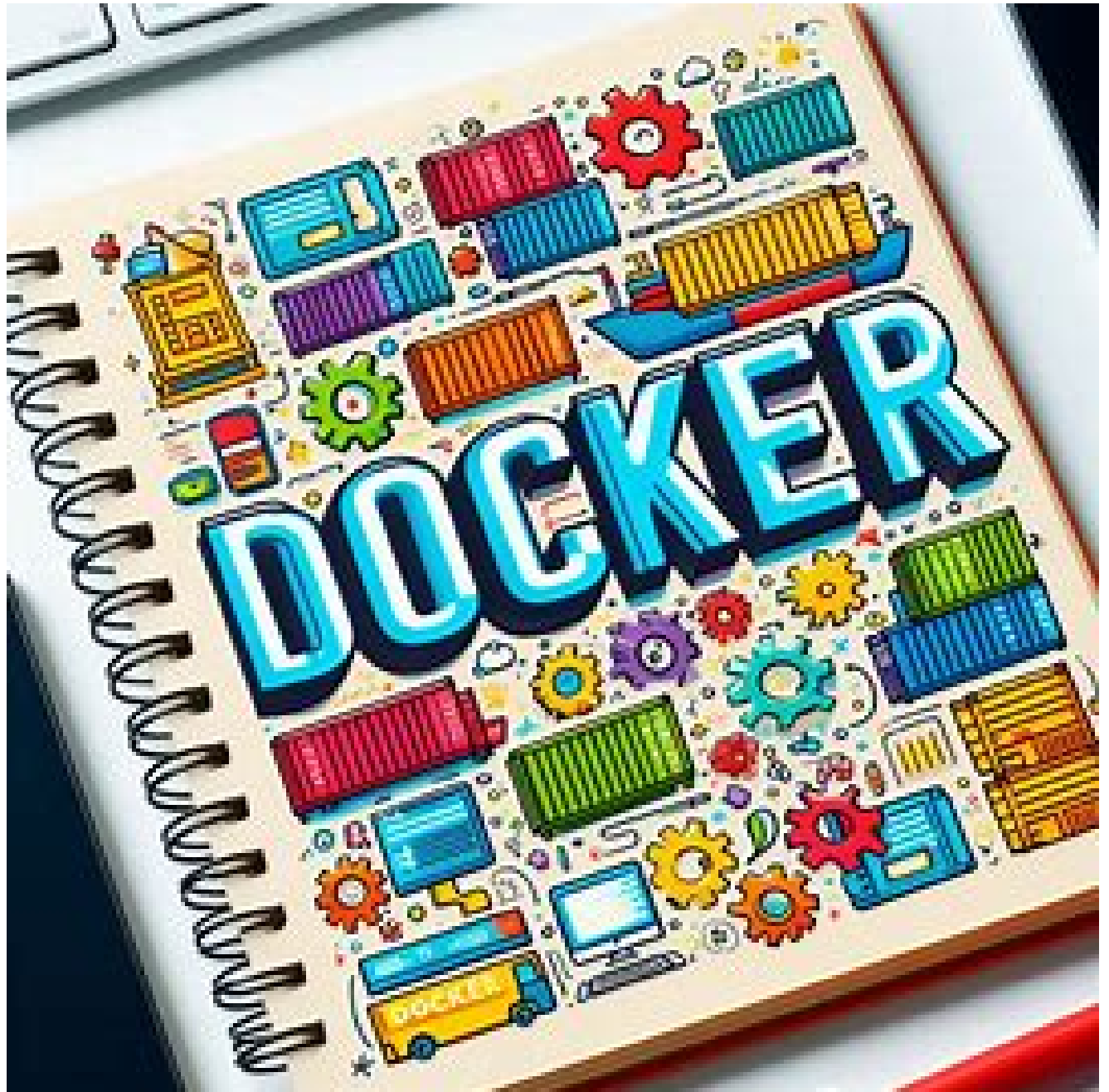


DOCKER



Índice:

Introducción.....	pag 2
Objetivos.....	pag 2
Material empleado.....	pag 2
Parte 1.....	pag 3
Parte 2.....	pag 4
Parte 3.....	pag 5
Parte 4.....	pag 10
Parte 5.....	pag 12
Parte 6.....	pag 15
Parte 7.....	pag 16
Problemas encontrados.....	pag 20
Webgrafía.....	pag 21

➤ *Introducción:*

En esta práctica vamos a ver todo lo relacionado a docker o contenedores, desde como crear uno totalmente “inservible” a como crear uno que tenga wordpress, php...etc

➤ *Objetivos:*

Aprender que es un docker, como crearlo, como agregarle utilidades y familiarizarnos con el.

➤ *Material empleado para realizar la práctica*

<i>COMPONENTS DEL TEU EQUIP</i>	<i>VALOR</i>
Marca y model del Processador	2 × Intel® Pentium® CPU G4400 @ 3.30GHz
Capacitat de Memòria RAM	Memoria: 7,7 GiB de RAM
Tipus de dispositiu d'emmagatzematge i capacitat (GiB)	8.191.471.616 bytes =7,63GiB

parte 1:

1. Empezaremos haciendo un update para actualizar la lista de paquetes
sudo apt update

```
uwu@uwu-virtualbox:~$ sudo apt update
[sudo] contraseña para uwu:
Obj:1 http://security.ubuntu.com/ubuntu focal-security InRelease
Obj:2 http://archive.ubuntu.com/ubuntu focal InRelease
Obj:3 http://archive.ubuntu.com/ubuntu focal-updates InRelease
Obj:4 http://archive.ubuntu.com/ubuntu focal-backports InRelease
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se pueden actualizar 335 paquetes. Ejecute «apt list --upgradable» para verlo
```

2. A continuación, instale algunos paquetes de requisitos previos que permitan a apt usar paquetes a través de HTTPS:

sudo apt install apt-transport-https ca-certificates curl software-properties-common

```
uwu@uwu-virtualbox:~$ sudo apt install apt-transport-https ca-certificates cu
rl software-properties-common
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
  libcurl4 python3-software-properties software-properties-qt
Se instalarán los siguientes paquetes NUEVOS:
  apt-transport-https curl
Se actualizarán los siguientes paquetes:
  ca-certificates libcurl4 python3-software-properties
  software-properties-common software-properties-qt
5 actualizados, 2 nuevos se instalarán, 0 para eliminar y 330 no actualizados
```

3. Luego, añada la clave de GPG para el repositorio oficial de Docker en su sistema:
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

```
uwu@uwu-virtualbox:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo
apt-key add -
OK
```

4. Agregamos el repositorio de Docker a las fuentes de APT con el siguiente comando:
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable"

```
uwu@uwu-virtualbox:~$ sudo add-apt-repository "deb [arch=amd64] https://download.dock
er.com/linux/ubuntu focal stable"
Des:1 https://download.docker.com/linux/ubuntu focal InRelease [57,7 kB]
Obj:2 http://security.ubuntu.com/ubuntu focal-security InRelease
Des:3 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages [43,6 kB]
Obj:4 http://archive.ubuntu.com/ubuntu focal InRelease
Des:5 http://archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Obj:6 http://archive.ubuntu.com/ubuntu focal-backports InRelease
Descargados 215 kB en 1s (269 kB/s)
Leyendo lista de paquetes... Hecho
```

5. Asegúrese de estar a punto de realizar la instalación desde el repositorio de Docker en lugar del repositorio predeterminado de Ubuntu:

apt-cache policy docker-ce

```
uwu@uwu-virtualbox:~$ apt-cache policy docker-ce
docker-ce:
  Instalados: (ninguno)
  Candidato: 5:26.1.3-1-ubuntu.20.04-focal
  Tabla de versión:
  5:26.1.3-1-ubuntu.20.04-focal 500
    500 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
  5:26.1.2-1-ubuntu.20.04-focal 500
    500 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
  5:26.1.1-1-ubuntu.20.04-focal 500
    500 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
  5:26.1.0-1-ubuntu.20.04-focal 500
    500 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
  5:26.0.2-1-ubuntu.20.04-focal 500
    500 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
  5:26.0.1-1-ubuntu.20.04-focal 500
    500 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
  5:26.0.0-1-ubuntu.20.04-focal 500
    500 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
  5:25.0.5-1-ubuntu.20.04-focal 500
    500 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
  5:25.0.4-1-ubuntu.20.04-focal 500
```

6. Por último, instale Docker:
sudo apt install docker-ce

```
uwu@uwu-virtualbox:~$ sudo apt install docker-ce
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
  containerd.io docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras
  docker-compose-plugin pigz slirp4netns
Paquetes sugeridos:
  aufs-tools cgroupfs-mount | cgroup-lite
Se instalarán los siguientes paquetes NUEVOS:
  containerd.io docker-buildx-plugin docker-ce docker-ce-cli
  docker-ce-rootless-extras docker-compose-plugin pigz slirp4netns
0 actualizados, 8 nuevos se instalarán, 0 para eliminar y 330 no actualizados.
Se necesita descargar 121 MB de archivos.
Se utilizarán 434 MB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n] s
Des:1 https://download.docker.com/linux/ubuntu focal/stable amd64 containerd.io amd64
  1.6.32-1 [30,0 MB]
Des:2 http://archive.ubuntu.com/ubuntu focal/universe amd64 pigz amd64 2.4-1 [57,4 kB]
Des:3 https://download.docker.com/linux/ubuntu focal/stable amd64 docker-buildx-plugi
  n amd64 0.14.0-1-ubuntu.20.04-focal [29,7 MB]
```

