

Introducción a DOCKER

Florencia Ibarra Freire

1º DAM

Sistemas Informáticos

ÍNDICE

Ejercicio 2.....	2
Introducción.....	2
Capturas de pantalla instalación Docker.....	2
Comandos utilizados.....	2
Capturas de pantalla instalación imagen Hello-World.....	3
Comandos utilizados.....	3
Ejercicio 3.....	4
Introducción.....	4
Capturas de pantalla Wordpress.....	4
Comandos utilizados.....	4
Capturas de pantalla Interfaz Gráfica mediante NoVNC.....	6
Comandos utilizados.....	6
Ejercicio 4.....	7
Introducción.....	7
Capturas de pantalla subida imagen a Docker Hub.....	7
Comandos utilizados.....	7
Capturas de pantalla ubuntu nano.....	8
Comandos utilizados.....	9
Capturas de pantalla APP en Node.....	10
Ejercicio 5.....	11
Introducción.....	11
Capturas de pantalla Wordpress + contenedores.....	11
Comandos utilizados.....	13
Ejercicio 6.....	13
Introducción.....	13
Capturas de pantalla Wordpress con Docker Compose.....	15
Comandos utilizados.....	16
Ejercicio 7.....	17
Introducción.....	17
Capturas de pantalla VS Code conectado a contenedores.....	17
Comandos utilizados.....	20
Conectar VS con contenedor.....	20

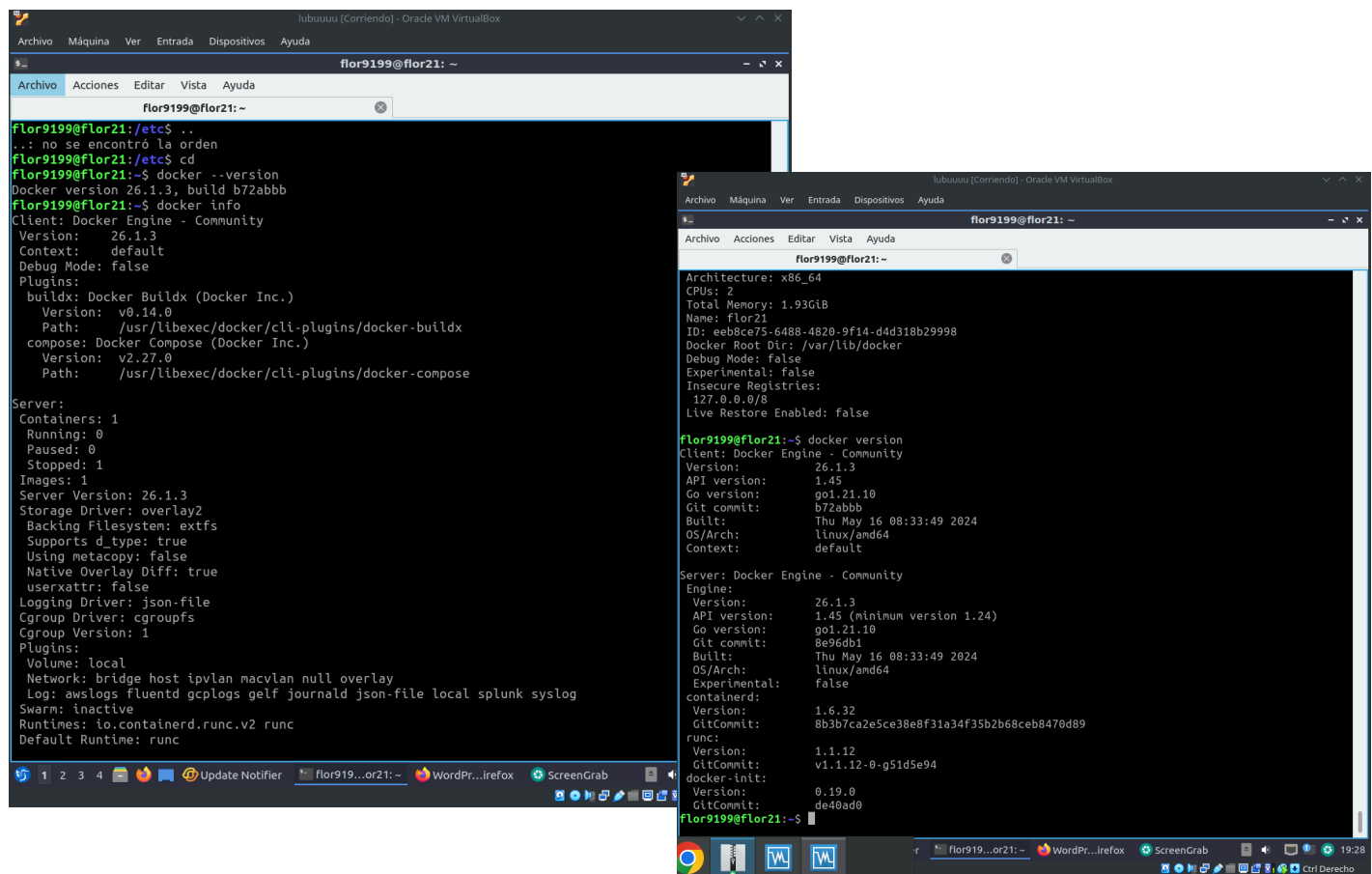
Ejercicio 2

Introducción

Con la ayuda de los PDFs y de los comandos iremos instalando Docker y diferentes complementos que harán posible la creación tanto de contenedores como de imágenes para los mismos.

