# Trabajo **COS – Curso 2022/2023 – (3)**

## Distribución de carga con HAproxy

Antes de empezar, sacaremos una instantánea de todas las máquinas del clúster. Se sugiere que la instantánea se llame "preHAproxy".

En la actualidad dado el crecimiento exponencial que ha sufrido Internet, el número de potenciales clientes que un servidor web debe soportar ha crecido también drásticamente. Algunos de estos servidores pueden recibir miles de peticiones simultáneas. Este hecho preocupa las empresas pues no quieren perder conexiones de clientes y además quieren que el servicio que ofrecen sea altamente disponible, es decir, que esté disponible 365 días al año y 24 horas al día. La caída del servicio un día importante, como por ejemplo el día de Navidad, supondría a una empresa como por ejemplo *Amazon*, pérdidas millonarias.

Es importante que el servidor web cuente entre otras con dos características muy importantes: escalabilidad y alta disponibilidad. El sistema debe ser fácilmente escalable para qué, si aumenta la carga, aumenten también los recursos disponibles para no perder conexiones y tener un tiempo de respuesta adecuado. También debemos garantizar que el servicio es altamente disponible, soportando fallos a nivel de hardware y/o software sin que la calidad del servicio se vea afectada.

Un servicio web alojado en un único servidor, no cumple con ninguna de las características comentadas anteriormente, pues si el número de peticiones crece mucho, no será capaz de servirlas y además ante cualquier fallo a nivel software o hardware el servicio dejará de estar disponible.

La solución es usar clústeres de servidores virtuales o físicos, interconectados usando redes de alta velocidad. Es necesario disponer de un mecanismo de reparto de carga (peticiones entrantes) entre todos los nodos que forman el clúster. El sistema es escalable, pues si la carga aumenta, el administrador aumenta el número de nodos dedicados a servir peticiones y es altamente disponible al disponer de varios nodos sirviendo peticiones. Para que el servicio dejase de funcionar deberían caer todos los nodos simultáneamente.

El software encargado de realizar el reparto de carga es el balanceador de carga *(Load Balancer).* Los dos balanceadores de carga de código abierto más usados son: *Linux Virtual Server (LVS)* y *HAproxy*. En este proyecto se propone el uso de *HAproxy*. Actualmente dispone de una versión de pago para empresas que incluye soporte técnico.

Su uso más común es balancear el tráfico *HTTP* y *HTTPS* dirigido a un servidor web. Puede balancear carga en el nivel 4 o en el nivel 7 del modelo *OSI*.

La arquitectura del sistema propuesto puede apreciarse en la figura 1, donde se distinguen 3 componentes principales:

- **Load Balancer**: Es el encargado de recibir las peticiones entrantes *(front end)* y repartirlas entre los servidores de la granja *(back end).* Los usuarios conocen una única dirección *IP* que es la que está asociada al servicio y que se conoce cómo *IP virtual*. Una vez recibida la petición el balanceador sustituye la IP virtual por la IP real de alguno de los servidores y le reenvía los paquetes.

- **Server farm**: Esta formada por un conjunto de servidores reales que implementan los servicios ofrecidos por el clúster, como por ejemplo web, ftp, mail, dns, etc. Las direcciones IP de estos servidores son privadas.

- **Storage**: Ofrece a los servidores de la granja un espacio de almacenamiento compartido, de esta manera todos los servidores ofrecen a los clientes la misma información actualizada.

