

# **UNIVERSIDAD DE LA SIERRA SUR**



**Licenciatura en Informática**

**Priemer parcial**

**Ejercicio Practico Integral: Docker**

**Tecnologías Web II**

**Grupo: 706**

**Profesor: M.T.I.E. Irving Ulises Hernández Miguel**

**Alumno: Jorge López López**

## **Indice**

|                                  |    |
|----------------------------------|----|
| 1. Introducción.....             | 3  |
| 2. Desarrollo de evidencias..... | 4  |
| Parte 1.....                     | 4  |
| Parte 2.....                     | 5  |
| Parte 3.....                     | 6  |
| Parte 4.....                     | 7  |
| Parte 5.....                     | 8  |
| Preguntas de reflexión.....      | 13 |
| 3. Conclusión.....               | 14 |

# 1. Introducción

El presente documento detalla la ejecución y los resultados del "Ejercicio Práctico Integral: Docker - Verificación de Conceptos". El objetivo principal de esta práctica fue aplicar y verificar la comprensión de los conceptos fundamentales de la plataforma Docker, tal como se estipuló en el documento guía.

A lo largo de este reporte, se presentarán las evidencias (capturas de pantalla) de la realización de cada una de las cinco partes del ejercicio. Estas secciones cubren el ciclo de vida completo de la gestión de contenedores, incluyendo:

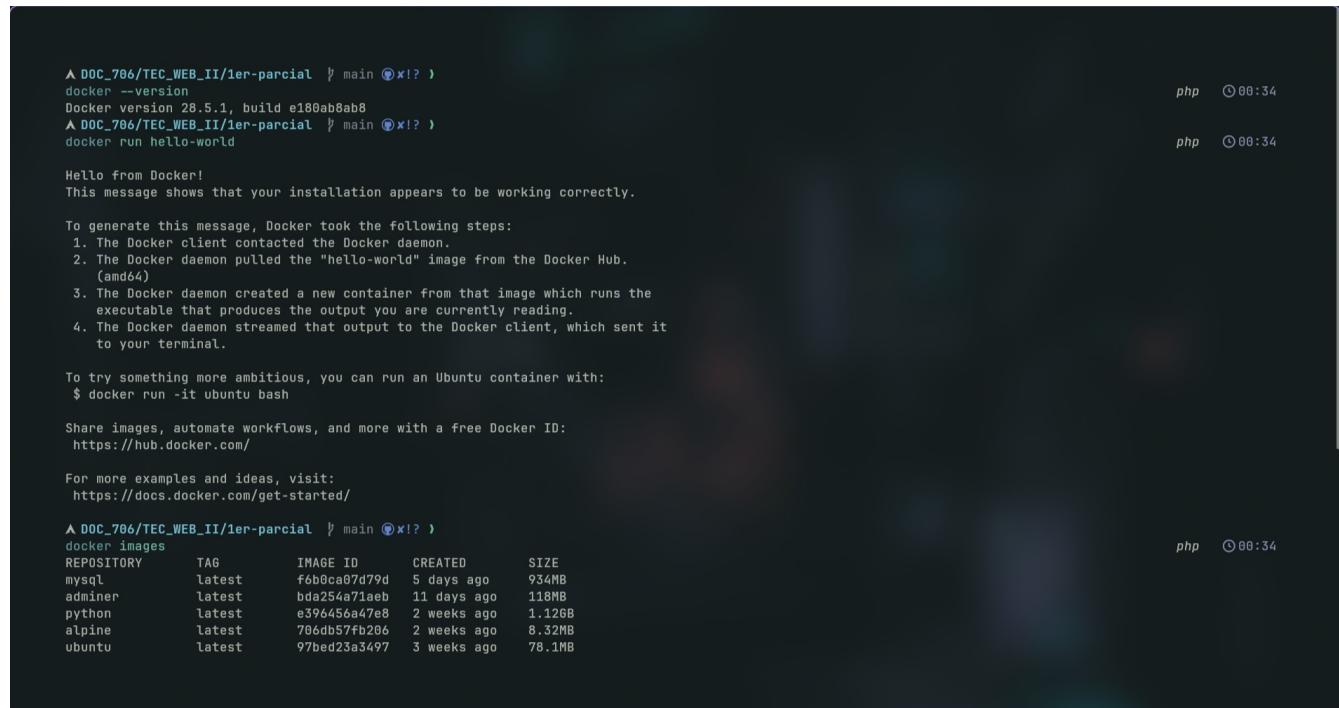
- Parte 1: La verificación de la instalación y comandos básicos.
- Parte 2: La gestión de contenedores interactivos, su ciclo de vida y el uso de docker exec.
- Parte 3: El uso de volúmenes para la persistencia de datos, demostrando que los datos sobreviven a la eliminación del contenedor.
- Parte 4: La configuración de redes personalizadas para la comunicación entre contenedores.
- Parte 5: El despliegue de una aplicación web multicontenedor (PHP y MySQL) en una red compartida.

