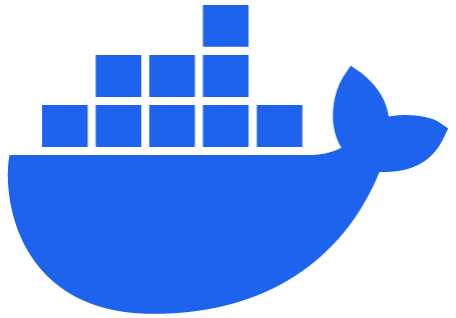
Introducción a

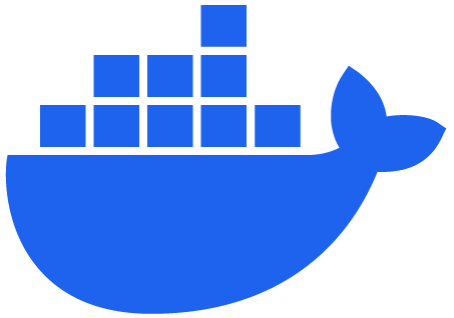
M.T.I.E. Irving Ulises Hernández Miguel

Contenido



- ¿Qué es Docker?
- ¿Por qué Docker?
- Docker vs Máquinas Virtuales
- Conceptos clave
- Funcionamiento Básico
- Casos de Uso
- Parte práctica – (Instalación y comandos básicos)

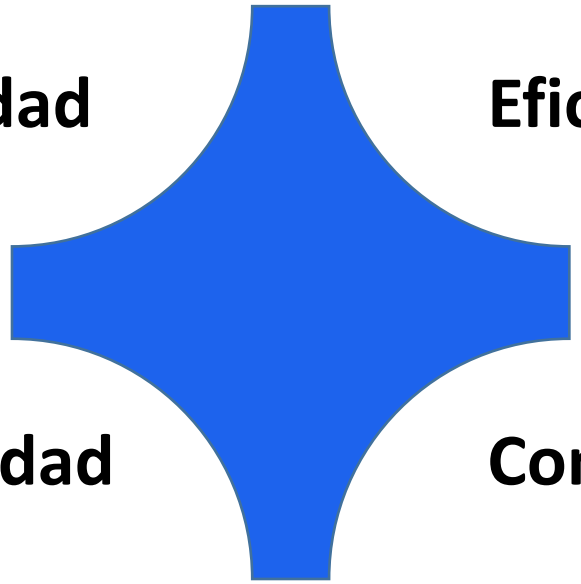
¿Qué es Docker?



Docker es una **plataforma** de software que permite a los desarrolladores crear, implementar y ejecutar aplicaciones dentro de contenedores.

Estos **contenedores** son unidades ligeras y portátiles que incluyen todo lo necesario para ejecutar una aplicación de manera consistente en cualquier entorno, desde la máquina del desarrollador hasta los servidores en producción.

¿Por qué Docker?



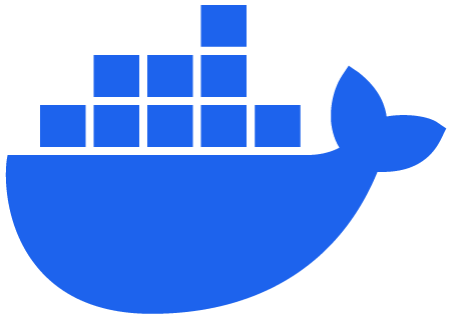
Portabilidad

Eficiencia

Escalabilidad

Consistencia

¿Por qué Docker?

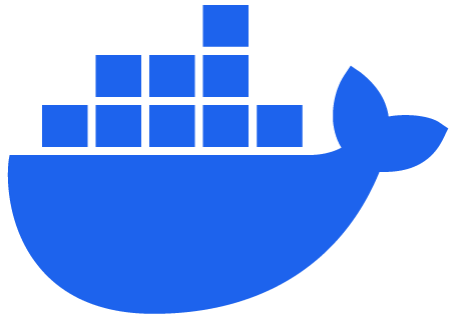


Portabilidad:

Los contenedores pueden ejecutarse en cualquier máquina con Docker instalado, independientemente de las configuraciones del sistema.

Ejemplo: Un contenedor que funciona en la computadora de un desarrollador funcionará igual en un servidor de producción.

¿Por qué Docker?

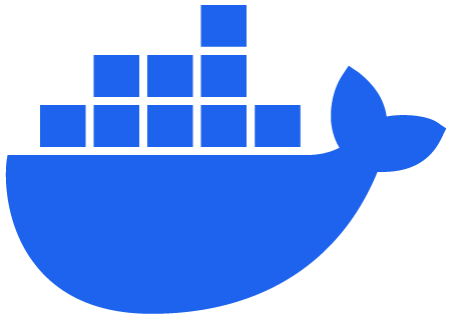


Eficiencia:

Los contenedores son ligeros y comparten el kernel del sistema operativo, reduciendo el uso de recursos en comparación con las máquinas virtuales.

Ejemplo: Múltiples contenedores pueden ejecutarse en una sola máquina sin consumir muchos recursos.

¿Por qué Docker?

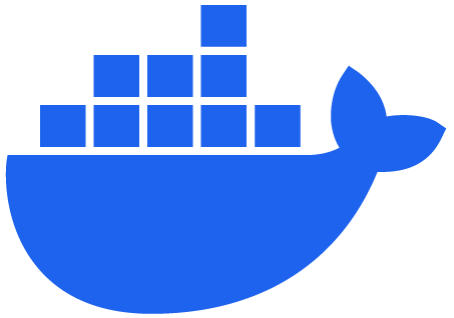


Escalabilidad:

Docker facilita el despliegue y la escalabilidad de aplicaciones, permitiendo la orquestación con herramientas como Kubernetes.

Ejemplo: Una aplicación puede escalar rápidamente de uno a muchos contenedores para manejar el aumento del tráfico.

¿Por qué Docker?

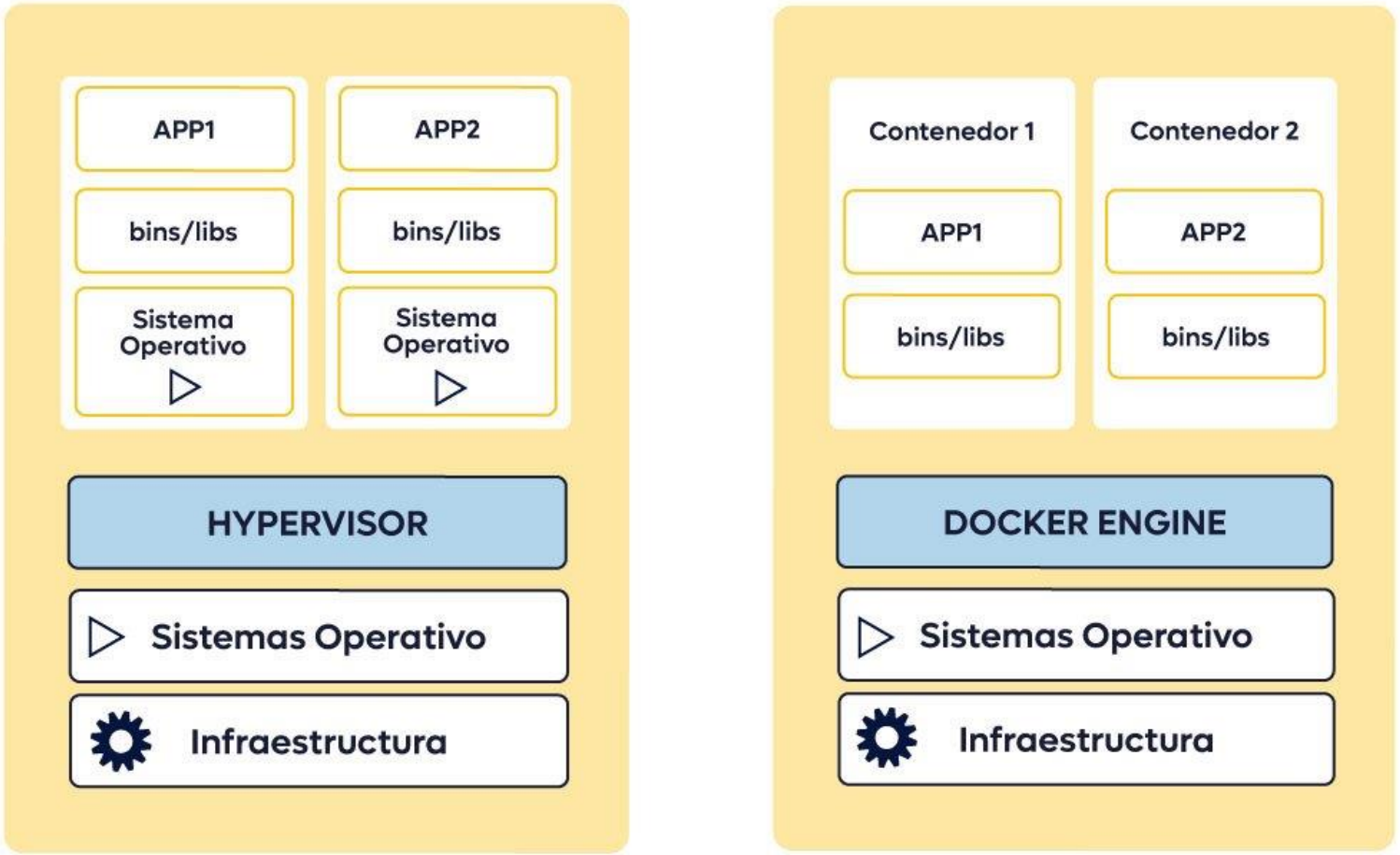


Consistencia:

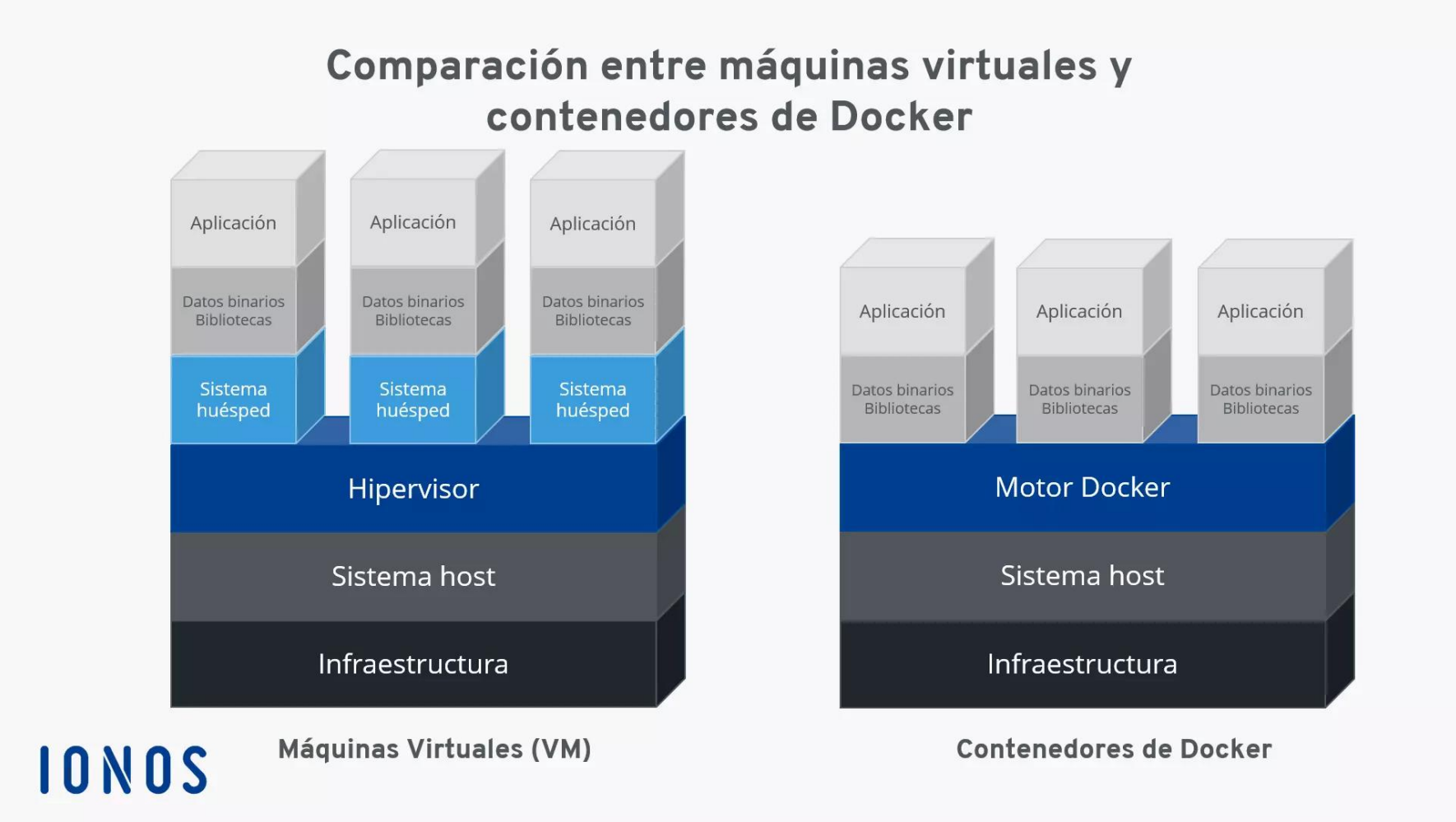
Garantiza que las aplicaciones funcionen de la misma manera en todos los entornos, eliminando el clásico problema de "funciona en mi máquina".