En esta parte de la tarea se procederá a instalar Docker, junto con la imagen "Hello-World"

Capturas de pantalla instalación Docker



Comandos utilizados

Para llegar a instalar Docker se han tenido que pasar por varios comandos, como:

- `sudo apt-get install apt-transport-https ca-certificates curl gnupg-agent software-properties-common`
- `curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -`
- `sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable"`

- `sudo apt update` (actualizando el paquete de base de datos con los paquetes de Docker del repositorio recién agregado)
- `sudo apt install docker-ce`
- `sudo systemctl status docker`

Tras estos comandos ya estaría instalado Docker y podríamos introducir nuevos comandos para poder ejecutar el comando Docker sin la necesidad de utilizar "sudo" esto se podrá realizar gracias a la ayuda del siguiente enlace: [Cómo instalar y usar Docker en Ubuntu 20.04 | DigitalOcean](https://www.digitalocean.com/articles/how-to-install-and-use-docker-on-ubuntu-20-04)

Capturas de pantalla instalación imagen Hello-World

```

flor9199@flor21: ~
Built: Thu May 16 08:33:49 2024
OS/Arch: linux/amd64
Experimental: false
containerd:
  Version: 1.6.32
  GitCommit: 8b3b7c2e5ce38e8f31a34f35b2b68ceb8470d89
runc:
  Version: 1.1.12
  GitCommit: v1.1.12-0-g51d5e94
docker-init:
  Version: 0.19.0
  GitCommit: de40ad0
flor9199@flor21:~$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:266b191e926f65542fa8daaec01a192c4d292bfff79426f47300a046e1bc576fd
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/
flor9199@flor21:~$
  
```

Comandos utilizados

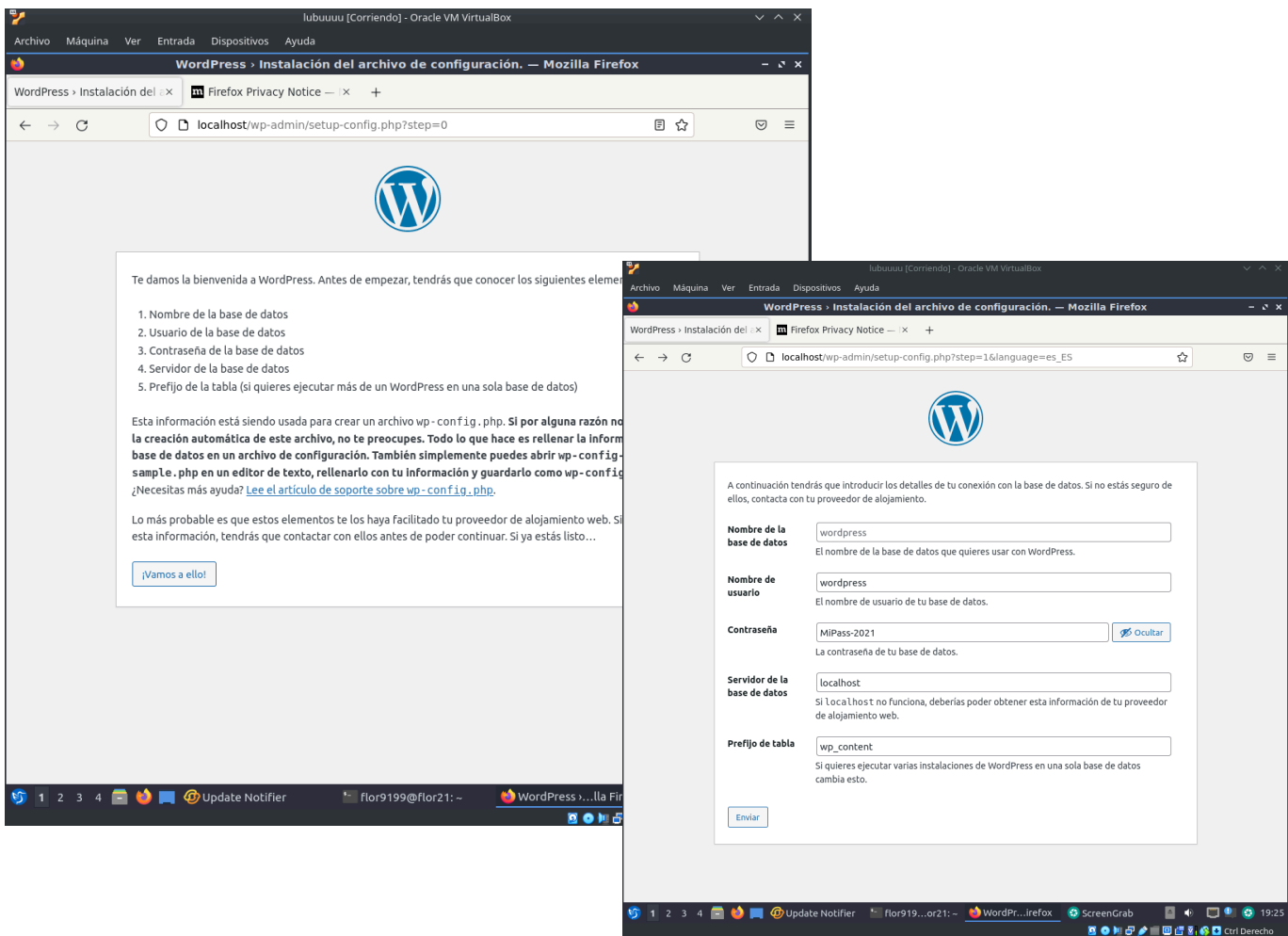
Para poder instalar la imagen pedida debemos ejecutar docker sin el comando sudo para poder meternos a realizar funciones desde dentro del contenedor. El comando necesario para saber si podemos acceder a imágenes es el siguiente: `docker run hello-world`. Y una vez ejecutado nos aparecerá el mensaje de la captura.

Ejercicio 3

Introducción

Para las prácticas de la tercera parte utilizaremos la imagen oficial de “**Ubuntu**” e instalaremos varios programas como los siguientes: MySQL, Apache, PHP, LAMP y Wordpress. Lo que queremos en estos casos es que nos funcione la página de Wordpress accediendo desde el *localhost* y también que se acceda a la interfaz gráfica desde el navegador mediante el cliente NoVNC.

Capturas de pantalla Wordpress



Comandos utilizados

Primero que nada, para llegar a la siguiente imagen necesitamos instalar un par de cosas utilizando los siguientes comandos:

- `docker run -it -p 8080:80 --name LAMP ubuntu /bin/bash` con el cual se creará un contenedor con la imagen de "Ubuntu" y lo situaremos en el lugar 80 dentro del puerto 8080 de nuestro propio sistema.
- `apt install wordpress php libapache2-mod-php mysql-server php-mysql` con este comando instalaremos varios paquetes, nombrados en el mismo.
- `service apache2 start` para iniciar apache2.

