Tarefas UD05
Bloque 01

Administración de sistemas operativos

Unidade Didáctica 05:
Integración de sistemas operativos en redes libres e privativas

Nome: Rubén Apelidos: Rey Feal

Data:

Índice

Tarefa 1. Cuestións iniciais sobre CIFS

Contesta ás seguintes cuestións:

1. Que é SMB e para que se emprega?

SMB (*Server Message Block*) é un protocolo de rede que permite compartir ficheiros, impresoras e outros recursos entre dispositivos nunha rede. Úsase principalmente en sistemas Windows, pero tamén está dispoñible en Linux e macOS.

2. A que nivel OSI traballa?

SMB traballa no **nivel de sesión** (Capa 5) do modelo OSI, pero tamén pode considerarse parte da Capa 7 (Aplicación), xa que se executa sobre protocolos como NetBIOS ou TCP/IP.

3. Que tipo e número de porto emprega?

Porto **445** TCP: Para comunicación SMB directa sobre TCP/IP.

Portos **137-139** TCP/UDP: Usados en versións máis antigas de SMB que empregaban NetBIOS.

4. Son o mesmo Samba que SMB? Que os diferencia, e cales son os seus orixes?

Non son o mesmo.

SMB é o protocolo desenvolvido por IBM nos anos 80 e adoptado por Microsoft en Windows.

Samba é unha implementación de código aberto de SMB, desenvolvida para permitir a comunicación entre sistemas UNIX/Linux e Windows.

5. Para que se pode empregar Samba?

Compartición de ficheiros e impresoras entre Linux e Windows.

Autenticación de usuarios nun dominio de Windows.

Actuar como controlador de dominio (AD) nunha rede.

6. Que servizos e protocolos implementa Samba?

SMB/CIFS: Para compartir ficheiros e impresoras.

NetBIOS sobre TCP/IP: Para compatibilidade con versións antigas.

LDAP, Kerberos e Active Directory: Para autenticación e xestión de usuarios.

WS-Discovery: Para detección de dispositivos na rede.

Tarefa 2. Compartir por SMB (opcional)

2.1. Compartir de Windows a GNU/Linux

Completa o seguintes enunciados, describindo brevemente os pasos ou comandos empregados facendo as capturas de pantalla que os demostre.

1. Crea un cartafol chamado `c:\publico` nun sistema operativo Windows. Comparte ese cartafol, con permisos de escritura e lectura para «Todos os usuarios» e comparte o recurso co nome **`public`**. Crea dentro do cartafol un ficheiro un chamado `samba-connection.txt`.
2. Logo dende un GNU/Linux da mesma rede, accede a ese recurso empregando un explorador de ficheiros ou un navegador.
3. Empregando o comando axeitado, monta no cartafol `/home/publicwin` o recurso public do equipo de Windows. Debe montarse con opcións de escritura e lectura. Escribe dentro do ficheiro `samba-connection.txt` e mostra logo o seu contido a través do terminal.
4. Finalmente fai que o paso anterior sexa automático durante o inicio do sistema operativo GNU/Linux. Indica o ficheiro tocado e a sintaxe empregada. Unha vez feito, usa o comando `mount -a` para forzar o seu montaxe e finalmente o comando `mount` onde se mostre que está montado.

2.2. Compartir de GNU/Linux a Windows

Agora vaise facer o proceso contrario ao anterior. Compartir un cartafol por en GNU/Linux empregando o protocolo CIFS e logo acceder a el dende unha máquina con Windows. Completa o seguintes enunciados, describindo brevemente os pasos ou comandos empregadose facendo as capturas de pantalla que os demostre.

1. Indica inicialmente os paquete necesarios a instalar para compartir recursos con Samba.
2. Comparte o cartafol `/home/publicgnu` como recurso denominado `linux`. Indica as liñas configuradas no ficheiro de configuración correpondente.
3. Establece un usuario e unha contrasinal para ser usando para acceder ao recurso compartido anteriormente.

4. Reinicia o servizo de Samba.
5. Accede ao recurso dende un equipo de Windows. Crea un ficheiro dentro e comproba que é posible facelo.
6. Finalmente, monta ese recurso na unidade G: de forma premanente.

Tarefa 3. Contedores con Docker

3.1. Cuestións iniciais sobre contedores

Explicar brevemente en que consiste a tecnoloxía de contedores e como Docker fai uso dela. Determinar cal é a diferenza entre un contedor e unha máquina virtual.

Os contedores permiten empaquetar aplicacións e dependencias nunha unidade portátil que comparte o kernel do SO host. Docker utiliza esta tecnoloxía para facilitar a creación, distribución e execución de contedores. A diferenza principal con máquinas virtuais é que estas virtualizan un sistema operativo completo, mentres que os contedores só illan aplicacións a nivel do SO.

Enumerar tres casos de uso comúns de Docker en contornos de desenvolvemento e produción.

- Entornos de desenvolvemento consistentes para equipos.
- Despregue de aplicacións en microservizos.
- Automatización de probas e integración continua (CI/CD).

Indicar as principais diferencias entre Docker Desktop e Docker CE. Tratar de dar resposta a cuestións coma dispoñibilidade para diversos sistemas operativos, se é software libre, se é preciso adquirir unha licenza de uso ou que funcionalidades aporta cada unha das dúas.

Docker Desktop é unha ferramenta completa para Windows e macOS que inclúe interface gráfica, mentres que Docker CE é unha versión de liña de comandos dispoñible para Linux. Docker Desktop require unha licenza para uso empresarial, e Docker CE é software libre.

Describir o papel que xoga o Docker Daemon dentro da arquitectura de Docker.

O Docker Daemon é o servizo en segundo plano que xestiona a creación, execución e supervisión de contedores. Recibe comandos da CLI ou API de Docker e controla os recursos do sistema.

Explicar a diferenza entre unha imaxe de Docker e un contedor de Docker.

Unha imaxe de Docker é un ficheiro inmutable que contén o software e dependencias necesarias. Un contedor é unha instancia en execución desa imaxe, con configuración e estado propios.

Definir que é un repositorio en Docker e cal é a súa función. Expoñer as diferencias entre Docker Hub e un repositorio privado.

Un repositorio en Docker almacena imaxes de contedores para a súa distribución. Docker Hub é un servizo público para compartir imaxes, mentres que un repositorio privado permite un control exclusivo e maior seguridade para as imaxes.

3.2. Instalación e configuración de Docker

Completa os seguintes enunciados empregando os comandos correspondentes e pegando as **capturas** de pantalla que demostren o correcto funcionamento.