Finalmente, el reporte incluye las respuestas a las preguntas de reflexión solicitadas, con el fin de demostrar el análisis y la comprensión de los beneficios de esta tecnología en entornos de desarrollo y producción.

## 2. Desarrollo de evidencias

A continuación, se presentan las capturas de pantalla solicitadas como evidencia de la correcta ejecución de cada paso de la práctica.

### Parte 1



```
A DOC_706/TEC_WEB_II/1er-parcial ~ main (A) x!?
docker --version
Docker version 28.5.1, build e180ab8ab8
A DOC_706/TEC_WEB_II/1er-parcial ~ main (A) x!?
docker run hello-world
Hello from Docker!
This message shows that your installation appears to be working correctly.

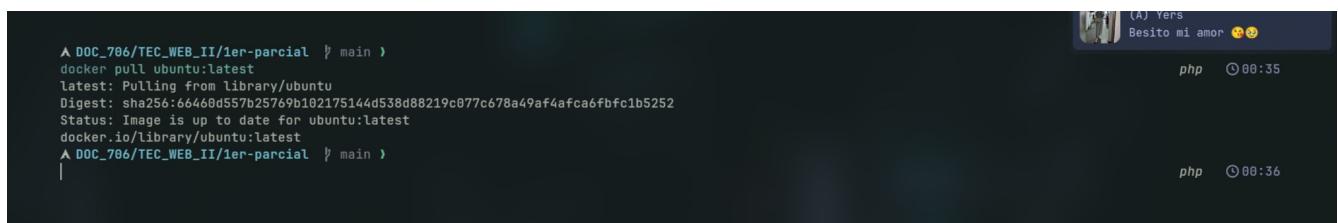
To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the 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/
A DOC_706/TEC_WEB_II/1er-parcial ~ main (A) x!?
docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
mysql          latest    f0b0ca0a7d79d  5 days ago   934MB
adminer        latest    bda254a71aeb  11 days ago  118MB
python          latest    e396456a47e8  2 weeks ago  1.12GB
alpine          latest    786db57fb206  2 weeks ago  8.32MB
ubuntu          latest    97bed23a3497  3 weeks ago  78.1MB
php      00:34
php      00:34
```

Figura 1: Verificación de instalación.



```
A DOC_706/TEC_WEB_II/1er-parcial ~ main
docker pull ubuntu:latest
latest: Pulling from library/ubuntu
Digest: sha256:664460d557b25769b102175144d538d88219c077c678a49af4afca6fbfc1b5252
Status: Image is up to date for ubuntu:latest
docker.io/library/ubuntu:latest
A DOC_706/TEC_WEB_II/1er-parcial ~ main
|
```

(A) Yers  
Besito mi amor 😊

```
php      00:35
php      00:36
```

Figura 2: Comandos básicos.

## Parte 2

```
A DOC_706/TEC_WEB_II/1er-parcial ¶ main )
docker run -it --name mi-ubuntu ubuntu /bin/bash
root@25efbbce91a9:/# ls -la
total 56
drwxr-xr-x  1 root root 4096 Oct 27 06:38 .
drwxr-xr-x  1 root root 4096 Oct 27 06:38 ..
-rwxr-xr-x  1 root root    0 Oct 27 06:38 .dockercfg
lrwxrwxrwx  1 root root    7 Apr 22 2024 bin → usr/bin
drwxr-xr-x  2 root root 4096 Apr 22 2024 boot
drwxr-xr-x  5 root root 360 Oct 27 06:38 dev
drwxr-xr-x  1 root root 4096 Oct 27 06:38 etc
drwxr-xr-x  3 root root 4096 Oct 27 02:10 home
lrwxrwxrwx  1 root root    7 Apr 22 2024 lib → usr/lib
lrwxrwxrwx  1 root root    9 Apr 22 2024 lib64 → usr/lib64
drwxr-xr-x  2 root root 4096 Oct 1 02:03 media
drwxr-xr-x  2 root root 4096 Oct 1 02:03 mnt
drwxr-xr-x  2 root root 4096 Oct 1 02:03 opt
dr-xr-xr-x  395 root root    0 Oct 27 06:38 proc
drwx----- 2 root root 4096 Oct 1 02:09 root
drwxr-xr-x  4 root root 4096 Oct 1 02:10 run
lrwxrwxrwx  1 root root    8 Apr 22 2024 sbin → usr/sbin
drwxr-xr-x  2 root root 4096 Oct 1 02:03 srv
dr-xr-xr-x  13 root root    0 Oct 27 03:09 sys
drwxrwxrwt  2 root root 4096 Oct 1 02:09 tmp
drwxr-xr-x  12 root root 4096 Oct 1 02:03 usr
drwxr-xr-x  11 root root 4096 Oct 1 02:09 var
root@25efbbce91a9:/# pwd
/
root@25efbbce91a9:/# echo "Hola desde Docker" > saludo.txt
root@25efbbce91a9:/# cat saludo.txt
Hola desde Docker
root@25efbbce91a9:/# exit
exit
A DOC_706/TEC_WEB_II/1er-parcial ¶ main )
|                                     php  ① 00:39
```

