



Las Fuentezuelas

Departamento de Informática

Informe técnico:

HAproxy y Wordpress en Docker.

Seguridad y Alta Disponibilidad.

Ciclo Superior de Grado Superior en
Administración de Sistemas Informáticos.

Este documento ha sido realizado únicamente con fines educativos. Se ruega que el uso de los contenidos del mismo sean con el mismo fin, y que no sea copiado.

Jorge Navarrete Secaduras.

Jaén, 3 de febrero de 2022.



Contenido

<i>1</i>	<i>Introducción.....</i>	<i>1</i>
<i>2</i>	<i>Instalación y preparación de archivos.....</i>	<i>1</i>
<i>3</i>	<i>Puesta en marcha</i>	<i>4</i>
<i>4</i>	<i>Configuración final y demostración</i>	<i>5</i>
<i>5</i>	<i>Conclusiones</i>	<i>7</i>

1 Introducción

Este documento detallará el proceso seguido para implantar cuatro frontales web tras un balanceador de carga HAproxy, incrementando así el rendimiento y fiabilidad de nuestra aplicación web gracias a este proxy inverso que distribuirá el tráfico entre los cuatro frontales. Cabe recalcar que, además estos cuatro frontales, a su vez, dependerán de una base de datos MySQL.

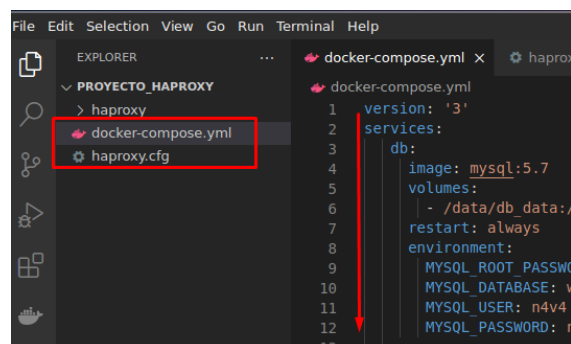
Esta implantación se realizará mediante contenedores Docker, y se lanzará mediante un único archivo Docker Compose, por lo que necesitamos tener instalada esta herramienta en un sistema con, al menos, 2'5GB de RAM. En este caso estaré usando Parrot OS Security, aunque la distribución que usemos es irrelevante.

2 Instalación y preparación de archivos

En primer lugar, procedemos a instalar Docker Compose, aunque no tengamos Docker instalado en el sistema, este comando nos bastará. Recomiendo, también, instalar algún editor de código como Visual Studio Code (code) o similares, puesto que nos facilitará la edición del archivo YAML.

```
> sudo apt install docker-compose code
[sudo] password for n4v4:
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Los paquetes indicados a continuación se instalaron de forma automática y ya no son necesarios.
 libct4 libmotif-common libxm4 python3-atfiles python3-fastapi python3-orjson python3-pydantic python3-slo
Utilice «sudo apt autoremove» para eliminarlos.
Se instalarán los siguientes paquetes adicionales:
 cgroupfs-mount containerd docker.io libintl-perl libintl-xs-perl libmodule-find-perl libmodule-scandeps-pe
python3-dockerpty python3-docopt runc tini
Paquetes sugeridos:
 containernetworking-plugins docker-doc aufs-tools debootstrap rinse rootlesskit zfs-fuse | zfsutils-linux
Paquetes recomendados:
 criu
```

A continuación, desde Visual Studio se han generado los archivos YAML de Docker Compose y el archivo de configuración necesario CFG de HAproxy. Ambos archivos se encontrarán subidos en mi [repositorio de Github](#). No obstante, explicaré ambos en este documento.



El primer archivo, llamado “docker-compose.yml” es el archivo YAML que permite a Docker Compose definir servicios junto a parámetros de configuración específicos, para que, de una forma prácticamente automatizada, se pongan en marcha todos ellos a base de contenedores Docker.