Tras esto vamos a crear dos ficheros, uno de ellos en el directorio de apache2 llamado "*wordpress.conf*", y tenemos que copiar el siguiente contenido dentro del mismo:

```
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>
```

Tras esto utilizaremos los siguientes comandos para habilitar "URL rewriting":

- `a2ensite wordpress`
- `a2enmod rewrite`
- `service apache2 reload`

Después iniciaremos MySQL:

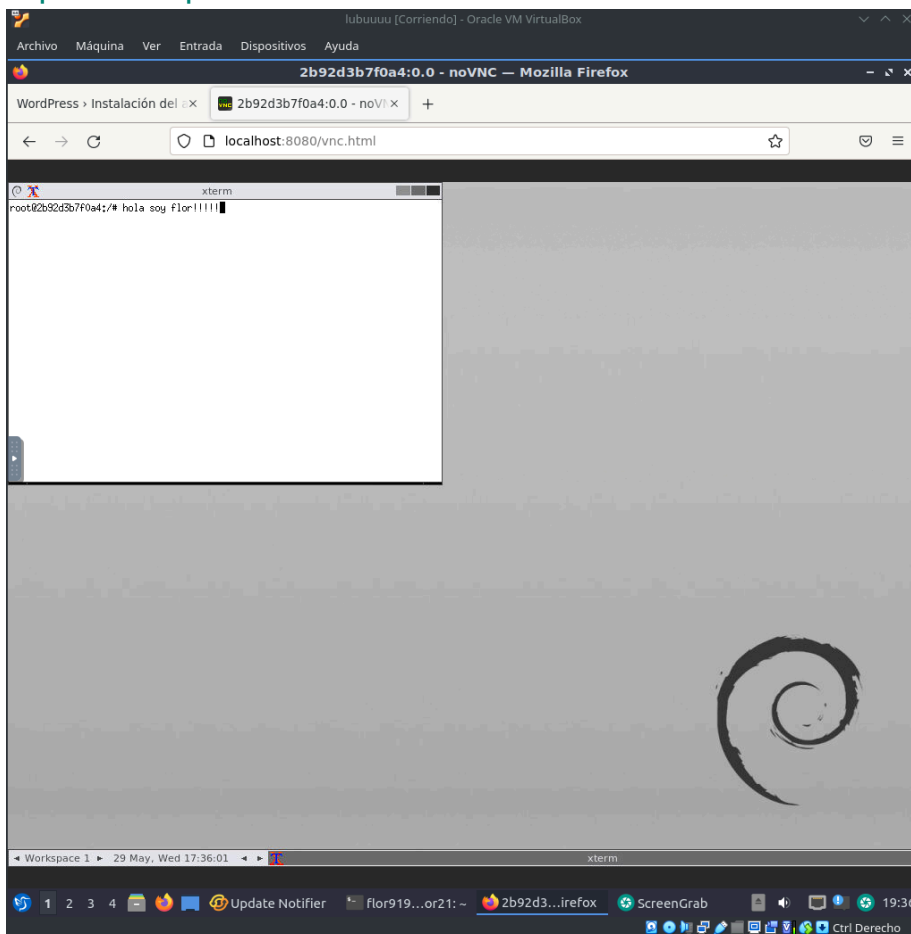
- `service mysql start`
- `mysql_secure_installation`
- `mysql -u root -p`
 - Dentro de MySQL escribiremos lo siguiente:
 - `CREATE DATABASE wordpress;`
 - `CREATE USER 'wordpress'@'%' IDENTIFIED BY 'MiPass-2021';`
 - `GRANT ALL PRIVILEGES ON wordpress.* TO 'wordpress'@'%' WITH GRANT OPTION;`
 - `FLUSH PRIVILEGES;`

Por último pero no menos importante, debemos crear un fichero dentro del directorio “wordpress” llamado “config-localhost.php” y le pondremos lo siguiente:

```
<?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');
?>
```

Tras guardarlo podremos acceder a la url <http://localhost:8080> y nos saldrá la imagen de las capturas.

Capturas de pantalla Interfaz Gráfica mediante NoVNC.



Comandos utilizados

Para llegar a la imagen necesitaremos un simple comando: `docker run --rm -it -p 8080:8080 theasp/novnc`

Si accedemos a la url: <http://localhost:8080/vnc.html> nos llevará a la interfaz.

Tenemos que ver que con este comando utilizamos el parámetro `-rm`, el cual nos indica, que, tras cerrar la terminal el contenedor se borrará.

Ejercicio 4

Introducción

En la cuarta parte de la actividad veremos varias cosas, como subir nuestras imágenes recién creadas (o no) a Docker Hub, teniendo varios repositorios con las imágenes que queramos subir. También instalaremos Visual Studio Code para poder crear dockerfiles y a partir de ahí crear imágenes.

Capturas de pantalla subida imagen a Docker Hub

The screenshot shows a terminal window on the left and a Docker Hub repository page on the right.

Terminal Output:

```

flor919@PC21: ~
4b3987470ec0 ubuntu "/bin/bash" 13 days ago Exited (127) 13 days ago
hungry_wilbur
00d109c14851 hello-world "/hello" 13 days ago Exited (0) 13 days ago
romantic_chebyshev
a7a5eba1f2e3 ubuntu "/bin/bash" 13 days ago Exited (127) 13 days ago
stupefied_satoshi
aba18a2ebe27 hello-world "/hello" 13 days ago Exited (0) 13 days ago
nifty_poitras

flor919@PC21:~$ docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
flor/ubuntumod 2024 4cf9f2049d7d 22 seconds ago 11 MB
<none> <none> 9d30db629099 About a minute ago 11 MB
ubuntu latest bf3dc08bfed0 4 weeks ago 76 MB
hello-world latest d2c94e258dcb 13 months ago 13 MB
alpine 3.10 e7b300aee9f9 3 years ago 5.5 MB

flor919@PC21:~$ docker login
Authenticating with existing credentials...
WARNING! Your password will be stored unencrypted in /home/flor919/.docker/config.json
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded

flor919@PC21:~$ docker commit -a "Flor" -m "Ubuntu modificado" 428dd77fc9a0
sha256:3bb9392fc9d09a90010cae10fd3f6f997cb8c44a4dd7efd7d07226fd07226fd07226fd

flor919@PC21:~$ docker push floppy21/prueba
Using default tag: latest
The push refers to repository [docker.io/floppy21/prueba]
52aef66c9df7: Pushing 34.64MB/35.63MB
80098e3d304c: Mounted from library/ubuntu

```

Docker Hub Repository Page:

Repository: **floppy21/prueba**
Updated 1 minute ago
This repository does not have a description **INCOMPLETE**
This repository does not have a category **INCOMPLETE**

Docker commands
To push a new tag to this repository:

```
docker push floppy21/prueba:tagname
```

Tags
This repository contains 1 tag(s).