Figura 3: Crear y acceder a un contenedor interactivo.

```
A DOC_706/TEC_WEB_II/1er-parcial ¶ main )
docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS               NAMES
25efbbce91a9        ubuntu              "/bin/bash"         About a minute ago   Exited (0) 31 seconds ago
5c5555d3ba31        hello-world        "/hello"            5 minutes ago      Exited (0) 5 minutes ago
ed5c1cf41d78        mysql:8.1          "docker-entrypoint.s..."  23 minutes ago     Up 23 minutes           3306/tcp, 33060/tcp   bold_ellis
9cbc3b630a76        alpine              "/bin/sh"           30 minutes ago     Exited (137) 24 minutes ago
d17610b51a5c        python              "python3"          36 minutes ago     Exited (137) 33 minutes ago
926c8ccb5703        php:8.1-apache     "docker-php-entrypoi..."  51 minutes ago     Exited (0) 33 minutes ago
0dd13773eefdf       adminer             "entrypoint.sh docke..."  About an hour ago   Exited (0) 59 minutes ago
b8ec98201466        mysql:latest       "docker-entrypoint.s..."  About an hour ago   Exited (0) 59 minutes ago
315ee9ad6b5c        alpine              "sh"                About an hour ago   Exited (137) 59 minutes ago
9cc738d9eca9        alpine              "sh"                About an hour ago   Exited (137) 59 minutes ago
b05844f4b1c5        mysql              "docker-entrypoint.s..."  2 hours ago        Exited (0) 59 minutes ago
bc7b0eb8842        dpage/pgadmin4   "/entrypoint.sh"      2 months ago       Exited (0) 2 months ago
A DOC_706/TEC_WEB_II/1er-parcial ¶ main )
docker start mi-ubuntu
mi-ubuntu
A DOC_706/TEC_WEB_II/1er-parcial ¶ main )
docker exec -it mi-ubuntu /bin/bash
root@25efbbce91a9:/# cat saludo.txt
Hola desde Docker
root@25efbbce91a9:/# exit
exit
A DOC_706/TEC_WEB_II/1er-parcial ¶ main )
|                                     php  ① 00:40
```

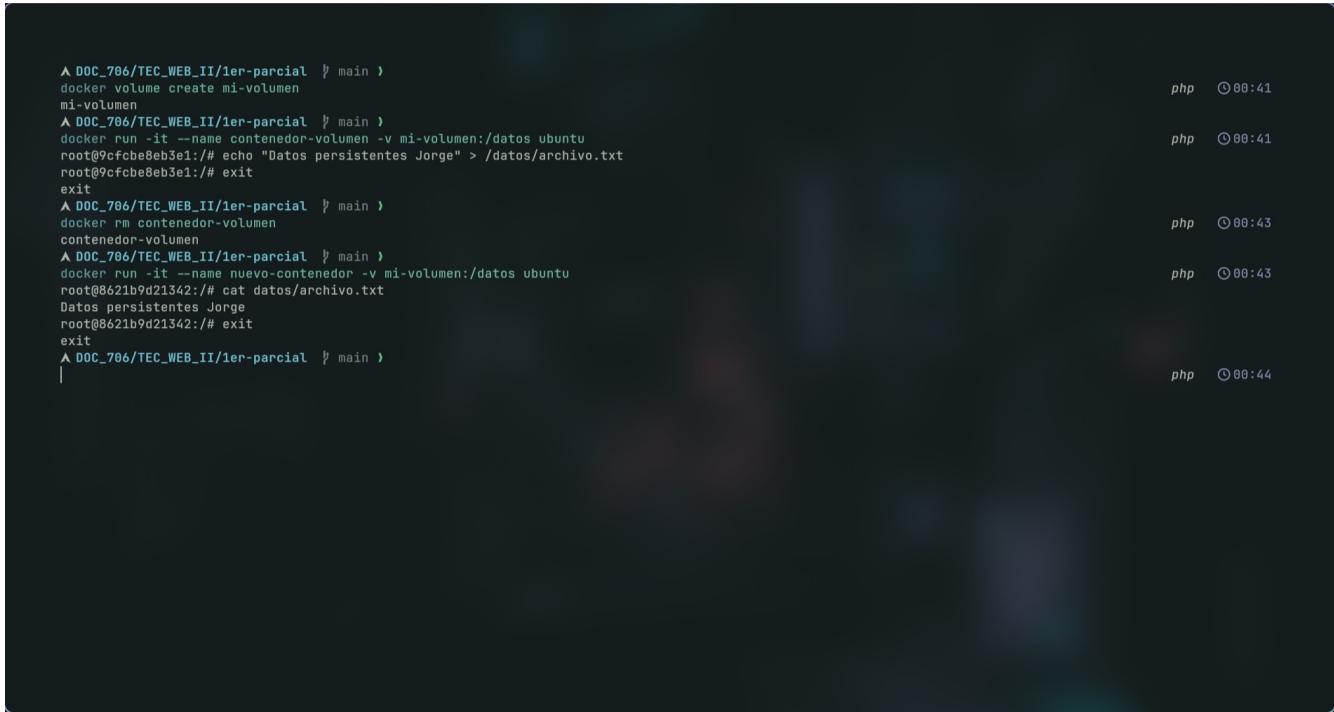
```
A DOC_706/TEC_WEB_II/1er-parcial ¶ main )
|                                     php  ① 00:40
```

```
A DOC_706/TEC_WEB_II/1er-parcial ¶ main )
|                                     php  ① 00:41
```