1. Instalar **Docker Engine** nunha máquina virtual con GNU/Linux, seguindo as instrucións específicas para a distribución que elixa. Verificar que Docker está correctamente instalado co comando `docker --version`.

```
root@debian12:~# apt-get install ca-certificates curl
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
ca-certificates ya está en su versión más reciente (20230311).
curl ya está en su versión más reciente (7.88.1-10+deb12u8).
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 47 no actualizados.
root@debian12:~# install -m 0755 -d /etc/apt/keyrings
```

```
root@debian12:~# curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
```

```
root@debian12:~# echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://downloa
d.docker.com/linux/debian $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | tee /etc/apt/sources.list.d/dock
er.list > /dev/null
root@debian12:~# apt-get update
Obj:1 http://security.debian.org/debian-security bookworm-security InRelease
Obj:2 http://deb.debian.org/debian bookworm InRelease
Obj:3 http://deb.debian.org/debian bookworm-updates InRelease
Des:4 https://download.docker.com/linux/debian bookworm InRelease [43,3 kB]
Des:5 https://download.docker.com/linux/debian bookworm/stable amd64 Packages [33,9 kB]
Descargados 77,2 kB en 1s (65,6 kB/s)
Leyendo lista de paquetes... Hecho
root@debian12:~#
```

```
root@debian12:~# apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
  docker-ce-rootless-extras git git-man iptables liberror-perl libglib2.0-0 libglib2.0-data libip6tc2 libltdl7
  libnetfilter-conntrack3 libnfnetlink0 libslirp0 patch pigz shared-mime-info slirp4netns xdg-user-dirs
Paquetes sugeridos:
  aufs-tools cgroupfs-mount | cgroup-lite git-daemon-run | git-daemon-sysvinit git-doc git-email git-gui gitk
  gitweb git-cvs git-mediawiki git-svn firewalld low-memory-monitor ed diffutils-doc
Se instalarán los siguientes paquetes NUEVOS:
  containerd.io docker-buildx-plugin docker-ce docker-ce-cli docker-ce-rootless-extras docker-compose-plugin git
  git-man iptables liberror-perl libglib2.0-0 libglib2.0-data libip6tc2 libltdl7 libnetfilter-conntrack3
  libnfnetlink0 libslirp0 patch pigz shared-mime-info slirp4netns xdg-user-dirs
0 actualizados, 22 nuevos se instalarán, 0 para eliminar y 47 no actualizados.
Se necesita descargar 138 MB de archivos.
Se utilizarán 518 MB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n] s
Des:1 http://deb.debian.org/debian bookworm/main amd64 pigz amd64 2.6-1 [64,0 kB]
Des:2 https://download.docker.com/linux/debian bookworm/stable amd64 containerd.io amd64 1.7.25-1 [29,6 MB]
Des:3 http://deb.debian.org/debian bookworm/main amd64 libip6tc2 amd64 1.8.9-2 [19,4 kB]
Des:4 http://deb.debian.org/debian bookworm/main amd64 libnfnetlink0 amd64 1.0.2-2 [15,1 kB]
Des:5 http://deb.debian.org/debian bookworm/main amd64 libnetfilter-conntrack3 amd64 1.0.9-3 [40,7 kB]
Des:6 http://deb.debian.org/debian bookworm/main amd64 iptables amd64 1.8.9-2 [360 kB]
Des:7 http://deb.debian.org/debian bookworm/main amd64 liberror-perl all 0.17029-2 [29,0 kB]
Des:8 http://deb.debian.org/debian bookworm/main amd64 git-man all 1:2.39.5-0+deb12u1 [2.054 kB]
Des:9 http://deb.debian.org/debian bookworm/main amd64 git amd64 1:2.39.5-0+deb12u1 [7.256 kB]
Des:10 https://download.docker.com/linux/debian bookworm/stable amd64 docker-buildx-plugin amd64 0.19.3-1~debian.12~b
ookworm [30,7 MB]
Des:11 http://deb.debian.org/debian bookworm/main amd64 libglib2.0-0 amd64 2.74.6-2+deb12u5 [1.403 kB]
Des:12 http://deb.debian.org/debian bookworm/main amd64 libglib2.0-data all 2.74.6-2+deb12u5 [1.209 kB]
Des:13 http://deb.debian.org/debian bookworm/main amd64 libltdl7 amd64 2.4.7-7~deb12u1 [393 kB]
Des:14 http://deb.debian.org/debian bookworm/main amd64 libslirp0 amd64 4.7.0-1 [63,0 kB]
Des:15 http://deb.debian.org/debian bookworm/main amd64 patch amd64 2.7.6-7 [128 kB]
Des:16 http://deb.debian.org/debian bookworm/main amd64 shared-mime-info amd64 2.2-1 [729 kB]
Des:17 http://deb.debian.org/debian bookworm/main amd64 slirp4netns amd64 1.2.0-1 [37,5 kB]
Des:18 http://deb.debian.org/debian bookworm/main amd64 xdg-user-dirs amd64 0.18-1 [54,4 kB]
Des:19 https://download.docker.com/linux/debian bookworm/stable amd64 docker-ce-cli amd64 5:27.5.0-1~debian.12~bookwo
rm [15,2 MB]
Des:20 https://download.docker.com/linux/debian bookworm/stable amd64 docker-ce amd64 5:27.5.0-1~debian.12~bookworm [
26,0 MB]
Des:21 https://download.docker.com/linux/debian bookworm/stable amd64 docker-ce-rootless-extras amd64 5:27.5.0-1~debi
an.12~bookworm [9.588 kB]
Des:22 https://download.docker.com/linux/debian bookworm/stable amd64 docker-compose-plugin amd64 2.32.3-1~debian.12~
bookworm [12,8 MB]
Descargados 138 MB en 3min 59s (577 kB/s)
Seleccionando el paquete pigz previamente no seleccionado.
```

2. Nunha máquina virtual con GNU/Linux e con *Docker CE* instalado, configurar Docker para que poida ser executado sen requerir privilexios de *root*. (Nota: engadir o teu usuario ao grupo `docker` do sistema).

```
docker:x:996:
root@debian12:~# cat /etc/group
```

```
root@debian12:~# usermod -aG docker rubenrf
```

```
docker:x:996:rubenrf
root@debian12:~# cat /etc/group
```

3. Probar a executar un contedor simple co comando `docker run hello-world` e observar o resultado.

```
root@debian12:~# docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:1b7a37f2a0e26e55ba2916e0c53bfbe60d9bd43e390e31aacd25cb3581ed74e6
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/

root@debian12:~#
```

3.3. Manipulación de imaxes e contedores

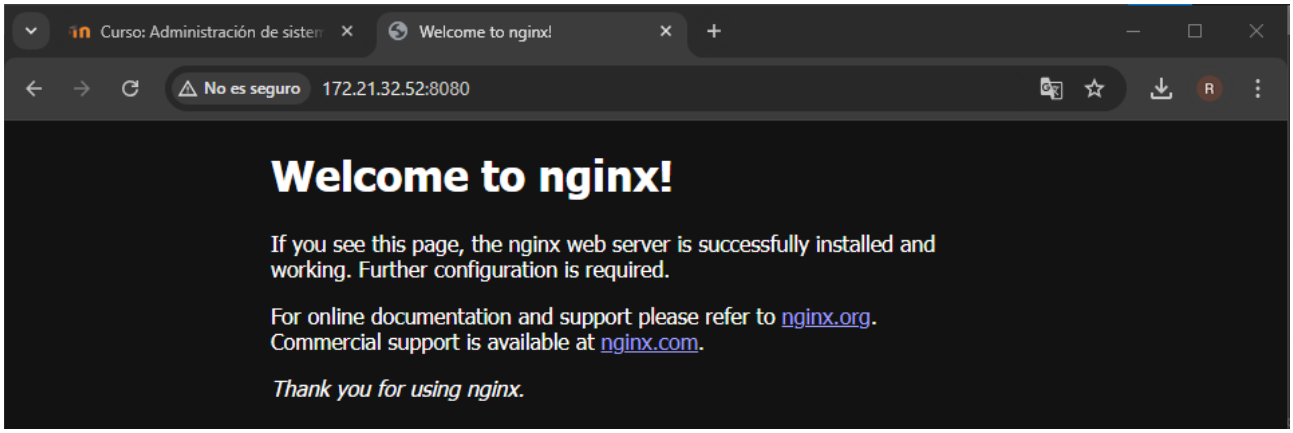
Completa os seguintes enunciados empregando os comandos correspondentes e pegando as **capturas** de pantalla que demostren o correcto funcionamento. Pódese usar unha mesma captura para varios apartados (indicalo en caso de facelo).