- Con esto, Docker quedará instalado, el demonio se iniciará y el proceso se habilitará para ejecutarse en el inicio. Compruebe que funcione:

sudo systemctl status docker

```
uwu@uwu-virtualbox:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: ena
   Active: active (running) since Sat 2024-05-25 12:10:32 CEST; 32s ago
   TriggeredBy: ● docker.socket
   Docs: https://docs.docker.com
   Main PID: 9238 (dockerd)
   Tasks: 7
   Memory: 28.1M
   CGroup: /system.slice/docker.service
           └─9238 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd

may 25 12:10:31 uwu-virtualbox systemd[1]: Starting Docker Application Container Eng
may 25 12:10:31 uwu-virtualbox dockerd[9238]: time="2024-05-25T12:10:31.88255591+02
may 25 12:10:31 uwu-virtualbox dockerd[9238]: time="2024-05-25T12:10:31.883187689+02
may 25 12:10:31 uwu-virtualbox dockerd[9238]: time="2024-05-25T12:10:31.955520064+02
may 25 12:10:32 uwu-virtualbox dockerd[9238]: time="2024-05-25T12:10:32.131126681+02
may 25 12:10:32 uwu-virtualbox dockerd[9238]: time="2024-05-25T12:10:32.178875316+02
may 25 12:10:32 uwu-virtualbox dockerd[9238]: time="2024-05-25T12:10:32.178988096+02
may 25 12:10:32 uwu-virtualbox dockerd[9238]: time="2024-05-25T12:10:32.235723789+02
may 25 12:10:32 uwu-virtualbox systemd[1]: Started Docker Application Container Eng
```

- Finalmente agregue su nombre de usuario al grupo docker:

sudo usermod -aG docker \${USER}

su - \${USER}

```
albertpopa@albertpopa-virtualbox:~$ sudo usermod -aG docker ${USER}
albertpopa@albertpopa-virtualbox:~$ su - ${USER}
Contraseña:
albertpopa@albertpopa-virtualbox:~$ id -nG
albertpopa adm cdrom sudo dip plugdev lpadmin docker sambashare
```

parte 2:

- Instalaremos la imagen hello-world de docker, para ello haremos un **docker run hello-world**

```
uwu@uwu-virtualbox:~$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:266b191e926f65542fa8daec01a192c4d292bff79426f4730a046e1bc576
fd
Status: Downloaded newer image for hello-world:latest
```

- Para ver si se ha instalado haremos un **docker images**

```
uwu@uwu-virtualbox:~$ docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
hello-world latest d2c94e258dcb 13 months ago 13.3kB
uwu@uwu-virtualbox:~$
```

parte 3:

1. Crearemos un contenedor con la imagen base "ubuntu", al que llamaremos LAMP y en el que expondremos su puerto 80 en el puerto 8080 de nuestro sistema. Además, al crearlo, dejaremos lista una "shell" para instalar los programas necesarios.

```
root@1c841954c3e5: /
albertpopa@albertpopa-virtualbox:~$ docker run -it -p 8080:80 --name LAMP ubuntu /bin/bash
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
49b384cc7b4a: Pull complete
Digest: sha256:3f85b7caad41a95462cf5b787d8a04604c8262cdcdf9a472b8c52ef83375fe15
Status: Downloaded newer image for ubuntu:latest
root@1c841954c3e5: /#
```

2. A continuación haremos un update.

```
root@1c841954c3e5: /# apt update
Ign:1 http://archive.ubuntu.com/ubuntu noble InRelease
Ign:2 http://archive.ubuntu.com/ubuntu noble-updates InRelease
Ign:3 http://archive.ubuntu.com/ubuntu noble-backports InRelease
Err:4 http://archive.ubuntu.com/ubuntu noble Release
404 Not Found [IP: 10.239.3.13 80]
Err:5 http://archive.ubuntu.com/ubuntu noble-updates Release
404 Not Found [IP: 10.239.3.13 80]
Err:6 http://archive.ubuntu.com/ubuntu noble-backports Release
404 Not Found [IP: 10.239.3.13 80]
Get:7 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:8 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [103 kB]
Get:9 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Package
```

3. Después de esto, instalamos los paquetes necesarios para instalar LAMP + WordPress

```
root@c9369bc20583: /# apt install wordpress php libapache2-mod-php mysql-server php-mysql
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  adduser apache2 apache2-bin apache2-data apache2-utils ca-certificates
  default-mysql-client fontconfig-config fonts-dejavu-core
  fonts-dejavu-mono javascript-common krb5-locales libaio1t64 libao-common
```

Seleccionamos los datos geográficos, que nos solicita para el tema de la configuración.

```
Please select the geographic area in which you live. Subsequent
configuration questions will narrow this down by presenting a list of
cities, representing the time zones in which they are located.

 1. Africa      4. Arctic      7. Australia   10. Pacific
 2. America     5. Asia        8. Europe      11. Etc
 3. Antarctica  6. Atlantic    9. Indian

Geographic area:
Geographic area: 1^H^H^H
Geographic area: 8

Please select the city or region corresponding to your time zone.

 1. Amsterdam   17. Guernsey   33. Monaco     49. Stockholm
 2. Andorra     18. Helsinki  34. Moscow     50. Tallinn
 3. Astrakhan   19. Isle_of_Man 35. Nicosia    51. Tirane
 4. Athens     20. Istanbul   36. Oslo       52. Tiraspol
 5. Belfast    21. Jersey     37. Paris      53. Ulyanovsk
 6. Belgrade   22. Kaliningrad 38. Podgorica  54. Vaduz
 7. Berlin     23. Kirov      39. Prague     55. Vatican
```

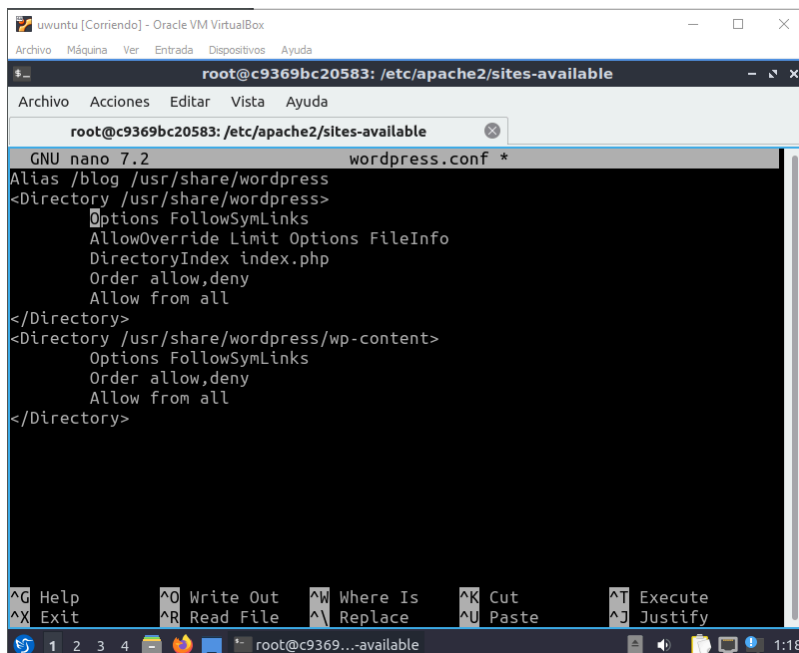
4. Después de esto, podemos lanzar el servicio Apache con el comando:

```
root@c9369bc20583: /# service apache2 start
* Starting 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
*
root@c9369bc20583: /#
```

5. Primero de todo, podemos instalar un editor de texto en modo consola que sea de vuestro gusto. Por ejemplo, con este comando podemos instalar "nano":

```
root@c9369bc20583:/# apt install nano
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Suggested packages:
  hunspell
The following NEW packages will be installed:
```

6. Una vez instalado vuestro editor favorito, procedemos a editar la configuración de Apache para trabajar con WordPress. Crearemos el archivo de configuración del sitio en Apache "/etc/apache2/sites-available/wordpress.conf" que configurará el acceso al sitio WordPress. Este archivo debe llevar el siguiente contenido



```
root@c9369bc20583: /etc/apache2/sites-available
GNU nano 7.2 wordpress.conf *
Alias /blog /usr/share/wordpress
<Directory /usr/share/wordpress>
    Options FollowSymLinks
    AllowOverride Limit Options FileInfo
    DirectoryIndex index.php
    Order allow,deny
    Allow from all
</Directory>
<Directory /usr/share/wordpress/wp-content>
    Options FollowSymLinks
    Order allow,deny
    Allow from all
</Directory>
```

7. Creado el archivo, deberemos cargar el sitio, habilitar "URL rewriting" y recargar Apache con los siguientes comandos:

```
root@c9369bc20583:/etc/apache2/sites-available# a2ensite wordpress
Enabling site wordpress.
To activate the new configuration, you need to run:
    service apache2 reload
root@c9369bc20583:/etc/apache2/sites-available# a2enmod rewrite
Enabling module rewrite.
To activate the new configuration, you need to run:
    service apache2 restart
root@c9369bc20583:/etc/apache2/sites-available# service apache2 reload
* Reloading Apache httpd web server apache2
*
root@c9369bc20583:/etc/apache2/sites-available#
```