Figura 4: Gestión de estados del contenedor.

## Parte 3

A screenshot of a terminal window with a dark background. The terminal session shows the creation of a Docker volume named 'mi-volumen' and its subsequent use within a container. The session starts with creating the volume, followed by running a container with the volume mounted at '/datos'. Inside the container, a file 'archivo.txt' is created with the contents 'Datos persistentes Jorge'. The session ends with exiting the container. The terminal prompt is visible at the bottom.

```
A DOC_706/TEC_WEB_II/1er-parcial ↵ main )
docker volume create mi-volumen
php  ⏎ 00:41
mi-volumen
A DOC_706/TEC_WEB_II/1er-parcial ↵ main )
docker run -it --name contenedor-volumen -v mi-volumen:/datos ubuntu
root@9cfcb8eb3e1:# echo "Datos persistentes Jorge" > /datos/archivo.txt
root@9cfcb8eb3e1:# exit
exit
A DOC_706/TEC_WEB_II/1er-parcial ↵ main )
docker rm contenedor-volumen
contenedor-volumen
A DOC_706/TEC_WEB_II/1er-parcial ↵ main )
docker run -it --name nuevo-contenedor -v mi-volumen:/datos ubuntu
root@8621b9d21342:# cat datos/archivo.txt
Datos persistentes Jorge
root@8621b9d21342:# exit
exit
A DOC_706/TEC_WEB_II/1er-parcial ↵ main )
|  php  ⏎ 00:43
|  php  ⏎ 00:44
```

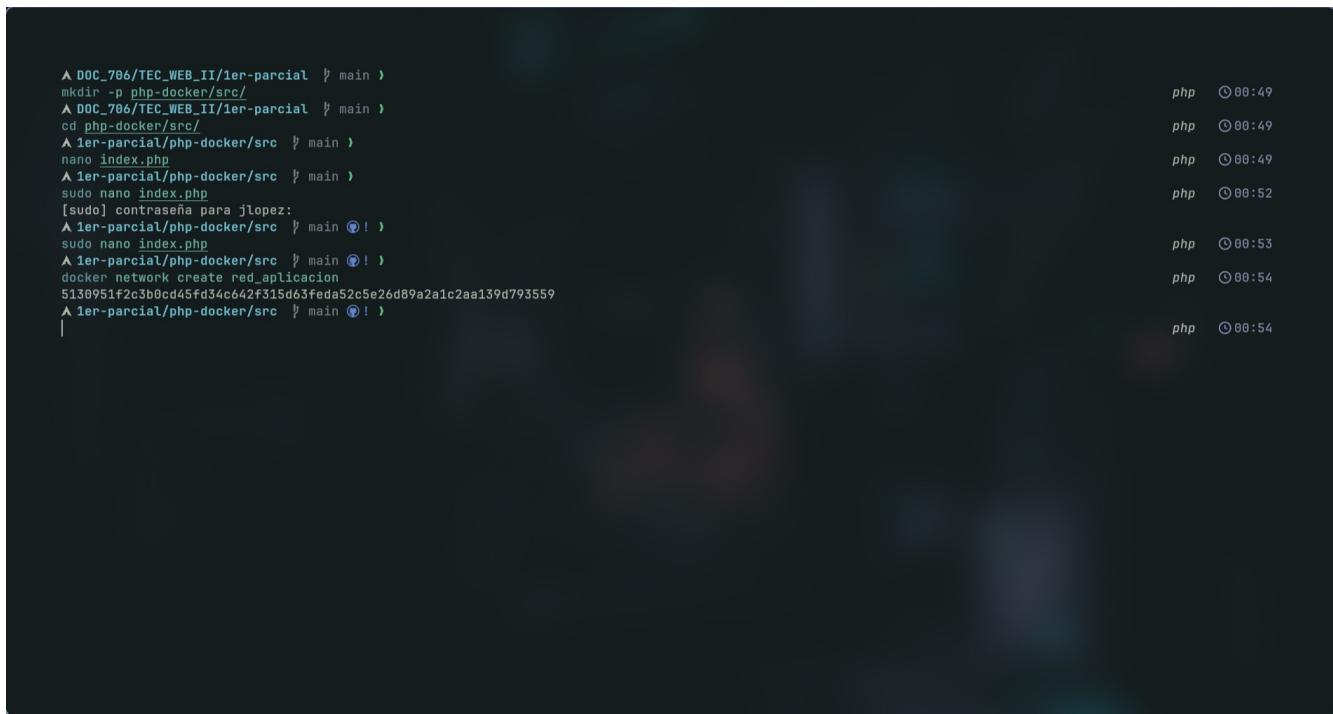
Figura 5: Crear y usar un volumen.

