Instalación de docker desktop

22/01/2026
CURSO:2ºDAW

Despliegue
Celia Caravaca Vega

índice

Parte 1: Instalación de Docker

1.1. Docker Engine vs Docker Desktop

Es necesario entender la diferencia entre esto dos.

- Docker Engine:

  Características:

  - Software base y fundamental de Docker
  - Se instala directamente en el sistema operativo
  - Funciona principalmente mediante línea de comandos (CLI)
  - Ligero y eficiente
  - Ideal para servidores Linux
  - Disponible para Linux únicamente (nativo)

  Componentes principales:

  - dockerd: Demonio de Docker (proceso en segundo plano)
  - docker CLI: Interfaz de línea de comandos
  - containerd: Runtime de contenedores
  - runc: Herramienta de bajo nivel para ejecutar contenedores

  Cuándo usar Docker Engine:

  - Servidores Linux en producción
  - Entornos CI/CD
  - Cuando solo necesitas el motor sin interfaz gráfica
  - Recursos limitados (consume menos memoria)

- Docker Desktop:

  Docker Desktop es una aplicación completa que incluye Docker Engine y herramientas adicionales.

  Características:

  - Incluye Docker Engine + herramientas adicionales
  - Interfaz gráfica (GUI) para gestión visual
  - Disponible para Windows, macOS y Linux
  - Integración con el sistema operativo
  - Fácil instalación y configuración
  - Incluye Docker Compose integrado
  - Kubernetes opcional (cluster local)

Cuándo usar Docker Desktop:

- Entornos de desarrollo en Windows o macOS
- Si prefieres interfaz gráfica
- Aprendizaje inicial de Docker
- Desarrollo local de aplicaciones
- Testing rápido de aplicaciones

1.2 Nosotros vamos a utilizar docker desktop :

Paso 1: Habilitar WSL 2

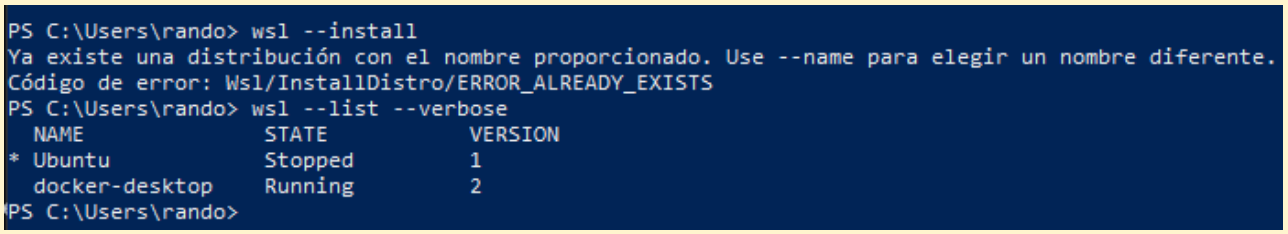
1. Abre PowerShell como Administrador y ejecuta:

   ```
   wsl --install
   ```

2. Reinicia el equipo cuando se te solicite.

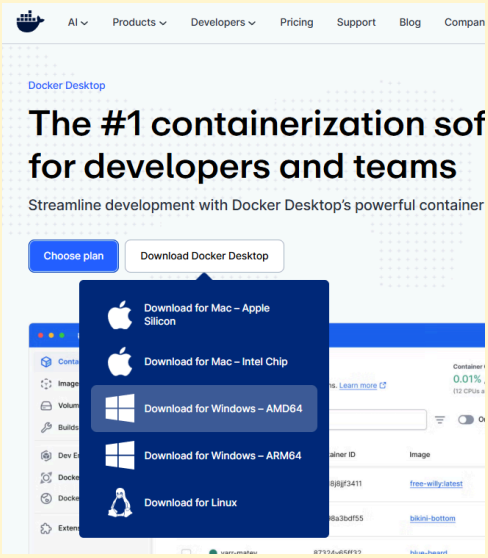
3. Verifica la instalación de WSL 2:

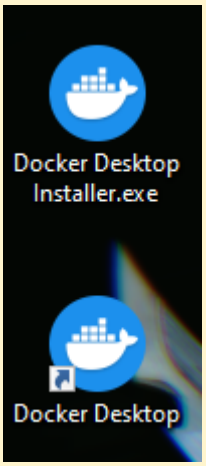
   ```
   wsl --list --verbose
   ```



Paso 2: Descargar Docker Desktop

1. Ve a la página oficial: https://www.docker.com/products/docker-desktop
2. Descarga Docker Desktop para Windows
3. Ejecuta el instalador descargado (Docker Desktop Installer.exe)



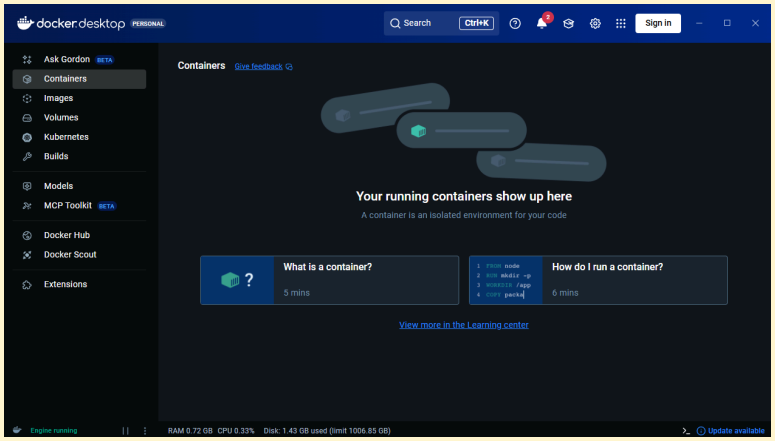


Paso 3: Instalar Docker Desktop

1. Durante la instalación, asegúrate de marcar:
    - Use WSL 2 instead of Hyper-V (recomendado)
    - Add shortcut to desktop (opcional)
2. Completa la instalación y reinicia el equipo si se solicita

Paso 4: Iniciar Docker Desktop

1. Ejecuta Docker Desktop desde el menú de inicio
2. Espera a que el motor de Docker se inicie (verás un icono en la bandeja del sistema)
3. Cuando el icono muestra "Docker Desktop is running", está listo



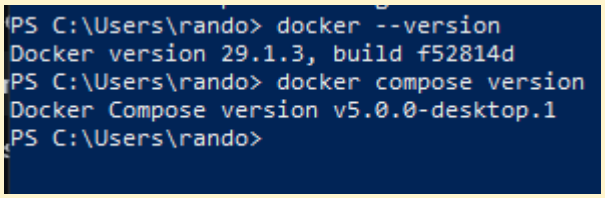
Paso 5: Configuración inicial

1. Docker Desktop te pedirá:
    - Aceptar los términos de servicio
    - Opcionalmente crear/iniciar sesión con una cuenta de Docker Hub
    - Completar una breve encuesta (puedes omitirla)

(Sin querer lo omiti)

Paso 6: Verificar la instalación

Abre PowerShell o Terminal y ejecuta:
docker --version
docker compose version



Paso 7: Probar Docker Desktop

docker run hello-world
Si ves el mensaje de bienvenida, ¡Docker Desktop está funcionando! 🎉(≧▽≦)ﾉ

```
PS C:\Users\rando> docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
17eec7bbc9d7: Pull complete
ea52d2000f90: Download complete
Digest: sha256:05813aedc15fb7b4d732e1be879d3252c1c9c25d885824f6295cab4538cb85cd
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/
```

Parte 2: Primeros pasos con contenedores

2.1. Tu primer contenedor: Hello World

Ya ejecutaste este comando durante la instalación, pero analicémoslo en detalle:
docker run hello-world
¿Qué sucede cuando ejecutas este comando?

