Seminario de Virtualización con contendores en la nube

Cuaderno de prácticas

Créditos

El contenido es este cuaderno de prácticas ha sigo generado en el transcurso del seminario “Virtualización con contendores en la nube” desarrollador en el IES Clara del Rey entre los meses de enero y mayo de 2024.

Cada practica ha sido desarrollada por alguno de los seis participantes.

Lista de prácticas / autores

PRÁCTICA	AUTOR
Uso de imágenes y contenedores	Guadalupe Bermejo Sánchez
Ejecución de Linux en Windows sin y con contenedores	Concepción Fernández Fernández
Redes y comunicación entre contenedores	Mercedes Rodríguez Villafáfila
Microservicio web creado a partir de DockerFile	Mercedes Rodríguez Villafáfila
Microservicios ACI Azure Cloud	Mercedes Rodríguez Villafáfila
Configuración de Oracle Database XE en un contenedor Docker	Carlos Moreno Martínez
Instalación de WSL	Camino Pardo de Vega
Conectar a una máquina Linux desde el escritorio remoto de Windows	Camino Pardo de Vega
Instalación de Portainer	Guadalupe Bermejo Sánchez
Mini-aplicación en un entorno “contenedorizado” con Docker Compose	José Ramón Rodríguez Sánchez
Comandos de Docker	José Ramón Rodríguez Sánchez

Uso de imágenes y contenedores

Desarrollada por **Guadalupe Bermejo Sánchez**

1.- OBJETIVOS

Partiendo de que docker está instalado, no importa sobre qué sistema operativo, el objetivo de esta práctica es entender los conceptos básicos de imágenes y contenedores y cómo trabajar con ellos desde la línea de comandos.

Para realizar esta práctica se parte de una imagen existente en DockerHub que contiene un SGBD, en este caso MySQL. Se crea un contenedor a partir de esa imagen. Se hacen modificaciones en el contenedor. Se crea una nueva imagen a partir del contenedor modificado y finalmente se sube la nueva imagen a DockerHub.

2.- PASOS A SEGUIR

1. Descarga la imagen de mysql (docker pull <imagen>)

```
# docker pull mysql
```

```
... where gbs-mysql is the name you want to assign to your container, my-secret-pw is
the password to be set for the MySQL root user and tag is the tag specifying the MySQL
version you want. See the list above for relevant tags.

Respuesta:
2c7ee6bdda45c297a4a7ab565583bb1637d29d0376c1eaeb0c9a991b9898907e
```

2. Comprueba que se ha descargado la imagen de sql

```
# docker pull mysql
```

```
REPOSITORY     TAG        IMAGE ID       CREATED          SIZE
mysql          latest     a3a2968869cf   7 days ago       538MB
```

3. Inspecciona la imagen e indica el número de versión descargado (docker [image] inspect <imagen>)

```
# docker inspect mysql | grep VERSION
```

```
"GOSU_VERSION=1.14",
                "MYSQL_VERSION=8.0.31-1.el8",
                "MYSQL_SHELL_VERSION=8.0.31-1.el8"
                "GOSU_VERSION=1.14",
                "MYSQL_VERSION=8.0.31-1.el8",
                "MYSQL_SHELL_VERSION=8.0.31-1.el8"
```

```
Por defecto se descarga la última versión pero si queremos asegurarnos, mejor indicar
“latest” al descargar.

# docker pull mysql:latest
```

4. Comprueba si existe algún contenedor asociado a esa imagen

```
# docker ps --all

CONTAINER ID     IMAGE     COMMAND     CREATED  STATUS     PORTS     NAMES

Todavía no existe ningún contenedor.
```

5. Crea un contenedor a partir de la imagen descargada (docker run). Utiliza tus iniciales para darle nombre al contendor (p.e gbs-mysql); los puertos por defecto de MySQL (3306); tu propia contraseña (p.e: my-secret-pw)

```
# docker run --name gbs-mysql -p 3306:3306 -e MYSQL_ROOT_PASSWORD=my-secret-pw -d mysql
```

```
... where gbs-mysql is the name you want to assign to your container, my-secret-pw is
the password to be set for the MySQL root user and tag is the tag specifying the MySQL
version you want. See the list above for relevant tags.

Respuesta:
2c7ee6bdda45c297a4a7ab565583bb1637d29d0376c1eaeb0c9a991b9898907e
```

6. Inspecciona el contenedor creado (docker [container] inspect <contenedor>)

```
# docker inspect gbs-mysql
[
    {
        "Id": "43a92a99360437e096a6a130809bdae92b3a9ff74c0184b312dc0c25bd8e4523",
        "Created": "2024-01-29T08:19:43.365264524Z",
        "Path": "docker-entrypoint.sh",
        "Args": [
            "mysqld"
        ],
        "State": {
            "Status": "exited",
            "Running": false,
            "Paused": false,
            "Restarting": false,
            "OOMKilled": false,
            "Dead": false,
            "Pid": 0,
            "ExitCode": 0,
            "Error": "",
            "StartedAt": "2024-05-03T21:05:34.366486226Z",
            "FinishedAt": "2024-05-03T22:33:06.753153815Z"
        },
        "Image":
"sha256:8da80fe49fcfad1ac311a2e34c42730c943706c2008083f5e4feeb6d77cdbc1f",
        "ResolvConfPath":
"/var/lib/docker/containers/43a92a99360437e096a6a130809bdae92b3a9ff74c0184b312dc0c25bd8
e4523/resolv.conf",
        "HostnamePath":
"/var/lib/docker/containers/43a92a99360437e096a6a130809bdae92b3a9ff74c0184b312dc0c25bd8
e4523/hostname",
```

```
        "HostsPath":
"/var/lib/docker/containers/43a92a99360437e096a6a130809bdae92b3a9ff74c0184b312dc0c25bd8
e4523/hosts",
        "LogPath":
"/var/lib/docker/containers/43a92a99360437e096a6a130809bdae92b3a9ff74c0184b312dc0c25bd8
e4523/43a92a99360437e096a6a130809bdae92b3a9ff74c0184b312dc0c25bd8e4523-json.log",
        "Name": "/gbs-mysql",
        "RestartCount": 0,
        "Driver": "overlay2",
        "Platform": "linux",
        "MountLabel": "",
        "ProcessLabel": "",
        "AppArmorProfile": "docker-default",
        "ExecIDs": null,
        "HostConfig": {
            "Binds": [
                "/home/gbs/SQL:/SQL:ro"
            ],
            "ContainerIDFile": "",
            "LogConfig": {
                "Type": "json-file",
                "Config": {}
            },
            "NetworkMode": "bridge",
            "PortBindings": {
                "3306/tcp": [
                    {
                        "HostIp": "",
                        "HostPort": "3306"
                    }
                ]
            },...
```


Se obtienen todos los detalles de cómo se ha creado el contenedor.

7. Comprueba que se ha creado un contenedor MySQL y cuál es su estado

```
# docker ps --all
```

```
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS                    NAMES
2c7ee6bdda45   mysql     "docker-entrypoint.s…"   2 minutes ago    Up 2 minutes    3306/tcp, 33060/tcp      gbs-mysql
```

8. Accede al contenedor que tiene instalado MySQL desde la máquina anfitriona, arrancando una shell (bash) interactiva (-it) (ej. docker exec -it <contenedor> bash).

```
# docker exec -it gbs-mysql bash
```

   - **docker exec**: This is the Docker command for executing a command in a running container.
   - **-it**: These are options that allow you to run the command interactively in the container, providing an interactive shell.
   - **gbs-mysql**: This is the name of the running MySQL container.

- **bash**: This is the command to be executed inside the container, which in this case opens an interactive bash shell.

9. Probar a acceder al entorno de MySQL

```
bash-4.4# mysql -u root -p
```

```
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 8.0.31 MySQL Community Server - GPL

Copyright (c) 2000, 2022, 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>
mysql> show databases;
```

10. Salir de MySQL

```
mysql> exit
```

11. Abandonar el contenedor

```
bash-4.4# exit
```

Ahora estamos de vuelta al sistema anfitrión.

```
#
```

12. Comprobar si el contenedor pertenece a alguna red por defecto y si hay otros hosts en esa red (docker network ls, docker network inspect <nombre de la red>).

```
# docker network ls
```

```
NETWORK ID     NAME              DRIVER    SCOPE
7dd1eb67384e   app-jugadores     bridge    local
1358d5ec7544   bridge            bridge    local
ebd38fab1b01   host              host      local
f48c733dd761   none              null      local
deb91658fb7a   prueba1           bridge    local
170f747c4463   prueba1_default   bridge    local
5e3ae4752005   prueba2           bridge    local
83dbeb72b9df   wordpress         bridge    local
```

```
Se verá más detallado el tema de redes en otra práctica.
```

13. Detener el contenedor (docker stop <contenedor>).

```
# docker stop gbs-mysql
```

14. Arrancar de nuevo el contenedor (docker start <contenedor>).

```
# docker start gbs-mysql
```

15. Parar el contenedor y crear una imagen del mismo para subirla más tarde a DockerHub (docker commit <nombre del contenedor> <nombre imagen>)

```
# docker stop gbs-mysql

# docker commit gbs-mysql 051123
sha256:dff9333cebc13efb525fdbac4fc86161f3dca861a27110c12787b1e2d8cf5d26
```

16. Verificar que se ha creado una nueva imagen

```
# docker images
REPOSITORY                    TAG       IMAGE ID       CREATED          SIZE
051123                        latest    dff9333cebc1   8 seconds ago    577MB
mi-jupyter-lab                latest    1bc87e47d0c0   6 months ago     1.59GB
c-sort-app                    latest    5cc25c85d209   6 months ago     1.38GB
app-jugadores                 0.1       584f6d465901   6 months ago     491MB
```

Una vez creada la imagen del contenedor, se puede utilizar esta imagen para crear nuevos contenedores con la configuración y el estado del contenedor original guardados en la imagen.

17. Crear una cuenta en Docker Hub (https://hub.docker.com/) si no se tiene aún para poder subir imágenes allí.

18. Acceder a Docker Hub

```
# docker login
```

19. Etiquetar la imagen que se quiere subir a Docker Hub (docker tag image_name repository_name/image_name:tag)

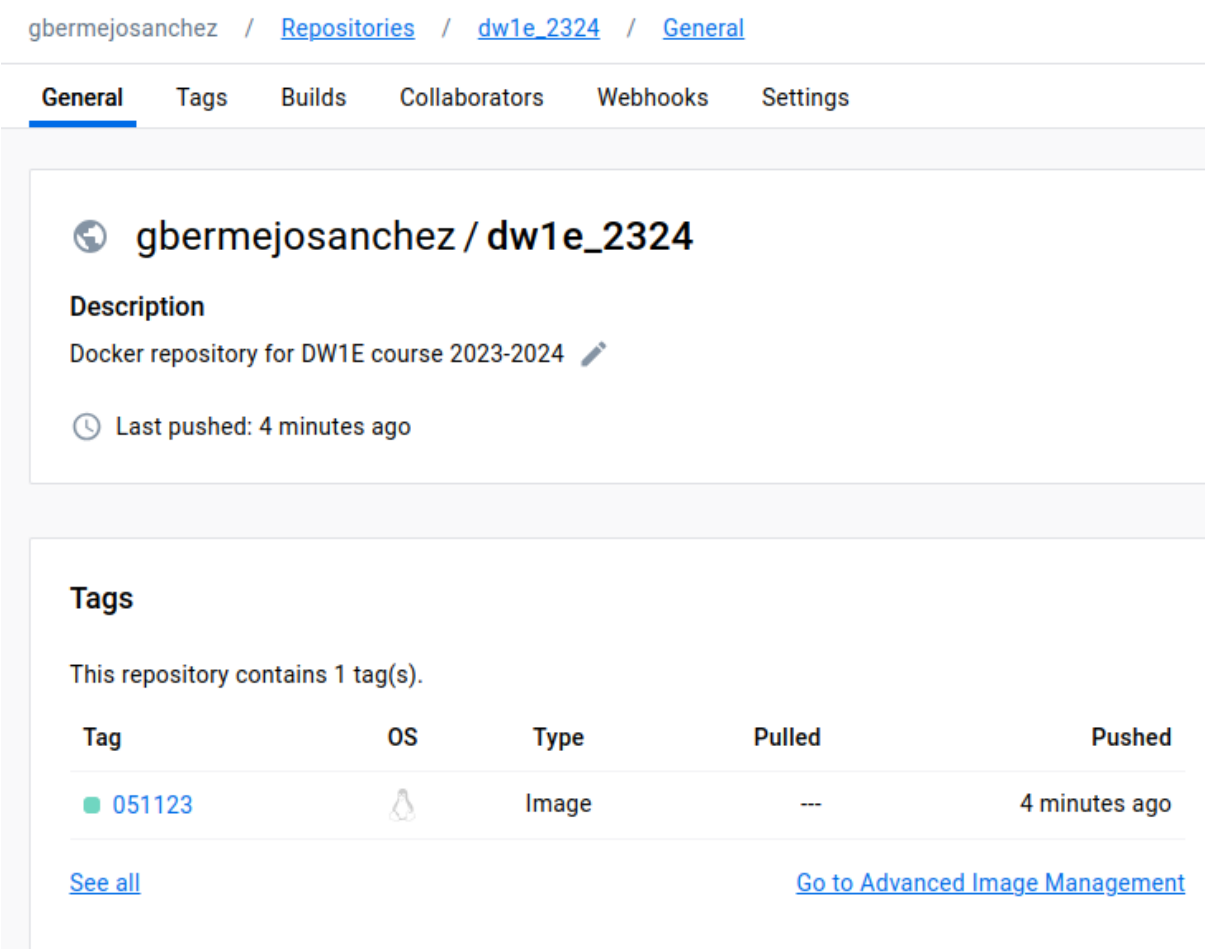
```
Sustituir nombre_imagen por el nombre de la imagen que se quiere subir,
nombre_repositorio por la ruta del repositorio donde se quiere almacenar la imagen, y
etiqueta por la etiqueta que se quiere asignar a la imagen.

# docker tag 051123 gbermejosanchez/dw1e_2324:051123
```

20. **Subir la imagen etiquetada** al repositorio de Docker (docker repository_name/image_name:tag)

```
# docker push gbermejosanchez/dw1e_2324:051123
```

Después de seguir estos pasos, la imagen se cargará en el repositorio Docker que haya especificado y estará disponible para su uso y distribución.

21. Verificar que se ha subido la imagen al repositorio de Docker Hub. Utilizar la interfaz web de Docker Hub.



3.- CONCLUSIONES

Explica con tus palabras lo que has aprendido con esta práctica

Ejecución de Linux en Windows sin y con contenedores

Desarrollada por **Mª Concepción Fernández Fernández**

1.- OBJETIVOS

Conocer las tecnologías Hyer-V, Windows Machic Plataform y WSL y su relación con contenedores

Ejecutar SO Linux en SO Windows Sin contenedores.

Ejecutar Contenedores en Windows con Docker

Ejecutar Contenedores en Windows con Docker Desktop

2.- DEBES SABER

En Windows tenemos varias tecnologías que permiten virtualizar SO incluidos Linux

- **Hyper-V** es una tecnología de virtualización que permite crear y administrar máquinas virtuales en entornos Windows. No está disponible en todos los hardware ni todas las versiones de SO Windows
- **Windows Machine Plataform** proporciona un **entorno ligero de virtualización** y usa Hiper-V
- **WSL** es una **capa de compatibilidad** entre llamadas de sistema Linux y Windows

Todas son característica avanzadas de Windows que podemos habilitar:

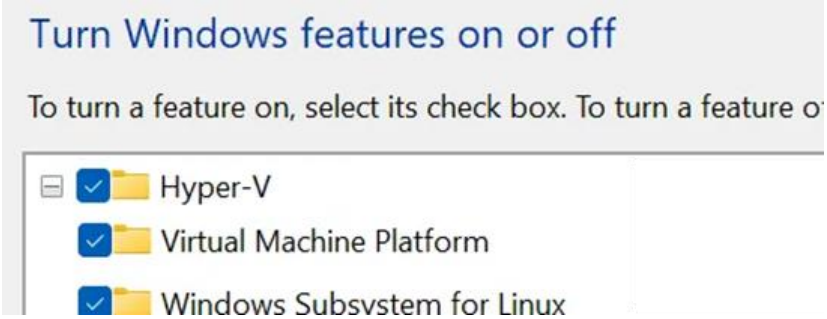
- Desde PowerShell:

  Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Windows-Subsystem-Linux

  Enable-WindowsOptionalFeature -Online -FeatureName VirtualMachinePlatform

  Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Hyper-V –All

- Por GUI



Puedes utilizar Docker en Windows 10 tanto con WSL (Windows Subsystem for Linux) como con **Hyper-V**. Ambos enfoques tienen sus ventajas y dependen de tus necesidades y preferencias específicas.

Además tenemos disponible la aplicación **Docker Desktop**. No es estrictamente necesario utilizar Docker Desktop, pero es una opción muy interesante que nos facilita las configuraciones y mejora la experiencia de usuario para trabajar con contenedores Docker de forma gráfica. Una de las utilidades más usadas es poder cambiar desde su icono de la barra de tareas si queremos virtualizar contenedores Windows o Linux. Ya que a la vez ¿no podemos tener los dos kernel?

3.- DEBES INVESTIGAR

Si solo quieres ejecutar un entorno linux Linux sobre WLS sin contenedores echa un vistazo a https://learn.microsoft.com/es-es/training/modules/wsl-introduction/install-and-setup
Si lo que necesitamos es trabajar con contenedores necesitados Docker
https://docs.docker.com/

Si quieres usar Docker sobre WSL2 https://blog-es.mimacom.com/docker-sin-docker-desktop-parte-1-windows-wsl2/
https://www.sitepoint.com/wsl2/

Si quieres tener una mejor experiencia de usuario usa Docker desktop en Windows
https://learn.microsoft.com/en-us/windows/wsl/tutorials/wsl-containers
https://docs.docker.com/desktop/install/windows-install/

Otros recursos Web interesantes:
https://www.youtube.com/watch?v=vuV9JoHiYaU
09:14 Instalación de Virtual Machine Platform
10:03 Instalación de WSL
10:27 Instalación de Ubuntu

10:12 Instalación de Docker Desktop
13:23 Instalación de Portainer
14:48 Docker Hub
15:54 Diagrama de contenedores a crear
16:37 Creación de volúmenes
17:03 Creación de redes
17:41 Creación del contenedor MySQL
22:17 Creación del contenedor Wordpress
25:59 Prueba del Wordpress instalado

https://www.youtube.com/watch?v=ZO4KWQfUBBc

4.- Reto 1: Ejecuta un sistema operativo Linux en Windows usando WSL2

1. Comprueba si la máquina cumple los requisitos necesarios
2. Antes de empezar decide que necesitas saber y hacer para conseguir el reto. Para ello ten en cuenta todo lo que has investigado.
3. Documenta la implementación. Esto te permitirá analizar el proceso si no consigues lo esperado y repasar lo hecho si todo ha salido bien.

5.- Reto 2: Ejecuta un sistema operativo Linux en un contenedor sobre sistemas Windows

1.- Comprueba si la máquina cumple los requisitos necesarios.
2.- Antes de empezar decide que necesitas saber y hacer para conseguir el reto. Para ello ten en cuenta todo lo que has investigado.
3.- Documenta la implementación. Esto te permitirá analizar el proceso si no consigues lo esperado y repasar lo hecho si todo ha salido bien.

6.- Conclusiones

Explica con tus palabras lo que has aprendido con esta práctica.

Añade las URL nuevas que te hayan servido para ampliar y reforzar los conocimientos necesarios para conseguir los retos.

Redes y comunicación entre contenedores

Desarrollada por **Mercedes Rodríguez Villafáfila**

Índice de Contenidos

Objetivo de la práctica

Nuestro objetivo es, en primer lugar, repasar el manejo de imágenes y contenedores Docker.

En segundo lugar, comprender cómo maneja Docker Engine las redes y las conexiones a las interfaces de red de los contenedores para, posteriormente, hacer pruebas de conexión de contenedores a redes diferentes y de comunicación entre contenedores.

Pruebas con una imagen existente en Docker Hub

Se puede descargar (pull) o directamente bajar y ejecutar (*run*) una imagen existente en Docker Hub con:

```
$ docker pull [developer/]image[:image-version]
$ docker run [-dit] [developer/]image[:image-version]
```

Si no se especifica version (tag) se utilizará la última versión (latest).

Por ejemplo, para las pruebas descarguemos la última imagen oficial "httpd" (apache2 bajo Debian):

```
$ docker pull httpd:latest
```

Para mostrar el listado de imágenes locales (**docker image ls** or **docker images**):

```
$ docker images
```

```
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
httpd         latest    6fd77d7e5eb7   2 months ago   167MB
hello-world   latest    d2c94e258dcb   8 months ago   13.3kB
```

Para mostrar la información de la imagen en formato JSON (**docker image inspect *image***):

```
$ docker image inspect httpd
```

```
[
    {
        "Id": "sha256:6fd77d7e5eb732dacab601d4556c04a6c312928fb8989fe3b0a47d82db772441",
        "RepoTags": [
            "httpd:latest"
        ],
        "RepoDigests": [
            "httpd@sha256:f0a93744d8006e6f7ee5086c9ddccdcfa33d1091f15269a00547b4c382459c1f"
        ],
        "Parent": "",
        "Comment": "buildkit.dockerfile.v0",
        "Created": "2023-10-19T11:31:14Z",
        "Container": "",
        "ContainerConfig": {
            "Hostname": "",
            "Domainname": "",
            "User": "",
            "AttachStdin": false,
            "AttachStdout": false,
            "AttachStderr": false,
```

```
            "Tty": false,
            "OpenStdin": false,
            "StdinOnce": false,
            "Env": null,
            "Cmd": null,
            "Image": "",
            "Volumes": null,
            "WorkingDir": "",
            "Entrypoint": null,
            "OnBuild": null,
            "Labels": null
        },
        "DockerVersion": "",
        "Author": "",
        "Config": {
            "Hostname": "",
            "Domainname": "",
            "User": "",
            "AttachStdin": false,
            "AttachStdout": false,
            "AttachStderr": false,
            "ExposedPorts": {
                "80/tcp": {}
            },
            "Tty": false,
            "OpenStdin": false,
            "StdinOnce": false,
            "Env": [

"PATH=/usr/local/apache2/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin",
                "HTTPD_PREFIX=/usr/local/apache2",
                "HTTPD_VERSION=2.4.58",

"HTTPD_SHA256=fa16d72a078210a54c47dd5bef2f8b9b8a01d94909a51453956b3ec6442ea4c5",
                "HTTPD_PATCHES="
            ],
            "Cmd": [
                "httpd-foreground"
            ],
            "ArgsEscaped": true,
            "Image": "",
            "Volumes": null,
            "WorkingDir": "/usr/local/apache2",
            "Entrypoint": null,
            "OnBuild": null,
            "Labels": null,
            "StopSignal": "SIGWINCH"
        },
        "Architecture": "amd64",
        "Os": "linux",
        "Size": 167482044,
        "VirtualSize": 167482044,
        "GraphDriver": {
            "Data": {
                "LowerDir":
"/var/lib/docker/overlay2/2db07802c7d4e09fb32c5d094480ef9ee7ade124f694d3bfba0d5ddb417276ae/
diff:/var/lib/docker/overlay2/e0ecebfeae46897290f9b1c2518fc5e6a457484e0545b75a6cfa9feca2598
383/diff:/var/lib/docker/overlay2/b119e7db4cb62ec5ad4c05f218c8a129c581bdc02775f1628ed8c079d
236c1a4/diff:/var/lib/docker/overlay2/9177e5cce1a0de674654dd57c7024e7fdc4c1cc72a86ae83c1847
36626f4b240/diff:/var/lib/docker/overlay2/bd7e402233d517aec2bc906cf35af89b63b1a0d2c5082eaa4
23a858e6e0d1994/diff",
```

```
                "MergedDir":
"/var/lib/docker/overlay2/822cb44ea5482cb00e1ba5d001c050aeca8edc2d52ba23bc0c7d04fabf4658da/
merged",
                "UpperDir":
"/var/lib/docker/overlay2/822cb44ea5482cb00e1ba5d001c050aeca8edc2d52ba23bc0c7d04fabf4658da/
diff",
                "WorkDir":
"/var/lib/docker/overlay2/822cb44ea5482cb00e1ba5d001c050aeca8edc2d52ba23bc0c7d04fabf4658da/
work"
            },
            "Name": "overlay2"
        },
        "RootFS": {
            "Type": "layers",
            "Layers": [
                "sha256:7292cf786aa89399bca4e3edd105d3b2ee0683a46ef1f5ff436c0f9d1d49e765",
                "sha256:1b65777f123810bd9b450a5e6505edfae1b4ffc4b326efcb66c827951893a1ef",
                "sha256:5f70bf18a086007016e948b04aed3b82103a36bea41755b6cddfaf10ace3c6ef",
                "sha256:87939ee964f4a95c403b7144dd470aa335042501d52cbca98eaf7ca2156657a4",
                "sha256:8c10599774e2a026e5338903f54170de38589f807613dd68665550d65a204b05",
                "sha256:8cd4026118f7e29ba95bc1c80c73f08acb8a4b9fee28b459a8e33404edcb9339"
            ]
        },
        "Metadata": {
            "LastTagTime": "0001-01-01T00:00:00Z"
        }
    }
]
```

EJECUCIÓN DE UN CONTENEDOR A PARTIR DE UNA IMAGEN

Recordando, en general para ejecutar un contenedor a partir de una imagen:

```
$ docker run -dit [--rm] [--name nombre-contenedor] [-p [ip-de-host:]port-de-host:port-de-
contenedor] imagen [commando]
```

El comando devuelve el identificador del contenedor.

Parámetros[1]:

- **"-d"** indica que se ejecute el contenedor en segundo plano. Esto inicia el contenedor sin ocupar la ventana del terminal. Por diseño, los contenedores iniciados en modo separado salen cuando el proceso raíz utilizado para ejecutar el contenedor sale. Para realizar entradas/salidas con un contenedor independiente, utilice conexiones de red o volúmenes compartidos. Estos son necesarios porque el contenedor ya no escucha la línea de comando donde se ejecutó docker run.
- **"-it"** indica que se ejecute el contenedor en modo interactivo (conectado a la entrada estándar por teclado) y con un TTY (conectando el terminal a los flujos de E/S del contenedor).
- **"--rm"** indica que Docker limpie automáticamente el contenedor y elimine el sistema de archivos cuando el contenedor salga, incluidos los volúmenes anónimos asociados con el contenedor.

[1] https://docs.docker.com/reference/cli/docker/container/run/

- **"--name"** permite dar nombre al contenedor. Si se excluye, Docker asigna un nombre aleatorio.
- **"-p"** ("**--publish**") permite mapear un puerto libre del host Docker con un puerto del contenedor. De forma predeterminada, el contenedor no expone ninguno de sus puertos al mundo exterior. Esto crea una regla de firewall en el host del tipo de redireccionamiento (-j DNAT).

Siguiendo nuestro ejemplo:

```
$ docker run -dit --rm --name apache httpd
```

En este caso, sin la bandera **-p**, el servidor Apache dentro del contenedor solo puede ser accedido desde un contenedor coectado a la misma red por defecto (bridge) o desde el host Docker.

Por tanto, si se quiere alcanzar al servidor Apache desde cualquier host, se necesita mapear puertos; por ejemplo:

```
$ docker run -dit --name apache -p 80:80 httpd
$ docker ps
```

```
CONTAINER ID    IMAGE    COMMAND              CREATED        STATUS         PORTS
NAMES
d19209381b1d    httpd    "httpd-foreground"   21 hours ago   Up 21 hours    0.0.0.0:80-
>80/tcp   apache
```

TRUCOS PARA EJECUTAR UN CONTENEDOR ININTERRUMPIDAMENTE

Al intentar ejecutar en un contenedor una imagen que no tiene un proceso que se ejecute indefinidamente (como un servidor HTTP), se detendrá tan pronto como se inicie. Para que la imagen no se detenga existen diferentes "trucos"[2] que consisten en iniciar el contenedor ejecutando un comando del sistema operativo del contenedor que se ejecute indefinidamente.

Por ejemplo, si queremos ejecutar de Ubuntu indefinidamente, se pueden usar varias opciones:

```
$ docker run ubuntu tail -f /dev/null
$ docker run ubuntu while true; do sleep 1; done
$ docker run ubuntu sleep infinity
```

INTERACCIÓN CON EL CONTENEDOR

Una vez que un contenedor está iniciado se puede usar **docker attach** para vincular la entrada y salida estándars del terminal al contenedor en ejecución. Esto va a permitir controlar interactivamente al contenedor y ejecutar comandos como si estuvieran ejecutando directamente en el terminal del host:

```
$ docker attach nombre-contenedor
```

[2] https://www.baeldung.com/ops/running-docker-containers-indefinitely

Alternativamente, se puede ejecutar un comando en un terminal intrecativo en el contenedor; por ejemplo:

```
$ docker exec -it apache bash
root@d19209381b1d:/usr/local/apache2#
```

Ahora podemos interactuar con el terminal del servidor Apache (dentro del contenedor):

```
root@d19209381b1d:/usr/local/apache2# ls
```

```
bin  build  cgi-bin  conf  error  htdocs  icons  include  logs  modules
```

```
root@89d010925729:/usr/local/apache2# ls conf
```

```
extra  httpd.conf  magic  mime.types  original
```

```
root@d19209381b1d:/usr/local/apache2# ls htdocs/
```

```
index.html
```

```
root@d19209381b1d:/usr/local/apache2# cat htdocs/index.html
```

```
<html><body><h1>It works!</h1></body></html>
```

SALIR DEL TERMINAL DE UN CONTENEDOR

Hay dos formas de salir del terminal de un contenedor:

- Ejecutando "exit" en la CLI del contenedor. El contenedor se parará.
- Con la secuencia de control de escape: "Ctrl + p + q". El contenedor seguirá ejecutándose.

CONEXIÓN DE CONTENEDORES A REDES Y COMUNICACIÓN ENTRE CONTENEDORES

Conexión por defecto

Por defecto, cuando se arranca un contenedor sin especificar red se añade automáticamente a la red "bridge" con IP dinámica del rango 172.17/16, mediante una interfaz lógica denominada "docker0" que se ha creado automáticamente en el host. El servidor Docker Engine hace de router NAT. La puerta de enlace es 172.17.0.1 y utiliza la misma configuración DNS que el host (aunque se puede cambiar). La primera IP del rango es 172.17.0.2 y se van asignando secuencialmente a los contenedores que se vayan lanzando. El alcance de la conexión del contenedor es local (solo está conectado al host Docker).

Cualesquiera contenedores arrancados en la red por defecto se pueden comunicar entre sí y con el host.

Por ejemplo, podemos comprobar la conectividad:

```
$ ip address show   ($ ip a)
```

```
3: docker0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
```

```
    link/ether 02:42:74:09:27:e4 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
    inet6 fe80::42:74ff:fe09:27e4/64 scope link
       valid_lft forever preferred_lft forever
```

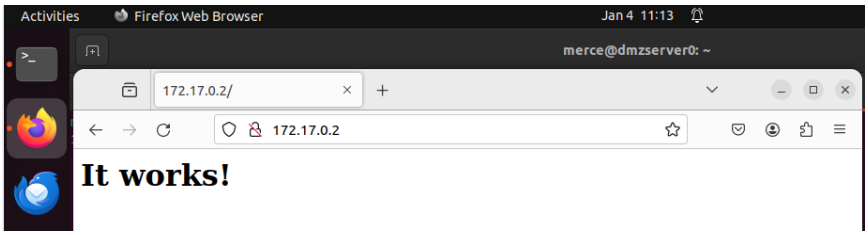
```
$ ping -c1 172.17.0.1
```

```
PING 172.17.0.1 (172.17.0.1) 56(84) bytes of data.
64 bytes from 172.17.0.1: icmp_seq=1 ttl=64 time=0.049 ms

--- 172.17.0.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.049/0.049/0.049/0.000 ms
```

```
$ ping -c1 172.17.0.2
```

```
PING 172.17.0.2 (172.17.0.2) 56(84) bytes of data.
64 bytes from 172.17.0.2: icmp_seq=1 ttl=64 time=0.036 ms

--- 172.17.0.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.036/0.036/0.036/0.000 ms
```

Como tenemos un servidor Apache instalado en el contenedor, podemos probar la URL http://172.17.0.2:



Alternativamente, se puede comprobar con el comando:

```
$ curl 172.17.0.2
```

```
<html><body><h1>It works!</h1></body></html>
```

Redes Docker predefinidas

Para listar las redes Docker predeterminadas:

```
$ docker network ls
```

```
NETWORK ID     NAME      DRIVER    SCOPE
a4909b60a618   bridge    bridge    local
334a2bd857b8   host      host      local
457198aadbe4   none      null      local
```

Como ya se ha explicado, si al arrancar un contenedor no se especifica ninguna red, este se conecta a la **red "bridge"** con prefijo 172.17/16. Esta red se usa principalmente cuando se quiere que una aplicación ejecutándose en el host local se comunique con otras aplicaciones o servicios corriendo en el mismo host. Para exponer cualquier puerto y que sea accesible desde el exterior es necesario que se inicie el contenedor con la opción "**-p**" para mapear puertos entre el contenedor y el host.

