

Introducción a Docker

UD 06. Caso práctico 03 - Proxy Nginx y balanceo escalado con Docker Compose



Fons Social Europeu

L'FSE inverteix en el teu futur

Autor: Sergi García Barea

Actualizado Abril 2021

Licencia



Reconocimiento – NoComercial - CompartirIgual (BY-NC-SA): No se permite un uso comercial de la obra original ni de las posibles obras derivadas, la distribución de las cuales se debe hacer con una licencia igual a la que regula la obra original.

Nomenclatura

A lo largo de este tema se utilizarán distintos símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:

Importante

Atención

Interesante

1. Introducción	3
2. Directorio "Apache": ficheros "Dockerfile" e "index.php"	3
3. Fichero de configuración "./nginxproxy/nginx.conf"	3
4. Fichero de configuración "docker-compose.yml"	4
5. Paso 1: Poniendo en marcha el sistema	4
6. Paso 2: Escalando localmente para aumentar nuevos servidores	5
7. Bibliografía	5

UD06. CASO PRÁCTICO 03

1. INTRODUCCIÓN

En este caso práctico vamos a poner en marcha un sistema de balanceo de carga, aprovechando el propio escalado que nos brinda “**Docker Compose**”. Tendremos como punto de entrada un proxy utilizando “**Nginx**” y los encargados de servir las imágenes serán servidores “**Apache + PHP**”.

2. DIRECTORIO “APACHE”: FICHEROS “DOCKERFILE” E “INDEX.PHP”

En este caso práctico, construiremos una imagen con un servidor web “**Apache + PHP**”. Para construirlo tenemos dos ficheros: “**Dockerfile**” e “**index.php**”. Aquí el contenido del primero:

```
#Imagen base: PHP 7.2 con apache
FROM php:7.2-apache
#Copiamos del anfitrión "index.php" a la imagen
COPY index.php /var/www/html/
```

Esta parte de una imagen base y simplemente copia del anfitrión el fichero “**index.php**” al directorio “**/var/www/html**” de la imagen.

El contenido del fichero “**index.php**” es el siguiente:

```
<html>
  <body>
    <h1>Servido por: Servidor con IP <?php echo $_SERVER['SERVER_ADDR'];?>
    y hostname <?php echo gethostname(); ?></h1>
  </body>
</html>
```

Básicamente, imprimirá la IP desde la que se sirve la petición y el hostname de la máquina que lo sirve (que en este caso, será el identificador Docker de la imagen).

3. FICHERO DE CONFIGURACIÓN “./NGINXPROXY/NGINX.CONF”

El fichero “**./nginxproxy/nginx.conf**” es un fichero que enlazaremos desde nuestro anfitrión con la imagen “**nginx**” que usaremos en “**Docker Compose**” y que nos permitirá configurar un proxy para tener un acceso común al sistema. Su contenido es el siguiente:

```
user nginx;
events {
    worker_connections 1000;
}
http {
    server {
        listen 4000;
        location / {
            proxy_pass http://apache:80;
        }
    }
}
```

Este fichero indica que “**nginx**” escuchará en el puerto 4000 y hará de proxy con el host llamado “**apache**” en el puerto 80. El propio “**Docker Compose**” se encargará de que ese host “**apache**” se balancee entre las imágenes que escalemos.

4. FICHERO DE CONFIGURACIÓN “DOCKER-COMPOSE.YML”

El contenido del fichero “**docker-compose.yml**” que incluimos comentado, es el siguiente:

```
version: "3.9"
#Versión del fichero docker-compose 3.9. No obligatorio desde la versión de docker-compose 1.27.0

#Indicamos los servicios a lanzar
services:
  #Plantilla del servicio "apache"
  apache:
    #Lo construimos con el Dockerfile del directorio "apache"
    build: ./apache
    #Indica que siempre que el servicio finalice, se reiniciará
    restart: always
    #Expone el puerto 80 de cada contenedor creado
    ports:
      - "80"

#Plantilla de nginx
nginxproxy:
  image: nginx:latest
  #Mapeamos el fichero de configuración de nuestro anfitrión al contenedor
  volumes:
    - ./nginxproxy/nginx.conf:/etc/nginx/nginx.conf:ro
  #Indicamos que depende de apache
  depends_on:
    - apache
  #Enlaza el puerto 4000 del contenedor con el 4000 del anfitrión
  ports:
    - "4000:4000"
```

Este fichero realiza las siguientes acciones:

- Define la plantilla para Apache, indicando que su imagen debe construirse con un “Dockerfile” situado en el directorio “./apache”. Los contenedores con esta plantilla:
 - Se reiniciará si se detienen.
 - Tendrán expuesto el puerto 80.
- Define la plantilla “nginx”. Esta plantilla:
 - Depende de al menos un contenedor “apache” funcionando.
 - Enlaza el fichero de configuración de “Nginx” desde el anfitrión.
 - Mapea puerto 4000 del contenedor en 4000 del anfitrión.