1. Docker busca la imagen hello-world localmente
2. Como no la encuentra, la descarga de Docker Hub
3. Crea un contenedor desde esa imagen
4. Ejecuta el contenedor (muestra un mensaje)
5. El contenedor finaliza automáticamente

¿Qué sucede cuando ejecutas este comando?

1. Docker busca la imagen hello-world localmente
2. Como no la encuentra, la descarga de Docker Hub
3. Crea un contenedor desde esa imagen
4. Ejecuta el contenedor (muestra un mensaje)
5. El contenedor finaliza automáticamente

2.2. Ejecutar un contenedor interactivo

Vamos a ejecutar un contenedor con una distribución Linux:

```
docker run -it ubuntu bash
```
¿Qué hace cada parte?

- docker run: Crea y ejecuta un contenedor
- -it: Modo interactivo con terminal
    - -i: Mantiene la entrada estándar abierta
    - -t: Asigna un pseudo-TTY (terminal)
- ubuntu: Imagen a utilizar
- bash: Comando a ejecutar (shell de bash)

¿Qué puedes hacer ahora?
(para salir exit)
Estás dentro de un contenedor Ubuntu. Prueba algunos comandos:

```
 $ 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/

PS C:\Users\rando> docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
PS C:\Users\rando> docker ps -a
CONTAINER ID   IMAGE         COMMAND    CREATED         STATUS
b7c1fed956b5   hello-world   "/hello"   3 minutes ago   Exited (0) 3 minutes ago
PS C:\Users\rando> docker container rm b7
b7
PS C:\Users\rando> docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
PS C:\Users\rando> docker run -it ubuntu bash
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
a3629ac5b9f4: Pull complete
1baf05536e37: Download complete
Digest: sha256:cd1dba651b3080c3686ecf4e3c4220f026b521fb76978881737d24f200828b2b
Status: Downloaded newer image for ubuntu:latest
root@833c38ff8e69:/#
```

```
root@833c38ff8e69:/# pwd
/
root@833c38ff8e69:/# ls
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@833c38ff8e69:/# cd usr
root@833c38ff8e69:/usr# ls
bin  games  include  lib  lib64  libexec  local  sbin  share  src
root@833c38ff8e69:/usr# cd games/
root@833c38ff8e69:/usr/games# ls
root@833c38ff8e69:/usr/games# echo "Aqui no hay juegos :(">mensaje.txt
root@833c38ff8e69:/usr/games# cat mensaje.txt
Aqui no hay juegos :(
root@833c38ff8e69:/usr/games# hostname
833c38ff8e69
root@833c38ff8e69:/usr/games# ls -la
total 16
drwxr-xr-x 1 root root 4096 Jan 22 13:56 .
drwxr-xr-x 1 root root 4096 Jan 13 02:04 ..
-rw-r--r-- 1 root root   22 Jan 22 13:56 mensaje.txt
root@833c38ff8e69:/usr/games# echo "mentira habian cosas pero estaban ocultas"
mentira habian cosas pero estaban ocultas
root@833c38ff8e69:/usr/games# apt get install -y curl
E: Invalid operation get
root@833c38ff8e69:/usr/games# apt-get install -y curl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
E: Unable to locate package curl
root@833c38ff8e69:/usr/games# curl --version
bash: curl: command not found
root@833c38ff8e69:/usr/games# curl --version
bash: curl: command not found
root@833c38ff8e69:/usr/games# apt-get install cowsay
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
E: Unable to locate package cowsay
root@833c38ff8e69:/usr/games# cowsay -m "hola"
bash: cowsay: command not found
root@833c38ff8e69:/usr/games# cowsay
bash: cowsay: command not found
root@833c38ff8e69:/usr/games#
```

2.3. Ejecutar un contenedor en segundo plano

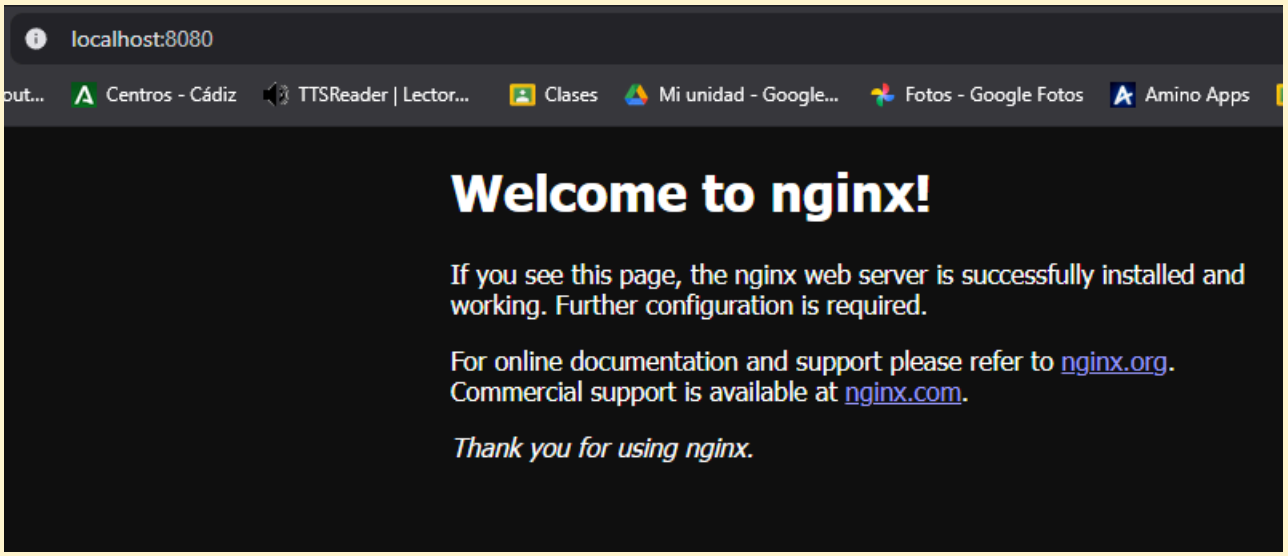
Vamos a ejecutar un servidor web Nginx:
docker run -d -p 8080:80 --name mi-nginx nginx
¿Qué hace cada parte?

- -d: Ejecuta en modo "detached" (segundo plano)
- -p 8080:80: Mapea el puerto 80 del contenedor al puerto 8080 del host
- --name mi-nginx: Asigna un nombre al contenedor
- nginx: Imagen a utilizar

Verificar que funciona:

1. Abre tu navegador web
2. Ve a: http://localhost:8080
3. Deberías ver la página de bienvenida de Nginx

```
PS C:\Users\rando> docker run -d -p 8080:80 --name mi-nginx nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
eaf8753feae0: Pull complete
500799c30424: Pull complete
119d43eec815: Pull complete
700146c8ad64: Pull complete
d989100b8a84: Pull complete
57f0dd1befe2: Pull complete
10b68cfefee1: Pull complete
e2dd2dbe6277: Download complete
785250c9bf9e: Download complete
Digest: sha256:c881927c4077710ac4b1da63b83aa163937fb47457950c267d92f7e4dedf4aec
Status: Downloaded newer image for nginx:latest
49055da7a78b6f33de5e7ccc0e448018b284f947471d35b890cb7876bbd8ee9e
```



2.4. Comandos básicos para gestionar contenedores

Ver contenedores en ejecución

docker ps
Deberías ver tu contenedor de Nginx.
Columnas importantes:

- CONTAINER ID: Identificador único del contenedor
- IMAGE: Imagen utilizada
- COMMAND: Comando ejecutado
- CREATED: Cuándo se creó
- STATUS: Estado actual
- PORTS: Puertos mapeados
- NAMES: Nombre del contenedor

Ver todos los contenedores (incluyendo detenidos)

docker ps -a
Ahora verás también el contenedor de Ubuntu y hello-world que ejecutaste antes.

