
Balanceo de carga

Andrés Millán Muñoz (amilmun@correo.ugr.es)

April 13, 2022

Contents

1	Nginx	2
1.1	Instalación y configuración básica	2
1.2	Otros tipos de configuración	5
1.2.1	Otros parámetros	7
2	Haproxy	9
2.0.1	Instalación y configuración básica	9
2.0.2	Haproxy con ponderación	11
2.0.3	Otras opciones de Haproxy	12
3	Estadísticas	14
4	Go-between	17
5	Zevenet	19
6	Pound	24
7	Análisis comparativo	27
7.1	Sobre Zevenet	28
7.2	Tiempos de respuesta	29
7.3	Sobre la fiabilidad de estos datos	32
7.4	Conclusión	32
8	Bibliografía	34

En esta práctica vamos a configurar un balanceador de carga para gestionar nuestras máquinas virtuales. Para ello, configuraremos una nueva VM, llamada **m3**, e instalaremos distintos tipos de software. Entre ellos, **nginx**, **haproxy**,...

La máquina **m3** se ha instalado de manera similar a las dos anteriores, a excepción de que no se ha instalado Apache. Las IPs, por tanto, quedan así:

- **M1:** 192.168.49.128.
- **M2:** 192.168.49.129.
- **M3:** 192.168.49.130.

1 Nginx

1.1 Instalación y configuración básica

Para instalar `nginx`, debemos poner el siguiente comando:

```
1 sudo apt-get install nginx
```

Podemos iniciar el servicio con

```
1 sudo systemctl start nginx
```

Se puede comprobar el estado del servicio con

```
1 systemctl status nginx
```

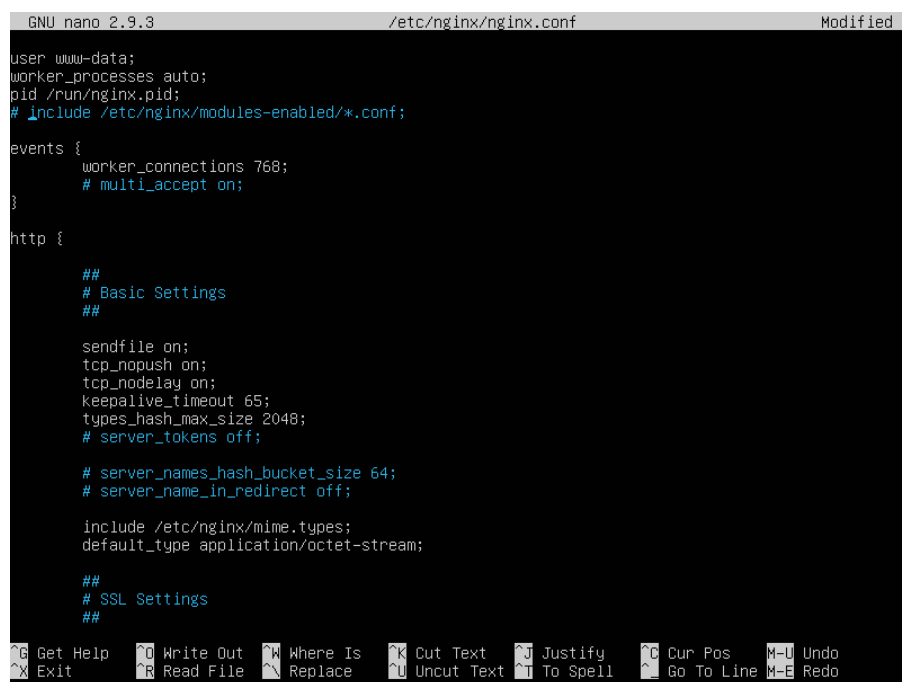
```
Setting up libjpeg-turbo8:amd64 (1.9.2-0ubuntu0.18.04.4) ...
Setting up libnginx-mod-mail (1.14.0-0ubuntu1.9) ...
Setting up libxpm4:amd64 (1:3.5.12-1) ...
Setting up libnginx-mod-http-xslt-filter (1.14.0-0ubuntu1.9) ...
Setting up libnginx-mod-http-geoip (1.14.0-0ubuntu1.9) ...
Setting up libwebp6:amd64 (0.6.1-2ubuntu0.18.04.1) ...
Setting up libjpeg8:amd64 (8c-2ubuntu8) ...
Setting up fontconfig-config (2.12.6-0ubuntu2) ...
Setting up libnginx-mod-stream (1.14.0-0ubuntu1.9) ...
Setting up libtiff5:amd64 (4.0.9-5ubuntu0.4) ...
Setting up libfontconfig1:amd64 (2.12.6-0ubuntu2) ...
Setting up libgd3:amd64 (2.2.5-4ubuntu0.5) ...
Setting up libnginx-mod-http-image-filter (1.14.0-0ubuntu1.9) ...
Setting up nginx-core (1.14.0-0ubuntu1.9) ...
Setting up nginx (1.14.0-0ubuntu1.9) ...
Processing triggers for systemd (237-3ubuntu10.52) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Processing triggers for ufw (0.36-0ubuntu0.18.04.1) ...
Processing triggers for ureadahead (0.100.0-21) ...
Processing triggers for libc-bin (2.27-3ubuntu1.5) ...
amilmun@m3-amilmun:~$ sudo systemctl start nginx
amilmun@m3-amilmun:~$ systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2022-03-30 16:45:50 UTC; 41s ago
     Docs: man:nginx(8)
  Main PID: 2227 (nginx)
    Tasks: 3 (limit: 1074)
   CGroup: /system.slice/nginx.service
           └─2227 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;
             └─2228 nginx: worker process
               └─2229 nginx: worker process

Mar 30 16:45:50 m3-amilmun systemd[1]: Starting A high performance web server and a reverse proxy se
Mar 30 16:45:50 m3-amilmun systemd[1]: nginx.service: Failed to parse PID from file /run/nginx.pid:
Mar 30 16:45:50 m3-amilmun systemd[1]: Started A high performance web server and a reverse proxy se
lines 1-14/14 (END)
```

Figure 1.1: Log con la instalación de `nginx`, activación del servicio y comprobación del estado

Para que actúe como balanceador, necesitamos deshabilitar la funcionalidad de servidor web. Para ello, editamos el archivo `/etc/nginx/nginx.conf`, comentando la línea

```
1 # include/etc/nginx/sites-enabled/*;
```



```
GNU nano 2.9.3 /etc/nginx/nginx.conf Modified
user www-data;
worker_processes auto;
pid /run/nginx.pid;
# include /etc/nginx/modules-enabled/*.conf;

events {
    worker_connections 768;
    # multi_accept on;
}

http {

    ##
    # Basic Settings
    ##

    sendfile on;
    tcp_nopush on;
    tcp_nodelay on;
    keepalive_timeout 65;
    types_hash_max_size 2048;
    # server_tokens off;

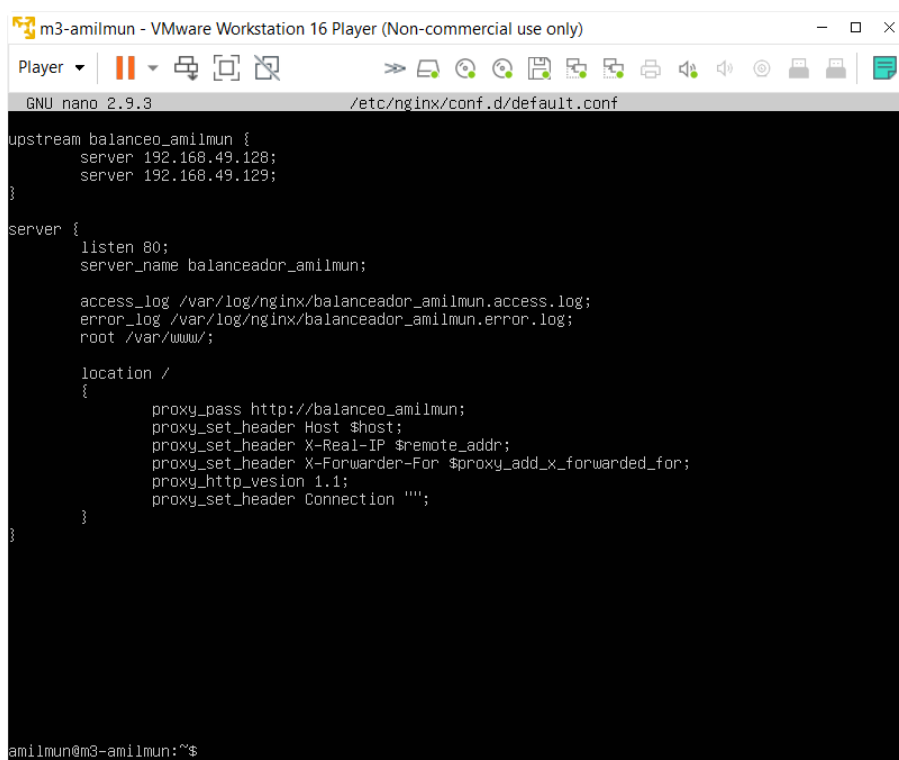
    # server_names_hash_bucket_size 64;
    # server_name_in_redirect off;

    include /etc/nginx/mime.types;
    default_type application/octet-stream;

    ##
    # SSL Settings
    ##
```

Ahora debemos configurar el *upstream* con las direcciones de las máquinas virtuales. El archivo pertinente se encuentra en `/etc/nginx/conf.d/default.conf`, y debe tener el siguiente contenido:

```
1 upstream balanceo_amilmun {
2     server ip_m1;
3     server ip_m2;
4 }
5
6 server {
7     # configuración del server. En particular, hay que hacer incapié en
8     ...
9     server_name balanceador_amilmun;
10
11     location {
12         proxy_pass http://balanceo_amilmun;
13     }
14 }
```



```
m3-amilmun - VMware Workstation 16 Player (Non-commercial use only)
Player
GNU nano 2.9.3 /etc/nginx/conf.d/default.conf

upstream balanceo_amilmun {
    server 192.168.49.128;
    server 192.168.49.129;
}

server {
    listen 80;
    server_name balanceador_amilmun;

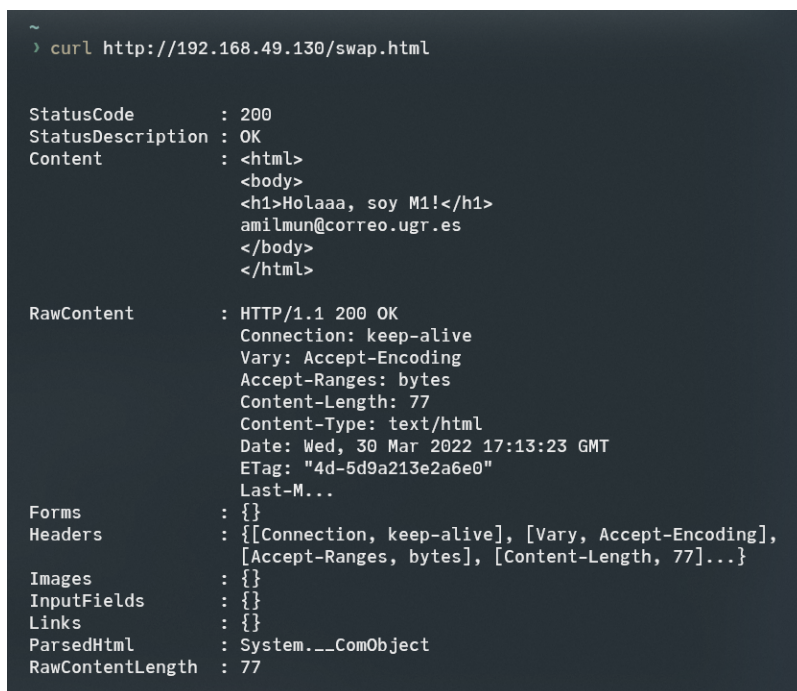
    access_log /var/log/nginx/balanceador_amilmun.access.log;
    error_log /var/log/nginx/balanceador_amilmun.error.log;
    root /var/www/;

    location /
    {
        proxy_pass http://balanceo_amilmun;
        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 '';
    }
}

amilmun@m3-amilmun:~$
```

Una vez se ha configurado el archivo, se reinicia el servicio con `sudo service nginx restart`. Si todo ha ido bien, no debería dar ningún error.

