

1. ¿Qué es Docker Compose? ¿Qué utilidades tiene? Instálalo y comprueba que está instalado.

Docker Compose es una herramienta para definir y ejecutar aplicaciones de Docker de varios contenedores.

En Compose, se usa un archivo YAML para configurar los servicios de la aplicación. Después, con un solo comando, se crean y se inician todos los servicios de la configuración.

Puede ser útil coordinar varias imágenes de contenedor en un solo equipo host.

Instalación:

1. Descargar versión 1.27.4 de GitHub:

```
sudo curl -L "https://github.com/docker/compose/releases/download/1.27.4/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

2. A continuación, estableceremos los permisos correctos para que el comando docker-compose sea ejecutable:

```
sudo chmod +x /usr/local/bin/docker-compose
```

3. Para verificar que la instalación se realizó correctamente, puede ejecutar:

```
docker-compose --version
```

```
sandra@sandra-UX410UAK:~$ sudo curl -L "https://github.com/docker/compose/releases/download/1.27.4/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total   Spent    Left   Speed
  0     0    0     0    0     0      0      0  --:--:-- --:--:-- --:--:--    0
100 11.6M 100 11.6M    0     0 1783k      0  0:00:06  0:00:06 --:--:-- 1825k
sandra@sandra-UX410UAK:~$ sudo chmod +x /usr/local/bin/docker-compose
sandra@sandra-UX410UAK:~$ docker-compose --version
docker-compose version 1.27.4, build 40524192
sandra@sandra-UX410UAK:~$
```

2. Haz un glosario de comandos de Docker Compose

NOTA: se puede utilizar docker-compose o doco, que es la abreviatura

docker-compose up

