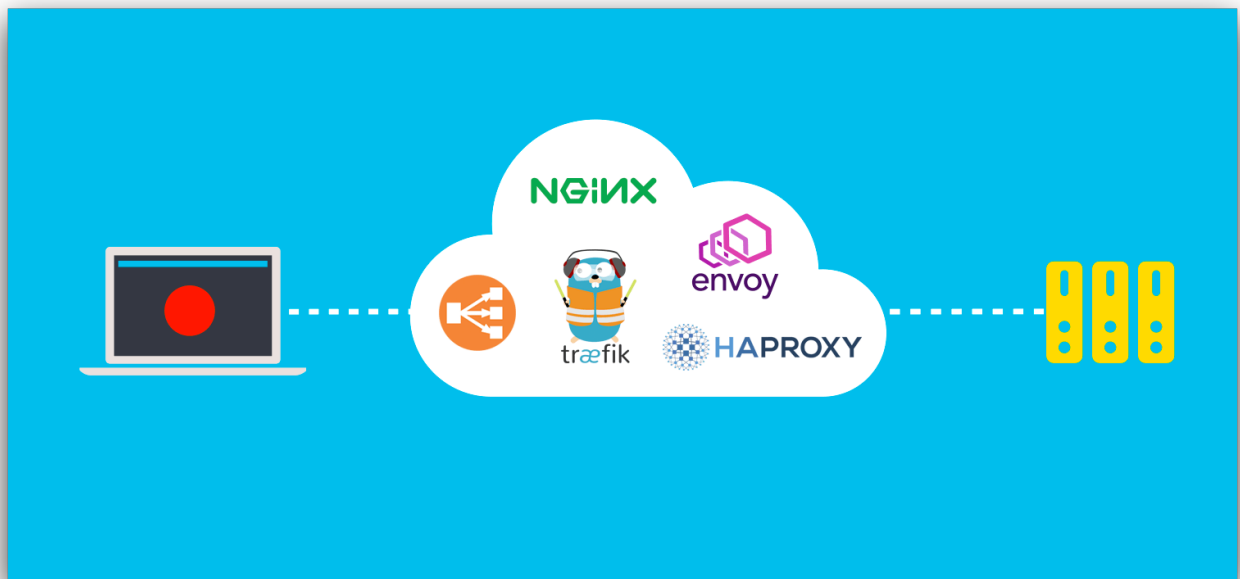


# BALANCEO DE CARGA EN UN SERVIDOR WEB

*Servidores Web de Altas Prestaciones - Práctica 3*



**Adrián Acosa Sánchez**

9/4/2022

3º Ingeniería Informática (Tecnologías de la Información)

## ÍNDICE

ÍNDICE	1
<b>NGINX</b>	<b>2</b>
Instalación de nginx	2
Balanceo de carga usando nginx	3
<b>HAProxy</b>	<b>8</b>
Instalación de HAProxy	8
Balanceo de carga usando HAProxy	9
Funcionamiento del balanceador HAProxy	10
Asignación de pesos a cada servidor con HAProxy	11
Habilitar estadísticas del balanceador	13
<b>gobetween</b>	<b>16</b>
Instalación de gobetween	16
Configuración del balanceador gobetween	17
<b>Zevenet</b>	<b>20</b>
Instalación de zevenet	20
Configuración del balanceador zevenet	21
<b>Pound</b>	<b>29</b>
Instalación de pound	29
<b>Someter a una alta carga el servidor balanceado</b>	<b>33</b>
Análisis comparativo	33
<b>BIBLIOGRAFÍA</b>	<b>36</b>

## NGINX

### Instalación de nginx

Comenzaremos la práctica instalando el paquete nginx en la máquina proxy *m3*. Antes de instalarlo vamos a actualizar el sistema para obtener la última versión del paquete de nginx:

```
adrianacosa@m3-adrianacosa:~$ sudo apt-get update && sudo apt-get dist-upgrade && sudo apt-get autoremove
[sudo] password for adrianacosa:
Hit:1 http://es.archive.ubuntu.com/ubuntu focal InRelease
Get:2 http://es.archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Get:3 http://es.archive.ubuntu.com/ubuntu focal-backports InRelease [108 kB]
Get:4 http://es.archive.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Get:5 http://es.archive.ubuntu.com/ubuntu focal-updates/main amd64 Packages [1,680 kB]
Get:6 http://es.archive.ubuntu.com/ubuntu focal-updates/main Translation-en [317 kB]
Get:7 http://es.archive.ubuntu.com/ubuntu focal-updates/main amd64 c-n-f Metadata [14.8 kB]
Get:8 http://es.archive.ubuntu.com/ubuntu focal-updates/restricted amd64 Packages [893 kB]
Get:9 http://es.archive.ubuntu.com/ubuntu focal-updates/restricted Translation-en [127 kB]
Get:10 http://es.archive.ubuntu.com/ubuntu focal-updates/universe amd64 Packages [913 kB]
Get:11 http://es.archive.ubuntu.com/ubuntu focal-updates/universe amd64 c-n-f Metadata [20.3 kB]
Get:12 http://es.archive.ubuntu.com/ubuntu focal-backports/universe amd64 Packages [22.7 kB]
Get:13 http://es.archive.ubuntu.com/ubuntu focal-backports/universe Translation-en [15.5 kB]
Get:14 http://es.archive.ubuntu.com/ubuntu focal-security/main amd64 Packages [1,349 kB]
Get:15 http://es.archive.ubuntu.com/ubuntu focal-security/main Translation-en [235 kB]
Get:16 http://es.archive.ubuntu.com/ubuntu focal-security/main amd64 c-n-f Metadata [9,804 B]
Get:17 http://es.archive.ubuntu.com/ubuntu focal-security/restricted amd64 Packages [832 kB]
Get:18 http://es.archive.ubuntu.com/ubuntu focal-security/restricted Translation-en [118 kB]
Get:19 http://es.archive.ubuntu.com/ubuntu focal-security/universe amd64 Packages [695 kB]
Fetched 7,577 kB in 4s (1,784 kB/s)
Reading package lists... Done
Reading package lists... Done
Building dependency tree
Reading state information... Done
Calculating upgrade... Done
The following package was automatically installed and is no longer required:
  libfwupdplugin1
Use 'sudo apt autoremove' to remove it.
The following NEW packages will be installed:
  libfwupdplugin5 libbim-glib4 libbim-proxy libmm-glib0 libqmi-glib5 libqmi-proxy modemmanager usb-modeswitch
  usb-modeswitch-data
The following packages will be upgraded:
  alsa-ucm-conf cloud-init command-not-found fwupd fwupd-signed libfwupd2 libjcat1 libnetplan0 netplan.io open-vm-tools
  python3-commandnotfound rsync sosreport
13 upgraded, 9 newly installed, 0 to remove and 0 not upgraded.
Need to get 6,525 kB of archives.
After this operation, 12.3 MB of additional disk space will be used.
Do you want to continue? [Y/n] _
```

Una vez tenemos actualizado y limpio el sistema, procedemos a la instalación:

```
adrianacosa@m3-adrianacosa:~$ sudo apt-get install nginx
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  fontconfig-config fonts-dejavu-core libfontconfig1 libgd3 libjbig0 libjpeg-turbo8 libjpeg8 libnginx-mod-http-image-filter
  libnginx-mod-http-xslt-filter libnginx-mod-mail libnginx-mod-stream libtiff5 libwebp6 libxpm4 nginx-common nginx-core
Suggested packages:
  libgd-tools fcgiwrap nginx-doc ssl-cert
The following NEW packages will be installed:
  fontconfig-config fonts-dejavu-core libfontconfig1 libgd3 libjbig0 libjpeg-turbo8 libjpeg8 libnginx-mod-http-image-filter
  libnginx-mod-http-xslt-filter libnginx-mod-mail libnginx-mod-stream libtiff5 libwebp6 libxpm4 nginx nginx-common nginx-core
0 upgraded, 17 newly installed, 0 to remove and 0 not upgraded.
Need to get 2,432 kB of archives.
After this operation, 7,891 kB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Y por último antes de usarlo, tenemos que habilitarlo para que en cada inicio del sistema

arranque el servicio e iniciarlo para poder ejecutarlo instantáneamente:

```
adrianacosa@m3-adrianacosa:~$ sudo systemctl enable nginx && sudo systemctl start nginx
Synchronizing state of nginx.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable nginx
adrianacosa@m3-adrianacosa:~$ _
```

### Balanceo de carga usando nginx

Ahora vamos a pasar a configurar el balanceador de carga. Como se comenta en el gui3n, la configuraci3n b3sica de nginx no nos vale ya que ejerce la funci3n de servidor web, cosa que no nos interesa. Para evitar este comportamiento tenemos que comentar la l3nea que aparece en la siguiente imagen del archivo de configuraci3n de nginx que se encuentra en el la ruta */etc/nginx/nginx.conf*:

```
include /etc/nginx/conf.d/*.conf;
#include /etc/nginx/sites-enabled/*;
```

Para empezar a configurar nuestro balanceador de carga, creamos el archivo */etc/nginx/conf.d/default.conf* y escribimos la siguiente informaci3n:

```
upstream balanceo_adrianacosa{
    server 192.168.122.43
    server 192.168.122.36
}

server{
    listen 80;
    server_name balanceador_adrianacosa;

    access_log /var/log/nginx/balanceador_adrianacosa.access.log;
    error_log /var/log/nginx/balanceador_adrianacosa.error.log;
    root /var/www/;

    location /
    {
        proxy_pass http://balanceo_adrianacosa;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_http_version 1.1;
        proxy_set_header Connection "";
    }
}
```

Donde en primer lugar, en upstream, definimos qu3 m3quinas formaran el cluster web

poniendo las IP de las dos máquinas que contienen los servidores que en este caso son m1 y m2 respectivamente. Una vez hecho esto, en la sección *server* debemos indicarle al *nginx* que ese grupo van a ser los servidores a los que va a repartirles la carga. Para que ésto funcione correctamente, es importante indicar que la conexión entre *nginx* y los servidores finales sea mediante el protocolo *HTTP* con la versión 1.1, así como especificarle que elimine la cabecera *Connection* para evitar que se pase al servidor final la cabecera que indica el usuario.

Esta es la configuración básica del balanceador, que por defecto usa un algoritmo *Round-Robin* en la que ambas máquinas tienen la misma prioridad. Si queremos asignarle más prioridad a una máquina que a otra (en el caso de que tenga más prestaciones un servidor que otro conviene asignarle una prioridad mayor al de mayores prestaciones), se puede hacer añadiendo la variable *weight* a la derecha de los servidores web e indicando que la máquina que tenga un mayor peso sea la que más peticiones reciba. La configuración sería la que aparece en la siguiente imagen:

```
upstream balanceo_adrianacosa{
    server 192.168.122.43 weight=2;
    server 192.168.122.36 weight=1;
}
```

En este caso, por cada 3 peticiones, la máquina 2 recibiría 1 petición y las otras dos las recibiría la máquina 1 (aquí suponemos que la máquina 1 tiene el doble de capacidad que la máquina 2).

Para aplicar la configuración lo único que tenemos que hacer es reiniciar el servicio de *nginx* y probar que se hace un reparto de las peticiones:

```
adrian ~ curl http://192.168.122.67
<!DOCTYPE html>
<html>
  <head>
    <title>MAQUINA 2</title>
  </head>
  <body>
    <h1>MAQUINA 2</h1>
    <p>Ahora mismo te encuentras en la maquina 2</p>
  </body>
</html>
adrian ~ curl http://192.168.122.67
<!DOCTYPE html>
<html>
  <head>
    <title>MAQUINA 1</title>
  </head>
  <body>
    <h1>MAQUINA 1</h1>
    <p>Ahora mismo te encuentras en la maquina 1</p>
  </body>
</html>
```

Podemos comprobar que asignando un peso el doble de mayor a la máquina 1 que a la 2, si hacemos 3 peticiones 2 de ellas irán a la máquina 1 y una de ellas a la 2 como aparece en la siguiente imagen:

```
adrian ~ curl http://192.168.122.67
<!DOCTYPE html>
<html>
  <head>
    <title>MAQUINA 2</title>
  </head>
  <body>
    <h1>MAQUINA 2</h1>
    <p>Ahora mismo te encuentras en la maquina 2</p>
  </body>
</html>
adrian ~ curl http://192.168.122.67
<!DOCTYPE html>
<html>
  <head>
    <title>MAQUINA 1</title>
  </head>
  <body>
    <h1>MAQUINA 1</h1>
    <p>Ahora mismo te encuentras en la maquina 1<p>
  </body>
</html>
adrian ~ curl http://192.168.122.67
<!DOCTYPE html>
<html>
  <head>
    <title>MAQUINA 1</title>
  </head>
  <body>
    <h1>MAQUINA 1</h1>
    <p>Ahora mismo te encuentras en la maquina 1<p>
  </body>
</html>
```

Hay que tener en cuenta que nos interesa que todas las peticiones provenientes de una misma IP deberían ir hacia la misma máquina siempre, tengan el peso que tengan los servidores web, por temas de mantener la información asociada a esa IP en el momento de volverse a conectar tras una conexión anterior. Para ello existe la opción *ip\_hash* que, situándola en la sección *upstream*, codifica la IP para realizar la tarea anteriormente mencionada.

```
upstream balanceo_adrianacosa{
    ip_hash;
    server 192.168.122.43 weight=2;
    server 192.168.122.36 weight=1;
```

Pero esta configuración tiene un problema, y es que para todas las IP privadas que hayan sufrido una traducción NAT (por ejemplo, todos los usuarios de una red doméstica que pasen por el mismo router) van a ir dirigidos a la misma máquina y todos van a conservar la misma información que el primero que hizo la petición al servidor.

También existe una configuración de nginx que nos permite utilizar conexiones *keepalive* entre nginx y los servidores web finales, con el fin de que se realice una conexión con una persistencia de múltiples peticiones HTTP en vez de tener que abrir una nueva conexión cada vez que se realiza una petición. Existen tres posibilidades para configurar el *keepalive* los cuales son: *keepalive* (número de peticiones seguidas), *keepalive\_time* (tiempo donde se realizan varias peticiones seguidas) y *keepalive\_timeout* (tiempo límite de espera para aceptar nuevas peticiones).

Otro algoritmo que podríamos usar para balancear la carga entre nuestros servidores web finales sería el de Least-connected, que permite que las instancias de una misma aplicación se puedan distribuir de manera más justa cuando algunas de las peticiones de ésta tardan más en completarse. Con ésta directiva, nginx intenta no sobrecargar el servidor con numerosas peticiones distribuyendo las nuevas peticiones que recibe hacia aquellas máquinas menos ocupadas. Para activar ésta directiva, la añadimos de la siguiente manera:

```
upstream balanceo_adrianacosa{
    least_conn;
    server 192.168.122.43 weight=2;
    server 192.168.122.36 weight=1;
}
```

Por último, una configuración recomendable es ajustar el parámetro *max\_fails* según como queramos que se filtren las peticiones que han fallado al llegar a nuestro servidor un número determinado de veces seguidas. Por defecto ésta configuración tiene asignado que sólo se pueda intentar la comunicación una sólo vez. Si queremos que no se compruebe éste parámetro, tendríamos que ajustarlo a 0. Si queremos permitir que se puedan producir varios fallos seguidos, simplemente tenemos que ajustar el parámetro a la cantidad de fallos que queramos.

Por ejemplo, si queremos que en nuestro servidor m1 sólo se permitan 5 fallos consecutivos y en el servidor m2 3 fallos consecutivos, tendríamos que escribir lo siguiente en el archivo de configuración:

```
upstream balanceo_adrianacosa{
    least_conn;
    server 192.168.122.43 weight=2 max_fails=5;
    server 192.168.122.36 weight=1 max_fails=3;
}
```

Como última opción a nombrar, tenemos la de *slow\_start*. Éste parámetro nos permite ajustar el tiempo durante el que el servidor irá recuperando su peso desde 0 hasta el valor que tenga asignado tras periodo de tiempo en el que el servidor ha dejado de estar disponible. Por defecto está asignado a 0, es decir, no está habilitado. Si quisiéramos habilitarlo, tendríamos que asignarle el tiempo deseado para recuperar el peso. Suponiendo que la máquina 1 se recupera mucho más rápido que la máquina 2, vamos a asignarle un *slow\_start* mayor a m2 que a m1:

```
upstream balanceo_adrianacosa{
    least_conn;
    server 192.168.122.43 weight=2 slow_start=30s;
    server 192.168.122.36 weight=1 slow_start=60s;
}
```



## HAProxy

### Instalación de HAProxy

Para instalar HAProxy lo hacemos con el comando de instalación de paquetes de ubuntu con el siguiente comando:

```
adrianacosa@m3-adrianacosa:~$ sudo apt-get install haproxy
[sudo] password for adrianacosa:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  liblua5.3-0
Suggested packages:
  vim-haproxy haproxy-doc
The following NEW packages will be installed:
  haproxy liblua5.3-0
0 upgraded, 2 newly installed, 0 to remove and 0 not upgraded.
Need to get 1,635 kB of archives.
After this operation, 3,778 kB of additional disk space will be used.
Do you want to continue? [Y/n] _
```

Una vez instalado, para cambiar el balanceador de carga usado por la máquina 3 lo primero que tenemos que hacer es deshabilitar el uso de nginx mediante el uso del comando `systemctl stop nginx` y `systemctl disable nginx`:

```
adrianacosa@m3-adrianacosa:~$ sudo systemctl stop nginx && sudo systemctl disable nginx
Synchronizing state of nginx.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install disable nginx
Removed /etc/systemd/system/multi-user.target.wants/nginx.service.
adrianacosa@m3-adrianacosa:~$
```

Y acto seguido habilitar e iniciar el servicio que nos ofrece HAProxy:

```
adrianacosa@m3-adrianacosa:~$ sudo systemctl enable haproxy && sudo systemctl start haproxy
Synchronizing state of haproxy.service with SysV service script with /lib/systemd/systemd-sysv-install
Executing: /lib/systemd/systemd-sysv-install enable haproxy
adrianacosa@m3-adrianacosa:~$ S
```

### Balanceo de carga usando HAProxy

Siguiendo el mismo patrón que con nginx, tendremos que modificar el archivo de configuración situado en `/etc/haproxy/haproxy.cfg` e indicarles los servidores a los que tendrá que balancear las peticiones.

Para realizar esta tarea, introducimos la siguiente configuración en el archivo comentado:

```
frontend http-in
    bind *:80
    default_backend balanceo_adrianacosa

backend balanceo_adrianacosa
    server m1 192.168.122.43:80 maxconn 32
    server m2 192.168.122.36:80 maxconn 32_
```

Donde le indicamos las ip de ambas máquinas escuchando en el puerto 80.

## Funcionamiento del balanceador HAProxy

Antes de comenzar con las comprobaciones, tenemos que guardar las configuraciones y reiniciar el servicio de HAProxy:

```
adrianacosa@m3-adrianacosa:~$ sudo /usr/sbin/haproxy -f /etc/haproxy/haproxy.cfg
adrianacosa@m3-adrianacosa:~$ sudo service haproxy restart
adrianacosa@m3-adrianacosa:~$
```

Y comprobamos cómo se realiza el balanceo desde mi PC:

```
adrian ~$ curl http://192.168.122.67
<!DOCTYPE html>
<html>
  <head>
    <title>MAQUINA 1</title>
  </head>
  <body>
    <h1>MAQUINA 1</h1>
    <p>Ahora mismo te encuentras en la maquina 1</p>
  </body>
</html>
adrian ~$ curl http://192.168.122.67
<!DOCTYPE html>
<html>
  <head>
    <title>MAQUINA 2</title>
  </head>
  <body>
    <h1>MAQUINA 2</h1>
    <p>Ahora mismo te encuentras en la maquina 2</p>
  </body>
</html>
adrian ~$
```

Y vemos como al realizar la misma petición al balanceador varias veces, unas veces las balancea a la máquina 1 y otras a la máquina 2.

### Asignación de pesos a cada servidor con HAProxy

Para realizar la misma configuración anteriormente hecha en nginx en la que asignamos un mayor peso a la máquina 1 que a la máquina 2, se hace añadiendo la palabra *weight* al final de cada declaración de los servidores web finales en el archivo de configuración. Por tanto quedaría de la siguiente manera:

```
backend balanceo_adrianacosa
    server m1 192.168.122.43:80 maxconn 32 check weight 100
    server m2 192.168.122.36:80 maxconn 32 check weight 50
```

Añadiendo al final *weight* y un porcentaje, podemos decirle a HAProxy la cantidad de peticiones que queremos que se lleve un servidor u otro por cada 100 peticiones. En este caso le hemos asignado que el servidor *m1* reciba el doble de peticiones que el servidor *m2*.

También podemos fijarnos en que en ambas máquinas le hemos añadido la opción *check*, la cual se encargará de proporcionar el estado de la máquina en cada instante y comprobar si se encuentra disponible o no el servidor web.

El resultado obtenido, aunque es el mismo que para nginx, lo muestro en la siguiente captura de pantalla:

```
adrian ~ curl 192.168.122.67
<!DOCTYPE html>
<html>
  <head>
    <title>MAQUINA 1</title>
  </head>
  <body>
    <h1>MAQUINA 1</h1>
    <p>Ahora mismo te encuentras en la maquina 1<p>
  </body>
</html>
adrian ~ curl 192.168.122.67
<!DOCTYPE html>
<html>
  <head>
    <title>MAQUINA 1</title>
  </head>
  <body>
    <h1>MAQUINA 1</h1>
    <p>Ahora mismo te encuentras en la maquina 1<p>
  </body>
</html>
adrian ~ curl 192.168.122.67
<!DOCTYPE html>
<html>
  <head>
    <title>MAQUINA 2</title>
  </head>
  <body>
    <h1>MAQUINA 2</h1>
    <p>Ahora mismo te encuentras en la maquina 2</p>
  </body>
</html>
adrian ~ □
```

## Habilitar estadísticas del balanceador

Como se propone en el guión, una opción que nos puede beneficiar es el uso del módulo de estadísticas del balanceador. Se puede habilitar añadiendo la siguiente información al mismo archivo de configuración anteriormente mencionado:

```
global
    stats socket /var/lib/haproxy/stats

listen stats
    bind *:9999
    mode http
    stats enable
    stats uri /stats
    stats realm HAProxy\ Statistics
    stats auth adrianacosa:adrianacosa
```

Una vez ya tenemos habilitadas las estadísticas del balanceador y los servidores web a los que se le balancearán las peticiones *HTTP*, podemos comenzar a probar el funcionamiento del balanceador.

Para ver las estadísticas que hemos habilitado del balanceador, accedemos de la siguiente manera:

[illegible]

A parte de las opciones que se proponen en el gui3n, yo voy a a1adir unas cuantas m1s. En primer lugar, voy a a1adir la opci3n “*stats hide-version*”, que nos permite ocultar la versi3n que estamos usando de HAProxy.  sto es  til ya que muchos ataques se deben a vulnerabilidades desconocidas por el distribuidor del software y conocida por la persona que quiere realizar un ataque. Luego si le ocultamos la versi3n, si llega a saber que en

nuestra versión existe una vulnerabilidad del tipo “x”, ya no puede saber si puede realizar ése ataque contra nuestro servidor.

Otra opción que añadiré será “*stats refresh 10s*”, que nos permite que las estadísticas del servidor se actualicen automáticamente cada 10 segundos.

Por último voy a añadir la línea “*stats timeout 1m*”, que provoca que el servidor no reciba ningún dato durante el primer minuto en el que nos conectamos.

Todas estas configuraciones añadidas quedarían de la siguiente manera:

```
global
    stats socket /var/lib/haproxy/stats
    stats timeout 1m

listen stats
    bind *:9999
    mode http

    stats enable
    stats hide-version
    stats refresh 10s
    stats uri /stats
    stats realm HAProxy\ Statistics
    stats auth adrianacosa:adrianacosa
```

Una vez reiniciamos el sistema, podemos comprobar que se han aplicado correctamente las configuraciones volviendo a acceder a la página de estadísticas:

**HAProxy**  
**Statistics Report for pid 752**

> General process information

pid = 752 (process #1, nbproc = 1, nbthread = 2)  
uptime = 0d 0h02m31s  
system limits: memmax = unlimited; ulimit-n = 1024  
maxsock = 1024; maxconn = 490; maxpipes = 0  
current conns = 2; current pipes = 0/0; conn rate = 1/sec; bit rate = 0.000 kbps  
Running tasks: 1/20; idle = 100 %

active UP  
active UP, going down  
active DOWN, going up  
active or backup DOWN  
active or backup DOWN for maintenance (MAINT)  
active or backup SOFT STOPPED for maintenance

backup UP  
backup UP, going down  
backup DOWN, going up  
not checked

Display option:  
• Scope :  
• Hide DOWN servers  
• Disable refresh  
• Refresh now  
• CSV export

External resources:  
• Primary site  
• Updates (v2.0)  
• Online manual

Note: "NOLB"/"DRAIN" = UP with load-balancing disabled.

http-in		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	Thrtle	
Frontend		0	0	-	0	0	0	0	0	490	0	0	0	0	0	0	0	0	0	0	0	0	0	OPEN								

balanceo adrianacosa		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	Thrtle
m1		0	0	-	0	0	0	0	0	32	0	0	?	0	0	0	0	0	0	0	0	0	2m21s UP	L4OK in 0ms	100	Y	-	1	1	0s	-
m2		0	0	-	0	0	0	0	0	32	0	0	?	0	0	0	0	0	0	0	0	0	2m24s UP	L4OK in 0ms	50	Y	-	1	1	0s	-
Backend		0	0		0	0	0	0	0	49	0	0	?	0	0	0	0	0	0	0	0	0	2m24s UP		150	2	0		1	6s	

stats		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	Thrtle
Frontend		1	2	-	2	2	490	13	0	0	0	0	0s	9 477	189 344	0	0	0	0	0	0	0	OPEN								
Backend		0	0		0	1	49	9	0	0	0	0	0s	9 477	189 344	0	0	0	9	0	0	0	2m31s UP		0	0	0	0	0	0	

En esta página podemos obtener información detallada sobre el funcionamiento actual de nuestro balanceador de carga. En la sección de *session rate* podemos encontrar información acerca del número de conexiones actuales, máximas y límite que tiene el servidor en este momento. Para poder cambiar el parámetro de límite, tendríamos que añadir en el archivo de configuración una línea especificando “*rate-limit <sesiones>*”, donde sesiones sería la cantidad máxima posible de sesiones concurrentes.

La columna *bytes* nos indica la cantidad de bytes que se reciben o envían.

La columna *denied* nos informa del número de peticiones y respuestas que fueron rechazadas por motivos de seguridad.

La columna *errors* da información sobre las solicitudes que no se han podido completar y han terminado en error. Entre las posibles causas de los errores podemos encontrarnos:

- El cliente cierra la conexión antes de tiempo
- Se produce un error de lectura en el cliente
- Al cliente envió una petición con formato incorrecto
- ...

Y por último, en la columna *server* se nos proporciona información general sobre el estado del servidor en ese momento como si está activo o no, el tiempo que lleva activo, el tiempo que ha estado no disponible, el peso que tiene, ...



## gobetween

### Instalación de gobetween

Para instalar el balanceador de carga *gobetween*, en su repositorio oficial nos dan tres opciones para linux:

- Instalarlo vía *snap*
- Descargar el repositorio que contiene el binario y ejecutarlo
- Descargar el repositorio y compilarlo
- Con imagen pública de docker

Yo voy a optar por descargar el repositorio y compilarlo:

```
adrianacosa@m3-adrianacosa:~$ git clone https://github.com/yyyar/gobetween.git
Cloning into 'gobetween'...
remote: Enumerating objects: 1960, done.
remote: Counting objects: 100% (36/36), done.
remote: Compressing objects: 100% (25/25), done.
remote: Total 1960 (delta 13), reused 16 (delta 8), pack-reused 1924
Receiving objects: 100% (1960/1960), 540.87 KiB | 2.75 MiB/s, done.
Resolving deltas: 100% (1139/1139), done.
```

Una vez descargado, la wiki de gobetween nos dice que ejecutemos la orden make dentro de la carpeta descargada:

```
adrianacosa@m3-adrianacosa:~/gobetween$ make
Building...
go build -v -o ./bin/gobetween -ldflags '-X main.version=0.8.0 -X main.revision=6e185295c8476810c64a27f5da28edd778e23423 -X main.branch=master'
make: go: Command not found
make: *** [Makefile:32: build] Error 127
```

Pero me salta un error, y es porque me falta la dependencia go, que se refiere al lenguaje Go, luego instalo el compilador de go con el paquete *golang*:

```
adrianacosa@m3-adrianacosa:~/gobetween$ sudo apt-get install golang
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  golang-1.13 golang-1.13-doc golang-1.13-go golang-1.13-race-detector-runtime golang-1.13-src golang-doc golang-go
  golang-race-detector-runtime golang-src pkg-config
Suggested packages:
  bzr | brz mercurial subversion
The following NEW packages will be installed:
  golang golang-1.13 golang-1.13-doc golang-1.13-go golang-1.13-race-detector-runtime golang-1.13-src golang-doc golang-go
  golang-race-detector-runtime golang-src pkg-config
0 upgraded, 11 newly installed, 0 to remove and 10 not upgraded.
Need to get 63.5 MB of archives.
After this operation, 329 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

Una vez vuelvo a ejecutar la orden make, todo funciona correctamente.

```
github.com/yyyar/gobetween/server/udp
github.com/yyyar/gobetween/service
github.com/burntsushi/toml
github.com/yyyar/gobetween/server
github.com/yyyar/gobetween/utls/profiler
github.com/spf13/pflag
github.com/yyyar/gobetween/utls/codec
github.com/yyyar/gobetween/manager
github.com/yyyar/gobetween/api
github.com/yyyar/gobetween/utls/pidfile
github.com/spf13/cobra
github.com/yyyar/gobetween/cmd
github.com/yyyar/gobetween/main
Done.
adrianacosa@m3-adrianacosa:~/gobetween$
```

Ahora nos dice la documentación que debemos ejecutar la orden *make install* pero indicándole unas opciones específicas de la siguiente manera:

```
adrianacosa@m3-adrianacosa:~/gobetween$ sudo -E make install
Building...
go build -v -o ./bin/gobetween -ldflags '-X main.version=0.8.0 -X main.revision=6e185295c8476810c64a27f5da28edd778e23423 -X main.branch=master' .
Done.
install -d /usr/local/bin/
install -m 755 ./bin/gobetween /usr/local/bin/gobetween
install ./config/gobetween.toml /etc/gobetween.toml
adrianacosa@m3-adrianacosa:~/gobetween$
```

Esto sirve para que podamos ejecutar sin problema la orden *gobetween* como si lo hubiésemos instalado desde *aptitude*. Una vez tenemos todo bien instalado, ya podemos pasar a configurar *gobetween* antes de ejecutarlo por primera vez y ponerlo en marcha.