## Parte 4

```
A DOC_706/TEC_WEB_II/1er-parcial ¶ main 
docker network create mi-red
11daef1743d3650c2377c583188ea7fb982f7edeb9c2caf43577aba9bb3e1551
A DOC_706/TEC_WEB_II/1er-parcial ¶ main 
docker run -d --name servidor --network mi-red alpine sleep 3600
c5eb176fcc6b9e77a4b59ff14964e71649a244ae5f5edcd8a7c66f439949d7907d
A DOC_706/TEC_WEB_II/1er-parcial ¶ main 
docker run -it --name cliente --network mi-red alpine
/ # ping servidor
PING servidor (172.20.0.2): 56 data bytes
64 bytes from 172.20.0.2: seq=0 ttl=64 time=0.057 ms
64 bytes from 172.20.0.2: seq=1 ttl=64 time=0.043 ms
64 bytes from 172.20.0.2: seq=2 ttl=64 time=0.043 ms
64 bytes from 172.20.0.2: seq=3 ttl=64 time=0.037 ms
^C
--- servidor ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.037/0.045/0.057 ms
/ # exit
A DOC_706/TEC_WEB_II/1er-parcial ¶ main 
|
```

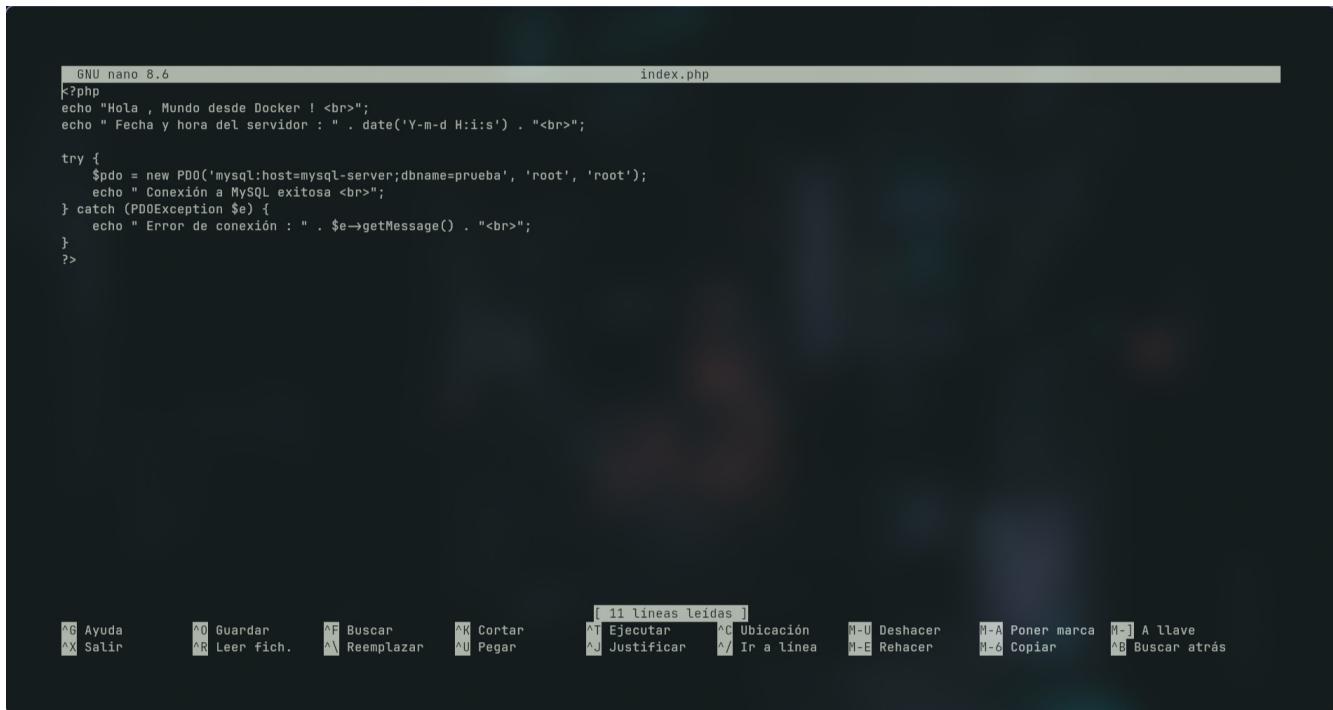
Figura 6: Crear red y ejecutar contenedores en la red.