1. Descargar a imaxe de `nginx` desde *Docker Hub* e executa un contedor chamado `web_server` en modo «*detached*».

```
root@debian12:~# docker run nginx:1.2.6.2
Unable to find image 'nginx:1.2.6.2' locally
docker: Error response from daemon: manifest for nginx:1.2.6.2 not found: manifest unknown: manifest unknown.
See 'docker run --help'.
root@debian12:~# docker run nginx:1.26.2
Unable to find image 'nginx:1.26.2' locally
1.26.2: Pulling from library/nginx
af302e5c37e9: Downloading [=====================================>             ]  21.46MB/28.21MB
fd5e42408d9a: Downloading [=========================>                         ]  22.17MB/41.88MB
f45aaf9ca822: Download complete
ae00e6f53029: Download complete
b8229349b16f: Download complete
092d2c15cc87: Download complete
4d9b8625ec33: Download complete
```

2. Executar un contedor nginx mapeando o porto 8080 do host ao porto 80 do contedor.

```
rubenrf@debian12:~$ docker run -d --name web_server -p 8080:80 nginx
a035048ed07a06123f12140d4dc7f1e7b4afe93308c6bd226730a770cd4b2c6d
rubenrf@debian12:~$ docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS
a035048ed07a   nginx     "/docker-entrypoint.…"   5 seconds ago   Up 4 seconds   0.0.0.0:8080->80/tcp, [::]:8080->80/tcp
```



3. Crear un volume chamado `nginx_data` e utiliza ese volume para montar o directorio /usr/share/nginx/html do contedor `nginx`.

```
rubenrf@debian12:~$ docker run -d -v nginx_data:/usr/share/nginx/html -p 8080:80 --name web_server1 nginx
c8a7e17a708f5f7970f517a7d9ec1baba2976f79c83be82d7bb0409e3193bbff
rubenrf@debian12:~$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS
c8a7e17a708f   nginx     "/docker-entrypoint.…"   4 seconds ago   Up 3 seconds   0.0.0.0:8080->80/tcp, [::]:8080->80/tcp
rubenrf@debian12:~$ docker volume ls
DRIVER    VOLUME NAME
local     nginx_data
rubenrf@debian12:~$ _
```

4. Executar un contedor nginx con variables de entorno `NGINX_HOST=localhost` e `NGINX_PORT=80` e verifica (inspeccionando o contedor) que as variables se estableceron correctamente.

```
root@debian12:/# docker run -d --name web_server2 -e NGINX_HOST=localhost -e NGINX_PORT=8000 -p 8081:8000 nginx:latest
643eefe6744941d784124dc39bd49a7bd8025dd6835347b74ba9a7bab7ab0630
root@debian12:/# docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS          PORTS
      NAMES
643eefe67449   nginx:latest   "/docker-entrypoint.…"   6 seconds ago   Up 5 seconds    80/tcp, 0.0.0.0:8081->8000/tcp, [::]:8081->8000
/tcp   web_server2
511d488e9940   nginx:prueba   "/docker-entrypoint.…"   8 minutes ago   Up 8 minutes    0.0.0.0:8090->80/tcp, [::]:8090->80/tcp
      wprueba
b25992924ee7   nginx          "/docker-entrypoint.…"   28 hours ago    Up 49 minutes   0.0.0.0:8080->80/tcp, [::]:8080->80/tcp
      web_server
root@debian12:/#
```

5. Listar todos os contedores en execución, pausar o contedor chamado `web_server` e logo reinicialo.

```
root@debian12:/var/lib/docker/volumes/nginx_data/_data# docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS
b25992924ee7   nginx     "/docker-entrypoint.…"   7 minutes ago   Up 7 minutes   0.0.0.0:8080->80/tcp, [::]:8080->80/tcp
root@debian12:/var/lib/docker/volumes/nginx_data/_data#
```

```
root@debian12:/var/lib/docker/volumes/nginx_data/_data# docker stop web_server
web_server
root@debian12:/var/lib/docker/volumes/nginx_data/_data# docker restart web_server
web_server
root@debian12:/var/lib/docker/volumes/nginx_data/_data# docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS
b25992924ee7   nginx     "/docker-entrypoint.…"   8 minutes ago   Up 2 seconds   0.0.0.0:8080->80/tcp, [::]:8080->80/tcp
root@debian12:/var/lib/docker/volumes/nginx_data/_data# _
```

6. Deter o contedor `web_server` e logo eliminalo completamente. Verificar que o contedor foi correctamente eliminado.

```
root@debian12:/# docker pause web_server2
web_server2
root@debian12:/# docker rm web_server2
Error response from daemon: cannot remove container "/web_server2": container is paused and must be unpaused first
root@debian12:/# docker unpause web_server2
web_server2
root@debian12:/# docker stop web_server2
web_server2
root@debian12:/# docker rm web_server2
web_server2
root@debian12:/# docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS                                   NAME
S
511d488e9940   nginx:prueba   "/docker-entrypoint.…"   13 minutes ago   Up 13 minutes   0.0.0.0:8090->80/tcp, [::]:8090->80/tcp   wpru
eba
b25992924ee7   nginx          "/docker-entrypoint.…"   28 hours ago     Up 55 minutes   0.0.0.0:8080->80/tcp, [::]:8080->80/tcp   web_
server
root@debian12:/#
```

7. Iniciar en segundo plano un contedor `nginx` chamado `inspeccion_nginx` e obtén información detallada do contedor utilizando `docker inspect`. Fixarse nos datos relativos á data de creación do contedor, o estado de execución, ó enderezo de rede IP e os portos de servizo abertos e as variables de entorno (marcar eses datos na captura de pantalla).

```
root@debian12:/# docker run -d --name inspeccion_nginx nginx:latest
214e327ef1018a6211ba00b756d3c1dc7946c11edff76c6fe6820469cec24da8
root@debian12:/# docker inspect
```

```
root@debian12:/# docker inspect inspeccion_nginx
[
    {
        "Id": "214e327ef1018a6211ba00b756d3c1dc7946c11edff76c6fe6820469cec24da8",
        "Created": "2025-01-22T13:09:16.108352756Z",
        "Path": "/docker-entrypoint.sh",
```

```
"State": {
    "Status": "running",
    "Running": true,
    "Paused": false,
    "Restarting": false,
    "OOMKilled": false,
    "Dead": false,
    "Pid": 6474,
    "ExitCode": 0,
    "Error": "",
    "StartedAt": "2025-01-22T13:09:16.603957767Z",
    "FinishedAt": "0001-01-01T00:00:00Z"
},
```

```
"Networks": {
    "bridge": {
        "IPAMConfig": null,
        "Links": null,
        "Aliases": null,
        "MacAddress": "02:42:ac:11:00:04",
        "DriverOpts": null,
        "NetworkID": "a528bcd0da5d6dad31eab671d7af8f48507c7d40e2342fe9dcc8dbc72f651c38",
        "EndpointID": "2370e4055d2bf8764d35a31b8cd2e943aff0c61c06f375410f7dcb20f87bcf32",
        "Gateway": "172.17.0.1",
        "IPAddress": "172.17.0.4",
        "IPPrefixLen": 16,
        "IPv6Gateway": "",
        "GlobalIPv6Address": "",
        "GlobalIPv6PrefixLen": 0,
        "DNSNames": null
    }
}
```

```
  "Ports": {
      "80/tcp": null
  },
```

```
"Env": [
    "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin",
    "NGINX_VERSION=1.27.3",
    "NJS_VERSION=0.8.7",
    "NJS_RELEASE=1~bookworm",
    "PKG_RELEASE=1~bookworm",
    "DYNPKG_RELEASE=1~bookworm"
],
```

8. Listar todas as imaxes locais descargadas e intenta eliminar a imaxe de `nginx`. Observa o que acontece. Indicar que hai que facer previamente para poder eliminar con éxito a imaxe.

```
root@debian12:/# docker images
REPOSITORY         TAG       IMAGE ID       CREATED          SIZE
nginx              prueba    a0d9e3b3dcc2   24 minutes ago   192MB
web_server-nginx   latest    58d3f6196d31   40 minutes ago   192MB
<none>             <none>    14ec56fbaf93   41 minutes ago   192MB
nginx              latest    9bea9f2796e2   8 weeks ago      192MB
nginx              1.26.2    0dcfd986e814   5 months ago     188MB
hello-world        latest    d2c94e258dcb   21 months ago    13.3kB
root@debian12:/# docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS             PORTS                                       N
AMES
214e327ef101   nginx:latest   "/docker-entrypoint.…"   6 minutes ago    Up 6 minutes       80/tcp                                      i
nspeccion_nginx
511d488e9940   nginx:prueba   "/docker-entrypoint.…"   22 minutes ago   Up 22 minutes      0.0.0.0:8090->80/tcp, [::]:8090->80/tcp     w
prueba
b25992924ee7   nginx          "/docker-entrypoint.…"   28 hours ago     Up About an hour   0.0.0.0:8080->80/tcp, [::]:8080->80/tcp     w
eb_server
root@debian12:/# docker rmi nginx
Error response from daemon: conflict: unable to remove repository reference "nginx" (must force) - container 214e327ef101 is using its
 referenced image 9bea9f2796e2
root@debian12:/#
```