### Configuración del balanceador gobetween

Para acceder a la configuración de *gobetween* y proporcionarle la información para que nos pueda balancear nuestros dos servidores web, hay que escribir la siguiente configuración dentro de */etc/gobetween.toml*:

```
[servers.balanceo_adrianacosa]
bind = "192.168.122.67:80"
protocol = "tcp"
balance = "roundrobin"

max_connections = 10000
client_idle_timeout = "10m"
backend_idle_timeout = "10m"
backend_connection_timeout = "2s"

[servers.balanceo_adrianacosa.discovery]
kind = "static"
static_list = [
    "192.168.122.43:80",
    "192.168.122.36:80"
]

[servers.balanceo_adrianacosa.healthcheck]
fails = 1
passes = 1
interval = "2s"
timeout = "1s"
kind = "ping"
ping_timeout_duration = "500ms"
```

Procedo a explicar por qué pongo lo que pongo en cada parte:

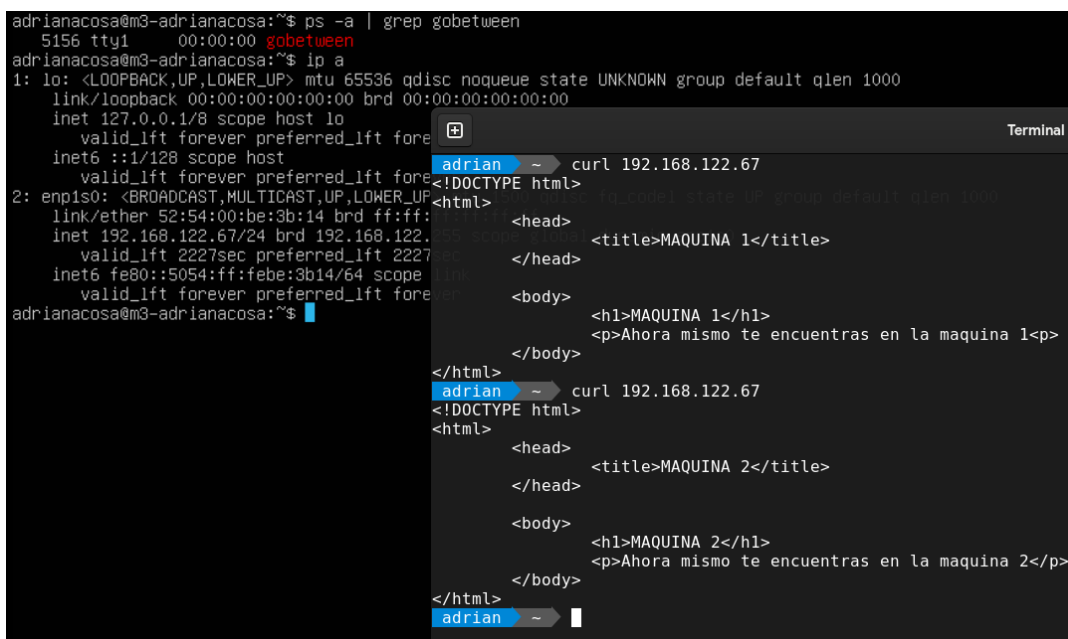
- El primer bloque sirve para definir el servidor que va a funcionar como balanceador de carga (el mismo donde se está configurando). Aquí tenemos que proporcionar la información de la IP del balanceador junto con el puerto donde va a escuchar las peticiones, el protocolo que usará, y el algoritmo para balancear. También, adicionalmente podemos indicarle parámetros como el *timeout* ante diversas situaciones como la inactividad por parte del cliente o del backend y el *timeout* para intentar una conexión hacia el backend (en este caso le he dicho que los intentos de conexión no pueden durar más de 2 segundos).

- El segundo bloque es el relativo al backend. Aquí definiremos el tipo de búsqueda de los servidores del backend, que en nuestro caso es por IP estática. Tras decirle este parámetro, creamos una lista donde le indicaremos las IP a las que tiene que balancear las peticiones en el puerto 80.

También existe la posibilidad de balancear por nombre DNS. Esta configuración se indicaría en el parámetro *kind* donde en vez de indicarle *static* para decirle las IPs directamente, le indicaríamos *srv* y tras esto *srv\_lookup\_server* para que busque la IP asociada al dominio que le indicamos.

- Por último, he añadido un bloque para comprobar el estado del servidor en todo momento mediante la utilización de *pings* para comprobar que existe conexión con el backend. Los parámetros son que en cada intervalo de 2 segundos se realice un ping, y si el ping dura más de 500 ms sin recibir respuesta el balanceador informa de que el servidor está inactivo.

Para poner en marcha el servicio, hay que hacer uso del siguiente comando: “*sudo gobetween -c /etc/gobetween.toml*”. Comprobación de que el balanceo funciona correctamente:



```

adrianacosa@m3-adrianacosa:~$ ps -a | grep gobetween
5156 tty1      00:00:00 gobetween
adrianacosa@m3-adrianacosa:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 52:54:00:be:3b:14 brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.67/24 brd 192.168.122.255 scope global dynamic noprefixroute enp1s0
        valid_lft 2227sec preferred_lft 2227sec
    inet6 fe80::5054:ff:febe:3b14/64 scope link
        valid_lft forever preferred_lft forever
adrianacosa@m3-adrianacosa:~$

```

```

adrian ~$ curl 192.168.122.67
<!DOCTYPE html>
<html>
  <head>
    <title>MAQUINA 1</title>
  </head>
  <body>
    <h1>MAQUINA 1</h1>
    <p>Ahora mismo te encuentras en la maquina 1</p>
  </body>
</html>
adrian ~$ curl 192.168.122.67
<!DOCTYPE html>
<html>
  <head>
    <title>MAQUINA 2</title>
  </head>
  <body>
    <h1>MAQUINA 2</h1>
    <p>Ahora mismo te encuentras en la maquina 2</p>
  </body>
</html>
adrian ~$

```

Si quisiésemos asignarle pesos a cada uno de los servidores como hemos estado haciendo para los otros balanceadores, simplemente tendríamos que hacer una pequeña modificación al archivo de configuración indicando la siguiente información:

```
[servers.balanceo_adrianacosa]
bind = "192.168.122.67:80"
protocol = "tcp"
balance = "weight"

max_connections = 10000
client_idle_timeout = "10m"
backend_idle_timeout = "10m"
backend_connection_timeout = "2s"

[servers.balanceo_adrianacosa.discovery]
kind = "static"
static_list = [
    "192.168.122.43:80 weight=100",
    "192.168.122.36:80 weight=50"
]

[servers.balanceo_adrianacosa.healthcheck]
fails = 1
passes = 1
interval = "2s"
timeout = "1s"
kind = "ping"
ping_timeout_duration = "500ms"
~
```

Indicando que tendrá que balancear por pesos he ajustado también dichos pesos a cada una de las máquinas. Aunque en la documentación dice que es así como ha de configurarse en caso de querer balancear por pesos, lo cierto es que no he conseguido ver bien que distribuya por cada 3 peticiones 2 a la primera máquina y 1 a la segunda, pero sí no que lo hace cada 100 como comprobaremos más adelante en los benchmarks. Entonces el resultado obtenido al ejecutarlo no se ve demasiado bien, pero si es cierto que claramente asigna pesos a ambos:

```
<h1>MAQUINA 1</h1>
<p>Ahora mismo te encuentras en la maquina 1<p>
</body>
</html>
adrian ~ curl 192.168.122.67
<!DOCTYPE html>
<html>
<head>
<title>MAQUINA 2</title>
</head>
<body>
<h1>MAQUINA 2</h1>
<p>Ahora mismo te encuentras en la maquina 2<p>
</body>
</html>
adrian ~ curl 192.168.122.67
<!DOCTYPE html>
<html>
<head>
<title>MAQUINA 1</title>
</head>
<body>
<h1>MAQUINA 1</h1>
<p>Ahora mismo te encuentras en la maquina 1<p>
</body>
</html>
adrian ~ curl 192.168.122.67
<!DOCTYPE html>
<html>
<head>
<title>MAQUINA 1</title>
</head>
<body>
<h1>MAQUINA 1</h1>
<p>Ahora mismo te encuentras en la maquina 1<p>
</body>
</html>
adrian ~
```

## Zevenet

### Instalación de zevenet

Al igual que *gobetween*, *zevenet* proporciona varios métodos de instalación de su servicio de balanceador de carga. Éstos dos métodos son:

- Descargar la imagen ISO de Debian con el balanceador de carga instalado proporcionada en su repositorio.
- Añadir los repositorios oficiales de *zevenet* a la lista de repositorios de *aptitude*.

