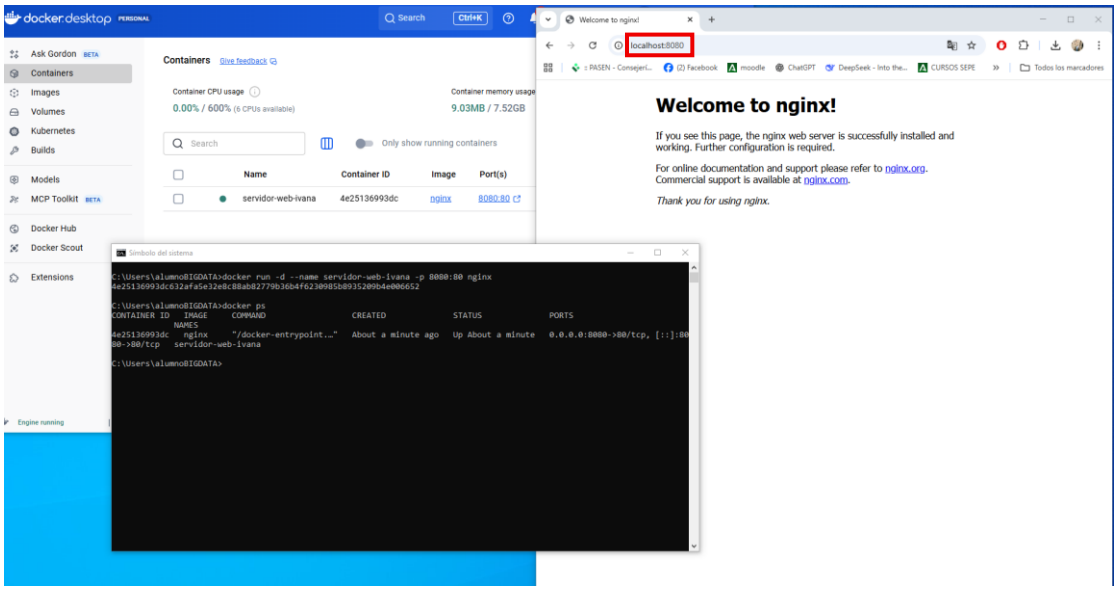
1- Crear un contenedor de Docker que tenga la capacidad de ejecutar un servidor web, utilizando una de las imágenes existentes en Docker Hub. Para ello, se debe acceder a Docker Hub, (repositorio en línea de imágenes para contenedores Docker), y buscar una imagen de servidor web que se adecue a las necesidades del proyecto. Luego, se debe utilizar el comando docker run junto con el nombre de la imagen seleccionada para crear y ejecutar el contenedor. Por ejemplo, se podría utilizar la imagen "nginx" para crear el contenedor de servidor web.

docker run -d --name servidor-web-ivana -p 8080:80 nginx



2- Crear una imagen de docker mediante un fichero Dockerfile que tenga las siguientes características:

3- Usar una plantilla de Debian.

4- Instalar Apache2. (apt update && apt install apache2 -y)

5- Establecer el directorio por defecto en /var/www/html

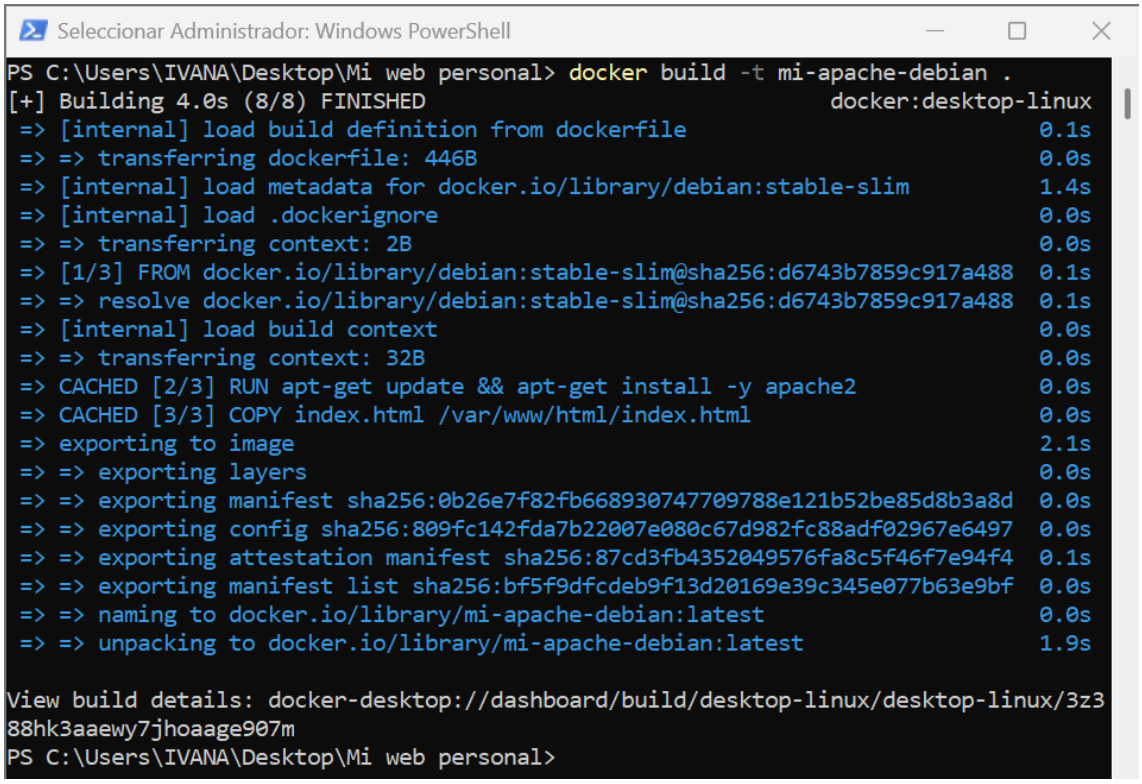
6- Mover una página web que os descarguéis de Internet a dicho directorio.