8. Una vez todo instalado, debemos preparar el servicio MySQL. En primer lugar, deberemos poner en marcha el servicio con el comando:

```
root@c9369bc20583:/etc/apache2/sites-available# service mysql start
* Starting MySQL database server mysqld
su: warning: cannot change directory to /nonexistent: No such file or directory
[ OK ]
root@c9369bc20583:/etc/apache2/sites-available#
```

9. Después de esto, deberemos ejecutar el comando para generar una contraseña de root de MySQL Server de forma segura (deberemos recordarla) y otras opciones. El comando a ejecutar es:

```
root@c9369bc20583:/etc/apache2/sites-available# mysql_secure_installation

Securing the MySQL server deployment.

Connecting to MySQL using a blank password.

VALIDATE PASSWORD COMPONENT can be used to test passwords
and improve security. It checks the strength of password
and allows the users to set only those passwords which are
secure enough. Would you like to setup VALIDATE PASSWORD component?

Press y|Y for Yes, any other key for No: █
```

10. Después de esto, accederemos a la base de datos con el cliente MySQL de la siguiente manera:

```
root@c9369bc20583:/etc/apache2/sites-available# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 12
Server version: 8.0.36-2ubuntu3 (Ubuntu)

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

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

mysql> █
```

11. Después de indicar la contraseña de "root", podremos escribir órdenes para MySQL. Escribiremos las siguientes comandos. En primer lugar, vamos a crear la base de datos "wordpress".

```
mysql> CREATE DATABASE wordpress;
Query OK, 1 row affected (0.01 sec)
```

12. Después de eso, creamos el usuario "wordpress" (con contraseña "MiPass-2021") y le otorgamos permisos totales a la base de datos "wordpress".

```
mysql> CREATE USER 'wordpress'@'%' IDENTIFIED BY 'MiPass-2021';
Query OK, 0 rows affected (0.01 sec)

mysql> GRANT ALL PRIVILEGES ON wordpress.* TO 'wordpress'@'%' WITH GRANT
-> OPTION;
Query OK, 0 rows affected (0.00 sec)

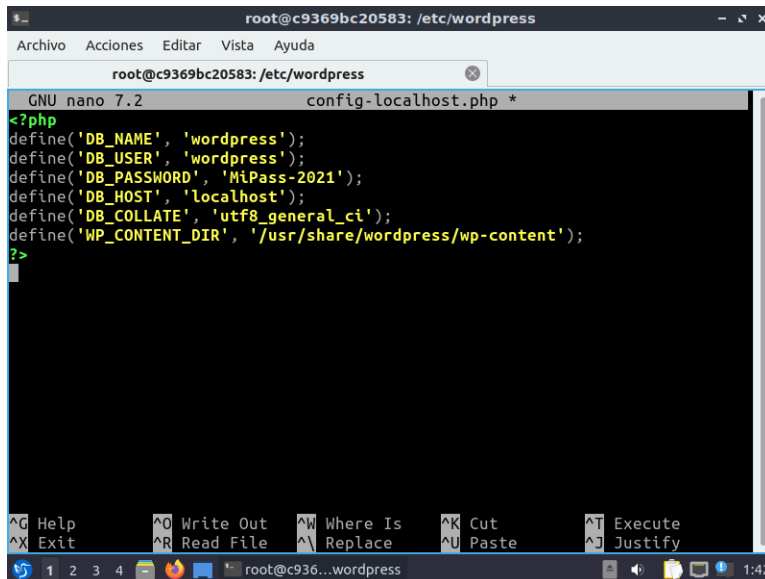
mysql> █
```

13. Finalmente, propagamos los privilegios establecidos para que estén operativos en el servidor.

```
mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)

mysql> █
```

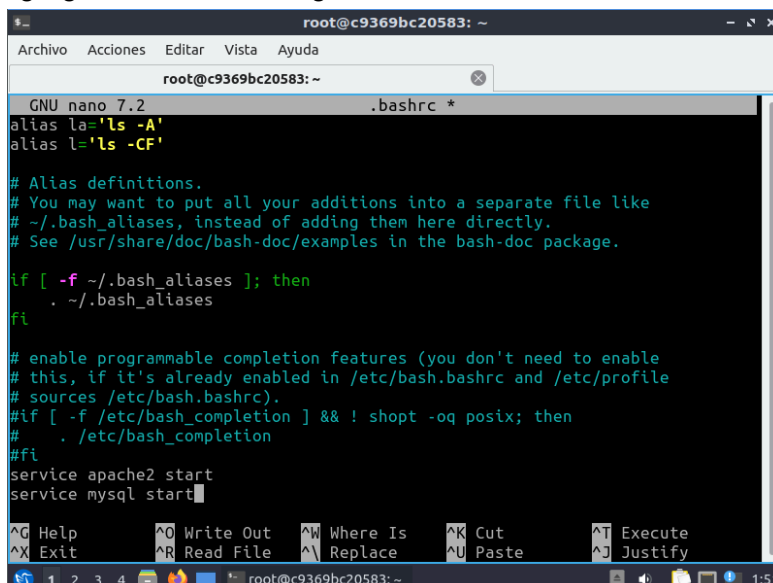
14. Usando un editor de texto de consola, editaremos el archivo de configuración de Wordpress `/etc/wordpress/config-localhost.php`, quedando de la manera que ahora indicamos.



```
root@c9369bc20583: /etc/wordpress
GNU nano 7.2 config-localhost.php *
<?php
define('DB_NAME', 'wordpress');
define('DB_USER', 'wordpress');
define('DB_PASSWORD', 'MiPass-2021');
define('DB_HOST', 'localhost');
define('DB_COLLATE', 'utf8_general_ci');
define('WP_CONTENT_DIR', '/usr/share/wordpress/wp-content');
?>
```

15. Accedemos a la carpeta de nuestro usuario, `/root`, escribiendo `cd`, `cd ~` o `cd /root`.

Aquí, con un editor de texto, modificamos el archivo `.bashrc` y al final del mismo agregamos las líneas siguientes.



```
root@c9369bc20583: ~
GNU nano 7.2 .bashrc *
alias la='ls -A'
alias l='ls -CF'

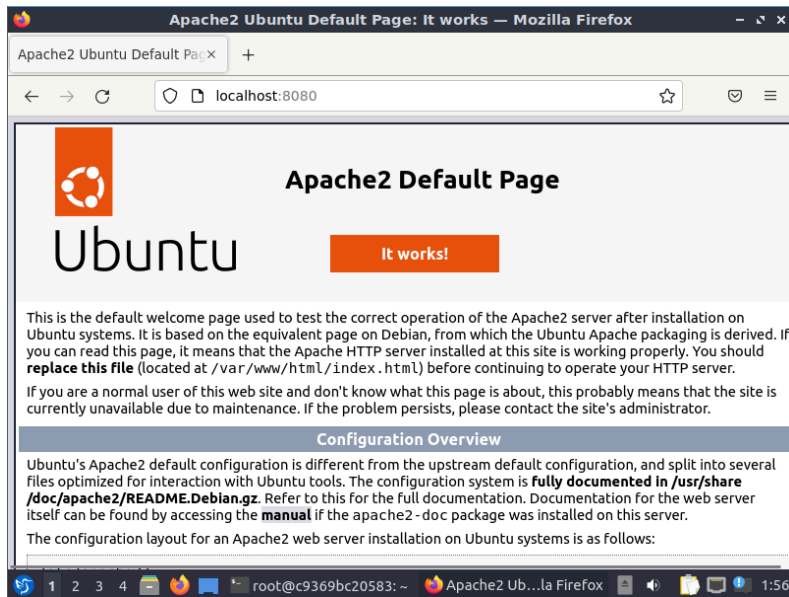
# Alias definitions.
# You may want to put all your additions into a separate file like
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi

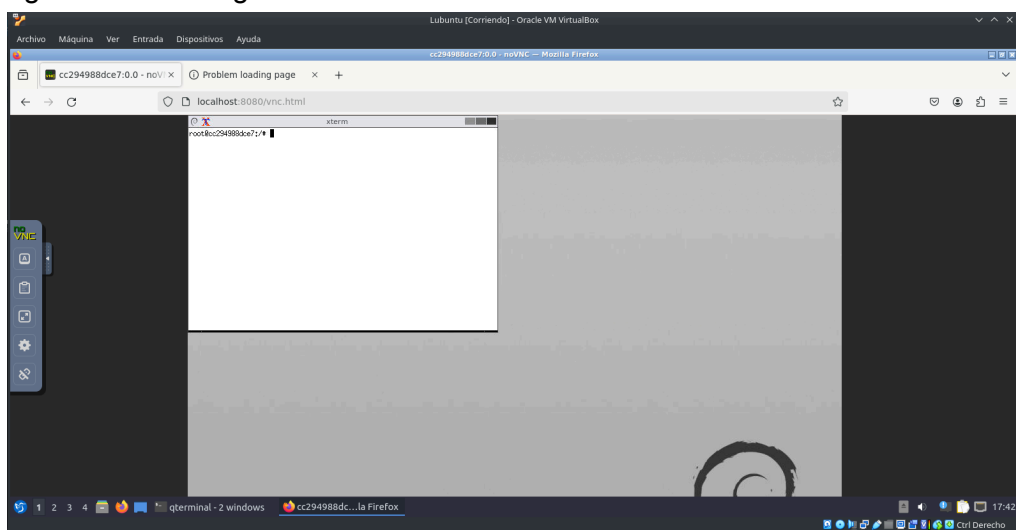
# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
#if [ -f /etc/bash_completion ] && ! shopt -oq posix; then
#    . /etc/bash_completion
#fi
service apache2 start
service mysql start
```

