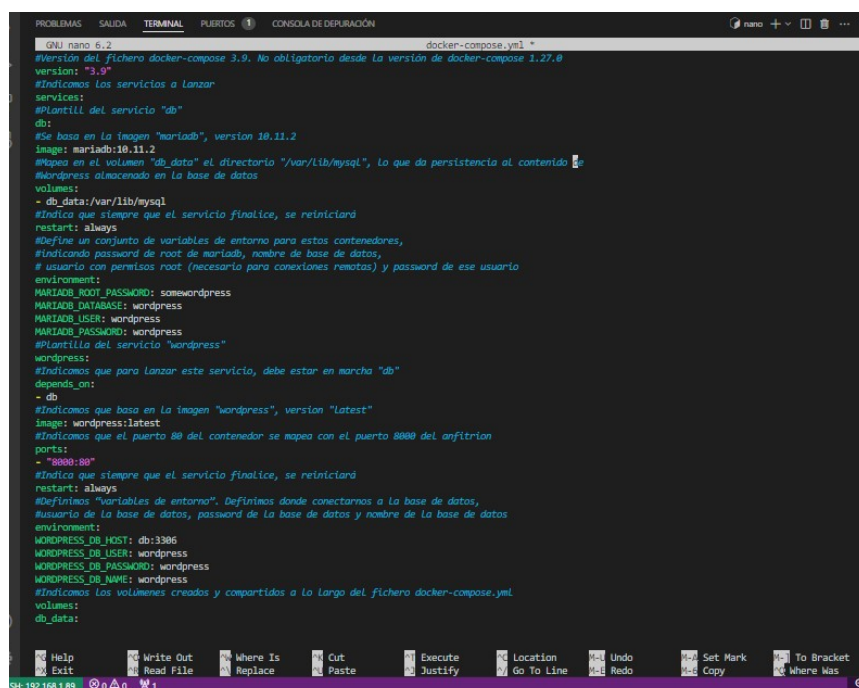


## Unidad 6

### Caso Práctico 1

En este caso práctico vamos a utilizar CMS Wordpress. Para ello usaremos un fichero “docker-compose.yml” comentado que nos pondrá en marcha dos contenedores: el primero utilizando “Apache + PHP” mientras que el segundo contendrá un servidor de bases de datos MariaDB. Esto es parecido a la práctica anterior pero utilizando un archivo en lugar de una red.

El fichero docker-compose.yml contendrá la información de los dos contenedores que se pondrán:(ten mucho en cuenta la tabulación, si no dará error. En esta imagen no se ve tabulado):



```
GNU nano 6.2 docker-compose.yml
#Versión del fichero docker-compose 3.9. No obligatorio desde la versión de docker-compose 1.27.0
version: "3.9"
#Indicamos los servicios a lanzar
services:
#Plantilla del servicio "db"
db:
#Se basa en la imagen "mariadb", version 10.11.2
image: mariadb:10.11.2
#Mapea en el volumen "db_data" el directorio "/var/lib/mysql", lo que da persistencia al contenido
#Wordpress almacenado en la base de datos
volumes:
- db_data:/var/lib/mysql
#Indica que siempre que el servicio finalice, se reiniciará
restart: always
#Define un conjunto de variables de entorno para estos contenedores,
#Indicando password de root de mariadb, nombre de base de datos,
# usuario con permisos root (necesario para conexiones remotas) y password de ese usuario
environment:
MARIADB_ROOT_PASSWORD: somewordpress
MARIADB_DATABASE: wordpress
MARIADB_USER: wordpress
MARIADB_PASSWORD: wordpress
#Plantilla del servicio "wordpress"
wordpress:
#Indicamos que para lanzar este servicio, debe estar en marcha "db"
depends_on:
- db
#Indicamos que basa en la imagen "wordpress", version "latest"
image: wordpress:latest
#Indicamos que el puerto 80 del contenedor se mapea con el puerto 8080 del anfitrión
ports:
- "8080:80"
#Indica que siempre que el servicio finalice, se reiniciará
restart: always
#Definimos "variables de entorno". Definimos donde conectarnos a la base de datos,
#usuario de la base de datos, password de la base de datos y nombre de la base de datos
environment:
WORDPRESS_DB_HOST: db:3306
WORDPRESS_DB_USER: wordpress
WORDPRESS_DB_PASSWORD: wordpress
WORDPRESS_DB_NAME: wordpress
#Indicamos los volúmenes creados y compartidos a lo largo del fichero docker-compose.yml
volumes:
db_data:
```

Figura 21: Configuración del fichero docker-compose.yml para conectar los dos contenedores a él

Para poder arrancarlo, simplemente nos situamos en el directorio donde tengamos el fichero “docker-compose.yml” y ponemos el comando necesario. Al ir al sitio local se observa la instalación para wordpress