Ejemplo: Los desarrolladores pueden crear entornos de prueba que sean idénticos a los de producción.

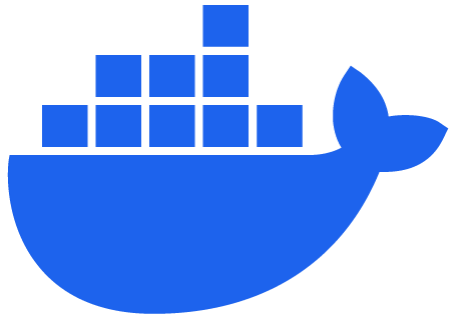
Docker vs Máquinas Virtuales



Docker vs Máquinas Virtuales

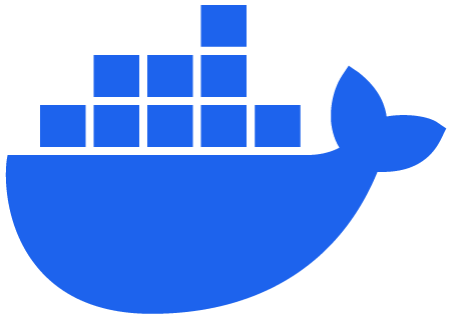


Conceptos clave



Contenedor	Imagen	Dockerfile	Docker Hub
Son instancias ejecutables de una imagen de Docker. Son ligeros y portátiles. *Ejemplo: Un contenedor que ejecuta una aplicación web con todas sus dependencias.*	Una plantilla de solo lectura utilizada para crear contenedores. Las imágenes se pueden versionar y reutilizar. *Ejemplo: Una imagen de Docker que contiene un servidor web Nginx configurado.*	Un archivo de texto con un conjunto de instrucciones para construir una imagen. Define qué se incluirá en la imagen y cómo se configurará. *Ejemplo: Un Dockerfile que especifica una imagen base de Ubuntu y las instrucciones para instalar Node.js.*	Un registro en línea donde los desarrolladores pueden almacenar y compartir imágenes de Docker. Hay imágenes oficiales y de la comunidad. *Ejemplo: Descargar una imagen de MySQL desde Docker Hub para usar en un proyecto.*

Funcionamiento básico

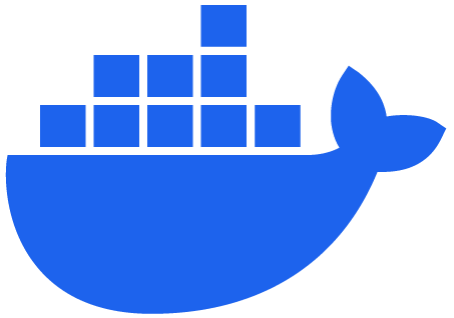


Construir una Imagen:

Utilizando un Dockerfile, se construye una imagen con el comando docker build.

Ejemplo: docker build -t myapp .

Funcionamiento básico

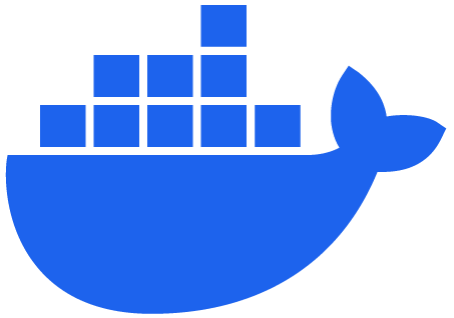


Implementar un Contenedor:

A partir de una imagen, se crea y se ejecuta un contenedor con el comando docker run.

Ejemplo: docker run -d -p 80:80 myapp

Funcionamiento básico



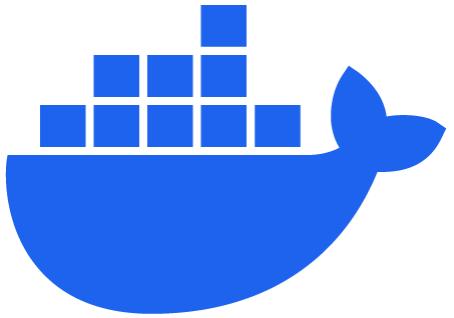
Gestionar Contenedores:

docker ps para listar contenedores en ejecución.

docker stop <container_id> para detener un contenedor.

docker start <container_id> para iniciar un contenedor detenido.

Casos de uso



Desarrollo y Pruebas:

Facilita la creación de entornos de desarrollo consistentes. Simplifica la prueba de aplicaciones en diferentes configuraciones.

Ejemplo: Desarrollar una aplicación en un contenedor que incluye todas las dependencias necesarias.

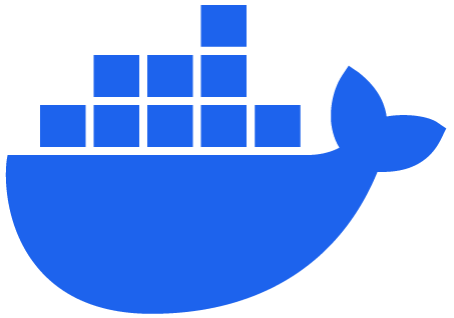
Casos de uso

Despliegue en Producción:

Simplifica el despliegue de aplicaciones en diferentes entornos. Facilita el escalado horizontal de servicios.

Ejemplo: Desplegar una aplicación web en múltiples contenedores para manejar más tráfico.

Casos de uso

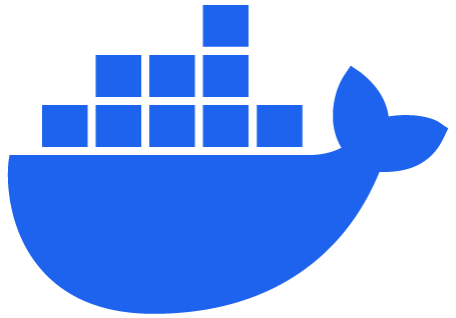


Microservicios:

Ideal para arquitecturas basadas en microservicios debido a su capacidad de aislar componentes.

Cada microservicio puede ejecutarse en su propio contenedor, gestionado de forma independiente.

Parte práctica – instalación – opción 1



Instalación en distros basadas en Debian:

```
sudo apt update
sudo apt install ca-certificates curl gnupg lsb-reléase
sudo mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
sudo chmod a+r /etc/apt/keyrings/docker.gpg
echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt update
sudo apt install docker-ce docker-ce-cli containerd.io
```

Parte práctica – instalación – opción 2

Después de actualizar el sistema.

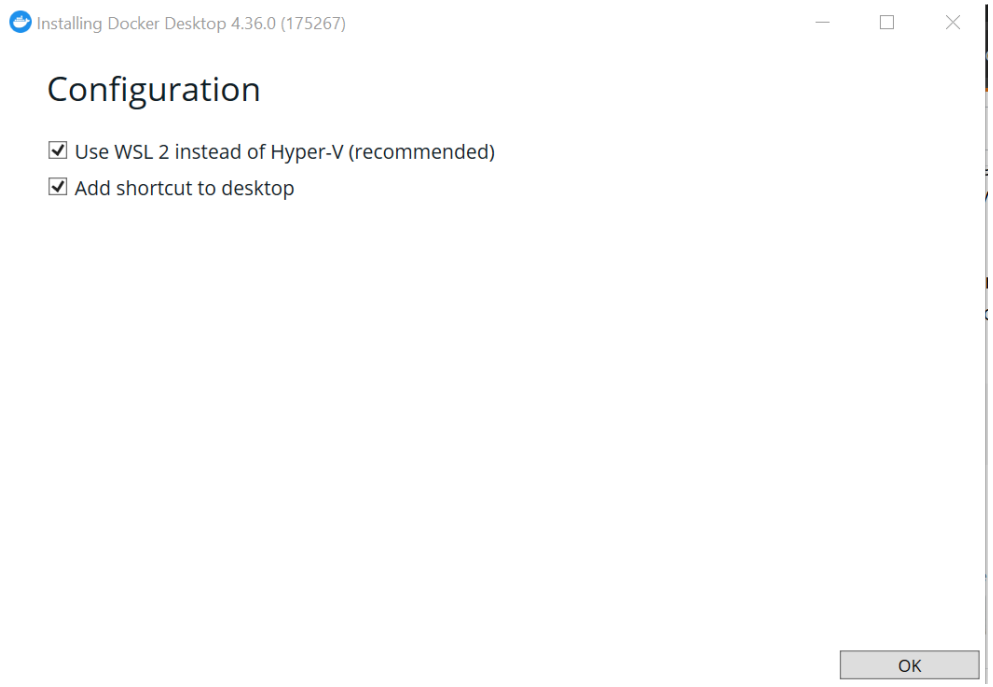
sudo apt update

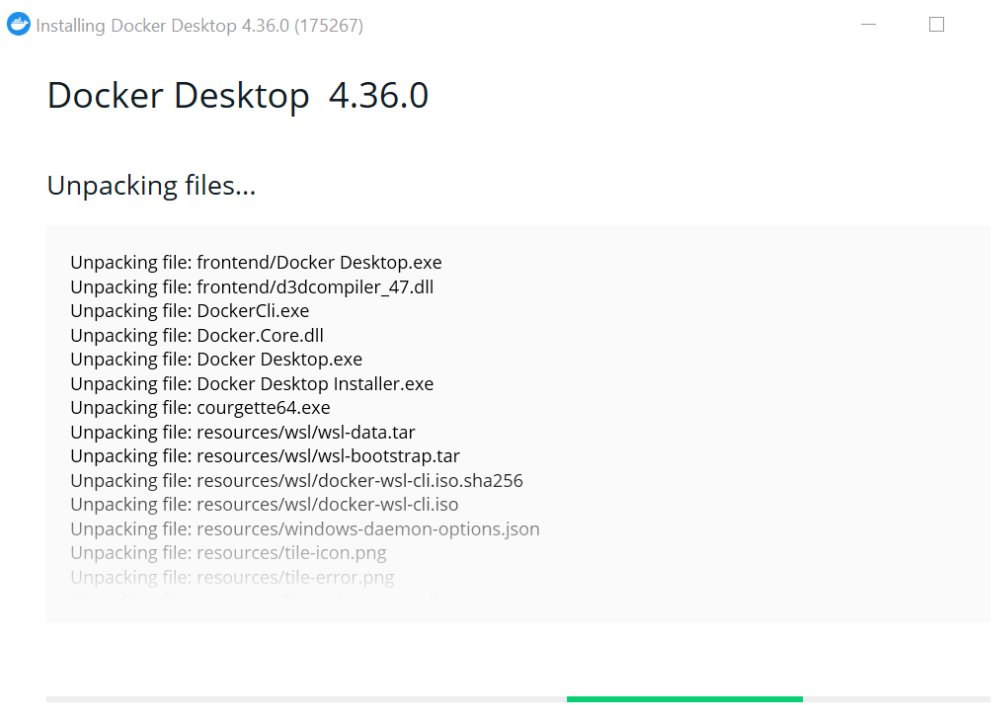
sudo apt install snapd

sudo snap install docker

Parte práctica – instalación – opción 3

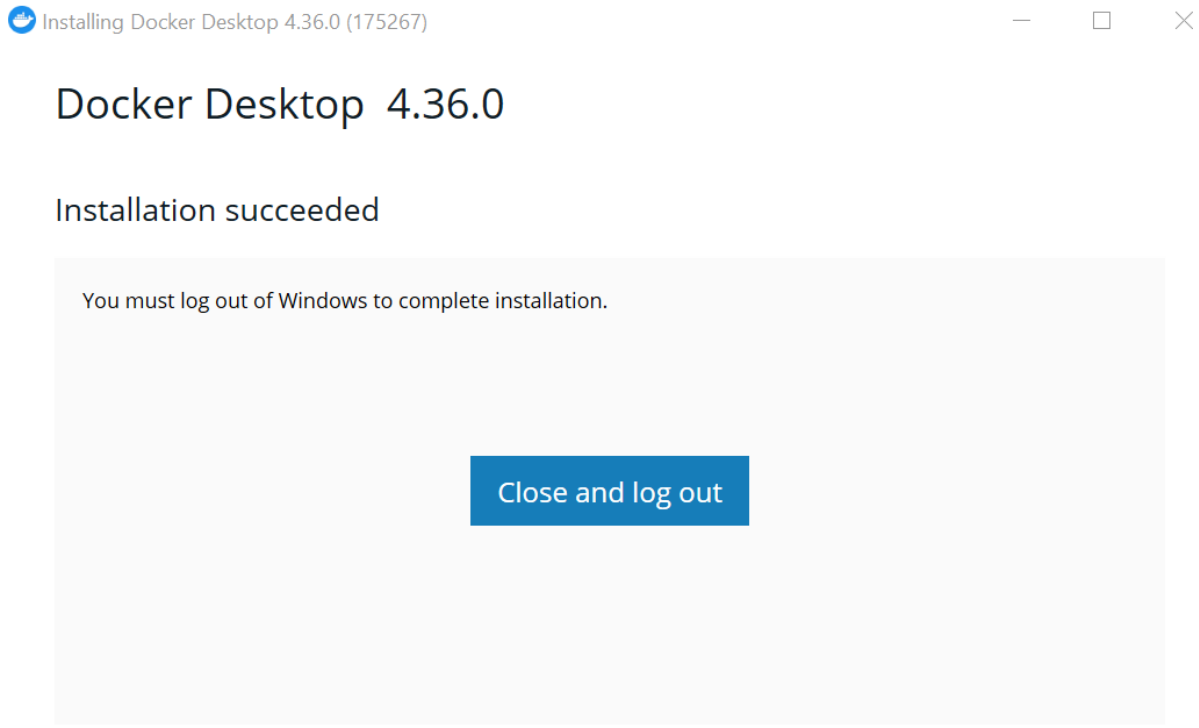
Instalación en Windows con WSL 2 (subsistema de Windows para Linux)