Ver logs de un contenedor

docker logs mi-nginx
Para ver logs en tiempo real:
docker logs -f mi-nginx
(Presiona Ctrl+C para salir)

Detener un contenedor

docker stop mi-nginx
Verifica que se detuvo:
docker ps
Ya no debería aparecer. Pero si ejecutas docker ps -a, lo verás con estado "Exited".

Iniciar un contenedor detenido

docker start mi-nginx
Verifica que volvió a ejecutarse:
docker ps

Reiniciar un contenedor

docker restart mi-nginx

Ver estadísticas de recursos

docker stats mi-nginx
Verás uso de CPU, memoria, red y disco. Presiona Ctrl+C para salir.

Inspeccionar un contenedor

docker inspect mi-nginx
Muestra toda la información detallada en formato JSON.

Ejecutar comandos dentro de un contenedor en ejecución

docker exec -it mi-nginx bash
Ahora estás dentro del contenedor. Prueba:

```
# Ver procesos
ps aux

# Ver archivos de configuración de Nginx
cat /etc/nginx/nginx.conf

# Salir
exit
```

Eliminar un contenedor

Primero detenlo (si está en ejecución):
docker stop mi-nginx
Luego elimínalo:
docker rm mi-nginx
Verifica que se eliminó:
docker ps -a
Atajo: Forzar eliminación (detiene y elimina en un solo comando):
docker rm -f mi-nginx

```
PS C:\Users\rando> docker run -d -p 8080:80 --name mi-nginx nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
eaf8753feae0: Pull complete
500799c30424: Pull complete
119d43eec815: Pull complete
700146c8ad64: Pull complete
d989100b8a84: Pull complete
57f0dd1befe2: Pull complete
10b68cfefee1: Pull complete
e2dd2dbe6277: Download complete
785250c9bf9e: Download complete
Digest: sha256:c881927c4077710ac4b1da63b83aa163937fb47457950c267d92f7e4dedf4aec
Status: Downloaded newer image for nginx:latest
49055da7a78b6f33de5e7ccc0e448018b284f947471d35b890cb7876bbd8ee9e
PS C:\Users\rando> docker stop 49
49
PS C:\Users\rando> docker container rm 49
49
PS C:\Users\rando>
```

Parte 3: Trabajando con imágenes de Docker Hub

3.1. ¿Qué es Docker Hub?

Docker Hub es el registro público de imágenes Docker. Es como el "GitHub" de las imágenes de contenedores.
URL: https://hub.docker.com
Contiene:

- Imágenes oficiales (mantenidas por Docker y los proyectos)
- Imágenes de la comunidad
- Imágenes privadas (con cuenta de pago)

3.2. Buscar imágenes

Buscar desde la línea de comandos
docker search nginx
Muestra las 25 imágenes más populares relacionadas con "nginx".
Columnas:

- NAME: Nombre de la imagen
- DESCRIPTION: Descripción
- STARS: Popularidad (estrellas)
- OFFICIAL: Si es imagen oficial
- AUTOMATED: Si se construye automáticamente

Buscar en el navegador web

1. Ve a https://hub.docker.com
2. Busca "nginx" en el cuadro de búsqueda
3. Explora la imagen oficial: nginx

Información útil en Docker Hub:

- Descripción y documentación
- Etiquetas (tags) disponibles
- Uso y ejemplos
- Dockerfile (cómo se construyó la imagen)
- Número de descargas

3.3. Descargar imágenes

Descargar una imagen
docker pull nginx
Esto descarga la última versión (latest).
Descargar una versión específica
docker pull nginx:1.25
Descargar diferentes variantes

```
# Versión Alpine (más ligera)
docker pull nginx:alpine

# Versión específica con Alpine
docker pull nginx:1.25-alpine
```

Parte 4: Ejercicios prácticos

Ejercicio 1: Explorar diferentes distribuciones Linux

Objetivo: Ejecutar contenedores de diferentes distribuciones Linux y explorarlas.
Tareas:

1. Ejecuta un contenedor de Debian
   docker run -it debian bash

```
PS C:\Users\rando> docker run -it debian bash
Unable to find image 'debian:latest' locally
latest: Pulling from library/debian
2ca1bfae7ba8: Pull complete
97c1aa41a61f: Download complete
Digest: sha256:5cf544fad978371b3df255b61e209b373583cb88b733475c86e49faa15ac2104
Status: Downloaded newer image for debian:latest
root@8e89d0321df5:/#
```

2. Dentro del contenedor, identifica:
   - La versión del sistema operativo
     cat /etc/os-release

```
root@8e89d0321df5:/# cat etc/os-release
PRETTY_NAME="Debian GNU/Linux 13 (trixie)"
NAME="Debian GNU/Linux"
VERSION_ID="13"
VERSION="13 (trixie)"
VERSION_CODENAME=trixie
DEBIAN_VERSION_FULL=13.3
ID=debian
HOME_URL="https://www.debian.org/"
SUPPORT_URL="https://www.debian.org/support"
BUG_REPORT_URL="https://bugs.debian.org/"
root@8e89d0321df5:/#
```

   - El hostname
     hostname

```
root@8e89d0321df5:/# hostname
8e89d0321df5
```

   - Los procesos
     apt update
     apt install -y procps
     ps aux

```
root@8e89d0321df5:/# ps aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.0   4328  3584 pts/0    Ss   12:37   0:00 bash
root         134  0.0  0.0   6392  3456 pts/0    R+   12:42   0:00 ps aux
root@8e89d0321df5:/#
```

3. Sal del contenedor
   exit

```
root@8e89d0321df5:/# exit
exit
PS C:\Users\rando>
```

4. Repite el proceso con
   - Ubuntu

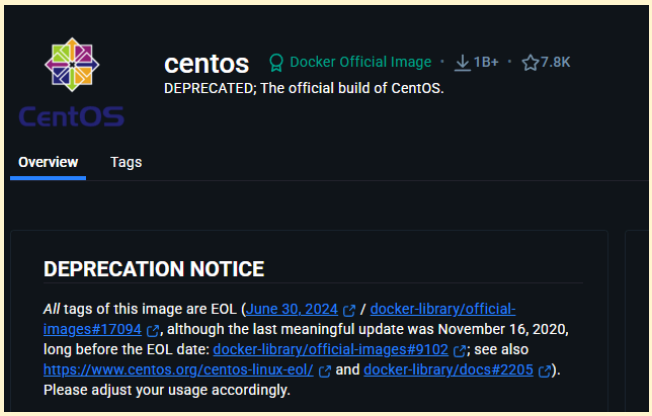
```
PS C:\Users\rando> docker run -it ubuntu bash
root@6f3db306824a:/# cat etc/os-release
PRETTY_NAME="Ubuntu 24.04.3 LTS"
NAME="Ubuntu"
VERSION_ID="24.04"
VERSION="24.04.3 LTS (Noble Numbat)"
VERSION_CODENAME=noble
ID=ubuntu
ID_LIKE=debian
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
UBUNTU_CODENAME=noble
LOGO=ubuntu-logo
root@6f3db306824a:/#
```

```
root@6f3db306824a:/# hostname
6f3db306824a
root@6f3db306824a:/# ps aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.1  0.0   4588  3200 pts/0    Ss   12:43   0:00 bash
root          12  0.0  0.0   7888  3712 pts/0    R+   12:44   0:00 ps aux
root@6f3db306824a:/# exit
exit
```