Usando curl desde el host, podemos comprobar que está funcionando:



```
~
> curl http://192.168.49.130/swap.html

StatusCode      : 200
StatusDescription : OK
Content         : <html>
                  <body>
                  <h1>Holaaa, soy M1!</h1>
                  amilmun@correo.ugr.es
                  </body>
                  </html>

RawContent      : HTTP/1.1 200 OK
                  Connection: keep-alive
                  Vary: Accept-Encoding
                  Accept-Ranges: bytes
                  Content-Length: 77
                  Content-Type: text/html
                  Date: Wed, 30 Mar 2022 17:13:23 GMT
                  ETag: "4d-5d9a213e2a6e0"
                  Last-M...

Forms           : {}
Headers         : {[Connection, keep-alive], [Vary, Accept-Encoding],
                  [Accept-Ranges, bytes], [Content-Length, 77]...}

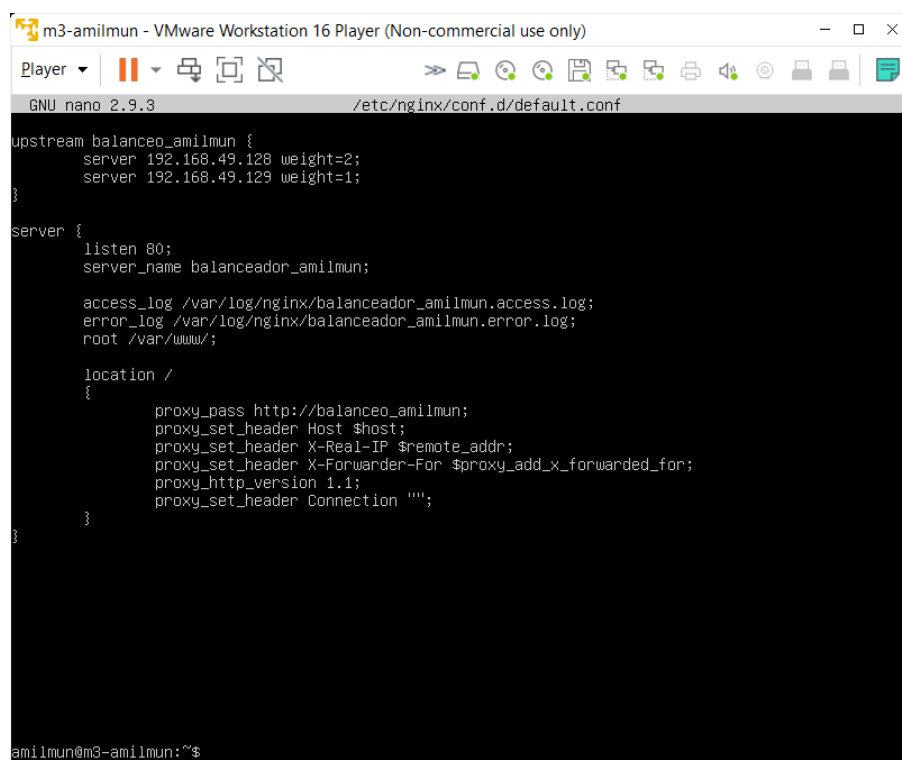
Images          : {}
InputFields     : {}
Links           : {}
ParsedHtml      : System.__ComObject
RawContentLength : 77
```

1.2 Otros tipos de configuración

La configuración anterior utiliza round robin. Podríamos cambiar a un balanceador de carga **con prioridad** usando el parámetro `weight` en `default.conf`:

```
1 upstream balanceo_amilmun {
2     server ip_m1 weight = n1;
3     server ip_m2 weight = n2;
4 }
```

Por ejemplo, si ponemos m1 con peso 2, y m2 con peso 1, quedaría de la siguiente forma:



```
m3-amilmun - VMware Workstation 16 Player (Non-commercial use only)
Player | [Icons] | GNU nano 2.9.3 | /etc/nginx/conf.d/default.conf

upstream balanceo_amilmun {
    server 192.168.49.128 weight=2;
    server 192.168.49.129 weight=1;
}

server {
    listen 80;
    server_name balanceador_amilmun;

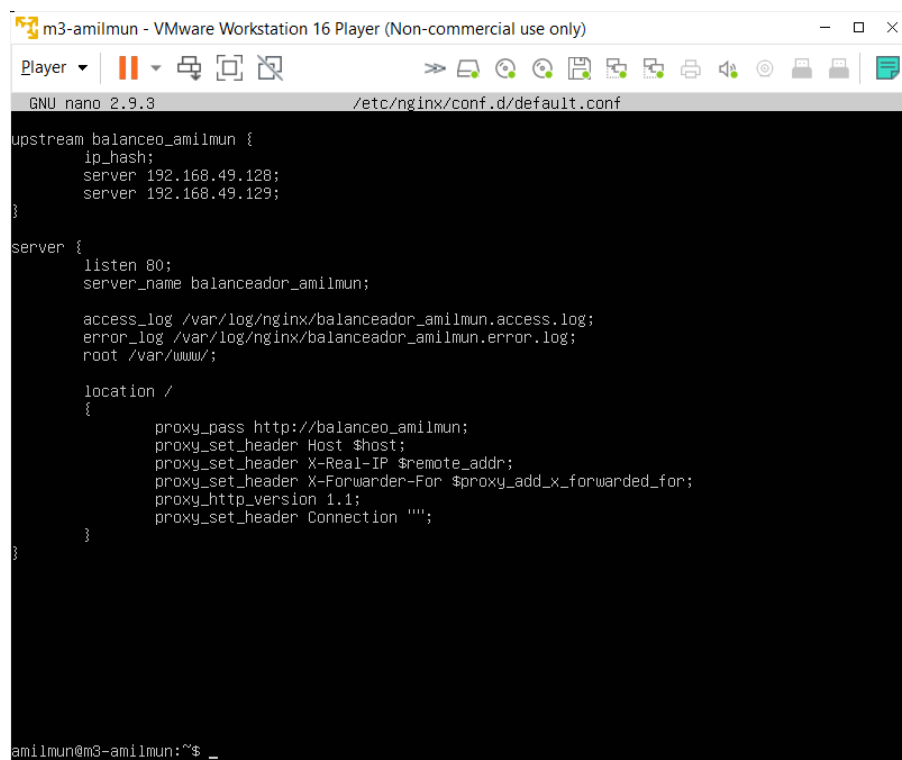
    access_log /var/log/nginx/balanceador_amilmun.access.log;
    error_log /var/log/nginx/balanceador_amilmun.error.log;
    root /var/www/;

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

amilmun@m3-amilmun:~$
```

Alternativamente, para usar un **balanceo por IP**, debemos indicar la directiva `ip_hash`:

```
1 upstream balanceo_amilmun {
2     ip_hash;
3     server ip_m1;
4     server ip_m2;
5 }
```



```
m3-amilmun - VMware Workstation 16 Player (Non-commercial use only)
Player
GNU nano 2.9.3 /etc/nginx/conf.d/default.conf

upstream balanceo_amilmun {
    ip_hash;
    server 192.168.49.128;
    server 192.168.49.129;
}

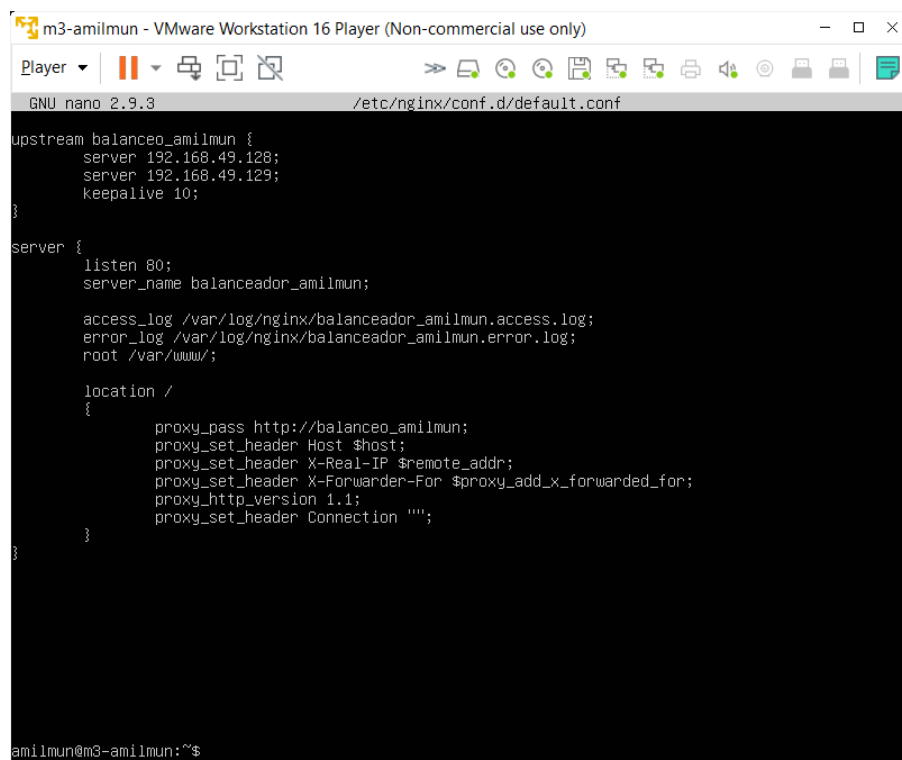
server {
    listen 80;
    server_name balanceador_amilmun;

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

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

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

Por último, podemos **mantener las conexiones activas** con `keepalive`. Para ello, usamos `keepalive {valor}`. Este `valor` limita el número de conexiones activas en idle almacenadas en cada máquina. Si se alcanza esta cifra, se cierra la conexión con la IP menos usada.