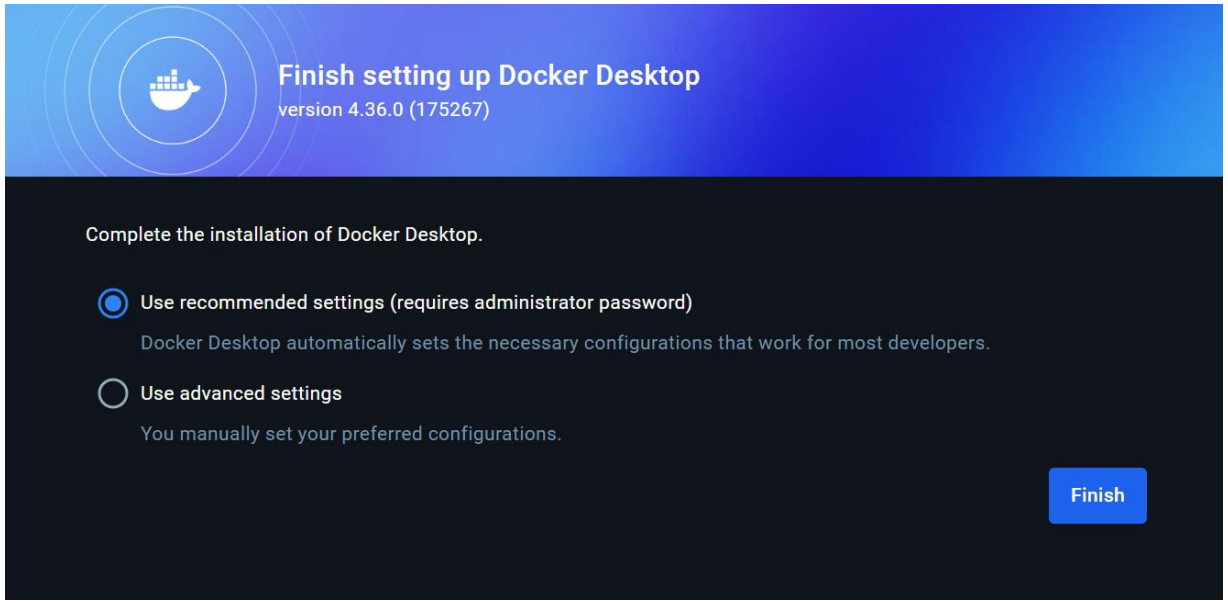




Parte práctica – instalación – opción 3

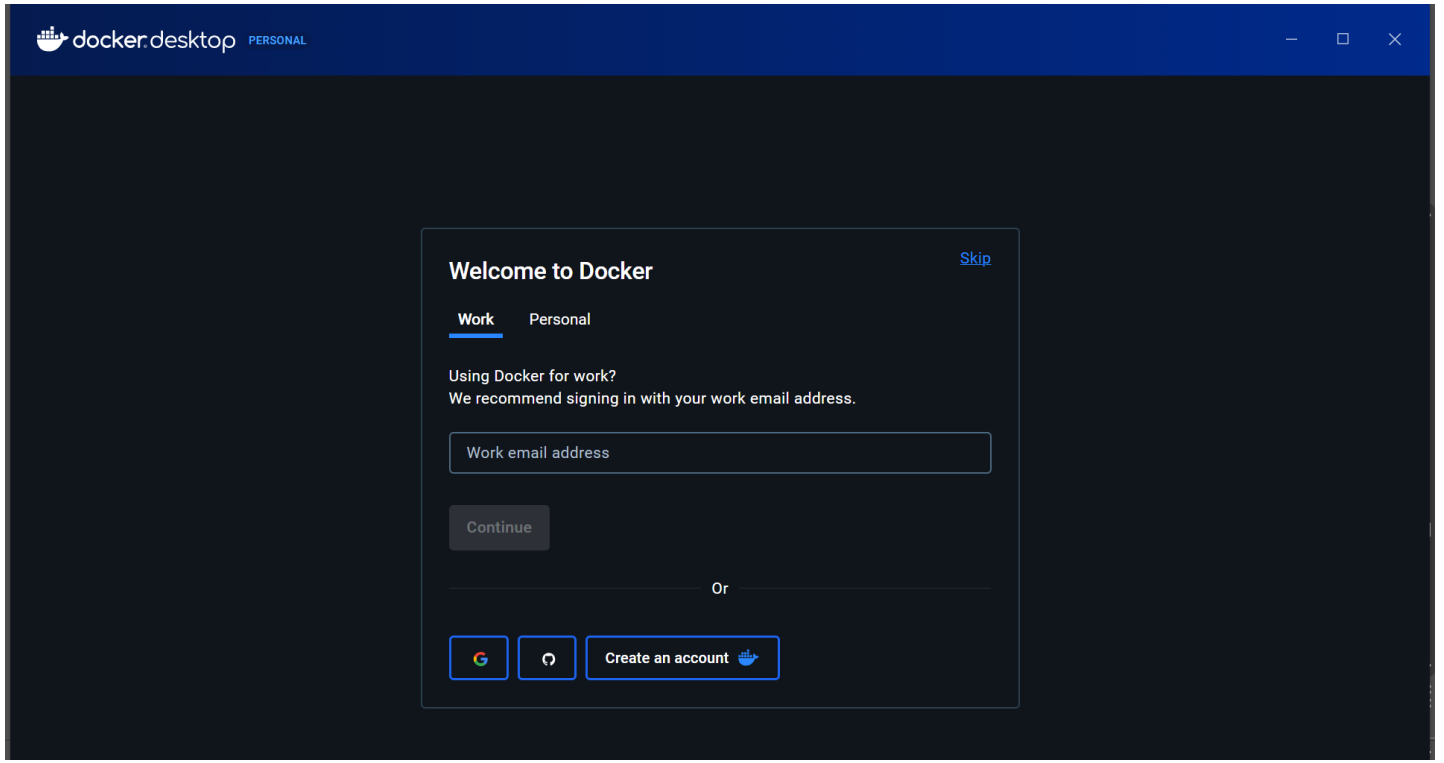
Instalación en Windows con WSL 2 (subsistema de Windows para Linux)





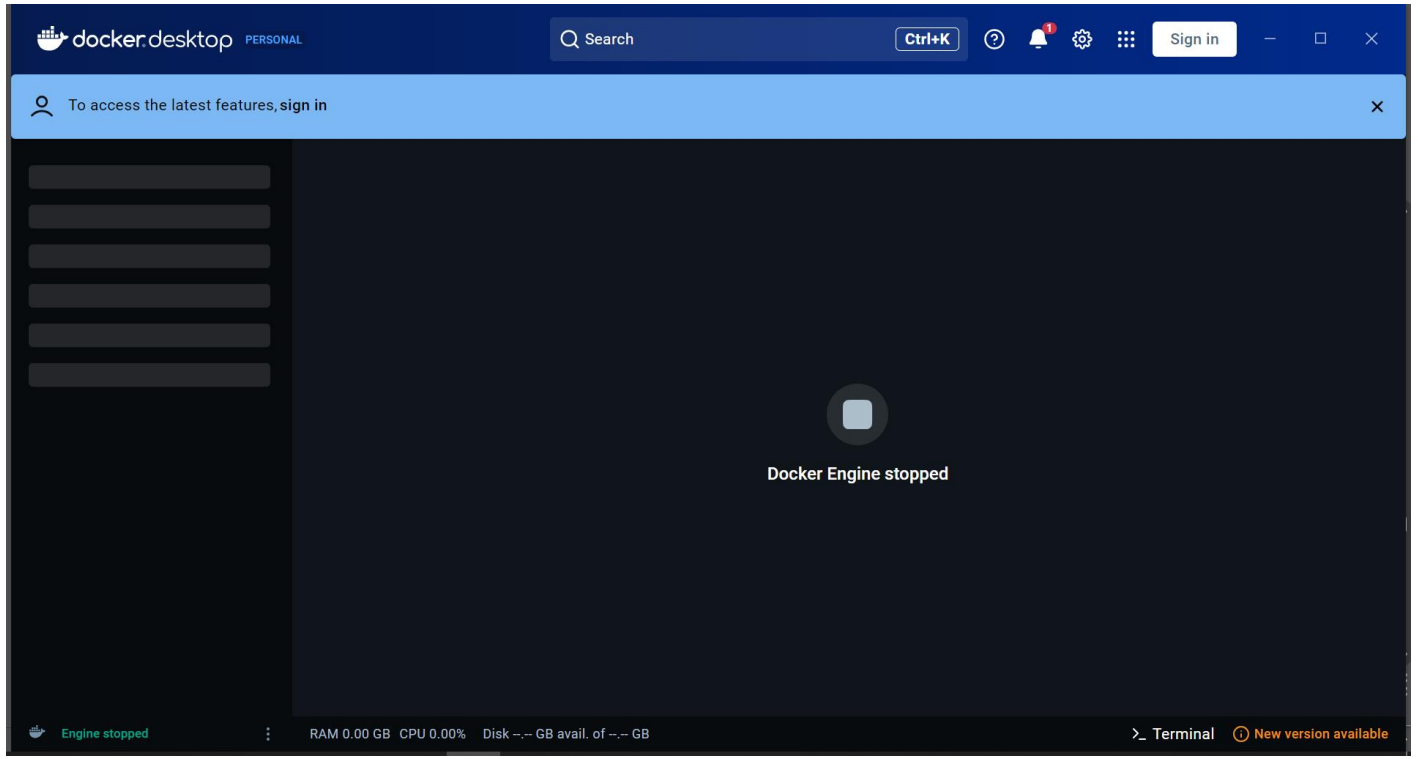
Parte práctica – instalación – opción 3

Instalación en Windows con WSL 2 (subsistema de Windows para Linux)