- Alpine Linux

```
PS C:\Users\rando> docker run -it alpine sh
Unable to find image 'alpine:latest' locally
latest: Pulling from library/alpine
1074353eec0d: Pull complete
5c1f58ba4e0d: Download complete
644afed44dca: Download complete
Digest: sha256:865b95f46d98cf867a156fe4a135ad3fe50d2056aa3f25ed31662dff6da4eb62
Status: Downloaded newer image for alpine:latest
/ # cat /etc/os-release
NAME="Alpine Linux"
ID=alpine
VERSION_ID=3.23.2
PRETTY_NAME="Alpine Linux v3.23"
HOME_URL="https://alpinelinux.org/"
BUG_REPORT_URL="https://gitlab.alpinelinux.org/alpine/aports/-/issues"
/ # hostname
cb8b4e5ca154
/ # ps
PID   USER     TIME  COMMAND
    1 root      0:00 sh
    9 root      0:00 ps
/ # exit
```

- CentOS



Al parecer hay una noticia de que CentOS Linux fue descontinuado y alcanzó su fin de vida (EOL).

5. Compara los tamaños de las imágenes:

Pregunta: ¿Cuál es la distribución más ligera? ¿Por qué crees que es así?

Distribución	Tamaño de la imagen
Alpine Linux	~5–7 MB
Debian	~120 MB
Ubuntu	~70–80 MB
CentOS Stream	~200 MB

Alpine Linux

Porque:

- Usa musl en lugar de glibc (más pequeña)
- Usa BusyBox (herramientas mínimas en un solo binario)
- Incluye solo lo imprescindible
- Está pensada desde el inicio para contenedores

No trae:

- Documentación
- Servicios innecesarios
- Herramientas de escritorio

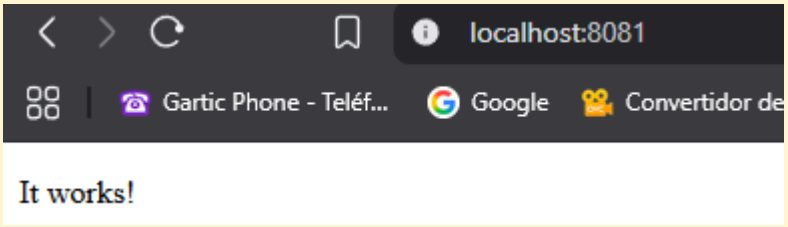
Ejercicio 2: Servidor web Apache

Objetivo: Ejecutar un servidor web Apache y personalizarlo.
Tareas:

1. Ejecuta un contenedor de Apache
   docker run -d -p 8081:80 --name mi-apache httpd

```
PS C:\WINDOWS\system32> docker run -d -p 8081:80 --name mi-apache httpd
Unable to find image 'httpd:latest' locally
latest: Pulling from library/httpd
a062e9825802: Pull complete
4f4fb700ef54: Pull complete
825b27552744: Pull complete
5e136ef74846: Pull complete
e0f36fcb159c: Pull complete
8aec31a82658: Download complete
a4ba44ff3baa: Download complete
Digest: sha256:dd178595edd6d4f49296f62f9587238db2cd1045adfff6fccc15a6c4d08f5d2e
Status: Downloaded newer image for httpd:latest
3c4b43fdbff5af23c058e0c6e87737f53109bda8797a4b0cbe31715ed6234e78
PS C:\WINDOWS\system32>
```

2. Accede a http://localhost:8081 en tu navegador. Deberías ver "It works!"



3. Personaliza la página:
4. docker exec -it mi-apache bash

```
PS C:\WINDOWS\system32> docker exec -it mi-apache bash
root@3c4b43fdbff5:/usr/local/apache2# ls
bin  build  cgi-bin  conf  error  htdocs  icons  include  logs  modules
root@3c4b43fdbff5:/usr/local/apache2# cd htdocs/
root@3c4b43fdbff5:/usr/local/apache2/htdocs# ls
index.html
```

5. Dentro del contenedor:

echo "<h1>Hola desde Docker!</h1>" > /usr/local/apache2/htdocs/index.html
(decidí editarlo a borrarlo todo, instalando nano con el comando:
apt update && apt install -y nano)

```
  GNU nano 8.4                                    index.html
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>It works! Apache httpd</title>
</head>
<body>
<p>It works!</p>
<p>Que en espaniol significa:</p>
<h1>Esta Funcionando!!</h1>
<h3>Y tambien se instalo nano :D</h3>
</body>
</html>
```

6. exit

```
root@3c4b43fdbff5:/usr/local/apache2/htdocs# exit
```

7. Recarga la página en tu navegador. ¿Qué ves ahora?

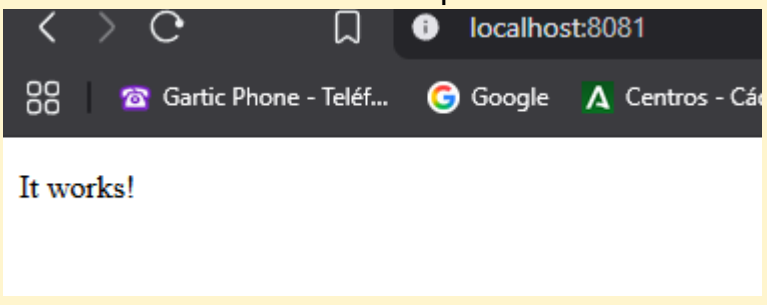
8. Detén y elimina el contenedor

```
PS C:\WINDOWS\system32> docker stop mi-apache
mi-apache
PS C:\WINDOWS\system32> docker container rm  mi-apache
mi-apache
PS C:\WINDOWS\system32> docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
PS C:\WINDOWS\system32>
```

9. Vuelve a crear el contenedor

```
PS C:\WINDOWS\system32> docker run -d -p 8081:80 --name mi-apache httpd
5220bf69987fdc6a04cfee611522b054aa164403281d5c1e4f2dedd354c096b2
PS C:\WINDOWS\system32> docker ps
CONTAINER ID   IMAGE     COMMAND              CREATED         STATUS         PORTS
AMES
5220bf69987f   httpd     "httpd-foreground"   7 seconds ago   Up 5 seconds   0.0.0.0:8081->80/tcp, [::]:8081->80/tcp
i-apache
PS C:\WINDOWS\system32>
```

10. Accede nuevamente a http://localhost:8081



:(

Pregunta: ¿Por qué volvió a aparecer "It works!" en lugar de tu mensaje personalizado?

Porque los docker son volatiles una vez se borran pierden toda la información que tenian.
Al crear uno nuevo se vuelve a lo que tenía la imagen por defecto.