Observación: El mapeo de puertos con el flag "**-p**" tiene un efecto colateral: el servicio que esté escuchando detrás de ese puerto puede ser alcanzado desde cualquier equipo conectado a la misma LAN que el host Docker.

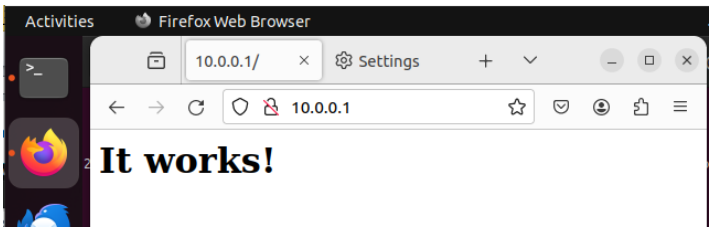
La **red** "**none**" aísla completamente a un contenedor del host y de otros contenedores. El contenedor no recibirá ninguna IP y no tendrá acceso a la red externa u otros contenedores. Únicamente tiene IP loopback.

Si se conecta un contenedor a la **red** "**host**", el contenedor usa directamente la conexión de red del host y copia toda su configuración dentro del contenedor (inclusive el hostname). El contenedor es accesible a través de la misma IP del host. Por ejemplo:

```
$ docker run -d --network host --name apache-test httpd
$ docker ps
```

```
CONTAINER ID   IMAGE   COMMAND              CREATED          STATUS          PORTS
NAMES
f7286ecc28e7   httpd   "httpd-foreground"   12 seconds ago   12 seconds ago   80/tcp
apache-test
```

Ahora nos podemos conectar a la IP de host desde un navegador, por ejemplo:




```
$ curl 10.0.0.1
```

```
<html><body><h1>It works!</h1></body></html>
```

Obtener información de una red

Para inspeccionar la información de red en formato JSON: "**docker network inspect** ***network-name***". Por ejemplo:

```
$ docker network inspect bridge
```

```
[
    {
        "Name": "bridge",
        "Id": "51a2f0be8c0c3f9015a9cb56a1d96e3f3eb73d1b05e060c08019702d272314c1",
        "Created": "2024-01-04T10:02:04.386402124-05:00",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "172.17.0.0/16",
                    "Gateway": "172.17.0.1"
                }
            ]
```

```
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
          "d19209381b1da35c9696b384171855b6e7d945489050e2fa88bfa3cefd204b8d": {
                "Name": "apache",
                "EndpointID":
"27a3d9af225ef06e18c0551a4f3e6a34abb26ed35b9c87741dd75684c28179db",
                "MacAddress": "02:42:ac:11:00:02",
                "IPv4Address": "172.17.0.2/16",
                "IPv6Address": ""
            }
        },
        "Options": {
            "com.docker.network.bridge.default_bridge": "true",
            "com.docker.network.bridge.enable_icc": "true",
            "com.docker.network.bridge.enable_ip_masquerade": "true",
            "com.docker.network.bridge.host_binding_ipv4": "0.0.0.0",
            "com.docker.network.bridge.name": "docker0",
            "com.docker.network.driver.mtu": "1500"
        },
        "Labels": {}
    }
]
```

Creación de una imagen sencilla a partir de un contenedor en ejecución para las pruebas

Para hacer pruebas de comunicación entre contenedores, vamos a partir de una imagen sencilla que crearemos usando la oficial **httpd** y a la que añadiremos herramientas de comprobación de la configuración de red:

```
$ docker run -dit --rm --name test -p 80:80 httpd
```

```
5f534a5199e30d1166a4fa96e567b4605ad9434cfa7e7d54272da46a2eb53672
```

```
merce@dmzserver0:~$ docker images
```

```
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
httpd        latest    2776f4da9d55   7 weeks ago   167MB
```

```
$ docker ps
```

```
CONTAINER ID  IMAGE  COMMAND            CREATED        STATUS         PORTS
NAMES
5f534a5199e3  httpd  "httpd-foreground" 28 seconds ago Up 27 seconds  0.0.0.0:80->80/tcp, :::80-
>80/tcp   test
```

```
$ docker exec -it test /bin/bash
root@5f534a5199e3:/usr/local/apache2# apt update
```

Ejecutamos la instalación de las herramientas de red que utilizaremos:

```
root@5f534a5199e3:/usr/local/apache2# apt update
root@5f534a5199e3:/usr/local/apache2# apt install -y iputils-ping net-tools dnsutils
iproute2
root@5f534a5199e3:/usr/local/apache2# apt install -y nano grep
root@5f534a5199e3:/usr/local/apache2# apt -y autoclean && apt -y autoremove
```

Saldremos del CLI del contenedor sin cerrarlo con la secuencia de escape "Ctrl + p + q":

```
root@5f534a5199e3:/usr/local/apache2# read escape sequence
```

Para crear una nueva imagen a partir del contenedor hay que usar el comando **docker commit**[3]:

```
$ docker commit test testimage
```

```
sha256:e3dcc23b7d2713ee74d96b44faa020ecd49be77555ed845e3b55b02eae5c6ea7
```

```
$ docker images
```

```
REPOSITORY   TAG       IMAGE ID       CREATED          SIZE
testimage    latest    e3dcc23b7d27   14 seconds ago   200MB
httpd        latest    2776f4da9d55   7 weeks ago      167MB
```

A partir de ahora podemos usar esta imagen para ejecutar varios contenedores y hacer pruebas de comunicación entre ellos.

Crear y conectar contenedores a redes personalizadas[4]

Para crear redes nuevas hay que usar el comando **docker network create**:

```
$ docker network create [-d driver] [options] network-name
```

Si no se especifica la opción "**-d**" o"**--driver**", se creará automáticamente una red "bridge".

> Nota: Las redes "bridge" en principio están aisladas en el host Docker local. Si se desea crear una red que abarque varios hosts Docker, cada uno de los cuales ejecute Docker Engine, se debe habilitar el modo Swarm y crear una red "overlay"[5].

Si se quiere borrar una red, hay que utilizar el comando **docker network rm**.

Modos de conectar un contenedor a una o varias redes[6]:

- Cuando se inicia un contenedor con la opción "**--network**" solo puede conectarse a una única red:

  ```
  $ docker run -itd --network=nombre-red nombre-imagen
  ```

- Posteriormente, se puede conectar un contenedor en ejecución a otras redes mediante el comando **docker network connect**[7]. Para desconectarlo con el comando **docker network disconnect.**

  ```
  $ docker network connect nombre-red nombre-contenedor
  ```

[3] https://docs.docker.com/reference/cli/docker/container/commit/

[4] https://docs.docker.com/reference/cli/docker/network/create/

[5] https://docs.docker.com/network/drivers/overlay/

[6] https://docs.docker.com/network/

[7] https://docs.docker.com/reference/cli/docker/network/connect/

```
$ docker network disconnect nombre-red nombre-contenedor
```

- En ambos casos, puede utilizar los indicadores "**--ip**" o "**--ip6**" para especificar la dirección IP del contenedor en esa red en particular.

Por defecto, el hostname de un contenedor es su ID. Sin embargo, puede definirse un hostname en el arranque del contenedor usando la opción "**--hostname**".

Al conectarse a una red existente mediante "**docker network connect"** se puede usar el indicador "**--alias**" para especificar un alias de red adicional para el contenedor.

Test: Generar dos contenedores "testimage" en la misma red net1 (10.1/16) y otros dos en la red net2 (10.3/16) y probar si se pueden comunicar entre ellos

Crear las redes:

```
$ docker network create --driver=bridge --subnet=10.1.0.0/16 --gateway=10.1.0.1 net1
```

```
c68f4b9425c5595640794e32592ff25066f1dd2f39f9b437ba184b950be3f988
```

```
$ docker network create --driver=bridge --subnet=10.2.0.0/16 --gateway=10.2.0.1 net2
```

```
d7fd738077570540213c95e5fa64043bdbd78748280b72cc9e4aca0ae11ab1a7
```

```
$ docker network ls
```

```
NETWORK ID     NAME      DRIVER    SCOPE
39553cb1a02e   bridge    bridge    local
fd5c6df4d58a   host      host      local
c68f4b9425c5   net1      bridge    local
d7fd73807757   net2      bridge    local
744f8ade2d6e   none      null      local
```

Arrancar los contenedores conectándolos a sus respectivas redes:

```
$ docker run --rm -dit --name c1 --network=net1 --ip=10.1.0.11 testimage
$ docker run --rm -dit --name c2 --network=net1 --ip=10.1.0.12 testimage
$ docker run --rm -dit --name c3 --network=net2 --ip=10.2.0.21 testimage
$ docker run --rm -dit --name c4 --network=net2 --ip=10.2.0.22 testimage
$ docker ps
```

```
CONTAINER ID   IMAGE       COMMAND                  CREATED          STATUS          PORTS
NAMES
50176ccd04f1   testimage   "httpd-foreground"   6 seconds ago    Up 5 seconds    80/tcp
c4
81828ff740e0   testimage   "httpd-foreground"   47 seconds ago   Up 46 seconds   80/tcp
c3
85b37a6a319a   testimage   "httpd-foreground"   47 seconds ago   Up 46 seconds   80/tcp
c2
ffa4c099e68c   testimage   "httpd-foreground"   48 seconds ago   Up 47 seconds   80/tcp
c1
```

En el host el comando **ip a** muestra las nuevas interfaces como puertas de enlace de las redes net1 y net2:

```
...
4: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group
default
    link/ether 02:42:25:6d:fe:71 brd ff:ff:ff:ff:ff:ff
```

```
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
6: br-c68f4b9425c5: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group
default
    link/ether 02:42:84:48:89:2d brd ff:ff:ff:ff:ff:ff
    inet 10.1.0.1/16 brd 10.1.255.255 scope global br-c68f4b9425c5
       valid_lft forever preferred_lft forever
    inet6 fe80::42:84ff:fe48:892d/64 scope link
       valid_lft forever preferred_lft forever
7: br-d7fd73807757: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group
default
    link/ether 02:42:99:dd:09:92 brd ff:ff:ff:ff:ff:ff
    inet 10.2.0.1/16 brd 10.2.255.255 scope global br-d7fd73807757
       valid_lft forever preferred_lft forever
    inet6 fe80::42:99ff:fedd:992/64 scope link
       valid_lft forever preferred_lft forever
Conecta al terminal del contenedor c1 y prueba a hacer ping al resto de contenedores.
...
```

Para inspeccionar los contenedores conectados, por ejemplo, a la red net1:

```
$ docker network inspect net1
```

```
...
       "Containers": {
            "85b37a6a319ab850e2a61cbb69bf84047086d44fb15b6f3e4746fdf8832756c5": {
                "Name": "c2",
                "EndpointID": "c96f63ca35aa48f85aff59f4c519d48bc8afd3c031b2a9cadb176649659a35c7",
                "MacAddress": "02:42:0a:01:00:0c",
                "IPv4Address": "10.1.0.12/16",
                "IPv6Address": ""
            },
            "ffa4c099e68cf88c67e0016a7ac1355c5504ce2b2dd7c5cdb1808cea0dd194b4": {
                "Name": "c1",
                "EndpointID": "6b30810985ead86e60ee04196a19c8f6f5466a96f125ff4c423884bc5c6f5323",
                "MacAddress": "02:42:0a:01:00:0b",
                "IPv4Address": "10.1.0.11/16",
                "IPv6Address": ""
            }
        },
       "Containers": {
            "85b37a6a319ab850e2a61cbb69bf84047086d44fb15b6f3e4746fdf8832756c5": {
                "Name": "c2",
                "EndpointID": "c96f63ca35aa48f85aff59f4c519d48bc8afd3c031b2a9cadb176649659a35c7",
                "MacAddress": "02:42:0a:01:00:0c",
                "IPv4Address": "10.1.0.12/16",
                "IPv6Address": ""
            },
            "ffa4c099e68cf88c67e0016a7ac1355c5504ce2b2dd7c5cdb1808cea0dd194b4": {
                "Name": "c1",
                "EndpointID": "6b30810985ead86e60ee04196a19c8f6f5466a96f125ff4c423884bc5c6f5323",
                "MacAddress": "02:42:0a:01:00:0b",
                "IPv4Address": "10.1.0.11/16",
                "IPv6Address": ""
            }
        },
...
```

Para conectarse a la consola del primer contenedor:

```
$ docker exec -it c1 /bin/bash
root@ffa4c099e68c:/usr/local/apache2# ip a
```

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
10: eth0@if11: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group
default
    link/ether 02:42:0a:02:00:15 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.1.0.11/16 brd 10.1.255.255 scope global eth0
       valid_lft forever preferred_lft forever
```

Hacemos ping a la IP del contenedor c2 (tiene éxito):

```
root@ffa4c099e68c:/usr/local/apache2# ping -c1 10.1.0.12
```

```
PING 10.1.0.12 (10.1.0.12) 56(84) bytes of data.
64 bytes from 10.1.0.12: icmp_seq=1 ttl=64 time=0.171 ms

--- 10.1.0.12 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.171/0.171/0.171/0.000 ms
```

Hacemos ping a la IP del contenedor c3 (no tiene éxito):

```
root@ffa4c099e68c:/usr/local/apache2# ping -c1 10.2.0.21
```

```
PING 10.2.0.21 (10.2.0.21) 56(84) bytes of data.

--- 10.2.0.21 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms
```

Hacemos ping a la IP del 172.17.0.1 (tiene éxito):

```
root@ffa4c099e68c:/usr/local/apache2# ping -c1 172.17.0.1
```

```
PING 172.17.0.1 (172.17.0.1) 56(84) bytes of data.
64 bytes from 172.17.0.1: icmp_seq=1 ttl=64 time=0.043 ms

--- 172.17.0.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.043/0.043/0.043/0.000 ms
```

Lanzamos otro contenedor en la red "bridge" y probamos a hacer ping desde él a alguno de los contenedores cx (no tiene éxito):

```
$ docker run -d --rm --name test testimage
root@e56cb9b02c51:/usr/local/apache2# ping 172.17.0.2
```

```
PING 172.17.0.2 (172.17.0.2) 56(84) bytes of data.
64 bytes from 172.17.0.2: icmp_seq=1 ttl=64 time=0.105 ms
64 bytes from 172.17.0.2: icmp_seq=2 ttl=64 time=0.048 ms
64 bytes from 172.17.0.2: icmp_seq=3 ttl=64 time=0.048 ms
^C
--- 172.17.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2031ms
rtt min/avg/max/mdev = 0.048/0.067/0.105/0.026 ms
```

```
root@e56cb9b02c51:/usr/local/apache2# ping 10.2.0.21
```

```
PING 10.2.0.21 (10.2.0.21) 56(84) bytes of data.
^C
--- 10.2.0.21 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3066ms
```

Más pruebas que se pueden hacer:

- Comprobar que desde el host Docker se puede hacer ping a todos los contenedores sin excepción.
- Conectar por ejemplo el contenedor 4 a la red net1 y probar las conectividades con el resto de contenedores.

LISTA RESUMEN DE COMANDOS DOCKER

Command	Use	Example
docker search [--no-trunc] *imagen*	**Buscar imágenes en el repositorio Docker Hub**	**docker search httpd**
docker pull [*desarrollador/*]*imagen*[:*image-version*]	Descargar una imagen existente	docker pull httpd
docker image ls docker images	Listar las imágenes instaladas en el host local	
docker image inspect *imagen*	Obtener información de las imágenes	docker image inspect httpd
docker run [-dit] [--rm] [--name *contenedor*] [-p *port*:*host-port*] *imagen* [*comando*]	Crear y ejecutar un contenedor de una imagen descargada	docker run -dit --name myhttpd -p 8080:80 httpd
docker exec [-it] [--rm] [--name *contenedor*] [-p *port*:*host-port*] *comando*	Ejecutar un comando en un contenedor con la posibilidad de mapear puertos entre el host local y el contenedor	docker exec -it --name myhttpd -p 8080:80 bash
docker ps docker container ls	Listar los contenedores actuales activos	
docker ps -a docker container ls -a	Listar los contenedores actuales activos y no activos	
docker network ls	Listar todas las redes	
docker network inspect *nombre-red*	Obtener información de una red	docker network inspect host
docker start *contenedor*	Iniciar un contenedor	docker start myhttpd
docker stop *contenedor*	Parar un contenedor	docker stop myhttpd
docker remove *contenedor* docker rm *contenedor*	Borrar un contenedor	docker rm myhttpd
docker rmi *imagen*	Borrar una imagen	docker rmi httpd
docker kill $(docker ps -q)	Parar todos los contenedores	
docker rm $(docker ps -a -q)	Borrar todos los contenedores	
docker rmi -f $(docker images -q)	Borrar todas las imágenes	
docker commit *contenedor imagen*	Crear una imagen a partir de un contenedor en ejecución	docker commit test testimage

Referencias

Documentación Docker: https://docs.docker.com/

Cómo iniciarse en Docker:

- https://www.docker.com/blog/top-questions-for-getting-started-with-docker/
- Docker quick hands-on guides: https://docs.docker.com/guides/get-started/

Comandos Docker:

https://docs.docker.com/get-started/docker_cheatsheet.pdf
Docker Cheat Sheet | Coursera
https://dockerlabs.collabnix.com/docker/cheatsheet/

Docker Networking:

- https://docs.docker.com/network/
- https://docs.docker.com/network/packet-filtering-firewalls/

Conclusiones

Explica con tus palabras lo que has aprendido con esta práctica.

Microservicio web creado a partir de DockerFile

Desarrollada por **Mercedes Rodríguez Villafáfila**

Objetivo de la práctica

Nuestro objetivo es crear un servicio web completo para alojar uno o varios hosts virtuales para aplicaciones web. Para ello vamos a crear una imagen personalizada para crear contenedores que encapsulen el servicio web. Lo haremos a partir de un fichero de descripción de la imagen y, posteriormente, publicaremos la imagen creada en un registro público de imágenes Docker, como es Docker Hub.

Crear una nueva cuenta en el repositorio "Docker Hub"

Docker Hub es una biblioteca online de imágenes Docker. La mayoría son públicas y se pueden bajar (*pull*) y usar libremente.

Regístrate en Docker Hub (https://hub.docker.com/signup). Es gratuito y solo requiere una cuenta de correo electrónico y una ID Docker (usuario). Esta ID proporciona un espacio personal de repositorio y permite el acceso a todos los servicios en la plataforma Docker Hub.

Observación: Se cuidadoso con tu nombre de usuario, porque no se puede cambiar a posteriori. Por ejemplo: si el usuario se llama **Fu**lanito **Men**gano **Zu**tano, un buen ID sería: "fumenzu".

Creación de una imagen para un servicio web con Dockerfile (microservicio)

Se quiere crear una imagen personalizada basada en una preexistente en Docker Hub de Apache sobre Ubuntu a la que se quiere añadir programas y utilidades y servicios web previamente definidos.

Un **Dockerfile** es un archivo de texto que contiene las instrucciones que crean la imagen. Debe guardarse dentro de un contexto de compilación, es decir, un directorio. Este directorio debe contener todos los archivos necesarios para construir la imagen.

Dentro de un Dockerfile, CMD o ENTRYPOINT se utilizan para especificar los programas o comandos que se ejecutarán al inicializar un contenedor desde una imagen. Las principales diferencias entre ambos son:

- CMD establece parámetros predeterminados que se pueden anular desde la interfaz de línea de comandos al iniciar un contenedor. Sólo se permite una sentencia CMD, y si hay varias, sólo surte efecto la última.

- ENTRYPOINT establece parámetros predeterminados que no se pueden anular al iniciar un contenedor.

Más información:

https://docs.docker.com/engine/reference/builder/
https://docs.docker.com/build/guide/layers/

Creación del contexto de la imagen

Primero hay que crear el contexto de la imagen, que consiste en un directorio con todos los ficheros que incluiremos en la imagen desde el Dockerfile:

```
$ mkdir -p /path/to/Docker/website-img
```

Por ejemplo, lo crearemos en nuestro directorio home:

```
$ mkdir -p ~/Docker/website-img
$ cd ~/Docker/website-img
$ nano Dockerfile
```

Edición de los ficheros de configuración del microservicio web

Crearemos en el contexto los ficheros de configuración del servicio web que necesitaremos en la imagen. En particular para este servicio web tendremos dos hosts virtuales:

- El primer host virtual dará servicio a una aplicación web comercial de una hipotética empresa, cuyo nombre de dominio denominaremos genéricamente “website”. El nombre website lo puedes sustituir por un FQDN: *ejemplo.org*, por ejemplo.

- El segundo dará servicio a la intranet de los empleados de la hipotética empresa.

Edita los ficheros de configuración del servicio web para el primer host virtuales:

```
$ nano website.conf
```

```
<VirtualHost *:80>
        ServerName      website
        ServerAlias     www.website
        DocumentRoot    /var/www/website/html
        DirectoryIndex  index.html

       <Directory /var/www/website/html>
           Options -Indexes
           AllowOverride None
       </Directory>

        ErrorLog /var/www/website/log/error.log
        CustomLog /var/www/website/log/access.log combined

        Alias /icons /var/www/icons

        ErrorDocument 404 "<H1>NOT FOUND!!!!</H1>"
        ErrorDocument 403 /var/www/website/403.html

</VirtualHost>
```

```
# nano 403.html
```

```
<H1>**** FORBIDDEN ACCESS (ErrorDocument 403) ****</H1>
```

Edita los ficheros de configuración del servicio web para el segundo host virtual:

```
$ nano intranet.website.conf
```

```
<VirtualHost *:8080>
        ServerName      intranet.website
        ServerAlias     intranet
        ServerAdmin     webmaster@localhost

        DocumentRoot    /var/www/intranet.website/html
        DirectoryIndex  index.html home.html

       <Directory /var/www/intranet.website/html>
           Options +Indexes
           AllowOverride AuthConfig
       </Directory>

        ErrorLog /var/www/intranet.website/log/error.log
        CustomLog /var/www/intranet.website/access.log combined

</VirtualHost>
```

Edita los ficheros de configuración de apache, al menos los puertos de escucha de Apache:

```
$ nano ports.conf
```

```
Listen 80
Listen 8080

<IfModule ssl_module>
        Listen 443
</IfModule>

<IfModule mod_gnutls.c>
        Listen 443
</IfModule>
```

Edición del Dockerfile

Edita en el contexto el fichero de definición de la imagen del microservicio web. Puedes actualizar la imagen de Ubuntu de partida a la versión que estés usando actualmente de la distribución Ubuntu.

```
$ nano Dockerfile
```

```
# The base image is the 22.04 version of official Ubuntu image:
FROM ubuntu:22.04

# Metadatos:
MAINTAINER "tunombredeusuarioDockerHub"
LABEL maintainer "Nombre del autor y correo electrónico"
LABEL container "Microservicio web basado en Apache bajo Ubuntu:22.04"

# Copia este Dockerfile al sistema de ficheros de la imagen:
COPY Dockerfile /Dockerfile

# Actualiza e instala nano, ping y otras utilidades:
RUN apt update -y
```

```
RUN apt install -y nano iputils-ping net-tools dnsutils iproute2
RUN apt install -y curl lynx ftp ftp-ssl grep
RUN apt install wget

# Instala el servidor Apache de forma no interactiva y con zona horaria de Madrid:
RUN DEBIAN_FRONTEND=NONINTERACTIVE apt install -y apache2 tzdata
ENV TZ=Europe/Madrid

# Elimina ficheros innecesarios tras las instalaciones:
RUN apt -y autoclean && apt -y autoremove

# Expone los puertos que usará Apache en la imagen:
EXPOSE 80
EXPOSE 8080
EXPOSE 443

# Copia los ficheros de configuración del servicio web dentro de la imagen:
#COPY ./apache2.conf           /etc/apache2/apache2.conf
COPY ./ports.conf              /etc/apache2/ports.conf
COPY ./website.conf            /etc/apache2/sites-available/website.conf
COPY ./intranet.website.conf   /etc/apache2/sites-available/intranet.website.conf

# Crea la estructura de directorios del servicio web:
RUN mkdir /var/www/website
RUN mkdir /var/www/website/html
RUN mkdir /var/www/website/log
RUN mkdir /var/www/intranet.website
RUN mkdir /var/www/intranet.website/html
RUN mkdir /var/www/intranet.website/log

# Copia el fichero 403.html:
COPY ./403.html     /var/www/website/403.html

# Descarga algún fichero de icono para las pruebas:
RUN mkdir /var/www/icons
RUN cd /var/www/icons
# Descarga algunos ficheros para las pruebas:
RUN wget https://cdn-icons-png.flaticon.com/128/566/566359.png
RUN wget https://cdn-icons-png.flaticon.com/128/126/126509.png

# Añade la variable de entorno LYNX a Apache (ojo, esto podría dar problemas):
RUN echo 'export APACHE_LYNX="www-browser -dump' >> /etc/apache2/apache2.conf

RUN echo "<H1>THIS IS THE **WEBSITE PORTAL** (intranet.website.conf) </H1>" >
var/www/website/html/index.html
RUN echo "<H1>**** FORBIDDEN ACCESS ****</H1>" > /var/www/website/403.html

RUN echo "<H1>THIS IS THE **intranet's WEBSITE** (intranet.website.conf) </H1>" >
var/www/intranet.website/html/home.html

# Podríamos incluso desplegar una aplicación web desde un fichero tar de la aplicación:
#ADD app /var/www/website/html/app

# Habilita algunos módulos de Apache útiles:
RUN a2enmod rewrite ssl auth_digest info userdir

# Habilita los dos host virtuales que se quieren proporcionar en el servidor web:
RUN a2ensite  website  intranet.website

###FINALIZA EL DOCKERFILE ACTIVANDO EL SERVICIO WEB. DOS POSIBILIDADES PARA ARRANCAR APACHE
DENTRO DEL CONTENEDOR:

##1ª POSIBILIDAD##
# Reinicia Apache:
RUN apache2ctl reload
# Ejecuta en CMD un script que mantiene al contenedor ejecutándose indefinidamente:
CMD sh -c 'trap "exit" TERM; while true; do sleep 1; done'
```

```
##2ª POSIBILIDAD##
# Ejecuta en CMD el demonio de Apache en primer plano:
# CMD ["httpd-foreground"]
```

Creación de la imagen del servicio web (*build*)

Para construir o crear la imagen en el directorio actual (*build context*):

```
$ docker build . --tag user-spacename/image:version
```

Por ejemplo:

```
$ docker build . --tag mercerovi/website-img:1.0
```

Por defecto, el comando **docker build** buscará un fichero Dockerfile en la ruta raiz del contexto.

Comprueba ahora la lista de imágenes:

```
$ docker images
```

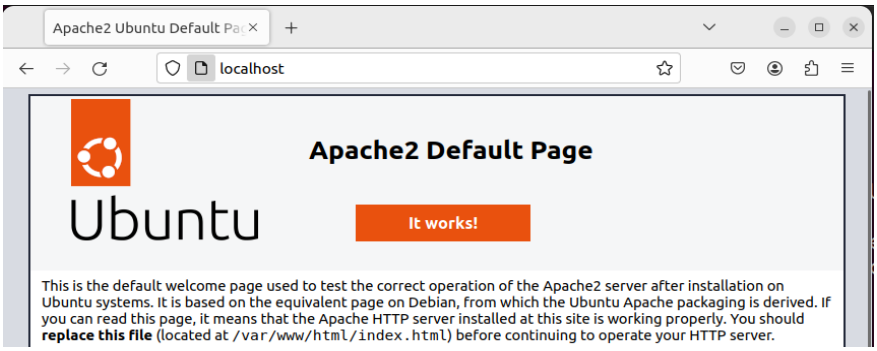
```
REPOSITORY              TAG       IMAGE ID       CREATED          SIZE
mercerovi/website-img   1.0       8e93020a2034   19 minutes ago   248MB
ubuntu                  22.04     174c8c134b2a   3 weeks ago      77.9MB
alpine                  latest    f8c20f8bbcb6   4 weeks ago      7.38MB
httpd                   latest    6fd77d7e5eb7   2 months ago     167MB
hello-world             latest    d2c94e258dcb   8 months ago     13.3kB
```

Prueba a iniciar un contenedor a partir de esa imagen y pruébalo:

```
$ docker run -dit --name mi-web -p 80:80 mercerovi/website-img:1.0
$ docker ps -a
```

```
CONTAINER      ID  IMAGE                    COMMAND                  CREATED         STATUS
PORTS                                       NAMES
10e0be6df158  mercerovi/website-img:1.0 "apache2ctl -D FOREG…"  21 minutes ago  Up 4
seconds  0.0.0.0:80->80/tcp, :::80->80/tcp, 443/tcp   mi-web
```

Ahora podemos conectar con el servidor web desde un navegador usando cualquiera de las IP: localhost, 172.17.0.2, desde el host local, o con la IP del host en la LAN, desde cualquier cliente. Mostrará el fichero index.html instalado por defecto o, en su caso, la aplicación web que hallamos creado y desplegado desde el Dockerfile:



SUBIDA DE LA IMAGEN A DOCKER HUB (*PUSH*)

La imagen recién creada todavía reside en el host local. Para logar que la imagen esté disponible desde cualquier equipo, hay que subirla (*push*) al repositorio de Docker Hub.

Primero hay que iniciar sesión en Docker Hub desde el equipo local:

```
$ docker login
```

```
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a
Docker ID, head over to https://hub.docker.com to create one.
Username: mercerovi
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/$credentials-store

Login Succeeded
```

Luego, hay que ejecutar el comando que realiza la subida:

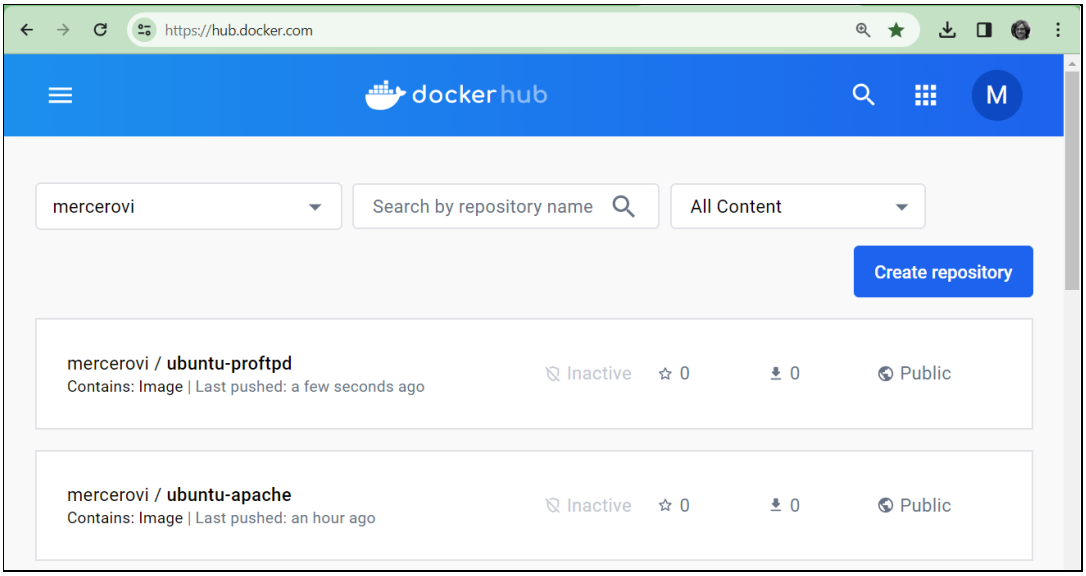
```
$ docker push user-spacename/image:version
```

Por ejemplo:

```
$ docker push mercerovi/website-img:1.0
```

```
The push refers to repository [docker.io/mercerovi/website-img]
45ed32690652: Pushed
9fb869267de7: Pushed
e00cbe888a66: Pushed
77cd5ea038ab: Pushed
86c605760a10: Pushed
7d8c457a030a: Pushed
25686f74ca5d: Pushed
95bc5d517917: Pushed
0a64636f9dc9: Pushed
a1360aae5271: Mounted from library/ubuntu
1.0: digest: sha256:c4c3745f48af87428f0bc42f9ff9b26e5fc1ff24bb0c23b382348d4dbe2d874e size:
2413
```

Iniciando sesión en Docker Hub a través de hub.docker.com, puedes comprobar en Docker Hub las imágenes ya subidas al repositorio. Por ejemplo:



Lista resumen de comandos Docker

Command	Use	Example
docker build ruta/al/contexto --tag nombreusuario/imagen:versión	Crear una imagen a partir de un contexto	
docker login	Iniciar session en Docker Hub	
docker push user-spacename/image:version	Subir una imagen a Docker Hub	

Referencias

Documentación Docker: https://docs.docker.com/

Dockerfile:

- https://docs.docker.com/engine/reference/builder/
- https://docs.docker.com/build/guide/layers/
- https://stackify.com/docker-build-a-beginners-guide-to-building-docker-images/
- https://devtron.ai/blog/cmd-and-entrypoint-differences/

Conclusiones

Explica con tus palabras lo que has aprendido con esta práctica.

Microservicios ACI Azure Cloud

Desarrollada por **Mercedes Rodríguez Villafáfila**

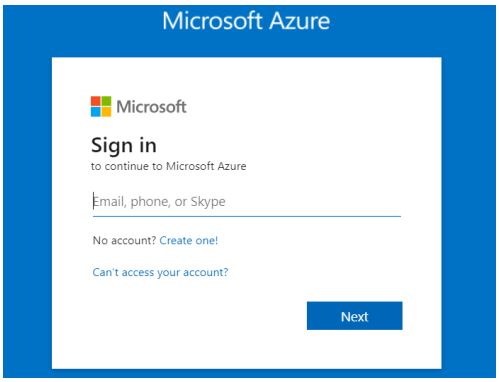
OBJETIVO DE LA PRÁCTICA

En esta práctica iniciarás sesión en una cuenta de administración de Azure e implementarás una instancia de contenedor (desde una imágen preexistente de Apache en Docker Hub) mediante Azure Portal.

INICIAR SESIÓN EN UNA CUENTA AZURE

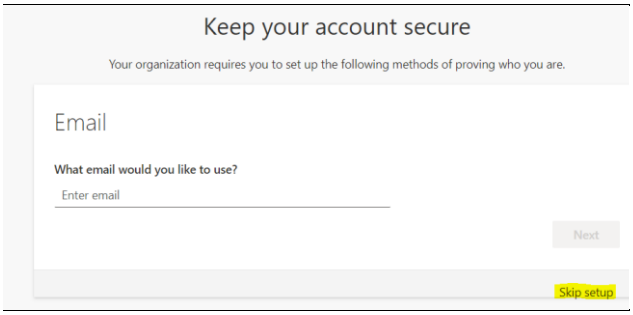
Inicia sesión en una cuenta Azure previamente creada:

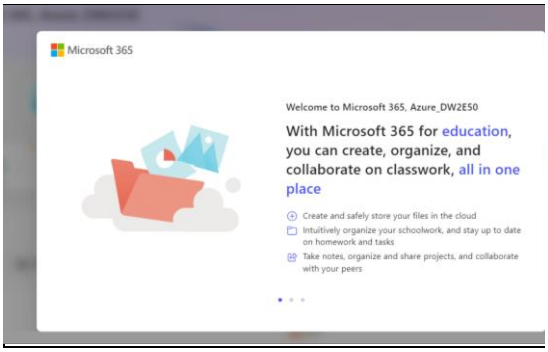
https://portal.azure.com/

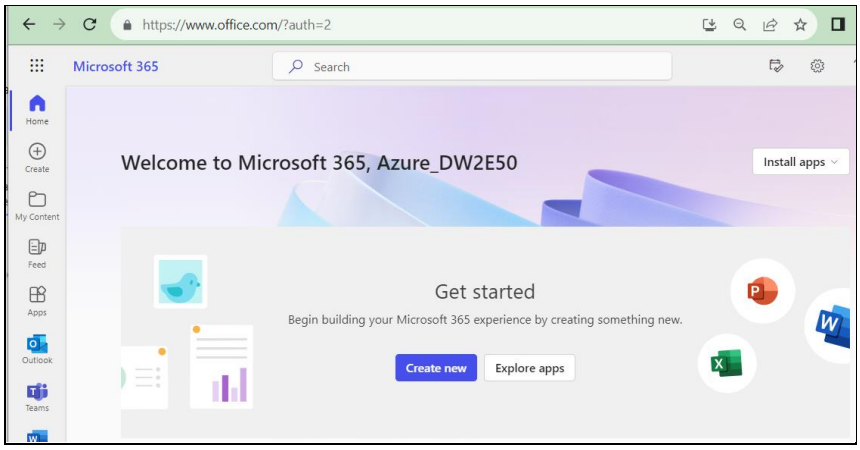


Tu cuenta te la proporcionará el profesor y es del tipo Office 365 A1 para estudiantes y se llamará: Azure_*grupo_num*@iesclaradelrey.es (por ejemplo, *Azure_DW2E50@esclaradelrey.es*).

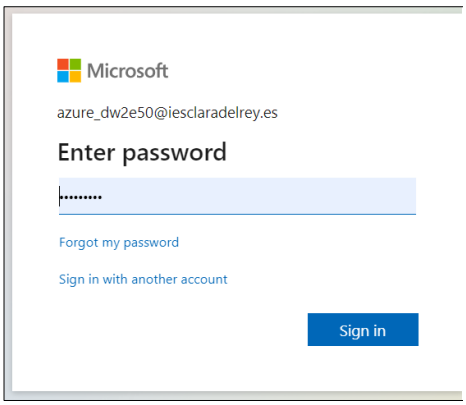
En el primer inicio de sesión, se pedirá el establecimiento de una nueva. Después de cambiar la contraseña, se puede solicitar un correo electrónico adicional. Este correo electrónico se utilizará para recuperar la contraseña. El paso no es obligatorio y lo omitiremos para seguir las pautas de protección de datos. (En el caso de que sea necesario restablecerlo, se abrirá una incidencia al TIC para su restablecimiento).

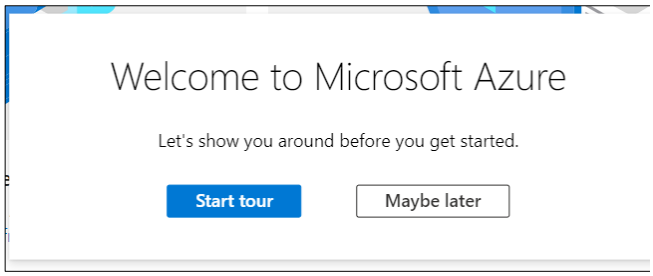




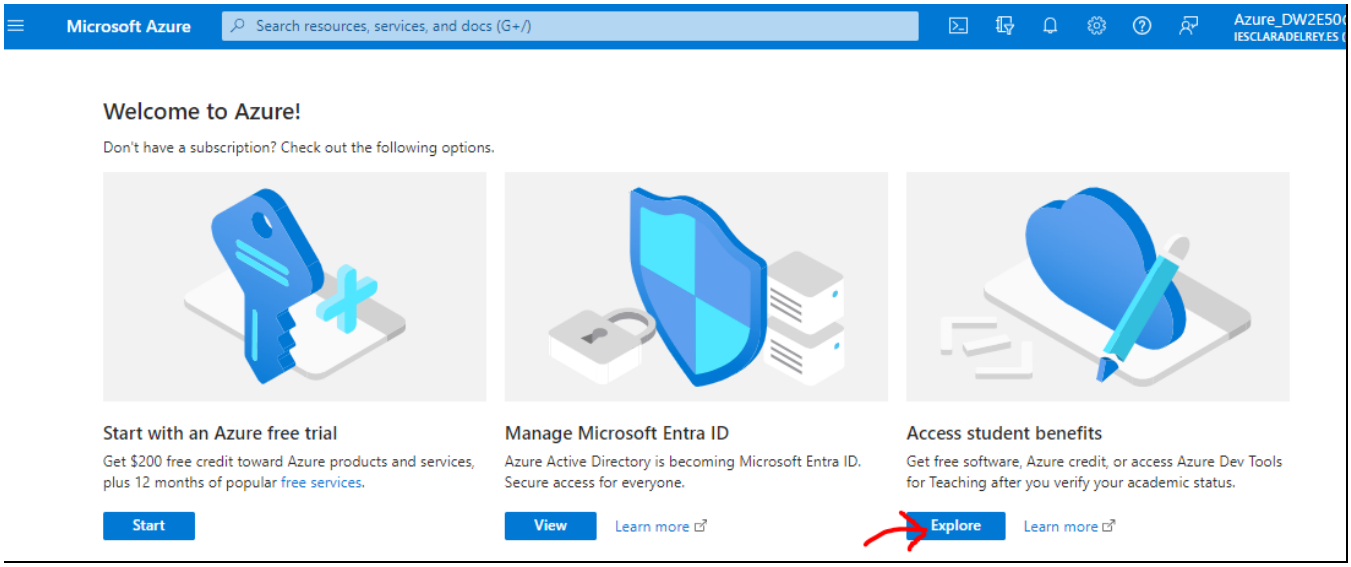


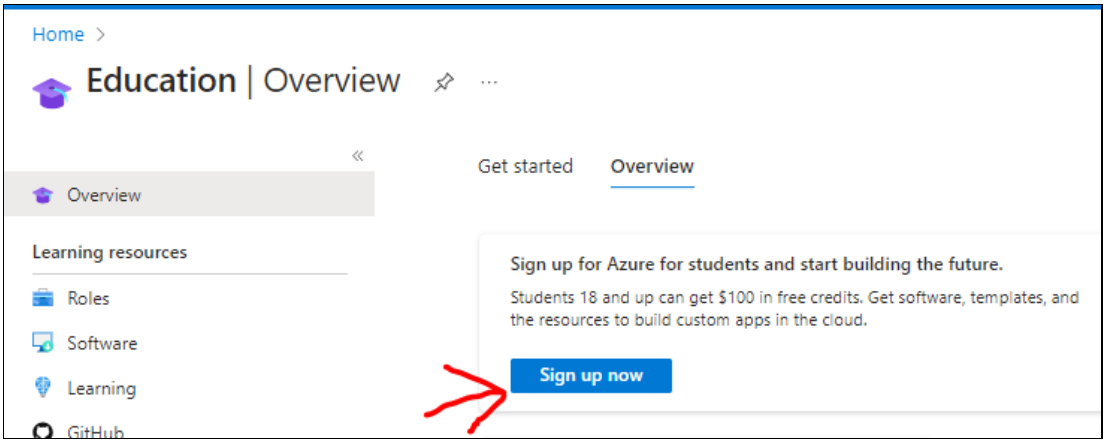
Una vez activada la cuenta de Microsoft 365 y cambiada la contraseña, inicia sesión en el Portal de Azure: https://portal.azure.com/, introduce la contraseña y accederás al servicio. La primera vez que accedamos al portal de Azure aparecerá un mensaje de bienvenida:



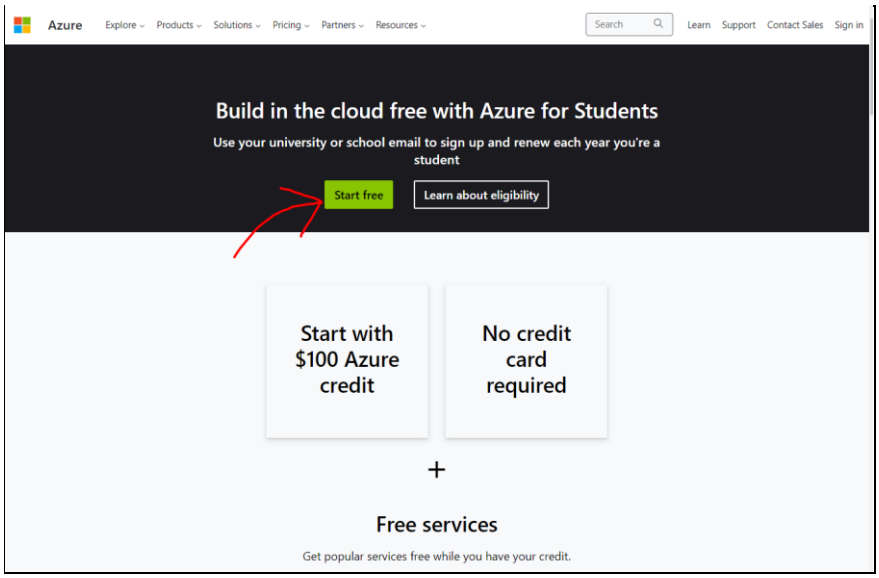


A continuación, hay que activar la suscripción de estudiante Azure Student. En la página de Azure, selecciona la opción "Acceder a los beneficios para estudiantes" en la parte superior derecha:





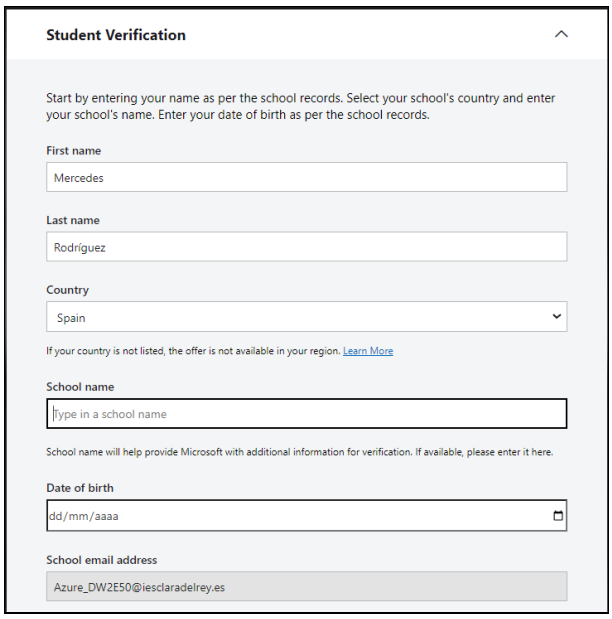
En la página siguiente hay un enlace "Comience gratis"; haz clic en él:



Por defecto, este programa académico de Azure proporciona un crédito de 100$ (90 € aprox.) para gastar en cualquier servicio de Azure y no requiere el uso de ningún tipo de tarjeta de crédito. Puede que, en principio, no sean suficientes para un uso generalizado de la plataforma, por eso debemos intentar mantener los gastos a un ritmo bajo. Cuando se agote el crédito de suscripción, los servicios ya no se podrán utilizar.

Volvemos a hacer clic en “Empezar gratis” para continuar con el proceso de validación de la cuenta educativa, que implicará varios pasos. En el panel "Verificación de estudiantes" evita utilizar datos personales. Por ejemplo: "Nombre" = DW2E*num*;

"Apellido" = iesclaradelrey; como "Dirección" puedes utilizar la dirección del instituto; etc.

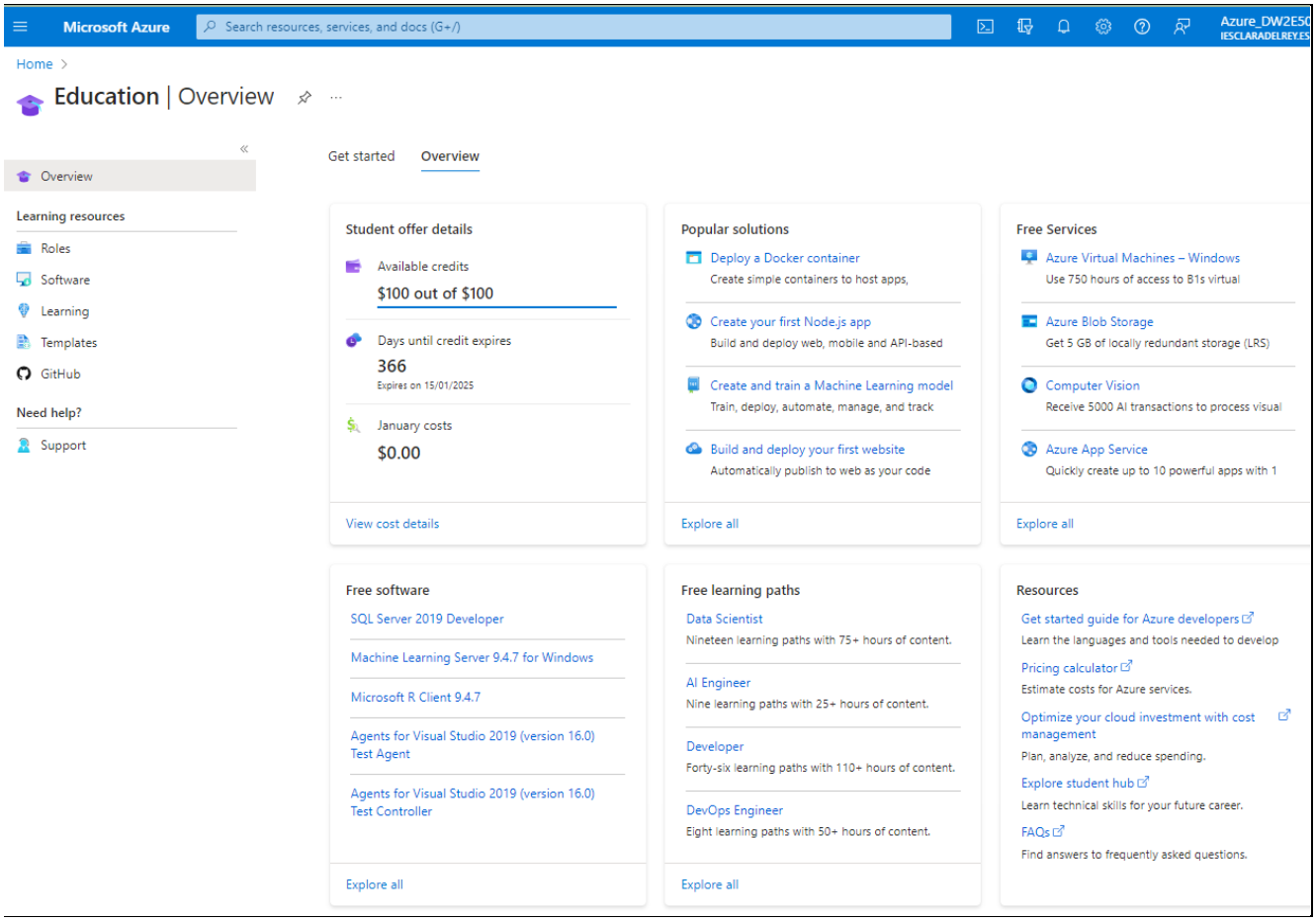


Observación: El proceso de activación de la suscripción no es del todo consistente y en ocasiones solicita al usuario un número de teléfono para realizar una activación por SMS para evitar bots. En versiones recientes han sustituido este mecanismo por un sistema captcha, pero en ocasiones sigue solicitando el número de teléfono para enviar un SMS. En caso de que suceda, tenemos que dar un número de teléfono. Microsoft utiliza este número para verificar nuestra identidad con un SMS o una llamada.

Después del proceso de validación veremos la "Página de descripción general" de nuestra suscripción con el crédito disponible:

https://portal.azure.com/#view/Microsoft_Azure_Education/EducationMenuBlade/~/overview

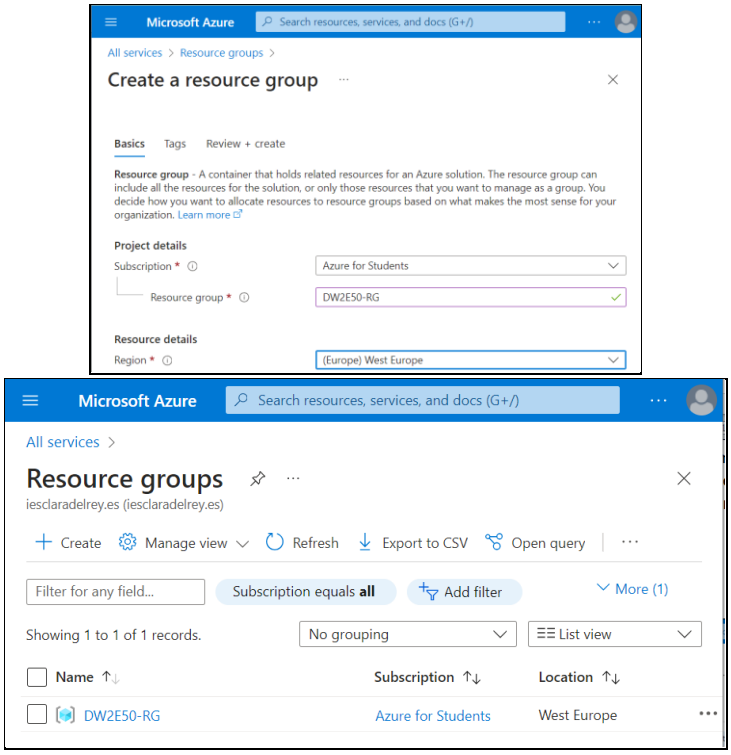
Cada vez que queramos consultar el crédito disponible, tenemos que acceder a la sección “Educación” de Azure. Si hemos accedido recientemente a este apartado, aparecerá en los servicios de acceso rápido en la parte superior.



CREA UN GRUPO DE RECURSOS (RESOURCE GROUP)

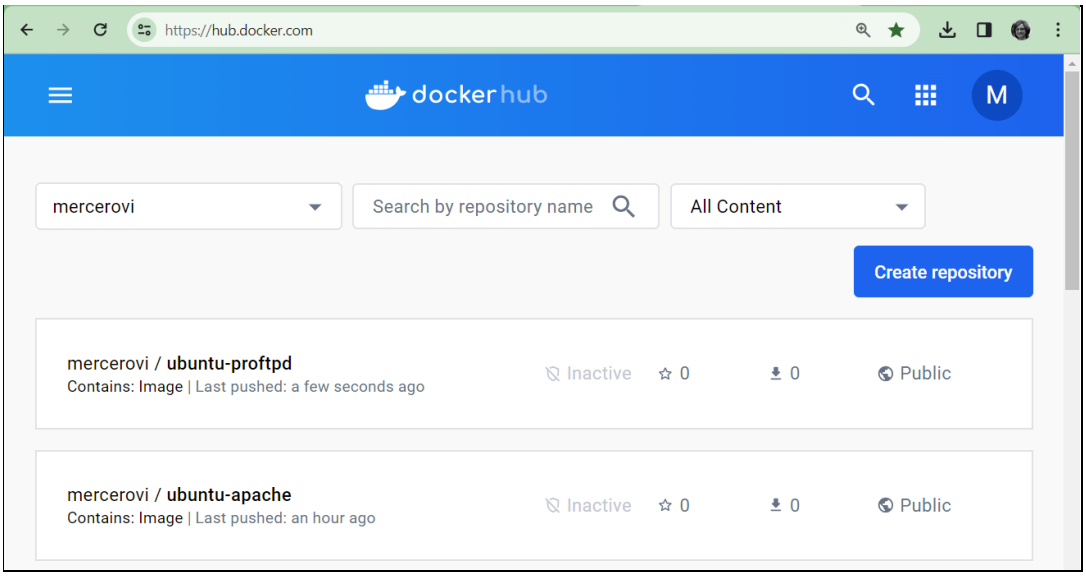
Las instancias de contenedor de Azure, al igual que otros recursos de Azure, se deben implementar en un grupo de recursos. Los grupos de recursos permiten organizar y administrar recursos relacionados.

Crea un nuevo grupo de recursos de nombre DW2E*num*-RG. Selecciona en Región Centro de Datos donde estará ubicado (West Europa):



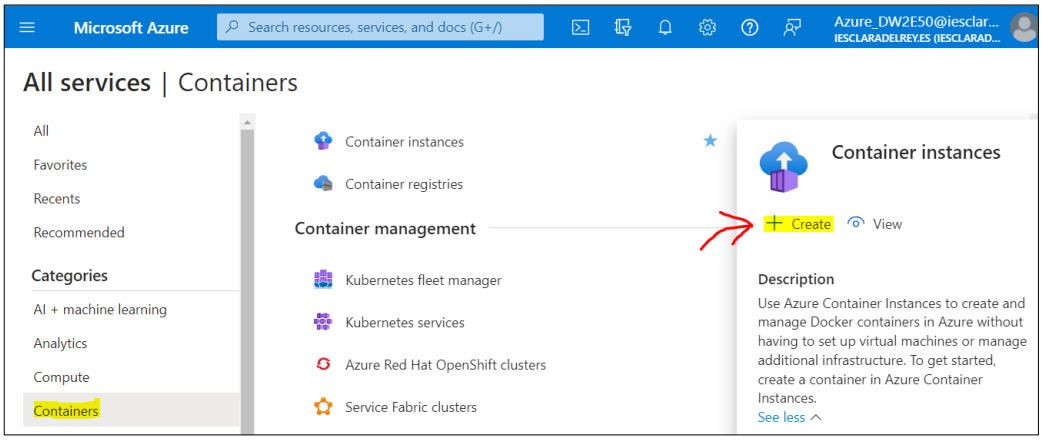
CREA UNA INSTANCIA DE CONTENEDOR EN AZURE (ACI)

Por ejemplo, vas a subir a la nube de Azure una de las imágenes que hayas creado y subido a Docker Hub. En este caso, haremos el ejemplo suponiendo que se tienen las siguientes imágenes en el espacio de nombres “mercerovi”:



Cada contenedor de Azure Container Instances (ACI) se implementa en un grupo de contenedores, y los grupos de contenedores pueden agrupar varias ACI. De momento, solo se admite con contenedores de Linux. Las ACI de un grupo de contenedores se programan en la misma máquina host y comparten un ciclo de vida, recursos, red local y volúmenes de almacenamiento.

Busca dentro del menú "Todos los servicios" la categoría "Contenedores":



Empezamos por la imagen del servicio web. Rellena los detalles:

1º) Parámetros básicos

- Grupo de recursos: **DW2E*num*-RG**
- Nombre de contenedor: **web-aci**
- Región: **(Europe) West Europe** (normalmente, en la misma región que el grupo de recursos).
- Availability Zones: **None**
- SKU: **Standard**
- Fuente de la imagen del contenedor: como tienes subida la imagen en Docker Hub, elige **Other Registry**.
- Descuento Azure Spot: **No**.
- Tipo de imágenes: si usas credenciales, elige **Private**, si no, **Public**. Con Private, tendrás que introducir las contraseñas de Docker Hub en **hub.docker.com**.
- Imagen: ***nombre-usuario/imagen:tag*** (por ejemplo, *mercerovi/ubuntu-apache:1.0*)
- Tipo OS: **Linux**

2º) Parámetros de red

- Tipo de red: **Public**
- Nombre DNS: ***dominio-web01*** (por ejemplo, *ejemplo-web01*)**: d**ebes especificar un nombre DNS para tu contenedor, es necesario para poder obtener una IP pública. El nombre debe ser único dentro de la región de Azure donde se crea la instancia de contenedor. Tu contenedor también será accesible públicamente en *<dominio-web01>.<hash>.<región>.azurecontainer.io*.
- Alcance y reutilización del nombre DNS: **Tenant** o inquilino (representa a la empresa u organización).

  Distintos tipos:

Nombre de la política de reutilización	Descripción
Tenant/inquilino	El hash se generará en función del nombre DNS y el ID del inquilino. La etiqueta del nombre de dominio del objeto se puede reutilizar dentro del mismo inquilino.
Subscription	El hash se generará en función del nombre DNS, el ID del inquilino y el ID de la suscripción. La etiqueta del nombre de dominio del objeto se puede reutilizar dentro de la misma suscripción.
Resource Group	El hash se generará en función del nombre DNS y el ID del inquilino, el ID de la suscripción y el nombre del grupo de recursos. La etiqueta del nombre de dominio del objeto se puede reutilizar dentro del mismo grupo de recursos.
No Reuse	El hash se generará utilizando un GUID. La etiqueta de dominio del objeto se puede reutilizar libremente, ya que este hash siempre será único.
Any reuse (unsecure)	No se generará hash. La etiqueta de dominio del objeto no se puede reutilizar dentro del grupo de recursos, suscripción o inquilino.

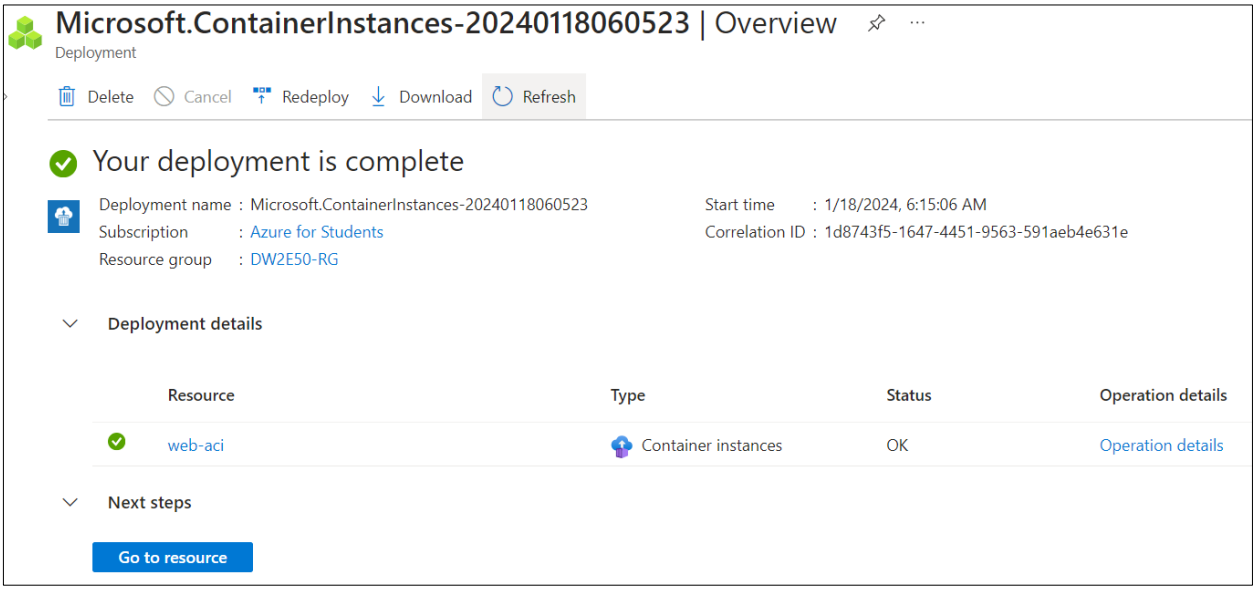
- Puertos: 80/TCP, 8080/TCP y 443/TCP (se requiere al menos un puerto cuando se incluye una dirección IP pública).

3º) Parámetros advanzados

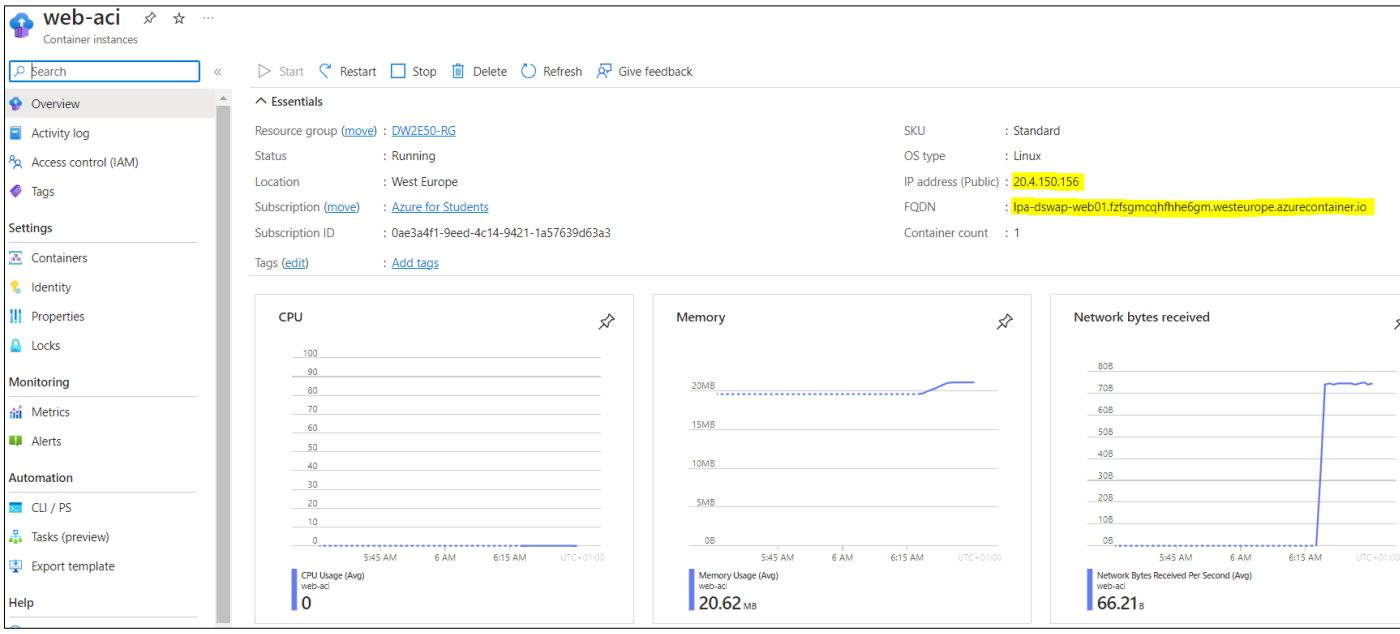
Deja los valores que vienen por defecto.

4º) Revisar+Crear

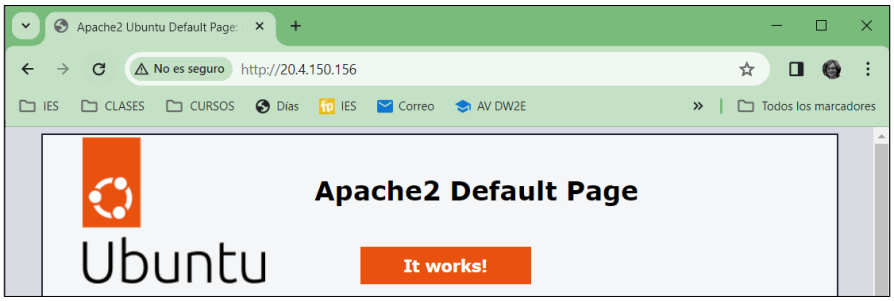
Después de validar el contenedor con éxito, crea el contenedor.

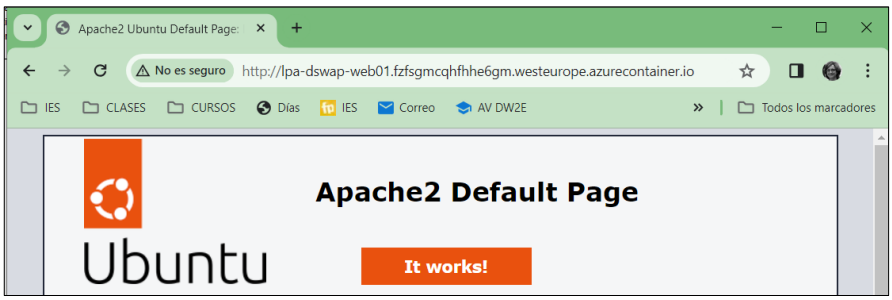


Puedes inspeccionar el uso del contenedor y su IP pública (botón "Ir al recurso"):



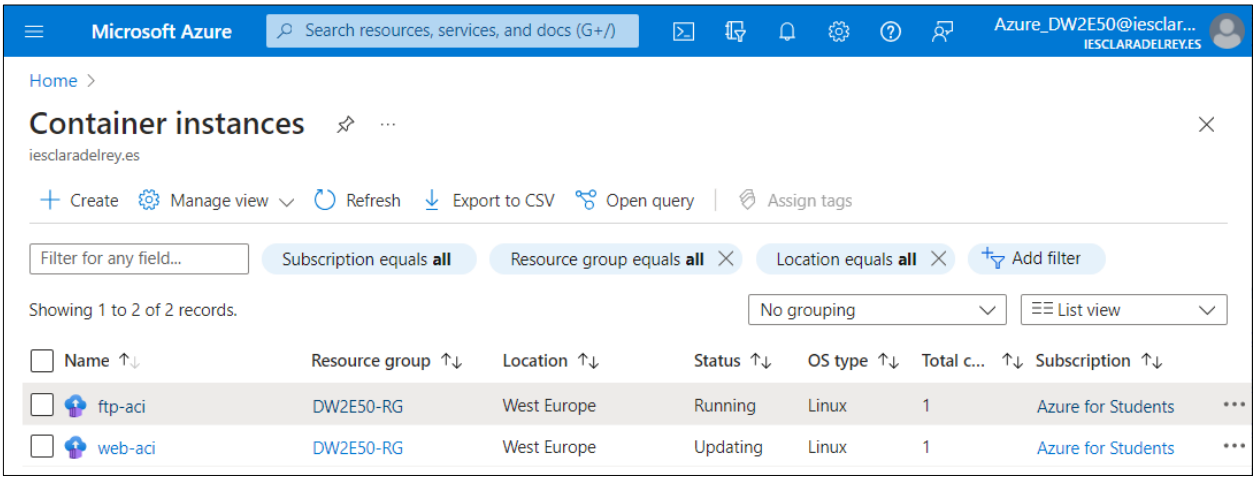
Intenta conectarse a la IP pública de tu contenedor o mediante la FQDN del servidor web (*<dominio-web01>.<hash>.<región>.azurecontainer.io*):



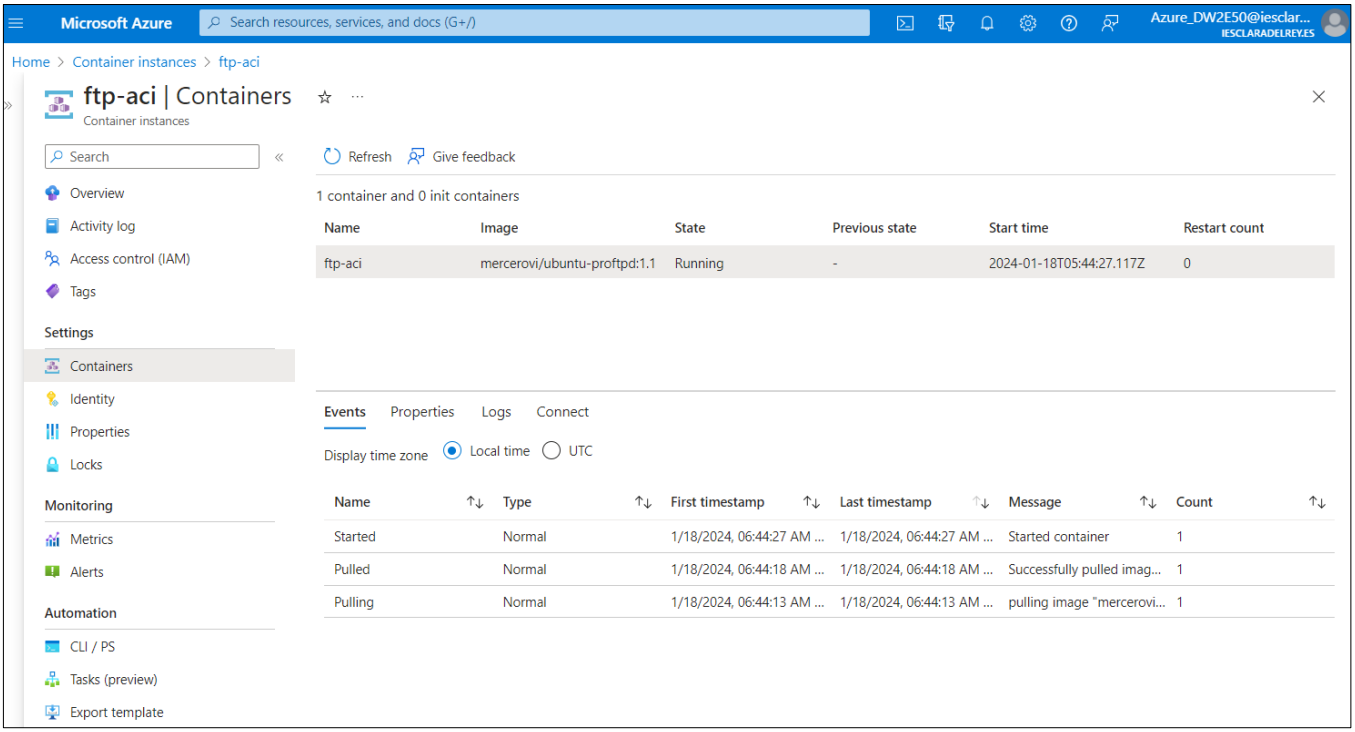


Adicionalmente, prueba a subir otra imagen y crea otra ACI, por ejemplo una imagen que tenga un servidor FTP.

Finalmente, comprueba en listado de tus ACI en Azure:



E inspeccionálas. Por ejemplo:



Cómo conectarse a un ACI usando la CLI de Azure

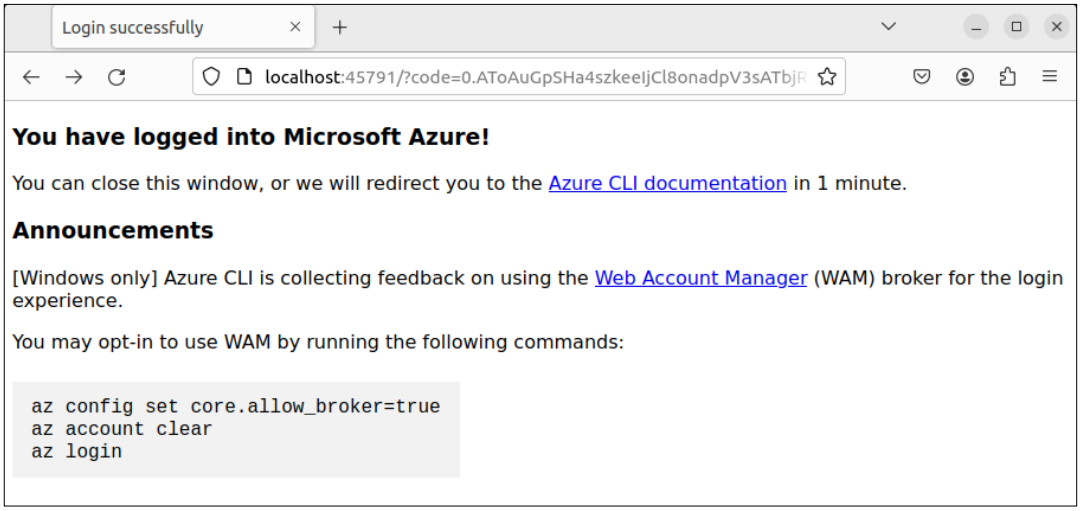
Utiliza como anfitrión un Ubuntu. Debes instalar en primer lugar la CLI de Azure en Ubuntu (https://learn.microsoft.com/en-us/cli/azure/install-azure-cli-linux?pivots=apt). Escribe en un terminal:

```
$ curl -sL https://aka.ms/InstallAzureCLIDeb | sudo bash
```

Una vez instalado, inicia sesión en la CLI de Azure mediante el comando az login. Para finalizar el proceso de autenticación, sigue los pasos indicados (es posible que se abra un navegador).

```
$ az login
```

```
A web browser has been opened at
https://login.microsoftonline.com/organizations/oauth2/v2.0/authorize.

Please continue the login in the web browser. If no web browser is available or if the web
browser fails to open, use device code flow with `az login --use-device-code`.
[
  {
    "cloudName": "AzureCloud",

    "homeTenantId": "1d526ab8-2cae-47ce-9e22-30a5f289da76",
    "id": "0ae3a4f1-9eed-4c14-9421-1a57639d63a3",
    "isDefault": true,
    "managedByTenants": [],
    "name": "Azure for Students",
    "state": "Enabled",
    "tenantId": "1d526ab8-2cae-47ce-9e22-30a5f289da76",
    "user": {
      "name": "Azure_DW2E50@iesclaradelrey.es",
      "type": "user"
    }
  }
]
```



Conéctate por ejemplo al terminal bash de la instancia del contenedor FTP e inspecciona los archivos de configuración del servidor FTP (cambia el nombre del grupo de recursos por el tuyo y el del contenedor también por el tuyo):

```
$ az container exec --resource-group DW2E50-RG --name ftp-aci --exec-command "/bin/bash"
root@SandboxHost-638411534464283831:/#
root@SandboxHost-638411534464283831:/# ll
```

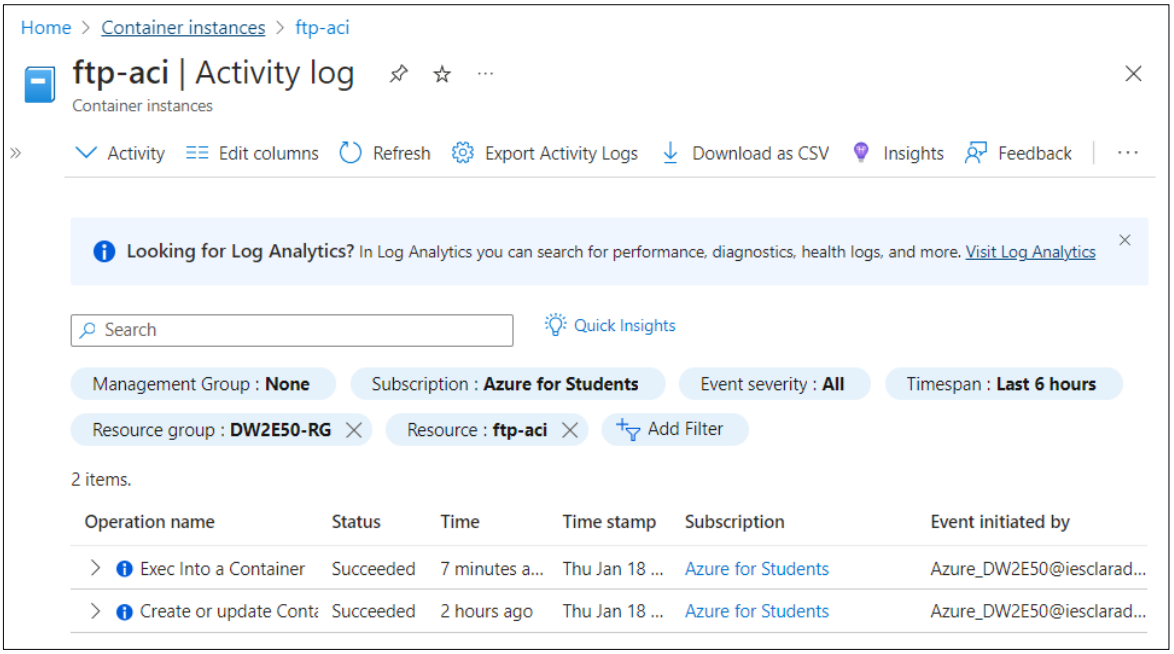
```
total 72
drwxr-xr-x   1 root root 4096 Jan 18 05:44 ./
drwxr-xr-x   1 root root 4096 Jan 18 05:44 ../
-rw-r--r--   1 root root 1089 Jan 18 04:52 Dockerfile
lrwxrwxrwx   1 root root    7 Jan 11 14:03 bin -> usr/bin/
drwxr-xr-x   2 root root 4096 Apr 18  2022 boot/
drwxr-xr-x   5 root root  340 Jan 18 05:44 dev/
drwxr-xr-x   1 root root 4096 Jan 18 04:53 etc/
drwxr-xr-x   1 root root 4096 Jan 18 04:53 home/
drwx------   1 root root 4096 Jan  1  1970 lost+found/
drwxr-xr-x   2 root root 4096 Jan 11 14:03 media/
drwxr-xr-x   2 root root 4096 Jan 11 14:03 mnt/
drwxr-xr-x   2 root root 4096 Jan 11 14:03 opt/
```

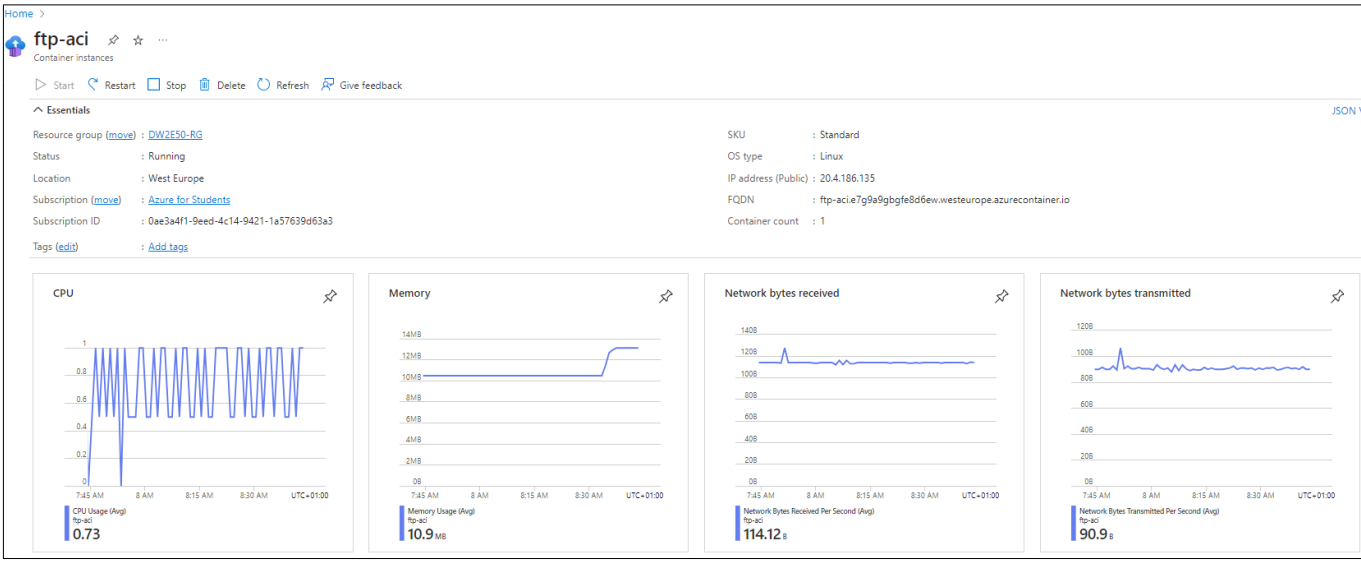
```
dr-xr-xr-x 124 root root    0 Jan 18 05:44 proc/
drwx------   2 root root 4096 Jan 11 14:06 root/
drwxr-xr-x   1 root root 4096 Jan 18 05:44 run/
lrwxrwxrwx   1 root root    8 Jan 11 14:03 sbin -> usr/sbin/
drwxr-xr-x   1 root root 4096 Jan 18 04:53 srv/
dr-xr-xr-x  11 root root    0 Jan 18 05:44 sys/
drwxrwxrwt   1 root root 4096 Jan 18 04:53 tmp/
drwxr-xr-x   1 root root 4096 Jan 11 14:03 usr/
drwxr-xr-x   1 root root 4096 Jan 11 14:06 var/
```

```
root@SandboxHost-638411534464283831:/# ll /etc/proftpd
```

```
total 1348
drwxr-xr-x 3 root root    4096 Jan 18 04:53 ./
drwxr-xr-x 1 root root    4096 Jan 18 04:53 ../
-rw-r--r-- 1 root root 1310700 Dec  3  2021 blacklist.dat
drwxr-xr-x 2 root root    4096 Dec  3  2021 conf.d/
-rw-r--r-- 1 root root    9420 Dec  3  2021 dhparams.pem
-rw-r--r-- 1 root root    4353 Jan 18 04:53 geoip.conf
-rw------- 1 root root     701 Jan 18 04:53 ldap.conf
-rw-r--r-- 1 root root    3454 Jan 18 04:53 modules.conf
-rw-r--r-- 1 root root    5819 Jan 18 04:53 proftpd.conf
-rw-r--r-- 1 root root    1186 Jan 18 04:53 sftp.conf
-rw-r--r-- 1 root root     982 Jan 18 04:53 snmp.conf
-rw------- 1 root root     862 Jan 18 04:53 sql.conf
-rw-r--r-- 1 root root    2082 Jan 18 04:53 tls.conf
-rw-r--r-- 1 root root     832 Jan 18 04:53 virtuals.conf
```

Verifica la actividad en el Portal Azure:

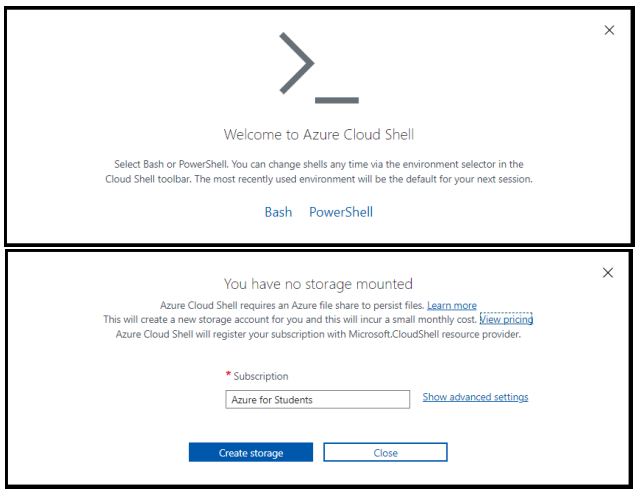


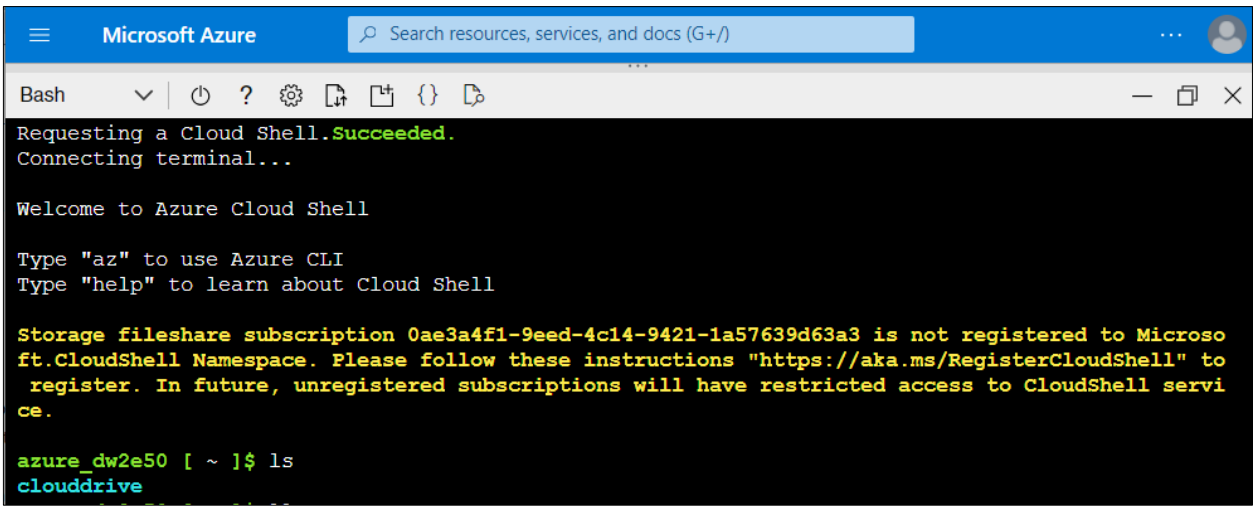


Repite los pasos con el contenedor ACI del servicio web.

Cómo conectarse a un ACI usando "Azure Cloud Shell"

Abra un navegador y conéctese a: https://portal.azure.com/#cloudshell/





Referencias

Azure tutorials for containers:

https://learn.microsoft.com/en-us/azure/container-instances/container-instances-overview

https://learn.microsoft.com/es-es/azure/container-instances/container-instances-quickstart-portal

https://learn.microsoft.com/en-us/azure/container-instances/container-instances-quickstart

Course "Run container images in Azure Container Instances";

https://learn.microsoft.com/en-gb/training/modules/create-run-container-images-azure-container-instances/

Execute a command in a running Azure container instance:

https://learn.microsoft.com/en-us/azure/container-instances/container-instances-exec

Example "Expose a static IP address for a container group":

https://learn.microsoft.com/en-us/azure/container-instances/container-instances-application-gateway

To learn more:

Azure CLI:

https://learn.microsoft.com/en-us/cli/azure/get-started-with-azure-cli

Azure Cloud Shell

https://learn.microsoft.com/en-us/azure/cloud-shell/get-started?tabs=azurecli

Use case example, Infraestructura LAMP con Docker Compose:

https://openwebinars.net/blog/infraestructura-lamp-con-docker-compose/

Conclusiones

Explica con tus palabras lo que has aprendido con esta práctica.

Configuración de Oracle Database XE en un contenedor Docker

Desarrollada por Carlos Moreno Martínez

1. Objetivos

Una de las alternativas mas versátiles de la actualidad para utilizar Oracle DataBase en entornos de desarrollo y aprendizaje es la creación de un **contenedor Docker con el software de Oracle DataBase XE**. Las tecnologías de contenedores permiten desplegar en poco tiempo un servicio con el SGBD de forma cómoda y sencilla.

El uso de Docker frente a máquinas virtuales tiene la ventaja de que Docker es una plataforma que empaqueta el software en unidades estandarizadas denominadas **contenedores** que contienen justo lo necesario para ejecutar el servicio, pudiendose escalar de forma rápida. Sin embargo, una máquina virtual es una copia digital de una máquina física con su sistema operativo; los contenedores Docker usan el sistema operativo de la máquina.

2. Pasos a seguir.

Partimos de la base de que ya se ha instalado en la máquina el software necesario para ejecutar contenedores Docker.

1. **Descarga de la imagen.**

   En primer debemos bajar a la máquina la imagen de Oracle DataBase XE. Oracle dispone de una serie de contenedores que se puede consultar en la url: **https://container-registry.oracle.com**.

   El contenedor de Oracle DataBase XE es el denominado express. Para descargarlo ejecuta el comando:

   docker pull container-registry.oracle.com/database/express:lastest

Para mayor comodidad, añade un alias de esta imagen. Para ello, ejecuta el comando (todo en la misma linea de comando):

docker image tag container-registry.oracle.com/database/express:lastest oracle-xe

Inserta aquí el resultado del comando docker images después de su descarga. Para ello, usa el comando **docker images**:

```
# docker images
```

2. **Creación del contenedor.**

   A partir de la imagen descargada creamos la imagen del contenedor que se va a ejecutar. El comando para realizar esta operación es (todo en la misma linea de comando):

   docker container create -it --name oracleBBDD -p 1521:1521 -e ORACLE_PWD=admin1234 container-registry.oracle.com/database/express:lastest

   Con este comando se ha creado un contenedor con el alias **oracleBBDD** (parámetro --name) y con contraseña para SYS y SYSTEM **admin1234**.(variable de entorno –e ORACLE_PWD).

   Muestra el resultado de la ejecución. El resultado tiene que ser una cadena hexadecimal del tipo que aparece en el recuadro (¡pero no el mismo!):

```
# docker container create -it --name oracleBBDD -p 1521:1521 -e ORACLE_PWD=admin1234
container-registry.oracle.com/database/express:lastest

88bc1399576cc3dd93dcd80616939c2d7587463464ff8623733e28eaa5bb173b
```

   Como puedes comprobar se creado un contenedor que tiene como ID una cadena de bytes, con un alias oracleBBDD.

   Pon en funcionamiento el contenedor. Para ello, usa el comando:

   docker start oracleBBDD

```
# docker start oracleBBDD

oracleBBDD
```

   Una vez que no deseemos seguir usando Oracle es necesario parar el contenedor. El comando es:

   docker stop [nombre]

   La salida es igual que en el arranque del contenedor.

2. **Conexión al CLI del contenedor y a SQLPlus de Oracle.**

   Una vez creado el contenedor vas a ponerlo en funcionamiento y a trabajar con el. En primer lugar, accede al CLI del contenedor mediante el comando:

   docker exec -it oracleBBDD bash

   Muestra el resultado de la ejecución

```
# docker exec -it oracleBBDD bash

bash-4.2$
```

Comprueba el directorio donde estás y con que usuario estas conectado.

```
bash-4.2$ pwd
/home/oracle
bash-4.2$ whoami
oracle
```

4. **El schema Human Resources (HR). Descarga e importación de los ficheros al contenedor.**

   En primer lugar debes descargar la última versión de los schema de ejemplo. Para ello, Oracle dispone de un repositorio e GitHub donde hay ejemplos utilizables en sus aplicaciones. La dirección del repositorio es:

   https://github.com/oracle-samples/db-sample-schemas

   Una vez descargados los ejemplos (ya sea la totalidad de ellos o tan solo el correspondiente a Human Resources) descomprimimos la carpeta la copiamos al repositorio.

   Para ello, colocate en la carpeta donde se descomprimiste los esquemas y ejecuta el comando:

   docker cp human_resources oracleBBDD:/home/oracle

   Accede al contenedor oracleBBDD. Puedes comprobar que hay un directorio llamado **human_resources**. Dentro de este directorio hay un fichero llamado **hr_install.sql**. Muestra el resultado:

   ```
   # docker cp human_resources oracleBBDD:/home/oracle
   # cd human_resources
   ```

```
Successfully copied 86kB to oracleBBDD:/home/oracle
bash-4.2$ ls
human_resources scott.sql
bash-4.2$ cd human_resources/
bash-4.2$ ls
README.md README.txt hr_code.sql hr_create.sql hr_install.sql hr_populate.sql
hr_uninstall.sql
```

5. **El esquema Human Resources (HR). Instalación en la Base de Datos.**

   Desde el directorio **human_resources** del contenedor accede al sqlplus con el usuario SYSTEM (contraseña: admin1234). Muestra el proceso:

```
# cd human_resources
# sqlplus
```

```
bash-4.2$ cd human_resources/
bash-4.2$ ls
README.md README.txt hr_code.sql hr_create.sql hr_install.sql hr_populate.sql
hr_uninstall.sql
bash-4.2$ sqlplus


SQL*Plus: Release 21.0.0.0.0 - Production on Thu Feb 8 10:50:16 2024
Version 21.3.0.0.0


Enter user-name: SYSTEM
Enter password:
Last Successful login time: Thu Feb 08 2024 10:26:22 +00:00


Connected to:
Oracle Database 21c Express Edition Release 21.0.0.0.0 - Production
Version 21.3.0.0.0
```

Oracle trabaja con las llamadas **PDB (Pluggable Database)**, introducida en la versión 12c. Una PDB es una base de datos autónoma que puede ser conectada y desconnectada de una base de datos de contenedor (CDB, por sus siglas en inglés) de manera segura y eficiente.

Por defecto, Oracle XE tiene creada una PDB llamada XEPDB1 a la cual hay que conectarse. Para ello, ejecuta el comando:

SQL> ALTER SESSION SET CONTAINER=XEPDB1;

Muestra el resultado:

```
SQL> ALTER SESSION SET CONTAINER=XEPDB1;
Session altered.
```

Una vez que te has conectado a XEPDB1 ejecuta el script de instalación del schema HR. Ejecuta debes ejecutar el comando:

SQL> @hr_install.sql;

Te solicitará una contraseña para el schema. Aquí puedes poner la que quieras; lo mas facil es poner HR (en mayúsculas) como el schema. Por otro lado, solicita el Tablespace en el que se alojara este schema; deja como Tablespace el que viene por defecto (USERS). A partir de ahí, se realizan las operaciones de creado de objetos y poblamiento de tablas.

Muestra el resultado de las últimas lineas:

```
SQL> @hr_install.sql;
```

```
Thank you for installing the Oracle Human Resources Sample Schema.
This installation script will automatically exit your database session
at the end of the installation or if any error is encountered.
The entire installation will be logged into the 'hr_install.log' log file.

SP2-0606: Cannot create SPOOL file "hr_install.log"
Enter a password for the user HR:



Enter a tablespace for HR [USERS]:
Do you want to overwrite the schema, if it already exists? [YES|no]:

... Proceso de creación objetos y poblamiento de tablas ...

Installation
-------------
Verification:

Table provided actual
----------- ---------- ----------
regions 5 5
countries 25 25
departments 27 27
locations 23 23
employees 107 107
jobs 19 19
job_history 10 10

Thank you!
-------------------------------------------------------
The installation of the sample schema is now finished.
Please check the installation verification output above.

You will now be disconnected from the database.

Thank you for using Oracle Database!

not spooling currently
Disconnected from Oracle Database 21c Express Edition Release 21.0.0.0.0 - Production
Version 21.3.0.0.0
```

Una vez realizado el proceso, ya está disponible el perfil **HR@XEPDB1** para su uso.

5. El esquema Human Resources (HR). Conexión desde SQL Developer.

   Dentro de SQL Developer configura una conexión nueva con los siguientes datos:

   Usuario y contraseñas. Las del perfil HR que has configurado. **¡Recuerda la diferencia entre mayúsculas y minúsculas!**.

   Nombre del servicio. A diferencia con las versiones antiguas (11g), las versiones modernas usan las tecnologías de CBD y PBD. Configuraremos como Nombre del Servicio: **XEPDB1**.

   Pulsa el botón "Probar" y si el resultado es "Correcto" guarda la conexión. Prueba que puedes conectarte, haz una consulta a la tabla COUNTRIES y muestra su contenido:

```
SQL> SELECT * FROM COUNTRIES;
```

```
COUNTRY_ID COUNTRY_NAME REGION_ID
AR Argentina 20
AU Australia 40
BE Belgium 10
BR Brazil 20
. . . .
```

7. Parada del contenedor.

   Si no deseamos seguir usando Oracle es necesario parar el contenedor. El comando es:

   docker stop [nombre]

   La salida es igual que en el arranque del contenedor. Una vez que hayas acabado la práctica acuerdate de parar el contendor.