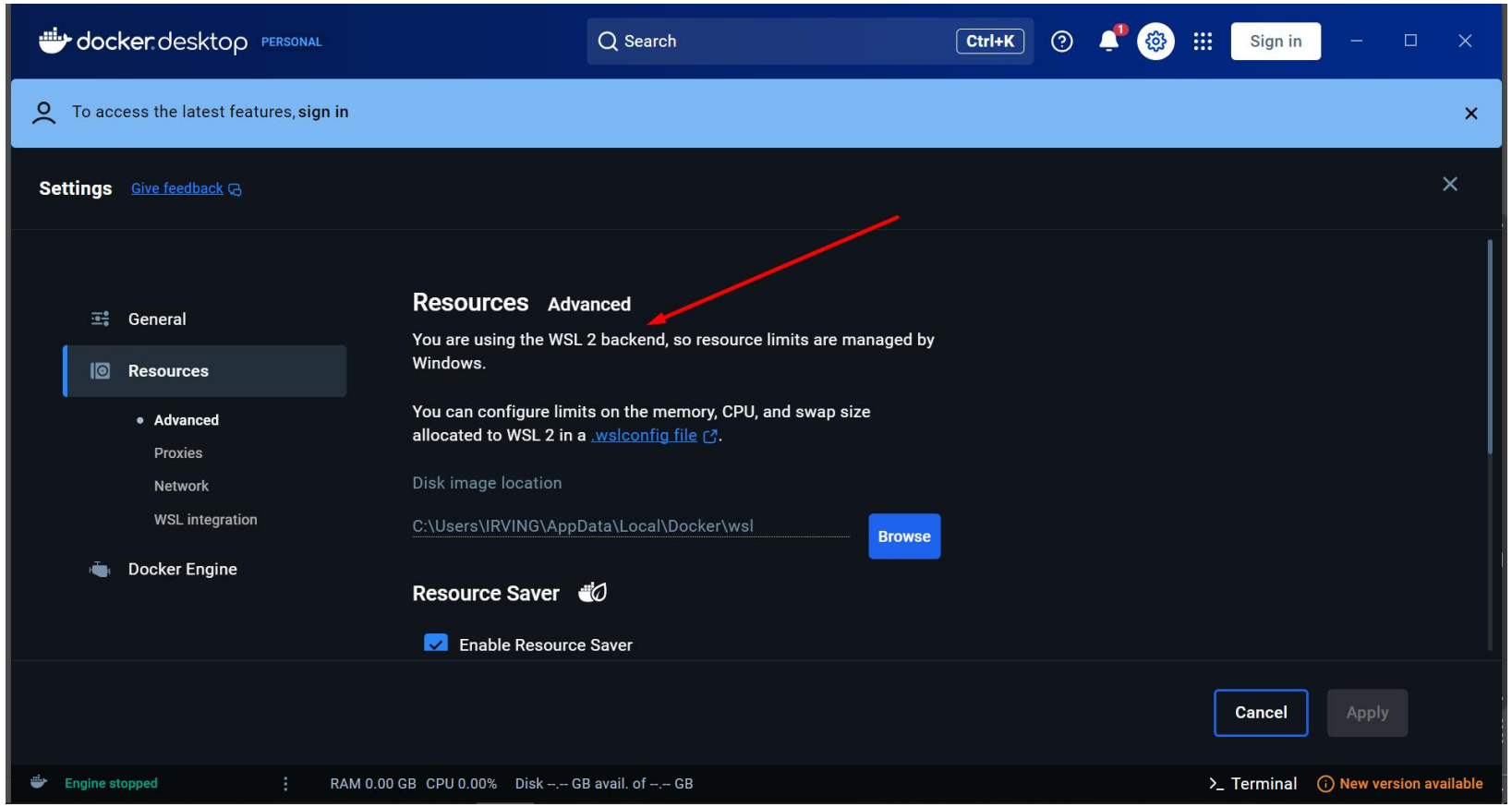
Parte práctica – instalación – opción 3

Instalación en Windows con WSL 2 (subsistema de Windows para Linux)



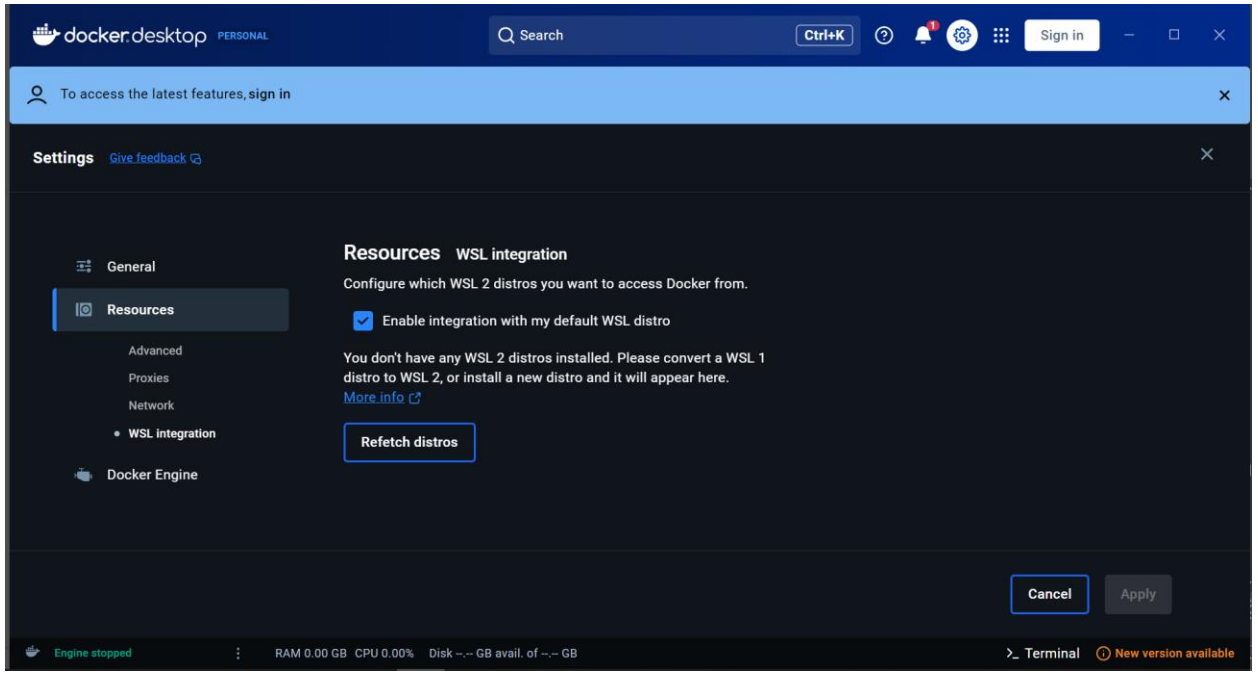
Parte práctica – instalación – opción 3

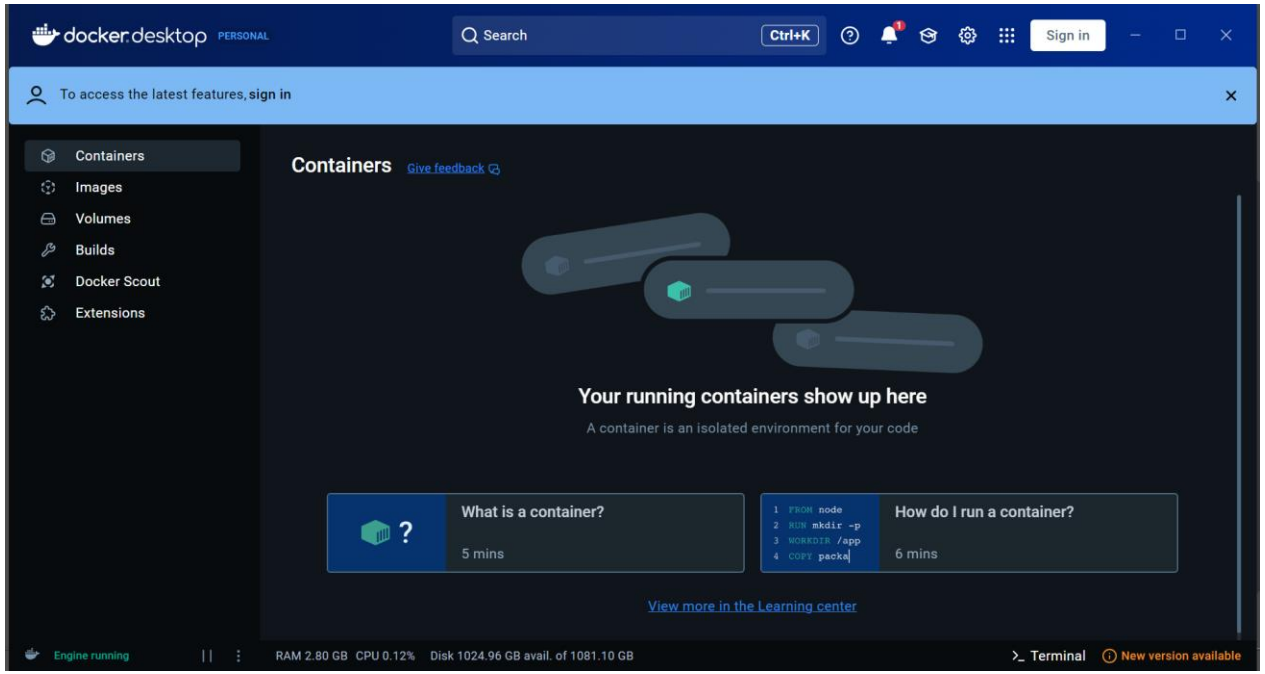
Instalación en Windows con WSL 2 (subsistema de Windows para Linux)



Parte práctica – instalación – opción 3

Instalación en Windows con WSL 2 (subsistema de Windows para Linux)

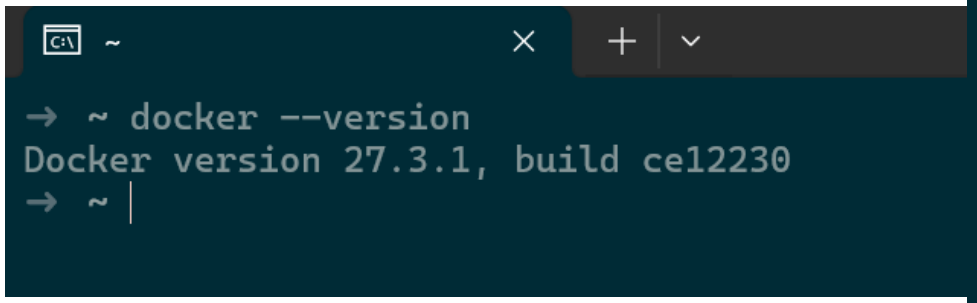


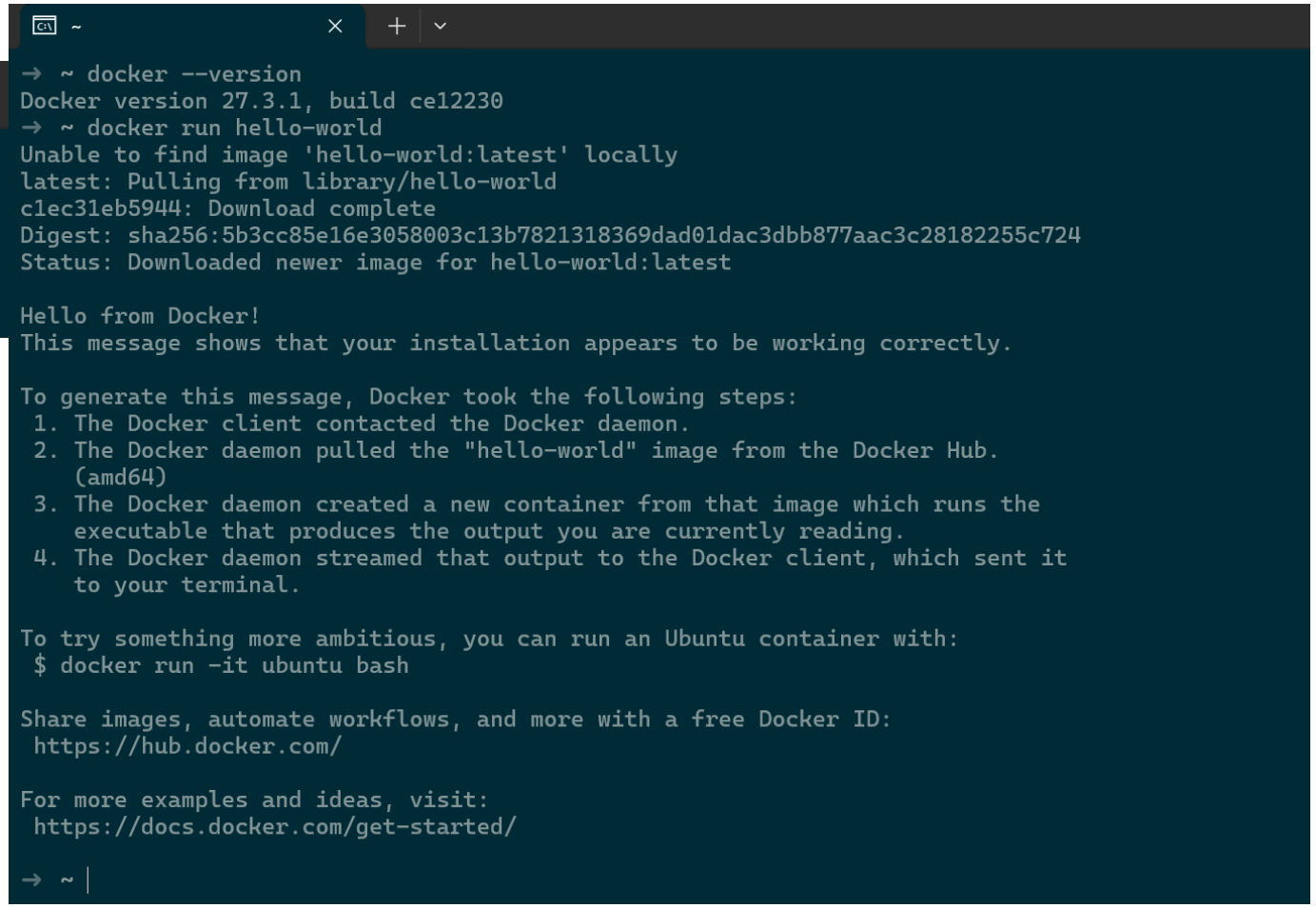


Ajustes adicionales https://docs.docker.com/desktop/features/wsl/

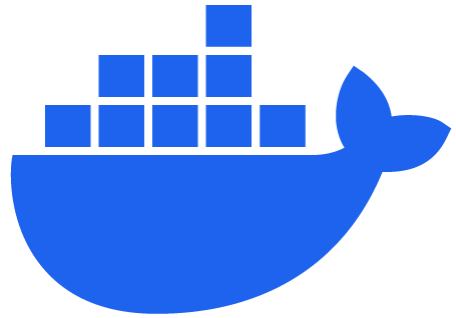
Comprobación de la instalación

Instalación en Windows con WSL 2 (subsistema de Windows para Linux)





