Docker Compose

Índice

1. Introducción

Docker Compose es otro proyecto open source que permite definir aplicaciones multi-contenedor de una manera sencilla y declarativa. Es una alternativa más cómoda al uso de los comandos docker run y docker build, que resultan un tanto tediosos cuando trabajamos con aplicaciones de varios componentes.

De esta forma se consiguen diferentes cosas:

- Centralizar todos los parámetros de ejecución en un mismo fichero.
- Homogeneizar la puesta a punto de los diferentes entornos de desarrollo, test y producción

El primer paso para utilizar Docker Compose es instalarlo. En este enlace encontrarás las instrucciones.

Docker Compose va a constar de los siguientes elementos:

- Fichero YALM: define contenedores, redes, volúmenes...
- Cliente docker compose, para automatizar tareas de desarrollo y test, con los ficheros YALM

En los siguientes apartados nos centraremos en cómo construir ficheros YALM para Docker Compose, así como los comandos más usuales.

2. Ficheros YALM

Los ficheros utilizados por Docker Compose están formados por el lenguaje de marcas YALM. En este enlace tienes más información sobre este lenguaje.

El nombre por defecto de este fichero es "docker-compose.yml", aunque es posible utilizar otro nombre (se detallará más adelante en este documento).

Aunque estos ficheros son utilizados por Docker Compose para la puesta a punto del entorno de desarrollo, también son utilizados para el despliegue en el entorno de producción mediante Docker Swarm.

Vamos a aprender cómo construir ficheros YALM, con varios ejemplos:

Ejemplo 1:

- Utiliza versión 2 de Docker Compose.
- Dispone de un solo servicio, que se trata de un contenedor identificado como jekyll:
  - Basado en la imagen del repositorio bretfisher/jekyll-serve (etiqueta "latest"),
  - Cuenta con un volumen (el directorio del host es el mismo donde se encuentra el fichero YALM, y el directorio del contenedor es /site)

- Expone el puerto 4000 del contenedor que se corresponde con el puerto 80 del host

Ejemplo 2:

- Utiliza versión 3 de Docker Compose.
- Dispone de dos servicios, uno es el web server de Apache, y el otro es nginx actuando como reverse proxy:
  - Nginx utiliza la versión 1.13, expone el puerto 80 con el 80 del host
  - Nginx dispone de un volumen de solo lectura (mediante el cual se proporciona el fichero de configuración de nginx al contenedor). Si abrimos nginx.conf, se hace referencia al servicio "web" (línea 7). Esto es posible al servicio DNS de Docker que posibilita la comunicación entre contenedores. De esta forma, nginx redirige las peticiones al servicio "web", y éste no es necesario que exponga ningún puerto. Apache va a mostrar su página de inicio por defecto.
  - El servicio "web" es un contenedor basado en la última imagen de Apache.

Ejemplo 3:

- Utiliza versión 2 de Docker Compose.
- Dispone de dos servicios, uno es el web server de Apache, y el otro es nginx actuando como reverse proxy:
  - El servicio "proxy" se basa en una imagen que se construye a partir de un Dockerfile, con el nombre "nginx.Dockerfile", y expone el puerto 80. Se hará el build de la imagen en el momento en el que se ejecute el comando docker-compose up (se verá más adelante este comando).
  - En nginx.Dockerfile podemos ver que la imagen se basa en la imagen

nginx:1.13, y lo único que se hace es copiar el fichero de configuración en la ruta correspondiente del contenedor. De esta forma, no es necesario establecer el volumen del ejemplo 2.

- El servicio "web" es un contenedor basado en la última imagen de Apache. Se establece un volumen por el cual Apache mostrará el contenido de la carpeta html. Sabemos que se ha de enlazar con la ruta /usr/local/apache2/htdocs/ por la propia documentación de Apache, en Docker Hub.