```
docker-compose up [options] [--scale SERVICE=NUM...] [SERVICE...]
```

Se utiliza para iniciar un proyecto. Intenta automatizar una serie de operaciones, incluida la creación de un espejo, (re)crear un servicio, iniciar un servicio y asociar un contenedor relacionado con el servicio.

A veces necesitará `docker-compose up --rebuild` después de hacer cambios en el código.

docker-compose down

`docker-compose down [options]`

Detiene contenedores. Y elimina contenedores, redes, volúmenes e imágenes (definidos en `docker-compose.yml`) creados por `up`.

docker-compose build

`docker-compose build [options] [SERVICE...]`

En el directorio del proyecto, se ejecuta `docker-compose build` para compilar (reconstruir) el servicio.

docker-compose start

`docker-compose start [SERVICE...]`

Inicia un contenedor de servicios existente.

docker-compose stop

`docker-compose up [options] [--scale SERVICE=NUM...] [SERVICE...]`

Detiene el funcionamiento de los contenedores sin quitarlos.

Se pueden iniciar de nuevo con `docker-compose start`.

docker-compose ps

`docker-compose ps [options] [SERVICE...]`

Muestra la lista de contenedores para un servicio.

docker-compose pause

`docker-compose pause [SERVICE...]`

Pausa la ejecución de contenedores de un servicio.

Se pueden reanudar con `docker-compose unpause`

docker-compose unpause

`docker-compose unpause [SERVICE...]`

Reanuda los contenedores en pausa de un servicio.

docker-compose bundle

`docker-compose bundle [options]`

Genera un paquete de aplicaciones distribuidas (DAB) a partir del archivo Compose.

Las imágenes deben tener resúmenes almacenados, lo que requiere la interacción con un registro de Docker.

Si no se almacenan resúmenes para todas las imágenes, se pueden obtener con `docker-compose pull` o `docker-compose push`.

docker-compose exec

`docker-compose exec [options] [-e KEY=VAL...] SERVICE COMMAND [ARGS...]`

Esto es similar a `docker exec`, pero los comandos por defecto asignan un TTY.

Ejemplo: se usa `docker-compose exec web sh` para obtener un aviso interactivo.

docker-compose config

`docker-compose config [options]`

Verifica que el formato del archivo compose sea correcto. Si es correcta, se muestra la configuración. Si el formato es incorrecto, se muestra la causa del error.

docker-compose help

docker-compose help COMMAND

Muestra ayuda e instrucciones de uso para un comando.

docker-compose events

docker-compose events [options] [SERVICE...]

Transmite eventos de contenedor para cada contenedor en el proyecto.

Ejemplo: `docker-compose events --json` para transmitir en formato JSON.

docker-compose kill

docker-compose kill [options] [SERVICE...]

Obliga a los contenedores en ejecución a detenerse mediante el envío de una señal SIGKILL.

Opcionalmente se puede pasar la señal, por ejemplo: `docker-compose kill -s SIGINT`

docker-compose logs

docker-compose logs [options] [SERVICE...]

Muestra la salida de registro de los servicios.

docker-compose restart

docker-compose restart [options] [SERVICE...]

Reinicia todos los servicios detenidos y en ejecución.

docker-compose port

docker-compose port [options] SERVICE PRIVATE_PORT

Imprime el puerto público al que se asigna un puerto de contenedor.

docker-compose rm

docker-compose rm [options] [SERVICE...]

Elimina los contenedores de servicio detenidos. Cualquier dato que no esté en un volumen se perderá.

De forma predeterminada, los volúmenes anónimos adjuntos a los contenedores no se eliminan. Esto se puede anular con `-v`.

docker-compose pull

docker-compose pull [options] [SERVICE...]

Extrae una imagen asociada con un servicio definido en un `docker-compose.yml`

docker-compose top

docker-compose top [SERVICE...]

Vea los procesos que se ejecutan dentro de cada contenedor de servicios.

docker-compose push

docker-compose push [options] [SERVICE...]

Empuja imágenes para servicios a sus respectivos `registry/repository`

docker-compose run

`docker-compose run [options] [-v VOLUME...] [-p PORT...] [-e KEY=VAL...] [-l KEY=VALUE...] SERVICE [COMMAND] [ARGS...]`

Ejecuta un comando único contra un servicio.

Por ejemplo, el siguiente comando inicia el servicio web y ejecuta bash como su comando `docker-compose run web bash`.

docker-compose version

`docker-compose version`

Imprime la versión de `docker-compose`.

3. ¿Qué es un fichero YAML? ¿Qué estructura tiene? Investiga y detállalo. Haz también un glosario de YAML.

El fichero YAML es un lenguaje de declaración de datos que facilita la legibilidad y la capacidad de escritura del usuario. Este formato de serialización de datos se encarga de almacenar archivos de configuración y se puede usar junto con todos los lenguajes de programación.

YAML es un acrónimo de su nombre en inglés *Yet Another Markup Language* o *YAML Ain't Markup Language*.

La sintaxis básica de este fichero indica que cada archivo YAML inicia con `---` y su extensión es `.yaml`.

Para la función de mapeo, la estructura es *clave: valor*. Es fundamental que se deje un espacio entre los indicadores.

Este formato de serialización también permite datos como caracteres, cadenas, valores flotantes, números enteros, así como colecciones (por ejemplo matrices) y listas construidas partiendo de otros datos básicos.

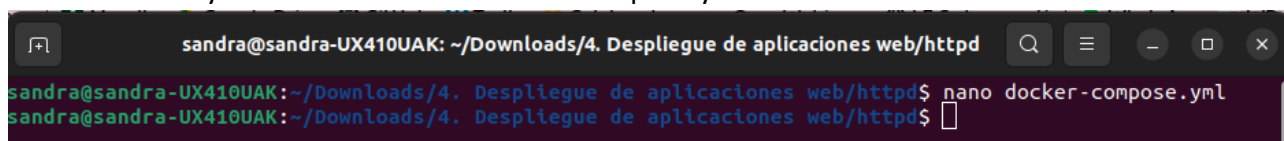
YAML solo admite espacios y tiene la capacidad de distinguir entre mayúsculas y minúsculas.

4. Crea un archivo de configuración para una aplicación que contiene un único servicio:

- **Imagen:** `httpd:2.4` (del servidor web Apache)
- **Puerto:** el host de Docker publicará el puerto 80 y hará una redirección con el puerto 80 del contenedor.
- **Bind mount:** lo creamos entre el directorio actual del host de Docker y el directorio `usr/local/apache2/htdocs/` del contenedor (que es el directorio que utiliza el servidor web para servir el contenido que encuentre en su interior).

Creamos un directorio llamado `httpd`.

Entramos en él y creamos el archivo `docker-compose.yml`