```
m3-amilmun - VMware Workstation 16 Player (Non-commercial use only)
Player
GNU nano 2.9.3 /etc/nginx/conf.d/default.conf

upstream balanceo_amilmun {
    server 192.168.49.128;
    server 192.168.49.129;
    keepalive 10;
}

server {
    listen 80;
    server_name balanceador_amilmun;

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

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

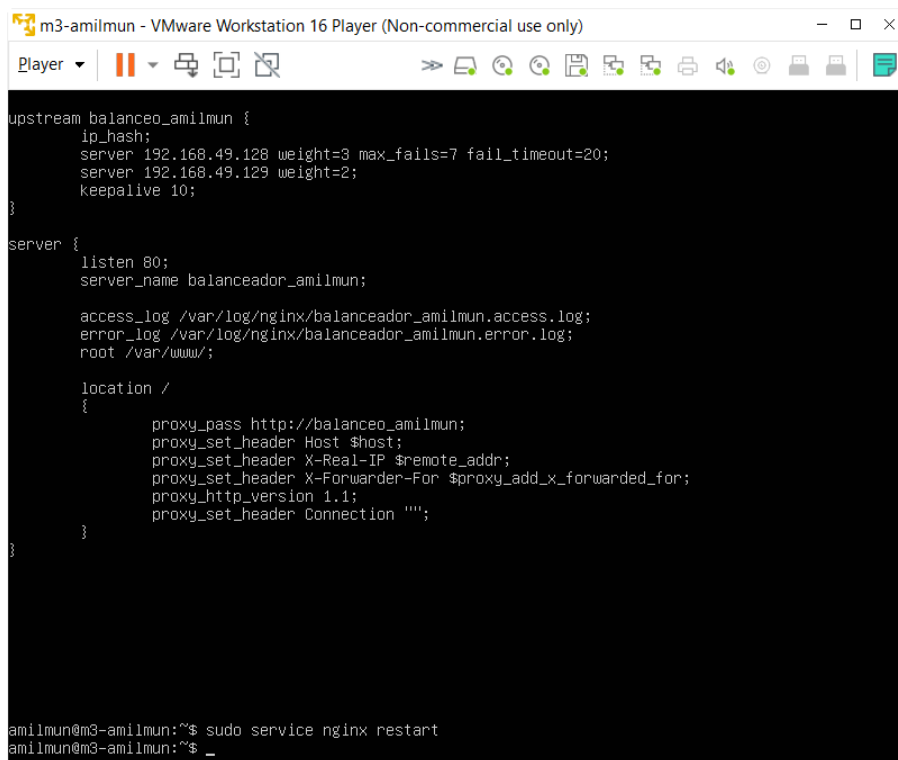
amilmun@m3-amilmun:~$
```

1.2.1 Otros parámetros

Todos estos valores se pone tras la IP del servidor en el upstream:

- **weight** = {valor}: indica el peso en la ponderación de la máquina.
- **max_fails** = {valor}: indica el número de fallos que se pueden tener en la conexión de la máquina antes de que se cierre.
- **fail_timeout** = {valor}: indica el periodo máximo de tiempo que debe ocurrir para que entre en efecto **max_fails**. Su valor por defecto es 10s.
- **down**: indica que la máquina está caída. Está pensado para utilizarse con **ip_hash**.
- **backup**: indica que la máquina está pensada para ser un respaldo. Así, si el resto no está disponible por algún motivo, entra en efecto.

Un ejemplo de configuración final sería el siguiente:



The screenshot shows a VMware Workstation 16 Player window titled "m3-amilmun - VMware Workstation 16 Player (Non-commercial use only)". The window contains a terminal window with the following Nginx configuration:

```
upstream balanceo_amilmun {
    ip_hash;
    server 192.168.49.128 weight=3 max_fails=7 fail_timeout=20;
    server 192.168.49.129 weight=2;
    keepalive 10;
}

server {
    listen 80;
    server_name balanceador_amilmun;

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

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

Below the configuration, the terminal shows the command to restart Nginx:

```
amilmun@m3-amilmun:~$ sudo service nginx restart
amilmun@m3-amilmun:~$ _
```

2 Haproxy

A continuación, instalaremos y configuraremos `haproxy`, de manera similar a `nginx`.

Antes de comenzar, debemos deshabilitar `nginx`, puesto que en caso contrario, ambos programas estarían luchando por el mismo puerto. Podemos conseguirlo con

```
1 sudo service nginx stop # solo para esta sesión
2 sudo systemctl disable nginx
```

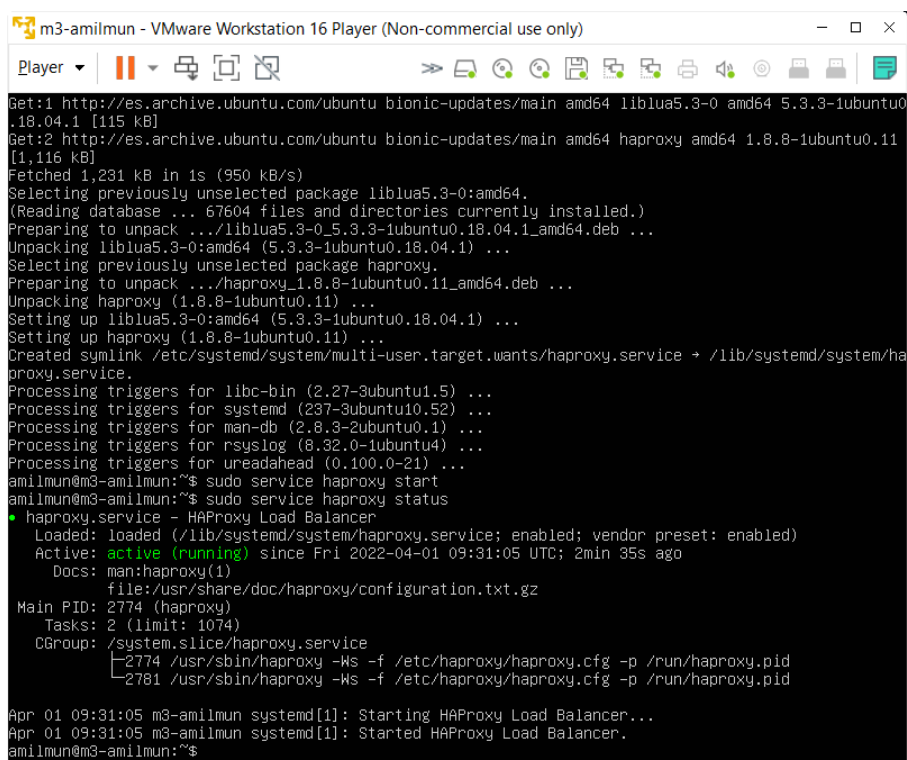
2.0.1 Instalación y configuración básica

Para instalar `haproxy`, debemos escribir el siguiente comando:

```
1 sudo apt-get install haproxy
```

Habilitamos el servicio con

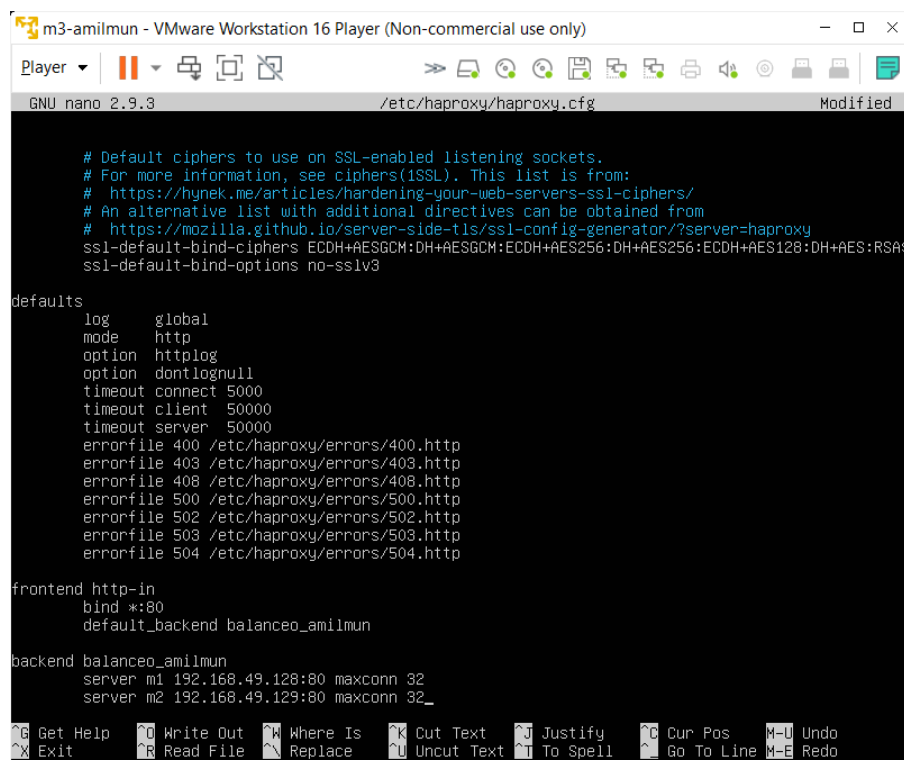
```
1 sudo systemctl start haproxy
2 # Alternativamente
3 sudo service haproxy start
```



```
m3-amilmun - VMware Workstation 16 Player (Non-commercial use only)
Player
Get:1 http://es.archive.ubuntu.com/ubuntu bionic-updates/main amd64 liblua5.3-0 amd64 5.3.3-1ubuntu0
.18.04.1 [115 kB]
Get:2 http://es.archive.ubuntu.com/ubuntu bionic-updates/main amd64 haproxy amd64 1.8.8-1ubuntu0.11
[1,116 kB]
Fetched 1,231 kB in 1s (950 kB/s)
Selecting previously unselected package liblua5.3-0:amd64.
(Reading database ... 67604 files and directories currently installed.)
Preparing to unpack .../liblua5.3-0-5.3.3-1ubuntu0.18.04.1_amd64.deb ...
Unpacking liblua5.3-0:amd64 (5.3.3-1ubuntu0.18.04.1) ...
Selecting previously unselected package haproxy.
Preparing to unpack .../haproxy_1.8.8-1ubuntu0.11_amd64.deb ...
Unpacking haproxy (1.8.8-1ubuntu0.11) ...
Setting up liblua5.3-0:amd64 (5.3.3-1ubuntu0.18.04.1) ...
Setting up haproxy (1.8.8-1ubuntu0.11) ...
Created symlink /etc/systemd/system/multi-user.target.wants/haproxy.service → /lib/systemd/system/ha
proxy.service.
Processing triggers for libc-bin (2.27-3ubuntu1.5) ...
Processing triggers for systemd (237-3ubuntu10.52) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Processing triggers for rsyslog (8.32.0-1ubuntu4) ...
Processing triggers for ureadahead (0.100.0-21) ...
amilmun@m3-amilmun:~$ sudo service haproxy start
amilmun@m3-amilmun:~$ sudo service haproxy status
• haproxy.service - HAProxy Load Balancer
  Loaded: loaded (/lib/systemd/system/haproxy.service; enabled; vendor preset: enabled)
  Active: active (running) since Fri 2022-04-01 09:31:05 UTC; 2min 35s ago
    Docs: man:haproxy(1)
           file:/usr/share/doc/haproxy/configuration.txt.gz
  Main PID: 2774 (haproxy)
    Tasks: 2 (limit: 1074)
   CGroup: /system.slice/haproxy.service
           └─2774 /usr/sbin/haproxy -Ws -f /etc/haproxy/haproxy.cfg -p /run/haproxy.pid
           └─2781 /usr/sbin/haproxy -Ws -f /etc/haproxy/haproxy.cfg -p /run/haproxy.pid

Apr 01 09:31:05 m3-amilmun systemd[1]: Starting HAProxy Load Balancer...
Apr 01 09:31:05 m3-amilmun systemd[1]: Started HAProxy Load Balancer.
amilmun@m3-amilmun:~$
```