1 - Stop todos los contenedores

2 - Rm todos los contenedores

3 - docker rmi nginx

9. Crear dous contedores (`compartido_nginx` e `Compartido_busybox`) que compartan un volume chamado `compartido_datos` e usar `docker exec` para crear un ficheiro no volume mediante unha shell.

```
root@debian12:/# docker volume create compartido_datos
compartido_datos
root@debian12:/# docker volume ls
DRIVER    VOLUME NAME
local     compartido_datos
local     nginx_data
root@debian12:/# docker -v compartido_datos:/usr/share/nginx/html/
Docker version 27.5.0, build a187fa5
root@debian12:/# docker run -d -v compartido_datos:/usr/share/nginx/html --name compartido_nginx nginx
535f2fbf25b0e86b2bc88a51e1e5b77a44777a30023ee2a7a2ae60ea71f284bd
root@debian12:/# docker run -d -v compartido_datos:/usr/share/nginx/html --name compartido_busybox nginx
cbabc7ed87b18475fb9868587b1fcf838cdae91bdcf19f49dbec56fdf6e6f725
root@debian12:/#
```

```
root@debian12:/# docker exec -it compartido_busybox bash -c "echo 'Hola desde Busybox' > /usr/share/nginx/html/hola.txt"
root@debian12:/#
```

```
root@debian12:/# docker exec -it compartido_nginx bash
root@535f2fbf25b0:/# ls /usr/share/nginx/html/
50x.html  hola.txt  index.html
root@535f2fbf25b0:/# cat /usr/share/nginx/html/hola.txt
Hola desde Busybox
root@535f2fbf25b0:/#
```

10. Construír unha imaxe personalizada chamada `personalizado_nginx` editando un ficheiro Dockerfile que copia un ficheiro `index.html` ao directorio `/usr/share/nginx/html`. Despois, executar un contedor usando a imaxe recen creada e verificar que o ficheiro `index.html` personalizado se serve correctamente.

Crear ficheiro `index.html` co seguinte contido:

```
<!DOCTYPE html>
<html>
<head>
    <title>Página Personalizada</title>
</head>
<body>
    <h1>Hola desde personalizado_nginx</h1>
</body>
</html>
```

```
rubenrf@debian12:~/pto10$ nano index.html
rubenrf@debian12:~/pto10$
```

rubenrf@debian12: ~/pto10

```
 GNU nano 7.2                                  index.html
<!DOCTYPE html>
<html>
<head>
    <title>Página Personalizada</title>
</head>
<body>
    <h1>Hola desde personalizado_nginx</h1>
</body>
</html>
```

```
rubenrf@debian12:~/pto10$ nano DockerFile
```

```
rubenrf@debian12:~/pto10$ docker build -t personalizado_nginx .
[+] Building 1.1s (7/7) FINISHED                                  docker:default
 => [internal] load build definition from Dockerfile                       0.0s
 => => transferring dockerfile: 87B                                        0.0s
 => [internal] load metadata for docker.io/library/nginx:latest            0.0s
 => [internal] load .dockerignore                                          0.0s
 => => transferring context: 2B                                            0.0s
 => [internal] load build context                                          0.1s
 => => transferring context: 185B                                          0.0s
 => [1/2] FROM docker.io/library/nginx:latest                              0.1s
 => [2/2] COPY index.html /usr/share/nginx/html/                           0.5s
 => exporting to image                                                     0.2s
 => => exporting layers                                                    0.1s
 => => writing image sha256:168d5c990769997f9023affe04a2e3c74429adaaa13fa7916c4b57dadbccbf57  0.0s
 => => naming to docker.io/library/personalizado_nginx                     0.0s
rubenrf@debian12:~/pto10$
```

```
rubenrf@debian12:~/pto10$ docker images
REPOSITORY            TAG       IMAGE ID       CREATED          SIZE
personalizado_nginx   latest    168d5c990769   35 seconds ago   192MB
nginx                 prueba    a0d9e3b3dcc2   7 days ago       192MB
web_server-nginx      latest    58d3f6196d31   7 days ago       192MB
nginx                 latest    9bea9f2796e2   2 months ago     192MB
nginx                 1.26.2    0dcfd986e814   5 months ago     188MB
hello-world           latest    d2c94e258dcb   21 months ago    13.3kB
rubenrf@debian12:~/pto10$
```

```
rubenrf@debian12:~/pto10$ docker run -d -p 8083:80 --name personalizado_nginx personalizado_nginx
6794b4873d26b9eeaf8b435b06a4859adc59c88a2153cec898286aa01af32c5d
rubenrf@debian12:~/pto10$ docker ps
CONTAINER ID   IMAGE               COMMAND                  CREATED         STATUS         PORTS
                NAMES
6794b4873d26   personalizado_nginx   "/docker-entrypoint.…"   5 seconds ago   Up 5 seconds   0.0.0.0:8083->80/tcp, [:
:]:8083->80/tcp   personalizado_nginx
rubenrf@debian12:~/pto10$
```

11. Descargar a imaxe de mysql desde Docker Hub e executa un contedor chamado `base_de_datos` en segundo plano con un volume para almacenar os datos da base de datos. Para a execución, establecer as variables de entorno `MYSQL_ROOT_PASSWORD=root` e `MYSQL_DATABASE=db1`. Verificar que o volume se creou correctamente utilizando a orde `docker volume inspect`. Comprobar tamén que o contedor está funcionando coas variable de entorno especificadas.

```
rubenrf@debian12:~/pto10$ docker run -d --name base_datos -e MYSQL_ROOT_PASSWORD=root -e MYSQL_DATABASE=db1 -v mysql_
datos:/var/lib/mysql mysql:latest
85f21c82c85a92ef7629cf659dead14ae0d7c8fc0e9cf6f00643ad7d8bcec3fe
rubenrf@debian12:~/pto10$
```

```
rubenrf@debian12:~/pto10$ docker inspect base_datos | grep volume
                "Type": "volume",
                "Source": "/var/lib/docker/volumes/mysql_datos/_data",
rubenrf@debian12:~/pto10$
```

```
rubenrf@debian12:~/pto10$ docker exec -it base_datos env
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=85f21c82c85a
TERM=xterm
MYSQL_ROOT_PASSWORD=root
MYSQL_DATABASE=db1
GOSU_VERSION=1.17
MYSQL_MAJOR=innovation
MYSQL_VERSION=9.2.0-1.el9
MYSQL_SHELL_VERSION=9.2.0-1.el9
HOME=/root
rubenrf@debian12:~/pto10$
```

12. Conectarse ó contedor *mysql* en execución creado no punto anterior e levar a cabo algunha operación na base de datos db1 utilizando `docker exec`. Por exemplo, probar a crear unha nova táboa na base de datos, ou calquera outra orde SQL.

```
rubenrf@debian12:~/pto10$ docker exec -it base_datos /bin/bash
bash-5.1# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 9.2.0 MySQL Community Server - GPL

Copyright (c) 2000, 2025, 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           |
+--------------------+
| db1                |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.05 sec)

mysql> use db1;
Database changed
mysql>
```