```
sandra@sandra-UX410UAK: ~/Downloads/4. Despliegue de aplicaciones web/httpd
sandra@sandra-UX410UAK:~/Downloads/4. Despliegue de aplicaciones web/httpd$ nano docker-compose.yml
sandra@sandra-UX410UAK:~/Downloads/4. Despliegue de aplicaciones web/httpd$
```

Dentro escribimos lo siguiente:

```
docker-compose.yml
1 version: '3.8'
2
3 services:
4   httpd:
5     image: httpd:2.4
6     ports:
7       - 80:80
8     volumes:
9       - ./usr/local/apache2/htdocs/
```

5. Consulta la lista de contenedores que están en ejecución y explica la salida. ¿Qué diferencia hay entre “docker ps” y “docker compose ps”?

docker-compose ps

El comando **docker ps** muestra todos los contenedores que están en ejecución dentro del host, mientras que **docker-compose ps** solo muestra los contenedores del archivo docker-compose.yml que están en ejecución.

6. Muestra la salida de docker-compose.yml

Si lo hago con **docker-compose logs**, salta una excepción, por permisos, ya que tengo instalado docker con sudo.

```
sandra@sandra-UX410UAK: ~/Downloads/4. Despliegue de aplicaciones web/httpd
sandra@sandra-UX410UAK:~/Downloads/4. Despliegue de aplicaciones web/httpd$ docker-compose logs
Traceback (most recent call last):
  File "urllib3/connectionpool.py", line 677, in urlopen
  File "urllib3/connectionpool.py", line 392, in _make_request
  File "http/client.py", line 1252, in request
  File "http/client.py", line 1298, in _send_request
  File "http/client.py", line 1247, in endheaders
  File "http/client.py", line 1026, in _send_output
  File "http/client.py", line 966, in send
  File "docker/transport/unixconn.py", line 43, in connect
PermissionError: [Errno 13] Permission denied

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "requests/adapters.py", line 449, in send
  File "urllib3/connectionpool.py", line 727, in urlopen
  File "urllib3/util/retry.py", line 403, in increment
  File "urllib3/packages/six.py", line 734, in reraise
  File "urllib3/connectionpool.py", line 677, in urlopen
  File "urllib3/connectionpool.py", line 392, in _make_request
  File "http/client.py", line 1252, in request
  File "http/client.py", line 1298, in _send_request
  File "http/client.py", line 1247, in endheaders
  File "http/client.py", line 1026, in _send_output
  File "http/client.py", line 966, in send
  File "docker/transport/unixconn.py", line 43, in connect
urllib3.exceptions.ProtocolError: ('Connection aborted.', PermissionError(13, 'Permission denied'))
```

```
During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "docker/api/client.py", line 205, in _retrieve_server_version
  File "docker/api/daemon.py", line 181, in version
  File "docker/utils/decorators.py", line 46, in inner
  File "docker/api/client.py", line 228, in _get
  File "requests/sessions.py", line 543, in get
  File "requests/sessions.py", line 530, in request
  File "requests/sessions.py", line 643, in send
  File "requests/adapters.py", line 498, in send
requests.exceptions.ConnectionError: ('Connection aborted.', PermissionError(13, 'Permission denied'))
```

```
During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "bin/docker-compose", line 3, in <module>
  File "compose/cli/main.py", line 67, in main
  File "compose/cli/main.py", line 123, in perform_command
  File "compose/cli/command.py", line 69, in project_from_options
  File "compose/cli/command.py", line 132, in get_project
  File "compose/cli/docker_client.py", line 43, in get_client
  File "compose/cli/docker_client.py", line 170, in docker_client
  File "docker/api/client.py", line 188, in __init__
  File "docker/api/client.py", line 213, in _retrieve_server_version
docker.errors.DockerException: Error while fetching server API version: ('Connection aborted.', PermissionError(13, 'Permission denied'))
[8709] Failed to execute script docker-compose
sandra@sandra-UX410UAK:~/Downloads/4. Despliegue de aplicaciones web/httpd$
```

Si hago el mismo comando con sudo, no me da error y me sale el siguiente mensaje:

```
sandra@sandra-UX410UAK:~/Downloads/4. Despliegue de aplicaciones web/httpd$ sudo docker-compose
logs
Attaching to
asandra@sandra-UX410UAK:~/Downloads/4. Despliegue de aplicaciones web/httpd$
```

7. Visiona el video relativo a docker compose del módulo 7.4.