```
# docker stop oracleBBDD
```

```
oracleBBDD
```

3. Conclusiones

Explica con tus palabras lo que has aprendido con esta práctica.

Instalación de WSL

Desarrollada por **Camino Pardo de Vega**

1.- OBJETIVOS

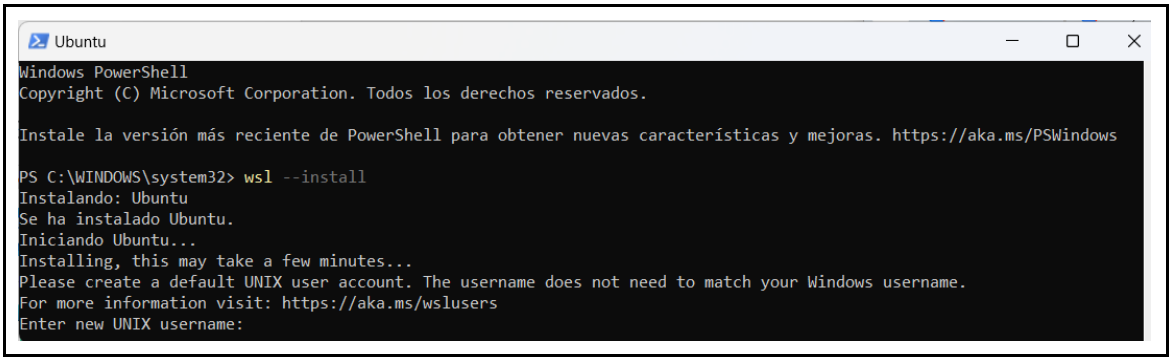
En Windows existe una característica llamada **Subsistema de Windows para Linux** (WSL) que da la posibilidad de ejecutar un entorno Linux en una máquina Windows, sin necesidad de usar una máquina virtual o tener que tener instalados los dos sistemas operativos y un arranque dual.

El objetivo de esta práctica es aprender a instalar la característica WSL y un entorno de ubuntu.

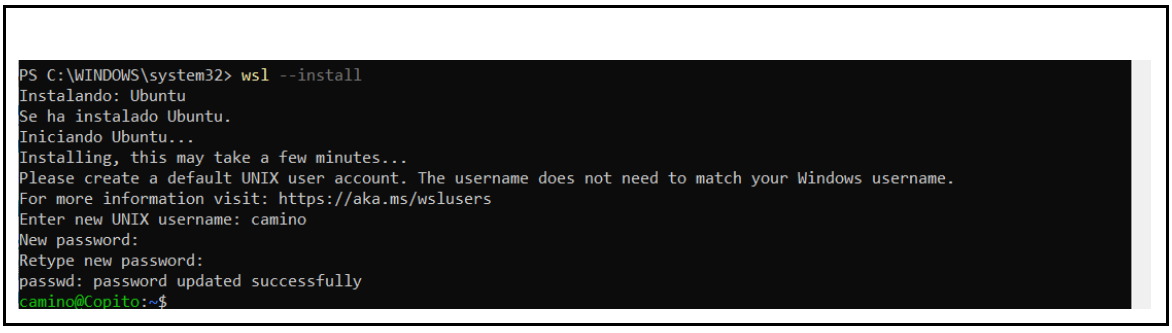
https://learn.microsoft.com/es-es/windows/wsl/install

2.- PASOS A SEGUIR

1. Abre PowerShell de Windows como administrador e instala wsl
   Inserta aquí la imagen donde se vea que se ha realizado la instalación y el comando utilizado

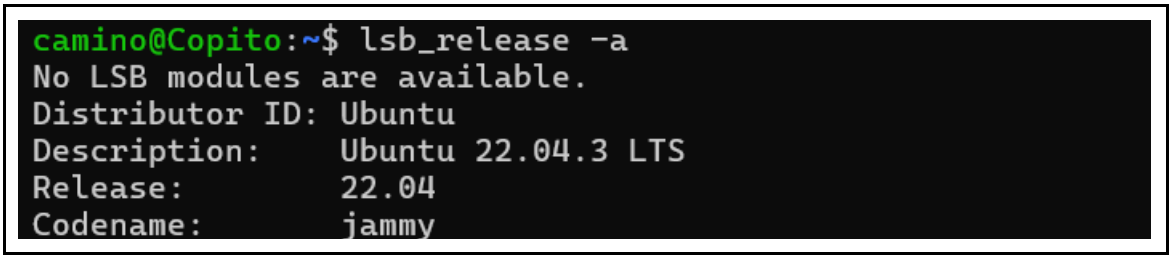


2. Una vez se haya instalado escribe tu nombre como username y asigna una contraseña
   Inserta aquí la imagen donde se vea que hemos accedido a Ubuntu



3. Cierra la ventana de PowerShell. Busca en la barra de tareas wsl y pulsa sobre la aplicación
   Inserta aquí la imagen con la sesión de Ubuntu abierta

4. Mira qué versión hay instalada de Ubuntu.
   Inserta aquí la imagen donde se vea la versión de Ubuntu con la que estás trabajando.



5. Cierra la ventana de Ubuntu y abre una ventana PowerShell de Windows. Muestra la lista válida de nombres de distribuciones disponibles de linux disponibles para instalar
   Inserta aquí la imagen donde se vea el comando usado y el resultado de dicho comando

6. Instala Ubuntu 20.04 LTS y pon el nombre de usuario y contraseña que quieras.
   Inserta aquí la imagen donde se vea el comando usado y el resultado de dicho comando

```
wsl –install -d Ubuntu-20.04
```

```
PS C:\Users\Camino> wsl --install Ubuntu-20.04
Instalando: Ubuntu 20.04 LTS
Se ha instalado Ubuntu 20.04 LTS.
Iniciando Ubuntu 20.04 LTS...
Installing, this may take a few minutes...
Please create a default UNIX user account. The username does not need
 to match your Windows username.
For more information visit: https://aka.ms/wslusers
Enter new UNIX username:
```

7. Desde la máquina ubuntu que acabas de instalar comprueba que la versión de Ubuntu que has instalado es la que querías.
   Inserta aquí la imagen donde se vea el comando usado y el resultado de dicho comando

```
camino20@Copito:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 20.04.6 LTS
Release:        20.04
Codename:       focal
```

8. Sal del la distribución linux que acabas de instalar.
9. Muestra la lista válida de nombres de distribuciones de linux instaladas
   Inserta aquí la imagen donde se vea el comando usado y el resultado de dicho comando

wsl -l -verbose

```
PS C:\Users\Camino> wsl -l -v
  NAME            STATE           VERSION
* Ubuntu          Running         2
  Ubuntu-20.04    Stopped         2
```

3.- CONCLUSIONES

Explica con tus palabras lo que has aprendido con esta práctica.

Conectar a una máquina Linux desde el escritorio remoto de Windows

Desarrollada por **Camino Pardo de Vega**

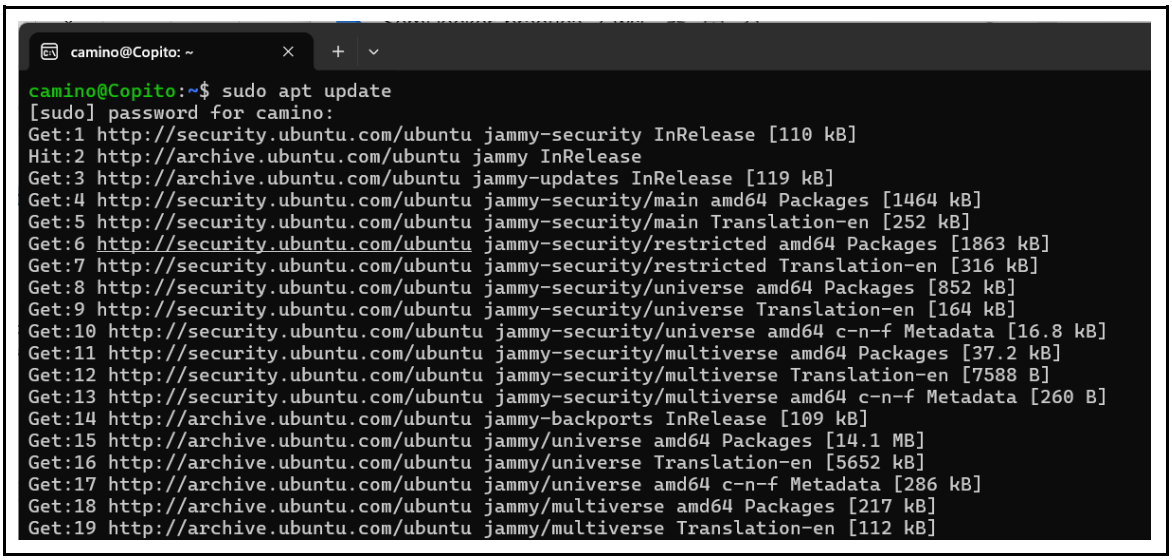
1.- OBJETIVOS

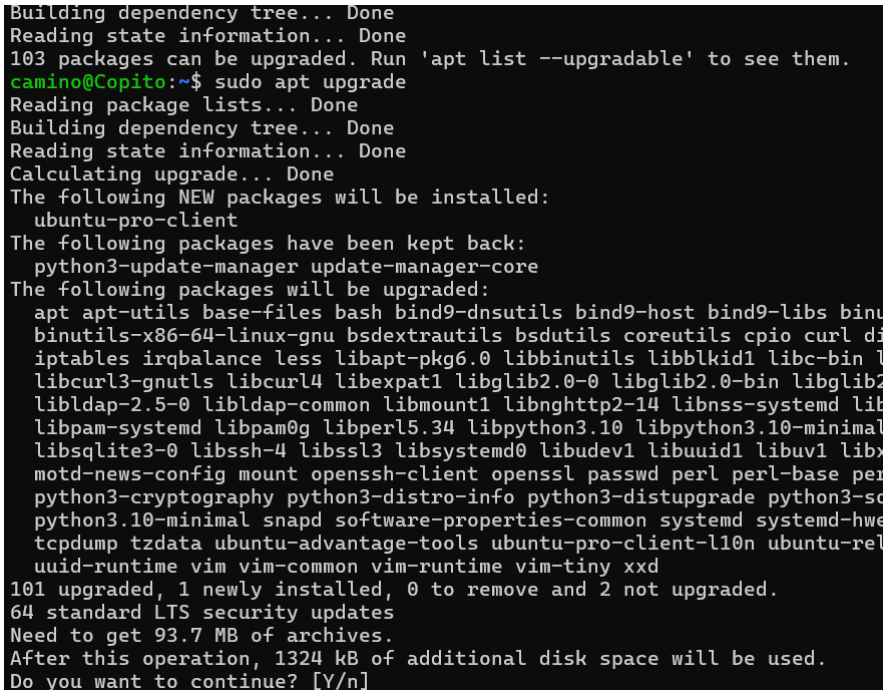
Conectar desde el escritorio remoto de Windows a una máquina Ubuntu en modo comando

Instalar un escritorio ligero en Ubuntu y trabajar en él con un escritorio remoto.

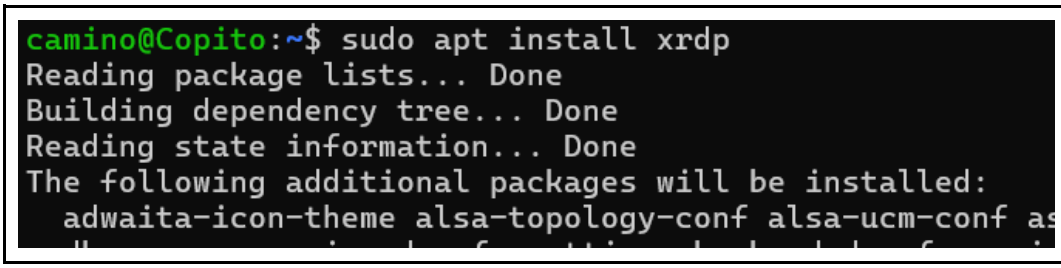
2.- PASOS A SEGUIR

1. Busca en la barra de tares Wsl y pincha sobre la aplicación
2. Actualiza el sistema operativo
   Inserta aquí las imágenes donde se vea que se ha actualizado el sistema operativo correctamente.

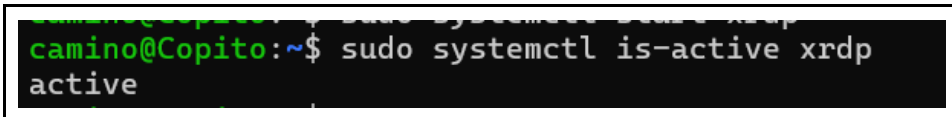




3. Instala el servicio de escritorio remoto xrdp que es un software de código abierto que le permite trabajar con un ordenador remoto usando una sesión de escritorio
   Inserta aquí la imagen donde se vea el comando usado y el resultado de dicho comando



4. Comprueba en ubuntu que el servicio de escritorio remoto está activado
   Inserta aquí la imagen donde se vea el comando y que el servicio está activado.



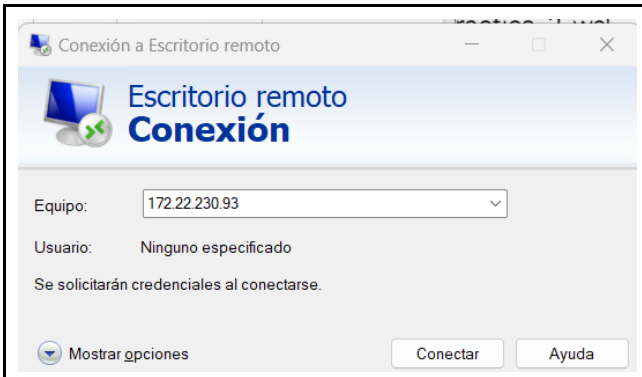
5. Instala el escritorio xfce4 que es un entorno de escritorio ligero para sistemas tipo UNIX. Su objetivo es ser rápido y usar pocos recursos del sistema, sin dejar de ser visualmente atractivo y fácil de usar.
   Inserta aquí la imagen donde se vea el comando usado y el resultado de dicho comando

```
camino@Copito:~$ sudo apt install -y xfce4
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  acl avahi-daemon colord colord-data desktop-base elementary-xfce-icon-t
  gir1.2-freedesktop gir1.2-gdkpixbuf-2.0 gir1.2-gtk-3.0 gir1.2-harfbuzz-
  gir1.2-xfconf-0 greybird-gtk-theme gtk2-engines-murrine ipp-usb libasou
  libcairomm-1.0-1v5 libcanberra-gtk3-0 libcanberra-gtk3-module libcolorh
  libdbusmenu-glib4 libdbusmenu-gtk3-4 libexif12 libexo-2-0 libexo-common
  libgarcon-common libgarcon-gtk3-1-0 libgd3 libglibmm-2.4-1v5 libgomp1 l
  libgtkmm-3.0-1v5 libgtop-2.0-11 libgtop2-common libgusb2 libieee1284-3
  libnotify-bin libnotify4 libnss-mdns libopenjp2-7 libpangomm-1.4-1v5 li
  libpoppler118 libpulse-mainloop-glib0 libpulsedsp libsane-common libsan
  libsnmp-base libsnmp40 libsoxr0 libspeexdsp1 libstartup-notification0 l
```

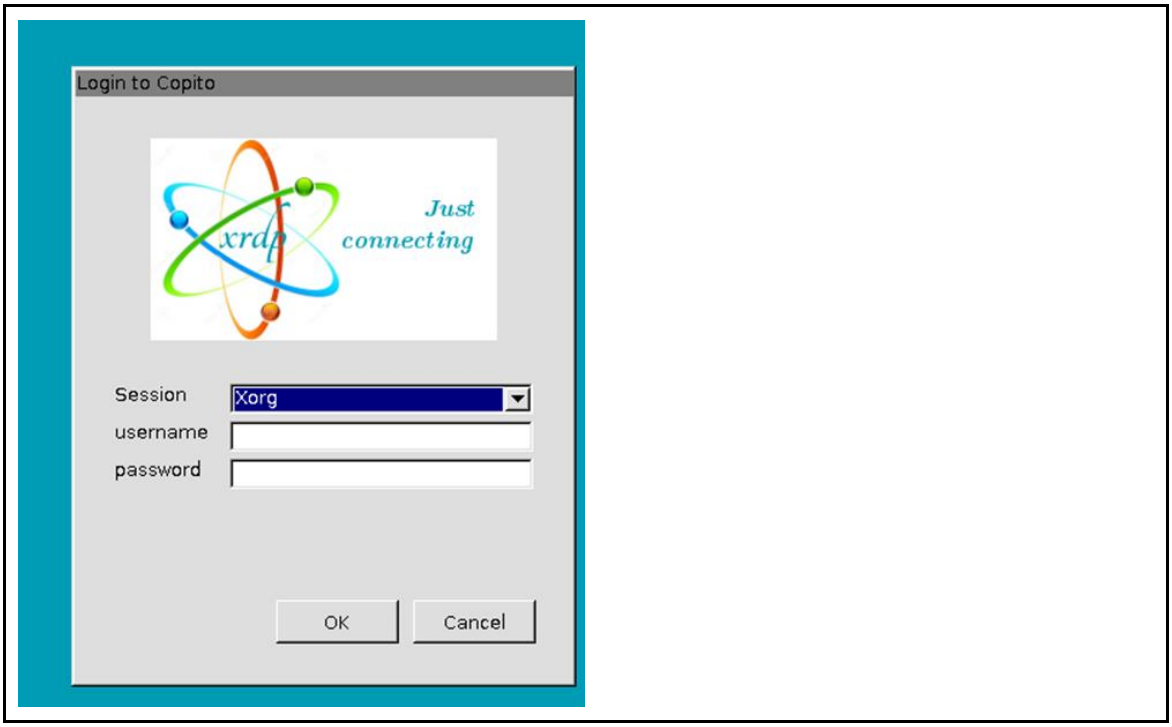
6. Mira la dirección ip de la máquina Ubuntu
   Inserta aquí la imagen donde se vea el comando usado y el resultado de dicho comando

```
camino@Copito:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:15:5d:76:1f:78 brd ff:ff:ff:ff:ff:ff
    inet 172.22.230.93/20 brd 172.22.239.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::215:5dff:fe76:1f78/64 scope link
       valid_lft forever preferred_lft forever
```

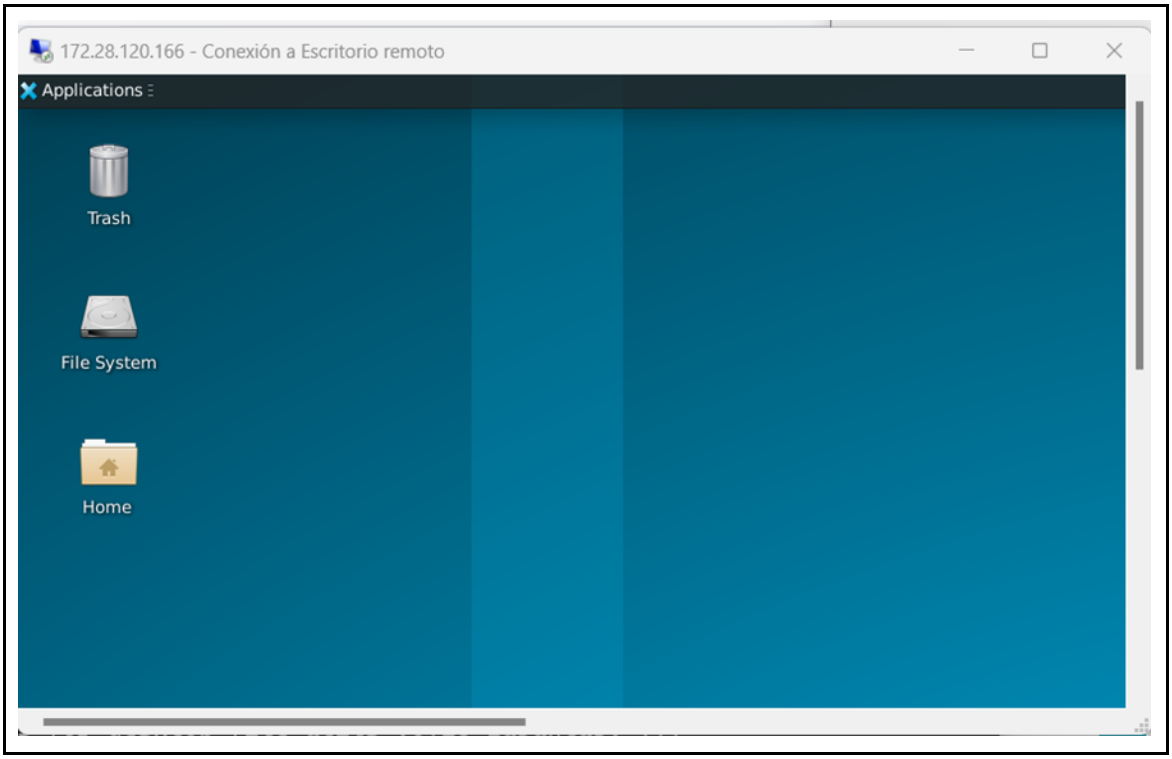
7. Abre el escritorio remoto de Windows y escribe la dirección ip para conectarte al Ubuntu
   Inserta aquí la imagen donde se vea que se ha introducido la dirección ip



8. Introduce el usuario y la contraseña
   Inserta aquí la imagen donde se vea que se ha introducido el usuario y la contraseña



9. Explora las opciones del escritorio que has instalado.
   Inserta aquí la imagen donde se vea que has accedido al escritorio que has instalado



3.- CONCLUSIONES

Explica con tus palabras lo que has aprendido con esta práctica.

Instalación de Portainer

Desarrollada por **Guadalupe Bermejo Sánchez**

1.- OBJETIVOS

Utilizar una interfaz gráfica para administrar contenedores a nivel local utilizando Portainer.

2.- HAY QUE SABER QUE PORTAINER ...

Es una imagen a partir de la cual se crea un contenedor que proporciona una interfaz gráfica para administrar entornos con contenedores tanto locales como en la nube. Lo podemos plantear como una práctica incluso.

Es una plataforma de gestión de contenedores que se utiliza para administrar y desplegar contenedores Docker de manera sencilla y accesible a través de una interfaz web intuitiva.

Es una herramienta útil para simplificar la administración de contenedores Docker sin tener que lidiar con la complejidad de los comandos de línea de comandos. Facilita la implementación y gestión de aplicaciones en contenedores de manera más eficiente y accesible.

Proporciona una interfaz de usuario web (localhost:9000) fácil de usar que permite a los usuarios administrar sus contenedores, imágenes, volúmenes y redes Docker sin necesidad de utilizar comandos de línea de comandos.

Permite administrar varios clústeres de Docker desde una única instancia. Esto es útil para administradores que gestionan contenedores en entornos distribuidos.

Proporciona una vista en tiempo real de los recursos utilizados por los contenedores, lo que ayuda a supervisar el rendimiento y el uso de recursos.

Permite crear, iniciar, detener, eliminar y gestionar contenedores Docker a través de la interfaz de Portainer. También pueden importar y exportar contenedores.

Permite buscar, descargar, eliminar y administrar imágenes Docker desde repositorios públicos o privados.

Ofrece opciones de autenticación y control de acceso basadas en roles, lo que permite definir quién puede realizar qué acciones en la plataforma.

Además de Docker, Portainer también es compatible con otros orquestadores de contenedores, como Kubernetes y Docker Swarm.

3.- INFRAESTRUCTURA (VOLÚMENES)

Portainer utiliza dos volúmenes en su configuración por defecto para persistir datos y configuraciones entre reinicios del contenedor. Estos volúmenes son necesarios para garantizar que los datos y la configuración de Portainer no se pierdan cuando el contenedor se detiene o reinicia. Aquí te explico brevemente acerca de estos volúmenes:

/data:

Este volumen almacena los datos persistentes de Portainer, como la configuración de usuarios, equipos, roles y otras preferencias de la aplicación.

Al persistir estos datos en un volumen separado, se asegura que no se pierdan incluso si el contenedor de Portainer se detiene o se elimina.

/var/run/docker.sock:

Este volumen es una conexión de socket al demonio de Docker del sistema anfitrión.

Permite que Portainer se comunique directamente con el motor de Docker del sistema anfitrión para realizar operaciones de Docker, como la creación y gestión de contenedores, imágenes, redes, volúmenes, etc.

Al montar este volumen, Portainer puede acceder al demonio de Docker del sistema anfitrión sin necesidad de instalar Docker dentro del contenedor.

Estos volúmenes se especifican en la configuración del contenedor de Portainer al crearlo utilizando el comando docker run. Asegúrate de mantener estos volúmenes persistentes para garantizar que Portainer funcione correctamente y que los datos no se pierdan entre reinicios del contenedor.

4.- PASOS A SEGUIR

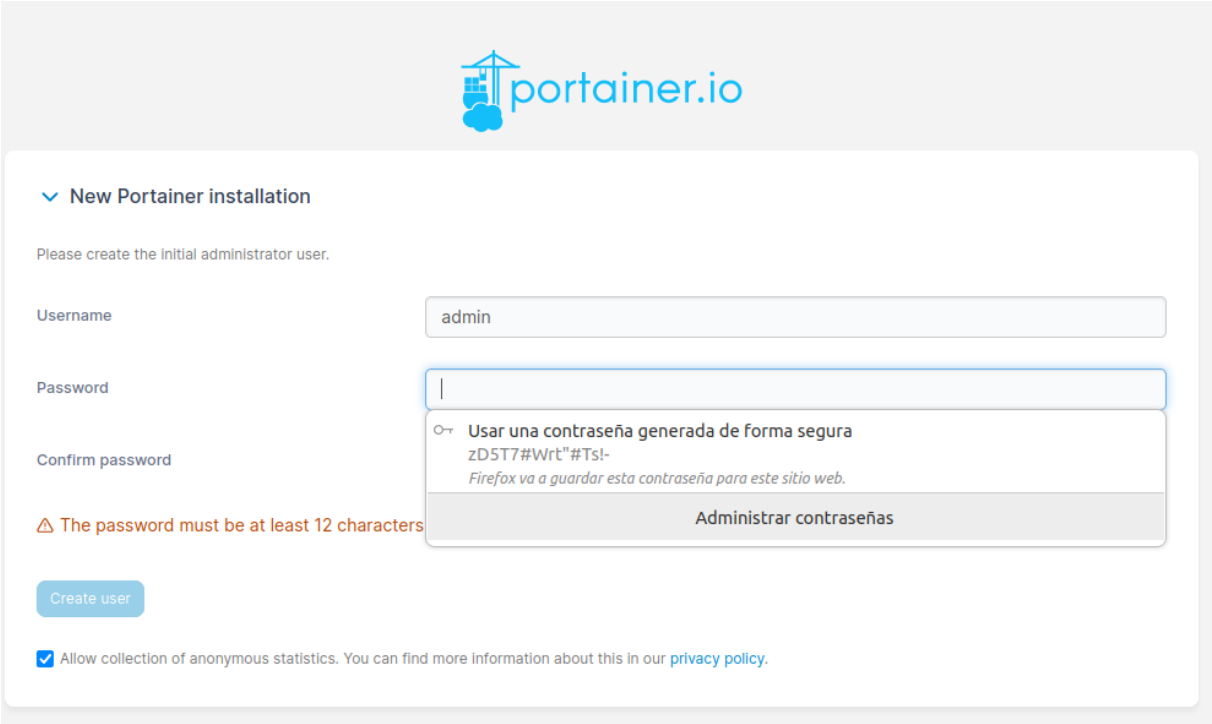
1. Crear un contenedor Portainer

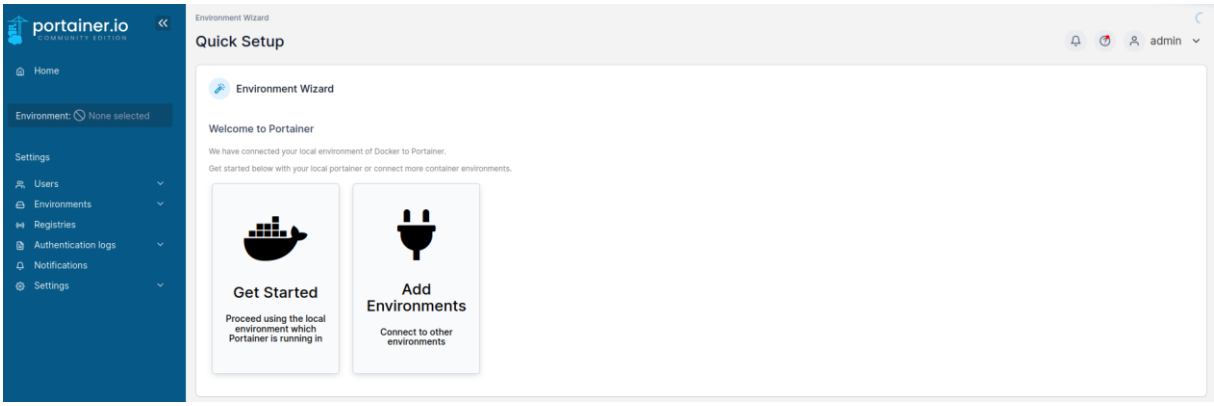
```
# $ docker run -d -p 9000:9000 --name Portainer -v
/var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data portainer/portainer
```

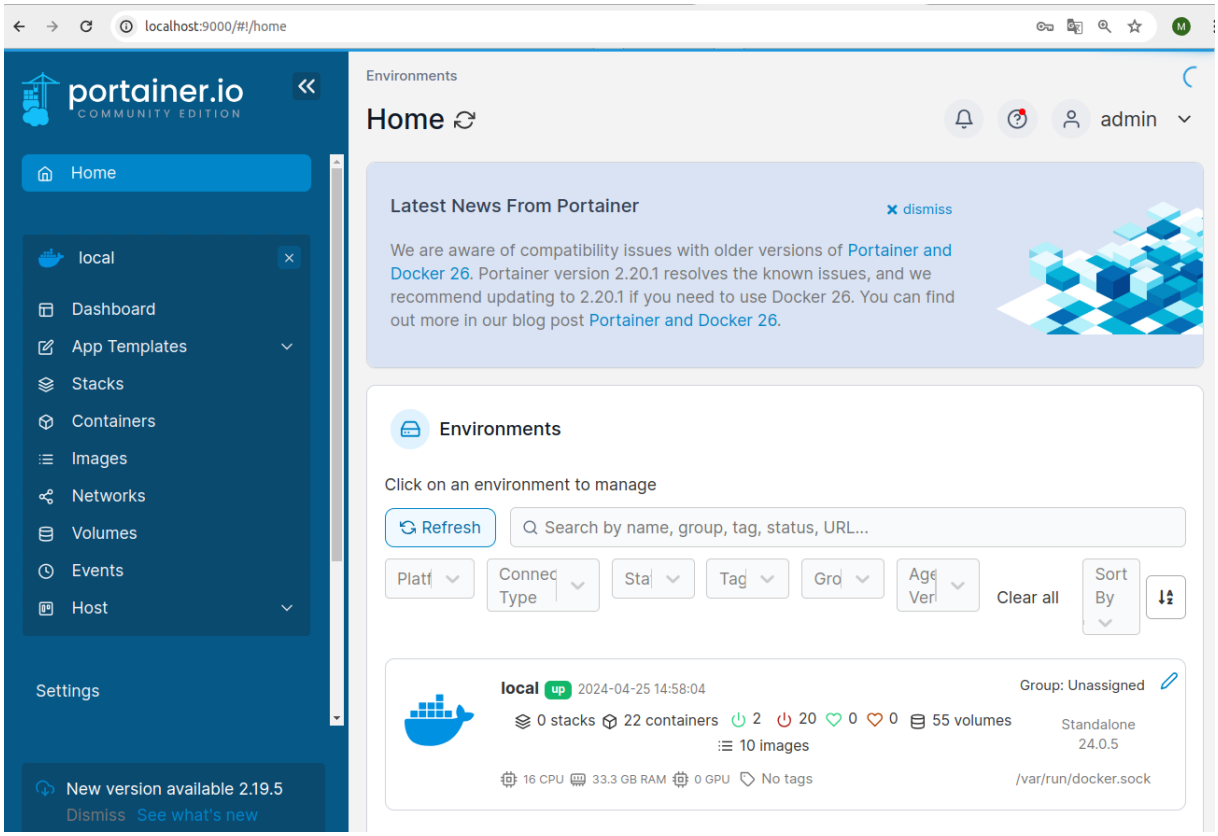
2. Acceder desde el navegador a http://localhost:9000 como admin y crear contraseña “les_Clara_del_Rey” (12 caracteres)

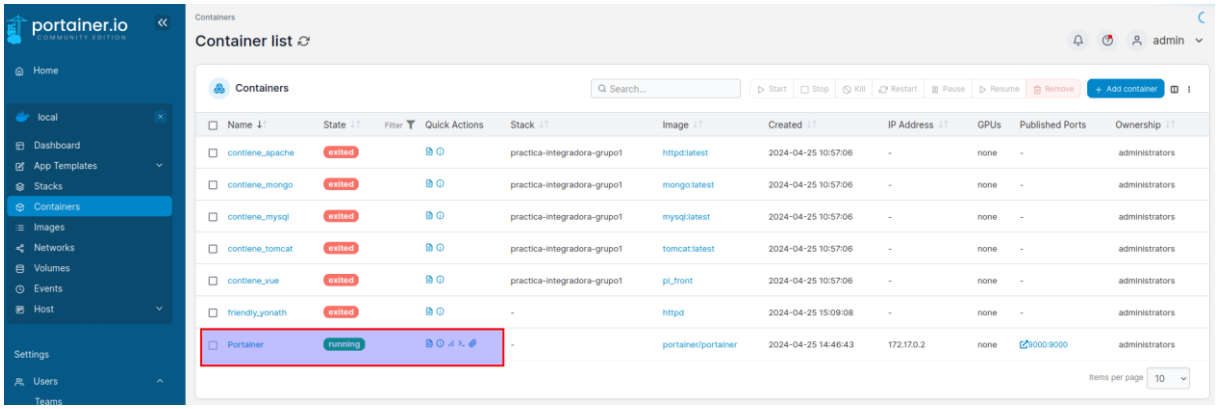
```
# docker pull mysql
```

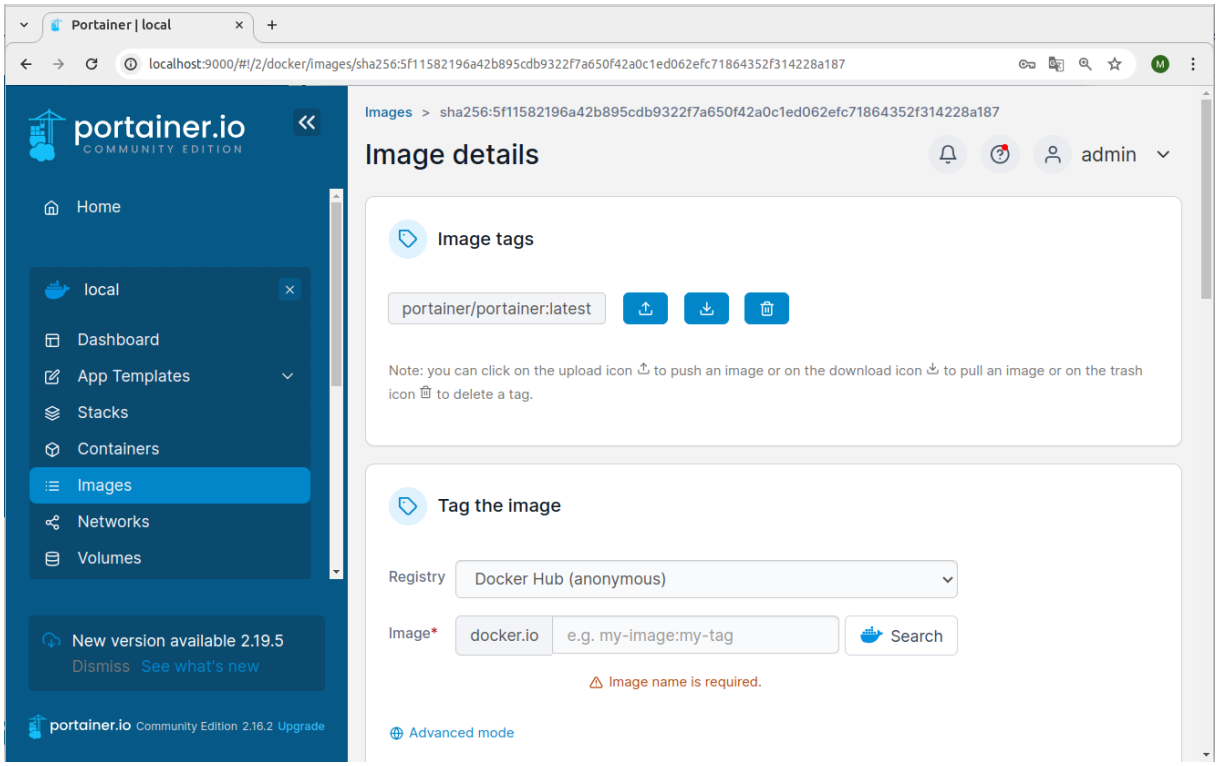
3. Navegar por la GUI de portainer

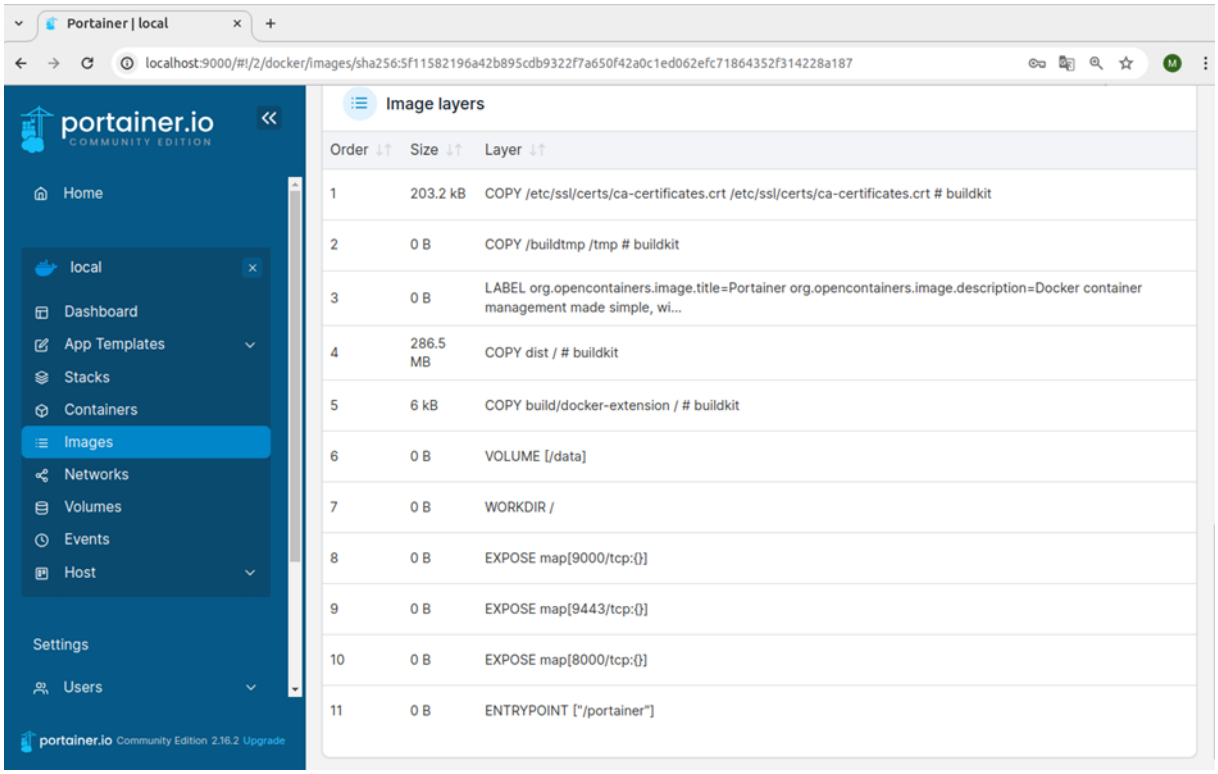


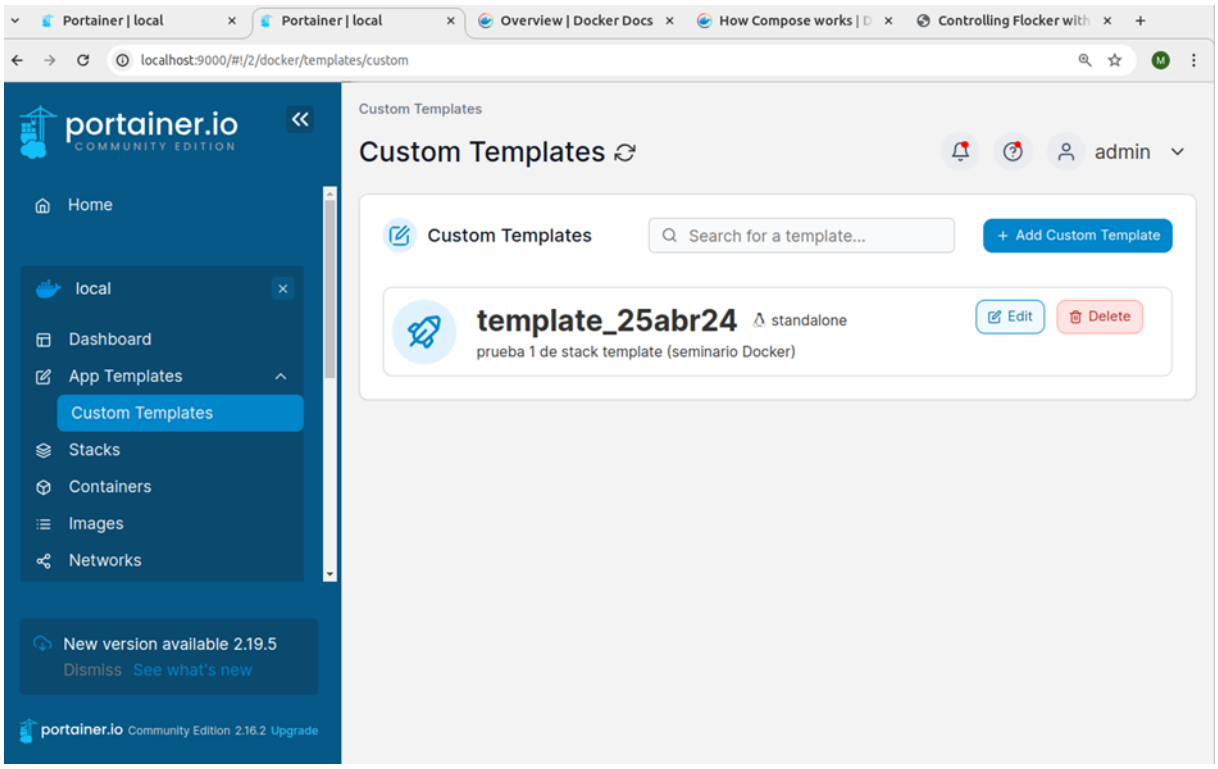



localhost:9000/#!/home
portainer.io
COMMUNITY EDITION
Environments
Home
admin
Home
local
Dashboard
App Templates
Stacks
Containers
Images
Networks
Volumes
Events
Host
Settings
New version available 2.19.5
Dismiss See what's new
Latest News From Portainer
dismiss
We are aware of compatibility issues with older versions of Portainer and Docker 26. Portainer version 2.20.1 resolves the known issues, and we recommend updating to 2.20.1 if you need to use Docker 26. You can find out more in our blog post Portainer and Docker 26.
Environments
Click on an environment to manage
Refresh
Search by name, group, tag, status, URL...
Clear all
Sort By
local up 2024-04-25 14:58:04
Group: Unassigned
0 stacks 22 containers 2 20 0 0 55 volumes
10 images
Standalone
24.0.5
16 CPU 33.3 GB RAM 0 GPU No tags
/var/run/docker.sock


portainer.io
Container list
admin
Containers
Add container
Name
State
Quick Actions
Stack
Image
Created
IP Address
GPUs
Published Ports
Ownership
contiene_apache exited practica-integradora-grupo1 httpd:latest 2024-04-25 10:57:06 none administrators
contiene_mongo exited practica-integradora-grupo1 mongo:latest 2024-04-25 10:57:06 none administrators
contiene_mysql exited practica-integradora-grupo1 mysql:latest 2024-04-25 10:57:06 none administrators
contiene_tomcat exited practica-integradora-grupo1 tomcat:latest 2024-04-25 10:57:06 none administrators
contiene_vue exited practica-integradora-grupo1 pi_front 2024-04-25 10:57:06 none administrators
friendly_yonath exited httpd 2024-04-25 15:09:08 none administrators
Portainer running portainer/portainer 2024-04-25 14:46:43 172.17.0.2 none administrators


Portainer | local
localhost:9000/#!/2/docker/images/sha256:5f11582196a42b895cdb9322f7a650f42a0c1ed062efc71864352f314228a187
portainer.io
COMMUNITY EDITION
Home
local
Dashboard
App Templates
Stacks
Containers
Images
Networks
Volumes
New version available 2.19.5
Dismiss See what's new
portainer.io Community Edition 2.16.2 Upgrade
Images > sha256:5f11582196a42b895cdb9322f7a650f42a0c1ed062efc71864352f314228a187
Image details
admin
Image tags
portainer/portainer:latest
Note: you can click on the upload icon to push an image or on the download icon to pull an image or on the trash icon to delete a tag.
Tag the image
Registry
Docker Hub (anonymous)
Image*
docker.io
e.g. my-image:my-tag
Search
Image name is required.
Advanced mode





5.- CONCLUSIONES

Explica con tus palabras lo que has aprendido con esta práctica.

Mini-aplicación en un entorno “contenedorizado” con Docker Compose

Desarrollada por **José Ramón Rodríguez Sánchez**

1. Objetivos

Desplegar una aplicación compuesta de varios contenedores usando Docker Compose.

2. Arquitectura

Se crearán dos contenedores a partir de sendas imágenes descargadas de Docker Hub:

- ubuntu/apache2
  - Se puede descargar la imagen **ubuntu/apache2**, que contiene Apache sobre Ubuntu, pero sin el módulo de PHP instalado, con lo que habría que instalarlo.
  - Se puede descargar la imagen **php:<versión>-apache**, que contiene Apache con el módulo de PHP ya instalado.
- Mysql

2.1. Configuración general

1. Se va a crear una red a la que se añadirían los contenedores:

```
$ docker network create --subnet=172.20.100.0/24 --gateway=172.20.100.1 red_aplicacion
0b5ecb0d6776e20031e00e18110ba27f36cb0204f294ef11024c0fd0ddb97f76
```

2. Se inspeccionan las características de la red recién creada:

```
$ docker network inspect red_aplicacion
[
    {
        "Name": "red_aplicacion",
        "Id": "0b5ecb0d6776e20031e00e18110ba27f36cb0204f294ef11024c0fd0ddb97f76",
        "Created": "2024-03-10T03:25:25.155597751+01:00",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "172.20.100.0/24",
```

```
                    "Gateway": "172.20.100.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {},
        "Options": {},
        "Labels": {}
    }
]
```

3. Se comprueba que en el anfitrión se creó una interfaz de red virtual que pertenece a esta red. El anfitrión se comportará como puerta de enlace de la red. El nombre de la interfaz de red virtual en modo bridge sigue el patrón br-<primeros_caracteres_id_red>:

```
$ ip a
...
59: br-0b5ecb0d6776: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state
DOWN group default
    link/ether 02:42:08:67:da:32 brd ff:ff:ff:ff:ff:ff
    inet 172.20.100.1/24 brd 172.20.100.255 scope global br-0b5ecb0d6776
       valid_lft forever preferred_lft forever
...
```

3. APACHE CON LA IMAGEN HTTPD

3.1. Instalación

1. Para localizar la imagen adecuada, bien se utiliza un buscador en un navegador, bien se busca con **`docker search`**:

```
$ docker search httpd
NAME    DESCRIPTION                     STARS     OFFICIAL AUTOMATED
httpd   The Apache HTTP Server Project  4629      [OK]
```

2. Se descarga la imagen oficial de Apache (httpd) de DockerHub:

```
$ docker pull httpd
```

3. A modo de prueba, también se descargará la otra imagen que se presentó anteriormente como alternativa, que contiene Apache con el módulo PHP ya instalado:

```
$ docker pull php:8.2-apache
8.2-apache: Pulling from library/php
```

```
2f44b7a888fa: Pull complete
3a95dcec6035: Pull complete
e22afa33f327: Pull complete
1aa61ea11ee8: Pull complete
ea67ae2bde33: Pull complete
ff5356c6468f: Pull complete
5e91cb51e886: Pull complete
cc4d86fac990: Pull complete
7cb6037707dd: Pull complete
c8f7dbd801ff: Pull complete
8ae0a6a9855f: Pull complete
578d63e69066: Pull complete
0b12a7da9776: Pull complete
Digest:
sha256:71eac2ebabafd604c5a2f63087cb0b84ba96ee1b7fd46b5615f4ea420ea9489a
Status: Downloaded newer image for php:8.2-apache
docker.io/library/php:8.2-apache

What's Next?
  1. Sign in to your Docker account → docker login
  2. View a summary of image vulnerabilities and recommendations →
docker scout quickview php:8.2-apache
```

4. Se listan las imágenes descargadas. Se puede comprobar que la segunda imagen descargada es mucho más voluminosa que la primera:

```
$ docker images
REPOSITORY    TAG          IMAGE ID       CREATED         SIZE
httpd         latest       e499c02ff073   2 months ago    174MB
php           8.2-apache   e7486dc45c18   15 hours ago    503MB
```

5. Se crea un volumen, llamado **volumen_apache**, para poder conservar los datos desplegados en el servidor apache aunque se destruya el contenedor:

```
$ docker volume create volumen_apache
volumen_apache
```

6. Se crea un contendor, de nombre **contiene_apache**, y se le vincula la imagen **httpd**. Se puede usar el comando **`docker container create`** o el abreviado **`docker create`**:

```
$ docker container create --name contiene_apache httpd
40590aa093d14856929d0cdc95ea1796fc568529a53fc29c85258f095c4afbc1
```

7. Se listan los contenedores existentes:

```
$ docker ps -a
CONTAINER ID  IMAGE  COMMAND              CREATED         STATUS   PORTS  NAMES
40590aa093d1  httpd  "httpd-foreground"  16 minutes ago  Created         contiene_apache
```

8. Se arranca la imagen **httpd** en el contenedor recién creado. Se puede usar el comando **`docker container start`** o el abreviado **`docker start`**:

```
$ docker container start contiene_apache
contiene_apache
```

9. Se comprueba que la imagen está corriendo correctamente en el contendor, listando de nuevo los contenedores:

```
$ docker ps
CONTAINER ID  IMAGE  COMMAND              CREATED         STATUS   PORTS  NAMES
40590aa093d1   httpd     "httpd-foreground"   26 minutes ago   Up 35 seconds   80/tcp
contiene_apache
```

En el listado se puede observar que el puerto 80/tcp está abierto en el contenedor.

10. Se podría haber creado y arrancado el contendor en un solo paso, usando **`docker run`**:

```
$ docker run -dit --name contiene_apache -p 8080:80
      -v "$PWD":/usr/local/apache2/htdocs/ httpd
```

11. Se comprueba a qué red y con qué dirección IP se ha asociado el contendor al arrancarse:

```
$ docker inspect contiene_apache
[
   {
      "Id":
"40590aa093d14856929d0cdc95ea1796fc568529a53fc29c85258f095c4afbc1",
      ...
      "NetworkSettings": {
        "Bridge": "",
        "SandboxID":
"ba1e34a54147e5437ff295de08a7a8b0de1faa2a1e2b3572b75affa9a726e8db",
        "HairpinMode": false,
        "LinkLocalIPv6Address": "",
        "LinkLocalIPv6PrefixLen": 0,
        "Ports": {
          "80/tcp": null
        },
        "SandboxKey": "/var/run/docker/netns/ba1e34a54147",
        "SecondaryIPAddresses": null,
        "SecondaryIPv6Addresses": null,
        "EndpointID":
"2fbc485df7b069e91f7e9d9a6a36208d9275e48f58e70d7ac405d6a5bedf14a5",
        "Gateway": "172.17.0.1",
        "GlobalIPv6Address": "",
        "GlobalIPv6PrefixLen": 0,
        "IPAddress": "172.17.0.2",
        "IPPrefixLen": 16,
        "IPv6Gateway": "",
        "MacAddress": "02:42:ac:11:00:02",
        "Networks": {
          "bridge": {
```

```
            "IPAMConfig": null,
            "Links": null,
            "Aliases": null,
            "NetworkID":
"1ff2db504c90b8ebd01e3a3dc3d3606e1863fcad755ff6972828d4fae31f1230",
            "EndpointID":
"2fbc485df7b069e91f7e9d9a6a36208d9275e48f58e70d7ac405d6a5bedf14a5",
            "Gateway": "172.17.0.1",
            "IPAddress": "172.17.0.2",
            "IPPrefixLen": 16,
            "IPv6Gateway": "",
            "GlobalIPv6Address": "",
            "GlobalIPv6PrefixLen": 0,
            "MacAddress": "02:42:ac:11:00:02",
            "DriverOpts": null
        }
    }
  }
]
```

3.2. Configuración

1. Se comprueba que se puede acceder desde la máquina anfitrión al Apache existente en el contenedor **contiene_apache**. Para ello, se ejecuta un navegador en el que se indica la dirección que Docker le asignó al contendor de Apache, es decir, http://172.17.0.2:

2. Se abre una shell en el contenedor para trabajar con su sistema de ficheros:

```
$ docker exec -it contiene_apache bash
root@40590aa093d1:/usr/local/apache2#
```

   Al arrancar la shell, el directorio donde se posiciona es **`/usr/local/apache2`**.

3. Se localiza el archivo que se sirve por defecto y se consulta su contenido:

```
root@40590aa093d1:/usr/local/apache2# ls
 bin  build  cgi-bin  conf  error  htdocs  icons  include  logs
modules

root@40590aa093d1:/usr/local/apache2# cd htdocs/

root@40590aa093d1:/usr/local/apache2/htdocs# ls
```

```
index.html

root@40590aa093d1:/usr/local/apache2/htdocs# cat index.html
<html><body><h1>It works!</h1></body></html>
```

4. Se comprueba la distribución y versión del Linux existente en el contenedor (en la página web de la imagen informa que es un Debian). Existen varios archivos de configuración y comandos que ofrecen esta información:

```
# cat /etc/issue
Debian GNU/Linux 12 \n \l
# cat /etc/debian_version
12.4

# uname -a
Linux 40590aa093d1 6.2.0-39-generic #40~22.04.1-Ubuntu SMP
PREEMPT_DYNAMIC Thu Nov 16 10:53:04 UTC 2 x86_64 GNU/Linux

# cat /proc/version
Linux version 6.2.0-39-generic (buildd@lcy02-amd64-045) (x86_64-
linux-gnu-gcc-11 (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0, GNU ld (GNU
Binutils for Ubuntu) 2.38) #40~22.04.1-Ubuntu SMP PREEMPT_DYNAMIC
Thu Nov 16 10:53:04 UTC 2

# cat /etc/os-release
PRETTY_NAME="Debian GNU/Linux 12 (bookworm)"
NAME="Debian GNU/Linux"
VERSION_ID="12"
VERSION="12 (bookworm)"
VERSION_CODENAME=bookworm
ID=debian
HOME_URL="https://www.debian.org/"
SUPPORT_URL="https://www.debian.org/support"
BUG_REPORT_URL="https://bugs.debian.org/"
```

5. Se vio anteriormente que el ejecutable que produce que el contenedor se quede abierto y "escuchado" es **httpd-foreground**. Se comprueba su contenido:

```
root@40590aa093d1:/usr/local/apache2# cd ../bin

root@40590aa093d1:/usr/local/bin# ls -la
total 16
drwxr-xr-x 1 root root 4096 Dec 13 19:50 .
drwxr-xr-x 1 root root 4096 Dec 13 19:48 ..
-rwxr-xr-x 1 root root  138 Dec 13 19:48 httpd-foreground

root@40590aa093d1:/usr/local/bin# cat httpd-foreground
#!/bin/sh
# set -e, exits if a command exits with a non-zero status
set -e
# Apache gets grumpy about PID files pre-existing
rm -f /usr/local/apache2/logs/httpd.pid
```

```
exec httpd -DFOREGROUND "$@"
```

4. APACHE CON LA IMAGEN UBUNTU/APACHE2

4.1. Instalación

1. Para localizar la imagen adecuada, bien se utiliza un buscador en un navegador, bien se busca con **`docker search`**:

```
$ docker search ubuntu
NAME    DESCRIPTION                     STARS     OFFICIAL AUTOMATED
ubuntu/apache2                  Apache, a secure & extensible open-source HT…   71
...
```

2. Se descarga la imagen:

```
$ docker pull ubuntu/apache2
...
```

3. Se listan las imágenes descargadas. Se puede comprobar que la segunda imagen descargada es mucho más voluminosa que la primera:

```
$ docker images
REPOSITORY       TAG              IMAGE ID       CREATED       SIZE
...
ubuntu/apache2   2.4-22.04_beta   e6ad5603c470   8 weeks ago   194MB
...
```

4. Se crea un volumen, llamado **volumen_apache**, para poder conservar los datos desplegados en el servidor apache aunque se destruya el contenedor:

```
$ docker volume create volumen_apache
volumen_apache
```

5. Se crea un contendor, de nombre **contiene_apache**, y se le vincula la imagen **ubuntu/apache2**. Se puede usar el comando **`docker container create`** o el abreviado **`docker create`**:

```
$ docker container create --name contiene_apache ubuntu/apche2
40590aa093d14856929d0cdc95ea1796fc568529a53fc29c85258f095c4afbc1
```

6. Se listan los contenedores existentes:

```
$ docker ps -a
CONTAINER ID   IMAGE           COMMAND             CREATED          STATUS   PORTS NAMES
...
40590aa093d1   ubuntu/apache2  "httpd-foreground"  16 minutes ago   Created        contiene_apache
...
```

7. Se arranca el contenedor recién creado. Se puede usar el comando **`docker container start`** o el abreviado **`docker start`**:

```
$ docker container start contiene_apache
```

```
contiene_apache
```

8. Se comprueba que la imagen está corriendo correctamente en el contendor, listando de nuevo los contenedores:

```
$ docker ps
CONTAINER ID IMAGE          COMMAND             CREATED        STATUS        PORTS
NAMES
...
40590aa093d1 ubuntu/apache2 "httpd-foreground"  26 minutes ago Up 35 seconds 80/tcp
contiene_apache
...
```

En el listado se puede observar que el puerto 80/tcp está abierto en el contenedor.

9. Se podría haber creado y arrancado el contendor en un solo paso, usando **`docker run`**:

```
$ docker run -it --name contiene_apache --hostname servidor-apache \
      -p 8080:80 -v "$PWD":/var/www --network red-aplicacion \
      ubuntu/apache2
```

10. Se comprueba a qué red y con qué dirección IP se ha asociado el contendor al arrancarse:

```
$ docker inspect contiene_apache
[
   {
      "Id":
"40590aa093d14856929d0cdc95ea1796fc568529a53fc29c85258f095c4afbc1",
      ...
      "NetworkSettings": {
        "Bridge": "",
        "SandboxID":
"ba1e34a54147e5437ff295de08a7a8b0de1faa2a1e2b3572b75affa9a726e8db",
        "HairpinMode": false,
        "LinkLocalIPv6Address": "",
        "LinkLocalIPv6PrefixLen": 0,
        "Ports": {
          "80/tcp": null
        },
        "SandboxKey": "/var/run/docker/netns/ba1e34a54147",
        "SecondaryIPAddresses": null,
        "SecondaryIPv6Addresses": null,
        "EndpointID":
"2fbc485df7b069e91f7e9d9a6a36208d9275e48f58e70d7ac405d6a5bedf14a5",
        "Gateway": "172.17.0.1",
        "GlobalIPv6Address": "",
        "GlobalIPv6PrefixLen": 0,
        "IPAddress": "172.17.0.2",
        "IPPrefixLen": 16,
        "IPv6Gateway": "",
        "MacAddress": "02:42:ac:11:00:02",
        "Networks": {
```

```
            "bridge": {
              "IPAMConfig": null,
              "Links": null,
              "Aliases": null,
              "NetworkID":
"1ff2db504c90b8ebd01e3a3dc3d3606e1863fcad755ff6972828d4fae31f1230",
              "EndpointID":
"2fbc485df7b069e91f7e9d9a6a36208d9275e48f58e70d7ac405d6a5bedf14a5",
              "Gateway": "172.17.0.1",
              "IPAddress": "172.17.0.2",
              "IPPrefixLen": 16,
              "IPv6Gateway": "",
              "GlobalIPv6Address": "",
              "GlobalIPv6PrefixLen": 0,
              "MacAddress": "02:42:ac:11:00:02",
              "DriverOpts": null
            }
        }
    }
  }
]
```

A partir de aquí se continua desde los pasos anteriores:

11. Se arranca el contendor:

```
$ docker run -it --name contiene_apache --hostname servidor-apache \
    --network red-aplicacion -v volumen_apache:/var/www \
    ubuntu/apache2
AH00558: apache2: Could not reliably determine the server's fully qualified domain
name, using 172.17.0.3. Set the 'ServerName' directive globally to suppress this
message
AH00558: apache2: Could not reliably determine the server's fully qualified domain
name, using 172.17.0.3. Set the 'ServerName' directive globally to suppress this
message
[Sat Mar 09 20:26:25.874450 2024] [mpm_event:notice] [pid 24:tid 133002255169408]
AH00489: Apache/2.4.57 (Ubuntu) configured -- resuming normal operations
[Sat Mar 09 20:26:25.874605 2024] [core:notice] [pid 24:tid 133002255169408]
AH00094: Command line: '/usr/sbin/apache2 -D FOREGROUND'
```

12. Se consultan los contendores que están activos:

```
$ docker ps
CONTAINER ID   IMAGE            COMMAND                  CREATED          STATUS
PORTS                NAMES
7790c0df586d   ubuntu/apache2   "apache2-foreground"     12 seconds ago   Up 12 seconds
80/tcp               contiene_apache
2932933a1d95   mysql            "docker-entrypoint.s…"   39 minutes ago   Up 39 minutes
3306/tcp, 33060/tcp  contiene_mysql
```

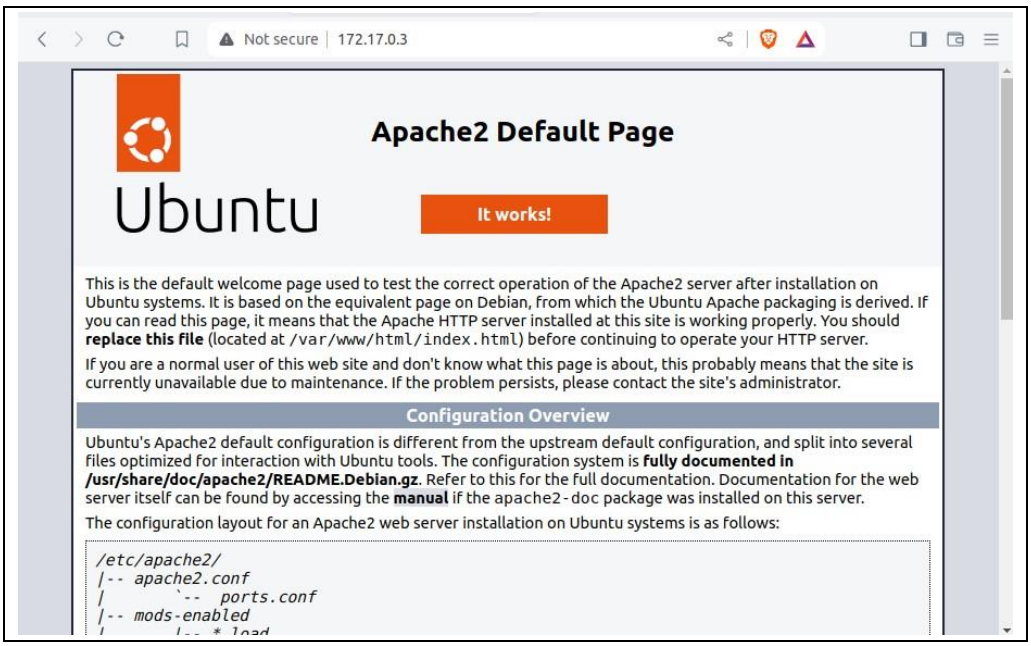
13. Se lista en el anfitrión el contendor del volumen llamado volumen_apache:

```
$ sudo su

# ls -la /var/lib/docker/volumes/volumen_apache/_data
total 12
drwxr-xr-x 3 root root 4096 mar  9 21:26 .
```

```
drwx-----x 3 root root 4096 mar  9 21:13 ..
drwxr-xr-x 2 root root 4096 mar  9 21:26 html
```

4.2. Configuración

1. Se comprueba que se está pudiendo acceder al Apache del contendor:



2. Se arranca una shell en el contendor. Se actualizan repositorios y se instala **php**, que incluye el módulo para apache. Se activa el módulo para apache:

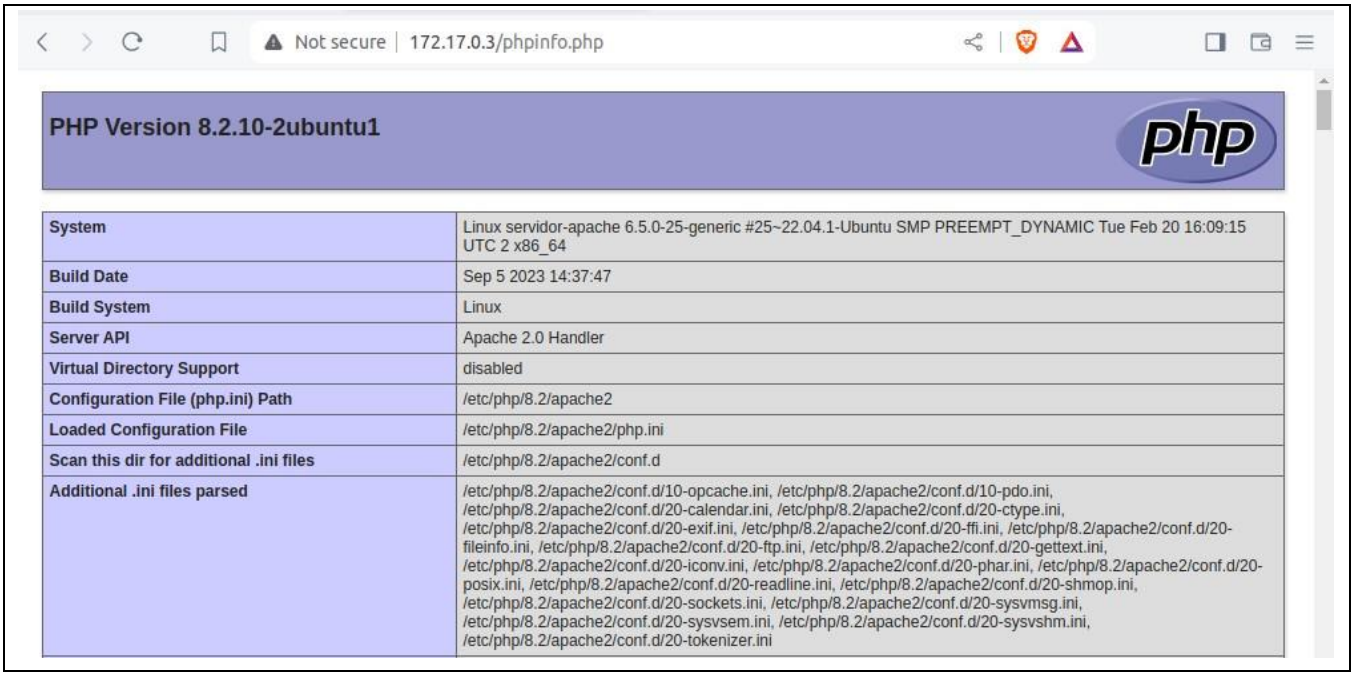
```
$ docker exec -it contiene_apache bash
#

# apt update
Get:1 http://archive.ubuntu.com/ubuntu mantic InRelease [256 kB]
Get:2 http://security.ubuntu.com/ubuntu mantic-security InRelease [109 kB]
Get:3 http://archive.ubuntu.com/ubuntu mantic-updates InRelease [109 kB]
Get:4 http://archive.ubuntu.com/ubuntu mantic-backports InRelease [90.8 kB]
Get:5 http://archive.ubuntu.com/ubuntu mantic/restricted amd64 Packages [180 kB]
...