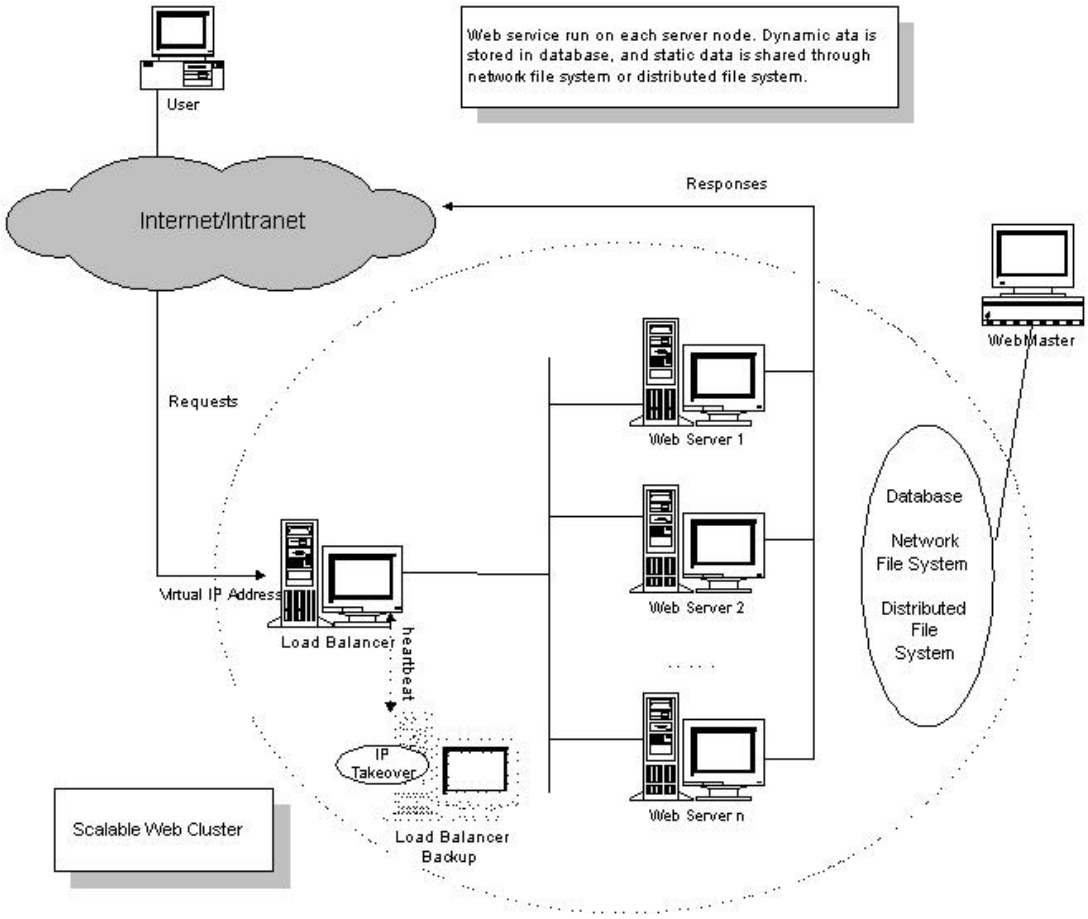



Figura 1. Arquitectura servidor web

## 1. Instalación y configuración de HAproxy

*HAproxy (High Availability proxy)* es un balanceador de carga y *proxy* de código abierto para servidores TCP/HTTP con sistema operativo Linux. Entre otras características incluye comprobación automática de la salud de los servidores de la granja *(automatic health check)*, un conjunto amplio de algoritmos de balanceo de carga, soporte para HTTPS/SSL, etc.

Vamos a instalar y configurar *HAproxy* en el nodo *Master*. Este repartirá las peticiones *HTTP* entrantes entre los tres servidores de la granja a través de la red interna, en función del algoritmo de balanceo de carga elegido. El servidor seleccionado, que tiene configurado un servidor *http Apache*, accede al directorio compartido del nodo de almacenamiento, obtiene el fichero *htmt, php,* etc y

responde al cliente a través del nodo master.