Tag	OS	Type	Pulled	Pushed
latest	linux	Image	---	a few seconds ago

Repositorio Docker Hub

Comandos utilizados

- `docker pull` (para poder descargar imágenes)
- `docker commit -a "autor" -m "comentario" ID/NOMBRE-CONTENEDOR usuario/imagen:[version]` con este comando lo que hacemos es crear nuestras propias imágenes.
- En mi caso: `flor919@PC21:~$ docker commit -a "Flor" -m "Ubuntu modificado" 428dd77fc9a0 floppy21/prueba`

Es importante loguearnos antes de hacer algún docker commit o un push.

- `docker push floppy21/prueba` para subir el commit a Docker Hub.

Capturas de pantalla ubuntu nano

```

flor919@PC21:~$ cd ~/Desktop/pruebaDockerFile/
flor919@PC21:~/Desktop/pruebaDockerFile$ docker build -t ubuntu nano ./
[+] Building 7.2s (6/6) FINISHED
=> [internal] load build definition from Dockerfile                                0.0s
=> => transferring dockerfile: 108B                                              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                                6.8s
=> exporting to image                                                            0.2s
=> => exporting layers                                                            0.2s
=> => writing image sha256:ed26983294bfcd1d97b8e9278229502b76b200ca7a45148ebb54 0.0s
=> => naming to docker.io/library/ubuntu nano                                  0.0s
flor919@PC21:~/Desktop/pruebaDockerFile$ docker history ubuntu nano
IMAGE          CREATED          CREATED BY          SIZE
COMMENT
ed26983294bf  32 seconds ago  CMD ["/bin/sh" "-c" "/bin/bash"]  0B
  buildkit.dockerfile.v0
<missing>     32 seconds ago  RUN /bin/sh -c apt update && apt install -y ...  37.3MB
  buildkit.dockerfile.v0
<missing>     4 weeks ago    /bin/sh -c #(nop)  CMD ["/bin/bash"]          0B
<missing>     4 weeks ago    /bin/sh -c #(nop)  ADD file:ac9d5a9d5b9b1217a...  76.2MB
<missing>     4 weeks ago    /bin/sh -c #(nop)  LABEL org.opencontainers...  0B
<missing>     4 weeks ago    /bin/sh -c #(nop)  LABEL org.opencontainers...  0B
<missing>     4 weeks ago    /bin/sh -c #(nop)  ARG LAUNCHPAD_BUILD_ARCH    0B
<missing>     4 weeks ago    /bin/sh -c #(nop)  ARG RELEASE                  0B

flor919@PC21:~/Desktop/pruebaDockerFile$ docker run -it ubuntu nano
root@997f8a33a953:/#

```

Para crear el dockerfile iremos a VS Code, donde escribiremos lo siguiente:

#Imagen base ubuntu

FROM ubuntu

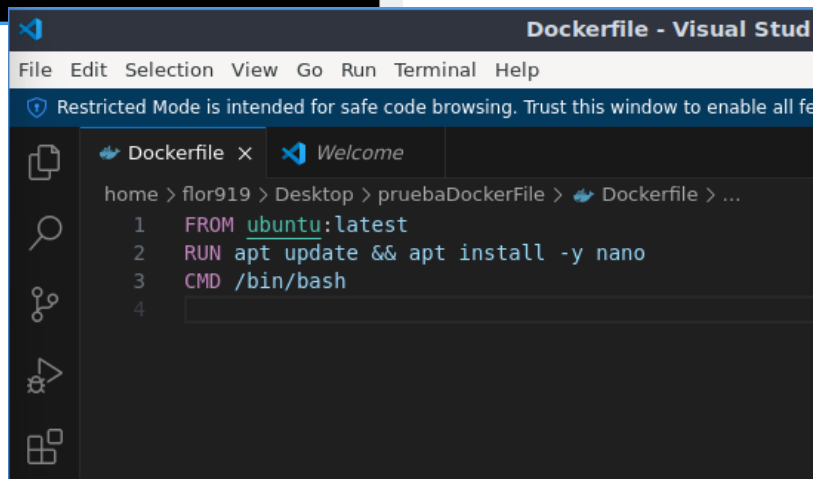
Actualizamos lista de paquetes e instalamos nano (-y para no preguntar)

Las últimas líneas son para hacer la imagen más 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



Comandos utilizados

Dentro de la terminal nos situaremos dentro del directorio donde hemos creado el dockerfile y pondremos los siguientes comandos:

- `docker build -t ubuntunano ./` donde crearemos la imagen
- `docker run -it ubuntunano`
- `docker commit -a "Flor-Ibarra" -m "Ubuntunano" 997f8a33a953 flopy21/ubuntunano`
- `docker push flopy21/ubuntunano`

```

997f8a33a953  ubuntunano      "/bin/sh -c /bin/bash"  2 days ago      Exited (0) 2 da
ys ago      gracious_ride
428dd77fc9a0  ubuntu          "/bin/bash"           2 weeks ago     Exited (0) 2 we
eks ago     LAMP
4b3987470ec0  ubuntu          "/bin/bash"           2 weeks ago     Exited (127) 2
weeks ago   hungry_wilbur
00d109c14851  hello-world     "/hello"              2 weeks ago     Exited (0) 2 we
eks ago     romantic_chebyshev
a7a5eba1f2e3  ubuntu          "/bin/bash"           2 weeks ago     Exited (127) 2
weeks ago   stupefied_satoshi
aba18a2ebe27  hello-world     "/hello"              2 weeks ago     Exited (0) 2 we
eks ago     nifty_poitras
flor919@PC21:~$ docker login
Authenticating with existing credentials...
WARNING! Your password will be stored unencrypted in /home/flor919/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
flor919@PC21:~$ docker commit -a "Flor-Ibarra" -m "Ubuntunano" 997f8a33a953 flopy21/ubu
ntunano
sha256:9820338957a9cf671950c0ec0f3557964cd1801e33d2ab181722dafa746a3b4f
flor919@PC21:~$ docker push flopy21/ubuntunano
Using default tag: latest
The push refers to repository [docker.io/flopy21/ubuntunano]
5c5f405952f0: Pushed
d8c24892e99d: Pushed
80098e3d304c: Mounted from flopy21/prueba
latest: digest: sha256:8ec3df06e2439cb3f04b9dde9f7f6edd833d9d889af24706feb8d2fc10f8d79d
size: 948
flor919@PC21:~$ █