13. Executar un contedor *postgres*, creando un volume chamado `pg_data` para a persistencia de datos. Mapear o porto 5432 do contedor ao porto 5432 do host. O contedor deberá executarse en modo *detached*, configurando as variables de entorno `POSTGRES_USER=usuario` e `POSTGRES_PASSWORD=contrasinal`. Comproba que o contedor está en execución e que o porto 5432 está dispoñible no host.

```
rubenrf@debian12:~/pto10$ docker pull postgres
Using default tag: latest
latest: Pulling from library/postgres
af302e5c37e9: Already exists
23db180a1f67: Pull complete
dc59dd9c8eb3: Pull complete
aec09e638045: Pull complete
4dd47a683737: Pull complete
7cebbe7849b3: Pull complete
dc4330b02129: Pull complete
498cc40b9fe9: Pull complete
6d3411bb4696: Pull complete
8f14f34d54d3: Pull complete
88d4f7416643: Pull complete
e91ad5cfb8d0: Pull complete
e0c4d5055fb9: Pull complete
254ee626d709: Pull complete
Digest: sha256:87ec5e0a167dc7d4831729f9e1d2ee7b8597dcc49ccd9e43cc5f89e808d2adae
Status: Downloaded newer image for postgres:latest
docker.io/library/postgres:latest
rubenrf@debian12:~/pto10$
```

```
rubenrf@debian12:~$ docker volume create pg_data
pg_data
rubenrf@debian12:~$ docker volume inspect pg_data
[
    {
        "CreatedAt": "2025-01-29T14:21:47+01:00",
        "Driver": "local",
        "Labels": null,
        "Mountpoint": "/var/lib/docker/volumes/pg_data/_data",
        "Name": "pg_data",
        "Options": null,
        "Scope": "local"
    }
]
rubenrf@debian12:~$
```

```
rubenrf@debian12:~$ docker run -d -p 5432:5432 -e POSTGRES_USER=usuario -e POSTGRES_PASSWORD=contrasinal -v pg_data:/
var/lib/postgresql/data postgres
dfc05e72df03de50dc8d848124095418dc362c9e4fb175448c4bd7906af605aa
rubenrf@debian12:~$
```

```
rubenrf@debian12:~$ docker ps
CONTAINER ID   IMAGE                COMMAND                  CREATED          STATUS          PORTS
                  NAMES
dfc05e72df03   postgres             "docker-entrypoint.s…"   22 seconds ago   Up 21 seconds   0.0.0.0:5432->5432/tcp
, :::5432->5432/tcp   jovial_hamilton
85f21c82c85a   mysql:latest         "docker-entrypoint.s…"   18 minutes ago   Up 18 minutes   3306/tcp, 33060/tcp
                  base_datos
6794b4873d26   personalizado_nginx   "/docker-entrypoint.…"   36 minutes ago   Up 36 minutes   0.0.0.0:8083->80/tcp,
[::]:8083->80/tcp     personalizado_nginx
rubenrf@debian12:~$
```

```
rubenrf@debian12:~$ netstat -tln
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:8083            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:5432            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp6       0      0 :::8083                 :::*                    LISTEN
tcp6       0      0 :::5432                 :::*                    LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
rubenrf@debian12:~$
```

14. Crear unha rede interna de Docker chamada `rede_privada`. Executa e conecta dous contedores a esta rede: un contedor coa imaxe *mysql* e outro coa imaxe *busybox*. Emprega a orde `docker exec` con este último contedor para comprobar a conectividade co primeiro.

```
rubenrf@debian12:~$ docker network create rede_privada
c9ab50960911f41be9e6dae665c983449bc07ca7b9aa6e2ab048238a0c142dd6
rubenrf@debian12:~$ docker network ls
NETWORK ID     NAME           DRIVER    SCOPE
cf2c5d3134ee   bridge         bridge    local
9f23a7a61cef   host           host      local
bc6d6fbda4a9   none           null      local
c9ab50960911   rede_privada   bridge    local
rubenrf@debian12:~$ docker network inspect rede_privada
[
    {
        "Name": "rede_privada",
        "Id": "c9ab50960911f41be9e6dae665c983449bc07ca7b9aa6e2ab048238a0c142dd6",
        "Created": "2025-01-29T14:36:43.474195383+01:00",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "172.18.0.0/16",
                    "Gateway": "172.18.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {},
        "Options": {},
        "Labels": {}
    }
]
rubenrf@debian12:~$
```

```
rubenrf@debian12:~$ docker run -d --name mysql_1 --net rede_privada -e MYSQL_ROOT_PASSWORD=root mysql
b0544cbf34d2c0c4529561d48d539fa99990a80a3b4aef72dafbe997f62f7d68
```

```
rubenrf@debian12:~$ docker run -d --name busybox_1 --net rede_privada busybox sleep 3600
530d5d0c4b415d101fa842e74cf15fc4f59bb36a1c74923f52f291c7983dc943
rubenrf@debian12:~$
```

```
rubenrf@debian12:~$ docker ps -a
CONTAINER ID   IMAGE                COMMAND                  CREATED          STATUS                      PORTS
                     NAMES
530d5d0c4b41   busybox              "sleep 3600"             16 seconds ago   Up 15 seconds
                     busybox_1
b0544cbf34d2   mysql                "docker-entrypoint.s…"   2 minutes ago    Up 2 minutes                3306/tcp, 33
060/tcp              mysql_1
```

```
                                    compartido_nginx
rubenrf@debian12:~$ docker exec -it busybox_1 sh -c "ping -c 4 mysql_1"
PING mysql_1 (172.18.0.2): 56 data bytes
64 bytes from 172.18.0.2: seq=0 ttl=64 time=0.276 ms
64 bytes from 172.18.0.2: seq=1 ttl=64 time=0.123 ms
64 bytes from 172.18.0.2: seq=2 ttl=64 time=0.157 ms
64 bytes from 172.18.0.2: seq=3 ttl=64 time=0.168 ms

--- mysql_1 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.123/0.181/0.276 ms
rubenrf@debian12:~$
```

15. Descargar e executar un contedor *redis* en modo *detached*, mapeando o porto `6379` ao host. Configurar a persistencia de datos creando un volume co nome `redis_data` e vinculándoo ó contedor creado. Verificar a persistencia de datos utilizando docker exec para engadir e recuperar datos en Redis.

```
rubenrf@debian12:~$ docker pull redis
Using default tag: latest
latest: Pulling from library/redis
af302e5c37e9: Already exists
01b95e092fd0: Pull complete
c111ca53a743: Pull complete
f7d6cf14046e: Pull complete
589f36d317d9: Pull complete
94041d0cae8f: Pull complete
4f4fb700ef54: Pull complete
4f5f785c9703: Pull complete
Digest: sha256:ca65ea36ae16e709b0f1c7534bc7e5b5ac2e5bb3c97236e4fec00e3625eb678d
Status: Downloaded newer image for redis:latest
docker.io/library/redis:latest
rubenrf@debian12:~$
```

```
root@debian12:~# docker run -d --name redis_cont -p 6379:6379 -v redis_data:/data redis
ac967556950769d7d9e390c2e847224825d11b236e9908fe206a9b96ed9d7cf0
root@debian12:~# docker exec -it redis_cont redis_cli set clave "Ola, Redis"
OCI runtime exec failed: exec failed: unable to start container process: exec: "redis_cli": executable file not found ir
root@debian12:~# docker exec -it redis_cont redis-cli set clave "Ola, Redis"
OK
root@debian12:~# docker exec -it redis_cont redis-cli get clave
"Ola, Redis"
root@debian12:~#
```

16. Configurar un contedor *mongo* para iniciar cun ficheiro de configuración personalizado `mongod.conf` e montar este ficheiro como un volume. É necesario crear primeiro o ficheiro `mongod.conf` no host coas configuracións personalizadas. Verificar que o contedor inicia correctamente utilizando a configuración personalizada.