Vamos a ver que versión de *Haproxy* es la disponible en los repositorios oficiales de *CentOS* y procedemos a instalarla en el nodo *Master*.

```
# yum info haproxy
# yum –y install haproxy
```

Para la configuración asumimos que disponemos de tres servidores con *Apache* instalado y direcciones *IP privadas, 10.0.100.11, 10.0.100.12, 10.0.100.13*. También asumimos que la dirección *IP* del nodo master es *192.168.1.101*.

Para ponerlo en funcionamiento debemos acceder al fichero */etc/haproxy/haproxy.cfg* y establecer los valores apropiados en los distintos parámetros para nuestro entorno.

- `# nano /etc/haproxy/haproxy.cfg`

```
global
log 127.0.0.1 local2
chroot              /var/lib/haproxy
pidfile             /var/run/haproxy.pid
maxconn 3000
user haproxy
group haproxy
stats timeout 30s
daemon
stats socket /var/lib/haproxy/stats
defaults
mode                    http
log                     global
option                  httplog
option                  dontlognull
option                  http-server-close
option forwardfor       except 127.0.0.0/8
option                  redispatch
retries                       3
timeout http-request          10s
timeout queue                 1m
timeout connect         10s
timeout client                1m
timeout server          1m
timeout http-keep-alive 10s
timeout check                 10s
```

```
frontend http_front
      bind 192.168.1.101:80
      stats enable
      stats uri /haproxy_stats
      stats auth admin:haproxy
      default_backend http_back
backend http_back
      balance roundrobin
      server cluster2 10.0.100.12:80  weight 1     maxconn 1000 check
      server cluster3 10.0.100.13:80  weight 1     maxconn 1000 check
      server cluster4 10.0.100.14:80  weight 1     maxconn 1000 check
```

Una vez hemos configurado el servicio lo arrancamos y habilitamos para que se inicie siempre que arranque el nodo.

```
# systemctl start haproxy
# systemctl enable haproxy
# systemctl status haproxy –l
```

En el fichero de configuración de *Haproxy* le hemos indicado que escuche en la dirección *IP 192.168.1.101* y en el puerto 80. Al comprobar el estado del servicio nos damos cuenta de que ha dado un error, pues hay otro servicio escuchando en el puerto 80, *Apache*. Cuando se instaló Apache en el nodo Master, por defecto se pone a escuchar en el puerto 80 de todas las interfaces de red, por lo que hay que hacer un ajuste en la configuración de este servicio, para que solo escuche en el puerto 80 de la dirección IP asociada a la red interna 10.0.100.1.

Editamos el fichero de configuración */etc/httpd/conf/httpd.conf y* sustituimos*:*

```
Listen 80
```

*Por:*

```
Listen 10.0.100.1:80
```

Y reiniciamos el servicio para que los cambios surtan efecto:

```
# systemctl restart httpd
```

Una vez liberado el puerto 80 hay que configurar el cortafuegos para que no bloquee el trafico al puerto 80 en la zona externa, que es donde está conectada la tarjeta de red de la *red NAT*.

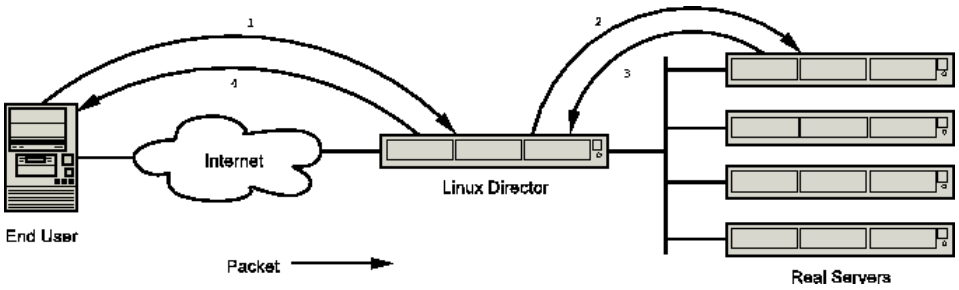
En primer lugar, comprobamos que la tarjeta de red *enp0s8* (red interna) está conectada a la zona interna y la tarjeta de red *enp0s3* (red NAT) está conectada a la zona externa.

```
# firewall-cmd --get-active-zones
```