Ejercicio 3: Base de datos MySQL

Objetivo: Ejecutar una base de datos MySQL y conectarte a ella.
Tareas:

1. Ejecuta un contenedor de MySQL:
2. docker run -d -p 3306:3306 --name mi-misql -e MYSQL_ROOT_PASSWORD=mipassword mysql

```
PS C:\WINDOWS\system32> docker run -d -p 3306:3306 --name mi-mysql -e MYSQL_ROOT_PASSWORD=mipassword my
Unable to find image 'mysql:latest' locally
latest: Pulling from library/mysql
d347ea4ed6ec: Download complete
3db79435343e: Download complete
7bfd97e63adb: Download complete
a843ee584592: Downloading [=========================>                         ]  83.89MB/160.6MB
16506d4b4233: Extracting 9 s
b639f2fb9b22: Download complete
013be34edccd: Download complete
f63365c83edc: Download complete
```

3. Espera unos segundos a que MySQL se inicie. Verifica los logs.

```
PS C:\WINDOWS\system32> docker logs mi-misql
2026-01-24 12:28:02+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 9.6.0-1.el9 started.
2026-01-24 12:28:04+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
2026-01-24 12:28:04+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 9.6.0-1.el9 started.
2026-01-24 12:28:05+00:00 [Note] [Entrypoint]: Initializing database files
2026-01-24T12:28:05.272673Z 0 [System] [MY-015017] [Server] MySQL Server Initialization - start.
2026-01-24T12:28:05.286893Z 0 [System] [MY-013169] [Server] /usr/sbin/mysqld (mysqld 9.6.0) initializing of server
ogress as process 80
2026-01-24T12:28:05.502031Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has started.
2026-01-24T12:28:17.013954Z 1 [System] [MY-013577] [InnoDB] InnoDB initialization has ended.
2026-01-24T12:28:43.438178Z 6 [Warning] [MY-010453] [Server] root@localhost is created with an empty password ! Ple
onsider switching off the --initialize-insecure option.
2026-01-24T12:30:02.758099Z 0 [System] [MY-015018] [Server] MySQL Server Initialization - end.
2026-01-24 12:30:02+00:00 [Note] [Entrypoint]: Database files initialized
2026-01-24 12:30:02+00:00 [Note] [Entrypoint]: Starting temporary server
2026-01-24T12:30:02.880128Z 0 [System] [MY-015015] [Server] MySQL Server - start.
2026-01-24T12:30:03.471691Z 0 [System] [MY-010116] [Server] /usr/sbin/mysqld (mysqld 9.6.0) starting as process 117
2026-01-24T12:30:03.471712Z 0 [System] [MY-015590] [Server] MySQL Server has access to 6 logical CPUs.
2026-01-24T12:30:03.471730Z 0 [System] [MY-015590] [Server] MySQL Server has access to 4054560768 bytes of physical
ry.
2026-01-24T12:30:04.116460Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has started.
2026-01-24T12:30:27.589260Z 1 [System] [MY-013577] [InnoDB] InnoDB initialization has ended.
2026-01-24T12:30:38.697726Z 0 [Warning] [MY-010068] [Server] CA certificate ca.pem is self signed.
2026-01-24T12:30:38.697865Z 0 [System] [MY-013602] [Server] Channel mysql_main configured to support TLS. Encrypted
ections are now supported for this channel.
2026-01-24T12:30:38.808406Z 0 [Warning] [MY-011810] [Server] Insecure configuration for --pid-file: Location '/var/
ysqld' in the path is accessible to all OS users. Consider choosing a different directory.
2026-01-24T12:30:39.351819Z 0 [System] [MY-011323] [Server] X Plugin ready for connections. Socket: /var/run/mysqld
lx.sock
2026-01-24T12:30:39.355768Z 0 [System] [MY-010931] [Server] /usr/sbin/mysqld: ready for connections. Version: '9.6.
ocket: '/var/run/mysqld/mysqld.sock'  port: 0  MySQL Community Server - GPL.
2026-01-24 12:30:39+00:00 [Note] [Entrypoint]: Temporary server started.
'/var/lib/mysql/mysql.sock' -> '/var/run/mysqld/mysqld.sock'
Warning: Unable to load '/usr/share/zoneinfo/iso3166.tab' as time zone. Skipping it.
Warning: Unable to load '/usr/share/zoneinfo/leap-seconds.list' as time zone. Skipping it.
Warning: Unable to load '/usr/share/zoneinfo/leapseconds' as time zone. Skipping it.
Warning: Unable to load '/usr/share/zoneinfo/tzdata.zi' as time zone. Skipping it.
Warning: Unable to load '/usr/share/zoneinfo/zone.tab' as time zone. Skipping it.
Warning: Unable to load '/usr/share/zoneinfo/zone1970.tab' as time zone. Skipping it.

2026-01-24 12:30:51+00:00 [Note] [Entrypoint]: Stopping temporary server
2026-01-24T12:30:51.968399Z 11 [System] [MY-013172] [Server] Received SHUTDOWN from user root. Shutting down mysqld
sion: 9.6.0).
2026-01-24T12:30:56.317962Z 0 [System] [MY-010910] [Server] /usr/sbin/mysqld: Shutdown complete (mysqld 9.6.0)  MyS
```

4. Conéctate a MySQL:
5. docker exec -it mi-misql mysql -uroot -pmipassword

```
PS C:\WINDOWS\system32> docker exec -it mi-misql mysql -uroot -pmipassword
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 9.6.0 MySQL Community Server - GPL

Copyright (c) 2000, 2026, 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>
```

6. Dentro de MySQL, ejecuta:

SHOW DATABASES;
CREATE DATABASE prueba;
USE prueba;
CREATE TABLE usuarios (id INT, nombre VARCHAR(50));
INSERT INTO usuarios VALUES (1, 'Juan');
SELECT * FROM usuarios;

```
mysql> SHOW DATABASES;
+--------------------+ a password on the command line interface can be insecure.
| Database           |onitor.  Commands end with ; or \g.
+--------------------+id is 9
| information_schema |MySQL Community Server - GPL
| mysql              |
| performance_schema |26, Oracle and/or its affiliates.
| sys                |
+--------------------+ trademark of Oracle Corporation and/or its
4 rows in set (0.078 sec)ay be trademarks of their respective
owners.
mysql> CREATE DATABASE prueba;
Query OK, 1 row affected (0.215 sec)'\c' to clear the current input statement.

mysql> USE prueba
Database changed
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| prueba             |
| sys                |
+--------------------+
5 rows in set (0.002 sec)
```

EXIT;

```
mysql> CREATE TABLE usuarios  (id int , juan varchar(10));
Query OK, 0 rows affected (0.548 sec)

mysql> INSERT INTO usuarios VALUES (1,'juan');
Query OK, 1 row affected (0.128 sec)

mysql> SELECT  * FROM usuarios;
+------+------+
| id   | juan |
+------+------+
|    1 | juan |
+------+------+
1 row in set (0.001 sec)

mysql> Exit
Bye
PS C:\WINDOWS\system32>
```

7. Detén y elimina el contenedor:

docker stop mi-mysql

8. docker rm mi-mysql

```
PS C:\WINDOWS\system32> docker stop mi-misql
mi-misql
PS C:\WINDOWS\system32> docker rm mi-misql
mi-misql
PS C:\WINDOWS\system32>
```

9. Vuelve a crear el contenedor con el mismo comando del paso 1.
10. Conéctate nuevamente y verifica:
11. docker exec -it mi-mysql mysql -uroot -pmipassword
12. Dentro de MySQL:
13. SHOW DATABASES;

```
PS C:\WINDOWS\system32> docker run -d -p 3306:3360 --name mi-misql -e MYSQL_ROOT_PASSWORD=mipassword mysql
3722e6545ec1e03d6d10a9ed3388e675057bc7d60fb2bc4cc7aba1ab938b2208
PS C:\WINDOWS\system32> docker exec -it mi-mysql mysql -uroot -pmipassword
Error response from daemon: No such container: mi-mysql
PS C:\WINDOWS\system32> docker exec -it mi-misql mysql -uroot -pmipassword
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 9.6.0 MySQL Community Server - GPL

Copyright (c) 2000, 2026, 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> SHOW DATABASES
    ->
    -> ;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.037 sec)
```

Pregunta: ¿Existe la base de datos "prueba"? ¿Por qué?