El primer servicio que le indico desde el archivo YAML es el de la base de datos, llamado “db”. Este contenedor usará la imagen de MySQL versión 5.7, le indicamos que la carpeta de “/var/lib/mysql” se encontrará ubicada en el host en “/data/db_data”, y por último le indicamos la variable de espejo para que se guarden las variables necesarias en el contenedor (root_password, database name, user y password).

```
compose.yml
version: '3'
services:
  db:
    image: mysql:5.7
    volumes:
      - /data/db_data:/var/lib/mysql
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: n4v4
      MYSQL_DATABASE: wordpress
      MYSQL_USER: n4v4
      MYSQL_PASSWORD: n4v4
```

Para los Wordpress, el primer parámetro es para indicar que depende del contenedor de la base de datos, seguidamente escribimos la imagen que se usará (wordpress:latest) y se mapean los puertos (externo:interno), abriendo con “expose” el 80. En su variable espejo se le debe indicar el nombre de usuario y contraseña de la base de datos, además del nombre del contenedor y de su puerto.

```
wordpress1:
  depends_on:
    - db
  image: wordpress:latest
  volumes:
    - /data/web_data:/var/www/html
  ports:
    - "17777:80"
  expose:
    - 80
  restart: always
  environment:
    WORDPRESS_DB_HOST: db:3306
    WORDPRESS_DB_USER: n4v4
    WORDPRESS_DB_PASSWORD: n4v4
```

Tras definir los cuatro Wordpress, o el número de frontales que deseemos en general, llegamos al HAproxy. Con el primer parámetro indicamos que depende de los anteriores Wordpress para que estos se inicien antes, enlazamos (conectamos) este contenedor con los cuatro Wordpress, y mapeamos el puerto 8080 del host al 80 del HAproxy, exponiendo este mismo.

```
74 haproxy:
75   depends_on:
76     - wordpress1
77     - wordpress2
78     - wordpress3
79     - wordpress4
80   image: haproxy
81   volumes:
82     - ./haproxy:/usr/local/etc/haproxy
83   links:
84     - wordpress1
85     - wordpress2
86     - wordpress3
87     - wordpress4
88   ports:
89     - "8080:80"
90   expose:
91     - 80
```

Por otro lado, el archivo de configuración de HAproxy es necesario para su funcionamiento. Se basa en cuatro apartados, el primero (global) define los ajustes globales en cuanto a seguridad y rendimiento; el segundo (defaults) permite configurar tiempos de timeout, mode HTTP o HTTPS, conexiones máximas etc.; en el tercero se especifica el frontend que acepta las peticiones de los clientes: y, en el último, los servidores backend.

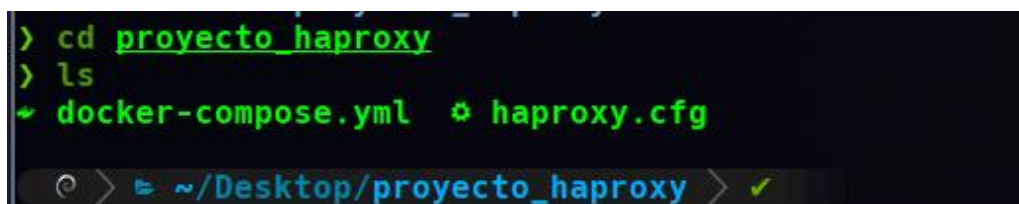
```
1 global
2     maxconn 1024
3     log /dev/log local0
4 defaults
5     log global
6     mode http
7     option httplog
8     option dontlognull
9     timeout connect 5000ms
10    timeout client 50000ms
11    timeout server 50000ms
12    stats uri /status
13 frontend balancer
14     bind 0.0.0.0:80
15     default backend web_backends
16 backend web_backends
17     balance roundrobin
18     server web1 wordpress1:80 check
19     server web2 wordpress2:80 check
20     server web3 wordpress3:80 check
21     server web4 wordpress4:80 check
```

En este caso, al ser un entorno de pruebas, no se ha extendido demasiado el archivo de configuración. En “global” se han establecido únicamente 1024 conexiones máximas para no cargar demasiado la memoria, y también se ha indicado que el log se encuentre en /dev/log. Podemos tener la posibilidad de usar también un server rsyslog remoto.