2.1. Directivas

Las directivas más usuales que podemos encontrar en ficheros YALM de Docker Compose son las siguientes:

- **Version**: si no se especifica esta directiva se asume la versión 1, pero esto no es recomendable. Como mínimo la versión 2.
- **Services**: Sección jerárquica: se tiene una subsección por cada servicio (contenedor). Diferentes servicios pueden estar basados en la misma imagen. Pueden contener datos como:
  - image: especifica la imagen a utilizar, si no se especifica se hará el build con los datos del Dockerfile.
  - build: en caso de que la imagen se haya de construir, contendrá instrucciones sobre cómo construir la imagen.
  - command: sobreescribe el CMD especificado en la imagen.
  - environment: establece variables de entorno.
  - volumes: enlaza con volúmenes existentes en la sección volumes.
  - networks: se asocia el contenedor con una red definida en networks.

  - depends_on: espera a que se inicie otro servicio para poder levantar éste.
- Volumes: creación de volúmenes para enlazar con contenedores
- Networks: creación de redes para intercomunicar diferentes contenedores entre sí

Todas las directivas, para la versión 3, se encuentran en este enlace.

3. Operaciones básicas

En este apartado vamos a ver las operaciones básicas a realizar con el cliente de Docker Compose, desde la línea de comandos. Se trata de operaciones enfocadas a un entorno de desarrollo, ya que, como se ha comentado anteriormente, para el despliegue en un entorno de producción lo adecuado es utilizar Docker Swarm.

Los comandos más usuales son:

- docker-compose up: construye (o reconstruye) imágenes, configura volúmenes, redes y arranca todos los contenedores.
- docker-compose down: para todos los contenedores, redes y volúmenes que ya no se necesiten. Los volúmenes de tipo bind mount o nombrados no se eliminan al ejectuar este comando.

Típicamente (si nuestros proyectos tienen un Dockerfile y un docker-compose.yml, y están subidos a un repositorio de Github), un nuevo desarrollador que se incorpore al equipo solo tendría que hacer lo siguiente para configurar su entorno de desarrollo:

```
git clone github.com/some/software
docker-compose up
```

Con todo esto, vamos a realizar un ejemplo, siguiendo los siguientes pasos:

1. Descargamos el repositorio de este enlace. Ya se ha explicado anteriormente cómo funciona el Dockerfile de este repositorio.

2. Vamos al directorio donde tengamos el docker-compose.yml y ejecutamos el comando:

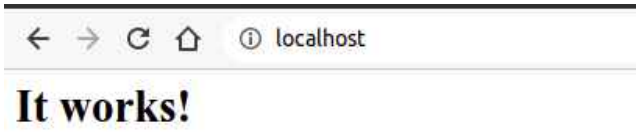
docker-compose up

```
manu@manu-HP-Laptop-15s-fq1xxx:~/                                    $ docker-compose up
Creating network "compose-sample-2_default" with the default driver
Pulling proxy (nginx:1.13)...
1.13: Pulling from library/nginx
f2aa67a397c4: Pull complete
3c091c23e29d: Pull complete
4a99993b8636: Pull complete
Digest: sha256:b1d09e9718890e6ebbbd2bc319ef1611559e30ce1b6f56b2e3b479d9da51dc35
Status: Downloaded newer image for nginx:1.13
Pulling web (httpd:)...
latest: Pulling from library/httpd
5eb5b503b376: Already exists
a43a76ccc967: Pull complete
942bd346e7f7: Pull complete
cdb155854ae6: Pull complete
10c4d45228bf: Pull complete
Digest: sha256:5cc947a200524a822883dc6ce6456d852d7c5629ab177dfbf7e38c1b4a647705
Status: Downloaded newer image for httpd:latest
Creating compose-sample-2_proxy_1 ... done
Creating compose-sample-2_web_1   ... done
Attaching to compose-sample-2_web_1, compose-sample-2_proxy_1
web_1    | AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 172.18.0.3. Set the 'ServerName' directive glo
bally to suppress this message
web_1    | AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 172.18.0.3. Set the 'ServerName' directive glo
bally to suppress this message
web_1    | [Thu Feb 17 16:09:16.198489 2022] [mpm_event:notice] [pid 1:tid 140465817963840] AH00489: Apache/2.4.52 (Unix) configured -- resuming norma
l operations
web_1    | [Thu Feb 17 16:09:16.199885 2022] [core:notice] [pid 1:tid 140465817963840] AH00094: Command line: 'httpd -D FOREGROUND'
```