Parte práctica



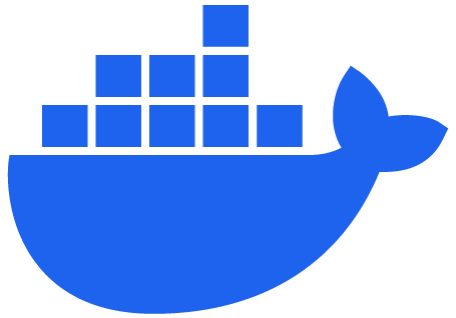
Comprobación de la instalación en Linux:

sudo docker run hello-world

```
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:7d246653d0511db2a6b2e0436cfd0e52ac8c066000264b3ce63331ac66d
Status: Downloaded newer image for hello-world:latest
```

```
Hello from Docker!
This message shows that your installation appears to be working correctl
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
 executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which s
 to your terminal.
```

Algunos comandos utilizados



docker version muestra la versión y detalles de docker

docker --help muestra ayuda

docker ps Lista contenedores (sólo los activos)

docker ps -a Lista contenedores (todos)

docker ps -l Lista el último contenedor creado (activo o no)

docker run IMAGEN COMANDO Crea un contenedor a partir de IMAGEN, ejecuta COMANDO en él y muestra el resultado (y termina)

docker run -t -i IMAGEN COMANDO …de forma interactiva (-i mantiene STDIN abierto y -t asigna un terminal virtual en el contenedor)

docker run -d IMAGEN COMANDO …en modo daemon (sin mostrar el resultado)

docker run -P IMAGEN COMANDO …mapea los puertos necesarios (se muestran en un ps)

docker run -p 80:1080 IMAGEN COMANDO …mapea los puertos indicados

docker run --name="NOMBRE" IMAGEN COMANDO …con el nombre NOMBRE

docker stop CONTENEDOR detiene CONTENEDOR

docker start CONTENEDOR inicia CONTENEDOR

docker restart CONTENEDOR reinicia CONTENEDOR

docker rm COMTENEDOR borra CONTENEDOR

docker images Lista imágenes

docker rmi IMAGEN borra IMAGEN

docker create IMAGEN Crea un contenedor a partir de IMAGEN (admite muchos de los modificadores de run, como -itP o --name)

docker exec -it CONTENEDOR bash inicia sesión interactiva bash en CONTENEDOR (-i mantiene STDIN abierto y -t asigna un terminal virtual en el contenedor)

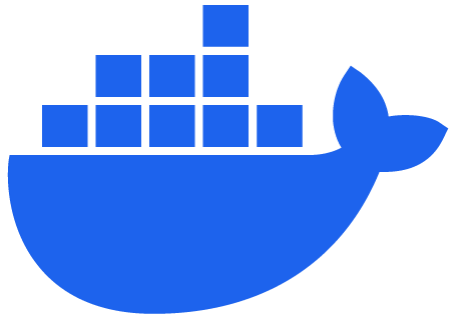
docker logs CONTENEDOR Muestra un log con las salidas de CONTENEDOR

docker logs -t CONTENEDOR …con timestamp

docker top CONTENEDOR muestra los procesos en el contenedor

docker commit CONTENEDOR REPO crea imagen REPO a partir de CONTENEDOR

Conclusiones



Docker ha revolucionado la forma en que se desarrollan y despliegan aplicaciones, ofreciendo una solución eficiente, escalable y portátil para el desarrollo moderno de software. Ha facilitado la adopción de prácticas DevOps y microservicios, mejorando la colaboración entre los equipos de desarrollo y operaciones.

Ejercicios con Docker

E1

Comprobar la ejecución de los siguiente comandos.

docker --version

docker run hello-world

```
~
→ ~ docker --version
Docker version 27.3.1, build ce12230
→ ~ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
c1ec31eb5944: Download complete
Digest: sha256:5b3cc85e16e3058003c13b7821318369dad01dac3dbb877aac3c28182255c724
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/

→ ~
```

Ejercicios con Docker

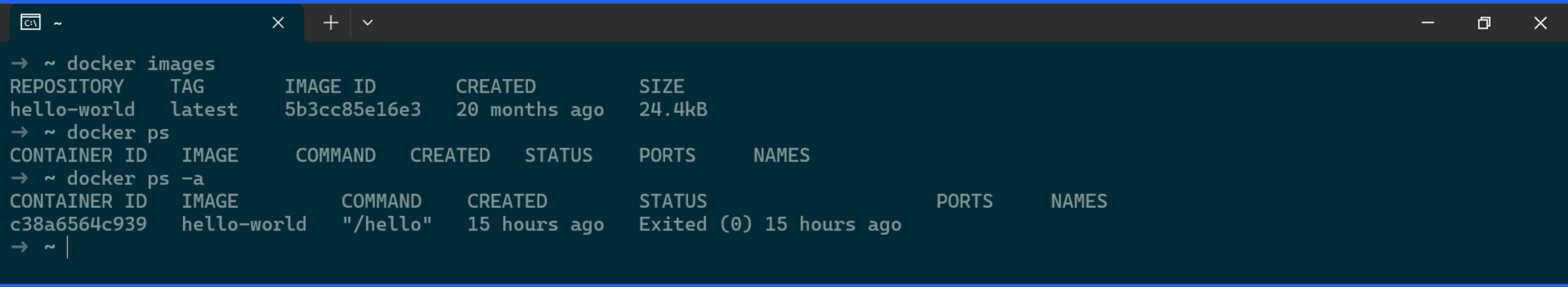
E2

Listar imágenes y contenedores

docker images *# Lista las imágenes disponibles*

docker ps *# Lista los contenedores en ejecución*

docker ps -a *# Lista todos los contenedores (incluidos detenidos)*



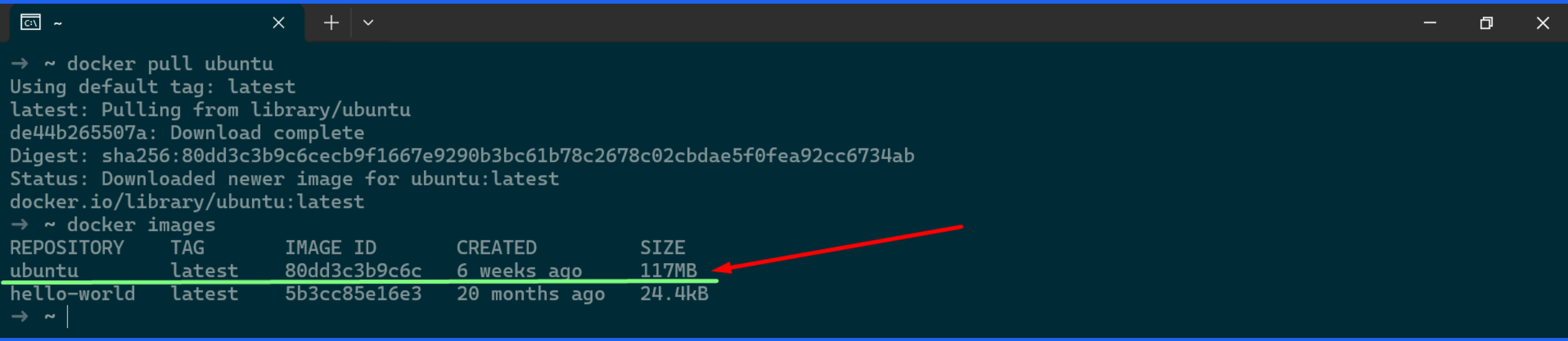
Ejercicios con Docker

E3

Descargar una imagen desde Docker Hub

docker pull ubuntu # Descargamos la imagen oficial de Ubuntu

docker images # Verificar la que la imagen se descargó



Ejercicios con Docker

E4

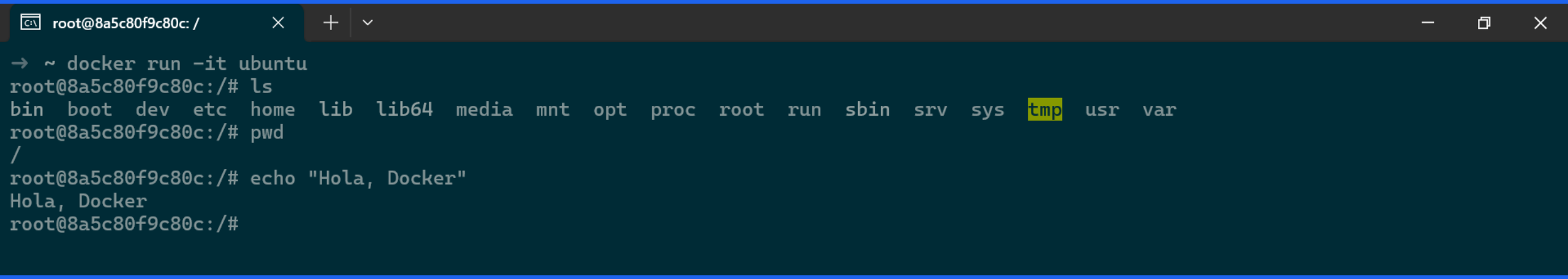
Crear y acceder a un contenedor

docker run –it ubuntu # Iniciamos un contenedor interactivo de Ubuntu

Dentro del contenedor, ejecuta comandos básicos como:

ls, pwd, echo “°¡Hola, Docker!”

Salir del contenedor con *exit*



Ejercicios con Docker

E5

Acceder a un contenedor

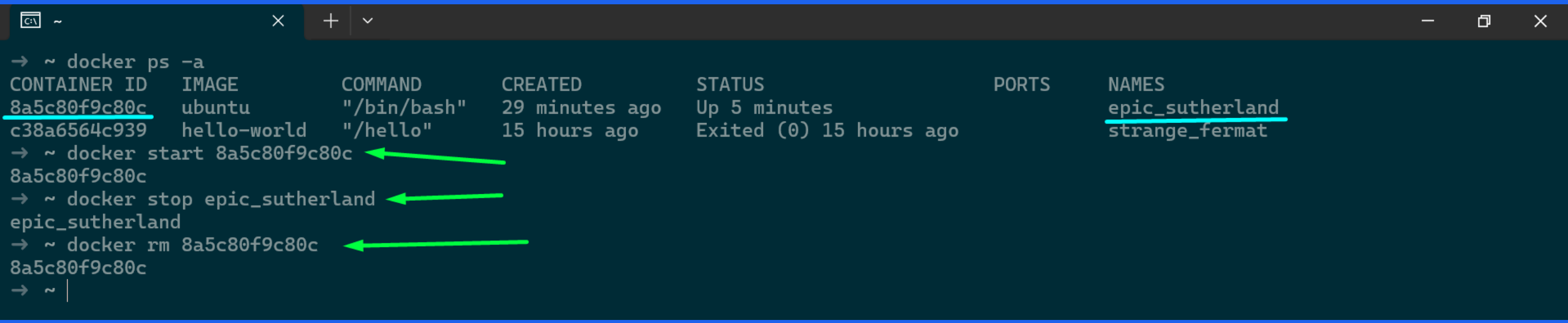
root@8a5c80f9c80c: /

```
→ ~ docker ps -a
CONTAINER ID   IMAGE         COMMAND       CREATED          STATUS                      PORTS     NAMES
8a5c80f9c80c   ubuntu        "/bin/bash"   22 minutes ago   Exited (0) 6 minutes ago              epic_sutherland
c38a6564c939   hello-world   "/hello"      15 hours ago     Exited (0) 15 hours ago               strange_fermat
→ ~ docker start epic_sutherland
epic_sutherland
→ ~ sudo docker exec -it epic_sutherland /bin/bash
root@8a5c80f9c80c:/# pwd
/
root@8a5c80f9c80c:/# ls -la
total 56
drwxr-xr-x   1 root root 4096 Jan  4 19:48 .
drwxr-xr-x   1 root root 4096 Jan  4 19:48 ..
-rwxr-xr-x   1 root root    0 Jan  4 19:48 .dockerenv
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 Jan  4 20:12 dev
drwxr-xr-x   1 root root 4096 Jan  4 19:48 etc
drwxr-xr-x   3 root root 4096 Nov 19 09:52 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 Nov 19 09:46 media
```