En el apartado de “defaults” se indica que se usará el modo HTTP, los tiempos de timeout y se usa “log global” para hacer que el frontend use el log que se indicó en la parte superior.

Ulteriormente, comenzamos con la sección de “frontend” haciendo uso de “bind” para permitir la escucha de una petición a cualquier IP y puerto 80, así mismo le indicamos el backend por defecto, llamado en mi caso “mis_backends”. Por último, en el backend indicamos el modo de balance (roundrobin o leastconn) y cada uno de los contenedores Wordpress.

Estos archivos se guardarán en una misma carpeta, desde donde usaremos el comando “docker-compose” para generar los contenedores y poner en marcha los servicios.



```
> cd proyecto_haproxy
> ls
docker-compose.yml  haproxy.cfg
~/Desktop/proyecto_haproxy
```

3 Puesta en marcha

Para poner en marcha los servicios, es tan sencillo como ejecutar el comando “docker-compose up -d” desde la carpeta donde tenemos ubicados los archivos anteriormente editados. Este proceso demorará unos segundos.

```
> sudo docker-compose up -d
Creating network "proyecto_haproxy_default" with the default driver
Pulling db (mysql:5.7)...
5.7: Pulling from library/mysql
6552179c3509: Downloading [=====] 14.56MB/27.15MB
d69aa66e4482: Download complete
3b19465b002b: Download complete
7b0d0cfe99a1: Waiting
9ccd5a5c8987: Waiting
2dab00d7d232: Waiting
64d3afdcd4a: Waiting
6992e58be0f2: Waiting
67313986b81d: Waiting
7c36a23db0a4: Waiting
d34c396e3198: Waiting
Status: Downloaded newer image for haproxy:latest
Creating proyecto_haproxy_db_1 ... done
Creating proyecto_haproxy_wordpress4_1 ... done
Creating proyecto_haproxy_wordpress3_1 ... done
Creating proyecto_haproxy_wordpress1_1 ... done
Creating proyecto_haproxy_wordpress2_1 ... done
Creating proyecto_haproxy_haproxy_1 ... done
~ /Desk/proyecto_haproxy > took 34s
```

En primera instancia podremos observar que el contenedor del HAproxy no se ha iniciado debido a un error, y es que falta su archivo de configuración. Esto se debe a que tenemos que colocar el archivo CFG de su configuración en la carpeta que se nos ha generado automáticamente al hacer el Docker-compose, llamada “haproxy”. Tras ello, iniciamos manualmente el contenedor de HAproxy mediante su nombre o ID, y este estará disponible.

```
> docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS
a40b27f78b18   wordpress:latest "docker-entrypoint.s..." 2 hours ago   Up 2 hours   0.0.0.0:17778->80/tcp
3cb294a36cc6   wordpress:latest "docker-entrypoint.s..." 2 hours ago   Up 2 hours   0.0.0.0:17779->80/tcp
bfff5958daf3   wordpress:latest "docker-entrypoint.s..." 2 hours ago   Up 2 hours   0.0.0.0:17780->80/tcp
26a95255d220   wordpress:latest "docker-entrypoint.s..." 2 hours ago   Up 2 hours   0.0.0.0:17777->80/tcp
5062af385e97   mysql:5.7      "docker-entrypoint.s..." 2 hours ago   Up 2 hours   3306/tcp, 33060/tcp

> sudo cp haproxy.cfg haproxy/
> docker start id
id
> docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS
1d3440b6c15e   haproxy        "docker-entrypoint.s..." 2 hours ago   Up 25 seconds 0.0.0.0:8080->80/tcp
a40b27f78b18   wordpress:latest "docker-entrypoint.s..." 2 hours ago   Up 2 hours   0.0.0.0:17778->80/tcp
3cb294a36cc6   wordpress:latest "docker-entrypoint.s..." 2 hours ago   Up 2 hours   0.0.0.0:17779->80/tcp
bfff5958daf3   wordpress:latest "docker-entrypoint.s..." 2 hours ago   Up 2 hours   0.0.0.0:17780->80/tcp
26a95255d220   wordpress:latest "docker-entrypoint.s..." 2 hours ago   Up 2 hours   0.0.0.0:17777->80/tcp
5062af385e97   mysql:5.7      "docker-entrypoint.s..." 2 hours ago   Up 2 hours   3306/tcp, 33060/tcp
```