Automáticamente se va a crear una red para que estos dos servicios se comuniquen entre ellos. Si listamos las redes:

```
manu@manu-HP-Laptop-15s-fq1xxx:~$ docker network ls
NETWORK ID          NAME                      DRIVER              SCOPE
d9cdba35d9ae        bridge                    bridge              local
558084f78875        compose-sample-2_default  bridge              local
6750ede529e5        host                      host                local
6c0762cea506        none                      null                local
```

Al no haber incluido el parámetro "-d", se muestran los logs de todos los contenedores. Si visitamos localhost, vemos que se refleja en los logs:



```
web_1    | AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 172.18.0.3. Set the 'ServerName' directive glo
bally to suppress this message
web_1    | AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 172.18.0.3. Set the 'ServerName' directive glo
bally to suppress this message
web_1    | [Thu Feb 17 16:09:16.198489 2022] [mpm_event:notice] [pid 1:tid 140465817963840] AH00489: Apache/2.4.52 (Unix) configured -- resuming norma
l operations
web_1    | [Thu Feb 17 16:09:16.199885 2022] [core:notice] [pid 1:tid 140465817963840] AH00094: Command line: 'httpd -D FOREGROUND'
web_1    | 172.18.0.2 - - [17/Feb/2022:16:20:35 +0000] "GET / HTTP/1.0" 200 45
proxy_1  | 172.18.0.1 - - [17/Feb/2022:16:20:35 +0000] "GET / HTTP/1.1" 200 45 "-" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Ge
cko) Chrome/98.0.4758.80 Safari/537.36" "-"
proxy_1  | 172.18.0.1 - - [17/Feb/2022:16:20:36 +0000] "GET /favicon.ico HTTP/1.1" 404 196 "http://localhost/" "Mozilla/5.0 (X11; Linux x86_64) AppleW
ebKit/537.36 (KHTML, like Gecko) Chrome/98.0.4758.80 Safari/537.36" "-"
web_1    | 172.18.0.2 - - [17/Feb/2022:16:20:36 +0000] "GET /favicon.ico HTTP/1.0" 404 196
```

3. Ahora pulsamos ctrl + c para parar los contenedores:

```
web_1    | 172.18.0.2 - - [17/Feb/2022:16:20:36 +0000] "GET /favicon.ico HTTP/1.0" 404 196
^CGracefully stopping... (press Ctrl+C again to force)
Stopping compose-sample-2_web_1    ... done
Stopping compose-sample-2_proxy_1 ... done
```

y volvemos a levantarlos en segundo plano con el comando:

docker-compose up -d

```
manu@manu-HP-Laptop-15s-fq1xxx:~/                                                    $ docker-compose up -d
Starting compose-sample-2_web_1   ... done
Starting compose-sample-2_proxy_1 ... done
```

4. Por último, vamos a inspeccionar lo que ocurre con la composición que hemos lanzado en segundo plano, al igual que hicimos para los contenedores de forma individual.

Para ver los logs producidos por los contenedores, utilizamos el siguiente comando:

docker-compose logs

```
manu@manu-HP-Laptop-15s-fq1xxx:~/                                                    $ docker-compose logs
Attaching to compose-sample-2_web_1, compose-sample-2_proxy_1
proxy_1  | 172.18.0.1 - - [17/Feb/2022:16:20:35 +0000] "GET / HTTP/1.1" 200 45 "-" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Ge
cko) Chrome/98.0.4758.80 Safari/537.36" "-"
proxy_1  | 172.18.0.1 - - [17/Feb/2022:16:20:36 +0000] "GET /favicon.ico HTTP/1.1" 404 196 "http://localhost/" "Mozilla/5.0 (X11; Linux x86_64) AppleW
ebKit/537.36 (KHTML, like Gecko) Chrome/98.0.4758.80 Safari/537.36" "-"
web_1    | AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 172.18.0.3. Set the 'ServerName' directive glo
bally to suppress this message
web_1    | AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 172.18.0.3. Set the 'ServerName' directive glo
bally to suppress this message
web_1    | [Thu Feb 17 16:09:16.198489 2022] [mpm_event:notice] [pid 1:tid 140465817963840] AH00489: Apache/2.4.52 (Unix) configured -- resuming norma
l operations
web_1    | [Thu Feb 17 16:09:16.199885 2022] [core:notice] [pid 1:tid 140465817963840] AH00094: Command line: 'httpd -D FOREGROUND'
web_1    | 172.18.0.2 - - [17/Feb/2022:16:20:35 +0000] "GET / HTTP/1.0" 200 45
web_1    | 172.18.0.2 - - [17/Feb/2022:16:20:36 +0000] "GET /favicon.ico HTTP/1.0" 404 196
web_1    | [Thu Feb 17 16:26:22.764421 2022] [mpm_event:notice] [pid 1:tid 140465817963840] AH00492: caught SIGWINCH, shutting down gracefully
web_1    | AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 172.18.0.2. Set the 'ServerName' directive glo
bally to suppress this message
web_1    | AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 172.18.0.2. Set the 'ServerName' directive glo
bally to suppress this message
web_1    | [Thu Feb 17 16:26:58.410507 2022] [mpm_event:notice] [pid 1:tid 139737161043264] AH00489: Apache/2.4.52 (Unix) configured -- resuming norma
l operations
web_1    | [Thu Feb 17 16:26:58.410602 2022] [core:notice] [pid 1:tid 139737161043264] AH00094: Command line: 'httpd -D FOREGROUND'
```