# apt install php
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libapache2-mod-php8.2 libargon2-1 libbsd0 libedit2 libmagic-mgc libmagic1
libsodium23 php-common php8.2
  php8.2-cli php8.2-common php8.2-opcache php8.2-readline psmisc ucf
Suggested packages:
  php-pear file
The following NEW packages will be installed:
  libapache2-mod-php8.2 libargon2-1 libbsd0 libedit2 libmagic-mgc libmagic1
libsodium23 php php-common php8.2
  php8.2-cli php8.2-common php8.2-opcache php8.2-readline psmisc ucf
```

```
0 upgraded, 16 newly installed, 0 to remove and 2 not upgraded.
Need to get 5715 kB of archives.
After this operation, 31.5 MB of additional disk space will be used.
Do you want to continue? [Y/n]
...

# a2enmod php8.2
Considering dependency mpm_prefork for php8.2:
Considering conflict mpm_event for mpm_prefork:
Considering conflict mpm_worker for mpm_prefork:
Module mpm_prefork already enabled
Considering conflict php5 for php8.2:
Module php8.2 already enabled

# apachectl restart
…
Se apacga el contenedor al apagar el servicio
Se arranca de nuevo
$ docker start contiene_apache

$ docker exec -it contiene_apache bash

# apachectl -S
AH00558: apache2: Could not reliably determine the server's fully qualified domain
name, using 172.17.0.3. Set the 'ServerName' directive globally to suppress this
message
VirtualHost configuration:
*:80                   172.17.0.3 (/etc/apache2/sites-enabled/000-default.conf:1)
ServerRoot: "/etc/apache2"
Main DocumentRoot: "/var/www/html"
Main ErrorLog: "/dev/stderr"
Mutex default: dir="/var/run/apache2/" mechanism=default
Mutex mpm-accept: using_defaults
Mutex watchdog-callback: using_defaults
PidFile: "/var/run/apache2/apache2.pid"
Define: DUMP_VHOSTS
Define: DUMP_RUN_CFG
User: name="www-data" id=33
Group: name="www-data" id=33
```

3. Se crea una página php de prueba en el directorio de publicación:

```
# nano /var/www/html/phpinfo.php

<?php
Phpinfo();
¿>
```

4. Se trata de acceder desde un navegador:



5. Se crea el script php que accederá a mysql. Para facilitar la tarea se instala nano en el contendor:

```
# apt install nano
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Suggested packages:
  hunspell
The following NEW packages will be installed:
  nano
0 upgraded, 1 newly installed, 0 to remove and 2 not upgraded.
Need to get 280 kB of archives.
After this operation, 860 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu mantic/main amd64 nano amd64 7.2-1 [280 kB]
...
```

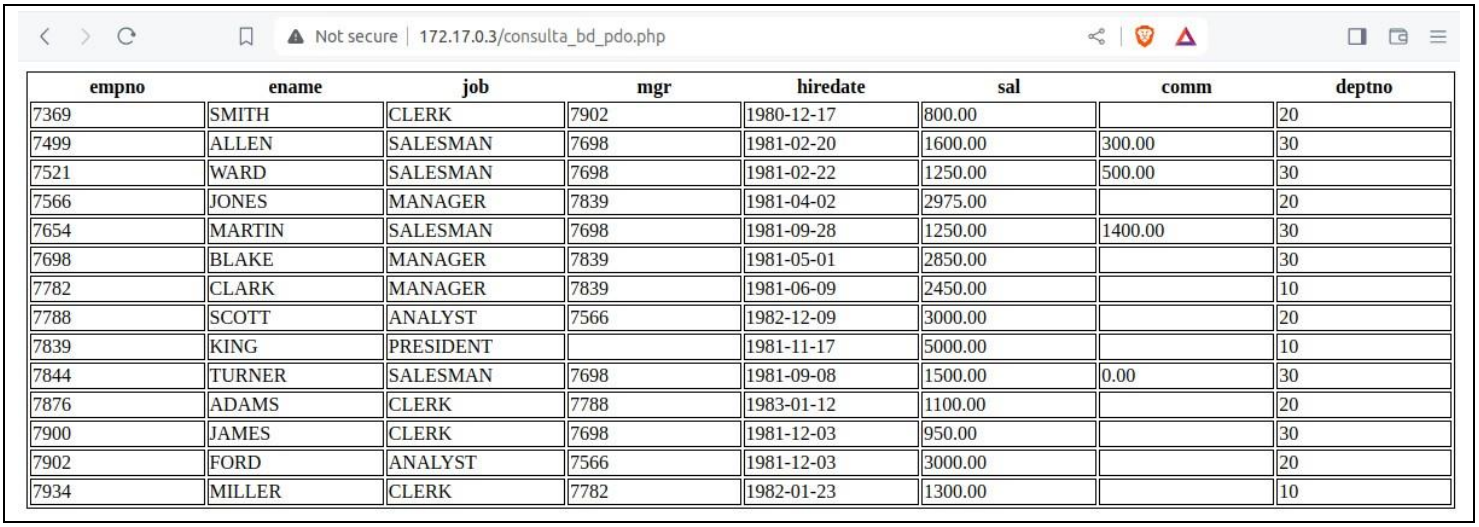
6. Se instala el módulo **mysqli** para poder hacer conexiones a MySQL usando las bibliotecas **mysqli** y **pdo**:

```
# apt install php-mysqli
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Note, selecting 'php8.2-mysql' instead of 'php-mysqli'
The following NEW packages will be installed:
  php8.2-mysql
0 upgraded, 1 newly installed, 0 to remove and 2 not upgraded.
Need to get 127 kB of archives.
After this operation, 466 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu mantic/main amd64 php8.2-mysql
amd64 8.2.10-2ubuntu1 [127 kB]
Fetched 127 kB in 3s (39.1 kB/s)
...
```

7. Se activa (descomentan) en el archivo de configuración de php, llamado php.ini, las extensiones necesarias:

```
# nano /etc/php/8.2/apache2/php.ini
...
extension=mysqli
...
extension=pdo_mysql
...
```

8. Se comprueba desde un navegador que se ejecuta correctamente la página:



empno	ename	job	mgr	hiredate	sal	comm	deptno
7369	SMITH	CLERK	7902	1980-12-17	800.00		20
7499	ALLEN	SALESMAN	7698	1981-02-20	1600.00	300.00	30
7521	WARD	SALESMAN	7698	1981-02-22	1250.00	500.00	30
7566	JONES	MANAGER	7839	1981-04-02	2975.00		20
7654	MARTIN	SALESMAN	7698	1981-09-28	1250.00	1400.00	30
7698	BLAKE	MANAGER	7839	1981-05-01	2850.00		30
7782	CLARK	MANAGER	7839	1981-06-09	2450.00		10
7788	SCOTT	ANALYST	7566	1982-12-09	3000.00		20
7839	KING	PRESIDENT		1981-11-17	5000.00		10
7844	TURNER	SALESMAN	7698	1981-09-08	1500.00	0.00	30
7876	ADAMS	CLERK	7788	1983-01-12	1100.00		20
7900	JAMES	CLERK	7698	1981-12-03	950.00		30
7902	FORD	ANALYST	7566	1981-12-03	3000.00		20
7934	MILLER	CLERK	7782	1982-01-23	1300.00		10

5. MySQL

5.1. Instalación

1. Se ejecuta la imagen **mysql** en un contendor llamado **contiene_mysql**. En un solo comando se realizan las siguientes acciones:
    a. Se descargará la imagen
    b. Se creará el contenedor
    c. Se correrá la imagen dentro del contendor

```
$ docker run --name contiene_mysql --hostname servidor-mysql \
      -v volumen_mysql:/var/lib/mysql \
      -e MYSQL_ROOT_PASSWORD=clave-de-root \
      mysql
defe847ca387c9bbc337adb9afa0df1bfa286a29a5a5981151d229e50677d165
```

Los parámetros utilizados en **docker run** son:

PARÁMETRO	SIGNIFICADO
--name	Establece el nombre del contenedor
--hostname	Establece el nombre de equipo del sistema operativo del contenedor
-v	Crea un volumen y se lo asocia al contendor
-e	Define variables de entorno

Las variables de entorno que se pueden establecer para MySQL son:

VARIABLE de ENTORNO	SIGNIFICADO
MYSQL_ROOT_PASSWORD	Es obligatoria. Permite definir la contraseña del usuario **root** de MySQL.
MYSQL_DATABASE	Es opcional. Permite especificar el nombre de una BD que se creará al iniciar la imagen. Si se proporcionó un usuario/contraseña (ver más abajo), a ese usuario se le otorgará acceso de superusuario (correspondiente a GRANT ALL) a esta base de datos.
MYSQL_USER	Es opcional. Se utilizan junto con la creación de un nuevo usuario y para establecer la contraseña de ese usuario. A este usuario se le otorgarán permisos de superusuario (ver arriba) para la BD especificada por la variable MYSQL_DATABASE. Ambas variables son necesarias para que se cree un usuario. No es necesario utilizar este mecanismo para crear el superusuario **root**, ese usuario se crea de forma predeterminada con la contraseña especificada por la variable MYSQL_ROOT_PASSWORD.
MYSQL_PASSWORD	Es opcional. Se utiliza junto con la creación de un nuevo usuario y para establecer la contraseña de ese usuario. A este usuario se le otorgarán permisos de superusuario (ver arriba) para la BD especificada por la variable MYSQL_DATABASE. Ambas variables son necesarias para que se cree un usuario. No es necesario utilizar este mecanismo para crear el superusuario **root**, ese usuario se crea de forma predeterminada con la contraseña especificada por la variable MYSQL_ROOT_PASSWORD.

MYSQL_ALLOW_EMPTY_PASSWORD	Es opcional. Si se establece a un valor no vacío, como **yes**, permite que el contenedor se inicie con una contraseña en blanco para el usuario **root**. NOTA: No se recomienda establecer esta variable en **yes** a menos que realmente se sepa lo que se está haciendo, ya que esto dejará la instancia de MySQL completamente desprotegida, lo que permitirá que cualquiera obtenga acceso completo de superusuario.
MYSQL_RANDOM_ROOT_PASSWORD	Es opcional. Si se establece a un valor no vacío, como **yes**, se genera una contraseña inicial aleatoria para el usuario **root** (usando el comando **pwgen**). La contraseña de **root** generada se imprimirá en stdout (GENERATED ROOT PASSWORD: .....).
MYSQL_ONETIME_PASSWORD	Establece el usuario **root** (no el usuario especificado en la variable MYSQL_USER) como expirado una vez que se completa el inicio, lo que obliga a un cambio de contraseña en el primer inicio de sesión. Cualquier valor no vacío activará esta configuración. NOTA: Esta función solo es compatible con MySQL 5.6+. El uso de esta opción en MySQL 5.5 arrojará un error apropiado durante la inicialización.
MYSQL_INITDB_SKIP_TZINFO	De forma predeterminada, el script de punto de entrada carga automáticamente los datos de zona horaria necesarios para la función CONVERT_TZ(). Si no es necesario, cualquier valor que no esté vacío deshabilita la carga de zona horaria.

2. Se comprueba el estado de contendor:

```
$ docker ps
CONTAINER ID IMAGE COMMAND                  CREATED         STATUS         PORTS
NAMES
...
defe847ca387 mysql "docker-entrypoint.s…" 46 seconds ago  Up 45 seconds  3306/tcp,33060/tcp
contiene_mysql
...
```

3. Para probar que MySQL funciona correctamente, se puede instalar en el anfitrión un cliente de MySQL, bien el de CLI, bien el IDE Workbench:

```
 # apt install mysql-client-core-8.0
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  mysql-client-core-8.0
0 upgraded, 1 newly installed, 0 to remove and 20 not upgraded.
Need to get 2.682 kB of archives.
...
```

5.2. Configuración

1. Se prueba el funcionamiento de MySQL ejecutando el comando **mysql** (cliente de MySQL de CLI). Se indican como parámetros la IP del contenedor (**-h**

172.17.0.3), el usuario con el que se accederá a MySQL (**-u root**) y que se solicite la contraseña de ese usuario (**-p**):

```
$ mysql -h 172.17.0.3 -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 12
Server version: 8.2.0 MySQL Community Server - GPL

Copyright (c) 2000, 2023, 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 (0,00 sec)
```

En este punto ya se puede (y se debe) ejecutar el script de php.

2. El siguiente comando iniciaría otra instancia de contenedor corriendo la imagen **mysql** y ejecutará el cliente de línea de comandos **mysql** en el contenedor mysql original, lo que le permite ejecutar sentencias SQL en su instancia de base de datos:

```
$ docker run -it --rm mysql
      mysql -h contiene-mysql -p
...
```

Los parámetros utilizados en **docker run** son:

PARÁMETRO	SIGNIFICADO
-i	El contenedor es interactivo
-t	Se ejecuta un terminal
--rm	El contendor se elimina cuando deja de ejecutarse

El comando que se ejecuta sobre el contenedor es la invocación del cliente mysql de línea de comandos:

PARÁMETRO	SIGNIFICADO
-h	Host
-p	Se solicitará la contraseña

6. DESARROLLO SOBRE LOS CONTENDORES

6.1. Contenedor de Apache

1. Se arranca una shell. El directorio de trabajo en el contenedor, donde se accede con la shell por defecto, es /usr/local/apache:

```
$ docker exec -it contiene_apache bash
root@40590aa093d1:/usr/local/apache2#
```

2. Se actualiza el contendor:

```
# apt update

# apt upgrade

# apt install nano

# apt install curl

# apt install lynx

# apt install iproute2
```

3. Se listan los módulos existentes y se comprueba que el de PHP no está instalado, por lo que se instala, tanto el lenguaje PHP, como el módulo de PHP para Apache. Al instalar PHP se comprueba que ya se instala por defecto el módulo de php:

```
# ls modules
httpd.exp  mod_cern_meta.so  mod_lbmethod_heartbeat.so
mod_proxy_wstunnel.so        mod_access_compat.so
mod_cgi.so                   mod_ldap.so
...

# apt install php
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  apache2 apache2-bin apache2-data apache2-utils libapache2-mod-
php8.2 libaprutil1-dbd-sqlite3 libargon2-1 libbsd0 libedit2 libgdbm-
compat4 liblua5.3-0 libmagic-mgc libmagic1 libperl5.36 libproc2-0
libsodium23 libsqlite3-0 media-types netbase perl perl-modules-5.36
php-common php8.2 php8.2-cli php8.2-common php8.2-opcache php8.2-
readline procps psmisc sensible-utils ssl-cert ucf
...
```

4. Al tratar de activar el módulo de php, se comprueba que ya está activo:

```
# a2enmod
```

```
Your choices are: access_compat actions alias allowmethods asis
auth_basic auth_digest auth_form authn_anon authn_core authn_dbd
authn_dbm authn_file authn_socache authnz_fcgi authnz_ldap authz_core
authz_dbd authz_dbm authz_groupfile authz_host authz_owner authz_user
autoindex brotli buffer cache cache_disk cache_socache cern_meta cgi
cgid charset_lite data dav dav_fs dav_lock dbd deflate dialup dir
dump_io echo env expires ext_filter file_cache filter headers
heartbeat heartmonitor http2 ident imagemap include info
lbmethod_bybusyness lbmethod_byrequests lbmethod_bytraffic
lbmethod_heartbeat ldap log_debug log_forensic lua macro md mime
mime_magic mpm_event mpm_prefork mpm_worker negotiation php8.2 proxy
proxy_ajp proxy_balancer proxy_connect proxy_express proxy_fcgi
proxy_fdpass proxy_ftp proxy_hcheck proxy_html proxy_http proxy_http2
proxy_scgi proxy_uwsgi proxy_wstunnel ratelimit reflector remoteip
reqtimeout request rewrite sed session session_cookie session_crypto
session_dbd setenvif slotmem_plain slotmem_shm socache_dbm
socache_memcache socache_redis socache_shmcb speling ssl status
substitute suexec unique_id userdir usertrack vhost_alias xml2enc

Which module(s) do you want to enable (wildcards ok)?
php8.2

Considering dependency mpm_prefork for php8.2:
Considering conflict mpm_event for mpm_prefork:
Considering conflict mpm_worker for mpm_prefork:
Module mpm_prefork already enabled
Considering conflict php5 for php8.2:
Module php8.2 already enabled
```

5. Se crea un archivo php de prueba, al que se llamará phpinfo.php, en el directorio por defecto de publicación, /usr/local/apache2/htdocs, para verificar que el php está activo:

```
# nano /usr/local/apache2/htdocs/phpinfo.php
<?php
phpinfo();
?>
```

7. Usando Docker Compose

7.1. Arranque de la infraestructura

1. Script docker-compose.yml:

```
version: "3.8"
services:
  mysql:
    image: mysql
#    container_name: contiene_mysql
    hostname: servidor-mysql
    volumes:
      - volumen_mysql:/var/lib/mysql
    environment:
```

```
      MYSQL_ROOT_PASSWORD: "clave-de-root"
    networks:
      - red_aplicacion
    deploy:
      restart_policy:
        condition: on-failure
        max_attempts: 3
#        delay: 5s
#        window: 120s
#    ports:
#      - "3306"
#    working_dir: /var/lib/mysql

  apache:
    image: ubuntu/apache2
#    container_name: contiene_apache
    hostname: servidor-apache
    volumes:
      - volumen_apache:/var/www
    networks:
      - red_aplicacion
    deploy:
      replicas: 2
      restart_policy:
        condition: on-failure
#    ports:
#      - "80"
    working_dir: /var/www/html

volumes:
  volumen_mysql:
  volumen_apache:

networks:
  red_aplicacion:
    ipam:
      driver: default
      config:
        - subnet: "172.20.100.0/24"
#        - gateway: "172.20.100.1"
```

2. Se ejecuta el script:

```
$ docker-compose up
WARNING: The Docker Engine you're using is running in swarm mode.

Compose does not use swarm mode to deploy services to multiple nodes in a swarm. All
containers will be scheduled on the current node.

To deploy your application across the swarm, use `docker stack deploy`.

Creating network "practica-apache-mysql_red_aplicacion" with the default driver
Creating practica-apache-mysql_apache_1 ... done
Creating practica-apache-mysql_apache_2 ... done
```

```
Creating practica-apache-mysql_mysql_1  ... done
Attaching to practica-apache-mysql_mysql_1, practica-apache-mysql_apache_1, practica-apache-mysql_apache_2
apache_1  | AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.20.100.4. Set the 'ServerName' directive globally to suppress this message
apache_1  | AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.20.100.4. Set the 'ServerName' directive globally to suppress this message
apache_1  | [Sun Mar 10 01:44:56.061209 2024] [mpm_event:notice] [pid 23:tid 135221871310720] AH00489: Apache/2.4.57 (Ubuntu) configured -- resuming normal operations
apache_1  | [Sun Mar 10 01:44:56.061364 2024] [core:notice] [pid 23:tid 135221871310720] AH00094: Command line: '/usr/sbin/apache2 -D FOREGROUND'
mysql_1   | 2024-03-10 01:44:55+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.2.0-1.el8 started.
mysql_1   | 2024-03-10 01:44:56+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
mysql_1   | 2024-03-10 01:44:56+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.2.0-1.el8 started.
apache_2  | AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.20.100.3. Set the 'ServerName' directive globally to suppress this message
apache_2  | AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.20.100.3. Set the 'ServerName' directive globally to suppress this message
apache_2  | [Sun Mar 10 01:44:56.104184 2024] [mpm_event:notice] [pid 24:tid 136540785342336] AH00489: Apache/2.4.57 (Ubuntu) configured -- resuming normal operations
apache_2  | [Sun Mar 10 01:44:56.104327 2024] [core:notice] [pid 24:tid 136540785342336] AH00094: Command line: '/usr/sbin/apache2 -D FOREGROUND'
mysql_1   | '/var/lib/mysql/mysql.sock' -> '/var/run/mysqld/mysqld.sock'
mysql_1   | 2024-03-10T01:44:56.274353Z 0 [System] [MY-015015] [Server] MySQL Server - start.
mysql_1   | 2024-03-10T01:44:56.501538Z 0 [Warning] [MY-011068] [Server] The syntax '--skip-host-cache' is deprecated and will be removed in a future release. Please use SET GLOBAL host_cache_size=0 instead.
mysql_1   | 2024-03-10T01:44:56.502592Z 0 [System] [MY-010116] [Server] /usr/sbin/mysqld (mysqld 8.2.0) starting as process 1
mysql_1   | 2024-03-10T01:44:56.509708Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has started.
mysql_1   | 2024-03-10T01:44:58.190004Z 1 [System] [MY-013577] [InnoDB] InnoDB initialization has ended.
mysql_1   | 2024-03-10T01:44:58.984687Z 0 [Warning] [MY-010068] [Server] CA certificate ca.pem is self signed.
mysql_1   | 2024-03-10T01:44:58.984718Z 0 [System] [MY-013602] [Server] Channel mysql_main configured to support TLS. Encrypted connections are now supported for this channel.
mysql_1   | 2024-03-10T01:44:58.988062Z 0 [Warning] [MY-011810] [Server] Insecure configuration for --pid-file: Location '/var/run/mysqld' in the path is accessible to all OS users. Consider choosing a different directory.
mysql_1   | 2024-03-10T01:44:59.117885Z 0 [System] [MY-011323] [Server] X Plugin ready for connections. Bind-address: '::' port: 33060, socket: /var/run/mysqld/mysqlx.sock
mysql_1   | 2024-03-10T01:44:59.117979Z 0 [System] [MY-010931] [Server] /usr/sbin/mysqld: ready for connections. Version: '8.2.0'  socket: '/var/run/mysqld/mysqld.sock'  port: 3306  MySQL Community Server - GPL.
```

Se crea:

- Una red: **practica-apache-mysql_red_aplicacion**
- Dos volúmenes:

        - **practica-apache-mysql_volumen_mysql**
        - **practica-apache-mysql_volumen_apache**
    - Tres servicios (contendores):
        - **practica-apache-mysql_apache_1**
        - **practica-apache-mysql_apache_2**
        - **practica-apache-mysql_mysql_1**
    - Se vinculan:
        - practica-apache-mysql_apache_1
        - practica-apache-mysql_apache_2
        - practica-apache-mysql_mysql_1

3. Se comprueban los servicios (contenedores) creados:

```
$ docker-compose ps
Name                              Command                       State    Ports
-------------------------------------------------------------------------------------------
practica-apache-mysql_apache_1    apache2-foreground            Up       80/tcp
practica-apache-mysql_apache_2    apache2-foreground            Up       80/tcp
practica-apache-mysql_mysql_1     docker-entrypoint.sh mysqld   Up       3306/tcp, 33060/tcp
```

4. Se comprueba que se creó la red:

```
$ docker network ls
NETWORK ID     NAME                                    DRIVER    SCOPE
...
ab4d07ae979b   practica-apache-mysql_red_aplicacion   bridge    local
...
```

5. Se inspeccionan las características de la red:

```
$ docker network inspect practica-apache-mysql_red_aplicacion
[
    {
        "Name": "practica-apache-mysql_red_aplicacion",
        "Id":
"ab4d07ae979bfead89cf17e84b4781d2cd4b38737c03118ad55df77d1a19be54",
        "Created": "2024-03-10T02:10:28.237613227+01:00",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "172.20.100.0/24"
                }
            ]
        },
        "Internal": false,
```

```
        "Attachable": true,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "027c1c657d66a8949185103c911b19384cbd2658db77684cd82ecadc98c3c32c":
{
                "Name": "practica-apache-mysql_apache_1",
                "EndpointID":
"defd873319d9397d8d67f51c000762553680e982d4a7efd85b7b99aa0c17edc7",
                "MacAddress": "02:42:ac:14:64:04",
                "IPv4Address": "172.20.100.4/24",
                "IPv6Address": ""
            },
            "954eeba787f5acde64133eea534ffe57e69c4eddcf308fa74226cd3090274d8d":
                "Name": "practica-apache-mysql_apache_2",
                "EndpointID":
"c98cf0edfea24e9aec43cedc245d3bb1b36d40b9d22fb848104895f60e6ae55c",
                "MacAddress": "02:42:ac:14:64:03",
                "IPv4Address": "172.20.100.3/24",
                "IPv6Address": ""
            },
            "e08e4c0a490374d4d4d02154b917cf6ea65fa2255e52e01f856ac0d68de17ade":
{
                "Name": "practica-apache-mysql_mysql_1",
                "EndpointID":
"c1e2ed1e2b81f18d1d21393a588f6413546eaa3a5858185c91af787d2c73fdb9",
                "MacAddress": "02:42:ac:14:64:02",
                "IPv4Address": "172.20.100.2/24",
                "IPv6Address": ""
            }
        },
        "Options": {},
        "Labels": {
            "com.docker.compose.network": "red_aplicacion",
            "com.docker.compose.project": "practica-apache-mysql",
            "com.docker.compose.version": "1.29.2"
        }
    }
]
```

6. Se comprueba que en el anfitrión se ha creado una interfaz de red virtual que pertenece a la red. El anfitrión es la puerta de enlace de la red:

```
$ ip a
...
47: br-ab4d07ae979b: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group
default
    link/ether 02:42:08:19:3c:01 brd ff:ff:ff:ff:ff:ff
    inet 172.20.100.1/24 brd 172.20.100.255 scope global br-ab4d07ae979b
       valid_lft forever preferred_lft forever
    inet6 fe80::42:8ff:fe19:3c01/64 scope link
       valid_lft forever preferred_lft forever
```

```
...
```

7. Se comprueba que se crearon los volúmenes:

```
$ docker volume ls
DRIVER    VOLUME NAME
...
local     practica-apache-mysql_volumen_apache
local     practica-apache-mysql_volumen_mysql
...
```

8. Se inspeccionan las características de los volúmenes:

```
$ docker volume inspect practica-apache-mysql_volumen_apache
[
    {
        "CreatedAt": "2024-03-10T01:48:00+01:00",
        "Driver": "local",
        "Labels": {
            "com.docker.compose.project": "practica-apache-mysql",
            "com.docker.compose.version": "1.29.2",
            "com.docker.compose.volume": "volumen_apache"
        },
        "Mountpoint": "/var/lib/docker/volumes/practica-apache-mysql_volumen_apache/_data",
        "Name": "practica-apache-mysql_volumen_apache",
        "Options": null,
        "Scope": "local"
    }
]

$ docker volume inspect practica-apache-mysql_volumen_mysql
[
    {
        "CreatedAt": "2024-03-10T01:48:00+01:00",
        "Driver": "local",
        "Labels": {
            "com.docker.compose.project": "practica-apache-mysql",
            "com.docker.compose.version": "1.29.2",
            "com.docker.compose.volume": "volumen_mysql"
        },
        "Mountpoint": "/var/lib/docker/volumes/practica-apache-mysql_volumen_mysql/_data",
        "Name": "practica-apache-mysql_volumen_mysql",
        "Options": null,
        "Scope": "local"
    }
]
```

7.2. Configuración de los servidores

Aquí se siguen los mismos pasos que cuando se crearon los contenedores manualmente.

7.2.1. MySQL

Conf. de MySQL.

7.2.2. Apache

Conf. de Apache.

7.3. Detención del compuesto (compose)

1. Se detiene la ejecución del compuesto:

```
$ docker-compose down
Stopping practica-apache-mysql_apache_2 ... done
Stopping practica-apache-mysql_mysql_1  ... done
Stopping practica-apache-mysql_apache_1 ... done
Removing practica-apache-mysql_apache_2 ... done
Removing practica-apache-mysql_mysql_1  ... done
Removing practica-apache-mysql_apache_1 ... done
Removing network practica-apache-mysql_red_aplicacion
```

8. REFERENCIAS Y RECURSOS

Referencia/Recurso	URL
httpd – Official Image	https://hub.docker.com/_/httpd
How to Use the Apache httpd Docker Official Image	https://www.docker.com/blog/how-to-use-the-apache-httpd-docker-official-image/
php – Official Image	https://hub.docker.com/_/php/
mysql – Official Image	https://hub.docker.com/_/mysql
Apache and PHP on Docker	https://writing.pupius.co.uk/apache-and-php-on-docker-44faef716150
The EMP and DEPT tables for MySQL	https://justinsomnia.org/2009/04/the-emp-and-dept-tables-for-mysql/
dotnet-db-samples/schemas	https://github.com/oracle/dotnet-db-samples/blob/master/schemas/scott.sql
SELECT query with PDO	https://phpdelusions.net/pdo_examples/select
CRUD operation with PDO Database Connection	https://www.cloudways.com/blog/crud-with-php-data-objects/
PDO for Elegant PHP Database Access	https://www.slashnode.com/articles/architecture/2014-02-24-pdo-for-elegant-database-access
A Comprehensive Guide to the PHP PDO Library for Database Access	https://reintech.io/blog/a-comprehensive-guide-to-php-pdo-library-for-database-access
Connecting to MySQL	https://www.phptutorial.net/php-pdo/pdo-connecting-to-mysql/
Construcción de una imagen de Apache sobre Linux con Dockerfile	https://github.com/dhanugupta/docker-apache-ubuntu/blob/master/Dockerfile
Gestionar imágenes con Docker	https://atareao.es/tutorial/docker/gestionar-imagenes-con-docker/

9. Scripts de creación de tablas (DDL) e inserción de datos (DML)

```
mysql> CREATE DATABASE scott;
Query OK, 1 row affected (0,03 sec)

mysql> USE SCOTT;
Database changed


DROP TABLE dept;
CREATE TABLE dept (
    deptno decimal(2,0),
    dname varchar(14),
    loc varchar(13),
    CONSTRAINT `PK_dept` PRIMARY KEY(deptno)
);

INSERT INTO dept VALUES
  (10,'ACCOUNTING','NEW YORK'),
  (20,'RESEARCH','DALLAS'),
  (30,'SALES','CHICAGO'),
  (40,'OPERATIONS','BOSTON');

DROP TABLE emp;
CREATE TABLE emp (
    empno decimal(4,0),
    ename varchar(10),
    job varchar(9),
    mgr decimal(4,0),
    hiredate date,
    sal decimal(7,2),
    comm decimal(7,2),
    deptno decimal(2,0),
    CONSTRAINT `PK_emp` PRIMARY KEY (empno),
    CONSTRAINT `FK_emp_dept_deptno` FOREIGN KEY (deptno)
        REFERENCES dept (deptno),
    CONSTRAINT `FK_emp_emp_mgr` FOREIGN KEY (mgr)
        REFERENCES emp (empno)
);


INSERT INTO emp VALUES
  ('7839','KING','PRESIDENT',NULL,'1981-11-17','5000.00',NULL,'10'),
  ('7566','JONES','MANAGER','7839','1981-04-02','2975.00',NULL,'20'),
  ('7698','BLAKE','MANAGER','7839','1981-05-01','2850.00',NULL,'30'),
  ('7782','CLARK','MANAGER','7839','1981-06-09','2450.00',NULL,'10'),
  ('7788','SCOTT','ANALYST','7566','1982-12-09','3000.00',NULL,'20'),
  ('7902','FORD','ANALYST','7566','1981-12-03','3000.00',NULL,'20'),
  ('7499','ALLEN','SALESMAN','7698','1981-02-20','1600.00','300.00','30'),
  ('7521','WARD','SALESMAN','7698','1981-02-22','1250.00','500.00','30'),
  ('7654','MARTIN','SALESMAN','7698','1981-09-28','1250.00','1400.00','30'),
  ('7844','TURNER','SALESMAN','7698','1981-09-08','1500.00','0.00','30'),
  ('7900','JAMES','CLERK','7698','1981-12-03','950.00',NULL,'30'),
  ('7934','MILLER','CLERK','7782','1982-01-23','1300.00',NULL,'10'),
  ('7876','ADAMS','CLERK','7788','1983-01-12','1100.00',NULL,'20'),
  ('7369','SMITH','CLERK','7902','1980-12-17','800.00',NULL,'20');


DROP TABLE bonus;
CREATE TABLE bonus (
```

```
    ename varchar(10),
    job varchar(9),
    sal decimal(7,2),
    comm decimal(7,2)
);

DROP TABLE salgrade;
CREATE TABLE salgrade (
    grade decimal(2,0),
    losal decimal(7,2),
    hisal decimal(7,2),
    CONSTRAINT `PK_salgrade` PRIMARY KEY (grade)
);
INSERT INTO salgrade VALUES
  (1,700,1200),
  (2,1201,1400),
  (3,1401,2000),
  (4,2001,3000),
  (5,3001,9999);

COMMIT;
```

10. SCRIPT EN PHP PARA ACCEDER A LA BD EXISTENTE EN MYSQL

Con PDO:

```
<?php
echo "<table style='border: solid 1px black;'>" . PHP_EOL;
echo "<tr><th>empno</th><th>ename</th><th>job</th><th>mgr</th>"
    . "<th>hiredate</th><th>sal</th><th>comm</th><th>deptno</th></tr>" . PHP_EOL;

class TableRows extends RecursiveIteratorIterator {
  function __construct($it) {
    parent::__construct($it, self::LEAVES_ONLY);
  }
  function current() {
    return "<td style='width:150px;border:1px solid black;'>" . parent::current().
"</td>" . PHP_EOL;
  }
  function beginChildren() {
    echo "<tr>" . PHP_EOL;
  }
  function endChildren() {
    echo "</tr>" . PHP_EOL;
  }
}

// parámetros de la BD
$servername = "172.17.0.2";
$username = "root";
$password = "clave-de-root";
$dbname = "scott";

try {
  $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);
  $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  $stmt = "SELECT * FROM emp ORDER BY deptno";
  $stmt->execute();
  // set the resulting array to associative
  $result = $stmt->setFetchMode(PDO::FETCH_ASSOC);
  foreach (new TableRows(new RecursiveArrayIterator($stmt->fetchAll())) as $k=>$v) {
    echo $v;
  }
} catch (PDOException $e) {
  echo "Error: " . $e->getMessage();
}
$conn = null;
echo "</table>" .PHP_EOL;
?>
```

Con mysqli funcional

```
<?php
```

```
$servername = "172.17.0.2";
$username = "root";
$password = "clave-de-root";
$dbname = "scott";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);

// Check connection
if (!$conn) {
  die("Connection failed: " . mysqli_connect_error());
}

$sql = "SELECT * FROM emp ORDER BY deptno";
$result = mysqli_query($conn, $sql);

if (mysqli_num_rows($result) > 0) {
  // output data of each row
  while ($row = mysqli_fetch_assoc($result)) {
    echo "empno: " . $row["empno"] . " - ename: " . $row["ename"]
            . " - job: " . $row["job"]. " - mgr: " . $row["mgr"]
            . " - hiredate: " . $row["hiredate"] . " - sal: " . $row["sal"]
            . " - comm: " . $row["comm"] . " - deptno: " . $row["deptno"]
            . "<br />" . PHP_EOL;
  }
} else {
  echo "0 results";
}

mysqli_close($conn);
?>
```

Con mysqli orientado a objeto

```
<?php
$servername = "172.17.0.2";
$username = "root";
$password = "clave-de-root";
$dbname = "scott";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT * FROM emp ORDER BY deptno";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
  // output data of each row
  while($row = $result->fetch_assoc()) {
```

```
    echo "empno: " . $row["empno"] . " - ename: " . $row["ename"]
            . " - job: " . $row["job"]. " - mgr: " . $row["mgr"]
            . " - hiredate: " . $row["hiredate"] . " - sal: " . $row["sal"]
            . " - comm: " . $row["comm"] . " - deptno: " . $row["deptno"]
            . "<br />" . PHP_EOL;
  }
} else {
  echo "0 results";
}
$conn->close();
?>
```

Comandos de Docker

Desarrollada por **José Ramón Rodríguez Sánchez**

Generales

COMANDO	USO	ATAJO
`docker login`	Autenticación en un registro	
`docker logout`	Desconexión de un registro	
`docker search`	Búsqueda de una imagen en Docker Hub	
`docker version`	Información sobre la versión de Docker	
`docker info`	Muestra información de todo el sistema	

docker login

Autenticación en un registro.

Ej.

