

1. ¿Que 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.

Docker Compose se encargará de ejecutar todas las acciones necesarias para mantener ese estado

```
estudiante@DAW1:~$ sudo curl -L "https://github.com/docker/compose/releases/download/1.26.0/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
[sudo] password for estudiante:
% 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 2835k      0  0:00:04 0:00:04 --:--:-- 3186k
estudiante@DAW1:~$ sudo chmod +x /usr/local/bin/docker-compose
estudiante@DAW1:~$ docker-compose --version
docker-compose version 1.26.0, build d4451659
estudiante@DAW1:~$
```

docker-compose up

2. Haz un glosario de comandos de Docker Compose

Siempre que usemos los comandos de docker-compose debemos hacerlo en el mismo directorio que el archivo docker-compose.yml

- **docker-compose up** : levantar toda la aplicación (todos los contenedores que contiene)
- **docker-compose start contenedor** : levantar un contenedor concreto
- **docker-compose stop contenedor** : parar un contenedor concreto
- **docker-compose down contenedor** : para frenar y eliminar contenedores
- **docker-compose ps** : lista los contenedores que están corriendo
- **docker-compose --versión** : comprobar la versión instalada de docker compose
- **docker-compose build** : generar la imagen

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

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 no propone etiquetas solo formato e identificadores mínimos, centrándose realmente en los datos.

Cada YAML comienza con --- que denota el inicio de un archivo YAML.

ESTRUCTURA Y GLOSARIO:

Los strings no deben estar entre comillas, aunque también es válido.

Los strings de líneas múltiples pueden ser escritos como un 'bloque literal' (usando pipes |) o como un 'bloque doblado' (usando >)

La indentación se usa para anidar elementos:

un_mapa_indentado:

```
  llave: valor
  otra_llave: otro valor
  otro_mapa_indentado:
    llave_interna: valor_interno
```

Las llaves de los mapas no requieren ser strings necesariamente.

0.25: una llave numérica

Las colecciones en YAML usan la indentación para delimitar el alcance y cada elemento de la colección inicia en su propia línea.

Las secuencias (equivalentes a listas o arreglos) se ven así:

- Amarillo
- Verde
- Azul

Las secuencias pueden tener distintos tipos en su contenido.

secuencia_combinada:

- texto
- 5
- 0.6
- llave: valor
- - Esta es una secuencia
 - ...dentro de otra secuencia

Dado que todo JSON está incluido dentro de YAML, también puedes escribir mapas con la sintaxis de JSON y secuencias:

mapa_de_json_1: {"llave": "valor"}

mapa_de_json_2:

```
  llave: valor
```

Ejemplos:

```
---  
  
MALE: FALSE  
  
GPA: 3.61  
  
ISSUES: NULL  
  
NAME: "BIGYAN"  
  
AGE: 16
```

```
55 definitions:  
56   person:  
57     properties:  
58       id:  
59         type: string  
60         description: Identificador unico de  
61                       la persona  
62       firstName:  
63         type: string  
64         description: Nombre  
65       lastName:  
66         type: string  
67         description: Apellidos  
68       age:  
69         type: number  
70         format: Int32  
71         minimum: 0  
72       gender:  
73         type: string  
74         enum:  
75           - Male  
76           - Female
```

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

- **Imagen:** httpd:2.4 (del servidor web Apache):

steps:

name: 'gcr.io/httpd:2.4r/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).

```
version: '3.7'
services:
  apache:
    image: httpd:2.4
    container_name: my-apache-app
    ports:
      - '8080:80'
    volumes:
      - ./website:/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”?

```
estudiante@DAW1:~/compose-demo$ docker-compose ps

```

Name	Command	State	Ports
my-apache-app	httpd-foreground	Up	0.0.0.0:8080->80/tcp, :::8080->80/tcp

```
estudiante@DAW1:~/compose-demo$
```

Name: Se refiere al nombre de nuestra app.

Command: Nos indica el tipo de servidor web que estamos utilizando en este caso utilizamos httpd de apache.

State: Nos indica el estado de nuestra app.

Ports: Nos aparece los puertos utilizados y los redireccionamientos del mismo.

`docker ps` enumera todos los contenedores en ejecución en el motor docker.
`docker-compose ps` enumera los contenedores relacionados con las imágenes declaradas en docker-compose file.

6. Muestra la salida de docker-compose.yml

```
version: '3.7'
services:
  apache:
    image: httpd:2.4
    container_name: my-apache-app
    ports:
      - '8080:80'
    volumes:
      - ./website:/usr/local/apache2/htdocs
```

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