Para listar los contenedores que están siendo ejecutados:

docker-compose ps

```
manu@manu-HP-Laptop-15s-fq1xxx:~/                                    $ docker-compose ps
          Name                        Command          State         Ports
-------------------------------------------------------------------------------
compose-sample-2_proxy_1   nginx -g daemon off;   Up      0.0.0.0:80->80/tcp
compose-sample-2_web_1     httpd-foreground       Up      80/tcp
```

Para ver todos los procesos que se están ejecutando dentro de los contenedores:

docker-compose top

```
manu@manu-HP-Laptop-15s-fq1xxx:~/                                    $ docker-compose top
compose-sample-2_proxy_1
  UID        PID    PPID    C   STIME   TTY     TIME                        CMD
-------------------------------------------------------------------------------------
root       12595   12561   0   17:26   ?     00:00:00   nginx: master process nginx -g daemon off;
systemd+   12735   12595   0   17:26   ?     00:00:00   nginx: worker process

compose-sample-2_web_1
  UID        PID    PPID    C   STIME   TTY     TIME           CMD
-------------------------------------------------------------------------
root       12605   12557   0   17:26   ?     00:00:00   httpd -DFOREGROUND
www-data   12736   12605   0   17:26   ?     00:00:00   httpd -DFOREGROUND
www-data   12737   12605   0   17:26   ?     00:00:00   httpd -DFOREGROUND
www-data   12738   12605   0   17:26   ?     00:00:00   httpd -DFOREGROUND
```

Finalmente, para ver todas las posiblidades del comando docker-compose, podemos utilizar el siguiente comando, similar a como lo hacíamos para los comandos básicos:

docker-compose --help

4. Limpieza del sistema

En muchas ocasiones, por diversas circunstancias, es posible que queden objetos incosistentes por fallos en la ejecución de estos comandos (especialmente docker-compose up). En este caso, es de especial utilidad los comandos de purgado descritos en el documento sobre Docker básico. En este enlace se dispone de toda la información sobre el comando de purgado de Docker.

5. Múltiples ficheros YALM

Aunque no nos vamos a extender en este aspecto, cabe mencionar que es posible utilizar diferentes ficheros de Docker Compose para diferentes entornos.

Por ejemplo:

- En el entorno de desarrollo podemos no querer utilizar un contenedor nginx, pero sí

en producción.

- Podemos querer utilizar nginx en desarrollo (por ejemplo, para probar nuestra arquitectura antes de desplegarla), pero solo vamos a configurar los certificados digitales en producción.
- En desarrollo queremos acceder a unas APIs de terceros (por ejemplo, el servicio de correo de Gmail), pero en producción podemos llegar a utilizar unas claves diferentes, no disponibles para todos los desarrolladores.

En este enlace se puede encontrar toda la información. En un ejemplo común, podemos encontrar los siguientes ficheros:

- docker-compose.yml: se trata del fichero base, que contiene las directivas que van a ser comunes a todos los entornos.
- docker-compose.override.yml: utilizado para sobreescribir o completar las directivas del fichero base, en un entorno de desarrollo/test.
- docker-compose.prod.yml: utilizado para sobreescribir o completar las directivas del fichero base, en un entorno de producción. Se utiliza con Docker Swarm.

