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

short line

Autor: Sergi García Barea

Actualizado Abril 2022

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**](#_dbh0n1vac4c8) **3**

**2.** [**Directorio “Apache”: ficheros “Dockerfile” e “index.php”**](#_vyhbfp4t666x) **3**

**3.** [**Fichero de configuración “./nginxproxy/nginx.conf”**](#_xu52i3hvtkkw) **3**

**4.** [**Fichero de configuración “docker-compose.yml”**](#_pext87mvnrlq) **4**

**5.** [**Paso 1: Poniendo en marcha el sistema**](#_3b05p2f899fh) **4**

**6.** [**Paso 2: Escalando localmente para aumentar nuevos servidores**](#_azyrdwgi0kv) **5**

**7.** [**Bibliografía**](#_72yv1gty1dke) **5**

UD06. Caso práctico 03

# 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***”.

# 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/ |
| --- |

Este 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).

# 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.

# 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:  *#Plantill 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 al anfitrión-
  + Mapea puerto 4000 del contenedor en 4000 del anfitrión.

# Paso 1: Poniendo en marcha el sistema

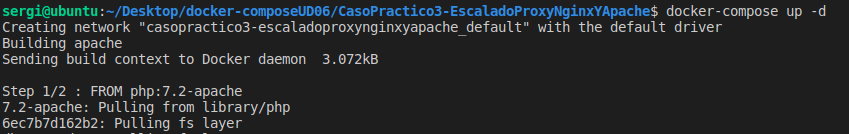
Podemos poner en marcha todo el sistema montado, simplemente con

| docker-compose up -d |
| --- |

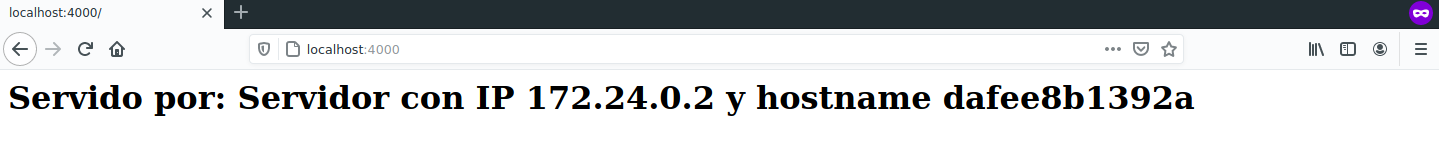
La opción “***-d***” indica que “***Docker Compose***” se ejecute en segundo plano.

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:



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:



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

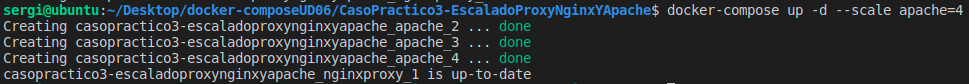
En este caso, al haber realizado simplemente “***docker-compose up -d***” solo nos ha creado un servidor “***apache***”, por lo cual aunque recarguemos la página, obtendremos el mismo mensaje.

# Paso 2: Escalando localmente para aumentar nuevos servidores

Con el sistema en marcha y un contenedor lanzado, si aplicamos la siguiente orden:

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

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

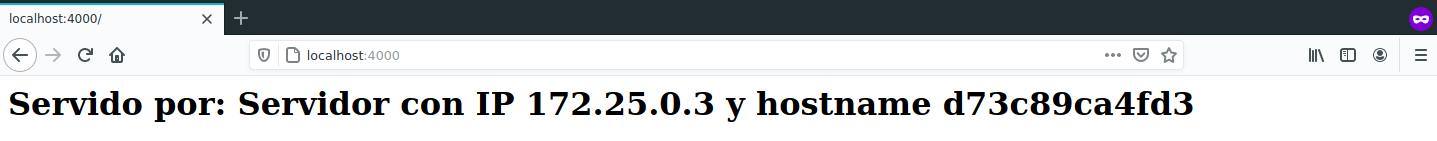
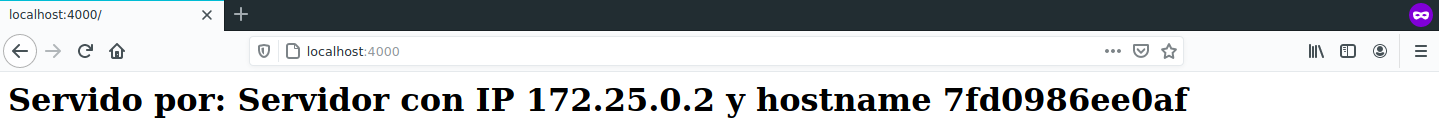


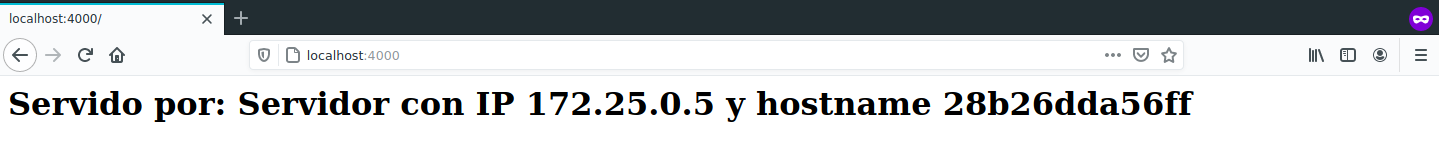
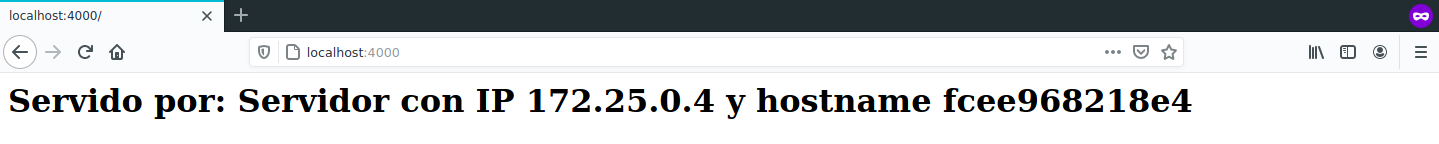
Si ahora accedemos a <http://localhost:4000>, cada vez que recarguemos la página, obtendremos una IP y un hostname diferente, dependiendo de cuál de los 4 contenedores te sirva la petición.

**¿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.





# Bibliografía

[1] Docker Docs <https://docs.docker.com/>

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