16. Finalmente, para comprobar que todo ha funcionado correctamente, intentaremos detener el contenedor, ponerlo en marcha nuevamente y verificar que podemos acceder a nuestro Wordpress en <http://localhost:8080/>. Para detener y lanzar el contenedor, utilizaremos los comandos:

docker stop LAMP
docker start LAMP

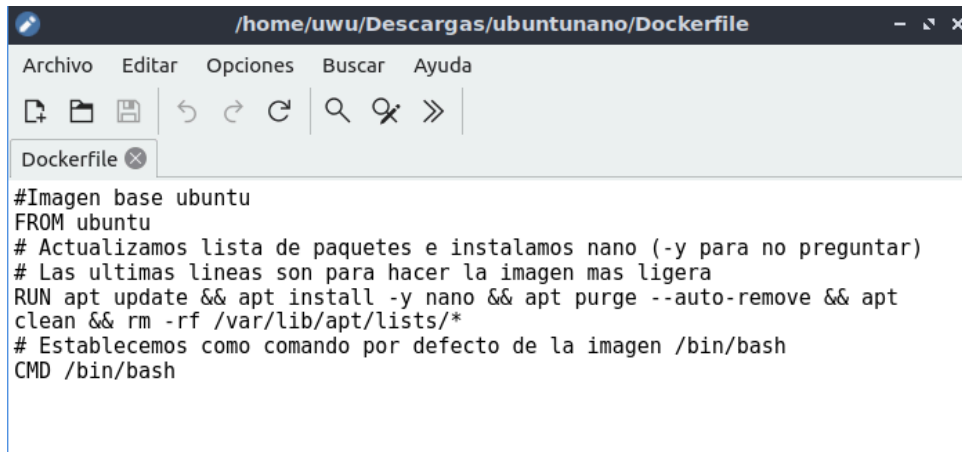


1. Hacemos un docker run --rm -it -p 8080:8080 theasp/novnc
2. Abrimos el navegador y buscamos la siguiente url <http://localhost:8080/vnc.html>
3. Una vez abierta, lo conectamos el cliente con el navegador y nos aparecerá la siguiente interfaz gráfica.



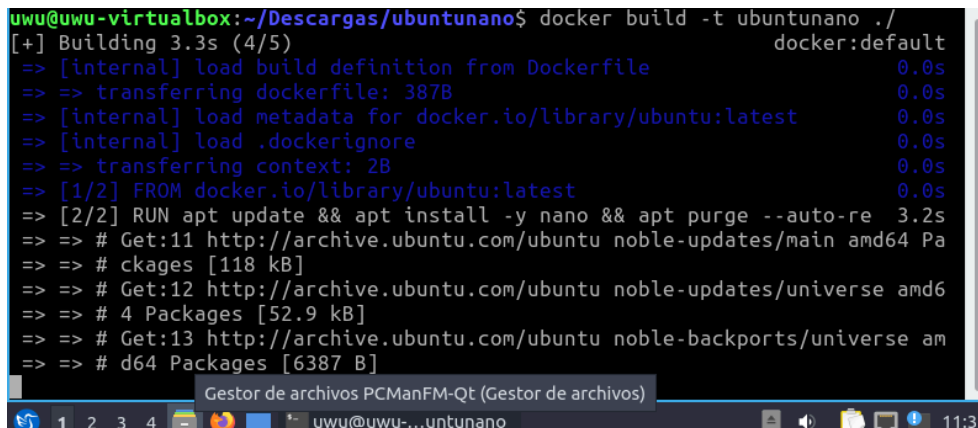
parte 4:

1. Creamos un fichero que se llame dockerfile y contenga las siguiente información:



```
/home/uwu/Descargas/ubuntuano/Dockerfile
Archivo  Editar  Opciones  Buscar  Ayuda
Dockerfile
#Imagen base ubuntu
FROM ubuntu
# Actualizamos lista de paquetes e instalamos nano (-y para no preguntar)
# Las ultimas lineas son para hacer la imagen mas ligera
RUN apt update && apt install -y nano && apt purge --auto-remove && apt
clean && rm -rf /var/lib/apt/lists/*
# Establecemos como comando por defecto de la imagen /bin/bash
CMD /bin/bash
```

2. El funcionamiento del mismo "Dockerfile" está definido por sus propios comentarios. Una vez preparado, crearemos la imagen con:



```
uwu@uwu-virtualbox:~/Descargas/ubuntuano$ docker build -t ubuntuano ./
[+] Building 3.3s (4/5)                                docker:default
=> [internal] load build definition from Dockerfile      0.0s
=> => transferring dockerfile: 387B                    0.0s
=> [internal] load metadata for docker.io/library/ubuntu:latest 0.0s
=> [internal] load .dockerignore                        0.0s
=> => transferring context: 2B                          0.0s
=> [1/2] FROM docker.io/library/ubuntu:latest           0.0s
=> [2/2] RUN apt update && apt install -y nano && apt purge --auto-re 3.2s
=> => # Get:11 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 Pa
=> => # ckages [118 kB]
=> => # Get:12 http://archive.ubuntu.com/ubuntu noble-updates/universe amd6
=> => # 4 Packages [52.9 kB]
=> => # Get:13 http://archive.ubuntu.com/ubuntu noble-backports/universe am
=> => # d64 Packages [6387 B]
```

3. Con el siguiente comando, podremos crear un contenedor con esta imagen, acceder a una shell dentro del contenedor y comprobar que el programa "nano" está instalado, ejecutando el editor con "nanoprueba.txt" o similar.



```
uwu@uwu-virtualbox:~/Descargas/ubuntuano$ docker run -it ubuntuano
root@15a75219596b:/#
```

4. Creamos un fichero que se llame dockerfile y contenga las siguiente información:

```
uwu@uwu-virtualbox: ~/Descar...07)/Part 04/sampledocker/app
GNU nano 4.8 Dockerfile
#Imagen base
FROM node:12-alpine
#Directorio de trabajo. Recordamos, que si no existe, se crea
WORKDIR /app
#Copiamos la carpeta actual en la carpeta del directorio de trabajo de la im
COPY . .
#Instalamos los paquetes necesarios en produccion para la APP0
RUN yarn install --production
#Comando para al arrancar el contenedor, lanzar la aplicaci3n
CMD ["node", "src/index.js"]
```

- El funcionamiento del mismo "Dockerfile" est3 definido por sus propios comentarios. Una vez preparado, crearemos la imagen con:

```
uwu@uwu-virtualbox:~/Descargas/Documentaci3n Introducci3n a Docker (Part 01 a 0
7)/Part 04/sampledocker/app$ docker build -t sampledocker ./
[+] Building 2.5s (4/8) docker:default
=> [internal] load build definition from Dockerfile 0.0s
=> => transferring dockerfile: 422B 0.0s
=> [internal] load metadata for docker.io/library/node:12-alpine 1.5s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [1/4] FROM docker.io/library/node:12-alpine@sha256:d4b15b3d48f4205 0.9s
=> => resolve docker.io/library/node:12-alpine@sha256:d4b15b3d48f4205 0.0s
=> => sha256:d4b15b3d48f42059a15bd659be60afe21762aae9 1.43kB / 1.43kB 0.0s
=> => sha256:4517380049fc3c9aacceae7764fcf3500354b0ac 1.16kB / 1.16kB 0.0s
=> => sha256:bb6d28039b8cec9aa8d9032f9aa640a792a60c2c 6.58kB / 6.58kB 0.0s
=> => sha256:df9b9388f04ad6279a7410b85cedfdcb2208c0a0 2.81MB / 2.81MB 0.2s
=> => sha256:3bf6d738020517f4622814e8c21db4b4aaa78a 24.91MB / 24.91MB 0.7s
=> => sha256:7939e601ee5e4737cf7fdb6d1dfe31ca4c269710 2.36MB / 2.36MB 0.6s
=> => extracting sha256:df9b9388f04ad6279a7410b85cedfdcb2208c0a003da7 0.2s
=> => sha256:31f0fb9de071269230cb0f786012ae4e81d26e489b1f 451B / 451B 0.5s
=> => extracting sha256:3bf6d738020517f4622814e8c21db4b4aaa78ae7cab4e 0.2s
=> [internal] load build context 0.1s
=> => transferring context: 4.63MB 0.1s
```

- Con el siguiente comando, podremos lanzar la aplicaci3n en el puerto 3000 de nuestra m3quina:

```
uwu@uwu-virtualbox:~/Descargas/Documentaci3n Introducci3n a Docker (Part 01 a 0
7)/Part 04/sampledocker/app$ docker run -dp 3000:3000 sampledocker
e7c1727d5717bf8f5d3685c89f9e26debad1199863b86e7e9f7b60d628802cdd
uwu@uwu-virtualbox:~/Descargas/Documentaci3n Introducci3n a Docker (Part 01 a 0
7)/Part 04/sampledocker/app$
```

