Introducción a Docker

UD 05. Caso práctico 02 - Balanceo de carga con HAProxy

Autor: Sergi García Barea

Actualizado Marzo 2023

Licencia

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 **2**

2. Paso 1: Creando las imágenes necesarias **3**

2.1 Dockerfile para imágenes de servidores Apache 3

2.2 Dockerfile para imagen de servidor HAProxy 3

3. Paso 2: Creando la red y los contenedores con Apache **5**

4. Paso 3: Creando contenedor con HAProxy **5**

5. Servidores de Apache usando un volumen común **6**

6. Bibliografía **7**

UD05. Caso práctico 02

1. Introducción

En este caso práctico instalaremos un contenedor Docker con HAProxy http://www.haproxy.org/ que realizará balanceo de carga entre dos contenedores con servidores Apache independientes y cada uno con su contenido. Tras ello, modificaremos la práctica para que ambos servidores lean los datos de un mismo volumen.

2. Paso 1: Creando las imágenes necesarias

En este paso hemos creado dos sencillos Dockerfiles: uno que será utilizado dos veces en los servidores Apache y otro que será utilizado en el servidor HAProxy.

Se adjunta un fichero que contiene estos Dockerfiles en la "***apache01***", "***apache02***" y "***haproxy***".

Para construir las imágenes, simplemente accederemos a los distintos directorios y ejecutaremos el comando "***docker build***".

```
cd apache01
```

```
docker build -t miapache01 .
```

```
cd ../apache02
```

```
docker build -t miapache02 .
```

```
cd ../haproxy
```

```
docker build -t mihaproxy .
```

Con esto habremos creado las imágenes locales "***miapache01***", "***miapache02***" y "***mihaproxy***".

A continuación describimos cómo se han creado las distintas imágenes.

2.1 Dockerfile para imágenes de servidores Apache

Las imágenes generadas se basan en la imagen https://hub.docker.com/_/httpd.

Este es el contenido del Dockerfile de "***apache01***" y "***apache02***"

```
FROM httpd:2.4
# Copiamos nuestro index.html al contenedor
COPY index.html /usr/local/apache2/htdocs/index.html
```

La diferencia entre una imagen y otra radica en el contenido del "***index.html***". El de "***apache01***" tiene un mensaje "Servido por: Servidor Apache 01" y el "***apache02***" un mensaje "Servido por: Servidor Apache 02".

2.2 Dockerfile para imagen de servidor HAProxy

La imagen generada se basan en la imagen https://hub.docker.com/_/haproxy.

Este es el contenido del Dockerfile de "***haproxy***"

```
FROM haproxy:1.7
#Copiamos la configuración personalizada de HAProxy al contenedor
COPY haproxy.cfg /usr/local/etc/haproxy/haproxy.cfg
```

Básicamente, lo que se copia dentro del contenedor es el fichero de configuración de HAProxy. Aquí lo podemos observar en detalle.

```
global
      log /dev/log local0
      log localhost local1 notice
      maxconn 2000
      daemon

defaults
      log global
      mode http
      option httplog
      option dontlognull
      retries 3
      timeout connect 5000
      timeout client 50000
      timeout server 50000

frontend http-in
      bind *:80
      default_backend webservers

backend webservers
      stats enable
      stats auth admin:admin
      stats uri /haproxy?stats
      balance roundrobin
      option httpchk
      option forwardfor
      option http-server-close
      server apache1 ${APACHE_1_IP}:${APACHE_EXPOSED_PORT} check
      server apache2 ${APACHE_2_IP}:${APACHE_EXPOSED_PORT} check
```

Este fichero de configuración entre otras cosas indica:

- Que HAProxy funcionará en el puerto 80.
- Que el balanceo de carga seguirá el algoritmo "***roundrobin***" (cada petición a un servidor).
- Que los servidores se definen así:
    - server apache1 ${APACHE_1_IP}:${APACHE_EXPOSED_PORT} check
    - server apache2 ${APACHE_2_IP}:${APACHE_EXPOSED_PORT} check
        - Esto implica, que la IP/HOST del servidor, así como el puerto del servidor, vendrán definidos por esas 3 variables de entorno, que definiremos al lanzar HAProxy con la opción "***-e***" de "***docker run***".

3. PASO 2: CREANDO LA RED Y LOS CONTENEDORES CON APACHE

En primer lugar, creamos la red "***balanceo***" con el siguiente comando:

```
docker network create balanceo
```

A continuación, creamos los servidores Apache dentro de la red "***balanceo***" y les damos como nombres "***ap01***" y "***ap02***". Esos nombres posteriormente serán pasados como IP/HOST para el servidor HAProxy. Los comandos para crearlos son:

```
docker run -it --name ap01 --network balanceo -d miapache01
```

```
docker run -it --name ap02 --network balanceo -d miapache02
```

4. PASO 3: CREANDO CONTENEDOR CON HAPROXY

Creamos el contenedor con HAProxy con nombre "ha01", dentro de la red "***balanceo***" y con el puerto 80 del servidor HAProxy se mapeará con el puerto 8080 de la máquina anfitrión.