El archivo de configuración se encuentra en `/etc/haproxy/haproxy.cfg`. Haremos una configuración muy similar a la de `nginx`. Para conseguirlo, debemos editar el archivo, escribiendo lo siguiente:



```
# Default ciphers to use on SSL-enabled listening sockets.
# For more information, see ciphers(1SSL). This list is from:
# https://hynek.me/articles/hardening-your-web-servers-ssl-ciphers/
# An alternative list with additional directives can be obtained from
# https://mozilla.github.io/server-side-tls/ssl-config-generator/?server=haproxy
ssl-default-bind-ciphers ECDH+AESGCM:DH+AESGCM:ECDH+AES256:DH+AES256:ECDH+AES128:DH+AES:RSA
ssl-default-bind-options no-sslv3

defaults
    log      global
    mode     http
    option   httplog
    option   dontlognull
    timeout  connect 5000
    timeout  client  50000
    timeout  server  50000
    errorfile 400 /etc/haproxy/errors/400.http
    errorfile 403 /etc/haproxy/errors/403.http
    errorfile 408 /etc/haproxy/errors/408.http
    errorfile 500 /etc/haproxy/errors/500.http
    errorfile 502 /etc/haproxy/errors/502.http
    errorfile 503 /etc/haproxy/errors/503.http
    errorfile 504 /etc/haproxy/errors/504.http

frontend http-in
    bind *:80
    default_backend balanceo_amilmun

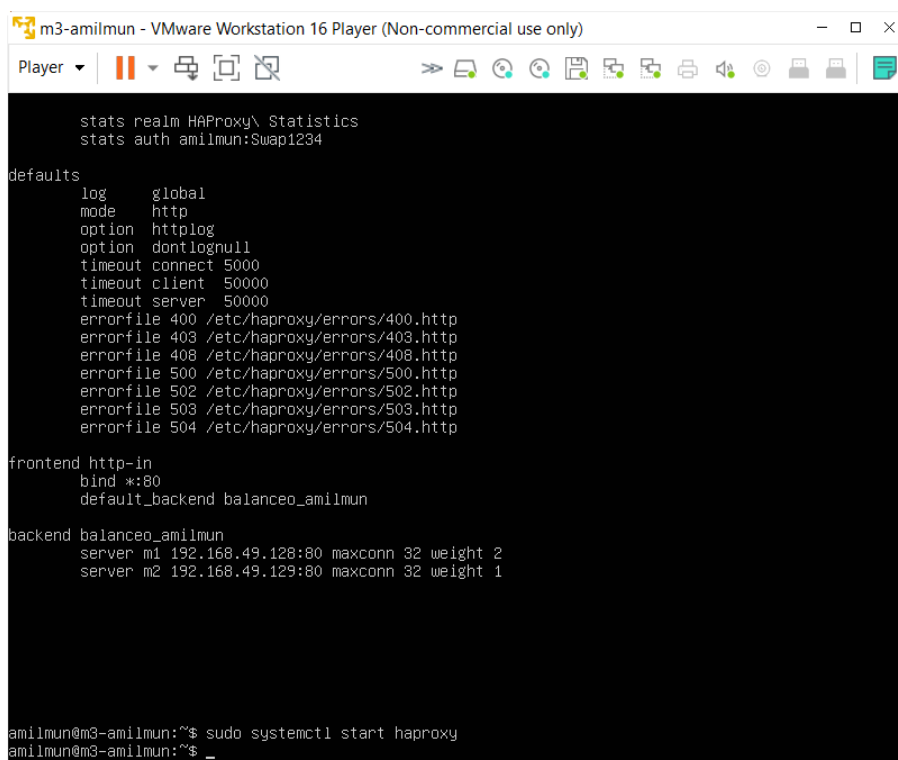
backend balanceo_amilmun
    server m1 192.168.49.128:80 maxconn 32
    server m2 192.168.49.129:80 maxconn 32_
```

De esta forma, hemos creado un frontend que recibe conexiones http desde el puerto 80, y se las manda al backend `balanceo_amilmun`. Este backend tiene dos máquinas (m1 y m2), soportando cada máquina un número máximo de conexiones (`maxconn`) de 32 usuarios.

Para comprobar que funciona correctamente, podemos hacer `curl 192.168.49.130/swap.html`. Dado que la salida es la misma que la que tuvimos con `nginx`, omitiré el pantallazo.

2.0.2 Haproxy con ponderación

Podemos configurar haproxy con ponderación usando el parámetro `weight {peso}`:



```
m3-amilmun - VMware Workstation 16 Player (Non-commercial use only)
Player
stats realm HAProxy\ Statistics
stats auth amilmun:Swap1234

defaults
    log      global
    mode     http
    option   httplog
    option   dontlognull
    timeout  connect 5000
    timeout  client  50000
    timeout  server  50000
    errorfile 400 /etc/haproxy/errors/400.http
    errorfile 403 /etc/haproxy/errors/403.http
    errorfile 408 /etc/haproxy/errors/408.http
    errorfile 500 /etc/haproxy/errors/500.http
    errorfile 502 /etc/haproxy/errors/502.http
    errorfile 503 /etc/haproxy/errors/503.http
    errorfile 504 /etc/haproxy/errors/504.http

frontend http-in
    bind *:80
    default_backend balanceo_amilmun

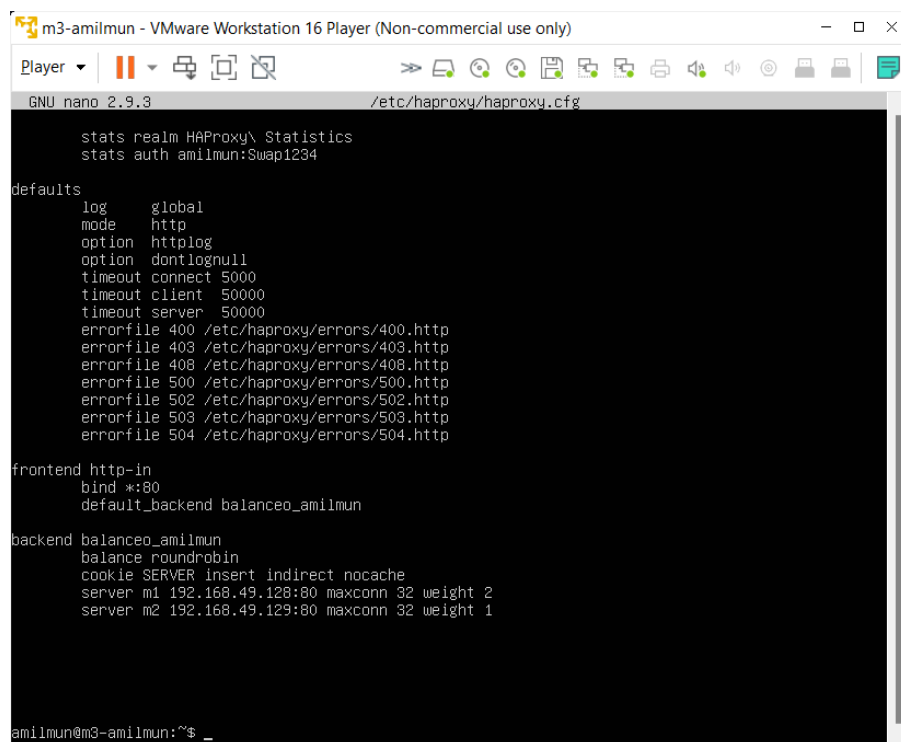
backend balanceo_amilmun
    server m1 192.168.49.128:80 maxconn 32 weight 2
    server m2 192.168.49.129:80 maxconn 32 weight 1

amilmun@m3-amilmun:~$ sudo systemctl start haproxy
amilmun@m3-amilmun:~$ _
```

2.0.3 Otras opciones de Haproxy

Como era de esperar, haproxy tiene muchas opciones diversas de configuración. En esta sección vamos a describir algunas de ellas; las que resulten más destacables.

- La clave `global` permite evitar la redundancia de código en el archivo de configuración. Se aplica tanto a backend como a frontend.
- La clave `defaults` tiene el mismo efecto, pero para los backends.
- Con respecto al backend:
 - Se puede seleccionar el tipo de balanceo con `balance {clave}`. Por defecto, se usa `roundrobin`.
 - Podemos inyectar una cookie utilizada por el balanceador para distribuir las solicitudes futuras al mismo server.



```
m3-amilmun - VMware Workstation 16 Player (Non-commercial use only)
Player
GNU nano 2.9.3 /etc/haproxy/haproxy.cfg

stats realm HAProxy\ Statistics
stats auth amilmun:Swap1234

defaults
    log          global
    mode         http
    option        httplog
    option        dontlognull
    timeout connect 5000
    timeout client 50000
    timeout server 50000
    errorfile 400 /etc/haproxy/errors/400.http
    errorfile 403 /etc/haproxy/errors/403.http
    errorfile 408 /etc/haproxy/errors/408.http
    errorfile 500 /etc/haproxy/errors/500.http
    errorfile 502 /etc/haproxy/errors/502.http
    errorfile 503 /etc/haproxy/errors/503.http
    errorfile 504 /etc/haproxy/errors/504.http

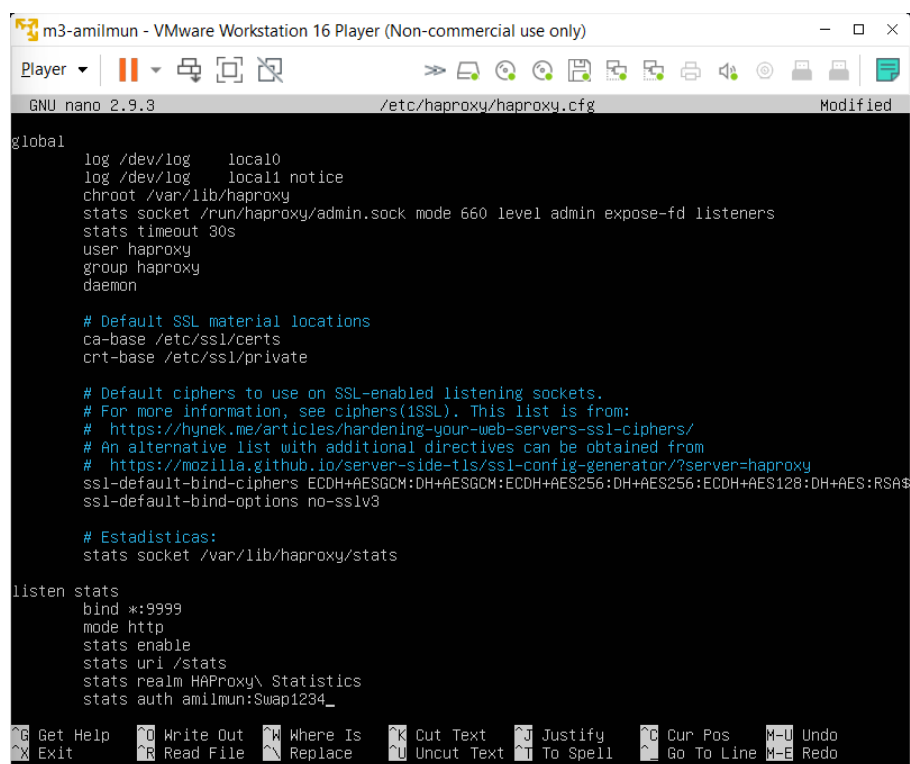
frontend http-in
    bind *:80
    default_backend balancede_amilmun

backend balancede_amilmun
    balance roundrobin
    cookie SERVER insert indirect nocache
    server m1 192.168.49.128:80 maxconn 32 weight 2
    server m2 192.168.49.129:80 maxconn 32 weight 1

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

3 Estadísticas

Una de las ventajas que ofrece **haproxy** es la facilidad para habilitar las estadísticas del balanceador. Para conseguirlo, modificamos la configuración, dejándola de la siguiente manera:



```
GNU nano 2.9.3 /etc/haproxy/haproxy.cfg Modified

global
    log /dev/log      local0
    log /dev/log      local1 notice
    chroot /var/lib/haproxy
    stats socket /run/haproxy/admin.sock mode 660 level admin expose-fd listeners
    stats timeout 30s
    user haproxy
    group haproxy
    daemon

    # Default SSL material locations
    ca-base /etc/ssl/certs
    crt-base /etc/ssl/private

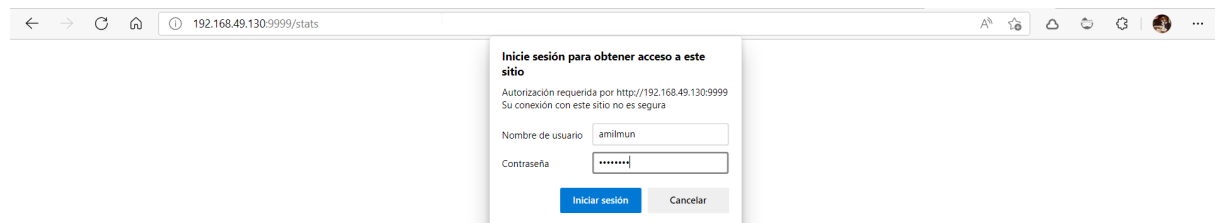
    # Default ciphers to use on SSL-enabled listening sockets.
    # For more information, see ciphers(1SSL). This list is from:
    # https://hynek.me/articles/hardening-your-web-servers-ssl-ciphers/
    # An alternative list with additional directives can be obtained from
    # https://mozilla.github.io/server-side-tls/ssl-config-generator/?server=haproxy
    ssl-default-bind-ciphers ECDH+AESGCM:DH+AESGCM:ECDH+AES256:DH+AES256:ECDH+AES128:DH+AES:RSA
    ssl-default-bind-options no-sslv3

    # Estadísticas:
    stats socket /var/lib/haproxy/stats

listen stats
    bind *:9999
    mode http
    stats enable
    stats uri /stats
    stats realm HAProxy\ Statistics
    stats auth amilmun:Swap1234_

^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos   M-U Undo
^X Exit      ^R Read File  ^N Replace   ^U Uncut Text ^T To Spell  ^_ Go To Line M-E Redo
```

Podemos acceder a la página desde el navegador entrando en <http://192.168.49.130:9999/stats>:



HAProxy version 1.8.8-1ubuntu0.11, released 2020/06/22

Statistics Report for pid 3191

> General process information

pid = 3191 (process #1, nproc = 1, nthread = 1)
 uptime = 0s 0h0m33s
 system limits: memmax = unlimited, ulimit-n = 4042
 maxsock = 4042, maxconn = 2000, maxpipes = 0
 current conns = 1, current pipes = 0/0, conn rate = 1/sec
 Running tasks: 1/6, idle = 100 %

active UP backup UP
 active UP going down backup UP going down
 active DOWN, going up backup DOWN, going up
 active or backup DOWN not checked
 active or backup DOWN for maintenance (MAINT)
 active or backup SOFT STOPPED for maintenance
 Note: "NOLE/DRAIN" = UP with load-balancing disabled.

Display option:

Scope:
 • [Hide DOWN servers](#)
 • [Refresh now](#)
 • [CSV export](#)

External resources:
 • [Primary info](#)
 • [Updates \(v1.8\)](#)
 • [Online manual](#)

Note: "NOUB/URAIN" = UP with load-balancing disabled.

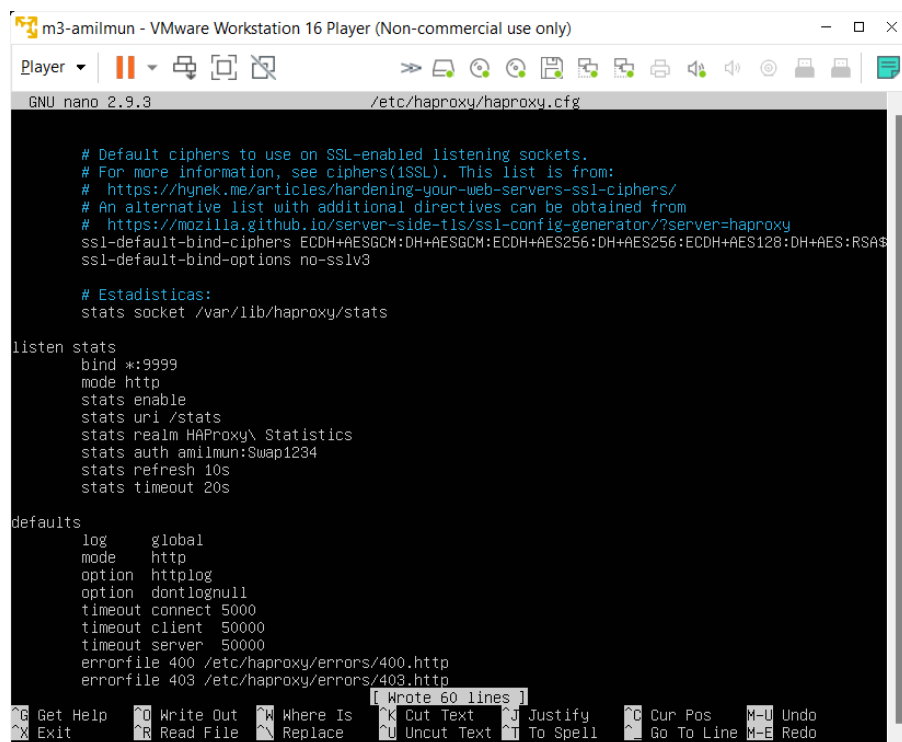
stats		Queue			Session rate			Sessions				Bytes		Denied		Errors		Warnings		Status		Server							
		Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Conn	Resp	Retr	Redis		LastChk	Wght	Act	Bck	Chk	Dwn	Dwntime	Thrtle
Frontend					1	1	-	1	1	2 000	4			1 475	686	0	0	0			OPEN								
Backend		0	0		0	1		0	1	200	2		0s	1 475	686	0	0	2	0	0	33s UP		0	0	0			0	

http-in		Queue			Session rate			Sessions				Bytes		Denied		Errors		Warnings		Status		Server							
		Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Conn	Resp	Retr	Redis		LastChk	Wght	Act	Bck	Chk	Dwn	Dwntime	Thrtle
Frontend					0	0	-	0	0	2 000	0			0	0	0	0	0			OPEN								

balancers_andaman		Queue			Session rate			Sessions				Bytes		Denied		Errors		Warnings		Status		Server							
		Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Conn	Resp	Retr	Redis		LastChk	Wght	Act	Bck	Chk	Dwn	Dwntime	Thrtle
m1		0	0	-	0	0		0	0	32	0		0	0	0	0	0	0	0	0	no check		1	Y	-			-	
m2		0	0	-	0	0		0	0	32	0		0	0	0	0	0	0	0	0	no check		1	Y	-			-	
Backend		0	0		0	0		0	0	200	0		0	0	0	0	0	0	0	0	33s UP		2	2	0		0	0s	

Algunas de las modificaciones interesantes que podemos hacer son las siguientes: - Se puede añadir la clave `stats refresh 10s` al archivo de configuración para decirle que refresque cada 10 segundos. - `stats admin if TRUE` hace un bypass del login. Podría ser útil para debuggear. - El timeout por defecto es 10s. Se puede cambiar con `stats timeout {n}s`.

La descripción de cada columna se puede hallar en [esta entrada del blog de Haproxy](#). Resulta especialmente útil como manual.



m3-amilmun - VMware Workstation 16 Player (Non-commercial use only)

Player | GNU nano 2.9.3 | /etc/haproxy/haproxy.cfg

```
# Default ciphers to use on SSL-enabled listening sockets.
# For more information, see ciphers(1SSL). This list is from:
# https://hynek.me/articles/hardening-your-web-servers-ssl-ciphers/
# An alternative list with additional directives can be obtained from
# https://mozilla.github.io/server-side-tls/ssl-config-generator/?server=haproxy
ssl-default-bind-ciphers ECDH+AESGCM:DH+AESGCM:ECDH+AES256:DH+AES256:ECDH+AES128:DH+AES:RSA$
ssl-default-bind-options no-sslv3

# Estadísticas:
stats socket /var/lib/haproxy/stats

listen stats
  bind *:9999
  mode http
  stats enable
  stats uri /stats
  stats realm HAProxy\ Statistics
  stats auth amilmun:Swap1234
  stats refresh 10s
  stats timeout 20s

defaults
  log global
  mode http
  option httplog
  option dontlognull
  timeout connect 5000
  timeout client 50000
  timeout server 50000
  errorfile 400 /etc/haproxy/errors/400.http
  errorfile 403 /etc/haproxy/errors/403.http
```

[Wrote 60 lines]

Get Help Write Out Where Is Cut Text Justify Cur Pos M-U Undo
Exit Read File Replace Uncut Text To Spell Go To Line M-E Redo

4 Go-between

El siguiente balanceador que usaremos será [gobetween](#).

Como antes, debemos deshabilitar cualquier otro balanceador que estuviera activo. En nuestro caso, ahora mismo es [haproxy](#).

Debemos instalarlo [a mano](#):

```
1 mkdir gobetween
2 cd gobetween
3 curl -s https://api.github.com/repos/yyyar/gobetween/releases | grep
  browser_download_url | grep linux_amd64 | cut -d '"' -f 4 | head -n
  1 | wget -i -
4 tar -zxvf *.tar.gz
5 nano config/gobetween.toml
6 sudo gobetween -c config/gobetween.toml
```

El archivo por defecto que viene está bastante completo. De hecho, tiene un ejemplo de tamaño considerable dentro del propio `.toml`.

Las que usaremos serán:

```
1 ...
2 [metrics]
3 enabled = true
4 ...
5
6 [servers]
7 [servers.balanceo_aamilmun]
8 bind = "192.168.49.130:80"
9 protocol = "tcp"
10 balance = "roundrobin"
11
12 max_connections = 10000
13 client_idle_timeout = "10m"
14 backend_idle_timeout = "10m"
15 backend_connection_timeout = "2s"
16
17 [servers.balanceo_aamilmun.discovery]
18 kind = "static"
19 static_list = [
20     "192.168.49.128:80 weight=2",
```

```

21     "192.168.49.129:80 weight=1"
22 ]
23
24 [servers.balanceo_amilmun.healthcheck]
25     fails = 1
26     passes = 1
27     interval = "2s"
28     timeout = "1s"
29     kind = "ping"
30     ping_timeout_duration = "500ms"

```

The screenshot shows a terminal window with two panes. The left pane displays the configuration for a load balancer, including the `static-list` and `[servers.balanceo_amilmun.healthcheck]` sections. The right pane shows the output of a `curl` command to `192.168.49.130/swap.html`, which returns a 200 status code and an HTML response.