7- El Puerto 80 debe de estar abierto.

8- Añadir al final del fichero la siguiente instrucción para que apache2 funcione en segundo plano:

CMD [“/usr/sbin/apache2ctl”,”-DFOREGROUND”]

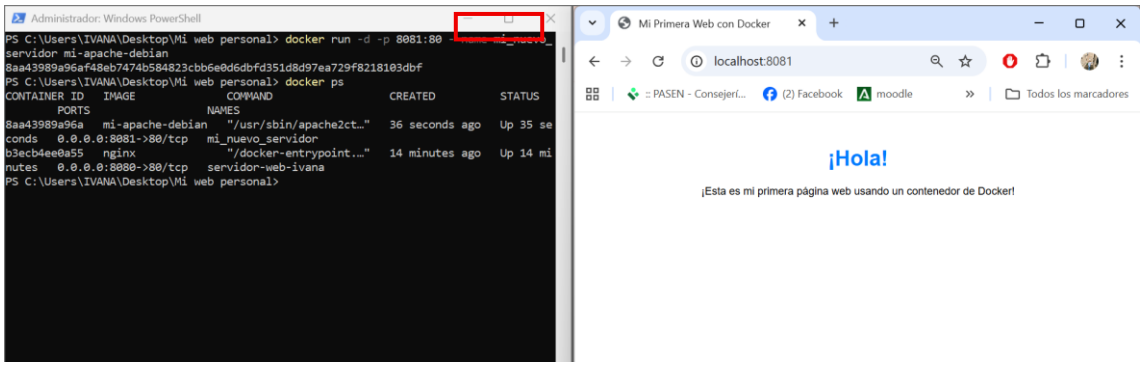
Lo haremos con el comando: *docker build -t mi-apache-debian .*



3- Crear un contenedor a partir del fichero anterior y arrancarlo. Comprobar que la web que has descargado se visualiza en localhost. ¿Qué harías para que la página web pudiera ser visible en localhost:8081?

Simplemente a la hora de crear el contenedor le especifico el puerto 8081 en localhost:

docker run -d -p 8081:80 --name mi_nuevo_servidor mi-apache-debian



4. ¿Le has dado algún nombre a la imagen? ¿Qué nombre tiene la imagen que has creado? ¿Qué nombre tiene el contenedor?

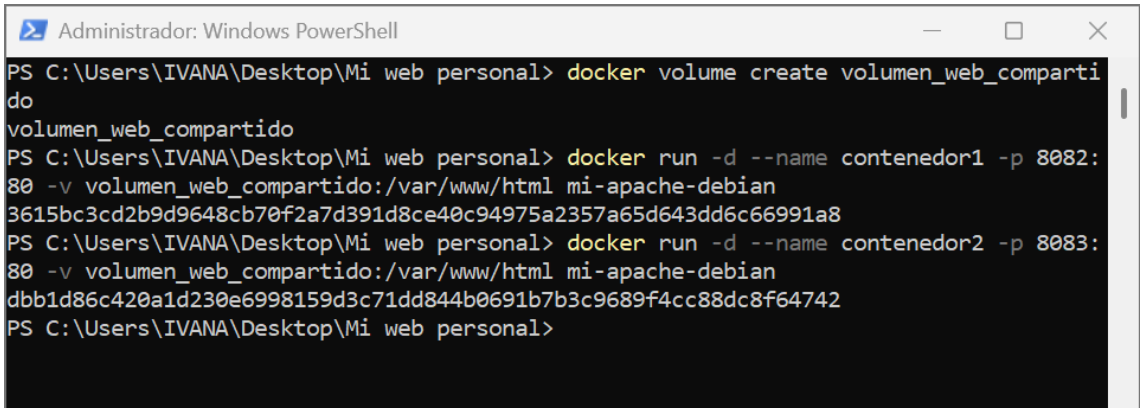
A la imagen le he dado el nombre de "mi_apache_debian" y al contendor le he dado el de "*mi-nuevo-servidor*"