- En este caso pr3ctico, crearemos una imagen que, partiendo de la imagen base "alpine", una de las im3genes m3s reconocidas como "imagen ligera", le instalaremos Apache 2, PHP y pondremos un simple `<?php phpinfo(); ?>` obtenido desde un archivo de Internet. Este archivo lo hemos obtenido de la URL <https://gist.github.com/SyntaxC4/5648247>

```
uwu@uwu-virtualbox: ~/Descarg...01 a 07)/Part 04/alpineapache
GNU nano 4.8 Dockerfile
FROM alpine
MAINTAINER Sergi <sergi.profesor@gmail.com>
#Actualizamos e instalamos paquetes con APK para Alpine
RUN apk update && apk add apache2 php php-apache2 openrc tar
#Copiamos script para lanzar Apache 2
ADD ./start.sh /start.sh
#Descargamos un ejemplo de <?php phpinfo(); ?> por enseñar como bajar algo d
#Podria haber sido simplemente
#RUN echo "<?php phpinfo(); ?>" > /var/www/localhost/htdocs/index.php
ADD https://gist.githubusercontent.com/SyntaxC4/5648247/raw/94277156638f9c30b
# Si quisieramos algo como Wordpress haríamos
#ADD http://wordpress.org/latest.tar.gz /var/www/localhost/htdocs/wordpress.
#RUN tar xvzf /var/www/localhost/htdocs/wordpress.tar.gz && rm -rf /var/www/
# Usamos usuario y grupo www-data. El grupo lo crea Apache, pero si quisiera
# Grupo www-data RUN set -x && addgroup -g 82 -S www-data
# Creamos usuario www-data y lo añadimos a ese grupo
RUN adduser -u 82 -D -S -G www-data www-data
[ 28 líneas leídas ]
^G Ver ayuda ^O Guardar ^W Buscar ^K Cortar Texto ^J Justificar
^X Salir ^R Leer fich. ^\ Reemplazar ^U Pegar ^T Ortografía
1 2 3 4 uwu@uwu-...neapache 12:15
```

- El funcionamiento del propio "Dockerfile" está definido por sus propios comentarios. Una vez preparado, crearemos la imagen con:

```
uwu@uwu-virtualbox:~/Descargas/Documentació Introducció a Docker (Part 01 a 07)/Part 04/alpineapache$ docker build -t alpineapache ./
[+] Building 2.2s (6/11) docker:default
=> [internal] load build definition from Dockerfile 0.0s
=> => transferring dockerfile: 1.44kB 0.0s
=> [internal] load metadata for docker.io/library/alpine:latest 1.4s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [1/6] FROM docker.io/library/alpine:latest@sha256:77726ef6b57ddf65 0.4s
=> => resolve docker.io/library/alpine:latest@sha256:77726ef6b57ddf65 0.0s
=> => sha256:77726ef6b57ddf65b551896826ec38bc3e53f75 1.85kB / 1.85kB 0.0s
=> => sha256:216266c86fc4dcef5619930bd394245824c2af52fd21 528B / 528B 0.0s
=> => sha256:1d34ffeaf190be23d3de5a8de0a436676b758f48 1.47kB / 1.47kB 0.0s
=> => sha256:d25f557d7f31bf7acf935859b5153da41d13c4 3.62MB / 3.62MB 0.3s
=> => extracting sha256:d25f557d7f31bf7acf935859b5153da41d13c41f2b4 0.1s
=> [4/6] ADD https://gist.githubusercontent.com/SyntaxC4/5648247/raw/ 0.3s
=> [internal] load build context 0.0s
=> => transferring context: 117B 0.0s
=> [2/6] RUN apk update && apk add apache2 php php-apache2 openrc tar 0.3s
=> => # fetch https://dl-cdn.alpinelinux.org/alpine/v3.20/main/x86_64/APKIN
=> => # DEX.tar.gz
=> => # fetch https://dl-cdn.alpinelinux.org/alpine/v3.20/community/x86_64/
=> => # APKINDEX.tar.gz
```

- Con el siguiente comando, podremos lanzar la aplicación en el puerto 80 de nuestra máquina:

```
uwu@uwu-virtualbox:~/Descargas/Documentació Introducció a Docker (Part 01 a 07)/Part 04/alpineapache$ docker run -dp 80:80 alpineapache
5188eacf5c4bcf26025f0ffaa10e1463055215d56143794dcd6005d1c45ab592
uwu@uwu-virtualbox:~/Descargas/Documentació Introducció a Docker (Part 01 a 07)/Part 04/alpineapache$
```

parte 5:

- El primer paso a dar en este caso práctico es crear la red que compartirán los contenedores que creemos, a la cual llamaremos "redwp". La podemos crear con el comando:

```
uwu@uwu-virtualbox:~$ docker network create redwp
c9a04aa72f50130c595572795be35acdf753039ecd8e13d4dad17da23a7a0fcc
uwu@uwu-virtualbox:~$
```

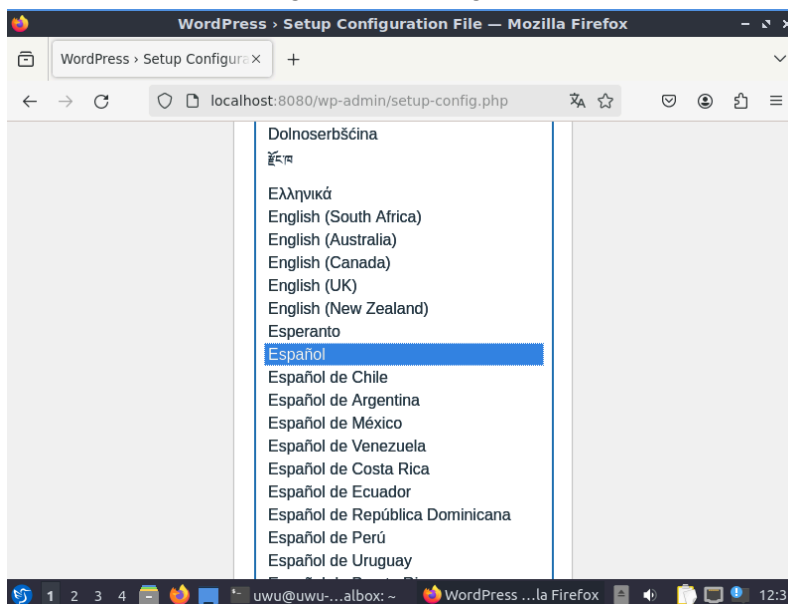
2. El primer paso será crear un contenedor con el servidor de bases de datos MySQL dentro de la red. Lo haremos utilizando la imagen https://hub.docker.com/_/mysql con el siguiente comando:

```
uwu@uwu-virtualbox:~$ docker run --name nuestromysql --network redwp -v /home/sergi/mysqldata:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=cefireeroot -e MYSQL_USER=cefireuser -e MYSQL_PASSWORD=cefirepass -e MYSQL_DATABASE=cefiredb -d mysql:5.6
Unable to find image 'mysql:5.6' locally
5.6: Pulling from library/mysql
35b2232c987e: Pull complete
fc55c00e48f2: Pull complete
0030405130e3: Pull complete
e1fef7f6a8d1: Pull complete
1c76272398bb: Pull complete
f57e698171b6: Pull complete
f5b825b269c0: Pull complete
dcb0af686073: Pull complete
27bbfeb886d1: Downloading 49.89MB/64.27MB
6f70cc868145: Download complete
1f6637f4600d: Download complete
```

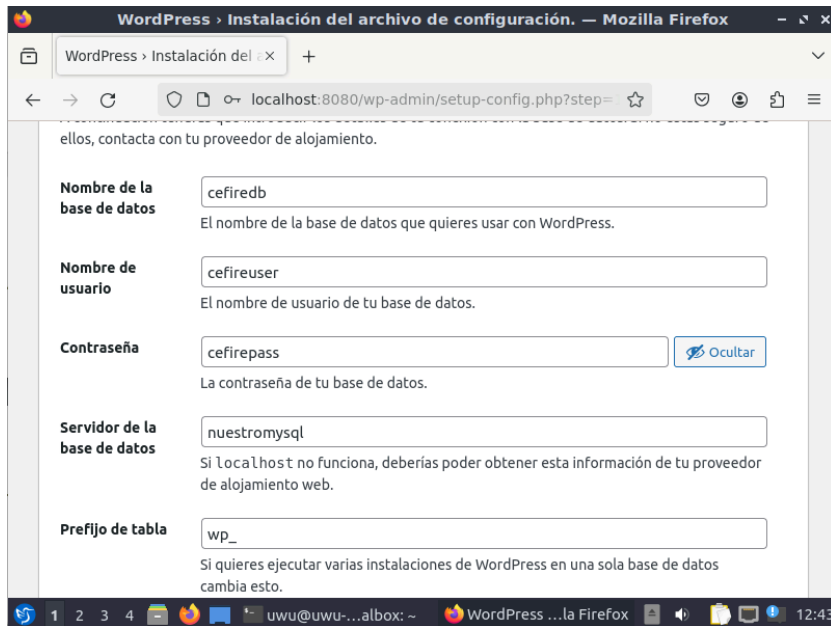
3. Ahora creamos un contenedor para Wordpress usando la imagen de Wordpress oficial:

```
uwu@uwu-virtualbox:~$ docker run --name nuestrowp --network redwp -p 8080:80 -d wordpress
Unable to find image 'wordpress:latest' locally
latest: Pulling from library/wordpress
09f376ebb190: Pull complete
76afcdc86551: Pull complete
ceed4541c527: Extracting 33.98MB/104.4MB
9ec84be954b0: Download complete
ff0e278869f9: Download complete
1693466e4cc6: Download complete
57c8d94a4882: Download complete
43af3fe8136a: Download complete
```

4. Accede a <http://localhost:8080> en tu navegador para iniciar la instalación de Wordpress. Usa las siguientes configuraciones:



5. Ahora debemos proceder con la instalación e indicar dónde se encuentra el sistema de bases de datos. Usaremos como nombre de anfitrión el nombre que se le dio al de MySQL. Como usuario y nombre de base de datos utilizaremos las variables usadas al crear el contenedor de MySQL. Quedará de manera similar a la siguiente imagen.