En principio pensaba usar el segundo método de instalación, ya que me resulta el más cómodo. Pero una vez sigo los pasos que se proporcionan en el repositorio de github oficial de *zevenet* pasa lo que aparece en la siguiente captura:

```
root@m3-adrianacosa:/home/adrianacosa# echo "deb http://repo.zevenet.com/ce/v5 buster main" >> /etc/apt/sources.list.d/zevenet.l
ist
root@m3-adrianacosa:/home/adrianacosa# wget -O - http://repo.zevenet.com/zevenet.com.gpg.key | apt-key add -
--2022-04-07 11:58:29-- http://repo.zevenet.com/zevenet.com.gpg.key
Resolving repo.zevenet.com (repo.zevenet.com)... 198.211.109.244
Connecting to repo.zevenet.com (repo.zevenet.com)|198.211.109.244|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 963 [application/octet-stream]
Saving to: 'STDOUT'

-                               100%[=====]           963  --.-KB/s    in 0s

2022-04-07 11:58:30 (93.4 MB/s) - written to stdout [963/963]

OK
root@m3-adrianacosa:/home/adrianacosa# exit
adrianacosa@m3-adrianacosa:~$ sudo apt-get update
Hit:1 http://es.archive.ubuntu.com/ubuntu focal InRelease
Get:2 http://repo.zevenet.com/ce/v5 buster InRelease [1,778 B]
Get:3 http://es.archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Get:4 http://repo.zevenet.com/ce/v5 buster/main amd64 Packages [4,391 B]
Get:5 http://es.archive.ubuntu.com/ubuntu focal-backports InRelease [108 kB]
Get:6 http://es.archive.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Fetched 342 kB in 1s (294 kB/s)
Reading package lists... Done
adrianacosa@m3-adrianacosa:~$ sudo apt-get install zevenet
Reading package lists... Done
Building dependency tree
Reading state information... Done
Some packages could not be installed. This may mean that you have
requested an impossible situation or if you are using the unstable
distribution that some required packages have not yet been created
or been moved out of Incoming.
The following information may help to resolve the situation:

The following packages have unmet dependencies:
 zevenet : Depends: nftables (>= 0.9.1) but it is not going to be installed
           Depends: linux-headers-amd64 (>= 4.19+105) but it is not installable
           Depends: linux-image-amd64 (>= 4.19+105) but it is not installable
           Depends: nftlb but it is not going to be installed
E: Unable to correct problems, you have held broken packages.
adrianacosa@m3-adrianacosa:~$
```

He intentado arreglarlo intentando instalar cada uno de los paquetes que fallan por separado pero no he podido. Por lo tanto, he decidido usar el método de instalación de la imagen ISO.

La instalación la he hecho con los requerimientos que se exigen en el guión de prácticas.

Es decir, he creado otra máquina como si fuese el m3 que he estado usando hasta ahora (incluso le he configurado la misma IP para que no se note el cambio, simplemente en este apartado tendré apagada la máquina 3 con Ubuntu Server). No veo necesario poner paso por paso cómo he realizado la instalación, ya que ya en la práctica 1 se vió como hacerlo y aunque sea otro sistema operativo la instalación es exactamente la misma. Por tanto, una vez instalado, podremos trabajar sobre las configuraciones de *zevenet* en el siguiente apartado.

### Configuración del balanceador zevenet

Partiendo de que ya tenemos la ISO instalada, al iniciar sesión en la máquina nos saldrá la siguiente información:

```
Debian GNU/Linux buster/sid m3-adrianacosa tty1
m3-adrianacosa login: adrianacosa
Password:
Linux m3-adrianacosa 4.19.0-6-amd64 #1 SMP Debian 4.19.67-2 (2019-08-28) x86_64

  ZEVENET

ZEVENET Community Edition
Software developed by Zevenet SL

If you want community support contact
zevenet-ce-users@zevenet.com

or if need professional support open a ticket at
https://central.zevenet.com/

Get information about our support services visiting
https://www.zevenet.com/support/

adrianacosa@m3-adrianacosa:~$ _
```

Por lo cual podemos comprobar que la instalación ha funcionado correctamente. Ahora vamos a pasar con la configuración del balanceador de carga.

Me he encontrado con un problema y es que la imagen no contiene el orden `sudo`. He tenido que reinstalar la imagen y asignarle una contraseña al usuario `root`. Una vez hecho esto, he podido acceder correctamente y cuando he ido a mirar si tenía la IP configurada, al no tener instalado el servicio DHCP, he tenido que tocar los archivos de configuración de las interfaces de red y asignarle a la máquina la misma IP que tenía mi m3 con Ubuntu Server de la siguiente manera:

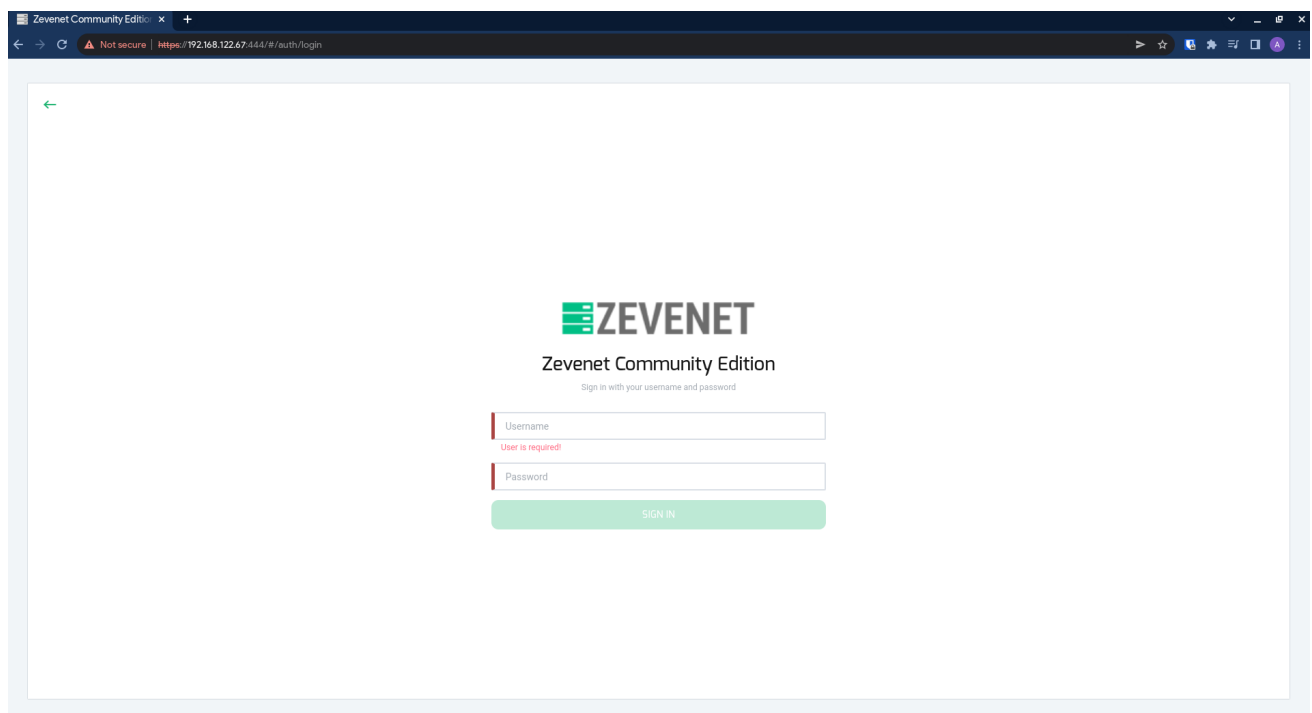
- En el fichero `/etc/network/interfaces`, he añadido el segundo párrafo:

```
auto enp1s0
iface enp1s0 inet static
address 192.168.122.67
netmask 255.255.255.0
gateway 192.168.122.1
```

- En el fichero `/etc/resolv.conf` he tenido que configurar el *gateway* por el que tiene que circular el tráfico saliente de la máquina para poder establecer una conexión entre la máquina anfitriona y la virtual:

```
GNU nano 3.2
domain localdomain
search localdomain
nameserver 192.168.122.1
```

Ahora sí, podemos comenzar a configurar el balanceo de la carga. Este balanceador es muy cómodo de configurar ya que nos proporciona una interfaz gráfica si nos conectamos desde el navegador a la máquina por el puerto 444:



En esta página de inicio debemos logearnos con las mismas credenciales del usuario que hemos creado para la máquina *zevenet*. Luego el usuario será *root* (por lo que he

comentado antes) y la contraseña *Swap1234*.

Una vez hemos entrado en la página principal del servicio, se nos enseña una vista general con información sobre el sistema donde se está alojando este servicio, así como de las granjas que tenemos configuradas. En nuestro caso no tenemos ninguna granja definida, luego vamos a definirlas.