NO, Como he dicho anteriormente los docker son volatiles y lo que acabamos de hacer es eliminarlo y pilla la imagen por defecto.

Ejercicio 4: Explorar Docker Hub

Objetivo: Familiarizarse con Docker Hub y encontrar imágenes útiles.
Tareas:

1. Ve a https://hub.docker.com en tu navegador.

2. Busca las siguientes imágenes oficiales y explora su documentación:
    - postgres (base de datos PostgreSQL)
    - redis (base de datos en memoria)
    - node (entorno de ejecución Node.js)
    - python (intérprete de Python)
3. Para cada imagen, documenta:
    - ¿Cuántas descargas tiene?
    - ¿Qué etiquetas (tags) principales tiene?
    - ¿Qué puerto utiliza por defecto?
    - ¿Qué variables de entorno importantes acepta?

- **Postgres (base de datos PostgreSQL)**

¿Cuántas descargas tiene?

- Esta semana
- Pulls: 20,412,897
- En general 1B+

¿Qué etiquetas (tags) principales tiene?

- 18.1, 18, latest, 18.1-trixie, 18-trixie, trixie
- 18.1-bookworm, 18-bookworm, bookworm
- 18.1-alpine3.23, 18-alpine3.23, alpine3.23, 18.1-alpine, 18-alpine, alpine
  Estas de aquí.

¿Qué puerto utiliza por defecto?

- Puerto por defecto: 5432

¿Qué variables de entorno importantes acepta?

- POSTGRES_PASSWORD (Para la contraseña)
- POSTGRES_USER (Para el usuario)
- POSTGRES_DB (Para la base de datos)

- Redis (base de datos en memoria)

¿Cuántas descargas tiene?

- Pulls totales: 1B+

¿Qué etiquetas (tags) principales tiene?

- latest
- 7, 7.2
- alpine
- bookworm

¿Qué puerto utiliza por defecto?

- 6379

¿Qué variables de entorno importantes acepta?

Redis no depende mucho de variables, pero las más usadas son:

- REDIS_PASSWORD (si se configura seguridad)
- Configuración normalmente se hace por archivo o comando

- Node (entorno de ejecución Node.js)

¿Cuántas descargas tiene?

- Pulls totales: 1B+

¿Qué etiquetas (tags) principales tiene?

- latest
- 20, 18
- alpine
- slim
- bookworm

¿Qué puerto utiliza por defecto?

Node no tiene un puerto fijo
Depende de la aplicación
(normalmente 3000, pero no es obligatorio)

¿Qué variables de entorno importantes acepta?

- NODE_ENV (development / production)
- PORT (si la app lo usa)

- Python (intérprete de Python)

¿Cuántas descargas tiene?

- Pulls totales: 1B+

¿Qué etiquetas (tags) principales tiene?

- latest
- 3.12, 3.11
- slim
- alpine
- bookworm

¿Qué puerto utiliza por defecto?

Python no usa puerto por defecto
Solo usa puerto si ejecutas un servidor (Flask, Django, etc.)

¿Qué variables de entorno importantes acepta?

- PYTHONUNBUFFERED
- PYTHONDONTWRITEBYTECODE
- PYTHONPATH

4. Descarga la imagen de Redis
   docker pull redis

```
PS C:\WINDOWS\system32> docker pull redis
Using default tag: latest
latest: Pulling from library/redis
9c6dc2f051d0: Download complete
4e5d87291b59: Downloading [==========================
b308a2348d28: Download complete
```

```
PS C:\WINDOWS\system32> docker images

IMAGE                ID             DISK USAGE   CONTENT SIZE   EXTRA
alpine:latest        865b95f46d98       13.1MB         3.95MB
debian:latest        5cf544fad978        186MB         52.5MB
hello-world:latest   05813aedc15f       25.9kB         9.52kB
httpd:latest         dd178595edd6        177MB         47.6MB
mysql:latest         6b18d01fb632       1.27GB          283MB
nginx:latest         c881927c4077        240MB         65.7MB
redis:latest         73dad4271642        205MB         55.4MB
ubuntu:latest        cd1dba651b30        119MB         31.7MB
PS C:\WINDOWS\system32>
```

5. Ejecuta un contenedor de Redis

```
PS C:\WINDOWS\system32> docker run -d --name mi-redis redis
167abd1293f98c827d201c64025b5280dc42eb783c96f889ed0700d72b40312e
PS C:\WINDOWS\system32>
```

6. Conéctate al cliente de Redis

```
PS C:\WINDOWS\system32> docker exec -it mi-redis redis-cli
127.0.0.1:6379>
```

7. Prueba algunos comandos de Redis:

SET nombre "Docker"
GET nombre
PING
EXIT

```
PS C:\WINDOWS\system32> docker exec -it mi-redis redis-cli
127.0.0.1:6379> SET nombre "Docker"
OK
127.0.0.1:6379> GET nombre
"Docker"
127.0.0.1:6379> PING
PONG
127.0.0.1:6379> EXIT
PS C:\WINDOWS\system32>
```

PONG ?
El comando PING se utiliza para comprobar que el servidor Redis está activo y responde. Redis responde con PONG como confirmación, siguiendo una convención clásica en sistemas cliente-servidor.
(Me parecio curioso la forma de comprobar el servidor)

8. Limpia todo

Ejercicio 5: Comparar versiones de imágenes

Objetivo: Entender las diferencias entre versiones y variantes de imágenes.
Tareas:

1. Descarga diferentes versiones de Python:

docker pull python:3.12
docker pull python:3.11
docker pull python:3.12-slim
docker pull python:3.12-alpine

```
PS C:\WINDOWS\system32> docker pull  python:3.12
3.12: Pulling from library/python
87ff098136a4: Pull complete
a50bcfa473bb: Pull complete
82e18c5e1c15: Pull complete
40aeb44508af: Pull complete
26d823e3848f: Pull complete
be442a7e0d6f: Pull complete
56ac5f46a5d3: Download complete
9ffbe20d1c67: Download complete
Digest: sha256:746223db759d2cbbde1e066cb49126f460bce66d22147e540ba604438d2296ed
Status: Downloaded newer image for python:3.12
docker.io/library/python:3.12
PS C:\WINDOWS\system32>
PS C:\WINDOWS\system32> docker pull  python:3.11
3.11: Pulling from library/python
f8979860b21b: Download complete
ee3ec68f096e: Pull complete
ce91906e19f2: Pull complete
67374e68056b: Pull complete
5b125ff9357e: Download complete
Digest: sha256:8999a3987fa4c44ed479c07a8fe6921a9168454d4b9b49c3f41fda5db3beb4fa
Status: Downloaded newer image for python:3.11
docker.io/library/python:3.11
PS C:\WINDOWS\system32> docker pull  python:3.12-slim
3.12-slim: Pulling from library/python
ac0d4eeeff34: Download complete
3d6ef8a4ce0a: Pull complete
671677b67e76: Pull complete
83e2eb8c4c73: Pull complete
f4bbd85b300a: Download complete
Digest: sha256:5e2dbd4bbdd9c0e67412aea9463906f74a22c60f89eb7b5bbb7d45b66a2b68a6
Status: Downloaded newer image for python:3.12-slim
docker.io/library/python:3.12-slim
PS C:\WINDOWS\system32> docker pull python:3.12-alpine
3.12-alpine: Pulling from library/python
9b37356de03a: Download complete
a4581c766c38: Pull complete
9e4742745279: Pull complete
522560f13f9b: Pull complete
e2b601d5578e: Download complete
Digest: sha256:68d81cd281ee785f48cdadecb6130d05ec6957f1249814570dc90e5100d3b146
Status: Downloaded newer image for python:3.12-alpine
docker.io/library/python:3.12-alpine
```