Ejercicios con Docker

E6

Inicia, parar y eliminar un contenedor



Ejercicios con Docker

E7

2 formas de ingresar a un contenedor

root@2e89aafbbba8: /

```
→ ~ docker ps -a
CONTAINER ID   IMAGE         COMMAND       CREATED          STATUS                      PORTS     NAMES
2e89aafbbba8   ubuntu        "/bin/bash"   14 minutes ago   Exited (0) 37 seconds ago             ecstatic_allen
c38a6564c939   hello-world   "/hello"      16 hours ago     Exited (0) 16 hours ago               strange_fermat
→ ~ docker start ecstatic_allen
ecstatic_allen
→ ~ docker ps -a
CONTAINER ID   IMAGE         COMMAND       CREATED          STATUS                      PORTS     NAMES
2e89aafbbba8   ubuntu        "/bin/bash"   14 minutes ago   Up 23 seconds                         ecstatic_allen
c38a6564c939   hello-world   "/hello"      16 hours ago     Exited (0) 16 hours ago               strange_fermat
→ ~ docker attach ecstatic_allen
root@2e89aafbbba8:/# ls
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@2e89aafbbba8:/#
```

root@2e89aafbbba8: /

```
→ ~ docker ps -a
CONTAINER ID   IMAGE         COMMAND       CREATED          STATUS                      PORTS     NAMES
2e89aafbbba8   ubuntu        "/bin/bash"   17 minutes ago   Exited (0) 19 seconds ago             ecstatic_allen
c38a6564c939   hello-world   "/hello"      16 hours ago     Exited (0) 16 hours ago               strange_fermat
→ ~ docker start ecstatic_allen
ecstatic_allen
→ ~ docker exec -it ecstatic_allen /bin/bash
root@2e89aafbbba8:/# ls
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@2e89aafbbba8:/#
```

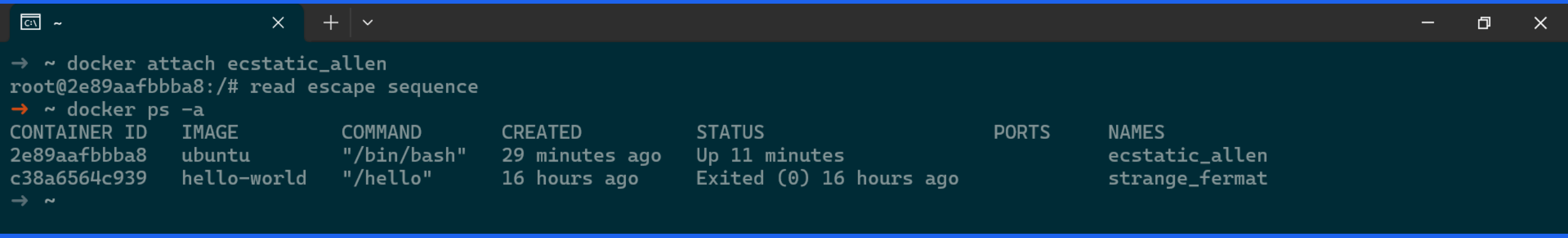
Irving Ulises Hernández Miguel

Ejercicios con Docker

E8

Salir del contenedor sin pararlo con *exit*

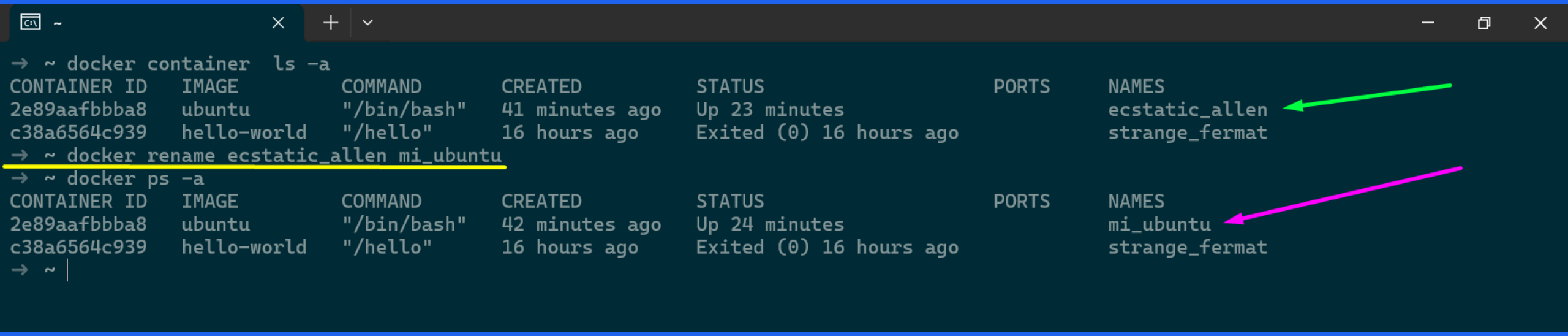
Para salir sin detener el contenedor, presiona Ctrl + P seguido de Ctrl + Q.



Ejercicios con Docker

E9

Renombrar contenedor



Ejercicios con Docker

E10

Ver detalles completos de un contenedor

```
→ ~ docker container ls -a
CONTAINER ID   IMAGE         COMMAND       CREATED          STATUS                      PORTS     NAMES
2e89aafbbba8   ubuntu        "/bin/bash"   45 minutes ago   Up 27 minutes                         mi_ubuntu
c38a6564c939   hello-world   "/hello"      16 hours ago     Exited (0) 16 hours ago               strange_fermat
→ ~ docker inspect mi_ubuntu
[
    {
        "Id": "2e89aafbbba81b47c88b1ae3d8728c40ca9e06b084df420934f7cbdffc1994c7",
        "Created": "2025-01-04T20:30:54.65352194Z",
        "Path": "/bin/bash",
        "Args": [],
        "State": {
            "Status": "running",
            "Running": true,
            "Paused": false,
            "Restarting": false,
            "OOMKilled": false,
            "Dead": false,
            "Pid": 1381,
            "ExitCode": 0,
            "Error": "",
            "StartedAt": "2025-01-04T20:48:55.510312186Z",
            "FinishedAt": "2025-01-04T20:47:45.618035544Z"
        },
```

Ejercicios con Docker

E11

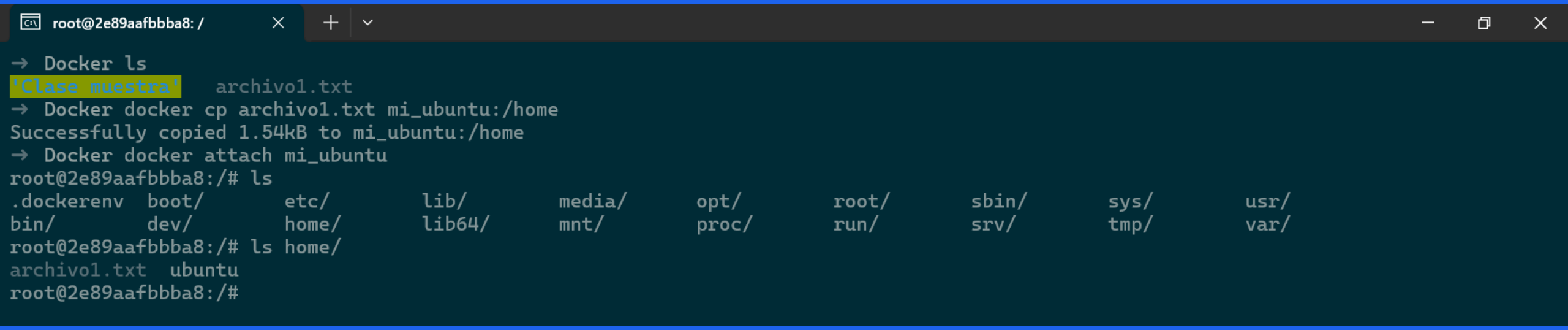
Ver logs de un contenedor

```
~
→ ~ docker container ls -a
CONTAINER ID   IMAGE         COMMAND       CREATED         STATUS                     PORTS     NAMES
2e89aafbbba8   ubuntu        "/bin/bash"   2 hours ago     Up About an hour                     mi_ubuntu
c38a6564c939   hello-world   "/hello"      17 hours ago    Exited (0) 17 hours ago              strange_fermat
→ ~ docker logs mi_ubuntu
root@2e89aafbbba8:/# exit
exit
root@2e89aafbbba8:/#
root@2e89aafbbba8:/# exit
exit
root@2e89aafbbba8:/# ls
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@2e89aafbbba8:/# pwd
/
root@2e89aafbbba8:/# exit
exit
root@2e89aafbbba8:/# ls
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@2e89aafbbba8:/# exit
exit
→ ~
```

Ejercicios con Docker

E12

Copiar archivos del host al contenedor



Ejercicios con Docker

E13

Copiar archivos del contenedor al host

```
..arcial/Docker
→ Docker ls
'Clase muestra'
→ Docker docker attach mi_ubuntu
root@2e89aafbbba8:/home# ls
archivo2.txt  ubuntu
root@2e89aafbbba8:/home# pwd
/home
root@2e89aafbbba8:/home# read escape sequence
→ Docker docker cp mi_ubuntu:/home archivo2.txt
Successfully copied 9.73kB to /mnt/c/Users/IRVING/Mis documentos/SEMESTRE 24-25-A/LI-706/3_Tercer Parcial/Docker/archivo2.txt
→ Docker ls -la
total 0
drwxrwxrwx 1 root root 4096 Jan  4 16:53  .
drwxrwxrwx 1 root root 4096 Jan  3 20:42  ..
drwxrwxrwx 1 root root 4096 Jan  4 16:30 'Clase muestra'
drwxrwxrwx 1 root root 4096 Jan  4 16:33  archivo2.txt
→ Docker
```

Ejercicios con Docker

E14

Comprobar el uso de recursos mediante la visualización de estadísticas.

..arcial/Docker

```
→ Docker docker stats
```

docker

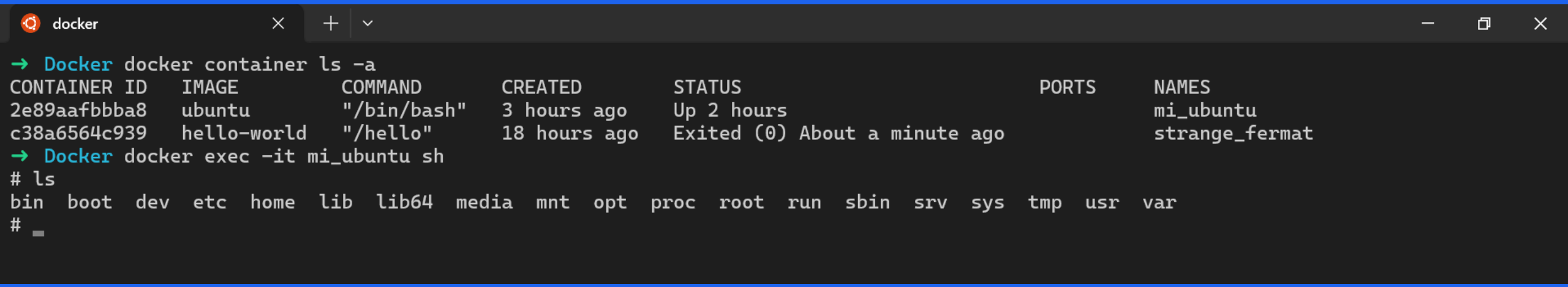
```
CONTAINER ID   NAME         CPU %     MEM USAGE / LIMIT     MEM %     NET I/O       BLOCK I/O   PIDS
2e89aafbbba8   mi_ubuntu    0.00%     2.258MiB / 7.652GiB   0.03%     1.05kB / 0B   0B / 0B     3
```

Ejercicios con Docker

E15

Obtener un Shell en cualquier contenedor.

Si el contenedor no tiene *bash* instalado (como puede ocurrir en imágenes mínimas), se puede usar *sh.*



Volúmenes en Docker y Persistencia de Datos

Los volúmenes en Docker son una forma de almacenar datos de manera persistente fuera del contenedor, de modo que los datos no se pierdan cuando el contenedor se detenga o elimine.

Un volumen es un directorio en el sistema de archivos de tu máquina que Docker gestiona. Los volúmenes permiten que los contenedores almacenen datos de manera persistente fuera de su sistema de archivos.

¿Por qué usar volúmenes?