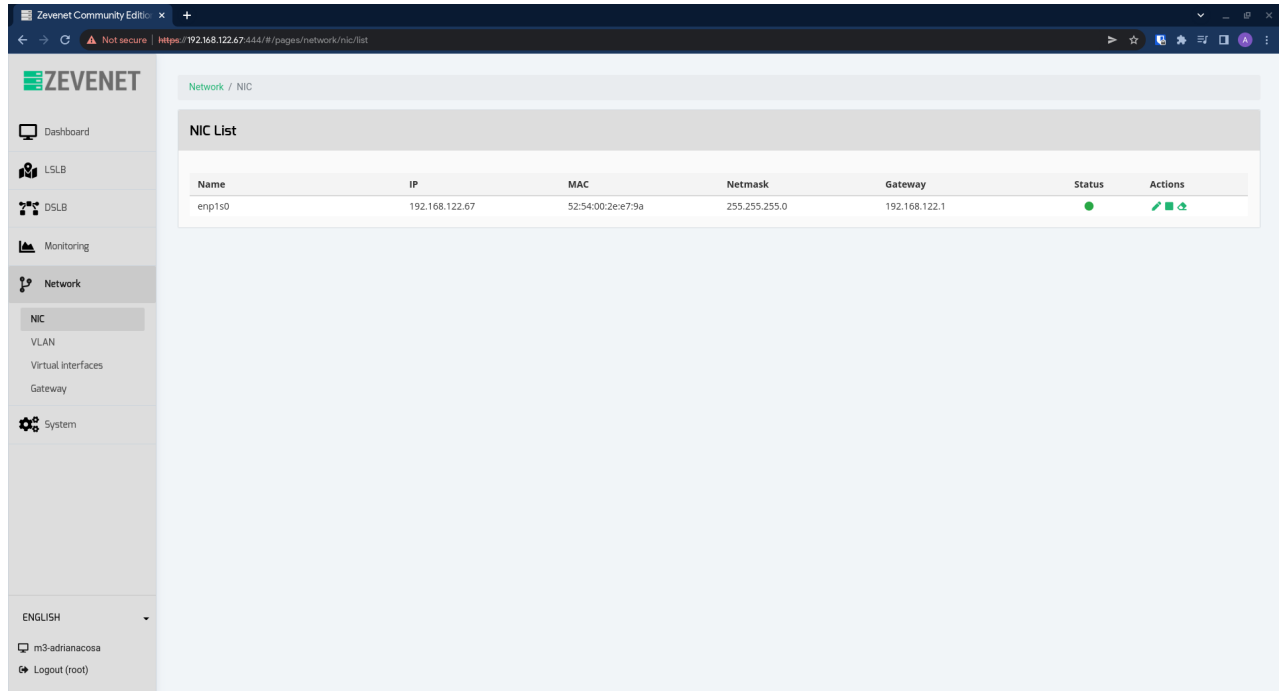
The screenshot displays the Zevenet dashboard interface. On the left is a sidebar with navigation links: Dashboard, LSLB, DSLB, Monitoring, Network, and System. The main content area is divided into several sections:

- Professional Products:** Includes ZNA Hardware Appliances, ZVA Virtual Appliances, ZBA Bare Metal Appliance, and ZWNcloud multiprovider, LB as a Service.
- Professional Services:** Includes Technical Support, Consulting Services, Zevenet CE v5.11.1 released, and Upgrade to Enterprise.
- News:** Features a Knowledge Base link, a 'New subscription Plan!' announcement, and social media follow links.
- Resources:** Includes links to Admin Guide CE 5.9, CE 5.10 & CE 5.11, How to build a cluster, How to configure APT, Enterprise differences, and Support Community.
- System Stats:** Displays CPU usage (5.5 / 100 %), Memory usage (186.83 / 987.01 Mb), and Load (0.09 / 2 Cores).
- System Information:** Lists Zevenet Version (5.11.0), Appliance Version (ZCE 6, hypervisor: KVM), Kernel Version (4.19.0-6-amd64), Hostname (m3-adrianacosa), and System Date (Thu Apr 7 22:29:29 2022).
- NIC Interfaces:** Shows a bar chart for network interfaces, with 'enp1s0' highlighted.
- Farm List:** A table with columns for Name, Profile, and Status. It currently shows 'No farm found'.

At the bottom left, there is a language selector set to 'ENGLISH' and a user profile section for 'm3-adrianacosa' with a 'Logout (root)' option.



En primer lugar, antes de definir la granja web que deberá balancear, debemos terminar de ajustar la configuración de red de nuestro balanceador en el apartado *Network* que aparece a la izquierda.



Entrando en el apartado de NIC debemos configurar nuestra interfaz que va a recibir el tráfico y que va a redirigirlo a nuestra granja web escribiendo su IP, máscara y puerta de enlace que deseemos para que así el balanceador sepa qué interfaz se va a usar.

The screenshot shows the 'Network / NIC / Edit' page. It features a 'Global Settings' section with form fields for configuring the NIC. The fields are: Name (enp1s0), IP Address (IPv4 / IPv6) (192.168.122.67), Gateway (192.168.122.1), MAC (52:54:00:2e:e7:9a), and Netmask (255.255.255.0). A green 'SUBMIT' button is located at the bottom left of the form.

Name	MAC
enp1s0	52:54:00:2e:e7:9a

IP Address (IPv4 / IPv6)	Netmask
192.168.122.67	255.255.255.0

Gateway
192.168.122.1

[SUBMIT](#)

Ahora ya estamos en disposición de indicarle al balanceador a qué servidores debe balancear la carga que reciba.

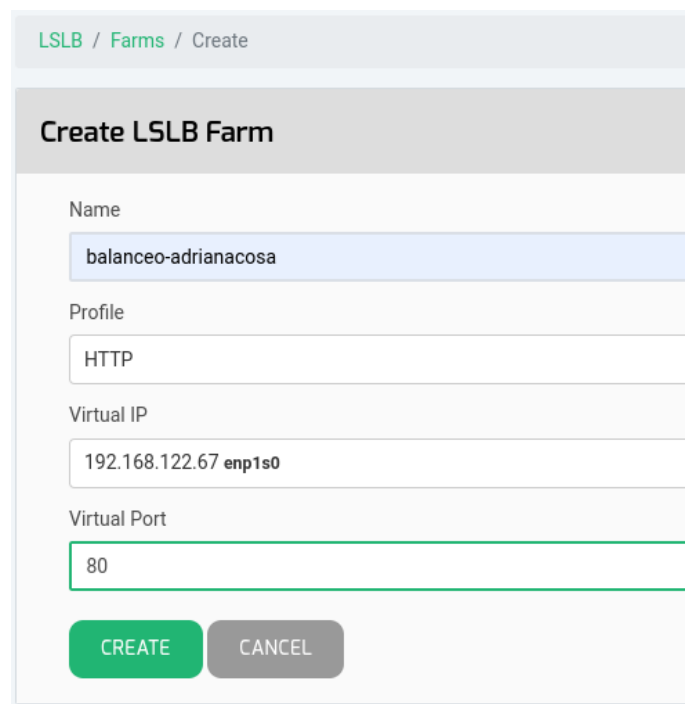
Zevenet nos permite usar GSLB (Global Server Load Balancing) o balanceo basado en balanceo con DNS, o LSLB (Local Service Load Balancing) o balanceo en un entorno local y controlado. Para ésta práctica, como no tenemos implementado un servicio de DNS, definiré el balanceo de carga con la segunda opción. Luego pulsamos en el botón del panel de la izquierda donde pone LSLB.

En el apartado de granjas, crearemos una que contendrá las IP de los dos servidores web que dispone la granja.



The screenshot shows the 'LSLB / Farms' page. At the top, there's a breadcrumb 'LSLB / Farms'. Below it is a section titled 'LSLB Farm List'. There is a green 'CREATE FARM' button. Below the button is a table with columns: Name, Profile, Virtual IP, Virtual Port, Status, and Actions. The table currently shows 'No farm found'.

Name	Profile	Virtual IP	Virtual Port	Status	Actions
No farm found					



The screenshot shows the 'Create LSLB Farm' form. It has a breadcrumb 'LSLB / Farms / Create'. The form fields are: Name (balanceo-adrianacosa), Profile (HTTP), Virtual IP (192.168.122.67 enp1s0), and Virtual Port (80). There are 'CREATE' and 'CANCEL' buttons at the bottom.

**Create LSLB Farm**

Name: balanceo-adrianacosa

Profile: HTTP

Virtual IP: 192.168.122.67 enp1s0

Virtual Port: 80

**CREATE** **CANCEL**

En la captura que aparece justo arriba, estamos definiendo por qué interfaz del balanceador se va a repartir las peticiones que lleguen desde fuera. En profile le indicamos que vamos a balancear peticiones HTTP y en Virtual Port le indicamos que las vamos a escuchar el puerto 80. Una vez creada nos aparecerá algo como lo siguiente:

LSLB / Farms

LSLB Farm List

CREATE FARM

Name	Profile	Virtual IP	Virtual Port	Status	Actions
balanceo-adrianacosa	http	192.168.122.67	80	●	   

Donde podemos apreciar que el servicio no está activo ya que no le hemos indicado a qué IPs tiene que balancear la carga. Si entramos en el menú de edición (el lápiz), podemos indicarle dichas IPs en el apartado de *services settings*.

Name

balanceo-adrianacosa

Only editable when the farm is down.

Virtual IP and Port

192.168.122.67 emp1s0 x 80

Listener

HTTP x

Advanced settings

Rewrite location headers

Enabled x

Backend connection timeout

20 FORM.seconds

Frequency to check resurrected backends

10 FORM.seconds

Message Error 414

Request URI is too long.

Message Error 501

This method may not be used.

HTTP verbs accepted

+ MS RPC extensions verbs x

Backend response timeout

45 FORM.seconds

Client request timeout

30 FORM.seconds

Message Error 500

An internal server error occurred. Please try again later.

Message Error 503

The service is not available. Please try again later.