```

static-list
  "192.168.49.129:80 weight=1"
  "192.168.49.129:80 weight=1"
]

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

# ----- tcp example ----- #
[servers.sample]
  protocol = "tcp"
  bind = "0.0.0.0:3000"

amilmun@amilmun:~$ gobetween -c config/gobetween.toml
2022-04-06 16:42:36 [INFO] gobetween v0.8.0
2022-04-06 16:42:36 [INFO] (manager): Initializing...
2022-04-06 16:42:36 [INFO] (server): Creating 'balanceo_amilmun': 192.168.49.130:80 roundrobin static-list
2022-04-06 16:42:36 [INFO] (server): Creating 'sample': 0.0.0.0:3000 weight static none
2022-04-06 16:42:36 [INFO] (scheduler): Starting scheduler balanceo_amilmun
2022-04-06 16:42:36 [INFO] (scheduler): Starting scheduler sample
2022-04-06 16:42:36 [INFO] (subserver): Creating UDP server 'udp-sample': localhost:4000 weight static none
2022-04-06 16:42:36 [INFO] (scheduler): Starting scheduler udp-sample
2022-04-06 16:42:36 [INFO] (manager): Initialized
2022-04-06 16:42:36 [INFO] (metrics): Starting metrics server :3004
2022-04-06 16:42:36 [INFO] (api): Starting api
2022-04-06 16:42:36 [INFO] (api): Starting HTTP server :8080

```

```

> curl 192.168.49.130/swap.html
StatusCode      : 200
StatusDescription : OK
Content         : <html>
                  <body>
                    <h1>Holaaa, soy M1</h1>
                    amilmun@correo.ugr.es
                  </body>
                  </html>

RawContent      : HTTP/1.1 200 OK
                  Vary: Accept-Encoding
                  Keep-Alive: timeout=5, max=100
                  Connection: Keep-Alive
                  Accept-Ranges: bytes
                  Content-Length: 77
                  Content-Type: text/html
                  Date: Wed, 06 Apr 2022 16:42:49 GMT
                  ...
Forms           : {}
Headers         : {[Vary, Accept-Encoding], [Keep-Alive, timeout=5, max=100], [Connection, Keep-Alive], [Accept-Ranges, bytes]...}
Images          : {}

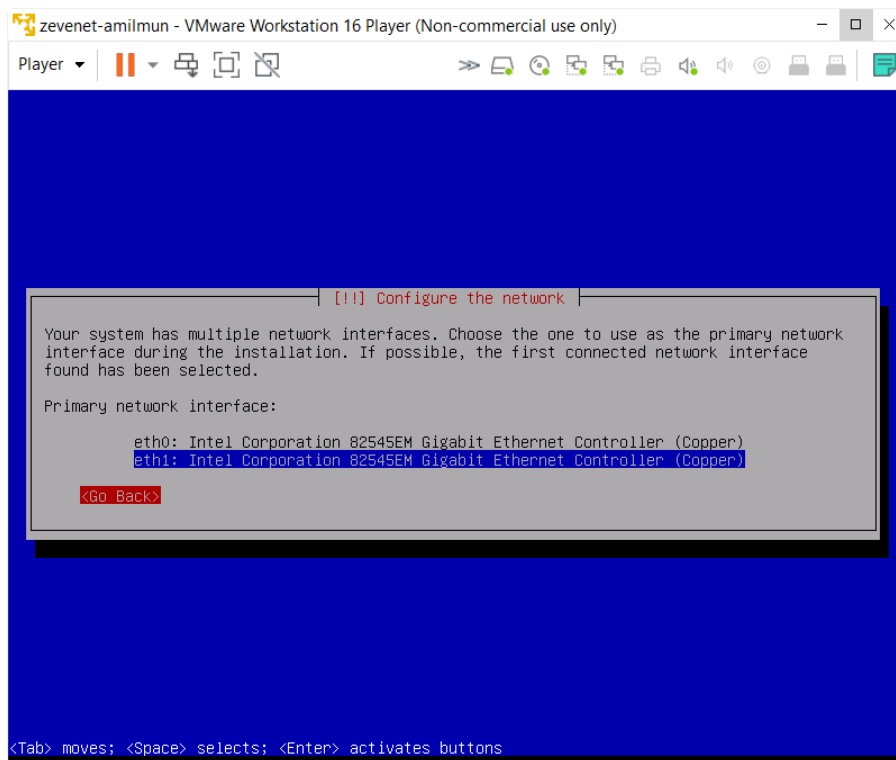
```

- El peso se puede cambiar con `{url_server:puerto} weight={peso}`.
- `leastconn` obliga a gobetween a seleccionar el backend con menos conexiones.
- `max_connections = {valor}` dicta el número máximo de conexiones al servidor. Otros parámetros relacionados son `client_idle_timeout`, `backend_idle_timeout` y `backend_connection_timeout`.
- En el apartado `[servers.balanceo_amilmun.healthcheck]` se puede configurar parámetros similares al anterior. No entraré en detalle, puesto que viene en el código anterior bien puesto; y son muy similares a los del resto de balanceadores.

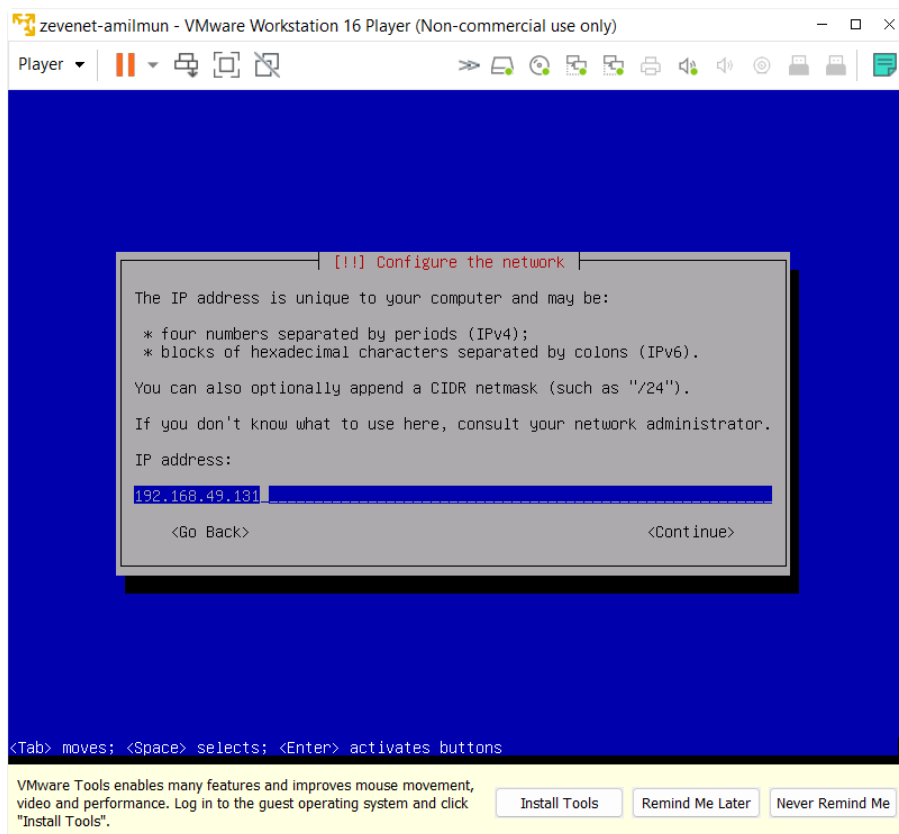
5 Zevenet

Para instalar Zevenet debemos proceder de manera algo diferente. Esto no es un programa como tal; sino una distribución. Proporcionan una ISO o un contenedor Docker. Nosotros optaremos por la primera opción.

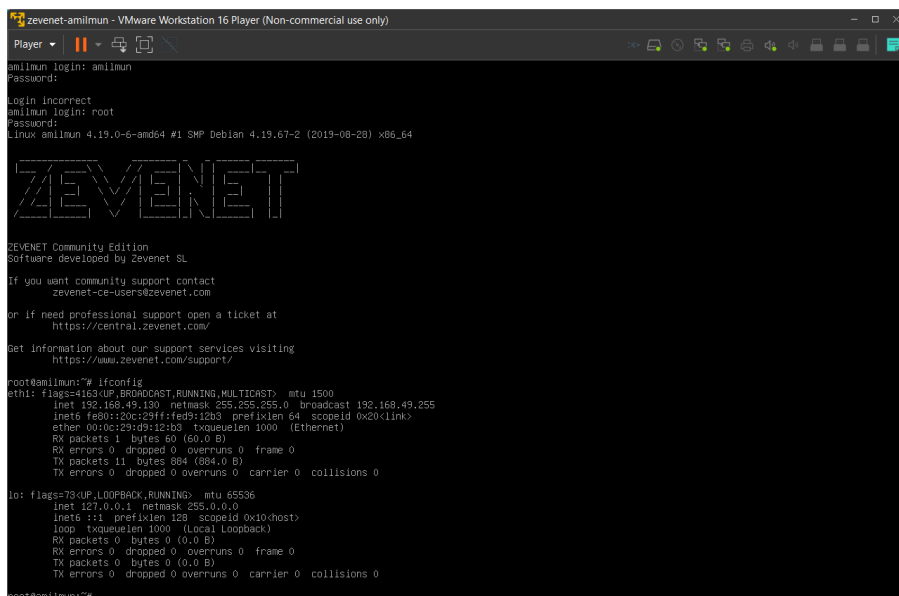
Es importante añadir las tarjetas de red antes de iniciar la instalación del sistema. De esta forma, Zevenet configurará correctamente la red. En la instalación, debemos marcar la tarjeta `eth1`:



Para la IP usaremos `192.168.49.131`. Aunque lo pondremos en el instalador, realmente, esto es irrelevante, puesto que luego lo configuraremos con `netplan`. El resto de parámetros tomarán el valor que viene por defecto. El hostname y la contraseña se ha configurado para ser los mismos que en el resto de máquinas.



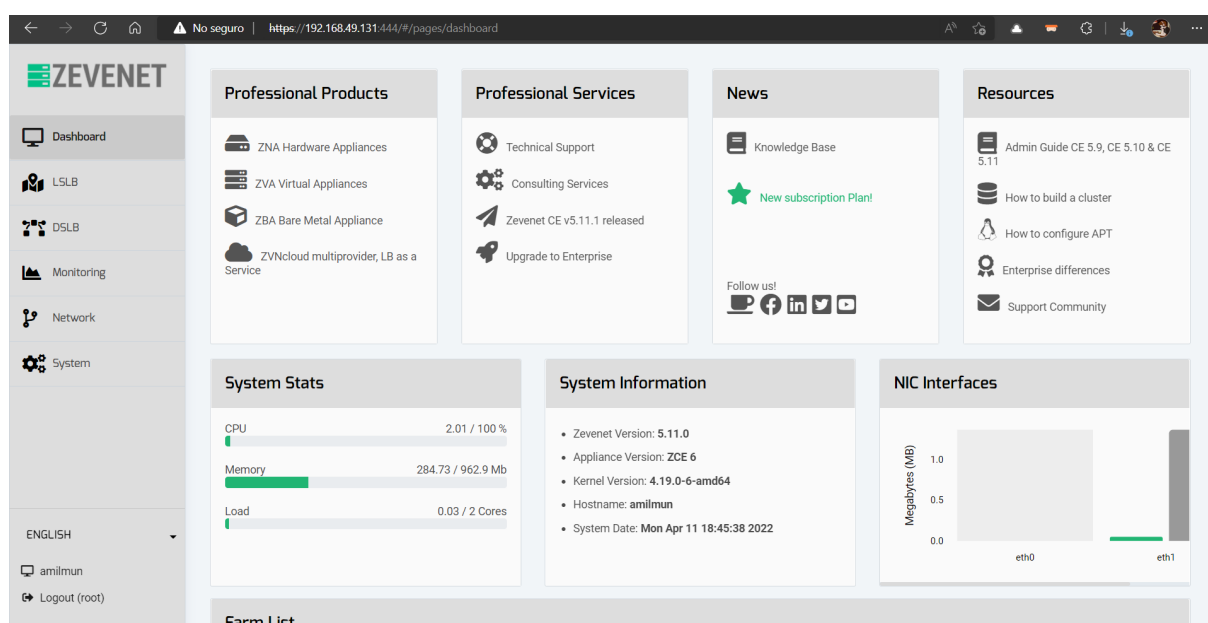
Una vez instalado, nos podemos loggear con usuario `root` y contraseña `Swap1234`.



Aplicamos el siguiente netplan:

```
zevenet-amilmun - VMware Workstation 16 Player (Non-commercial use only)
Player
network:
version: 2
ethernets:
eth0:
dhcp4: true
eth1:
dhcp4: true
addresses: [192.168.49.131/24]
```

Podemos configurar el balanceador accediendo desde el navegador a la página <https://192.168.49.131:444/>, usando los mismos credenciales que utilizamos para logearnos:



Para crear el balanceo, podemos irnos a LSLB -> Farms. Creamos una nueva granja, añadiendo los backend M1 y M2:

The screenshot displays the ZEVENET Global Settings interface. The left sidebar contains navigation links: Dashboard, LSLB, Farms, SSL Certificates, DSLB, Monitoring, Network, NIC, VLAN, Virtual interfaces, Gateway, and System. The main content area is titled 'Global Settings' and is divided into two sections.

Global Settings (Top Section):

- Name:** balanceo-amilmun
- Virtual IP and Port:** 192.168.49.131 eth1 80
- Listener:** HTTP
- Advanced settings:**
 - Rewrite location headers:** Enabled
 - 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
 - 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.

Global Settings (Bottom Section):

- Redirect:** ☐
- Persistence:** No persistence
- Farmguardian:**
 - Health Checks for backend:** Disabled
- Backend:**
 - ADD BACKEND** button
 - | ID | IP | Port | Timeout | Weight | Actions |
|----|----------------|------|---------|--------|---------|
| 0 | 192.168.49.128 | 80 | 30 | 2 | |
| 1 | 192.168.49.129 | 80 | 30 | 1 | |
 - SUBMIT** button

Con esto, el balanceador está funcionando. La persistencia del servicio se puede activar con la opción [persistence](#):

LSLB

- Farms
- SSL Certificates
- DSLB
- Monitoring
- Network
- NIC
- VLAN
- Virtual interfaces
- Gateway
- System

ENGLISH

amilmun

Logout (root)

Redirect

Redirect

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

ID	IP	Port	Timeout	Weight	Actions
0	192.168.49.128	80	30	2	
1	192.168.49.129	80	30	1	

SUBMIT

6 Pound

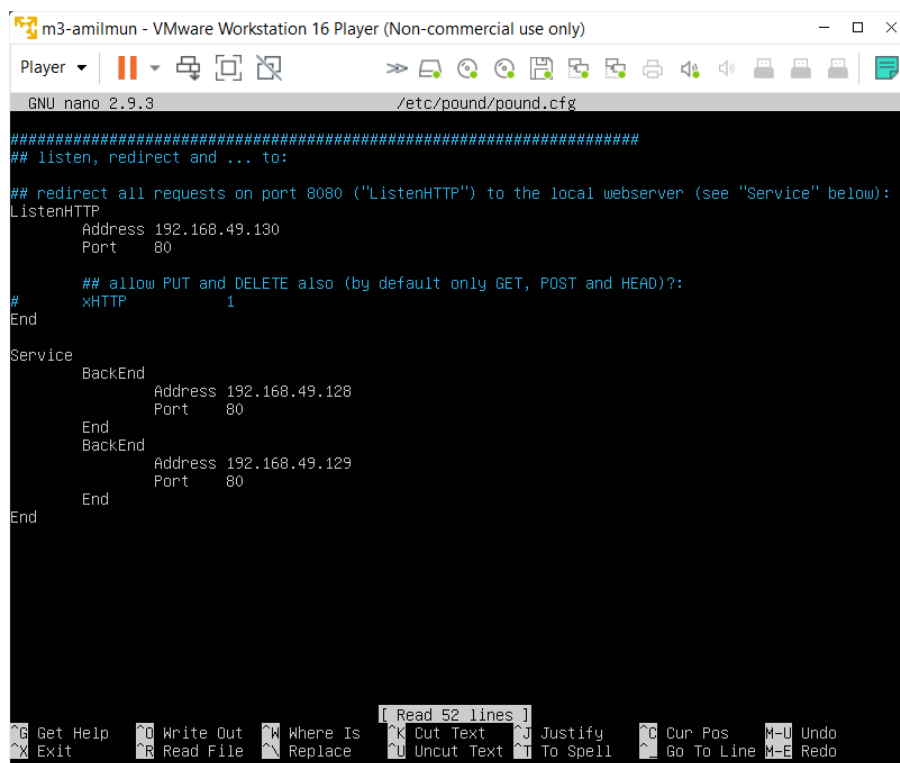
Pound es un servidor proxy reverso y balanceador de carga sencillo.

La instalación de este programa será algo diferente. Pound no se encuentra en los repositorios de Ubuntu server 18.04, pero podemos bajarnos directamente el archivo `.deb`:

```
1 sudo apt-get update && sudo apt-get upgrade
2 wget http://launchpadlibrarian.net/384724960/init-system-helpers_1.54
   _all.deb
3 wget http://archive.ubuntu.com/ubuntu/pool/universe/p/pound/pound_2.8-2
   _amd64.deb
4 sudo dpkg -i init-system-helpers_1.54_all.deb
5 sudo dpkg -i pound_2.8-2_amd64.deb
```

En el proceso hemos tenido que actualizar `init-system-helpers`, puesto que no cumplía el requisito de que la versión debe ser mayor o igual a 1.54.

El archivo de configuración se encuentra en `/etc/pound/pound.cfg`. Una configuración básica para nuestro escenario sería la siguiente:



```
m3-amilmun - VMware Workstation 16 Player (Non-commercial use only)
Player
GNU nano 2.9.3 /etc/pound/pound.cfg

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

## redirect all requests on port 8080 ("ListenHTTP") to the local webserver (see "Service" below):
ListenHTTP
    Address 192.168.49.130
    Port    80

    ## allow PUT and DELETE also (by default only GET, POST and HEAD)?:
    xHTTP   1
#
End

Service
    BackEnd
        Address 192.168.49.128
        Port    80
    End
    BackEnd
        Address 192.168.49.129
        Port    80
    End
End

[ Read 52 lines ]
Get Help  Write Out  Where Is  Cut Text  Justify  Cur Pos  M-U Undo
Exit      Read File  Replace   Uncut Text  To Spell  Go To Line M-E Redo
```

Para iniciarlo, primero modificamos el archivo `/etc/default/pound` poniendo `startup=1`. Luego, hacemos `systemctl restart pound.service`. Nos pide loggearnos, por lo que usamos nuestra contraseña del usuario (`Swap1234`).

Añadiendo el parámetro `Priority` {valor entre 1 y 9}, modificamos los pesos.

```

m3-amilmun - VMware Workstation 16 Player (Non-commercial use only)
Player | [Icons] | GNU nano 2.9.3 | /etc/pound/pound.cfg | Modified
Control "/var/run/pound/poundctl1.socket"

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

## redirect all requests on port 8080 ("ListenHTTP") to the local webserver (see "Service" below):
ListenHTTP
    Address 192.168.49.130
    Port 80

    ## allow PUT and DELETE also (by default only GET, POST and HEAD)?:
    # xHTTP 1
End

Service
    BackEnd
        Address 192.168.49.128
        Port 80
        Priority 2
    End
    BackEnd
        Address 192.168.49.129
        Port 80
        Priority 1_
    End
End

```

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos M-U Undo
 ^X Exit ^R Read File ^_ Replace ^U Uncut Text ^I To Spell ^_ Go To Line M-E Redo

Algunos parámetros interesantes que podemos modificar son...

- **Alive** {valor (s)}: es el periodo de tiempo que el servidor esperará a que el backend responda.
- **Client** {valor (s)}: es el periodo de tiempo que el servidor esperará a que el cliente responda. Pasado este tiempo, el servidor cortará la conexión.
- **Timeout** {valor (s)}: periodo de tiempo que Pound esperará al Backend para una respuesta.
- Se puede configurar un backend de emergencia con **Emergency**. Este se activará cuando el resto falle. La configuración sería idéntica a la que hemos hecho con **Backend** (a excepción de las IPs).

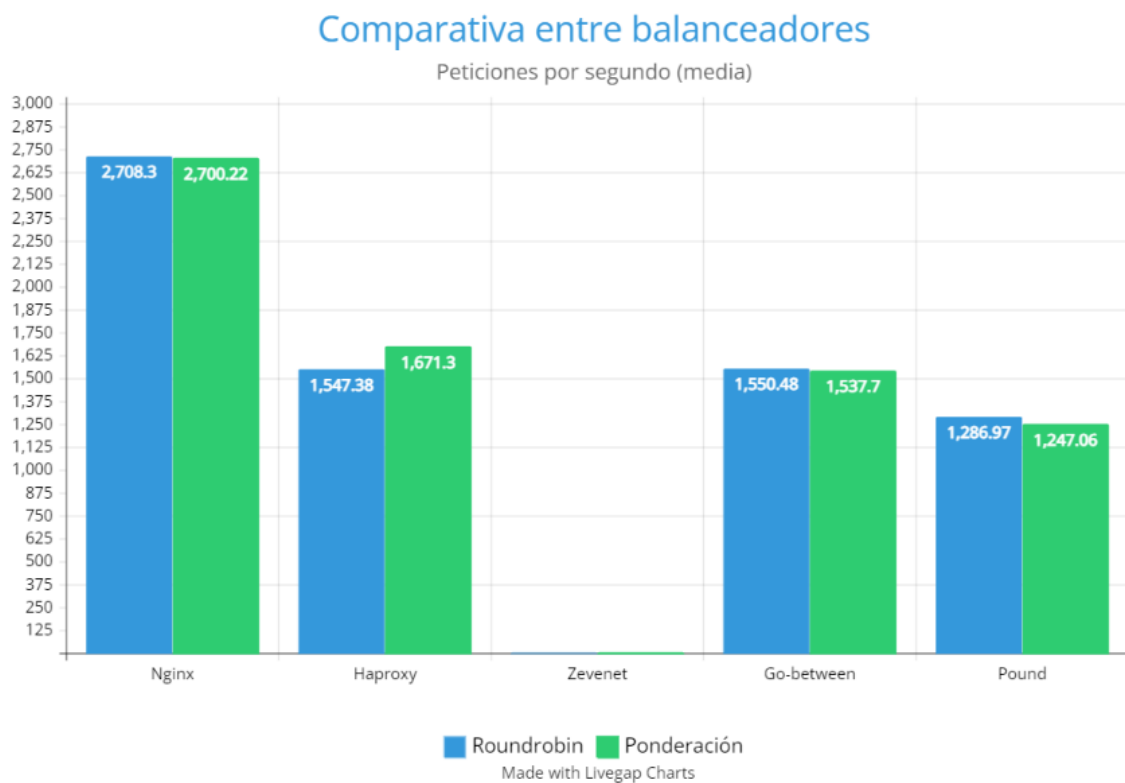
7 Análisis comparativo

Para comprobar cuál de todos los balanceadores que hemos instalado se comporta mejor, usaremos la herramienta `ab`. Ambas máquinas sirven el mismo contenido, que está basado en la página html que creamos en la práctica 1 y sincronizamos entre las dos máquinas con `rsync` en la práctica 2.

Desde el host, haremos `ab -l -n 10000 -c 10 -i http://192.168.49.130/swap.html`. Generaremos el archivo de resultados en csv usando el parámetro `-g {archivo.csv}`. Este csv está preparado para mostrar los tiempos de respuesta con `gnuplot`. Además, guardaremos el tiempo medio de peticiones por segundo de cada balanceador:

Balanceador	Peticiones por segundo
Nginx (rr)	2708.30
Nginx (weight)	2700.22
Haproxy (rr)	1547.38
Haproxy (weight)	1671.30
Zevenet (rr)	Error
Zevenet (weight)	Error
Go-between (rr)	1550.48
Go-between (weight)	1537.70
Pound (rr)	1286.97
Pound (weight)	1247.06

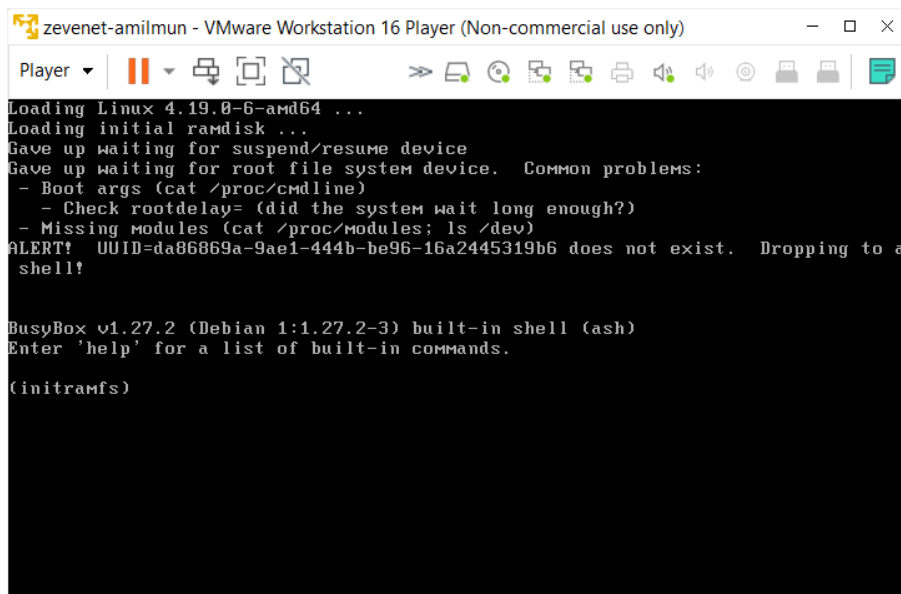
En general, **estos datos nos proporcionan un buen resumen** de cómo se comporta cada balanceador:



Hablaremos en profundidad sobre estos resultados en una sección posterior.

7.1 Sobre Zevenet

No se ha podido hacer el benchmark de Zevenet. Cuando se ha intentado iniciar, sale un mensaje de error.