Persistencia: *Los volúmenes persisten incluso después de detener o eliminar contenedores.*

Compartir datos entre contenedores: *Varios contenedores pueden acceder al mismo volumen.*

Gestión fácil: *Docker gestiona los volúmenes de manera eficiente, facilitando su copia, respaldo y restauración.*

Ejercicios Volúmenes - Docker V1

Crear un volumen: *Puedes crear un volumen usando el siguiente comando.*

```
..arcial/Docker
→ Docker docker volume create mi_volumen
mi_volumen
→ Docker
```

Usar el volumen en un contenedor:

Opción -v: *Para usar un volumen en un contenedor, monta el volumen en el contenedor con la opción -v o --mount. Esto crea un contenedor de Ubuntu y monta el volumen mi_volumen en el directorio /data dentro del contenedor.*

```
root@9988afcc2561: /
→ Docker docker volume create mi_volumen
mi_volumen
→ Docker docker run -it -v mi_volumen:/data ubuntu
root@9988afcc2561:/# ls
bin  boot  data  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@9988afcc2561:/#
```

Opción --mount (más recomendada para mayor control): *Ambos comandos hacen lo mismo, pero la opción --mount es más flexible y fácil de entender.*

```
root@111d9c351ddd: /
→ Docker docker run -it --mount source=mi_volumen,target=/data ubuntu
root@111d9c351ddd:/# ls
bin  boot  data  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@111d9c351ddd:/#
```

Ejercicios Volúmenes - Docker V2

Volúmenes y persistencia después de eliminar contenedores

Si eliminas el contenedor, el volumen persiste en tu máquina.

Paso 1: Eliminar el contenedor - Puedes detener y eliminar el contenedor sin perder los datos:

```
..arcial/Docker
→ Docker docker container ls -a
CONTAINER ID   IMAGE         COMMAND       CREATED          STATUS                      PORTS     NAMES
111d9c351ddd   ubuntu        "/bin/bash"   13 minutes ago   Exited (0) 9 minutes ago              peaceful_jackson
9988afcc2561   ubuntu        "/bin/bash"   19 minutes ago   Exited (0) 13 minutes ago             reverent_moser
2e89aafbbba8   ubuntu        "/bin/bash"   8 hours ago      Up 8 hours                            mi_ubuntu
c38a6564c939   hello-world   "/hello"      23 hours ago     Exited (0) 5 hours ago                strange_fermat
→ Docker docker rm 111d9c351ddd 9988afcc2561
111d9c351ddd
9988afcc2561
→ Docker
```

Paso 2: Verificar que el volumen sigue existiendo. Verifica que el volumen aún exista usando:

```
..arcial/Docker
→ Docker docker volume ls
DRIVER    VOLUME NAME
local     mi_volumen
→ Docker _
```

El volumen sigue estando disponible para ser utilizado por otros contenedores.

Ejercicios Volúmenes - Docker

V3

Montar directorios locales como volúmenes

Paso 1: Usar un directorio local como volumen. Supongamos que tienes un directorio llamado /mi/directorio/local en tu máquina, puedes montarlo en el contenedor con:

```
root@975798dd4220: /
→ Docker mkdir -p ./mi/directorio/local
→ Docker tree mi
mi
└── directorio
    └── local

3 directories, 0 files
→ Docker docker run -it -v ./mi/directorio/local:/data ubuntu
root@975798dd4220:/# ls
bin  boot  data  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@975798dd4220:/#
```

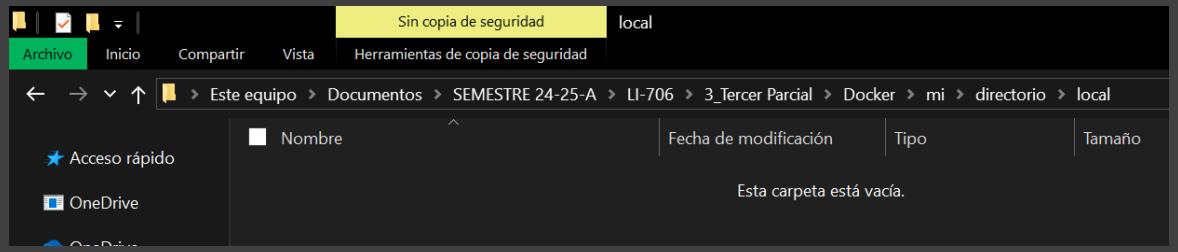
Esto montará tu directorio local dentro del contenedor en /data. Los cambios realizados dentro de /data serán reflejados en tu directorio local.

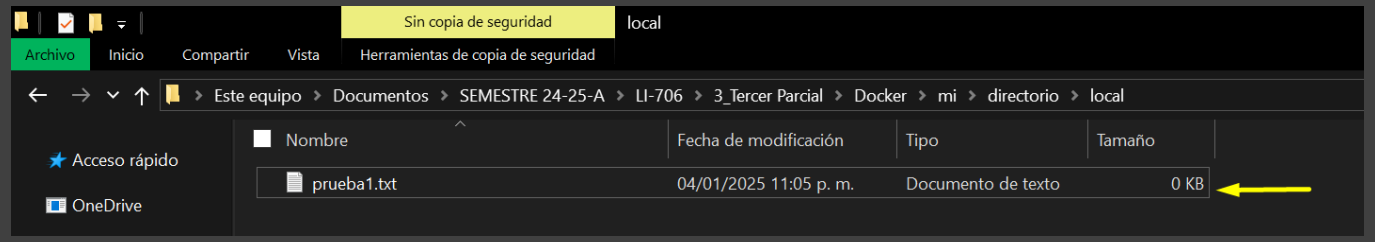
Ejercicios Volúmenes - Docker

V4

Montar directorios locales como volúmenes - comprobar

```
root@975798dd4220: /data
→ Docker docker ps -a
CONTAINER ID   IMAGE         COMMAND       CREATED          STATUS                      PORTS     NAMES
975798dd4220   ubuntu        "/bin/bash"   18 minutes ago   Up 18 minutes                         zealous_dhawan
2e89aafbbba8   ubuntu        "/bin/bash"   9 hours ago      Up 8 hours                            mi_ubuntu
c38a6564c939   hello-world   "/hello"      24 hours ago     Exited (0) 6 hours ago                strange_fermat
→ Docker docker exec -it zealous_dhawan bash
root@975798dd4220:/# ls
bin  boot  data  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@975798dd4220:/# cd data
root@975798dd4220:/data# touch prueba1.txt
root@975798dd4220:/data# ls
prueba1.txt
root@975798dd4220:/data#
```





Ejercicios Volúmenes - Docker V5

Ejemplo completo de persistencia de datos con un contenedor de base de datos

Vamos a usar un contenedor con una base de datos MySQL para demostrar la persistencia de datos.

Paso 1: Iniciar un contenedor MySQL con un volumen:

-v mysql_data:/var/lib/mysql: Monta el volumen mysql_data en el directorio donde MySQL guarda sus datos.

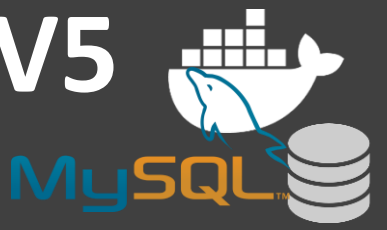
-e MYSQL_ROOT_PASSWORD=root: Establece la contraseña para el usuario root de MySQL.

..arcial/Docker

```
→ Docker docker run -d --name mysql-db -e MYSQL_ROOT_PASSWORD=root -v mysql_data:/var/lib/mysql mysql
Unable to find image 'mysql:latest' locally
latest: Pulling from library/mysql
2c0a233485c3: Download complete
a841bff36f3c: Download complete
cd0d5df9937b: Download complete
5e49e1f26961: Download complete
570d30cf82c5: Download complete
1f87d67b89c6: Download complete
80ba30c57782: Download complete
cb5a6a8519b2: Download complete
ced670fc7f1c: Download complete
0b9dc7ad7f03: Download complete
Digest: sha256:0255b469f0135a0236d672d60e3154ae2f4538b146744966d96440318cc822c6
Status: Downloaded newer image for mysql:latest
eca4bd8c373273e9ec13196865a68d16249f81ccf42f2fef28e50c67d25a8446
→ Docker
```

Ejercicios Volúmenes - Docker

V5



Ejemplo completo de persistencia de datos con un contenedor de base de datos

Paso 2: Acceder a MySQL.

docker

```
→ Docker docker exec -it mysql-db mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 9.1.0 MySQL Community Server - GPL

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>
```

Ejercicios Volúmenes - Docker

V5

Ejemplo completo de persistencia de datos con un contenedor de base de datos

Paso 3: Comandos básicos de MySQL para verificar la persistencia.

```
mysql> CREATE DATABASE pokemon;
Query OK, 1 row affected (0.07 sec)

mysql> USE pokemon;
Database changed
```

```
mysql> exit
Bye
→ Docker _
```

docker

```
mysql> CREATE TABLE ejemplo (
    ->     id INT AUTO_INCREMENT PRIMARY KEY,
    ->     nombre VARCHAR(50)
    -> );
Query OK, 0 rows affected (0.09 sec)

mysql> INSERT INTO ejemplo (nombre) VALUES ('Docker');
Query OK, 1 row affected (0.02 sec)

mysql> SELECT * FROM ejemplo;
+----+--------+
| id | nombre |
+----+--------+
|  1 | Docker |
+----+--------+
 row in set (0.00 sec)

 ysql> _
```

Ejercicios Volúmenes - Docker V5

Ejemplo completo de persistencia de datos con un contenedor de base de datos

Paso 4: Verificar la persistencia de los datos.

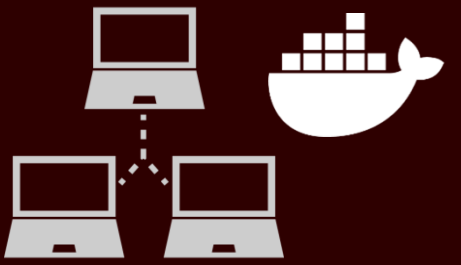
..arcial/Docker

```
→ Docker docker stop mysql-db
mysql-db
→ Docker docker rm mysql-db
mysql-db
→ Docker docker run -d --name mysql-db -e MYSQL_ROOT_PASSWORD=root -v mysql_data:/var/lib/mysql mysql
eea8a2208848b8afd2ed897234f84aa5bf8226f7f4f45922340d6df285aab4b6
→ Docker docker exec -it mysql-db mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 9.1.0 MySQL Community Server - GPL

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> _
```

docker

```
mysql> USE pokemon;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> SELECT * FROM ejemplo;
+----+---------+
| id | nombre |
+----+---------+
|  1 | Docker |
+----+---------+
1 row in set (0.00 sec)

mysql>
```

Redes con Docker



Las redes en Docker permiten que los contenedores puedan comunicarse entre sí o con el exterior de manera controlada y eficiente. Docker ofrece varias opciones de redes predefinidas que cubren diferentes casos de uso.

Docker provee tres tipos principales de redes:

- **Bridge (puente):** Red por defecto para contenedores que no especifican una red. Los contenedores en esta red pueden comunicarse entre sí, pero no desde fuera del host.
- **Host:** El contenedor usa directamente la red del host, eliminando la capa de virtualización de red.
- **None:** El contenedor no tiene conectividad de red.

Además, puedes crear redes personalizadas, que son útiles para definir topologías más complejas.

Redes con Docker

R1

Paso 1: Crear una red personalizada.

- *mi_red_personalizada* es el nombre de la red.
- Por defecto, se usa el controlador bridge.

```
..arcial/Docker
→ Docker docker network create mi_red_personalizada
29fa2ae423c08b946328f09c3cd880d5df2209f18464423f5b54d1ba75bb28a2
→ Docker
```

Paso 2: Ejecutar contenedores en la red. Crea dos contenedores que usen esta red y que puedan comunicarse entre sí:

```
..arcial/Docker
→ Docker docker run -dit --name contenedor1 --network mi_red_personalizada alpine sh
docker run -dit --name contenedor2 --network mi_red_personalizada alpine sh

Unable to find image 'alpine:latest' locally
latest: Pulling from library/alpine
38a8310d387e: Download complete
Digest: sha256:21dc6063fd678b478f57c0e13f47560d0ea4eeba26dfc947b2a4f81f686b9f45
Status: Downloaded newer image for alpine:latest
f3d9c52b84ede7aa5bf82afe798ead6151e2716970b70811e38a989f89d2e7d1
87a03ffa8a0233e8f666e4c8ee64ec5eb2ff91579c2c2813a803f4a64690c84e
```

Redes con Docker

R1

Paso 3: Probar la comunicación entre contenedores.