Para migrar de MySQL 5.6 a 5.7, sigue estos pasos:

1. Detén y elimina el contenedor nuestromysql:

```
uwu@uwu-virtualbox:~$ docker stop nuestromysql
nuestromysql
uwu@uwu-virtualbox:~$ docker rm nuestromysql
nuestromysql
uwu@uwu-virtualbox:~$
```

2. Crea un nuevo contenedor usando MySQL 5.7, mapeando los mismos datos:

```
uwu@uwu-virtualbox:~$ docker run --name nuestromysql --network redwp -v /home
/sergi/mysqldata:/var/lib/mysql -d mysql:5.7
Unable to find image 'mysql:5.7' locally
5.7: Pulling from library/mysql
20e4dcae4c69: Extracting 33.55MB/50.5MB
1c56c3d4ce74: Download complete
e9f03a1c24ce: Download complete
68c3898c2015: Download complete
6b95a940e7b6: Download complete
90986bb8de6e: Download complete
ae71319cb779: Downloading 21.56MB/25.53MB
ffc89e9dfd88: Download complete
43d05e938198: Waiting
064b2d298fba: Download complete
df9a4d85569b: Waiting
```

3. Finalmente volveremos a buscar <http://localhost:8080> para ver que Mysql funciona.

parte 6:

1. Creamos un fichero que se llame dockerfile y contenga las siguiente información

```
uwu@uwu-virtualbox: ~/Descar...t 06/CasoPractico1-Wordpress
GNU nano 4.8 docker-compose.yml
#Version del fichero docker-compose 3.9. No obligatorio desde la version de
version: "3.9"

#Indicamos los servicios a lanzar
services:
  #Plantill del servicio "db"
  db:
    #Se basa en la imagen "mysql", version 5.7
    image: mysql:5.7
    #Mapea en el volumen "db_data" el directorio "/var/lib/mysql", lo que da
    #Wordpress almacenado en la base de datos
    volumes:
      - db_data:/var/lib/mysql
    #Indica que siempre que el servicio finalice, se reiniciara
    restart: always
    #Define un conjunto de variables de entorno para estos contenedores,
    #indicando password de root de mysql, nombre de base de datos,
    # usuario con permisos root (necesario para conexiones remotas) y passwo
    environment:
      MYSQL_ROOT_PASSWORD: somewordpress
      MYSQL_DATABASE: wordpress
      MYSQL_USER: wordpress
      MYSQL_PASSWORD: wordpress

volumes:
  db_data:

45 líneas leídas
Ver ayuda  Guardar  Buscar  Cortar Texto  Justificar
Salir  Leer fich.  Reemplazar  Pegar  Ortografía
1 2 3 4 uwu@uwu-...ordpress 13:03
```

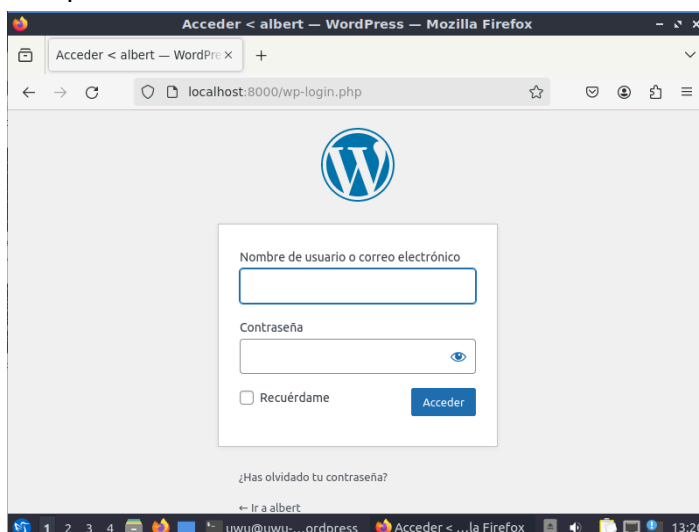
2. Para poner en marcha este sistema, simplemente nos situamos en el directorio donde tengamos el archivo "docker-compose.yml" de este caso práctico y escribimos:

```
uwu@uwu-virtualbox:~/Descargas/Documentació Introducció a Docker (Part 01 a 0
7)(1)/Part 06/CasoPractico1-Wordpress$ docker-compose up -d
Creating network "casopractico1-wordpress_default" with the default driver
Creating volume "casopractico1-wordpress_db_data" with default driver
Creating casopractico1-wordpress_db_1 ... done
Creating casopractico1-wordpress_wordpress_1 ... done
uwu@uwu-virtualbox:~/Descargas/Documentació Introducció a Docker (Part 01 a 0
7)(1)/Part 06/CasoPractico1-Wordpress$
```

*Hago un breve paréntesis y en caso de que nos de error de versión hay que ir al fichero anterior y cambiar la versión en vez de 3.9 poner 3.7
captura del error:*

```
ERROR: Version in "../docker-compose.yml" is unsupported. You might be seeing
this error because you're using the wrong Compose file version. Either specif
y a supported version (e.g. "2.2" or "3.3") and place your service definitions
under the 'services' key, or omit the 'version' key and place your service d
efinitions at the root of the file to use version 1.
For more on the Compose file format versions, see https://docs.docker.com/com
pose/compose-file/
```

3. Siguiendo con el caso práctico,accedemos a <http://localhost:8000> para ver que wordpress funciona



- Para reiniciar el sistema, simplemente ejecuta el siguiente comando:

```
uwu@uwu-virtualbox:~/Descargas/Documentació Introducció a Docker (Part 01 a 07)(1)/Part 06/CasoPractico1-Wordpress$ docker-compose down
Stopping casopractico1-wordpress_wordpress_1 ... done
Stopping casopractico1-wordpress_db_1 ... done
Removing casopractico1-wordpress_wordpress_1 ... done
Removing casopractico1-wordpress_db_1 ... done
Removing network casopractico1-wordpress_default
uwu@uwu-virtualbox:~/Descargas/Documentació Introducció a Docker (Part 01 a 07)(1)/Part 06/CasoPractico1-Wordpress$
```

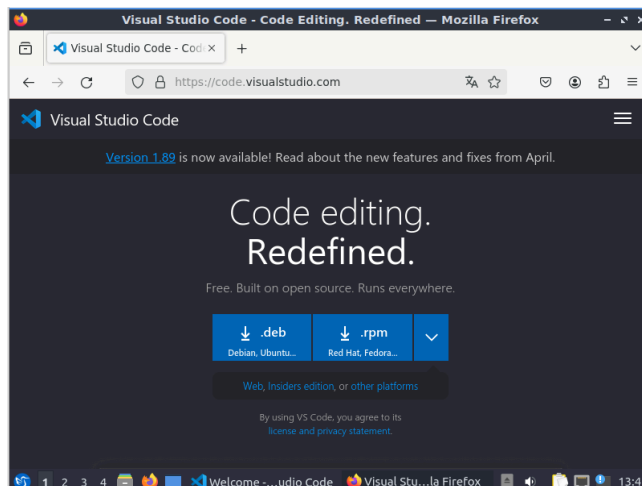
Este comando detiene y elimina los contenedores, pero no elimina las imágenes ni los volúmenes. Esto significa que los datos de la base de datos se mantendrán persistentes gracias al volumen mapeado.