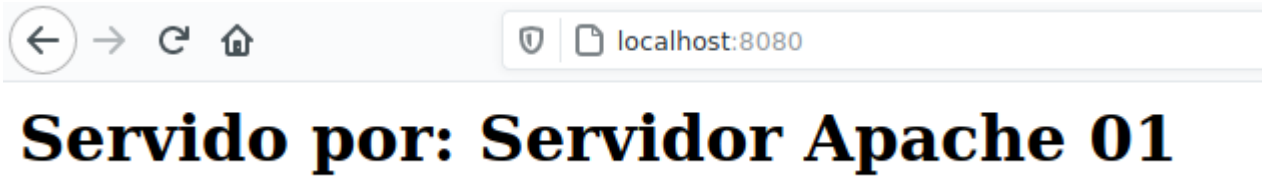
Lo creamos con este contenedor:

```
docker run -it --name ha01 --network balanceo -p 8080:80 -e
APACHE_1_IP=ap01 -e APACHE_2_IP=ap02 -e APACHE_EXPOSED_PORT=80 -d
mihaproxy
```

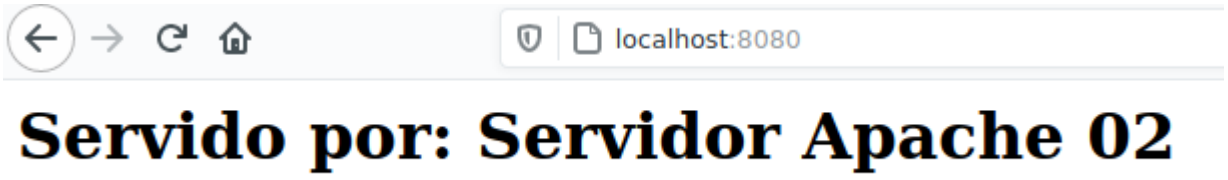
En este contenedor, hemos creado 3 variables de entorno (**APACHE_1_IP**, **APACHE_2_IP**, **APACHE_EXPOSED_PORT**) que serán utilizadas en el servidor HAProxy tal como vimos en su fichero de configuración.

Accediendo en nuestra máquina anfitriona a http://localhost:8080 si todo ha funcionado bien observaremos las siguientes pantallas, que irán cambiando según recarguemos la página, ya que a cada recarga le atenderá a la petición un servidor diferente.

Servidor de contenedor "***ap01***".



Servidor de contenedor "***ap02***".



> ! **Atención:** ¿No puedes ver correctamente como la web es servida por varios servidores? Prueba a recargar la página con CTRL + F5 para evitar recibir la página cacheada. Si no funciona, puedes probar a utilizar el modo incógnito del navegador o en última instancia, pedir la página, borrar la caché y tras ello pedir de nuevo la página para observar que lo sirve otro servidor.

5. SERVIDORES DE APACHE USANDO UN VOLUMEN COMÚN

En este caso práctico, los servidores de cada contenedor servían información distinta. En este paso modificamos un poco lo realizado anteriormente para que usen un volumen común.

En primer lugar, creamos una nueva imagen, usando el Dockerfile proporcionado dentro de la carpeta "***apachevolumen***".

```
cd ../apachevolumen
```

```
docker build -t miapachevolumen .
```

Detenemos y eliminamos los anteriores contenedores "ap01" y ap02"

```
docker stop ap01 ap02
```

```
docker rm ap01 ap02
```

Tras ello, creamos de manera similar a los anteriores un nuevo volumen con esa imagen:

```
docker run -it --name ap01 --network balanceo -v
datosapache:/usr/local/apache2/htdocs/ -d miapachevolumen
```

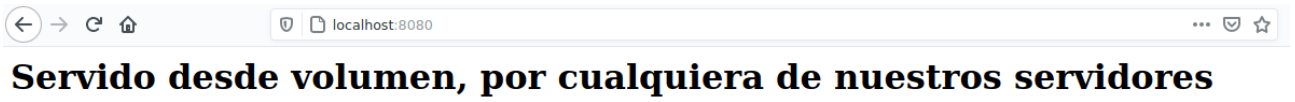
La principal diferencia ante otros cambios es que se crea un volumen llamado "***datosapache***" y se mapea en el directorio del contenedor "***/usr/local/apache/htdocs/***".

Este directorio contiene un fichero "***index.html***". Recordamos que si se crea un volumen nuevo sobre un directorio no vacío, se copia el contenido de ese directorio al volumen (se puebla).

Como ya hemos poblado el volumen, ya no necesitamos usar el Dockerfile y podemos usar la imagen original "***httpd:2.4***" como hacemos aquí. Lo que la hará especial será que se mapeara el mismo volumen que en el otro contenedor.

```
docker run -it --name ap02 --network balanceo -v
datosapache:/usr/local/apache2/htdocs/ -d httpd:2.4
```

Si todo va bien, accediendo en nuestra máquina anfitriona a http://localhost:8080 si todo ha funcionado bien observaremos la siguiente pantalla. Aunque la pantalla será la misma, cada refresco del navegador será servido por un servidor distinto.



6. BIBLIOGRAFÍA

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