5. Crear dos nuevos contenedores a partir de la imagen anterior con las siguientes características:
   - Un volumen que será compartido entre los dos contendores y que conectará dentro del contenedor con la ruta /var/www/html
   - Los puertos de acceso a los contenedores deben ser 8082 y 8083

   docker volume create volumen_web_compartido

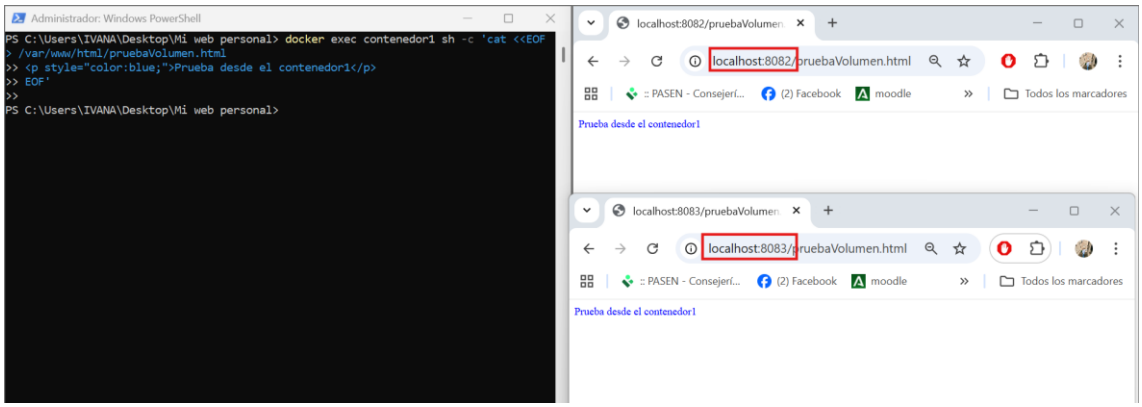
docker run -d --name contenedor1 -p 8082:80 -v volumen_web_compartido:/var/www/html mi-apache-debian

docker run -d --name contenedor2 -p 8083:80 -v volumen_web_compartido:/var/www/html mi-apache-debian



Conéctate al primero de los contenedores y crea un fichero pruebaVolumen.html escribe dentro lo siguiente: Prueba desde el contenedor 1 .
Accede a la ruta http://localhost:8082/pruebaVolumen.html.

docker exec contenedor1 sh -c 'cat <<EOF > /var/www/html/pruebaVolumen.html
*** Prueba desde el contenedor1 ***
EOF'



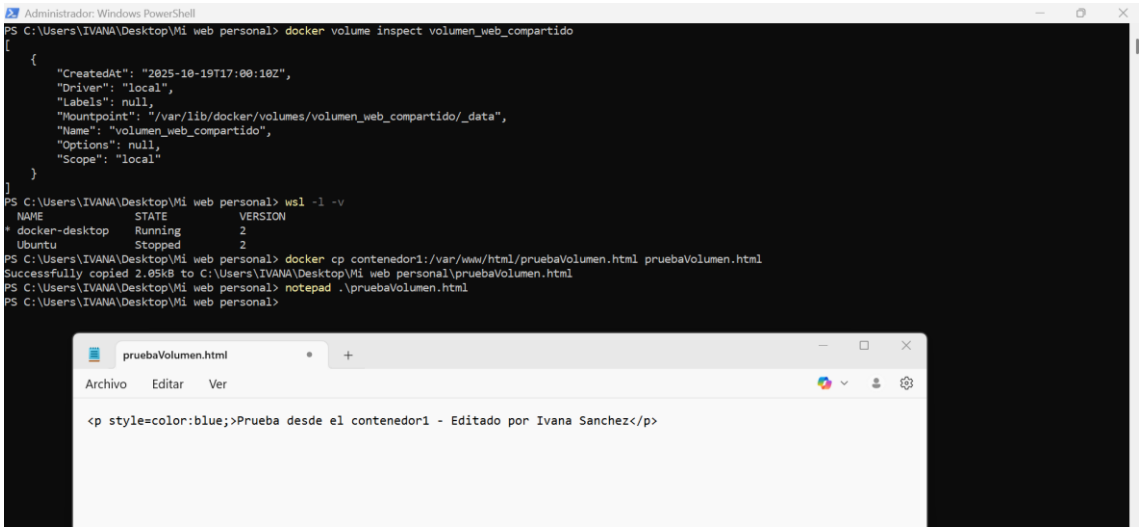
¿Que aparece en el otro contenedor? ¿Existe en la siguiente ruta http://localhost:8081/pruebaVolumen.html?.