```

[Image Layer Details - flopy21/ubuntunano:latest | Docker Hub](#)

Capturas de pantalla APP en Node

```

flor919@PC21: ~/Desktop/sampledocker/app
flor919@PC21:~/Desktop/sampledocker/app$ docker build -t sampledocker ./
[+] Building 5.2s (10/10) FINISHED                                docker:default
=> [internal] load build definition from Dockerfile                0.0s
=> => transferring dockerfile: 138B                               0.0s
=> [internal] load metadata for docker.io/library/node:12-alpine  1.5s
=> [auth] library/node:pull token for registry-1.docker.io        0.0s
=> [internal] load .dockerignore                                  0.0s
=> => transferring context: 2B                                     0.0s
=> [1/4] FROM docker.io/library/node:12-alpine@sha256:d4b15b3d48f42059a15bd659be60af 2.9s
=> => resolve docker.io/library/node:12-alpine@sha256:d4b15b3d48f42059a15bd659be60af 0.0s
=> => sha256:d4b15b3d48f42059a15bd659be60afe21762aae9d6cbea6f1244408 1.43kB / 1.43kB 0.0s
=> => sha256:4517380049fc3c9aacceae7764fcf3500354b0ac8a47e4afb35b5bb 1.16kB / 1.16kB 0.0s
=> => sha256:bb6d28039b8cec9aa8d9032f9aa640a792a60c2cb1644691627bf04 6.58kB / 6.58kB 0.0s
=> => sha256:df9b9388f04ad6279a7410b85cedfdbcb2208c0a003da7ab5613af71 2.81MB / 2.81MB 0.4s
=> => sha256:3bf6d738020517f4622814e8c21db4b4aaa78ae7cab4e4f872c11 24.91MB / 24.91MB 1.6s
=> => sha256:7939e601ee5e4737cf7fdb6d1dfe31ca4c2697109290462f6947107 2.36MB / 2.36MB 0.6s
=> => extracting sha256:df9b9388f04ad6279a7410b85cedfdbcb2208c0a003da7ab5613af7107914 0.1s
=> => sha256:31f0fb9de071269230cb0f786012ae4e81d26e489b1fe922e57b5201e6b 451B / 451B 0.6s
=> => extracting sha256:3bf6d738020517f4622814e8c21db4b4aaa78ae7cab4e4f872c11696886c 1.1s
=> => extracting sha256:7939e601ee5e4737cf7fdb6d1dfe31ca4c2697109290462f694710761450 0.1s
=> => extracting sha256:31f0fb9de071269230cb0f786012ae4e81d26e489b1fe922e57b5201e6bc 0.0s
=> [internal] load build context                                  0.0s
=> => transferring context: 138B                                   0.0s
=> [2/4] WORKDIR /app                                           0.0s
=> [3/4] COPY . .                                                0.0s
=> [4/4] RUN yarn install --production                          0.0s
=> exporting to image                                           0.0s
=> => exporting layers                                           0.0s
=> => writing image sha256:2120013f164cd731cf9c                 0.0s
=> => naming to docker.io/library/sampledocker                  0.0s
flor919@PC21:~/Desktop/sampledocker/app$

```

```

Dockerfile - Visual Studio Code
File Edit Selection View Go Run Terminal Help
Restricted Mode is intended for safe code browsing. Trust this window to enable all features.
Welcome Dockerfile x
home > flor919 > Desktop > sampledocker > app > Dockerfile > ...
1 FROM node:12-alpine
2 WORKDIR /app
3 COPY . .
4 RUN yarn install --production
5 CMD ["node", "src/index.js"]
6

```

```

flor919@PC21: ~/Desktop/sampledocker/app
flor919@PC21:~/Desktop/sampledocker/app$ docker run -dp 3000:3000 sampledocker
8b98c20282f47682c547d744b8ad6ad9f2d39002eddcbb11facbc98ac27080660

```

Tras crear el dockerfile se hace exactamente lo mismo que en las anteriores capturas pero esta vez sin subirlo a Docker Hub, solo que el Dockerfile contendrá otros datos.

Ejercicio 5

Introducción

En la quinta parte de la tarea instalaremos Wordpress, creando una red donde se conectarán dos contenedores, uno de ellos utilizando Apache y PHP y el segundo tendrá un servidor de base de datos MySQL.


Capturas de pantalla Wordpress + contenedores.

```
flor919@PC21:~$ docker run --name nuestromysql --network redwp -v /home/flor919/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: Pull complete
6f70cc868145: Pull complete
1f6637f4600d: Pull complete
Digest: sha256:20575e3e3e6216036d25dab5903808211f1e9ba63dc7825ac20cb975e34cfcae
Status: Downloaded newer image for mysql:5.6
e3722f4dfa1313087ce64b733c4ddc766b915cf225e9cdfaaa33458b
flor919@PC21:~$ docker run --name nuestrowp --network redwp -v /home/flor919/wordpress:/var/www/html -e WORDPRESS_DB_NAME=cefiredb -e WORDPRESS_DB_USER=cefireuser -e WORDPRESS_DB_PASSWORD=cefirepass -d wordpress:latest
Unable to find image 'wordpress:latest' locally
latest: Pulling from library/wordpress
09f376ebb190: Pull complete
76afcdc86551: Pull complete
ceed4541c527: Pull complete
9ec84be954b0: Pull complete
ff0e278869f9: Pull complete
1693466e4cc6: Pull complete
57c8d94a4882: Pull complete
43af3fe8136a: Pull complete
ddef75e08a5d: Pull complete
a4ba0b0dbbf0a: Pull complete
29e89bc69515: Pull complete
bdad27815722: Pull complete
dbbbc9a88332: Pull complete
ca4ce74a758b: Pull complete
b8447b6adfa2: Pull complete
6384f9388f98: Pull complete
ad26f7163064: Pull complete
4b2fdddf63d23: Pull complete
5967a1599e89: Pull complete
cabecefe25649: Pull complete
c9f0fbb308c3: Pull complete
Digest: sha256:f468bab53528df6f87dfe11a80de26eff57e0f515ff0e2781693466e4cc657c8d94a43af3feddef75e08a5da4ba0b0db29e89bcdad278dbbbc9a
```