2. Compara los tamaños:
3. Ejecuta cada versión y verifica el tamaño real:

docker run -it --rm python:3.12 python --version
docker run -it --rm python:3.12-slim python --version
docker run -it --rm python:3.12-alpine python --version

```
PS C:\WINDOWS\system32> docker run -it --rm python:3.12 python --version
Python 3.12.12
PS C:\WINDOWS\system32> docker run -it --rm python:3.12-slim python --version
Python 3.12.12
PS C:\WINDOWS\system32> docker run -it --rm python:3.12-alpine python --version
Python 3.12.12
PS C:\WINDOWS\system32>
```

4. Entra en un contenedor Alpine y explora:
5. docker run -it python:3.12-alpine sh

```
PS C:\WINDOWS\system32> docker run -it python:3.12-alpine sh
```

6. Dentro del contenedor:

Ver el tamaño de los directorios
du -sh /*

```
/ # du -sh /*
792.0K  /bin
0       /dev
1.2M    /etc
4.0K    /home
756.0K  /lib
16.0K   /media
4.0K    /mnt
4.0K    /opt
0       /proc
8.0K    /root
8.0K    /run
124.0K  /sbin
4.0K    /srv
0       /sys
4.0K    /tmp
50.2M   /usr
72.0K   /var
```

Ver paquetes instalados
apk list
Salir
exit

```
/ # apk list
WARNING: opening from cache https://dl-cdn.alpinelinux.org/alpine/v3.23/main/x86_64/APKINDEX.tar.gz: No such file or directory
WARNING: opening from cache https://dl-cdn.alpinelinux.org/alpine/v3.23/community/x86_64/APKINDEX.tar.gz: No such file or directory
.python-rundeps-20251218.004441 noarch {.python-rundeps} () [installed]
alpine-baselayout-3.7.1-r8 x86_64 {alpine-baselayout} (GPL-2.0-only) [installed]
alpine-baselayout-data-3.7.1-r8 x86_64 {alpine-baselayout} (GPL-2.0-only) [installed]
alpine-keys-2.6-r0 x86_64 {alpine-keys} (MIT) [installed]
alpine-release-3.23.2-r0 x86_64 {alpine-base} (MIT) [installed]
apk-tools-3.0.3-r1 x86_64 {apk-tools} (GPL-2.0-only) [installed]
busybox-1.37.0-r30 x86_64 {busybox} (GPL-2.0-only) [installed]
busybox-binsh-1.37.0-r30 x86_64 {busybox} (GPL-2.0-only) [installed]
ca-certificates-20251003-r0 x86_64 {ca-certificates} (MPL-2.0 AND MIT) [installed]
ca-certificates-bundle-20251003-r0 x86_64 {ca-certificates} (MPL-2.0 AND MIT) [installed]
gdbm-1.26-r0 x86_64 {gdbm} (GPL-3.0-or-later) [installed]
keyutils-libs-1.6.3-r4 x86_64 {keyutils} (GPL-2.0-or-later AND LGPL-2.0-or-later) [installed]
krb5-conf-1.0-r2 x86_64 {krb5-conf} (MIT) [installed]
krb5-libs-1.22.1-r0 x86_64 {krb5} (MIT) [installed]
libapk-3.0.3-r1 x86_64 {apk-tools} (GPL-2.0-only) [installed]
libbz2-1.0.8-r6 x86_64 {bzip2} (bzip2-1.0.6) [installed]
libcom_err-1.47.3-r0 x86_64 {e2fsprogs} (GPL-2.0-or-later AND LGPL-2.0-or-later AND BSD-3-Clause AND MIT) [installed]
libcrypto3-3.5.4-r0 x86_64 {openssl} (Apache-2.0) [installed]
libffi-3.5.2-r0 x86_64 {libffi} (MIT) [installed]
libintl-0.24.1-r1 x86_64 {gettext} (LGPL-2.1-or-later) [installed]
libncursesw-6.5_p20251123-r0 x86_64 {ncurses} (X11) [installed]
libnsl-2.0.1-r1 x86_64 {libnsl} (LGPL-2.0-or-later) [installed]
libpanelw-6.5_p20251123-r0 x86_64 {ncurses} (X11) [installed]
libssl3-3.5.4-r0 x86_64 {openssl} (Apache-2.0) [installed]
libtirpc-1.3.5-r1 x86_64 {libtirpc} (BSD-3-Clause) [installed]
libtirpc-conf-1.3.5-r1 x86_64 {libtirpc} (BSD-3-Clause) [installed]
libuuid-2.41.2-r0 x86_64 {util-linux} (BSD-3-Clause) [installed]
libverto-0.3.2-r2 x86_64 {libverto} (MIT) [installed]
musl-1.2.5-r21 x86_64 {musl} (MIT) [installed]
musl-utils-1.2.5-r21 x86_64 {musl} (MIT AND BSD-2-Clause AND GPL-2.0-or-later) [installed]
ncurses-terminfo-base-6.5_p20251123-r0 x86_64 {ncurses} (X11) [installed]
readline-8.3.1-r0 x86_64 {readline} (GPL-3.0-or-later) [installed]
scanelf-1.3.8-r2 x86_64 {pax-utils} (GPL-2.0-only) [installed]
sqlite-libs-3.51.1-r0 x86_64 {sqlite} (blessing) [installed]
ssl_client-1.37.0-r30 x86_64 {busybox} (GPL-2.0-only) [installed]
tzdata-2025c-r0 x86_64 {tzdata} (Public-Domain) [installed]
xz-libs-5.8.1-r0 x86_64 {xz} (GPL-2.0-or-later AND 0BSD AND Public-Domain AND LGPL-2.1-or-later) [installed]
zlib-1.3.1-r2 x86_64 {zlib} (Zlib) [installed]
/ # exit
PS C:\WINDOWS\system32>
```

Pregunta: ¿Cuál es la diferencia de tamaño entre las versiones? ¿Cuándo usarías cada una?

python:3.12 (normal)

- La más grande
- Basada en Debian/Bookworm
- Trae:
    - Muchas librerías del sistema
    - Herramientas de compilación
    - Más compatibilidad

Tamaño aproximado: 1.61GB

Cuándo usarla

- Desarrollo
- Cuando necesitas instalar muchas dependencias
- Cuando algo falla en slim/alpine

python:3.12-slim

- Más pequeña que la normal
- Sigue usando Debian
- Quita cosas innecesarias (docs, herramientas extra)

Tamaño: 179.07MB

Cuándo usarla

- Producción
- Apps normales
- Buen equilibrio tamaño / compatibilidad

python:3.12-alpine

- La más ligera
- Usa Alpine Linux (musl en vez de glibc)
- Muy pocos paquetes instalados

Tamaño: 74.94MB

Cuándo usarla

- Contenedores muy ligeros
- Microservicios
- Cuando sabes EXACTAMENTE qué necesitas

OJO: Puede dar problemas con librerías Python nativas.