```
$ docker login
Authenticating with existing credentials...
WARNING! Your password will be stored unencrypted in
/home/alumno/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-
store

Login Succeeded
```

docker logout

Desconexión de un registro.

Ej.

```
$ docker logout
Removing login credentials for https://index.docker.io/v1/
```

docker search

Búsqueda de una imagen en Docker Hub.

Ej. Búsqueda de una imagen de Oracle.

```
$ docker search oracle
NAME                              DESCRIPTION                                     STARS OFFICIAL
```

```
oraclelinux                      Official Docker builds of Oracle Linux.       1047  [OK]
kasmweb/oracle-8-desktop         Oracle Linux 8 desktop for Kasm Workspaces    3
dockette/adminer                 My most tiniest (10mb) Adminer image with su… 21
dockette/mvn                     My MVN 3 based on Oracle Java Dockerfile      4
redislabs/redis-connect-oracle   Redis Connect Oracle Connector for Continuou… 1
dockette/jdk8                    My Oracle Java 8 Dockerfile                   5
sismics/debian-java              Debian Jessy + Oracle JDK                     1
oracleinanutshell/oracle-xe-11g                                                286
oraclecoherence/coherence-ce     Coherence Community Edition                   5
oracledb19c/oracle.19.3.0-ee                                                   35
oracledemo1/hello-world          Test docker build from github                 0
gvenzl/oracle-xe                 Oracle Database XE (21c, 18c, 11g) for every… 292
iamseth/oracledb_exporter        A Prometheus exporter for Oracle modeled aft… 7
```

docker version

Muestra la versión de Docker instalada en la máquina.

Ej.

```
$ docker version
Client: Docker Engine - Community
 Cloud integration: v1.0.35+desktop.5
 Version:           25.0.3
 API version:       1.44
 Go version:        go1.21.6
 Git commit:        4debf41
 Built:             Tue Feb  6 21:13:09 2024
 OS/Arch:           linux/amd64
 Context:           default

Server: Docker Engine - Community
 Engine:
  Version:          25.0.3
  API version:      1.44 (minimum version 1.24)
  Go version:       go1.21.6
  Git commit:       f417435
  Built:            Tue Feb  6 21:13:09 2024
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.6.28
  GitCommit:        ae07eda36dd25f8a1b98dfbf587313b99c0190bb
 runc:
  Version:          1.1.12
  GitCommit:        v1.1.12-0-g51d5e94
 docker-init:
  Version:          0.19.0
  GitCommit:        de40ad0
```

docker info

Muestra información general del sistema.

Ej.

```
$ docker info
Client: Docker Engine - Community
 Version:    25.0.3
 Context:    default
 Debug Mode: false
 Plugins:
  buildx: Docker Buildx (Docker Inc.)
    Version:  v0.11.2-desktop.5
    Path:     /usr/lib/docker/cli-plugins/docker-buildx
  compose: Docker Compose (Docker Inc.)
    Version:  v2.23.0-desktop.1
    Path:     /usr/lib/docker/cli-plugins/docker-compose
  dev: Docker Dev Environments (Docker Inc.)
    Version:  v0.1.0
    Path:     /usr/lib/docker/cli-plugins/docker-dev
  extension: Manages Docker extensions (Docker Inc.)
    Version:  v0.2.20
    Path:     /usr/lib/docker/cli-plugins/docker-extension
  init: Creates Docker-related starter files for your project
(Docker Inc.)
    Version:  v0.1.0-beta.9
    Path:     /usr/lib/docker/cli-plugins/docker-init
  sbom: View the packaged-based Software Bill Of Materials (SBOM)
for an image (Anchore Inc.)
    Version:  0.6.0
    Path:     /usr/lib/docker/cli-plugins/docker-sbom
  scan: Docker Scan (Docker Inc.)
    Version:  v0.26.0
    Path:     /usr/lib/docker/cli-plugins/docker-scan
  scout: Docker Scout (Docker Inc.)
    Version:  v1.0.9
    Path:     /usr/lib/docker/cli-plugins/docker-scout


Server:
 Containers: 45
  Running: 3
```

```
 Paused: 0
 Stopped: 42
Images: 31
Server Version: 25.0.3
Storage Driver: overlay2
 Backing Filesystem: extfs
 Supports d_type: true
 Using metacopy: false
 Native Overlay Diff: true
 userxattr: false
Logging Driver: json-file
Cgroup Driver: systemd
Cgroup Version: 2
Plugins:
 Volume: local
 Network: bridge host ipvlan macvlan null overlay
 Log: awslogs fluentd gcplogs gelf journald json-file local splunk
syslog
Swarm: inactive
Runtimes: io.containerd.runc.v2 runc
Default Runtime: runc
Init Binary: docker-init
containerd version: ae07eda36dd25f8a1b98dfbf587313b99c0190bb
runc version: v1.1.12-0-g51d5e94
init version: de40ad0
Security options:
 apparmor
 seccomp
  Profile: builtin
 cgroupns
Kernel Version: 6.5.0-17-generic
Operating System: Ubuntu 22.04.3 LTS
OSType: linux
Architecture: x86_64
CPUs: 4
Total Memory: 5.653GiB
Name: cliente
```

```
 ID: 9608cecb-987d-4da4-8fd3-b4a8f64bba09
 Docker Root Dir: /var/lib/docker
 Debug Mode: false
 Experimental: false
 Insecure Registries:
  127.0.0.0/8
 Live Restore Enabled: false
```

De sistema

COMANDO	USO	ATAJO
`docker system df`	Muestra el uso de disco duro que hace de Docker	
`docker system events`	Obtiene eventos del servidor en tiempo real	
`docker system info`	Muestra información de todo el sistema	
`docker system prune`	Elimina elementos no usados (contenedores detenidos, redes no utilizadas por al menos un contenedor, imágenes colgantes (*dangling images*) caché de compilación no utilizada)	

docker system df

Muestra el uso de disco duro que hace de Docker.

Ej.

```
$ docker system df
TYPE            TOTAL     ACTIVE    SIZE      RECLAIMABLE
Images          31        18        17.89GB   3.905GB (21%)
Containers      45        0         4.691GB   4.691GB (100%)
Local Volumes   37        37        3.073GB   0B (0%)
Build Cache     97        0         16.97GB   16.97GB
```

docker system events

Obtiene eventos del servidor en tiempo real

docker system info

Muestra información de todo el sistema.

Es equivalente a **`docker info`**.

docker system prune

Elimina elementos no usados (contenedores detenidos, redes no utilizadas por al menos un contenedor, imágenes colgantes (*dangling images*) caché de compilación no utilizada).

Ej.

```
$ docker system prune
WARNING! This will remove:
  - all stopped containers
  - all networks not used by at least one container
  - all dangling images
  - unused build cache
Are you sure you want to continue? [y/N]
```

Sobre imágenes

COMANDO	USO	ATAJO
`docker image build`	Crea una imagen a partir del script existente en un Dockerfile	`docker build`
`docker image history`	Mostrar el historial de una imagen	`docker history`
`docker image import`	Importa el contenido de un archivo empaquetado (*tarball*) para crear una imagen del sistema de archivos	`docker import`
`docker image inspect`	Muestra información de una imagen (su nombre, variables de entorno definidas, directorio de trabajo, comando de ejecución…)	
`docker image load`	Carga una imagen desde un archivo empaquetado (tarball) o desde la entrada estándar (STDIN)	`docker load`
`docker image ls`	Lista las imágenes existentes en el equipo	`docker images`
`docker image prune`	Elimina todas las imágenes que no están en uso	
`docker image pull`	Descarga una imagen de un registro	`docker pull`
`docker image push`	Sube una imagen a un registro	`docker push`
`docker image rm`	Elimina una imagen	`docker rmi`
`docker image save`	Guarda una imagen en un archivo empaquetado (*tarball*) enviado a la salida estándar (STDOUT) de forma predeterminada	`docker save`
`docker image tag`	Crea una nueva imagen que referencia a una imagen ya existente	`docker tag`

docker image build

Crea una imagen a partir del script existente en un Dockerfile.

Su alias es **`docker build`**.

docker image history

Muestra el historial de una imagen.

Su alias es **`docker history`**.

Ej. Mostrar el historial de una imagen de nombre **httpd**:

```
$ docker history httpd
IMAGE          CREATED        CREATED BY                                      SIZE
COMMENT
e499c02ff073   3 months ago   CMD ["httpd-foreground"]                        0B
buildkit.dockerfile.v0
<missing>      3 months ago   EXPOSE map[80/tcp:{}]                           0B
buildkit.dockerfile.v0
<missing>      3 months ago   COPY httpd-foreground /usr/local/bin/ # buil…   138B
buildkit.dockerfile.v0
<missing>      3 months ago   STOPSIGNAL SIGWINCH                             0B
buildkit.dockerfile.v0
<missing>      3 months ago   RUN /bin/sh -c set -eux;   savedAptMark="$(a…   89.1MB
buildkit.dockerfile.v0
<missing>      3 months ago   ENV HTTPD_PATCHES=                              0B
buildkit.dockerfile.v0
<missing>      3 months ago   ENV HTTPD_SHA256=fa16d72a078210a54c47dd5bef2…   0B
buildkit.dockerfile.v0
<missing>      3 months ago   ENV HTTPD_VERSION=2.4.58                        0B
buildkit.dockerfile.v0
<missing>      3 months ago   RUN /bin/sh -c set -eux;  apt-get update;  a…   10.3MB
buildkit.dockerfile.v0
<missing>      3 months ago   WORKDIR /usr/local/apache2                      0B
buildkit.dockerfile.v0
<missing>      3 months ago   RUN /bin/sh -c mkdir -p "$HTTPD_PREFIX"  && …   0B
buildkit.dockerfile.v0
<missing>      3 months ago   ENV PATH=/usr/local/apache2/bin:/usr/local/s…   0B
buildkit.dockerfile.v0
<missing>      3 months ago   ENV HTTPD_PREFIX=/usr/local/apache2              0B
buildkit.dockerfile.v0
<missing>      3 months ago   /bin/sh -c #(nop)  CMD ["bash"]                 0B
<missing>      3 months ago   /bin/sh -c #(nop) ADD file:d261a6f6921593f1e…   74.8MB
```

docker image import

Importa el contenido de un archivo empaquetado (*tarball*) para crear una imagen del sistema de archivos.

Su alias es **`docker import`**.

Ej. Creación de una imagen no etiquetada:

```
docker import https://example.com/exampleimage.tgz
```

Ej. Importación con un pipe y STDIN.

```
cat exampleimage.tgz | docker import - exampleimagelocal:new
```

docker image inspect

Muestra información de una imagen (su nombre, variables de entorno definidas, directorio de trabajo, comando de ejecución…).

Ej. Inspeccionar la normación de una imagen de nombre tomcat:

```
$ docker image inspect tomcat
[
    {
        "Id":
"sha256:fa51ad4dee227446060be37223b75f4e6a351508d90d1f08bcb7856108c1b59c"
,
        "RepoTags": [
            "tomcat:latest"
        ],
        "RepoDigests": [
"tomcat@sha256:86ada4acd78ca43972ad9f31e09ef1da00ca181cc0a204e7c31dc566d1
e5cd60"
        ],
        "Parent": "",
...
```

docker image load

Carga una imagen desde un archivo empaquetado (*tarball*) o desde la entrada estándar (STDIN).

Su alias es **`docker load`**.

Ej. Carga de una imagen desde la estrada estándar (STDIN):

```
$ docker load < busybox.tar.gz
```

Ej. Carga de una imagen desde un archivo:

```
$ docker load --input debian.tar
```

docker image ls

Lista las imágenes existentes en el equipo.

Su alias es **`docker images`**.

docker image prune

Elimina todas las imágenes que no están en uso.

docker image pull

Descarga una imagen de un registro.

Su alias es **`docker pull`**.

docker image push

Sube una imagen a un registro. Es preciso autenticarse en el registro antes de subir la imagen, usando **`docker login`**.

Su alias es **`docker push`**.

docker image rm

Elimina una imagen.

Su alias es **`docker rmi`**.

docker image save

Guarda una imagen en un archivo empaquetado (*tarball*) enviado a la salida estándar (STDOUT) de forma predeterminada.

Su alias es **`docker save`**.

Ej. Creación una copia de seguridad que posteriormente se pueda cargar con **`docker load`**:

```
$ docker save busybox > busybox.tar

$ ls -sh busybox.tar
2.7M busybox.tar

$ docker save --output busybox.tar busybox

$ ls -sh busybox.tar
2.7M busybox.tar

$ docker save -o fedora-all.tar fedora

$ docker save -o fedora-latest.tar fedora:latest
```

docker image tag

Crea una nueva imagen que referencia a una imagen ya existente.

Su alias es **`docker tag`**.

Ej. Etiquetado de una imagen local **httpd** como ubuntu/httpd con la etiqueta v3.0:

```
$ docker tag httpd ubuntu/httpd:v3.0
```

Sobre contenedores

COMANDO	USO	ATAJO

`docker container attach`	Asocia los flujos locales de entrada, salida y error estándar a un contenedor en ejecución	`docker attach`
`docker container commit`	Crea una nueva imagen a partir del contenido de un contendor	`docker commit`
`docker container cp`	Copia archivos/carpetas entre el sistema de ficheros de un contenedor y el sistema de ficheros del anfitrión	`docker cp`
`docker container create`	Crea un contenedor	`docker create`
`docker container diff`	Lista los archivos y directorios modificados en el sistema de archivos de un contenedor desde que se creó	`docker diff`
`docker container exec`	Ejecuta un comando dentro de un contenedor	`docker diff`
`docker container export`	Exporta el sistema de ficheros de un contendor a un empaquetado tar	`docker export`
`docker container inspect`	Muestra la información de un contenedor (nombre, imagen que se corre dentro, red a la que pertenece…)	`docker inspect`
`docker container kill`	Fuerza la detención de un contenedor en ejecución	`docker kill`
`docker container ls`	Lista los contenedores existentes en el equipo anfitrión	`docker ps`
`docker container logs`	Muestra los logs de un contenedor	`docker logs`
`docker container pause`	Pausa todos los procesos dentro de un contendor	`docker pause`
`docker container port`	Muestra los puertos mapeados en un contendor	`docker port`
`docker container prune`	Elimina todos los contendores detenidos	
`docker container rename`	Cambia el nombre de un contenedor.	`docker rename`
`docker container restart`	Reinicia un contenedor	`docker restart`
`docker container rm`	Elimina un contenedor que se encuentre detenido	`docker rm`
`docker container run`	Crea y arranca un contenedor con una imagen asociada, descargándola si no lo estaba ya	`docker run`
`docker container start`	Arranca un contenedor detenido	`docker start`
`docker container stats`	Muestra estadísticas en vivo del uso de recursos de un contenedor	
`docker container stop`	Detiene un contenedor en ejecución	`docker stop`
`docker container top`	Muestra los procesos de un contenedor en ejecución.	`docker top`
`docker container unpause`	Reactiva los procesos pausados en un contendor	`docker unpause`
`docker container update`	Actualiza la configuración de un contenedor	`docker update`
`docker container wait`	Bloquea hasta que un contendor se detenga, imprimiendo a continuación sus códigos de salida	`docker wait`

docker container attach

Asocia los flujos locales de entrada, salida y error estándar a un contenedor en ejecución

Su alias es **`docker attach`**.

docker container commit

Crea una nueva imagen a partir del contenido de un contendor.

Su alias es **`docker commit`**.

docker container cp

Copiar archivos/carpetas entre el sistema de ficheros de un contenedor y el sistema de ficheros del anfitrión.

Su alias es **`docker cp`**.

Ej. Copia del archivo /etc/passwd, existente en el sistema de ficheros del anfitrión, en el directorio /work del sistema de ficheros de un contenedor de nombre **contiene_bind**:

```
docker cp /etc/passwd contiene_bind:/work
```

Ej. Copia el contenido del directorio **/var/log**, existente en el sistema de ficheros del contendor **contiene_oracle**, a la ubicación **/tmp/app_logs**,del sistema de ficheros del anfitrión:

```
docker cp contiene_oracle:/var/log/ /tmp/app_logs
```

docker container create

Crea un contenedor.

Su alias es **`docker create`**.

Ej. Creación de un contenedor con terminal interactiva, de nombre **contiene_apache**:

```
docker container create -it --name contiene_apache
```

docker container diff

Lista los archivos/directorios modificados en el sistema de archivos de un contenedor desde que se creó. Se realiza un seguimiento de tres tipos diferentes de cambios:

A – El archivo/directorio fue **a**ñadido (***a****dded*)

D – El archivo/directorio fue eliminado (***d****eleted*)

C – El archivo/directorio fue modificado (***c****hanged*)

Su alias es **`docker diff`**.

Ej. Diferencias existentes en un contenedor llamado **contiene_mysql**:

```
$ docker diff contiene_mysql
C /run
C /run/mysqld
A /run/mysqld/mysqld.pid
A /run/mysqld/mysqld.sock
```

```
A /run/mysqld/mysqld.sock.lock
A /run/mysqld/mysqlx.sock
A /run/mysqld/mysqlx.sock.lock
```

docker container exec

Ejecuta un comando dentro de un contenedor.

Su alias es **`docker exec`**.

Ej. Listar los archivos del directorio raíz del sistema de ficheros del contendor de nombre **contiene_ubuntu**:

```
docker container exec contiene_ubuntu ls /
```

docker container export

Exporta el sistema de ficheros de un contendor a un empaquetado tar.

Su alias es **`docker export`**.

docker container inspect

Muestra la información de un contenedor (nombre, imagen que se corre dentro, red a la que pertenece…).

Su alias es **`docker inspect`**.

docker container kill

Fuerza la detención (“mata”) de un contenedor en ejecución.

El proceso principal dentro del contenedor recibe por defecto la señal SIGKILL, o la señal que se especifica con la opción --signal, que puede terminar la ejecución del proceso de forma abrupta.

Su alias es **`docker kill`**.

docker container logs

Muestra los logs de un contenedor.

Su alias es **`docker logs`**.

Ej. Mostrar los logs de un contendor de nombre **contiene_apache**:

```
$ docker logs contenedor_apache
AH00558: httpd: Could not reliably determine the server's fully
qualified domain name, using 172.17.0.2. Set the 'ServerName'
directive globally to suppress this message
AH00558: httpd: Could not reliably determine the server's fully
qualified domain name, using 172.17.0.2. Set the 'ServerName'
directive globally to suppress this message
```

```
[Thu Jan 11 09:04:05.076662 2024] [mpm_event:notice] [pid 1:tid
139788029007744] AH00489: Apache/2.4.58 (Unix) configured --
resuming normal operations
[Thu Jan 11 09:04:05.158482 2024] [core:notice] [pid 1:tid
139788029007744] AH00094: Command line: 'httpd -D FOREGROUND'
[Thu Jan 11 09:16:44.369217 2024] [mpm_event:notice] [pid 1:tid
139788029007744] AH00492: caught SIGWINCH, shutting down gracefully
AH00558: httpd: Could not reliably determine the server's fully
qualified domain name, using 172.17.0.3. Set the 'ServerName'
directive globally to suppress this message
AH00558: httpd: Could not reliably determine the server's fully
qualified domain name, using 172.17.0.3. Set the 'ServerName'
directive globally to suppress this message
[Tue Feb 13 19:09:05.443476 2024] [mpm_event:notice] [pid 1:tid
140555691779968] AH00489: Apache/2.4.58 (Unix) configured --
resuming normal operations
[Tue Feb 13 19:09:05.468990 2024] [core:notice] [pid 1:tid
140555691779968] AH00094: Command line: 'httpd -D FOREGROUND'
```

docker container ls

Lista los contenedores existentes en el equipo anfitrión.

Su alias es **`docker ps`**.

docker container pause

Pausa todos los procesos dentro de un contendor.

Su alias es **`docker pause`**.

docker container port

Muestra los puertos mapeados en un contendor.

Su alias es **`docker port`**.

Ej. Muestra los puertos mapeados de un contenedor de nombre cassandra_1:

```
$ docker container port cassandra_1
9042/tcp -> 0.0.0.0:9042
9042/tcp -> [::]:9042
```

docker container prune

Elimina todos los contendores detenidos.

Su alias es **`docker stop`**.

docker container rename

Cambia el nombre de un contenedor.

Su alias es **`docker rename`**.

docker container restart

Reinicia un contenedor.

Su alias es **`docker restart`**.

docker container rm

Elimina un contenedor que se encuentre detenido.

Su alias es **`docker rm`**.

docker container run

Crea y arranca un contenedor con una imagen asociada, descargándola si no lo estaba ya.

Su alias es **`docker run`**.

Ej. Se crea un contendor de nombre **contiene_apache**, se vincula el puerto **8888** del sistema anfitrión con el puerto **80** del contenedor, se asocia al contenedor la imagen **apache** y se arranca el contenedor:

```
docker container run --name contiene_apache -p 8888:80 apache
```

docker container start

Arranca un contenedor detenido.

Su alias es **`docker start`**.

docker container stats

Muestra estadísticas en vivo del uso de recursos de un contenedor

docker container stop

Detiene un contenedor en ejecución.

Su alias es **`docker stop`**.

docker container top

Muestra los procesos de un contenedor en ejecución.

Su alias es **`top`**.

Ej. Muestra los procesos de un contendor en ejecución de nombre contiene_apache:

```
$ docker top contiene_apache
```

```
UID                 PID                 PPID                C
STIME               TTY                 TIME                CMD
root                5678                5658                0
20:09               ?                   00:00:00            httpd -DFOREGROUND
www-data            5700                5678                0
20:09               ?                   00:00:00            httpd -DFOREGROUND
www-data            5701                5678                0
20:09               ?                   00:00:00            httpd -DFOREGROUND
www-data            5702                5678                0
20:09               ?                   00:00:00            httpd -DFOREGROUND
```

docker container unpause

Reactiva los procesos pausados en un contendor.

Su alias es **`unpause`**.

docker container update

Actualiza la configuración de un contenedor.

Su alias es **`docker update`**.

Ej. Actualización de un contenedor, de nombre **contiene_ubuntu**, con recursos compartidos de CPU y memoria:

```
$ docker update --cpu-shares 512 -m 300M contiene_ubuntu
```

docker container wait

Bloquea hasta que un contendor se detenga, imprimiendo a continuación sus códigos de salida.

Su alias es **`wait`**.

Ej. Se crea y arranca en segundo plano un contendor, de nombre **contiene_ubuntu**, asociado a una imagen de **unbuntu**. Para que el contenedor se mantenga en ejecución, se ejecuta el comando **bash**:

```
$ docker run -dit --name=contiene_ubuntu ubuntu bash
28c532c08cfe2b6281cbd83e7e4687f75b42c75948e0128c3d4806059716ca72

$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED
STATUS          PORTS                NAMES
28c532c08cfe   ubuntu    "bash"                   6 seconds ago   Up
5 seconds                          contiene_ubuntu
...
```

Se ejecuta **`docker wait`**, que debería bloquearse hasta que se cierre el contenedor.

```
$ docker wait contiene_ubuntu
<Se bloquea la terminal>
```

En otra terminal, se detendrá el primer contenedor. El comando **`docker wait`** ejecutado en la primera terminal, que quedó a la espera bloqueando la primera terminal, termina y devuelve el código de salida.

```
$ docker stop contiene_ubuntu
contiene_ubuntu
```

La primera terminal se desbloquea, devolviendo el código de salida.

```
$ docker wait contiene_ubuntu
137
```

Sobre volúmenes

COMANDO	USO	ATAJO
`docker volumen create`	Crea un volumen	
`docker volumen inspect`	Muestra información detallada sobre un volumen (su nombre, punto de montaje, driver…)	
`docker volumen ls`	Listado de volúmenes	
`docker volumen prune`	Elimina volúmenes (por defecto anónimos) que no estén en uso.	
`docker volumen rm`	Elimina un volumen que no esté siendo usado por ningún contenedor	

docker volume create

Crea un volumen.

Ej. Creación de un volumen, de nombre **voluminoso**, y arranque de un contenedor que lo usa:

```
$ docker volume create voluminoso
voluminoso

$ docker run -d -v voluminoso:/world busybox ls /world
Unable to find image 'busybox:latest' locally
latest: Pulling from library/busybox
9ad63333ebc9: Pull complete
Digest:
sha256:6d9ac9237a84afe1516540f40a0fafdc86859b2141954b4d643af7066d598b74
Status: Downloaded newer image for busybox:latest
66ca1a149ac021b12986fb7880185c03a142141d89a6137aaa0ab1af7ec74675
```

docker volume inspect

Muestra información detallada sobre un volumen (su nombre, punto de montaje, driver…).

Ej. Inspección de la información sobre un volumen de nombre **voluminoso**:

```
$ docker volume inspect voluminoso
```

```
[
    {
        "CreatedAt": "2024-02-09T19:12:45+01:00",
        "Driver": "local",
        "Labels": null,
        "Mountpoint": "/var/lib/docker/volumes/voluminoso/_data",
        "Name": "voluminoso",
        "Options": null,
        "Scope": "local"
    }
]
```

docker volume ls

Listado de volúmenes.

docker volume prune

Elimina volúmenes (por defecto anónimos) que no estén en uso.

Ej. Eliminación de volúmenes locales anónimos no en uso por algún contenedor:

```
$ docker volume prune
WARNING! This will remove anonymous local volumes not used by at
least one container.
Are you sure you want to continue? [y/N] y
Deleted Volumes:
888053e9ac4529c245aaac2eecd8a40dec7061f36e8f99331174947d7ff0e482
ed538a01f52b525820a5553c0b310ca39c62b086b14abad90246a47381079cde
c37c2ef786f4bcd68adb8a08c26c23bf1f0f0d3366fe10b3f14ed3ff216213e2
3773100e2c932ced085db6b52623d07256b9dd531364426ac904a306cc1f8d3c
8ee9fc45ce219a5df907cfd76f7c9142222eab1200678f318b221aaaac71cb85
c455521b962fe7d6bdf377346446730e6ac6a16318b9442db189b145fac62549
25e8f8d60081c988df1df3cf47bd9dc678d7ef60a8857048ee1fbe0023ae6370
87fbb28616a82b21876da9dd58e25534c5ef044fcf2bc983e732491f94744b7e
8ff330e8389626eed1673ab2ae221ce71ad82c0b7c3b7efb33e31d5f3cab2e23
ebc1f6564a977320be21af756e5cf8f1f553967777bdd32ec291d75be1346234
Total reclaimed space: 5.681GB
```

docker volume rm

Eliminación de un volumen que no esté en uso.

Sobre redes

COMANDO	USO	ATAJO
`docker network connect`	Conecta un contenedor a una red	
`docker network create`	Crea una red	
`docker network disconnect`	Desconecta un contenedor en ejecución de una red	
`docker network inspect`	Muestra información detallada de una red (su nombre, dirección, puerta de enlace, driver…)	
`docker network ls`	Lista las redes existentes	

`docker network prune`	Elimina de todas las redes no usadas	
`docker network rm`	Elimina una red	

docker network connect

Conecta un contenedor a una red. Una vez conectado, el contenedor puede comunicarse con otros contenedores de la misma red.

Ej. Conexión de un contenedor en ejecución, de nombre **contiene_mongo_4**, a una red llamada **red-cluster-mongo**:

```
$ docker network connect red-cluster-mongo contiene_mongo_4
```

Ej. Arranque de un contendor haciéndolo formar parte de una red llamada **red-cluster-mongo**:

```
$ docker run -itd –network=red-cluster-mongo mongo
```

docker network create

Crea una red.

docker network disconnect

Desconecta un contenedor en ejecución a una red.

Ej. Conexión de un contenedor en ejecución, de nombre **contiene_mongo_4**, a una red llamada **red_cluster_mongo**:

```
$ docker network connect red_cluster_mongo contiene_mongo_4
```

Ej. Arranque de un contendor haciéndolo formar parte de una red llamada **red_cluster_mongo**:

```
$ docker run -itd –network=red_cluster_mongo mongo
```

docker network inspect

Muestra información detallada de una red (su nombre, dirección, puerta de enlace, driver…).

Ej. Información de la red de nombre **red_cluster_mongo**:

```
$ docker network inspect red_cluster_mongo
[
    {
        "Name": "red_cluster_mongo",
        "Id":
"6d955b074f1c4712025d2cebcfcd3bac673e1eaf6fe3031d9dea1aa14fd7beb6",
        "Created": "2024-02-13T22:50:37.0518398+01:00",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
```

```
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "172.20.0.0/16",
                    "Gateway": "172.20.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {},
        "Options": {},
        "Labels": {}
    }
]
```

docker network ls

Lista las redes existentes.

docker network prune

Elimina las redes no usadas.

Ej. Eliminar las redes no usadas:

```
$ docker network prune
WARNING! This will remove all custom networks not used by at least
one container.
Are you sure you want to continue? [y/N] y
Deleted Networks:
minikube
mongo-red-cluster
la_red
sharded-mongodb-docker-main_internalnetwork
```

docker network rm

Elimina una red.

CRÉDITOS

El contenido es este cuaderno de prácticas ha sigo generado en el transcurso del seminario “Virtualización con contendores en la nube” desarrollador en el IES Clara del Rey entre los meses de enero y mayo de 2024.

Cada practica ha sido desarrollada por alguno de los seis participantes.

Lista de prácticas / autores

PRÁCTICA	AUTOR
Uso de imágenes y contenedores	Guadalupe Bermejo Sánchez
Ejecución de Linux en Windows sin y con contenedores	Concepción Fernández Fernández
Redes y comunicación entre contenedores	Mercedes Rodríguez Villafáfila
Microservicio web creado a partir de DockerFile	Mercedes Rodríguez Villafáfila
Microservicios ACI Azure Cloud	Mercedes Rodríguez Villafáfila
Configuración de Oracle Database XE en un contenedor Docker	Carlos Moreno Martínez
Instalación de WSL	Camino Pardo de Vega
Conectar a una máquina Linux desde el escritorio remoto de Windows	Camino Pardo de Vega
Instalación de Portainer	Guadalupe Bermejo Sánchez
Mini-aplicación en un entorno “contenedorizado” con Docker Compose	José Ramón Rodríguez Sánchez
Comandos de Docker	José Ramón Rodríguez Sánchez