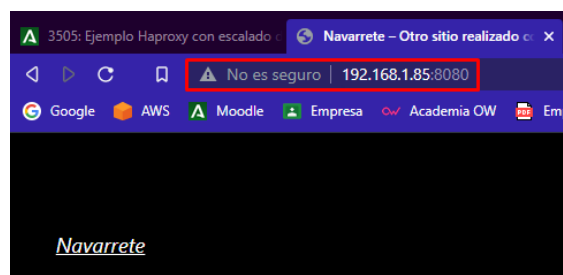
A partir de este momento, la estructura será totalmente funcional. Los cuatro contenedores que harán de frontales web tienen Wordpress, los cuales dependen del contenedor de la base de datos con MySQL 5.7; y HAproxy hará de balanceador de carga entre nuestros Wordpress.

4 Configuración final y demostración

Para confirmar el funcionamiento de toda la implantación, se configurará Wordpress tal y como se hace normalmente. Al haber indicado en su variable espejo la base de datos que usará, únicamente tenemos que poner el nombre de usuario y contraseña para administrar el Wordpress.



Después de esto, como mapeamos el puerto 8080 del host al 80 del contenedor, podremos acceder desde cualquier otra máquina de la red privada escribiendo "http://IP:8080" en el navegador.



En el panel de monitorización de HAProxy podremos observar los cuatro backends, por cada uno de ellos se muestra información respecto a sesiones actuales, sesiones máximas, bytes de entrada y salida, etc.

Statistics Report for HAP. x Navarrete – Otro sitio realiza x +

localhost:8080/status

balancer

	Queue			Session rate			Sessions					Bytes		Denied		Errors		Warnings		Server									
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme
Frontend	0	5	-	2	6		524	266	13				54 751	1 687 824	0	0	1					OPEN							

web_backends

	Queue			Session rate			Sessions					Bytes		Denied		Errors		Warnings		Server									
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme
web1	0	0	-	1	12	0	1	-	23	23	1s	14 369	75 808	0	0	0	0	0	0	0	0	6m33s UP	L4OK in 0ms	1/1	Y	-	0	0	0s
web2	0	0	-	1	11	0	2	-	23	23	1s	13 260	122 813	0	0	0	0	0	0	0	0	6m33s UP	L4OK in 0ms	1/1	Y	-	0	0	0s
web3	0	0	-	1	12	0	3	-	22	22	2s	13 639	1 232 246	0	0	0	0	0	0	0	0	6m33s UP	L4OK in 0ms	1/1	Y	-	0	0	0s
web4	0	0	-	1	12	0	2	-	22	22	1s	12 842	233 329	0	0	0	0	0	0	0	0	6m33s UP	L4OK in 0ms	1/1	Y	-	0	0	0s
Backend	0	0	-	0	47	0	6	52 427	90	90	1s	54 110	1 664 196	0	0	0	0	0	0	0	0	6m33s UP		4/4	4	0	0	0	0s



Después de haber accedido varias veces, podemos observar como la sesión actual de un cliente conectado está en el tercero, y que las últimas peticiones fueron hace 19 segundos para el primero, 10 segundos para el segundo, y 20 segundos para el cuarto.

Para mostrar más claramente cómo funciona HAproxy, añadiré a Wordpress un pequeño código para que nos muestre el hostname del contenedor que visitamos. Para ello usaré el plugin “XYZ PHP code”, el cual me permitirá insertar código PHP de una forma sencilla a una entrada.