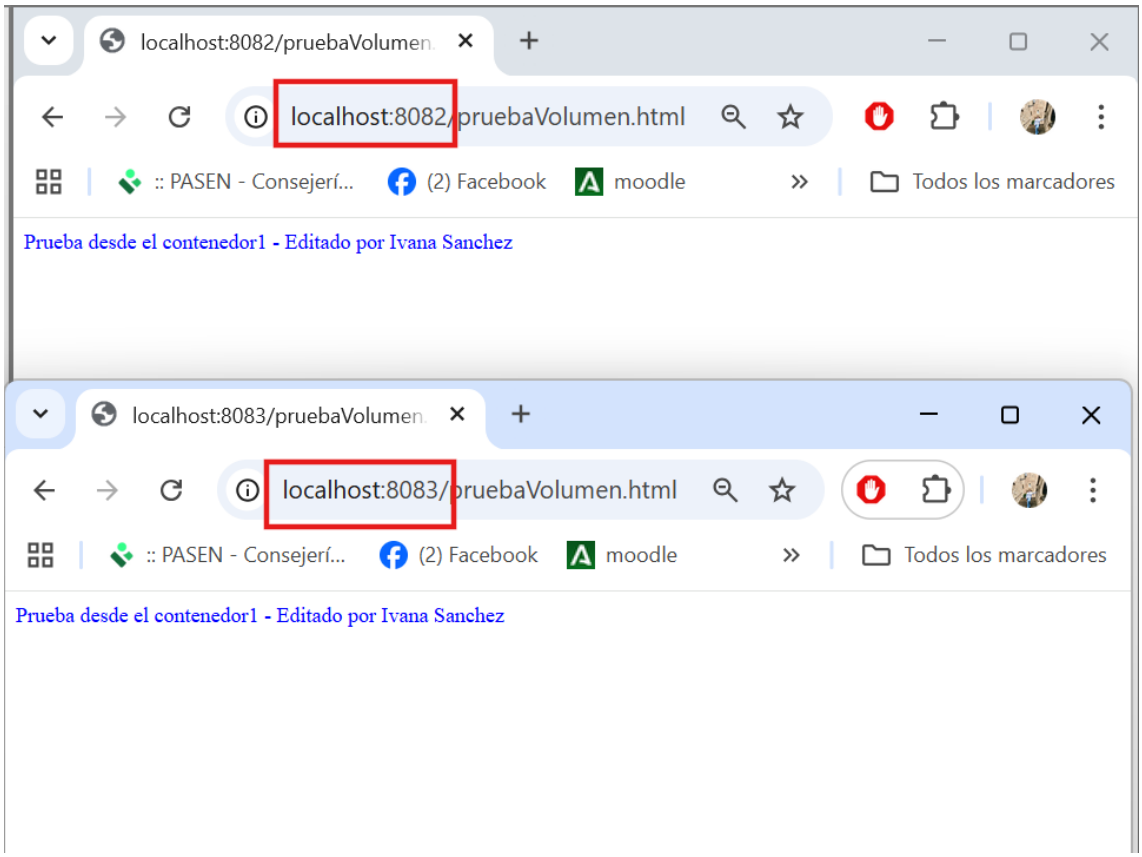
Ambas rutas (contenedor1 y contenedor2) están leyendo los datos del mismo volumen: *volumen-compartido*, el mensaje que he creado en el contenedor1 →*Prueba desde el contenedor 1.*

Edita el fichero desde el host y pon tu nombre, observa los cambios en ambos contenedores

Como utilizamos WSL y no se puede acceder directamente desde el explorador del archivos al html, copiaré el archivo *pruebaVolumen.html* desde el contenedor al escritorio de mi proyecto. Desde ahí lo abrimos y modificamos con Notepad, guardamos y lo copiamos de nuevo al contenedor.
Desde la ubicación de mi carpeta de la tarea:

1- **docker volume inspect volumen_web_compartido**
2- **wsl -l -v**
3- **docker cp contenedor1:/var/www/html/pruebaVolumen.html pruebaVolumen.html**
4- **notepad .\pruebaVolumen.html**
5- **docker cp pruebaVolumen.html contenedor1:/var/www/html/pruebaVolumen.html**

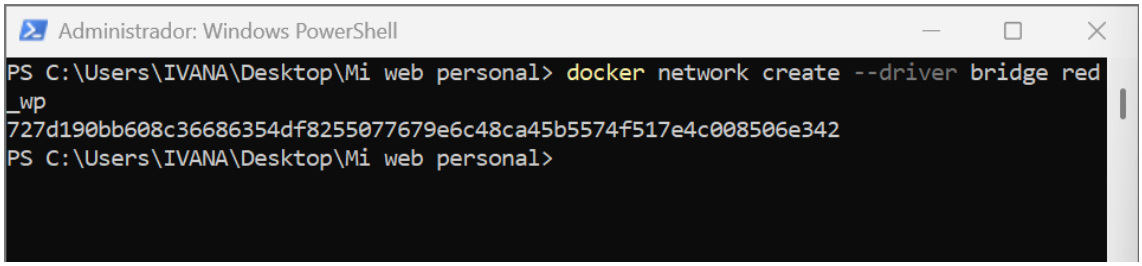

Administrador: Windows PowerShell
PS C:\Users\IVANA\Desktop\Mi web personal> docker volume inspect volumen_web_compartido
[
{
"CreatedAt": "2025-10-19T17:00:10Z",
"Driver": "local",
"Labels": null,
"Mountpoint": "/var/lib/docker/volumes/volumen_web_compartido/_data",
"Name": "volumen_web_compartido",
"Options": null,
"Scope": "local"
}
]
PS C:\Users\IVANA\Desktop\Mi web personal> wsl -l -v
NAME STATE VERSION
* docker-desktop Running 2
Ubuntu Stopped 2
PS C:\Users\IVANA\Desktop\Mi web personal> docker cp contenedor1:/var/www/html/pruebaVolumen.html pruebaVolumen.html
Successfully copied 2.05kB to C:\Users\IVANA\Desktop\Mi web personal\pruebaVolumen.html
PS C:\Users\IVANA\Desktop\Mi web personal> notepad .\pruebaVolumen.html
PS C:\Users\IVANA\Desktop\Mi web personal>
pruebaVolumen.html
Archivo Editar Ver
Prueba desde el contenedor1 - Editado por Ivana Sanchez


localhost:8082/pruebaVolumen
localhost:8082/pruebaVolumen.html
:: PASEN - Consejerí...
(2) Facebook
moodle
Todos los marcadores
Prueba desde el contenedor1 - Editado por Ivana Sanchez

localhost:8083/pruebaVolumen
localhost:8083/pruebaVolumen.html
:: PASEN - Consejerí...
(2) Facebook
moodle
Todos los marcadores
Prueba desde el contenedor1 - Editado por Ivana Sanchez

6. Vamos a conectar dos contenedores en red, para ello, crea una red de tipo bridge y a continuación crea los siguientes dos contenedores utilizando la red creada anteriormente.