5. PASO 1: PONIENDO EN MARCHA EL SISTEMA

Podemos poner en marcha todo el sistema montado, simplemente con

```
docker-compose up -d
```

La opción “**-d**” indica que “**Docker Compose**” se ejecute en segundo plano.

La opción “**up**”, descarga y construye imágenes (si no estaban ya). Tras ello lanza los contenedores asociados, siguiendo orden de dependencia.

Si todo ha ido bien, el sistema comenzará a crear y descargar imágenes en orden de dependencia, obteniendo un mensaje similar a este:

```
sergi@ubuntu:~/Desktop/docker-composeUD06/CasoPractico3-EscaladoProxyNginxYApache$ docker-compose up -d
Creating network "casopractico3-escaladoproxynginyapache_default" with the default driver
Building apache
Sending build context to Docker daemon 3.072kB

Step 1/2 : FROM php:7.2-apache
7.2-apache: Pulling from library/php
6ec7b7d162b2: Pulling fs layer
```

Una vez finalizado el sistema de puesta en marcha, podemos, probar que todo funciona OK accediendo a <http://localhost:4000> donde observaremos algo similar a:



Servido por: Servidor con IP 172.24.0.2 y hostname dafee8b1392a

Este mensaje indica la IP y hostname (ID contenedor Docker) de la máquina que lo ha servido.

En este caso, al haber realizado simplemente “**docker-compose up -d**” solo nos ha creado un servidor “**apache**”, por lo cual aunque recarguemos la página, obtendremos el mismo mensaje.

6. PASO 2: ESCALANDO LOCALMENTE PARA AUMENTAR NUEVOS SERVIDORES

Si ahora ejecutamos la siguiente orden:

```
docker-compose down
```

para detener el sistema y lo relanzamos con

```
docker-compose up -d --scale apache=4
```

se añadirán 3 contenedores más “apache”, hasta un total de 4, de forma similar a:

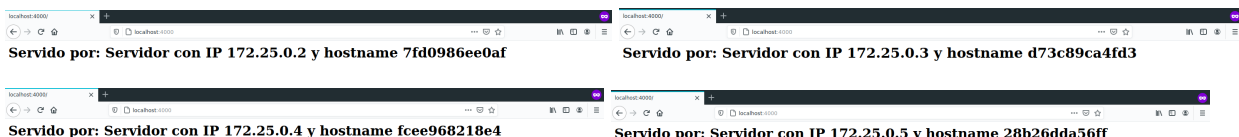
```
sergi@ubuntu:~/Desktop/docker-composeUD06/CasoPractico3-EscaladoProxyNginxYApache$ docker-compose up -d --scale apache=4
Creating casopractico3-escaladoproxynginyapache_apache_2 ... done
Creating casopractico3-escaladoproxynginyapache_apache_3 ... done
Creating casopractico3-escaladoproxynginyapache_apache_4 ... done
casopractico3-escaladoproxynginyapache_nginxproxy_1 is up-to-date
```

Si ahora accedemos a <http://localhost:4000>, cada vez que recarguemos la página, obtendremos una IP y un hostname diferente, dependiendo de cuál de los 4 contenedores te sirva la petición.

¿Cómo es esto posible, si no hemos especificado ni nombre de contenedor ni IP ni ningún otro dato?

En este caso, “**Docker Compose**” se encarga que las peticiones al host “**apache**” (nombre que le hemos dado al servicio en el fichero “**docker-compose.yml**”) se repartan de forma transparente usando “round robin” entre los servidores escalados a partir de ese servicio.

Aquí los 4 ejemplos según quien haya servido la petición.



Servido por: Servidor con IP 172.25.0.2 y hostname 7fd0986ee0af

Servido por: Servidor con IP 172.25.0.3 y hostname d73c89ca4fd3

Servido por: Servidor con IP 172.25.0.4 y hostname fcee968218e4

Servido por: Servidor con IP 172.25.0.5 y hostname 28b26dda56ff

7. BIBLIOGRAFÍA

- [1] Docker Docs <https://docs.docker.com/>
- [2] Docker Compose Docs <https://docs.docker.com/compose/>