Primero se debe crear un “Snippet” con el código desde el menú del propio plugin.

Para conseguir lo que necesito basta con hacer un “echo” de la función “gethostname()”;

Después, en una nueva página o entrada, podemos añadir como “shortcode” el nombre del Snippet que nos proporciona el plugin desde su panel.

Statistics Report for pid 8

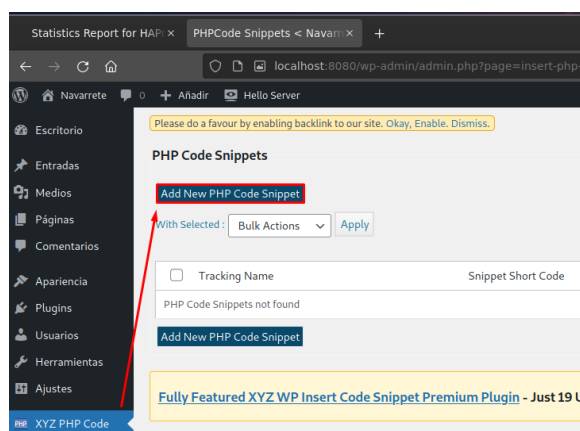
> General process information

pid = 8 (process #1, nbproc = 1, nbthread = 6)
 uptime = 0d 0h 11m 18s
 system limits: memmax = unlimited; ulimit_n = 1048576
 maxsock = 1048576; maxconn = 524266; maxpipes = 0
 current conns = 2; current pipes = 0/0; conn rate = 0/sec; bit rate = 74.503 kbps
 Running tasks: 0/27; idle = 100 %

Note: "N" active active active active active active

balancer		Queue			Session rate			Sessions					By	
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	
Frontend	0	0	-	0	5	-	2	6	-	524 266	24	-	144 874	

web_backends		Queue			Session rate			Sessions					Bytes	
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out
web1	0	0	-	0	16	-	0	2	-	48	48	19s	31 459	4
web2	0	0	-	0	16	-	0	2	-	48	48	10s	29 607	3
web3	0	0	-	1	15	-	0	3	-	48	48	1s	31 429	19
web4	0	0	-	0	16	-	0	3	-	47	47	20s	30 522	6
Backend	0	0	-	1	63	-	0	6	-	52 427	191	1s	123 017	33



Add PHP Snippet

Tracking Name * :

PHP code * :

```
<?php
echo gethostname();
?>
```

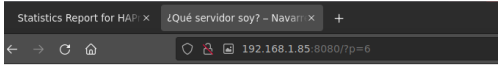
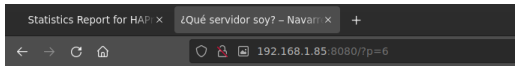
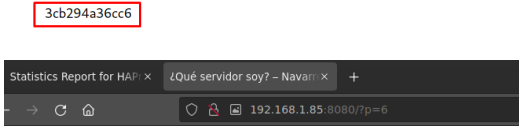
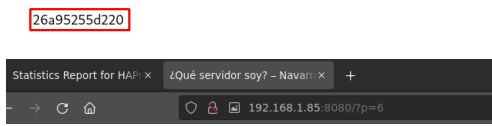
¿Qué servidor soy?

[/] Shortcode

Teclea / para elegir un bloque



Actualizando la misma página de esta entrada podemos observar como el hostname va cambiando por cada petición, ya que HAproxy está balanceando la carga entre los cuatro Wordpress de una forma transparente para el usuario.

	
¿Qué servidor soy?	¿Qué servidor soy?
	
¿Qué servidor soy?	¿Qué servidor soy?

5 Conclusiones

Como se ha demostrado en este documento, mediante Docker Compose podemos realizar proyectos desde cero o a partir de una base de la comunidad para implantar servicios de una forma rápida y sencilla, además de eficaz. Tal y como mencioné en este mismo documento, los archivos necesarios para esta implantación se encuentran en mi [Github](#).

Jorge Navarrete, 2º ASIR.