Una vez comprobado, le indicamos al cortafuegos que no bloquee el tráfico al puerto 80 en la zona externa, pues es donde escuchará el servicio *Haproxy*.

```
# firewall-cmd --zone=external --add-port=80/tcp --permanent
# firewall-cmd --reload
```

Los paquetes de los clientes llegan al **master** donde la dirección VIP (Virtual IP) se cambia por la dirección RIP (Real IP) de uno de los **servidores reales**. A su vez, las respuestas de los servidores reales pasan por el master que sustituye su dirección RIP por la dirección VIP.




En esta sesión, vamos a realizar una configuración básica de un clúster HAproxy. El clúster ofrecerá un servicio web (HTTP).

Usaremos el nodo maestro `cluster1` como director y el resto de nodos (`cluster2, cluster3` y `cluster4`) como servidores reales. Como cliente del servicio web, usaremos la máquina física (`host`) donde se ubican las máquinas virtuales. Para ello, debemos habilitar el acceso al puerto `80` de `cluster1` desde un puerto (por ejemplo el `5080`) de la máquina física (`host`).

Vamos a cambiar la dirección *IP* que tiene el nodo master asociado al adaptador *enp0s3* conectado a la *red-nat-1:*

`vi /etc/sysconfig/network-scripts/ifcfg-enp0s3:`

```
TYPE=Ethernet
BOOTPROTO=static
NAME=enp0s3
DEVICE=enp0s3
ONBOOT=yes
UUID=b42bf641-b67d-4360-aef2-e20fdb9b23d3
IPADDR=192.168.1.101
NETMASK=255.255.255.0
GATEWAY=192.168.1.1
DNS1=8.8.8.8
NM_CONTROLLED=no
```

Una vez realizados los cambios reiniciamos el servicio:

```
# systemctl restart network
```

Comprobamos los cambios:

```
# ip addr show
```

Para finalizar comprobamos que seguimos teniendo conexión con el mundo exterior

```
# ping www.upv.es
```

Una vez conocida la dirección *IP*, podemos hacer la redirección de puertos en las opciones de configuración de la *Red NAT*. Se puede realizar desde el interfaz gráfico de *VirtualBox* (Preferencias→Red), o mediante la línea de órdenes del host anfitrión.

```
vboxmanage natnetwork modify --netname red-nat-1 --port-forward-4
“http:tcp:[]:5080:[192.168.1.101]:80”
```

En este caso hemos redirigido el tráfico al puerto 5080 del host, al puerto 80 de la máquina virtual master con IP 192.168.1.101. Una vez configurado el servicio *http*, podemos solicitar desde el exterior la URL http://direccion_IP_host:5080 o desde un navegador en el propio host http://localhost:5080.

En resumen, las direcciones IP implicadas son:

| | Nombre | Dirección IP | |
|---|---|---|---|
| Director | `cluster1` `master` | IP: `10.0.100.1` `(enp0s8)` | IP: `192.168.1.101` `(enp0s3)` |
| Servidores Reales | `cluster2` `cluster3` `cluster4` | IP: `10.0.100.12` `10.0.100.13` `10.0.100.14` `(enp0s3)` | |
| | | | |

## 2. [cluster2-4] Instala apache2 en los servidores reales

En el proceso de instalación de los servidores se instalo en cada uno de ellos el servidor web *Apache (httpd).*

El directorio por defecto donde *Apache* busca los ficheros *html* es`/var/www/html` y el fichero que entrega al navegador es *index.html*, salvo que se indique otro distinto. Inicialmente, modificaremos este documento en cada servidor para visualizar qué servidor está dando el servicio y así comprobar el funcionamiento. Para ello, cada servidor incluirá su nombre en la página de inicio. Esto es provisional, ya que nuestro objetivo final es que todos los servidores sirvan los mismos contenidos ubicados en un espacio de almacenamiento compartido por NFS.