WordPress › Instalación del archivo de configuración. — Mozilla Firefox

WordPress › Instalación del archivo de configuración. — Mozilla Firefox

← → ↻ 🔍 localhost:8080/wp-admin/setup-config.php?step=1&lang=es



A continuación tendrás que introducir los detalles de tu conexión con la base de datos. Si no estás seguro de ellos, contacta con tu proveedor de alojamiento.

Nombre de la base de datos
El nombre de la base de datos que quieres usar con WordPress.

Nombre de usuario
El nombre de usuario de tu base de datos.

Contraseña [Ocultar](#)
La contraseña de tu base de datos.

Servidor de la base de datos
Si localhost no funciona, deberías poder obtener esta información de tu proveedor de alojamiento web.

Prefijo de tabla
Si quieres ejecutar varias instalaciones de WordPress en una sola base de datos cambia esto.



¡Muy bien! Ya has terminado esta parte de la instalación. Ahora WordPress puede comunicarse con tu base de datos. Si estás listo, es el momento de...

[Realizar la instalación](#)



¡Lo lograste!

WordPress ya está instalado. ¡Gracias, y que lo disfrutes!

Nombre de usuario cefireuser

Contraseña *La contraseña que has elegido.*

[Acceder](#)

Escritorio < Flor — WordPress — Mozilla Firefox

localhost:8080/wp-admin/

Hola, cefireuser

Opciones de pantalla Ayuda

¡Te damos la bienvenida a WordPress!

[Aprende más sobre la versión 6.5.3.](#)

Crea contenido rico con bloques y patrones

Los patrones de bloques son diseños de bloques preconfigurados. Úsalos para inspirarte o crear nuevas páginas en un instante.

[Añadir una nueva página](#)

Personaliza todo tu sitio con temas de bloques

Diseña todo en tu sitio — Desde la cabecera hasta el pie de página. Todo usando bloques y patrones.

[Abrir el editor del sitio](#)

Cambia la apariencia de tu sitio con los estilos

¡Retoca tu sitio o dale un aspecto completamente nuevo! Sé creativo — ¿Qué tal una nueva paleta de color o una nueva fuente?

[Editar estilos](#)

```

flor919@PC21:~$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS
64f28b6071c7   wordpress     "docker-entrypoint.s..." 11 minutes ago Up 10 minutes 0.0.0:8080->80/tcp, :::8080->80/tcp
e3722f4dfa13   mysql:5.6     "docker-entrypoint.s..." 12 minutes ago Up 12 minutes 3306/tcp
e5751ea56a3c   alpine        "/bin/sh"                17 minutes ago Up 17 minutes

```

Comandos utilizados

- `docker network create redwp` utilizado para crear la red la cual se llamará "redwp"
- `docker run --name nuestromysql --network redwp -v /home/sergi/mysqldata:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=cefireroot -e MYSQL_USER=cefireuser -e MYSQL_PASSWORD=cefirepass -e MYSQL_DATABASE=cefiredb -d mysql:5.6` Creamos un contenedor llamado "**nuestromysql**", dentro de la red.
- `docker run --name nuestrowp --network redwp -p 8080:80 -d wordpress` se crea el contenedor que contendrá Apache + PHP y Wordpress.
- Por último utilizamos `docker ps` para poder ver los dos contenedores que se están utilizando.

Ejercicio 6

Introducción

En la penúltima parte de la práctica pondremos en marcha otra vez Wordpress pero esta vez desde un fichero `.yaml`.

Antes que nada debemos instalar Docker Compose que es una aplicación que simplifica el poder lanzar múltiples contenedores con orquestador simple. La instalación del Docker Compose lo haremos a través del comando "curl":

```

sudo curl -L "https://github.com/docker/compose/releases/download/1.29.0/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

```

Tras eso creamos el fichero "**docker-compose.yml**" en Visual Studio Code, con el siguiente contenido:

#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 "mysql", version 5.7

image: mysql:5.7

#Mapea en el volumen "db_data" el directorio "/var/lib/mysql", lo que da persistencia al contenido de

#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 mysql, nombre de base de datos,

usuario con permisos root (necesario para conexiones remotas) y password de ese usuario

environment:

MYSQL_ROOT_PASSWORD: somewordpress

MYSQL_DATABASE: wordpress

MYSQL_USER: wordpress

MYSQL_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 8000 del anfitrión

ports:

- "8000:80"

#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:

Capturas de pantalla Wordpress con Docker Compose

```

flor919@PC21: ~/Desktop/doc...D06/CasoPractico1-Wordpress
GNU nano 4.8 docker-compose.yml
#Versión del fichero docker-compose 3.9. No obligatorio desde la versión de docker-compose 1.
version: "3.9"

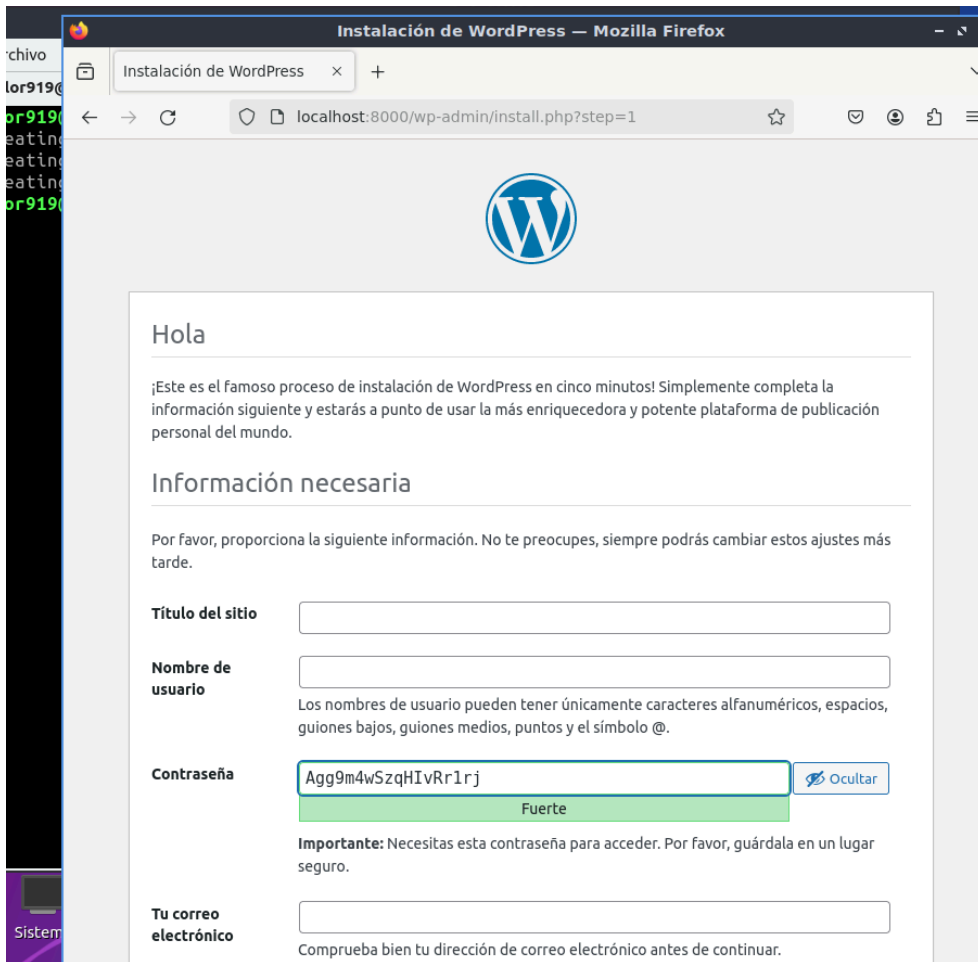
#Indicamos los servicios a lanzar
services:
  #Plantilla del servicio "db"
  db:
    #Se basa en la imagen "mysql", version 5.7
    image: mysql:5.7
    #Mapea en el volumen "db_data" el directorio "/var/lib/mysql", lo que da persistencia al
    #Wordpress almacenado en la base de datos
    volumes:
      - 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 mysql, nombre de base de datos,
    # usuario con permisos root (necesario para conexiones remotas) y password de ese usuario
    environment:
      MYSQL_ROOT_PASSWORD: somewordpress
      MYSQL_DATABASE: wordpress
      MYSQL_USER: wordpress
      MYSQL_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 8000 del anfitrión
    ports:
      - "8000:80"
    #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

```

```

flor919@PC21: ~/Desktop/docker-composeUD06/CasoPractico1-Wordpress
Archivo Acciones Editar Vista Ayuda
flor919@PC21: ~/Desktop/doc...D06/CasoPractico1-Wordpress
flor919@PC21:~/Desktop/docker-composeUD06/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
flor919@PC21:~/Desktop/docker-composeUD06/CasoPractico1-Wordpress$

```

Comandos utilizados

Tras configurar el fichero nos meteremos en la ruta del directorio desde la terminal y pondremos el siguiente comando para poner en marcha el sistema: *docker-compose up -d*

- La opción -d hará que Docker Compose se ejecute en segundo plano
- Mientras que la opción up descarga y construye la imagen.

Ejercicio 7

Introducción

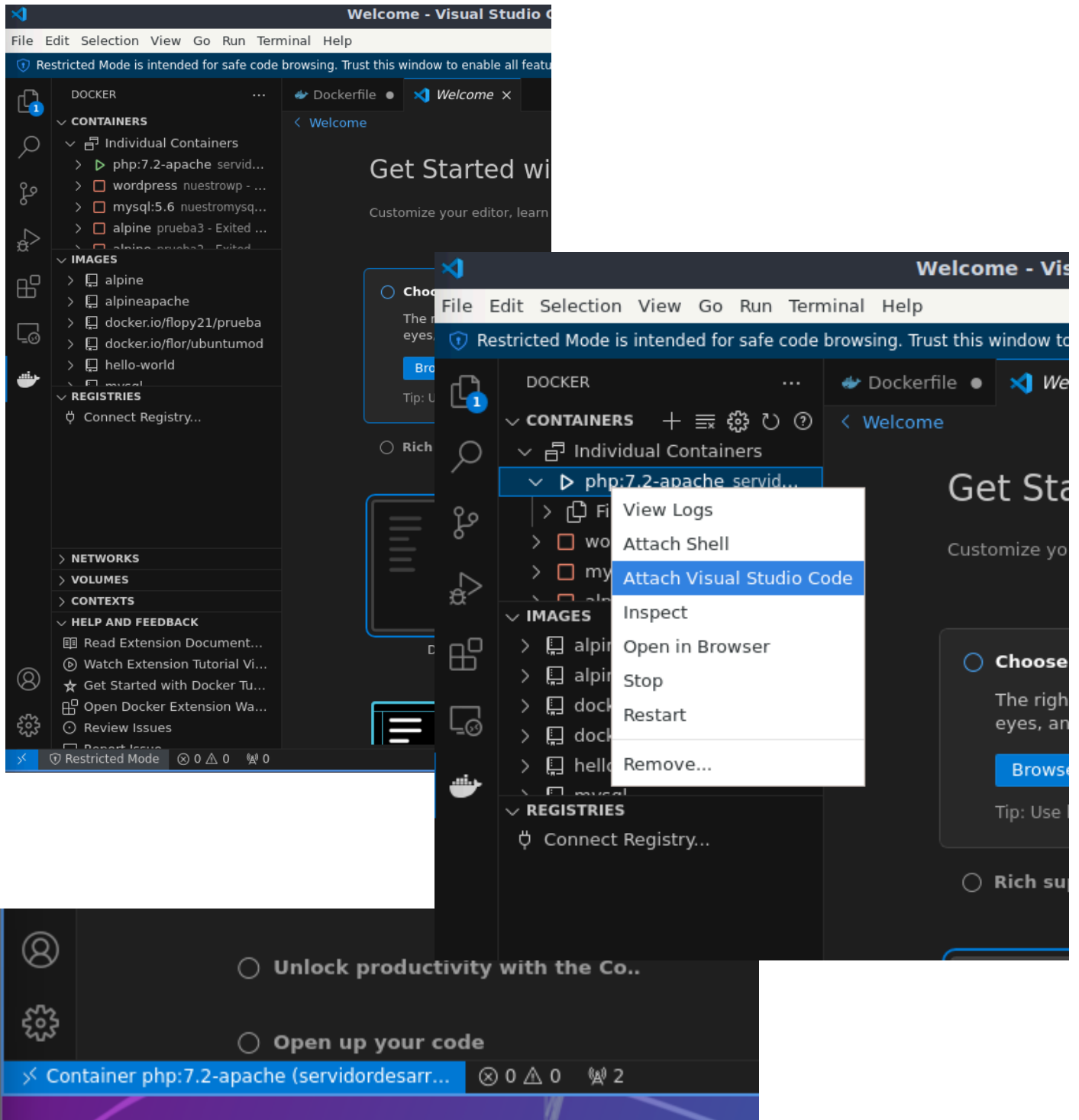
En esta última parte se podrá desarrollar una aplicación web en un contenedor juntando conectores relacionados con Docker en Visual Studio Code.