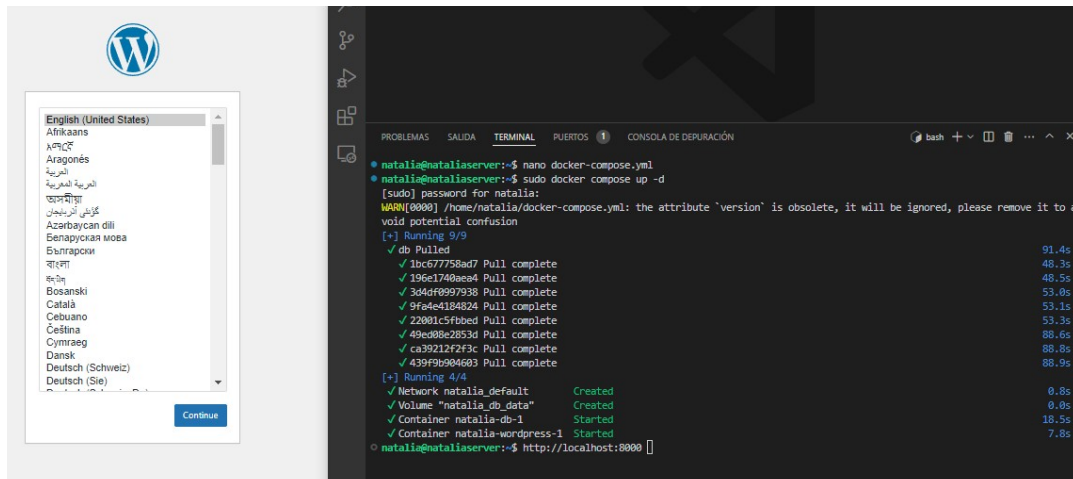


Figura 22: Comprobación al arrancar fichero docker-compose.yml

Al parar el programa se eliminarán los contenedores pero no sus imágenes, y así al lanzarlo aprovecha las imágenes ya creadas para acelerar el proceso.

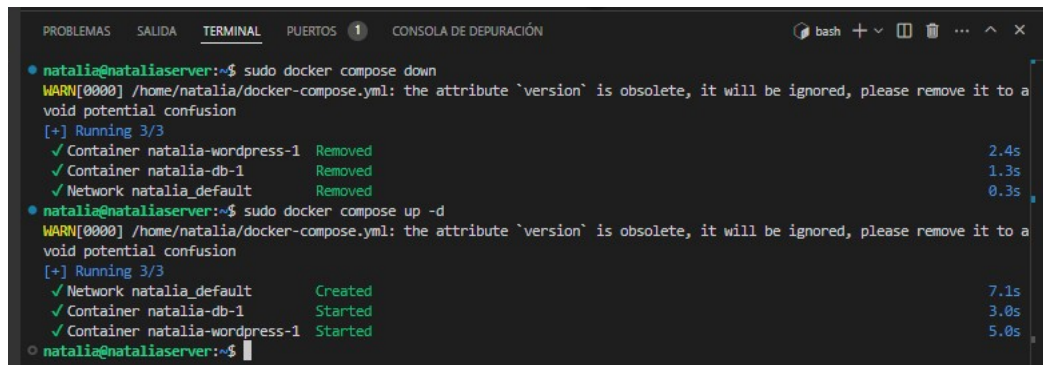
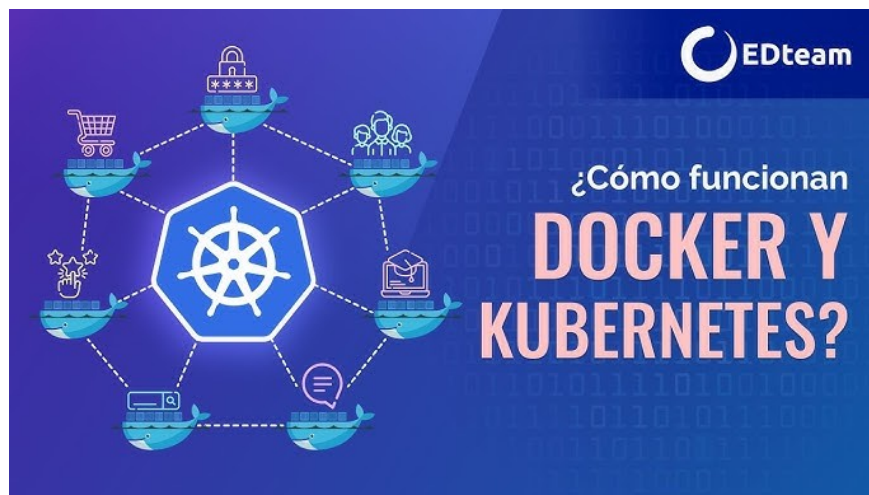


Figura 23: Parón del compose eliminado y volviendo a crear

### Actividad 3 – Kubernetes



En los contenedores hay fallos constantes y hay que arreglarlos cada vez que pasan. Kubernetes nos da portabilidad, reproducción y escalabilidad automática para llevar a cabo todo esto. Se automatizan los contenedores gracias a kubernetes maximiza la capacidad (distribuye los contenedores de una forma lógica y eficiente). Además de todo esto también es de código abierto para todo el mundo.

Kubernetes es como un sistema que ayuda a mantener todo ordenado y funcionando bien cuando tienes aplicaciones que se ejecutan en múltiples contenedores.

## **Problemas encontrados y sugerencias**

Problemas de conexión de la máquina con visual studio y de espacio(especificar que espacio necesita el disco duro)

## **Conclusión**

Buena actividad para aprender lo principal de los docker.

## **Bibliografía/webgrafía**

No he utilizado ningún medio de información aparte del pdf ofrecido por el profesor y preguntas al profesor.