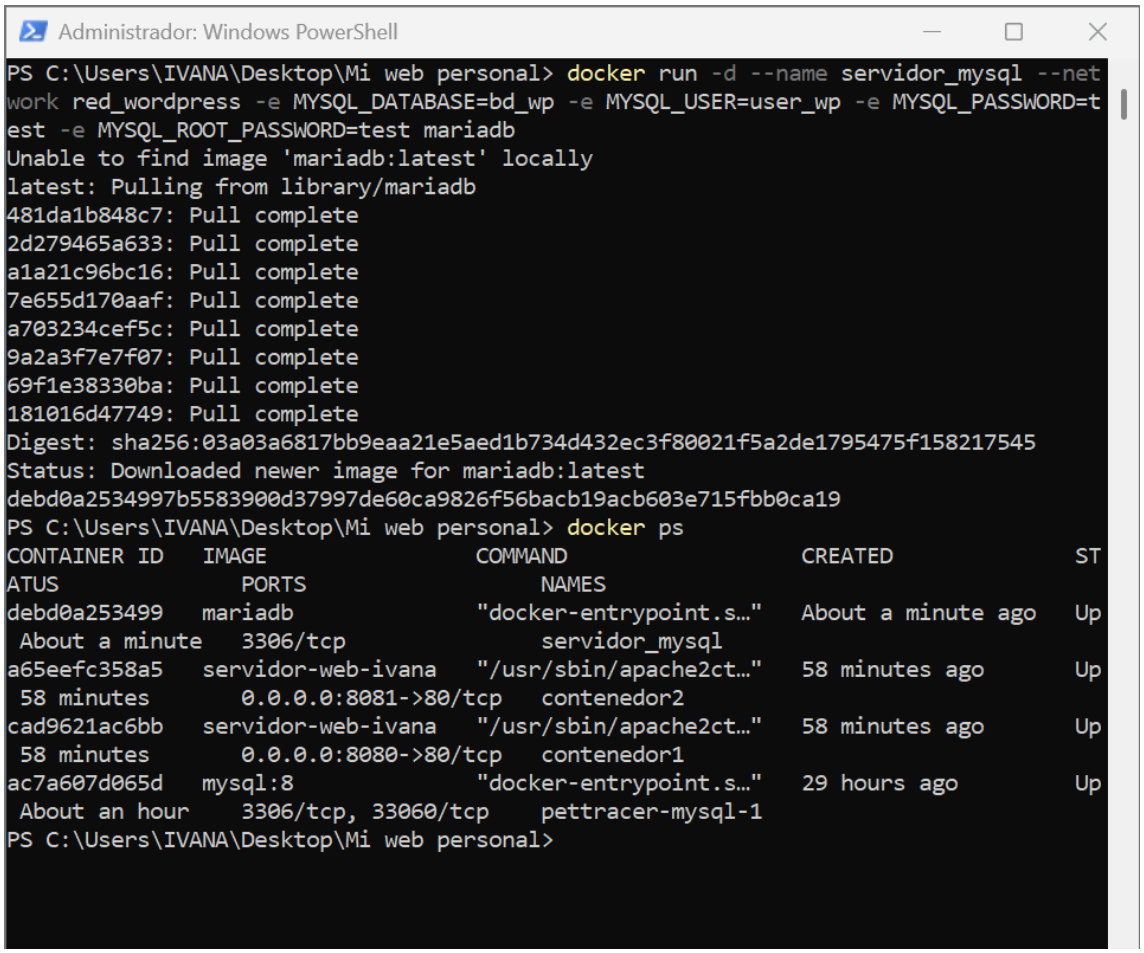
docker network create --driver bridge red_wp



El primer contenedor será una base de datos mariadb y lo llamaremos servidor_mysql:

docker run -d --name servidor_mysql --network XXXX -e MYSQL_DATABASE=bd_wp -e MYSQL_USER=user_wp -e MYSQL_PASSWORD=test -e MYSQL_ROOT_PASSWORD=test mariadb