Dicho esto, vamos a ilustrar la utilización de estos ficheros mediante comandos, dependiendo del entorno en el que estemos. Asumimos que los tres archivos se encuentran en la misma ruta, en la cual ejecutamos los comandos:

Desarrollo

docker-compose up

Este comando tiene en cuenta los archivos:

- docker-compose.yml
- docker-compose.override.yml

De esta forma, el archivo docker-compose.override.yml sobreescribe y/o extiende la

configuración de docker-compose.yml

Producción

```
docker-compose -f docker-compose.yml -f docker-compose.prod.yml up -d
```

Este comando tiene en cuenta los archivos

- docker-compose.yml
- docker-compose.prod.yml

De esta forma, el archivo docker-compose.prod.yml sobreescribe y/o extiende la configuración de docker-compose.yml.

NOTA: Ya se ha comentado que en producción, lo más adecuado es utilizar Docker Swarm, no Docker Compose, aunque los comandos son muy parecidos y la sobreescritura del fichero docker-compose.prod.yml funciona del mismo modo. En cualquier caso, nada evitaría que utilizásemos Docker Compose para levantar la aplicación en un servidor, aunque no es la práctica recomendada.

6. ACTIVIDADES

TEST

1. ¿Cuál es la versión mínima recomendada para el fichero docker-compose.yml?
   - 1
   - 2
   - 3.1

2. ¿Cuál de los siguientes elementos crea Docker Compose, aunque no se especifique explícitamente en el archivo de compose?
   - Network
   - Backup
   - Pizza
   - Registro

3. ¿Cuál es la función del build context en el fichero compose?
   - La construcción ha de ocurrir antes del comando de ejecución
   - La la imagen referenciada ha de existir en Docker Hub antes de ejecutarse el comando
   - Compose ha de construir una imagen a partir de una imagen construida en un fichero anterior
   - Especifica de dónde se encuentra le Dockerfile para construir la imagen

4. La directiva "key" en un fichero de compose tiene la misma función que la directiva EXPOSE en un Dockerfile:
   - Verdadero
   - Falso

5. El usuario de una base de datos, ¿en qué sección se especifica dentro de un fichero compose?
   - image:
   - environment:
   - identity:
   - volumes:

6. ¿De dónde deriva Compose el nombre DNS?
   - Nombre de servicio
   - Nombre de imagen
   - Nombre de volumen
   - Nombre de usuario de reddit

EJERCICIO

En esta actividad se va a crear un fichero de docker compose. Para ello sigue las siguientes directivas:

- Utiliza la versión 2 de Docker Compose. Puedes encontrar la sintaxis en este enlace.
- Imágenes a utilizar: drupal (gestor de contenidos) y postgres (base de datos). Han de ser las

oficiales.

- Exponer el puerto correspondiente de drupal al 8080 del host. Tendrás que averiguar qué puerto utiliza drupal internamente (detalla cómo lo has averiguado). No has de exponer el puerto de postgres.
- Configurar postgres con la variable POSTGRES_PASSWORD.
- En cuanto a volúmenes, deberás investigar:
  - Según la documentación oficial de la imagen de Drupal, qué volúmenes has de crear para el servicio.
  - Según la documentación de docker compose, cómo crear esos volúmenes dentro del fichero de docker compose (https://docs.docker.com/compose/compose-file/compose-file-v2/#volume-configuration-reference)
- Una vez se lancen los servicios, es necesario configurar Drupal mediante el navegador
  - Consejo: al configurar Drupal con el navegador, para que encuentre la BBDD normalmente se especifica localhost, pero en este caso habrá que indicar el nombre del servicio de postgres (utilizado con nombre DNS, para intercomunicar servicios). En opciones avanzadas de Drupal también se ha de cambiar localhost por el nombre del servicio de postgres.
- Verificar mediante los logs que todos los componentes se crean, se arrancan y se comunican correctamente.
- Finalmente, hacer limpieza del sistema y también de los volúmenes (expresamente).