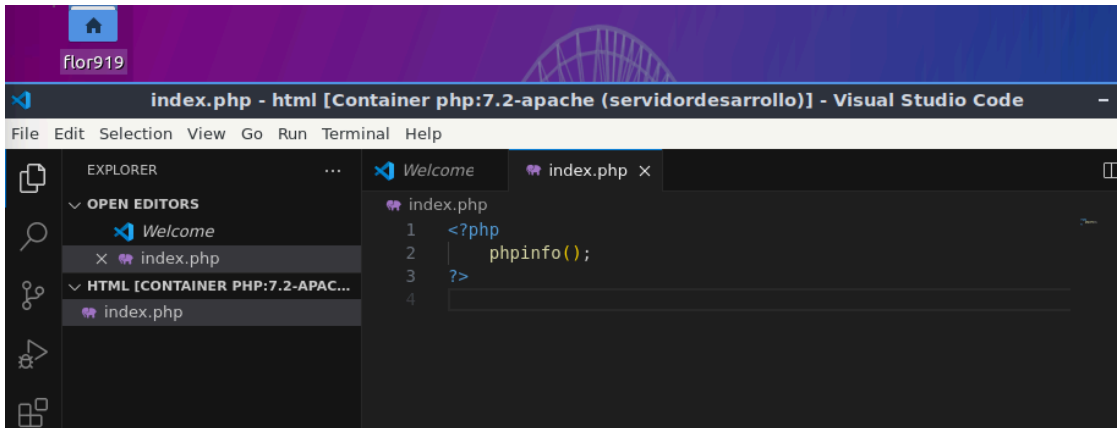
Para que esto funcione debemos instalar algunas extensiones como Docker y Remote - Containers de Microsoft.

En VSCode aparecerá un nuevo ícono, el de una ballena, ese será el que utilizaremos.

Capturas de pantalla VS Code conectado a contenedores

```
flor919@PC21:~$ 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
6c13f026e6da: Pull complete
e5e6cd163689: Pull complete
5c5516e56582: Pull complete
154729f6ba86: Pull complete
Digest: sha256:4dc0f0115acf8c2f0df69295ae822e49f5ad5fe849725847f15aa0e5802b55f8
Status: Downloaded newer image for php:7.2-apache
cbca462850b7b1618c189d253c3aefaa8a46c7ec8a8a92608c81b3778ce9e25b
flor919@PC21:~$
```





FlorLubu [Corriendo] - Oracle VM VirtualBox

Archivo Máquina Ver Entrada Dispositivos Ayuda

phpinfo() — Mozilla Firefox

phpinfo()

localhost:8080

PHP Version 7.2.34

System	Linux cbca462850b7 5.15.0-107-generic #117~20.04.1-Ubuntu SMP Tue Apr 30 10:35:57 UTC 2024 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-option-checking=fatal' '--with-mhash' '--with-pic' '--enable-ftp' '--enable-mbstring' '--enable-mysqlnd' '--with-password-argon2' '--with-sodium=shared' '--with-pdo-sqlite=/usr' '--with-sqlite3=/usr' '--with-curl' '--with-libedit' '--with-openssl' '--with-zlib' '--with-libdir=lib/x86_64-linux-gnu' '--with-apxs2' '--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, tlsv1.2
Registered Stream Filters	zlib.*, convert.iconv.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, dechunk

This program makes use of the Zend Scripting Language Engine:
Zend Engine v3.2.0, Copyright (c) 1998-2018 Zend Technologies

zendengine

Comandos utilizados

En esta parte se ha utilizado solo un comando: *docker run -d --name servidordesarrollo -p 8080: 80 php: 7.2-apache*, con el cual se crea un contenedor que se utilizará en el puerto 8080. Tendrá Apache + PHP.

Conectar VS con contenedor

Para conectar VS deberemos meternos en el icono de la ballena, allí nos aparecerá un menú en el cual estarán los contenedores, depende de cuántos tengamos se verá más grande o menos la opción de "Individual Containers".

- Buscaremos el que hemos creado con el comando, el cual pone php: 7.2-apache.
- Le daremos con el botón derecho del mouse en el contenedor y pinchamos donde pone "Attach Visual Studio Code"
- Se abrirá una nueva ventana en VS donde podemos ver en la esquina inferior izquierda que se ha conectado al contenedor que queríamos.
- En esa nueva ventana iremos al explorador y abriremos el index.php
- Dentro copiaremos lo siguiente:

```
<?php
    phpinfo();
?>
```

- Por último si accedemos a la url <http://localhost:8080> se nos abrirá la aplicación.