Ejercicio 6: Limpieza del sistema

Objetivo: Aprender a limpiar recursos de Docker.
Tareas:

1. Detén todos los contenedores en ejecución:

docker stop $(docker ps -q)
Por tema de versión:
docker ps -q | ForEach-Object { docker stop $_ }

```
PS C:\WINDOWS\system32> docker stop $(docker ps -q)
docker: 'docker stop' requires at least 1 argument

Usage:  docker stop [OPTIONS] CONTAINER [CONTAINER...]

See 'docker stop --help' for more information
PS C:\WINDOWS\system32> docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
PS C:\WINDOWS\system32> docker ps -a
CONTAINER ID   IMAGE                COMMAND   CREATED       STATUS                     PORTS     NAMES
b8726aa40108   python:3.12-alpine   "sh"      2 hours ago   Exited (0) 2 hours ago             wizardly_brattain
PS C:\WINDOWS\system32> docker rm b8
b8
PS C:\WINDOWS\system32> docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
PS C:\WINDOWS\system32> clear
```

2. Elimina todos los contenedores:

docker rm $(docker ps -aq)
docker ps -q | ForEach-Object { docker rm $_ }

```
PS C:\WINDOWS\system32> docker ps -q | ForEach-Object { docker rm $_ }
PS C:\WINDOWS\system32> docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
PS C:\WINDOWS\system32>
```

3. Elimina todas las imágenes:

docker rmi $(docker images -q)

```
PS C:\WINDOWS\system32> docker rmi $(docker images -q)
Untagged: mysql:latest
Deleted: sha256:6b18d01fb632c0f568ace1cc1ebffb42d1d21bc1de86f6d3e8b7eb18278444d9
Untagged: python:3.11
Deleted: sha256:8999a3987fa4c44ed479c07a8fe6921a9168454d4b9b49c3f41fda5db3beb4fa
Untagged: python:3.12
Deleted: sha256:746223db759d2cbbde1e066cb49126f460bce66d22147e540ba604438d2296ed
Untagged: ubuntu:latest
Deleted: sha256:cd1dba651b3080c3686ecf4e3c4220f026b521fb76978881737d24f200828b2b
Untagged: python:3.12-slim
Deleted: sha256:5e2dbd4bbdd9c0e67412aea9463906f74a22c60f89eb7b5bbb7d45b66a2b68a6
```

4. Limpieza completa del sistema (cuidado, esto elimina todo):
5. docker system prune -a
6. Verifica que todo está limpio:

docker ps -a

```
PS C:\WINDOWS\system32> docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
```

```
PS C:\WINDOWS\system32> docker system prune -a
WARNING! This will remove:
  - all stopped containers
  - all networks not used by at least one container
  - all images without at least one container associated to them
  - all build cache

Are you sure you want to continue? [y/N] y
Total reclaimed space: 0B
PS C:\WINDOWS\system32>
```

docker images

```
PS C:\WINDOWS\system32> docker images

IMAGE   ID             DISK USAGE   CONTENT SIZE   EXTRA
```

Parte 5: Documentación de la práctica

Sección 1: Instalación

- ¿Qué sistema operativo usas?
  Windows 10
- ¿Instalaste Docker Engine o Docker Desktop? ¿Por qué?
  Instalé Docker Desktop porque incluye Docker Engine y muchas otras funciones, lo que me permite experimentar y aprender más sobre Docker de manera completa.

Sección 2: Contenedores

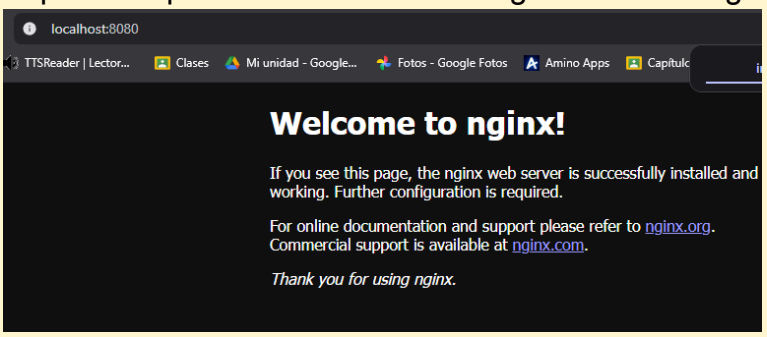
- Captura de pantalla de docker ps mostrando al menos 2 contenedores en ejecución

```
PS C:\Users\rando> docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS                                   NAMES
84ee5b96b4f7   nginx     "/docker-entrypoint.…"   3 minutes ago   Up 3 minutes   0.0.0.0:8080->80/tcp, [::]:8080->80/tcp   tercero-nginx
PS C:\Users\rando> docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS                     PORTS                                   NAMES
84ee5b96b4f7   nginx     "/docker-entrypoint.…"   3 minutes ago   Up 3 minutes               0.0.0.0:8080->80/tcp, [::]:8080->80/tcp   tercero-nginx
dd9481115abf   python    "python3"                5 minutes ago   Exited (0) 4 minutes ago                                           segundo-py
c3d0a0323b87   mysql     "docker-entrypoint.s…"   7 minutes ago   Exited (0) 7 minutes ago                                           primero-sql
PS C:\Users\rando>
```

- Captura de pantalla de docker logs de uno de tus contenedores

```
PS C:\Users\rando> docker logs segundo-py
PS C:\Users\rando> docker logs primero-sql
PS C:\Users\rando>
```

- Captura de pantalla accediendo a Nginx en el navegador



Sección 3: Imágenes

- Captura de pantalla de docker images mostrando al menos 5 imágenes diferentes

```
PS C:\WINDOWS\system32> docker images

IMAGE                ID             DISK USAGE   CONTENT SIZE   EXTRA
alpine:latest        865b95f46d98       13.1MB         3.95MB
debian:latest        5cf544fad978        186MB         52.5MB
hello-world:latest   05813aedc15f       25.9kB         9.52kB
httpd:latest         dd178595edd6        177MB         47.6MB
mysql:latest         6b18d01fb632       1.27GB          283MB
nginx:latest         c881927c4077        240MB         65.7MB
redis:latest         73dad4271642        205MB         55.4MB
ubuntu:latest        cd1dba651b30        119MB         31.7MB
PS C:\WINDOWS\system32>
```

- Tabla comparativa de tamaños de imágenes de Python (Ejercicio 5)
  En el eje 5 está.

Sección 4: Ejercicios

Para cada ejercicio (1-5):

- Todos los comandos ejecutados
- Capturas de pantalla de los resultados
- Respuesta a las preguntas planteadas

Sección 5: Reflexión

Responde a las siguientes preguntas:

1. Diferencias entre contenedores y máquinas virtuales:
    - ¿Qué diferencias principales observas?
        - Los docker no utilizan SO completos como las máquinas virtuales, sino que usan el del host.
        - La estructura de cada uno.
    - ¿Qué ventajas tiene Docker sobre las VMs tradicionales?
        - Ventajas:
          Rapidez, flexibilidad,
2. Persistencia de datos:
    - ¿Qué pasa con los datos cuando eliminas un contenedor?
        - Como los contenedores son volatiles se pierde toda la información que hay en ellos.
    - ¿Cómo crees que se podría solucionar este problema?
        - Utilizando los volúmenes para almacenar los datos.
3. Imágenes:
    - ¿Por qué las imágenes Alpine son más pequeñas?
        - Porque se tienen solamente lo primordial para funcionar.
    - ¿Cuándo usarías una imagen completa vs. una imagen slim o alpine?
        - Cuando quiera un programa grande que tenga más dependencias.
        - Cuando sepa exactamente lo que quiero que voy a necesitar sin más cosas o para docker de microservicios.
4. Uso de Docker:
    - Menciona 3 casos de uso donde Docker sería útil
        - Desplegar una base de datos.
        - Contener el forntend.
        - Gestionar el backend.
    - ¿Qué desventajas o limitaciones has encontrado?
        - Por ahora ninguna.
          Pero la complejidad de los puertos y redes puede ser complejo para los que empiezan.