## Parte 5



```
A DOC_706/TEC_WEB_II/1er-parcial ¶ main 
mkdir -p php-docker/src/
A DOC_706/TEC_WEB_II/1er-parcial ¶ main 
cd php-docker/src/
A 1er-parcial/php-docker/src ¶ main 
nano index.php
A 1er-parcial/php-docker/src ¶ main 
sudo nano index.php
[sudo] contraseña para jlopez:
A 1er-parcial/php-docker/src ¶ main (2) !
sudo nano index.php
A 1er-parcial/php-docker/src ¶ main (2) !
docker network create red_aplicacion
5130951f2c3b0cd45fd34c642f315d63fedaf52c5e26d89a2a1c2aa139d793559
A 1er-parcial/php-docker/src ¶ main (2) !
|                                     php  ① 00:49
|                                     php  ① 00:49
|                                     php  ① 00:49
|                                     php  ① 00:52
|                                     php  ① 00:53
|                                     php  ① 00:54
|                                     php  ① 00:54
```

Figura 7: Crear estructura del proyecto y crear red para la aplicación.



```
GNU nano 8.6                               index.php
<?php
echo "Hola , Mundo desde Docker ! <br>";
echo " Fecha y hora del servidor : " . date('Y-m-d H:i:s') . "<br>";

try {
    $pdo = new PDO('mysql:host=mysql-server;dbname=prueba', 'root', 'root');
    echo " Conexión a MySQL exitosa <br>";
} catch (PDOException $e) {
    echo " Error de conexión : " . $e->getMessage() . "<br>";
}
?>
```

[ 11 líneas leidas ]

^G Ayuda ^D Guardar ^F Buscar ^K Cortar ^T Ejecutar ^C Ubicación M-U Deshacer M-A Poner marca M-[ A llave
^X Salir ^R Leer fich. ^V Reemplazar ^U Pegar ^J Justificar ^/ Ir a línea M-B Rehacer M-C Copiar ^B Buscar atrás

Figura 8: Archivo index.php



```
A 1er-parcial/php-docker/src ⌂ main
docker run -d --name mysql-server \
--network red_aplicacion \
-e MYSQL_ROOT_PASSWORD=root \
-e MYSQL_DATABASE=prueba \
-v mysql-data:/var/lib/mysql \
mysql:latest
647c5035bb6e65516d319316d11ed8cc797a97dbf9af6adec537eaeb5531198c3
A 1er-parcial/php-docker/src ⌂ main
docker run -d --name php-app \
--network red_aplicacion \
-p 8080:80 \
-v $(pwd):/var/www/html \
php:8.1-apache
f61858be079ad8afbf50b29ae2add1a0106bba4ff21e74c2dd3d303908c16396
A 1er-parcial/php-docker/src ⌂ main
|
|
```

The screenshot shows a terminal window with several Docker commands being run. The first command starts a MySQL container named 'mysql-server' with specific environment variables and volumes. The second command starts a PHP application container named 'php-app' with port mapping and volume mounting. The third command returns to the directory 'src'. The terminal interface includes a battery status bar at the top right.

Figura 9: Ejecutar contenero MySQL y aplicación PHP.