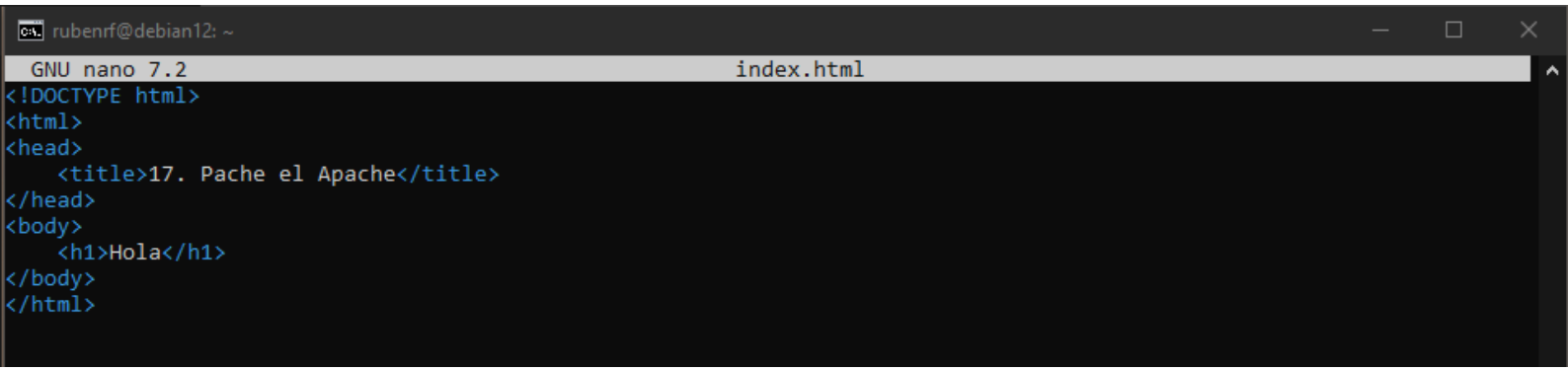
```
rubenrf@debian12:~$ docker pull mongo
Using default tag: latest
latest: Pulling from library/mongo
de44b265507a: Pull complete
add2cfa32b4d: Pull complete
0d3422d31c84: Pull complete
e9869afb5187: Pull complete
9284108c06f8: Pull complete
17351a831ef1: Pull complete
2613644e011d: Pull complete
05cc0f1cded4: Pull complete
Digest: sha256:c7ac28ef4d8137358ed86014a9d10dda2730a64046ce2a49610ad4bd9788d4cb
Status: Downloaded newer image for mongo:latest
docker.io/library/mongo:latest
rubenrf@debian12:~$
```

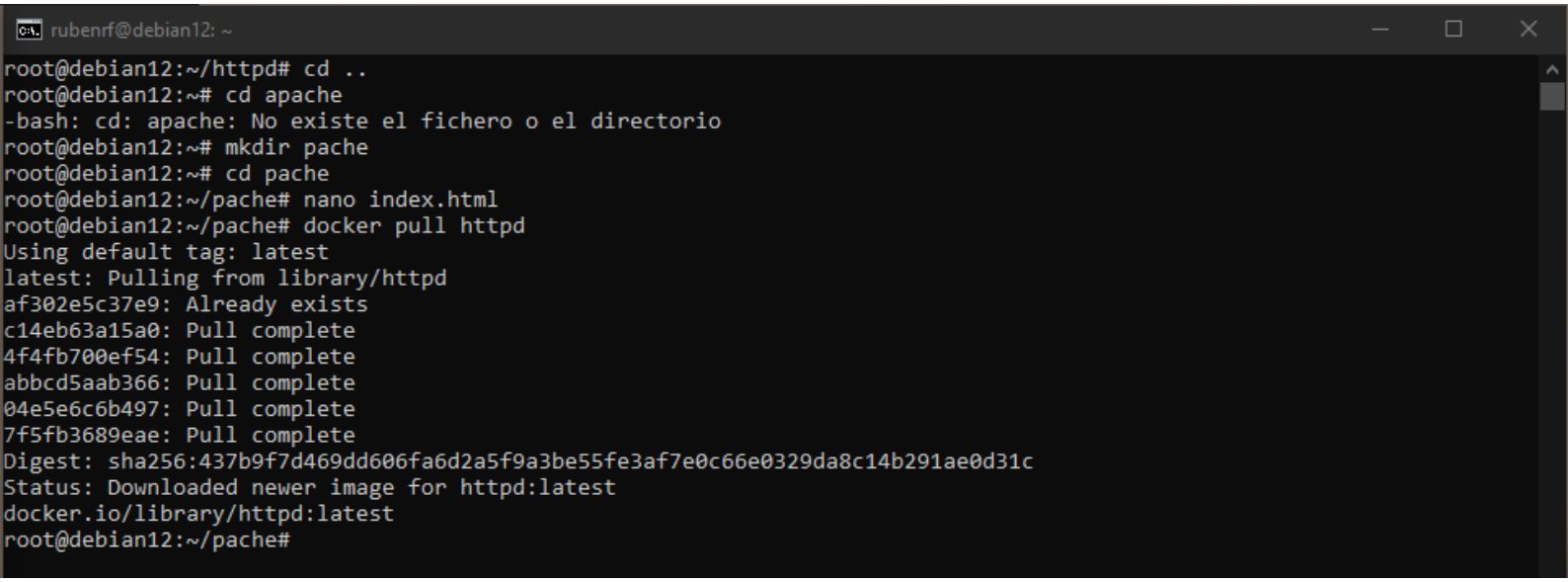
Crear archivo `mongod.conf` co contido:

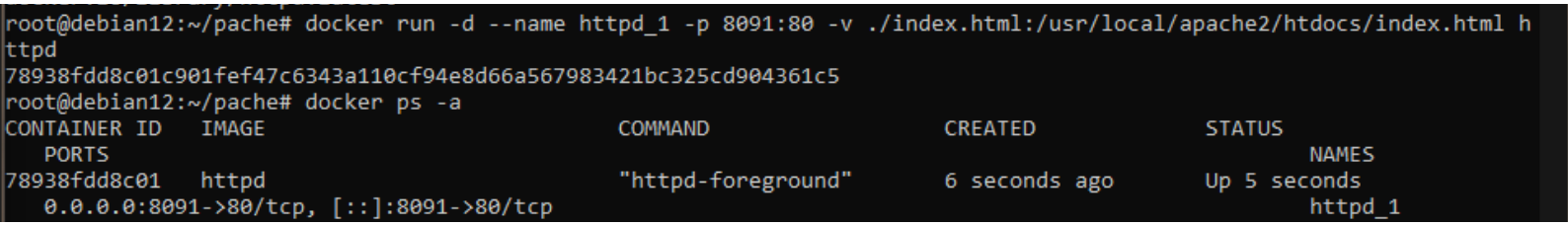
```
storage:
  dbPath: /data/db
net:
  bindIp: 0.0.0.0
  port: 27017
```

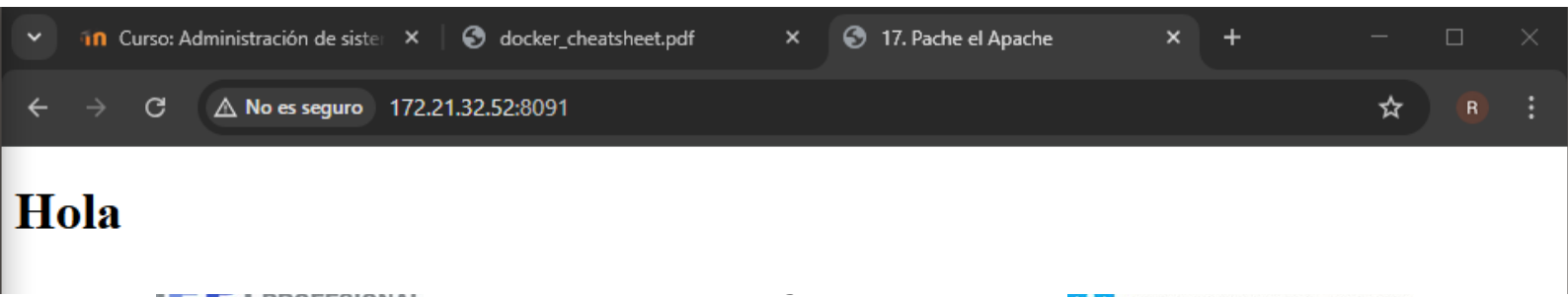
```
root@debian12:~/mongo# docker run -d --name mongo_1 -p 27017:27017 -v ./mongod.conf:/etc/mongo/mongod.conf -v mongo_d
ata:/data/db mongo
1db41b5004037fede7e97908843819122292ed9a2b9164469107387c4610dec5
root@debian12:~/mongo# docker ps
CONTAINER ID   IMAGE                          COMMAND                  CREATED             STATUS             PORTS
NAMES
1db41b500403   mongo                          "docker-entrypoint.s…"   5 seconds ago       Up 4 seconds       0.0.0.
0:27017->27017/tcp, :::27017->27017/tcp
mongo_1
ac9675569507   redis                          "docker-entrypoint.s…"   14 minutes ago      Up 14 minutes      0.0.0.
0:6379->6379/tcp, :::6379->6379/tcp
redis_cont
b8263ca974e1   portainer/portainer-ce:2.21.5   "/portainer"             About an hour ago   Up About an hour   0.0.0.
0:8000->8000/tcp, :::8000->8000/tcp, 0.0.0.0:9443->9443/tcp, :::9443->9443/tcp, 9000/tcp   portainer
root@debian12:~/mongo# docker ps
CONTAINER ID   IMAGE                          COMMAND                  CREATED             STATUS             PORTS
NAMES
1db41b500403   mongo                          "docker-entrypoint.s…"   13 seconds ago      Up 12 seconds      0.0.0.
0:27017->27017/tcp, :::27017->27017/tcp
mongo_1
ac9675569507   redis                          "docker-entrypoint.s…"   14 minutes ago      Up 14 minutes      0.0.0.
0:6379->6379/tcp, :::6379->6379/tcp
redis_cont
b8263ca974e1   portainer/portainer-ce:2.21.5   "/portainer"             About an hour ago   Up About an hour   0.0.0.
0:8000->8000/tcp, :::8000->8000/tcp, 0.0.0.0:9443->9443/tcp, :::9443->9443/tcp, 9000/tcp   portainer
root@debian12:~/mongo# docker logs mongo_1
```