SUBMIT

Services Settings

NEW SERVICE

Para ello le damos a *new service* para poder crear nuestro conjunto de servidores y una vez en este menú, añadimos las IP de m1 y m2:

**balanceo-adrianacosca**

Virtual Host: 192.168.122.67

URL Pattern:

Least Response: ☐

HTTPS Backends: ☐

Redirect: ☐

Persistence: No persistence

Farmguardian:

Health Checks for backend: Disabled

Backend:

**ADD BACKEND**

ID	IP	Port	Timeout	Weight	Actions
0	192.168.122.43	80			
1	192.168.122.36	80			

**SUBMIT**

Cuando hayamos terminado, le damos a submit y el servicio ya estará disponible para realizar el balanceo. Con una simple comprobación podemos ver que por defecto usa el algoritmo Round-Robin:

**Professional Products**

- ZNA Hardware Appliances
- ZVA Virtual Appliances
- ZBA Bare Metal Appliance
- ZVNCcloud multiprovider, LB as a Service

**System Stats**

Resource	Usage
CPU	3.5 / 100%
Memory	214.19 / 987.01 MB
Load	0.3 / 2 Cores

**Farm List**

Name	Profile	Status
balanceo-adrianacosca	http	<span style="color: green;">●</span>

```
adrian ~$ curl http://192.168.122.67
<!DOCTYPE html>
<html>
<head>
<title>MAQUINA 2</title>
</head>
<body>
<h1>MAQUINA 2</h1>
<p>Ahora mismo te encuentras en la maquina 2</p>
</body>
</html>
adrian ~$ curl http://192.168.122.67
<!DOCTYPE html>
<html>
<head>
<title>MAQUINA 1</title>
</head>
<body>
<h1>MAQUINA 1</h1>
<p>Ahora mismo te encuentras en la maquina 1</p>
</body>
</html>
adrian ~$
```

Como podemos apreciar en las capturas anteriores de la configuración del balanceo HTTP, podemos editar todos los parámetros desde el dashboard como pueden ser los timeouts de respuesta, los de conexión, mensajes de error personalizados, ... Y no solo en las opciones globales se nos proporciona ésta facilidad. En la configuración de cada uno de los servidores a los que se balancea podemos configurar el peso que queremos para cada servidor, así como si queremos configurar algún tipo de persistencia como las que aparecen en la captura siguiente, si queremos configurar una redirección a una URL de un determinado tipo o si queremos que exista una monitorización del estado de nuestra granja web.

Persistence

Persistence

No persistence

**No persistence**

IP: Client address

BASIC: Basic authentication

URL: A request parameter

PARM: An URI parameter

COOKIE: A certain cookie

HEADER: A certain request header

ADD BACKEND

Redirect

Redirect

Redirect URL

Redirect Type

Redirect is required

Persistence

Farmguardian

Health Checks for backend

Disabled

**Disabled**

check\_https

Send HTTPS request to backend and expect a 200 OK response with a timeout of 5 secs.

check\_tcp

Send TCP connection to backend

check\_http

Send HTTP request to backend and expect a 200 OK response with a timeout of 5 secs.

check\_http\_response\_string

Send HTTP request to backend, expect a 200 OK response and a given string in the HTML with a timeout of 5 secs. Change string by value

## Pound

### Instalación de pound

La instalación del balanceador de carga *pound* es igual de sencilla que *nginx* y *haproxy*, simplemente usando el instalador de paquetes de ubuntu *aptitude* podemos instalar el paquete y una vez instalado se iniciará automáticamente el demonio asociado:

```
adrianacosa@m3-adrianacosa:~$ sudo apt-get install pound
[sudo] password for adrianacosa:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  pound
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 89.6 kB of archives.
After this operation, 263 kB of additional disk space will be used.
Get:1 http://es.archive.ubuntu.com/ubuntu focal/universe amd64 pound amd64 2.8-2 [89.6 kB]
Fetched 89.6 kB in 1s (166 kB/s)
Selecting previously unselected package pound.
(Reading database ... 126509 files and directories currently installed.)
Preparing to unpack .../archives/pound_2.8-2_amd64.deb ...
Unpacking pound (2.8-2) ...
Setting up pound (2.8-2) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for systemd (245.4-4ubuntu3.15) ...
adrianacosa@m3-adrianacosa:~$ _
```

```
adrianacosa@m3-adrianacosa:~$ sudo systemctl status pound.service
• pound.service - LSB: reverse proxy and load balancer
   Loaded: loaded (/etc/init.d/pound; generated)
   Active: active (exited) since Thu 2022-04-07 21:09:10 UTC; 1min 0s ago
     Docs: man:systemd-sysv-generator(8)
    Tasks: 0 (limit: 1066)
   Memory: 0B
   CGroup: /system.slice/pound.service

Apr 07 21:09:10 m3-adrianacosa systemd[1]: Starting LSB: reverse proxy and load balancer...
Apr 07 21:09:10 m3-adrianacosa pound[1279]: * pound will not start unconfigured.
Apr 07 21:09:10 m3-adrianacosa pound[1279]: * Please configure; afterwards, set startup=1 in /etc/default/pound.
Apr 07 21:09:10 m3-adrianacosa systemd[1]: Started LSB: reverse proxy and load balancer.
adrianacosa@m3-adrianacosa:~$
adrianacosa@m3-adrianacosa:~$ _
```

## Configuración del balanceador pound

Tras la instalación, podemos pasar a configurar nuestro balanceador. Para ello tenemos que editar el archivo de configuración que se encuentra en `/etc/pound/pound.cfg` y poner lo siguiente:

```
#####
## global options:
User      "www-data"
Group     "www-data"
#RootJail  "/chroot/pound"

## Logging: (goes to syslog by default)
##      0   no logging
##      1   normal
##      2   extended
##      3   Apache-style (common log format)
LogLevel  1

## check backend every X secs:
Alive     30

## use hardware-acceleration card supported by openssl(1):
#SSLEngine  "hw"

# poundctl control socket
Control   "/var/run/pound/poundctl.socket"

#####
## listen, redirect and ... to:

## redirect all requests on port 8080 ("ListenHTTP") to the local webserver (see "Service" below):
ListenHTTP
  Address 192.168.122.67
  Port    80

  Service
    BackEnd
      Address 192.168.122.43
      Port    80
    End
    BackEnd
      Address 192.168.122.36
      Port    80
    End
  End
End
```

Al final del archivo debemos indicarle en qué direcciones se van a escuchar las peticiones HTTP y en qué puerto, y luego en service habría que poner cada una de las máquinas que se encuentran en el backend indicando su dirección IP y el puerto donde están escuchando. Si lo dejamos así, el balanceador funcionará con el algoritmo Round-Robin. Un dato importante a tener en cuenta es que si no cambiamos la variable *startup* a 1 del archivo `/etc/default/pound`, el servicio no funcionará incluso estando activado.

```
# Defaults for pound initscript
# sourced by /etc/init.d/pound
# installed at /etc/default/pound by the maintainer scripts

# prevent startup with default configuration
# set the below variable to 1 in order to allow pound to start
startup=1
```

Ahora si comprobamos el funcionamiento:

```
adrian ~$ curl http://192.168.122.67
<!DOCTYPE html>
<html>
  <head>
    <title>MAQUINA 1</title>
  </head>
  <body>
    <h1>MAQUINA 1</h1>
    <p>Ahora mismo te encuentras en la maquina 1</p>
  </body>
</html>
adrian ~$ curl http://192.168.122.67
<!DOCTYPE html>
<html>
  <head>
    <title>MAQUINA 2</title>
  </head>
  <body>
    <h1>MAQUINA 2</h1>
    <p>Ahora mismo te encuentras en la maquina 2</p>
  </body>
</html>
adrian ~$ curl http://192.168.122.67
```

Para poder configurar el balanceador con un algoritmo de ponderación, debemos indicarle en cada uno de los servicios del backend:

```
Service
  Backend
    Address 192.168.122.43
    Port 80
    Priority 9
  End
  Backend
    Address 192.168.122.36
    Port 80
    Priority 4
  End
End
```



Podemos comprobar como la asignación de pesos se realiza correctamente:

```
adrian ~ curl http://192.168.122.67
<!DOCTYPE html>
<html>
  <head>
    <title>MAQUINA 1</title>
  </head>
  <body>
    <h1>MAQUINA 1</h1>
    <p>Ahora mismo te encuentras en la maquina 1<p>
  </body>
</html>
adrian ~ curl http://192.168.122.67
<!DOCTYPE html>
<html>
  <head>
    <title>MAQUINA 1</title>
  </head>
  <body>
    <h1>MAQUINA 1</h1>
    <p>Ahora mismo te encuentras en la maquina 1<p>
  </body>
</html>
adrian ~ curl http://192.168.122.67
<!DOCTYPE html>
<html>
  <head>
    <title>MAQUINA 2</title>
  </head>
  <body>
    <h1>MAQUINA 2</h1>
    <p>Ahora mismo te encuentras en la maquina 2<p>
  </body>
</html>
adrian ~ curl http://192.168.122.67
```

## Someter a una alta carga el servidor balanceado

Para someter a una alta carga a los distintos balanceadores que he instalado, he usado la herramienta apache benchmark usando la siguiente orden:

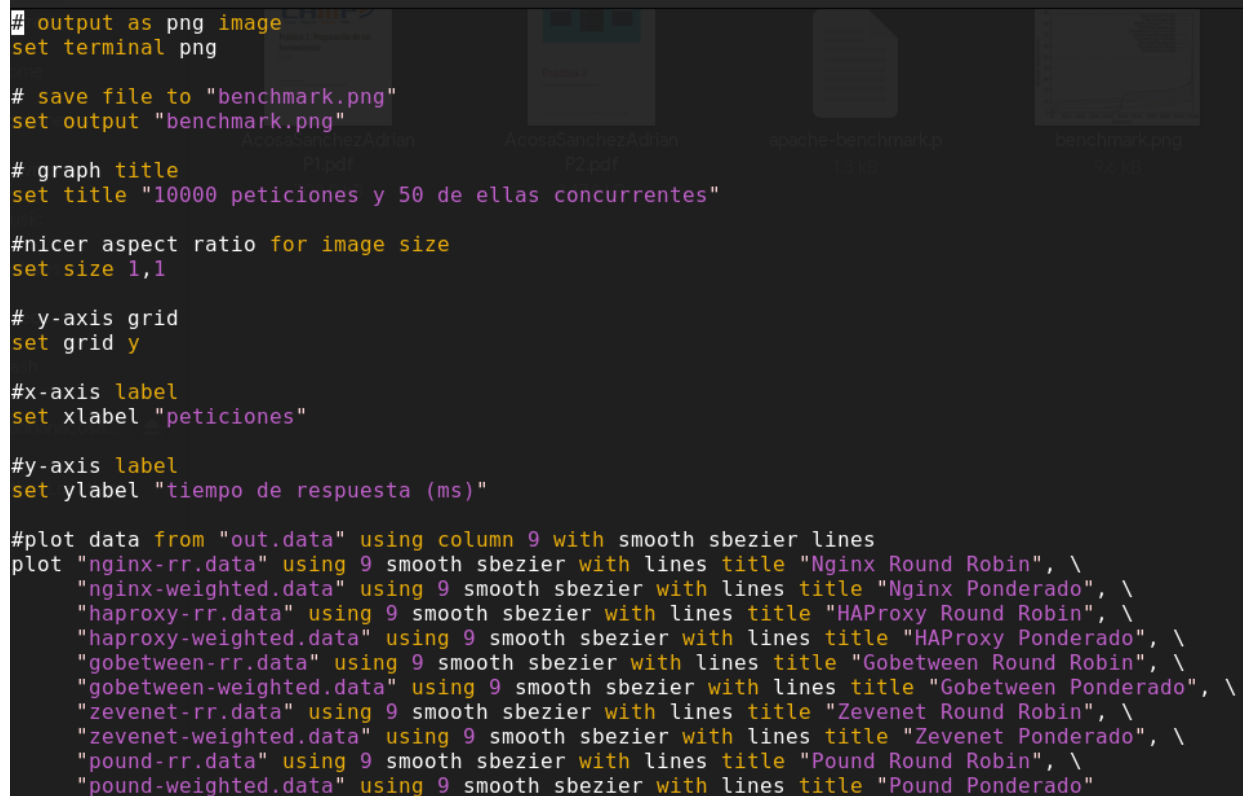
```
ab -n 10000 -c 50 -g <nombre-balanceador>-<algoritmo>.data http://192.168.122.67/
```