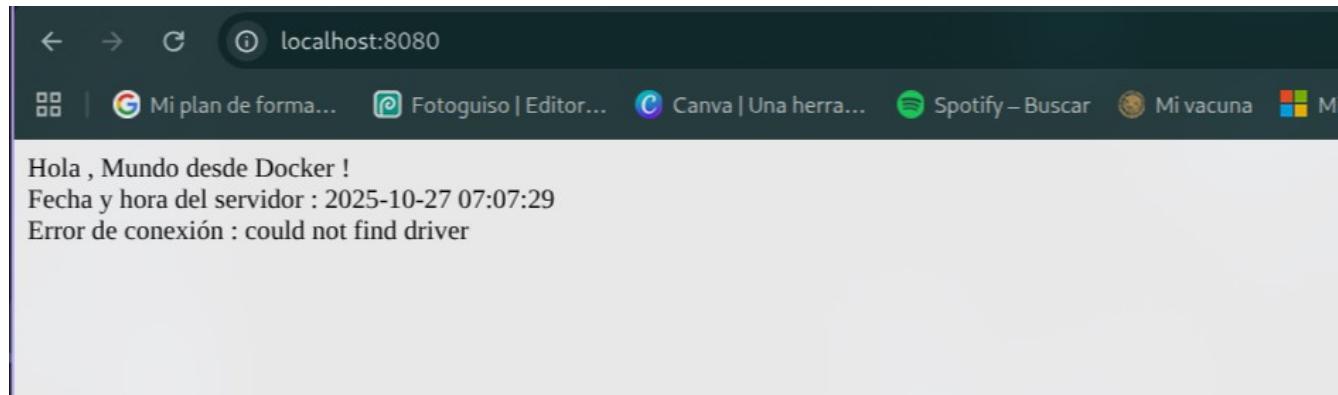


Figura 10: Problemas con conexión a MySQL.

## Solución

The screenshot shows a terminal window with a dark background. In the center, there is a nano text editor displaying a Dockerfile. The file content is as follows:

```
GNU nano 8.6
# Usa la imagen oficial de PHP 8.1 con Apache como base
FROM php:8.1-apache

# Instala la extensión de PHP para MySQL (PDO)
RUN docker-php-ext-install pdo_mysql
```

At the bottom of the terminal, there is a status bar with various keyboard shortcuts for nano, such as **^G Ayuda**, **^O Guardar**, **^F Buscar**, **^K Cortar**, **^T Ejecutar**, **^C Ubicación**, **^U Deshacer**, **M-A Poner marca**, **M-]**, **A llave**, **^X Salir**, **^R Leer fich.**, **^V Reemplazar**, **^U Pegar**, **^J Justificar**, **^I Ir a línea**, **M-B Rehacer**, **M-G Copiar**, and **^B Buscar atrás**.

Figura 11: Crear un archivo Dockerfile, con la extensión de PHP para MySQL.

The screenshot shows a terminal window with a dark background. It displays the output of a Docker build command. The logs show the following steps:

```
A 1er-parcial/php-docker/src  ⟳ main ⏺? 
docker stop php-app
docker rm php-app
php-app
php-app
A 1er-parcial/php-docker/src  ⟳ main ⏺? 
docker build -t php-mysql-app .
php-mysql-app
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
Install the buildx component to build images with BuildKit:
https://docs.docker.com/go/buildx

Sending build context to Docker daemon 3.072kB
Step 1/2 : FROM php:8.1-apache
--> f0de658f7931
Step 2/2 : RUN docker-php-ext-install pdo_mysql
--> Running in ad1bb3f5e17f
Configuring for:
PHP Api Version: 20210902
Zend Module Api No: 20210902
Zend Extension Api No: 420210902
checking for grep that handles long lines and -e... /usr/bin/grep
checking for egrep... /usr/bin/grep -E
checking for a sed that does not truncate output... /usr/bin/sed
checking for pkg-config... /usr/bin/pkg-config
checking pkg-config is at least version 0.9.0... yes
checking for cc... cc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether the compiler supports GNU C... yes
checking whether cc accepts -g... yes
checking for cc option to enable C11 features... none needed
checking how to run the C preprocessor... cc -E
```

Figura 12: Utilizar el Dockerfile para crear e inicializar el contenedor.

```
A ier-parcial/php-docker/src ⌂ main ⌂ ? 
docker run -d --name php-app \
--network red_aplicacion \
-p 8080:80 \
-v $(pwd):/var/www/html \
php-mysql-app
3ec77a0d0754c2ed1b324dc5654229214cecc3bd0323b3d75ee782ad19a9f7c6
A ier-parcial/php-docker/src ⌂ main ⌂ ? 
|                                     php ⌂ 01:12
|                                     php ⌂ 01:12
```

Figura 13: Ejecución de contenedor.