- Ingresa al contenedor1 y prueba el acceso al contenedor2 por su nombre:

docker

```
→ Docker docker exec -it contenedor1 sh
/ # ping contenedor2
PING contenedor2 (172.18.0.3): 56 data bytes
64 bytes from 172.18.0.3: seq=0 ttl=64 time=0.268 ms
64 bytes from 172.18.0.3: seq=1 ttl=64 time=0.147 ms
64 bytes from 172.18.0.3: seq=2 ttl=64 time=0.140 ms
64 bytes from 172.18.0.3: seq=3 ttl=64 time=0.142 ms
64 bytes from 172.18.0.3: seq=4 ttl=64 time=0.142 ms
64 bytes from 172.18.0.3: seq=5 ttl=64 time=0.142 ms

64 bytes from 172.18.0.3: seq=6 ttl=64 time=0.142 ms
64 bytes from 172.18.0.3: seq=7 ttl=64 time=0.138 ms
64 bytes from 172.18.0.3: seq=8 ttl=64 time=0.140 ms
^C
--- contenedor2 ping statistics ---
9 packets transmitted, 9 packets received, 0% packet loss
round-trip min/avg/max = 0.138/0.155/0.268 ms
/ #
```

Redes con Docker

R2

Caso práctico: Conectar Adminer con MySQL

..arcial/Docker

```
→ Docker docker network create red_db
31577a1f2089496a11aa6d582f93e5e67c73a38abc692182c9bc3c70913ec716
```

..arcial/Docker

```
→ Docker docker run -dit --name mysql_server --network red_db -e MYSQL_ROOT_PASSWORD=root -e MYSQL_DATABASE=prueba mysql:latest

5629243b236a7d69a861d23097cabee5d9a0869e0d79a27093fe543cc90560d0
```

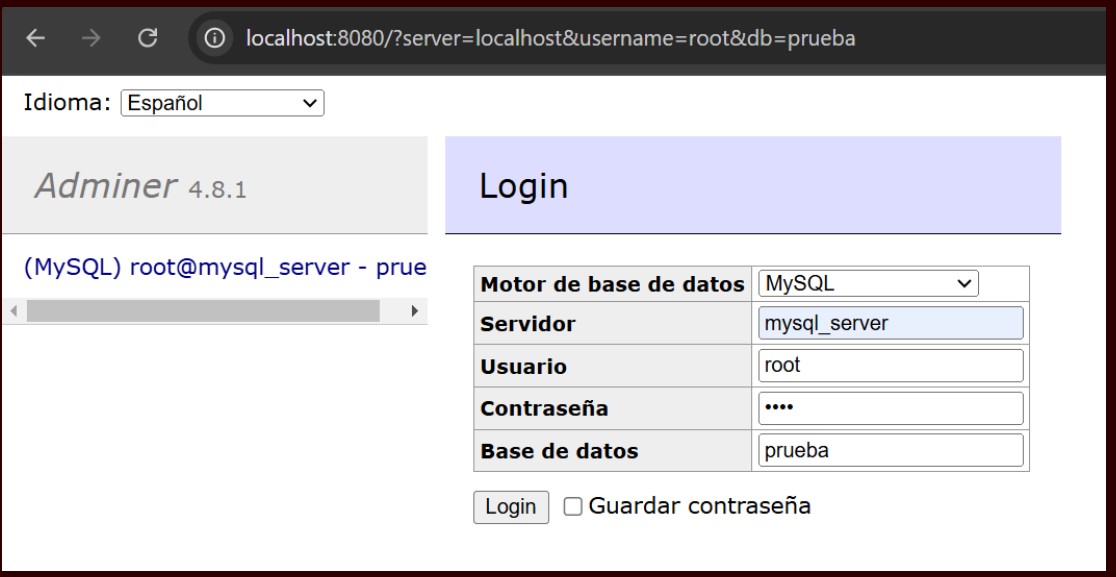
..arcial/Docker

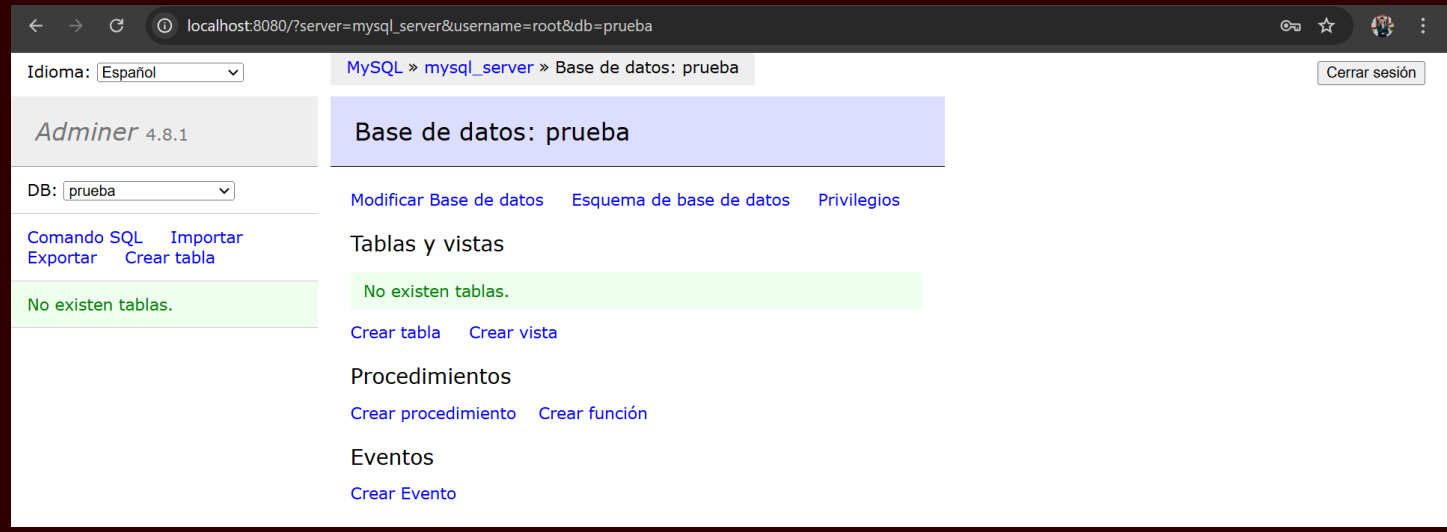
```
→ Docker docker run -dit --name adminer --network red_db -p 8080:8080 adminer

Unable to find image 'adminer:latest' locally
latest: Pulling from library/adminer
c82cd9b427d9: Download complete
798a45c9628c: Download complete
9a4cd7b75371: Download complete
73226dab8db5: Download complete
ed94e1c95a57: Download complete
574dfab7cda2: Download complete
884bce373183: Download complete
Digest: sha256:34d37131366c5aa84e1693dbed48593ed6f95fb450b576c1a7a59d3a9c9e8802
Status: Downloaded newer image for adminer:latest
13edcb77e38747a7c098cec33e892f58de79cb42164605d65a615c5665b30d9e
```

Redes con Docker

R2

Caso práctico: Conectar Adminer con MySQL - comprobar





PHP con Docker

P1

Crear la siguiente estructura del proyecto

..arcial/Docker

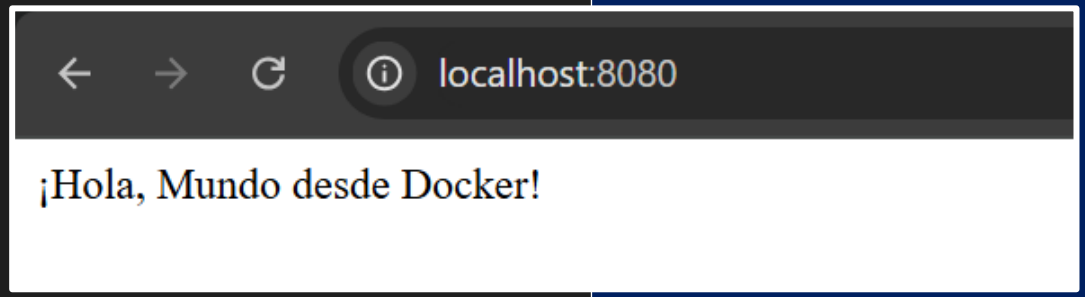
```
→ Docker tree php-docker
php-docker
└── src
    └── index.php

2 directories, 1 file
→ Docker cat php-docker/src/index.php
<?php
echo "¡Hola, Mundo desde Docker!";
→ Docker
```

PHP con Docker

P1

Ejecutar lo siguiente

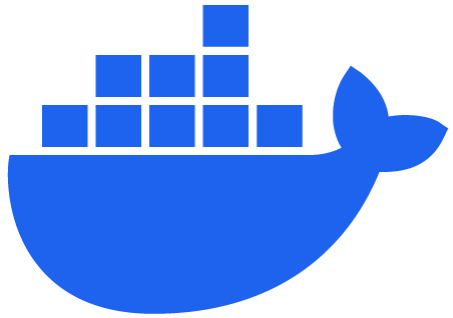
```
→ Docker docker run -d -p 8080:80 -v ./php-docker/src:/var/www/html php:8.1-apache
Unable to find image 'php:8.1-apache' locally
8.1-apache: Pulling from library/php
fd674058ff8f: Download complete
0338d69c00c7: Download complete
79943767d299: Download complete
7e604bd181ad: Download complete
05775c00cf1b: Download complete
b20eabd12e26: Download complete
211155857099: Download complete
f211bd629d11: Download complete
1ff6aa39a0d6: Download complete
4b38a4dd2d71: Download complete
99bdab6367ae: Download complete
4f4fb700ef54: Download complete
26eee4fd24fe: Download complete
d936aca1ab89: Download complete
Digest: sha256:a0e7876e55bc9f790924e112c651c769d7d06b98cfb21bb40406376f5093ed99
Status: Downloaded newer image for php:8.1-apache
397ab9df7ede0da90a4bbe5880e35fc76a6683cb6a97445c79817f73740ac987
```



Explicación del comando:

- *docker run: Este comando crea y ejecuta un contenedor.*
- *-d: Ejecuta el contenedor en segundo plano (modo "detached").*
- *-p 8080:80: Mapea el puerto 8080 de tu máquina local al puerto 80 del contenedor. Esto te permitirá acceder a tu proyecto desde el navegador en http://localhost:8080.*
- *-v ./php-docker/src:/var/www/html: Mapea el directorio ./src de tu máquina local al directorio /var/www/html dentro del contenedor. Esto hace que tu código PHP esté disponible para Apache dentro del contenedor.*
- *php:8.1-apache: Especifica la imagen oficial de PHP con Apache para crear el contenedor.*

Ejercicio chiquitín - Foro



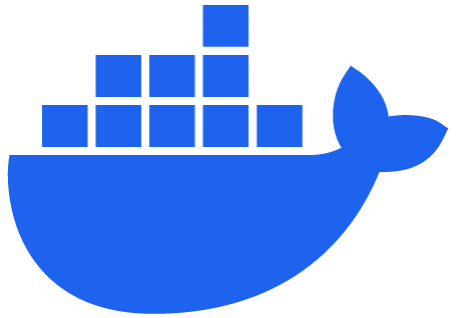
Instalar contenedor de Python

docker pull python
docker image ls
docker run -i -t --rm python

Acceso remoto
docker exec –it <container name> /bin/bash

Comenta en el foro asignado a esta actividad

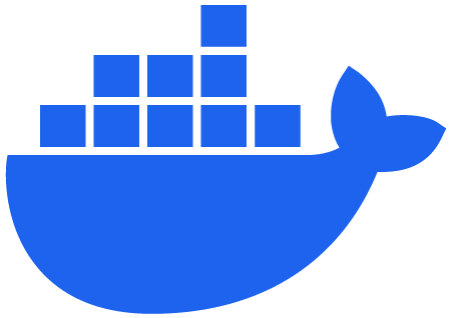
Tarea 1



Investigar cómo eliminar el requerimiento de privilegios root para ejecutar comandos CLI de docker. A modo que se pueda ejecutar el comando de la siguiente manera sin **sudo**. *Si usaste la opción 3 de instalación considera esta tarea como completada.*

docker run hello-world

Tarea 2



Instalar un contenedor de Alpine. Listar sus directorios entrando con el comando **exec** (adjuntar capturas de pantalla completa del ingreso de los comandos en un archivo como evidencia)

Tarea 3

Instalar un contenedor de MySQL y crear la siguiente base de datos mediante. (adjuntar capturas de pantalla completa del ingreso de los comandos en un archivo como evidencia)

Nombre	Apellido	ID. Depto
Alicia	Villareal	2
Blanca	Díaz	2
Daniel	Palacios	2
Víctor	Lemus	5
Karen	Sánchez	5

ID Depto.	Departamento
2	Mercadotecnia
5	Informática

Bibliografía

Sánchez, J. J. (2022). *Aprender Docker – Un enfoque práctico*. Alfaomega.

González, A. (2017). Docker - Guía práctica. RC Libros.

«Home». (2024, 21 mayo). Docker Documentation. https://docs.docker.com