Donde *-n* indica el número total de peticiones a realizar, *-c* el número de peticiones concurrentes, *-g* indica que nos devuelva como resultado un fichero que podrá imprimirse con gnuplot y al final le indicamos la IP a la que vamos a someter a carga. Es muy importante poner la barra final ya que si no la ponemos *ab* nos devolverá un error.

Dicho esto, pasamos a ver las comparativas entre los distintos balanceadores con Round Robin y ponderaciones.

### Análisis comparativo

Para crear la gráfica he usado un script de gnuplot donde le he indicado que cada fichero me lo muestre usando la novena columna del fichero generado por *ab*:



```
# output as png image
set terminal png

# save file to "benchmark.png"
set output "benchmark.png"

# graph title
set title "10000 peticiones y 50 de ellas concurrentes"

# nicer aspect ratio for image size
set size 1,1

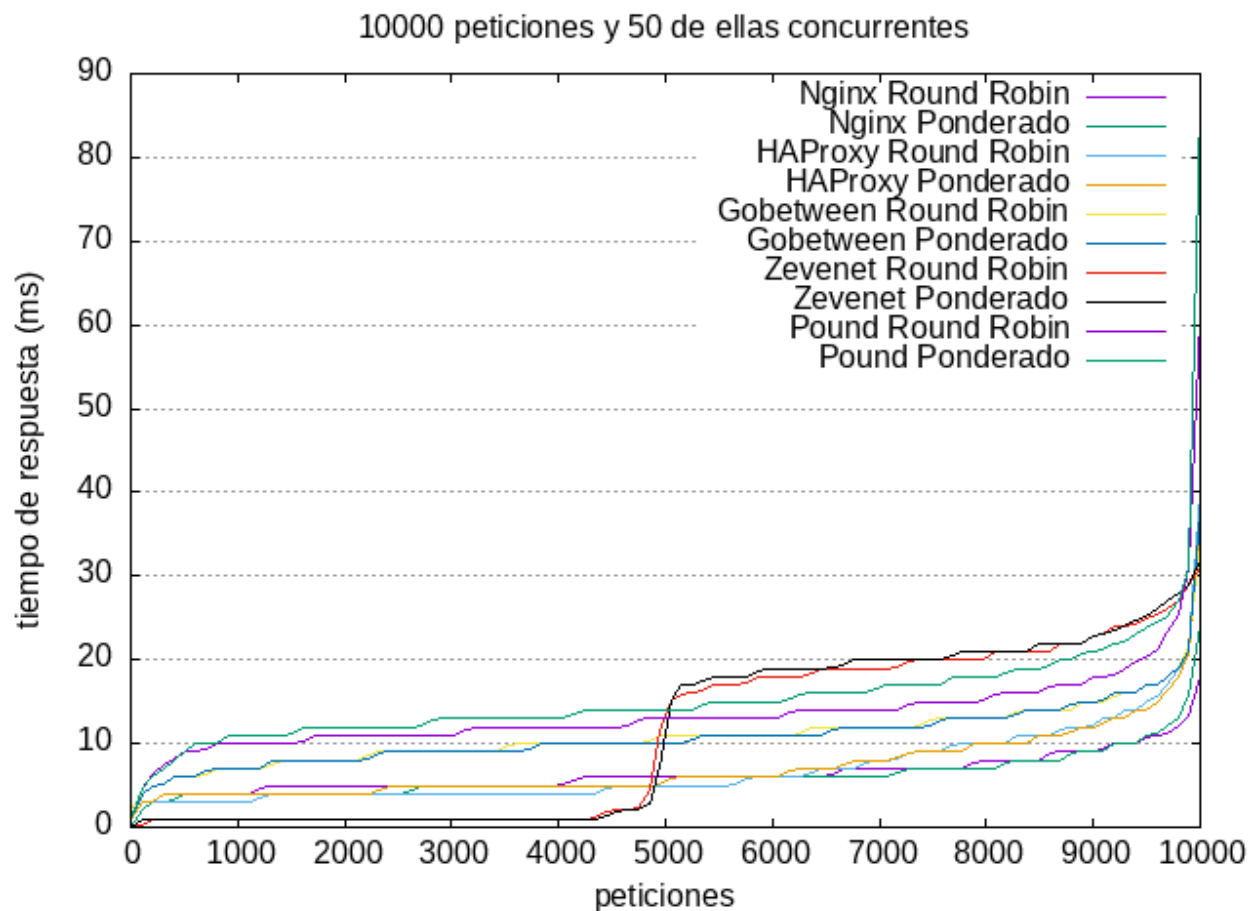
# y-axis grid
set grid y

# x-axis label
set xlabel "peticiones"

# y-axis label
set ylabel "tiempo de respuesta (ms)"

# plot data from "out.data" using column 9 with smooth sbezier lines
plot "nginx-rr.data" using 9 smooth sbezier with lines title "Nginx Round Robin", \
     "nginx-weighted.data" using 9 smooth sbezier with lines title "Nginx Ponderado", \
     "haproxy-rr.data" using 9 smooth sbezier with lines title "HAProxy Round Robin", \
     "haproxy-weighted.data" using 9 smooth sbezier with lines title "HAProxy Ponderado", \
     "gobetween-rr.data" using 9 smooth sbezier with lines title "Gobetween Round Robin", \
     "gobetween-weighted.data" using 9 smooth sbezier with lines title "Gobetween Ponderado", \
     "zevenet-rr.data" using 9 smooth sbezier with lines title "Zevenet Round Robin", \
     "zevenet-weighted.data" using 9 smooth sbezier with lines title "Zevenet Ponderado", \
     "pound-rr.data" using 9 smooth sbezier with lines title "Pound Round Robin", \
     "pound-weighted.data" using 9 smooth sbezier with lines title "Pound Ponderado"
```

Una vez tenemos las herramientas, la gráfica generada por gnuplot es la siguiente:



En esta gráfica podemos observar todos los balanceadores probados con el código de color de arriba a la derecha.

De forma general, el balanceador que mejores resultados obtiene es Nginx, ya que no varía demasiado con respecto a la cantidad de peticiones. Después de éste le sigue HAProxy, el cual podemos ver que tiene un tiempo de respuesta un poco mayor para los números de peticiones más altos, pero igual que nginx para los más bajos. El siguiente balanceador por prestaciones tendríamos a gobetween, ya que nos da unos resultados peores en general, pero es mejor que zevenet para cargas más altas. En el siguiente puesto tendríamos a Pound, que obtiene peores resultados en general que gobetween. Y en último lugar yo consideraría a zevenet, porque aunque tenga los mejores resultados para cargas bajas de peticiones, a partir de las 5000 tiene un pico que nos muestra que no soporta tanto esas cargas y por tanto su rendimiento cae en picado.

Si comparamos los algoritmos para cada uno de los balanceadores, teniendo en cuenta que los he configurado para que la máquina 1 reciba el doble de peticiones que la máquina 2, vemos que la diferencia está en que empeora un poco el rendimiento unas cuantas peticiones antes que el round robin, y ésto tiene sentido ya que al estar suponiendo que m1 es mejor que m2 cuando son iguales, al someter al doble de carga a m1 la gestión de esas peticiones se ralentiza como se espera. En el único que hay una diferencia significativa entre los algoritmos es en Pound, que el rendimiento empeora mucho y mucho antes en comparación con el resto de balanceadores, los cuales no varían demasiado en rendimiento al elegir un algoritmo u otro.

## BIBLIOGRAFÍA

- 1.- [http://nginx.org/en/docs/http/load\\_balancing.html](http://nginx.org/en/docs/http/load_balancing.html)
- 2.- [http://nginx.org/en/docs/http/nginx\\_http\\_upstream\\_module.html](http://nginx.org/en/docs/http/nginx_http_upstream_module.html)
- 3.- <https://www.haproxy.com/documentation/hapee/2-4r1/onepage/#1>
- 4.- <https://www.datadoghq.com/blog/how-to-collect-haproxy-metrics/>
- 5.- <https://www.haproxy.org/download/1.4/doc/configuration.txt>
- 6.- <https://www.haproxy.com/blog/exploring-the-haproxy-stats-page/>
- 7.- <https://github.com/yyyar/gobetween/wiki/Installation>
- 8.- <https://gobetween.io/documentation.html#Introduction>
- 9.- <https://github.com/zevenet/zlb>
- 10.- [https://servidordebien.org/es/squeeze/config/network/static\\_ip](https://servidordebien.org/es/squeeze/config/network/static_ip)
- 11.- <https://www.tecmint.com/setting-up-pound-web-server-load-balancing-in-linux/>
- 12.- <https://memorynotfound.com/using-gnuplot-to-plot-apache-benchmark-data/>