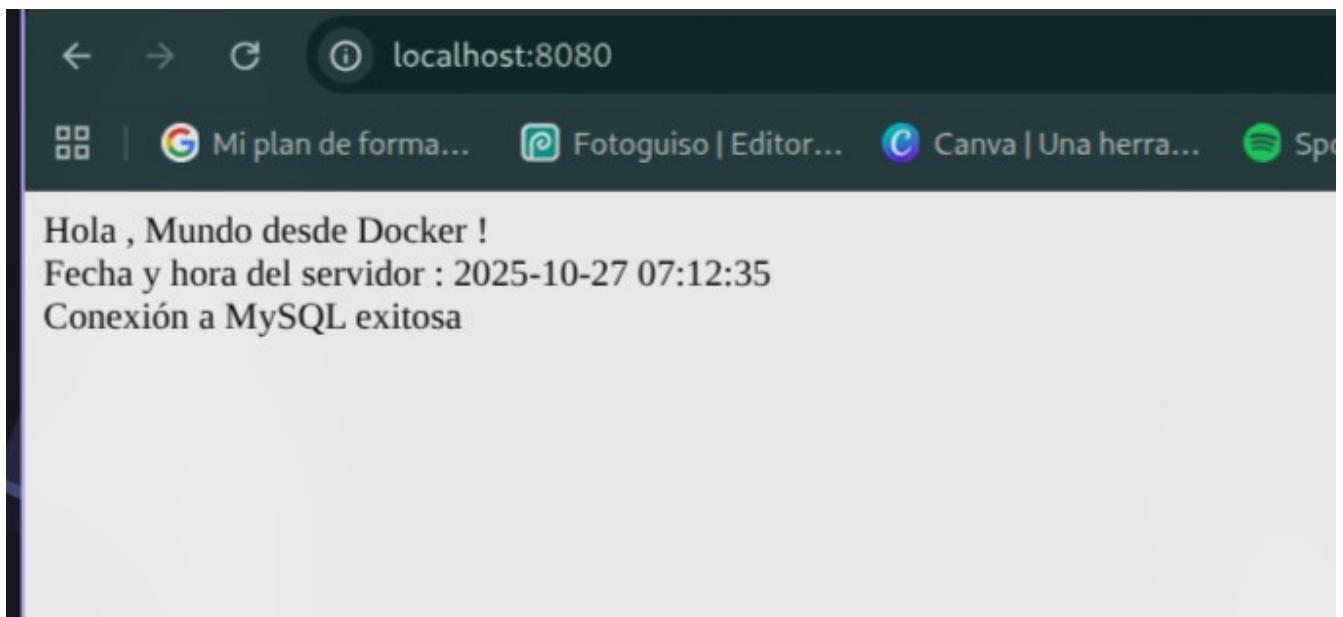


Figura 14: Fucionamiento correcoto de la aplicación.

## Preguntas de reflexión

### 1. ¿Qué ventajas observas al usar contenedores frente a máquinas virtuales?

La eficiencia y la portabilidad. Los contenedores son más ligeros y rápidos porque comparten el kernel del SO anfitrión, usando menos recursos. Las VMs, en cambio, virtualizan un SO completo .

### 2. ¿Cómo asegura Docker la consistencia entre entornos de desarrollo y producción?

A través de las Imágenes. Una imagen es una plantilla que empaqueta la aplicación con todas sus dependencias. La misma imagen se usa para crear contenedores idénticos en cualquier entorno, eliminando el problema de "funciona en mi máquina".

### 3. ¿Por qué es importante usar volúmenes en aplicaciones con bases de datos?

Para persistir los datos. Los contenedores son efímeros; si se eliminan, sus datos internos se pierden. Los volúmenes guardan la información (como la de una BD) de forma permanente fuera del contenedor, asegurando que los datos sobrevivan aunque el contenedor se reemplace.

### 4. ¿Qué beneficios ofrece el uso de redes personalizadas en Docker?

Aislamiento y comunicación por nombre. Crean una red controlada donde los contenedores pueden encontrarse y comunicarse usando sus nombres (ej. ping servidor ) en lugar de direcciones IP, lo cual es clave para aplicaciones multicontenedor .

### **3. Conclusión**

La ejecución de este ejercicio práctico integral ha sido fundamental para consolidar los conceptos teóricos de Docker y traducirlos en habilidades prácticas. A través de la realización de las cinco partes del ejercicio, se demostró con éxito la capacidad de instalar Docker, gestionar el ciclo de vida de los contenedores, asegurar la persistencia de datos críticos mediante volúmenes y establecer una comunicación de red aislada y funcional entre servicios.

El despliegue de la aplicación web PHP/MySQL, en particular, sirvió como una síntesis perfecta, integrando todos estos conceptos en un escenario que simula un entorno de producción real. Esta experiencia práctica confirma las ventajas de Docker en cuanto a portabilidad, eficiencia en el uso de recursos y, sobre todo, la consistencia que ofrece para el desarrollo y despliegue de aplicaciones modernas.