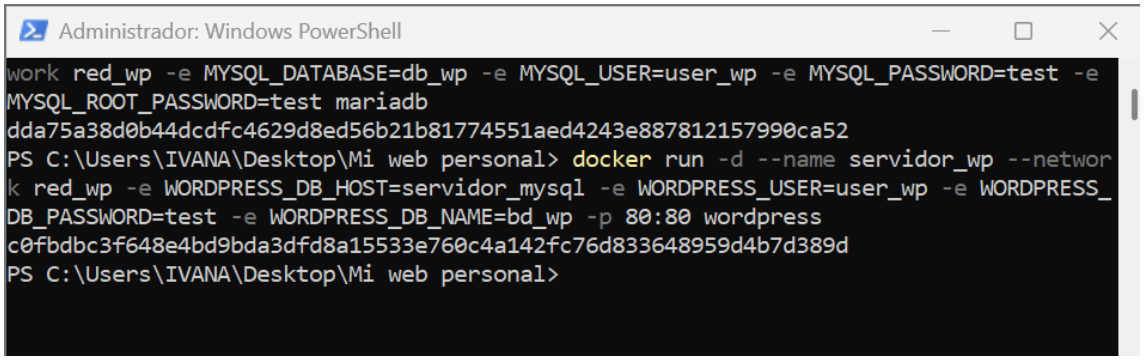
docker run -d –name servidor_mysql –network red_wp -e MYSQL_DATABASE=db_wp -e MYSQL_USER=user_wp -e MYSQL_PASSWORD=test -e MYSQL_ROOT_PASSWORD=test mariadb



A continuación vamos a crear un contenedor con el nombre servidor_wp, a partir de la imagen wordpress, conectada a la misma red y con las variables de entorno necesarias:

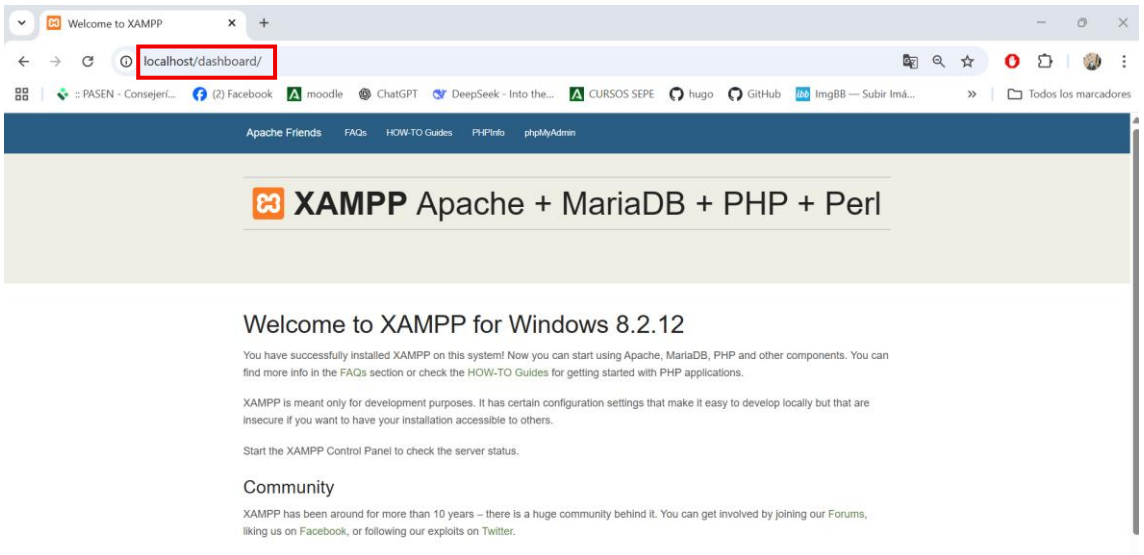
docker run -d --name servidor_wp --network XXXX -e WORDPRESS_DB_HOST=servidor_mysql -e WORDPRESS_DB_USER=user_wp
-e WORDPRESS_DB_PASSWORD=test -e WORDPRESS_DB_NAME=bd_wp -p 80:80 wordpress

docker run -d –name servidor_wp –network red_w -e WORDPRESS_DB_HOST=servidor_mysql -e WORDPRESS_DB_USER=user_wp -e WORDPRESS_DB_PASSWORD=test -e WORDPRESS_DB_NAME=bd_wp -p 80:80 wordpress



Accede a la ip del servidor docker y comprueba la instalación de wordpress. La variable WORDPRESS_DB_HOST hace referencia a la IP donde está la base de datos mysql ¿Por qué crees que hemos podido poner “servidor_mysql” en la variable WORDPRESS_DB_HOST?