En cada uno de los servidores haremos lo siguiente

```
# echo "<html><body>Pagina de prueba para COS. Servidor i </body></html>"
> /var/www/html/index.html
```

Para reiniciar el servicio `apache`, lanzaremos en cada servidor:

```
# systemctl restart httpd
```

Podemos comprobar que todo funciona con un navegador en modo texto:

```
# w3m localhost
```

O con

```
# curl localhost
```

## 5. [host] Genera tráfico y comprueba el funcionamiento de HAproxy

Necesitamos generar tráfico desde "fuera del clúster", dirigido a la dirección VIP (192.168.1.101). Como hemos hecho un túnel entre el puerto 5080 de la máquina física y el puerto 80 de la máquina virtual `cluster1`, podemos generar tráfico desde la máquina física (host) accediendo con un navegador (o mediante las órdenes w3m o curl , si están disponibles) a la URL http://localhost:5080 o desde otra máquina del laboratorio a la dirección http://direccion_IP_host:5080.

## 6. [cluster2-4] Cambio de página de inicio

Ahora modifica la configuración de los servidores para que la localización de los contenidos que todos ellos sirven esté ubicada en un espacio de almacenamiento compartido. En nuestro caso, será `/mnt/web` para ello debemos modificar el fichero */etc/httpd/conf/httpd.conf*

y cambiar

```
DocumentRoot /var/www/html
```

por

```
DocumentRoot /mnt/web
```

Y

```
<Directory /var/www>
. . . .
</Directory>
```

por

```
<Directory /mnt/web>
. . . .
</Directory>
```

En ese directorio *(/mnt/web)* es donde crearemos una página web de prueba, por ejemplo:

```
echo "<html><body>Pagina de prueba para COS. NFS</body></html>"
>/mnt/web/index.html
```

Finalmente, queda reiniciar el servicio:

```
# systemctl restart httpd
```

Comprueba, con ayuda del navegador desde los tres servidores que la nueva página de inicio es accesible. Ahora los tres servidores sirven la misma página.

Sin embargo, con esta nueva página de inicio no es posible saber qué servidor físico la está suministrando. Además, como el contenido es estático, no se aprecia bien cuando se recarga la página. La solución pasa por ofrecer un contenido dinámico. En nuestro caso, lo haremos con la ayuda de *PHP*.

Generaremos una sencilla página (`/mnt/web/index.php`) con el siguiente contenido:

```
<html>
<body>
Pagina de prueba para COS. NFS
<?php
date_default_timezone_set(´CET´);
echo “Hoy es “.date(“d/m/y”).“<br>”;
echo “Son las “.date(“h:i:s”).“<br>”;
echo “Soy el servidor “.$_SERVER[‘SERVER_ADDR’].“<br>”;
?>
</body>
</html>
```

Podemos comprobar la nueva página abriendo en el navegador, en la máquina física *(`host`)* la URL `http://localhost:5080/index.php`

Con *HAproxy* configurado y en ejecución, es posible acceder a la página de estadísticas accediendo con un navegador a la página *http://localhost:5080/haproxy_stats*. Se ha configurado esa dirección en el fichero de configuración con *stats uri*. Para poder visualizar su contenido hay que identificarse con el usuario y contraseña establecidos con *stats auth*.

Una vez cargada la página de estadísticas, si todos los servidores de la granja aparecen en verde, es que la configuración es correcta y el servicio esta funcionando sin problemas. Si algún servidor aparece en rojo, habría que analizar cuál es el problema, comprobando que se puede acceder a él mediante *ping*, comprobar que *apache* esta corriendo y escuchando en el puerto 80, que los servidores tienen acceso al directorio compartido *NFS*. etc.