17. Executar un contedor *httpd* (Apache HTTP Server) mapeando o directorio interno do contedor `/usr/local/apache2/htdocs/` ao host e crear un ficheiro `index.html` personalizado. O ficheiro index.html ha de atoparse no directorio mapeado. Mapea tamén o porto 80 do contedor ó porto 8080 do host. Verificar que se serve correctamente a páxina visitando a ligazón http://localhost:8080.









18. Crea un contedor node para executar unha aplicación de Node.js e monta o código da aplicación desde o host. Para facelo, crea primeiro un directorio app no host e engade un ficheiro app.js con unha aplicación simple de Node.js. Mapea o porto que corresponda para que sexa accesible dende o host. Verifica que a aplicación está en execución accedendo ao porto configurado cun navegador.

```
root@debian12:~/pache# docker pull node
Using default tag: latest
latest: Pulling from library/node
fd0410a2d1ae: Pull complete
bf571be90f05: Pull complete
684a51896c82: Pull complete
fbf93b646d6b: Pull complete
6ec6ad1ddcfd: Pull complete
1ef32208f9d7: Pull complete
bd8f51f2401a: Pull complete
0c12196e597a: Pull complete
Digest: sha256:3b73c4b366d490f76908dda253bb4516bbb3398948fd880d8682c5ef16427eca
Status: Downloaded newer image for node:latest
docker.io/library/node:latest
root@debian12:~/pache#
```

Crear `~/app/app.js`:

```
const http = require('http');

const hostname = '0.0.0.0';

const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hola desde Node.js\n');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

Crear `~/app/package.json`:

```
{
  "name": "node-app",
  "version": "1.0.0",
```

```
  "description": "A simple Node.js app",
  "main": "app.js",
  "scripts": {
    "start": "node app.js"
  },
  "dependencies": {}
}
```

```
root@debian12:~/app# ls
app.js  package.json
root@debian12:~/app# cat app.js
const http = require('http');
const hostname = '0.0.0.0';
const port = 3000;
const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hola desde Node.js\n');
});
server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
root@debian12:~/app# cat package.json
{
  "name": "node-app",
  "version": "1.0.0",
  "description": "A simple Node.js app",
  "main": "app.js",
  "scripts": {
    "start": "node app.js"
  },
  "dependencies": {}
}
root@debian12:~/app#
```