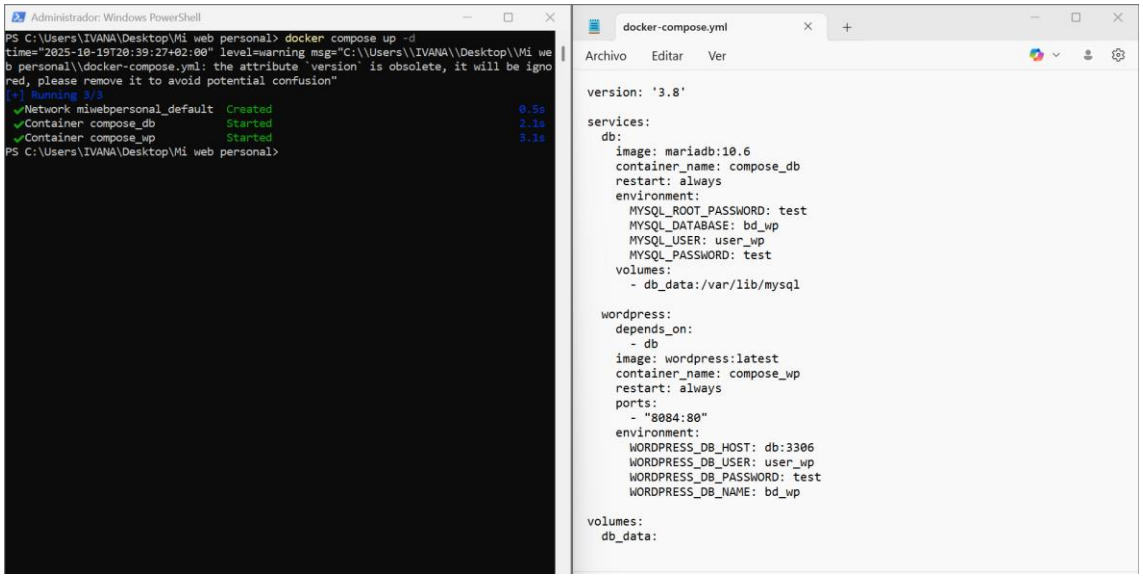
Se puede usar el nombre del contenedor (servidor_mysql) en vez de su IP porque, al conectar ambos contenedores (servidor_msql y servidor_wp) a la misma red de tipo puente (red_wordpress), Docker configura automáticamente un servicio interno de DNS. Esto permite que los contenedores dentro de esa red se comuniquen usando el nombre que le he dado. Esto hace que la configuración sea más flexible y fácil de usar, ya que las IP internas de Docker son dinámicas.



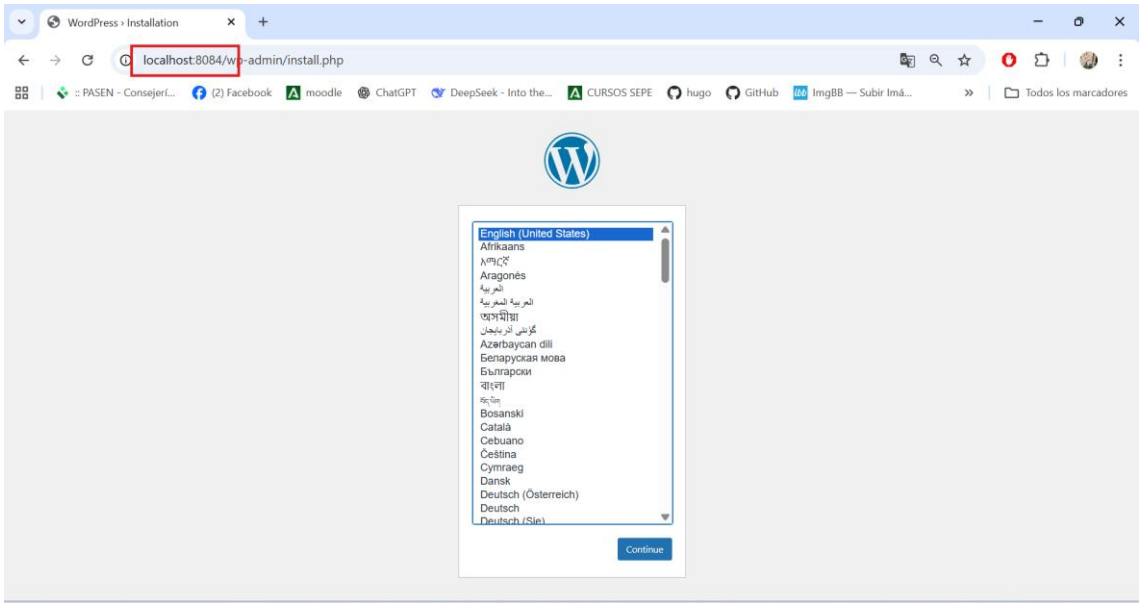
7. Buscar en docker hub la imagen de wordpress y cómo se instala con docker-compose. Generar el fichero correspondiente para arrancarlo en vuestras máquinas.

notepad docker-compose.yml

Para levantar el contendor utilizaremos el comando: ***docker compose up -d***



En el navegador ponemos el puerto que le hemos dado a WordPress en el fichero de docker-compose.yml →8084



¿Qué servicios están siendo definidos en este archivo de configuración?

Se definen 2 servicios: db para la base de datos y worpress para la aplicación web

¿Qué puertos están siendo mapeados para el servicio de Wordpress?