- Para reiniciar el sistema, simplemente ejecuta el siguiente comando:

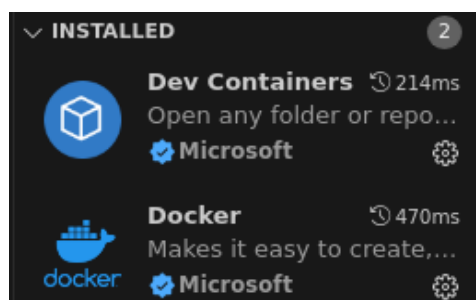
```
uwu@uwu-virtualbox:~/Descargas/Documentació Introducció a Docker (Part 01 a 07)(1)/Part 06/CasoPractico1-Wordpress$ docker-compose up -d
Creating network "casopractico1-wordpress_default" with the default driver
Creating casopractico1-wordpress_db_1 ... done
Creating casopractico1-wordpress_wordpress_1 ... done
uwu@uwu-virtualbox:~/Descargas/Documentació Introducció a Docker (Part 01 a 07)(1)/Part 06/CasoPractico1-Wordpress$
```

parte 7:

- Puedes descargar el editor "Visual Studio Code" en <https://code.visualstudio.com/> y si no lo conocemos y queremos saber más, podemos acceder a <https://code.visualstudio.com/learn>. Al instalarlo, si "Visual Studio Code" detecta que Docker está instalado en el sistema, el mismo editor nos sugerirá una serie de complementos. Estos complementos son los que se pueden ver en la imagen:



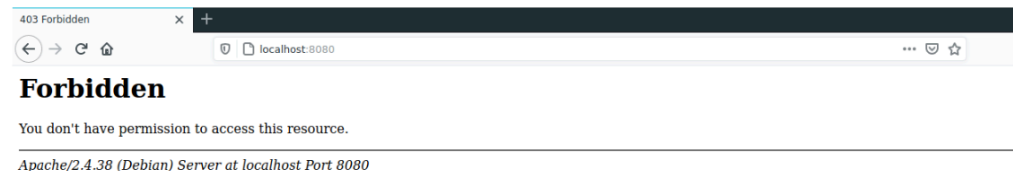
- Una vez realizada la instalación, abrimos visual studio code y nos vamos a las extensiones e instalaremos la extensión 'Docker' y 'Remote Containers' o 'Dev Containers'.



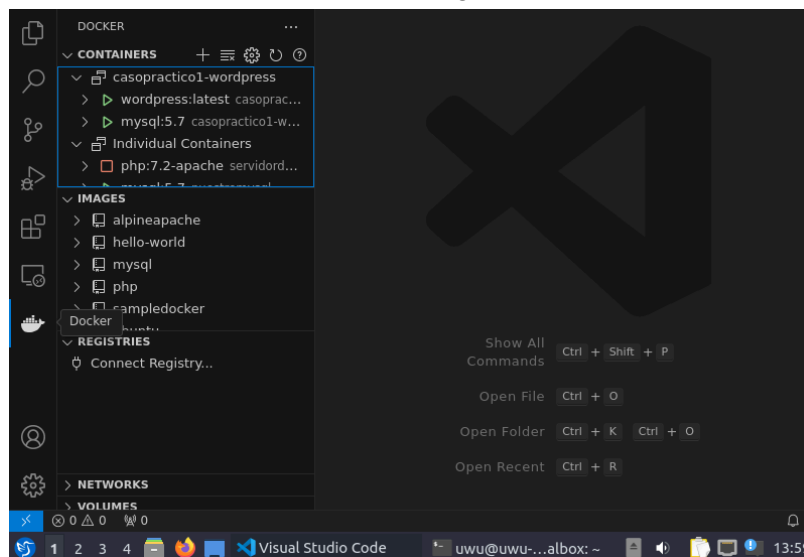
3. Con el siguiente comando, puedes crear un contenedor básico que servirá en el puerto 8080. El contenedor contendrá "Apache+PHP" y su sitio web estará almacenado en el directorio "/var/www/html":

```
uwu@uwu-virtualbox:~$ docker run -d --name servidordesarrollo -p 8080:80 php:7.2-apache
Unable to find image 'php:7.2-apache' locally
7.2-apache: Pulling from library/php
6ec7b7d162b2: Pull complete
db606474d60c: Pull complete
afb30f0cd8e0: Pull complete
3bb2e8051594: Pull complete
4c761b44e2cc: Pull complete
c2199db96575: Pull complete
1b9a9381eea8: Pull complete
fd07bbc59d34: Pull complete
72b73ab27698: Pull complete
983308f4f0d6: Pull complete
6e12f026e6da: Pull complete
```

4. Una vez creado nuestro contenedor, intentaremos acceder a <http://localhost:8080> y veremos algo similar a:



5. Después de eso, abriremos el editor Visual Studio Code y accederemos a las utilidades de Docker (icono de la ballena) y encontraremos un menú con varias opciones, tal como se ve en la imagen:



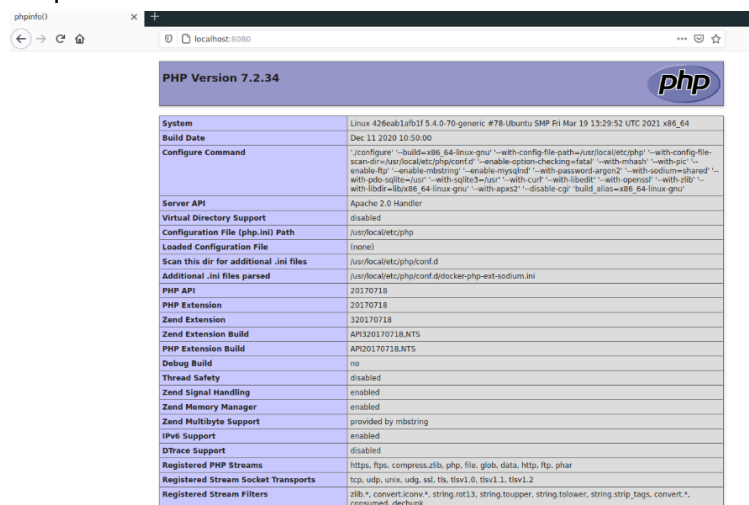
Si desplegamos la pestaña de menú "Containers", veremos todos los contenedores y podremos operar sobre ellos. En la siguiente imagen, vemos el contenedor que hemos creado anteriormente en funcionamiento.

6. Haremos clic derecho sobre el contenedor y seleccionaremos la opción "Attach Visual Studio Code". Esto abrirá una nueva instancia de Visual Studio Code conectada a este contenedor. Podemos confirmar que es correcto observando en la parte inferior de la pantalla.

- Una vez en esta instancia, podemos abrir el directorio `"/var/www/html"` y editarlo como si estuviéramos trabajando localmente. Aquí crearemos un simple archivo `"index.php"` con el siguiente contenido:

```
index.php
1  <?php
2      phpinfo();
3  <?>
```

- Si ahora accedemos a `http://localhost:8080`, observamos que todo funciona correctamente y que hemos podido desarrollar una aplicación dentro de un contenedor de forma gráfica y sin necesidad de mapear archivos en nuestra máquina anfitriona.



System	Linux 426eb1afb1f5 4.0-70-generic #78-Ubuntu SMP Fri Mar 19 13:29:52 UTC 2021 x86_64
Build Date	Dec 11 2020 10:50:00
Configure Command	./configure '--build=x86_64-linux-gnu' '--with-config-file-path=/usr/local/etc/php' '--with-config-file-scan-dir=/usr/local/etc/php/conf.d' '--enable-opcache-checking=fatal' '--with-mbstring' '--with-pdo' '--enable-fg' '--enable-mbstring' '--enable-mysqlnd' '--with-password-argon2' '--with-sodium=shared' '--with-pdo-sqlite=sqlite' '--with-sqlite3=sqlite' '--with-curl' '--with-ldap' '--with-openssl' '--with-zlib' '--with-iodbc=libx86_64-linux-gnu' '--with-apex2' '--disable-cgi' 'build_alias=x86_64-linux-gnu'
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/usr/local/etc/php
Loaded Configuration File	(none)
Scan this dir for additional .ini files	/usr/local/etc/php/conf.d
Additional .ini files parsed	/usr/local/etc/php/conf.d/docker-php-ext-sodium.ini
PHP API	20170718
PHP Extension	20170718
Zend Extension	320170718
Zend Extension Build	API320170718.NTS
PHP Extension Build	API20170718.NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring
IPv6 Support	enabled
DTrace Support	disabled
Registered PHP Streams	https, ftps, compress.zlib, php, file, glob, data, http, ftp, phar
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, tls, tlsv1.0, tlsv1.1, tlsv1.2
Registered Stream Filters	zlib.*, convert.iconv.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, dechunk

- En el sitio web <https://documentation.portainer.io/v2.0/deploy/ceinstalldocker> encontrarás información sobre cómo instalar "PortainerCE" en varios sistemas. Aquí seguiremos las instrucciones para instalar en sistemas Linux. "PortainerCE", en lugar de ser una aplicación propiamente dicha, se aloja dentro de un contenedor Docker, por lo que ponerlo en marcha significa poner en marcha sus contenedores.