```
root@debian12:~/app# docker run -d --name mi-node-app -v ./:/usr/src/app -w /usr/src/app -p 3000:3000 node sh -c "npm
 install && npm start"
6444b04a90faf18589cdccdffdbc8b381204b694eea32f2e738d86b719f349c0
root@debian12:~/app# docker ps
CONTAINER ID   IMAGE                          COMMAND                  CREATED          STATUS          PORTS
                                                                                  NAMES
6444b04a90fa   node                           "docker-entrypoint.s…"   10 seconds ago   Up 8 seconds    0.0.0.0:3000
->3000/tcp, :::3000->3000/tcp                                                     mi-node-app
78938fdd8c01   httpd                          "httpd-foreground"       30 minutes ago   Up 30 minutes   0.0.0.0:8091
->80/tcp, [::]:8091->80/tcp                                                       httpd_1
1db41b500403   mongo                          "docker-entrypoint.s…"   39 minutes ago   Up 38 minutes   0.0.0.0:2701
7->27017/tcp, :::27017->27017/tcp                                                 mongo_1
ac9675569507   redis                          "docker-entrypoint.s…"   52 minutes ago   Up 52 minutes   0.0.0.0:6379
->6379/tcp, :::6379->6379/tcp                                                     redis_cont
b8263ca974e1   portainer/portainer-ce:2.21.5   "/portainer"             2 hours ago      Up 2 hours      0.0.0.0:8000
->8000/tcp, :::8000->8000/tcp, 0.0.0.0:9443->9443/tcp, :::9443->9443/tcp, 9000/tcp   portainer
root@debian12:~/app# curl http://localhost:3000
Hola desde Node.js
root@debian12:~/app#
```

Tarefa 4. Siglas

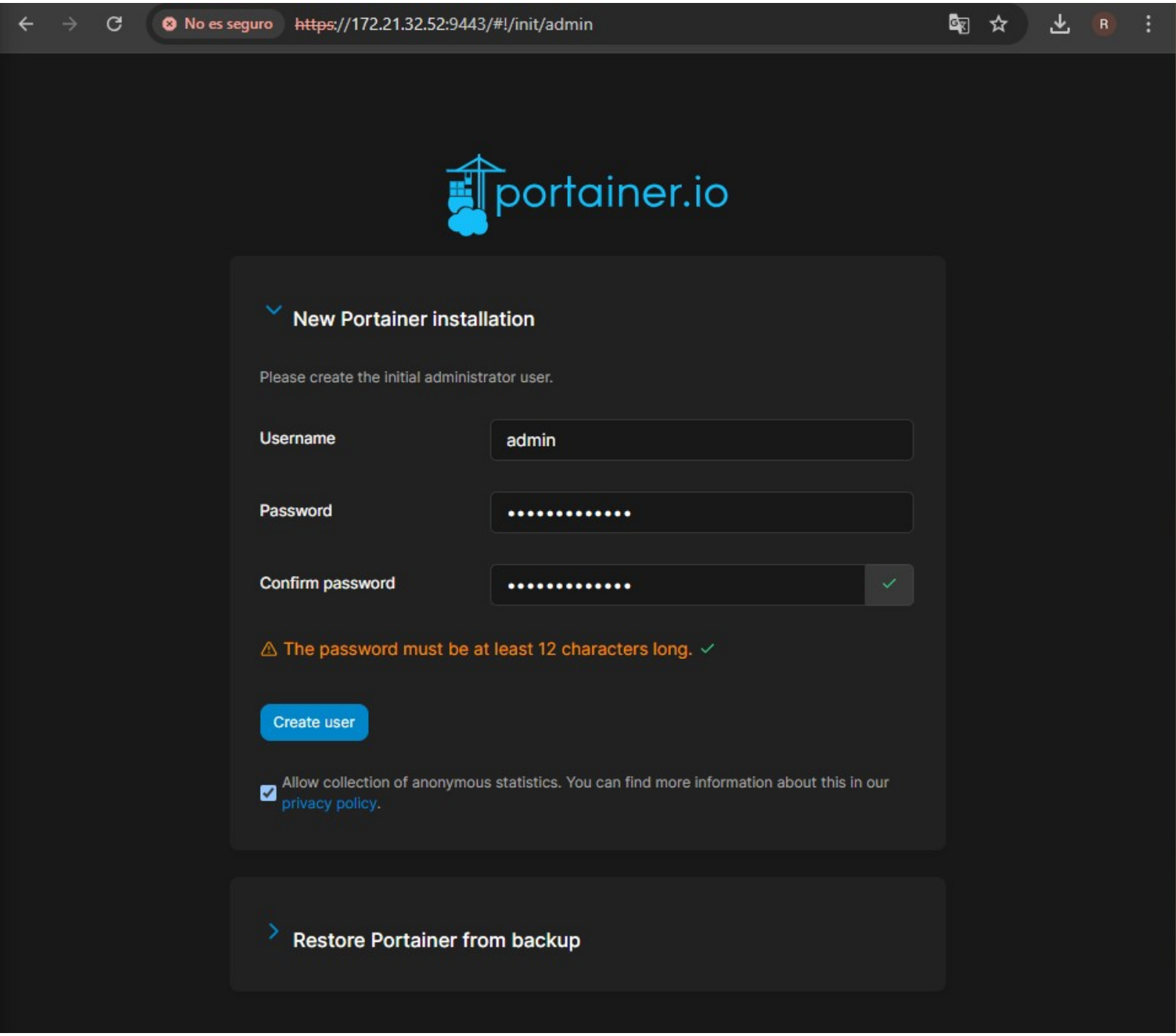
Busca e traduce as seguintes siglas **relacionados coa UD**:

Siglas	Significado	Tradución
SMB	Server Message Block	Bloque de Mensaxe do Servidor
CIFS	Common Internet File System	Sistema Común de Arquivos de Internet
NFS	Network File System	Sistema de Arquivos de Rede
IPX	Internetwork Packet Exchange	Intercambio de Paquetes entre Redes
NetBIOS	Network Basic Input/Output System	Sistema Básico de Entrada/Saída de Rede
NetBEUI	NetBIOS Extended User Interface	Interfaz de Usuario Estendida de NetBIOS

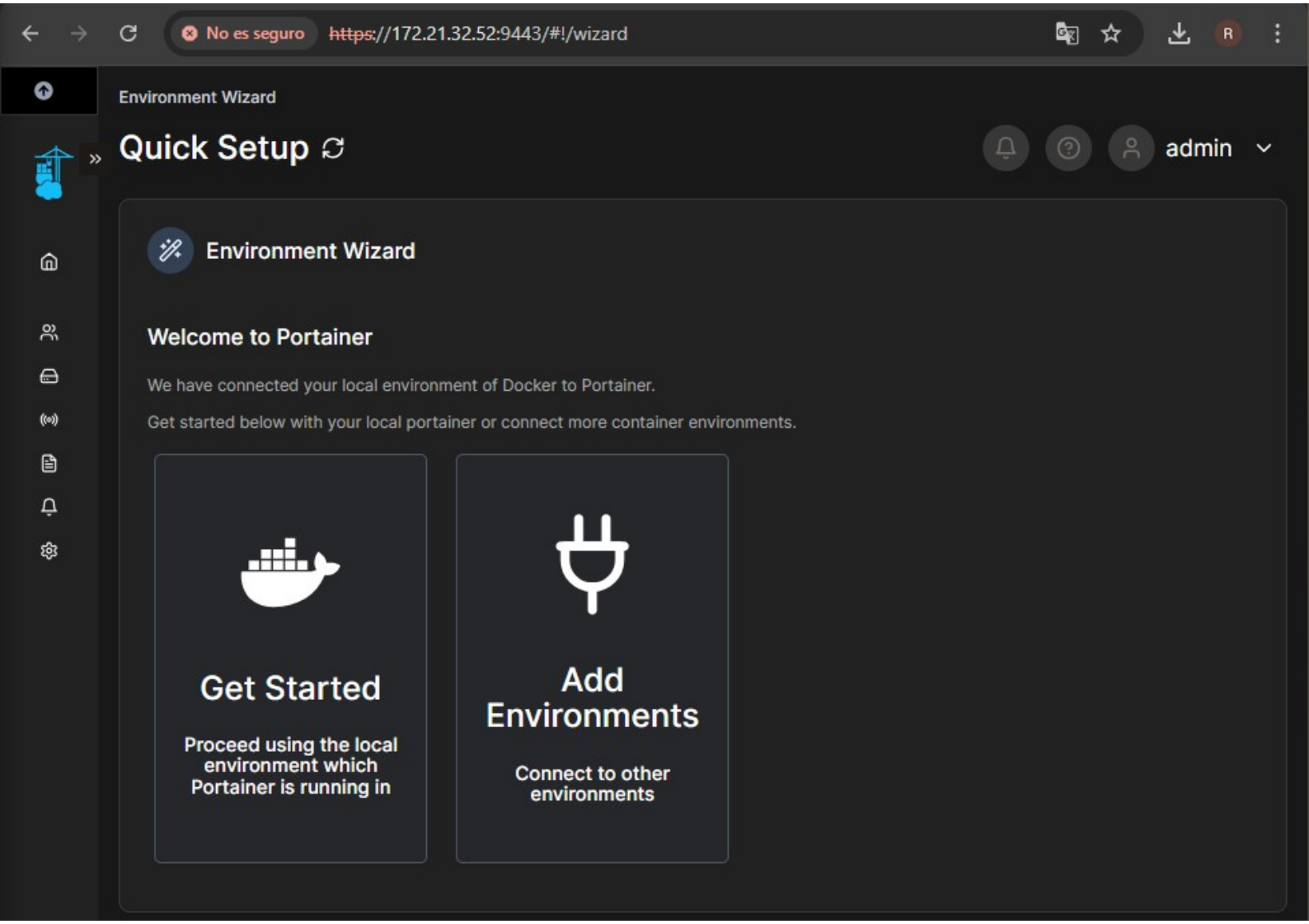
Tarefa 5. Instalar Docker Portainer

```
rubenrf@debian12:~$ docker volume create portainer_data
portainer_data
rubenrf@debian12:~$ docker volume inspect portainer_data
[
    {
        "CreatedAt": "2025-01-30T12:18:34+01:00",
        "Driver": "local",
        "Labels": null,
        "Mountpoint": "/var/lib/docker/volumes/portainer_data/_data",
        "Name": "portainer_data",
        "Options": null,
        "Scope": "local"
    }
]
rubenrf@debian12:~$ docker volume ls
DRIVER    VOLUME NAME
local     b8d1e1ddbcbeb19e5bd63d5088081c2e185800d9c1cc1cb691c43dcd43f63f59
local     c32588cde9d1612b1e087ae03eb03e46e6fc2f15c98dcff5abb090dd7b868a37
local     compartido_datos
local     db4bcc87ade3a793734ac63c6558dd0276292159f54e181db0af52bf55ff443f
local     mysql_datos
local     nginx_data
local     pg_data
local     portainer_data
rubenrf@debian12:~$
```

```
rubenrf@debian12:~$ docker run -d -p 8000:8000 -p 9443:9443 --name portainer --restart=always -v /var/run/docker.sock
:/var/run/docker.sock -v portainer_data:/data portainer/portainer-ce:2.21.5
Unable to find image 'portainer/portainer-ce:2.21.5' locally
2.21.5: Pulling from portainer/portainer-ce
dc8df0f2921e: Pull complete
c82aa9c9fb45: Pull complete
d40df14c1d7a: Pull complete
a3939f2dc487: Pull complete
204b2fbb824e: Pull complete
a53c840f28bf: Pull complete
9e1dadc4be73: Pull complete
6f01ec19fa2b: Pull complete
e2f767fe3885: Pull complete
793e77bf062e: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:bd8f7a6d98e2a512e18272c38914abd1e92d663451f3c925d502a8557a3b92d7
Status: Downloaded newer image for portainer/portainer-ce:2.21.5
b8263ca974e1e4e92c034f2a5da9ac03d6cd04ebb7f9213beb0868d0480d3525
rubenrf@debian12:~$
```



abcdef123456.


No es seguro
https://172.21.32.52:9443/#!/wizard
Environment Wizard
Quick Setup
admin
Environment Wizard
Welcome to Portainer
We have connected your local environment of Docker to Portainer.
Get started below with your local portainer or connect more container environments.
Get Started
Proceed using the local environment which Portainer is running in
Add Environments
Connect to other environments