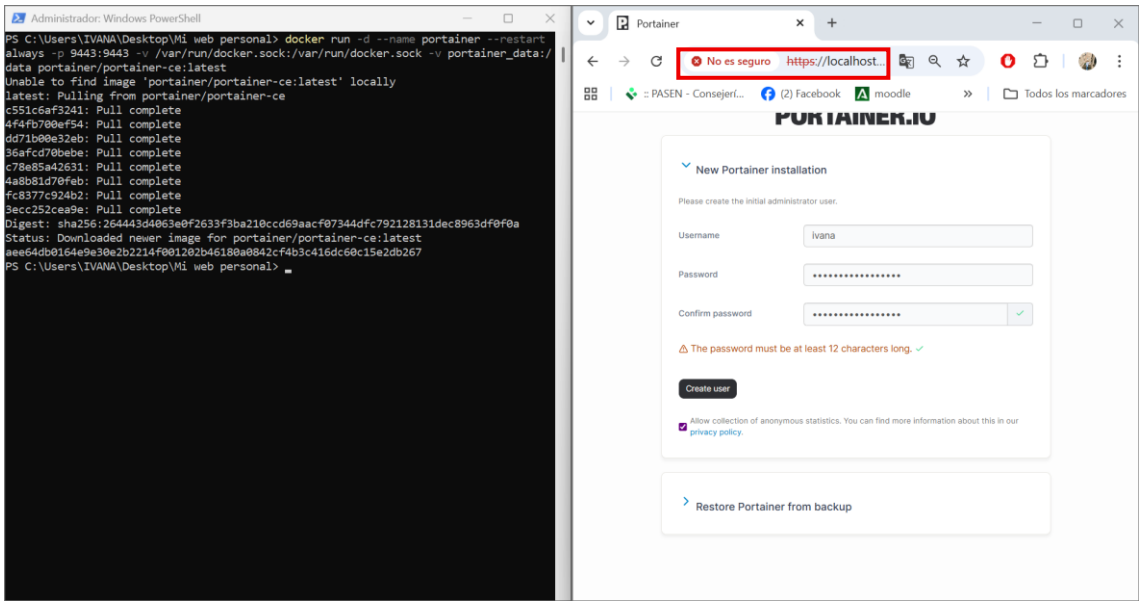
El puerto interno del contenedor (80) se está mapeando al puerto 8082 de mi pc para no interferir con el puerto 80 que ya se está usando.

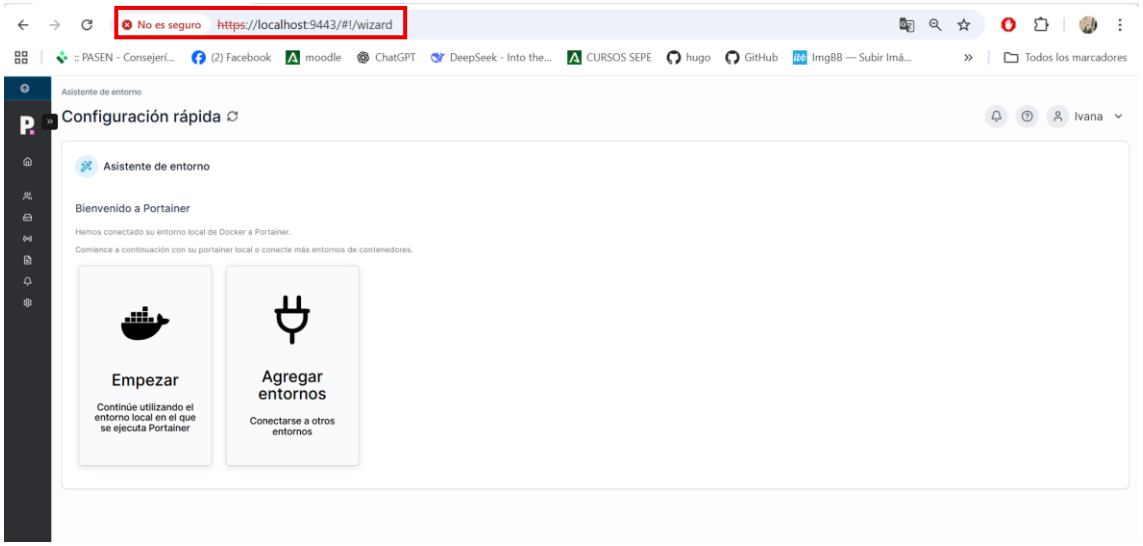
¿Qué significa el atributo "restart: always" en la definición de ambos servicios?
Que si un contenedor se detiene por cualquier razón, Docker Compose intentará reiniciarlo automáticamente.
¿Qué imagen está siendo utilizada para el servicio de base de datos MySQL?
La imagen de mariadb:10.6

8. Buscar en docker hub la aplicación Portainer y levantar un contenedor. Investigar sobre su funcionamiento, ¿Para que se utiliza esta aplicación?

Portainer es una herramienta de gestión gráfica para Docker. Su principal función es simplificar la administración del entorno Docker. Actúa como un panel de control web que permite a los usuario gestionar, monitorear y desplegar contenedores, imágenes, volúmenes y redes mediante una interfaz visual, eliminando el uso de los comandos.

docker run -d –name portainer –restart always -p 9443:9443 -v /var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data portainer/portainer-ce:latest




No es seguro
https://localhost:9443/#!/wizard
:: PASEN - Consejerí...
(2) Facebook
moodle
ChatGPT
DeepSeek - Into the...
CURSOS SEPE
hugo
GitHub
ImgBB — Subir Imá...
Todos los marcadores
Asistente de entorno
Configuración rápida
Ivana
Asistente de entorno
Bienvenido a Portainer
Hemos conectado su entorno local de Docker a Portainer.
Comience a continuación con su portainer local o conecte más entornos de contenedores.
Empezar
Continúe utilizando el entorno local en el que se ejecuta Portainer
Agregar entornos
Conectarse a otros entornos