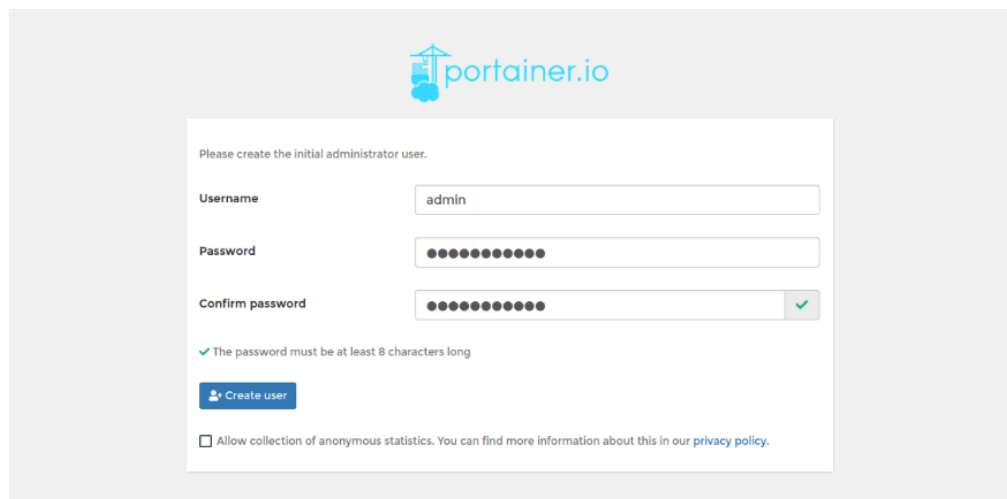
En primer lugar, crearemos un volumen donde almacenar la información. Lo haremos con:

```
uwu@uwu-virtualbox:~$ docker volume create portainer_data
portainer_data
uwu@uwu-virtualbox:~$
```

- Una vez creado el volumen, procederemos a lanzar el contenedor que contiene todo lo necesario para que funcione. "PortainerCE" se gestiona a través de una interfaz web, en este ejemplo mapearemos el servicio web a los puertos del anfitrión 8000 (para el túnel TCP si actúa como servidor de otros agentes de Portainer CE, opcional) y 9000 (para acceder a la interfaz web).

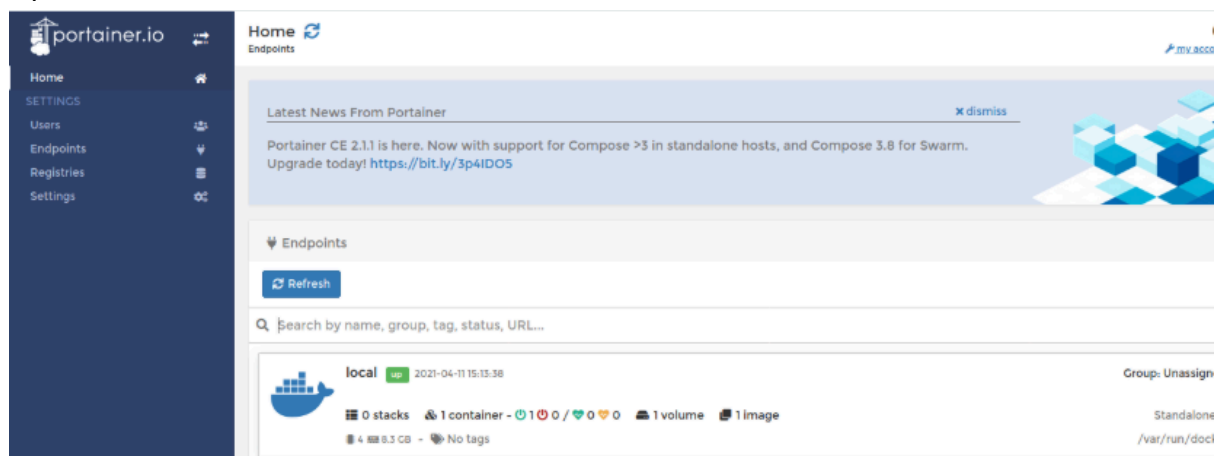
```
uwu@uwu-virtualbox:~$ docker run -d -p 8000:8000 -p 9000:9000 --name=portainer
r --restart=always -v /var/run/docker.sock:/var/run/docker.sock -v portainer_
data:/data portainer/portainer-ce
Unable to find image 'portainer/portainer-ce:latest' locally
latest: Pulling from portainer/portainer-ce
57654d40e0a5: Pull complete
1f476acfabd6: Pull complete
5171176db7f2: Pull complete
52e9438966a5: Extracting 196.6kB/17.99MB
43d4775415ac: Downloading 6.658MB/16.3MB
c1cad9f5200f: Downloading 9.452MB/15.27MB
```

- Después de esto, podremos acceder a la interfaz de "Portainer CE" accediendo a <http://localhost:9000>. En el primer acceso, se nos solicitará que creamos una contraseña para el usuario "admin", con al menos 8 caracteres de longitud. Desde aquí proponemos la contraseña "cefireadmin".



Entramos y seleccionamos la opción de docker.

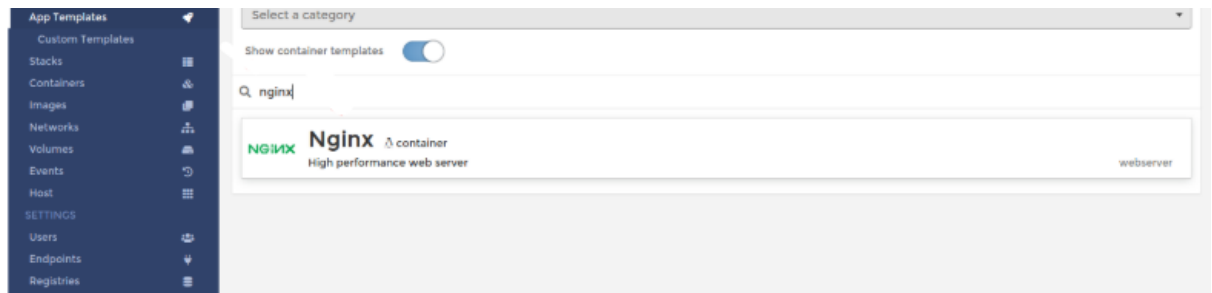
Y si todo ha ido bien, tendremos "Portainer CE" ya en funcionamiento como se ve aquí:



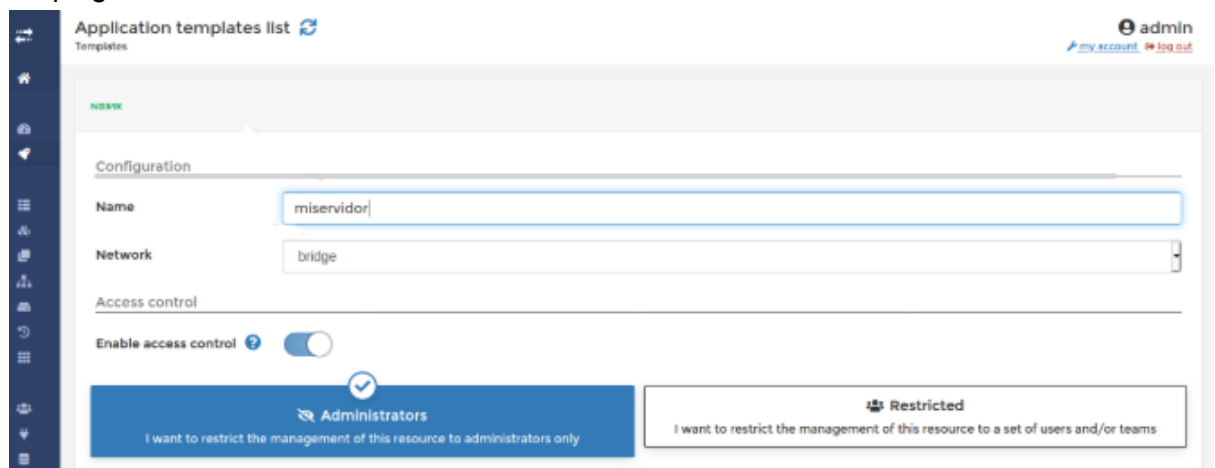
Haremos clic en "local" para poder conectarnos y gestionar nuestro Docker local.

- Una vez conectado, aparecerá a la izquierda un menú con opciones relacionadas con la gestión de Docker, sobre las cuales podremos operar para gestionar nuestros contenedores, imágenes, redes, volúmenes, etc.

Nos dirigiremos a la opción "AppTemplates" para instalar un contenedor con el sitio web "Nginx". Para ello, haremos clic en "AppTemplates" y luego, en donde dice "Search", escribiremos "nginx" y haremos clic en el resultado que aparezca.



- Después de eso, prepararemos el contenedor, indicando la información solicitada (nombre del contenedor, red). En "Show advanced options" podemos indicar información como el mapeo de puertos y volúmenes, pero para este ejemplo no lo manipularemos. Con todo listo, haremos clic en "Deploy the container" para desplegarlo.



- Una vez desplegado, nos redirigirá automáticamente a la pestaña con información de los contenedores, tal como se muestra en la siguiente imagen:

<input type="checkbox"/>	miservidor	running		-	nginx:latest	2021-04-11 20:02:22	49192:80 49191:443	administrators
<input type="checkbox"/>	portainer	running		-	portainer/portainer-ce	2021-04-11 15:09:47	8000:8000 9000:9000	administrators

En esta imagen observamos que el puerto 80 del contenedor con el servidor "Nginx" ha sido mapeado al puerto del anfitrión 49192, por lo tanto, accediendo a <http://localhost:49192> podremos acceder a nuestros servidores "Nginx" desplegados y gestionados desde "Portainer CE".

problemas encontrados:

Hay comandos que fallaban y había que solucionarlos.
Al comienzo de la práctica fallaba demasiado.

Webgrafía:

Documentación entregada por nuestro profesor de Sistemas.

Fin

Oscar apruebame porfa plis 🥺