```
zevenet-amilmun - VMware Workstation 16 Player (Non-commercial use only)
Player
Loading Linux 4.19.0-6-amd64 ...
Loading initial ramdisk ...
Gave up waiting for suspend/resume device
Gave up waiting for root file system device. Common problems:
- Boot args (cat /proc/cmdline)
- Check rootdelay= (did the system wait long enough?)
- Missing modules (cat /proc/modules; ls /dev)
ALERT! UUID=da86869a-9ae1-444b-be96-16a2445319b6 does not exist. Dropping to a
shell!

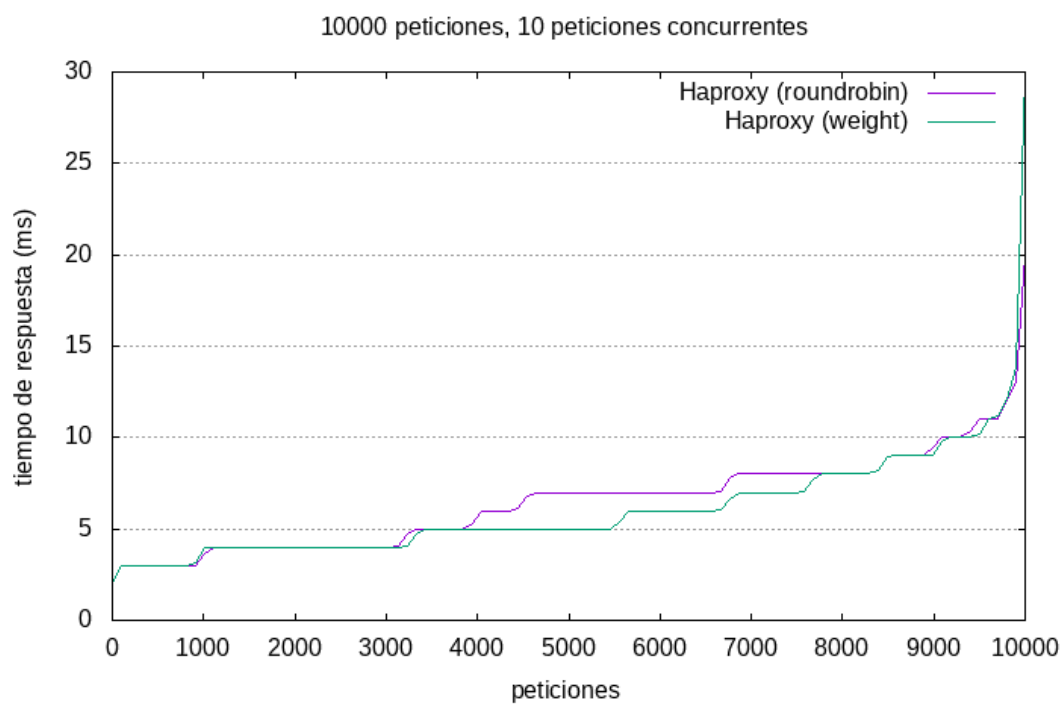
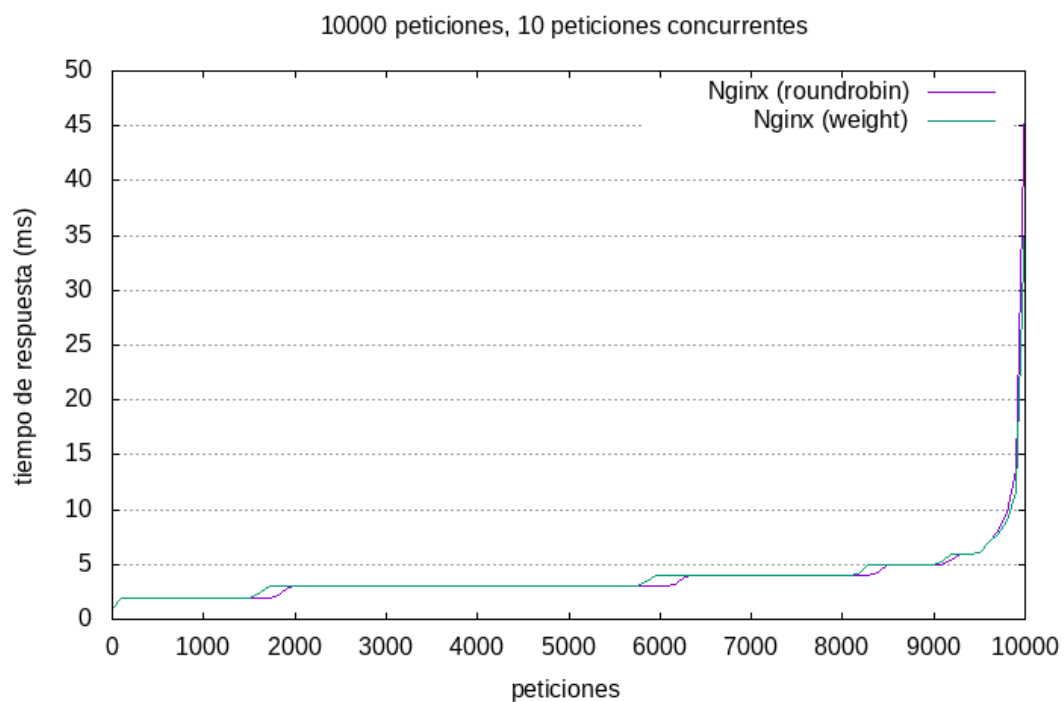
BusyBox v1.27.2 (Debian 1:1.27.2-3) built-in shell (ash)
Enter 'help' for a list of built-in commands.

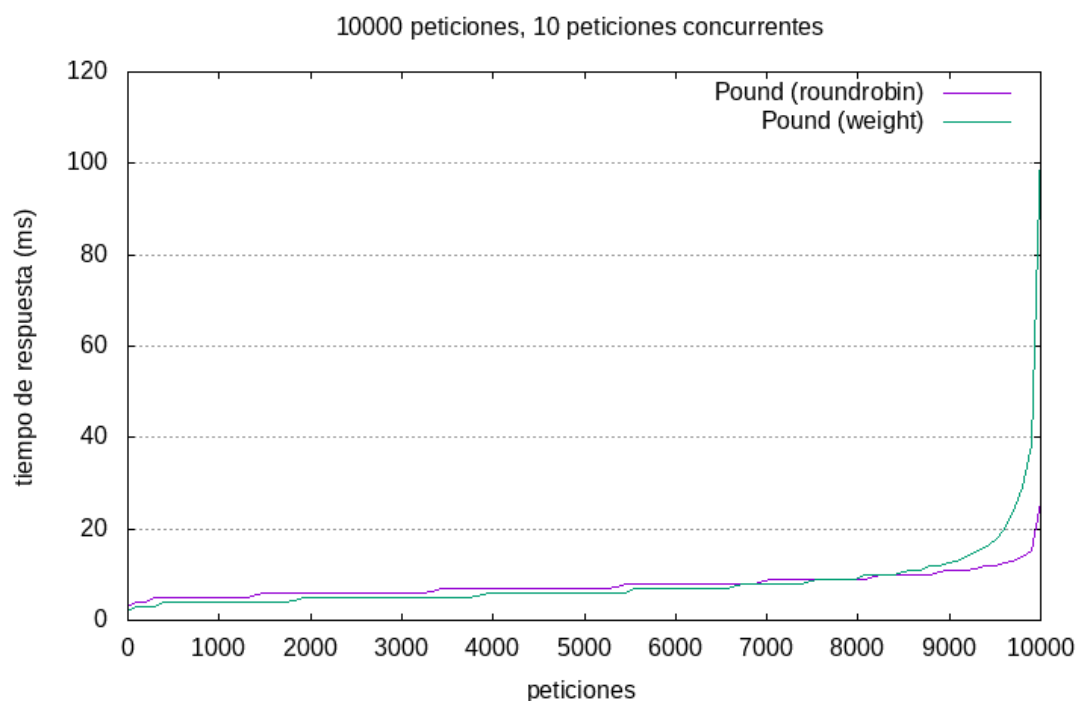
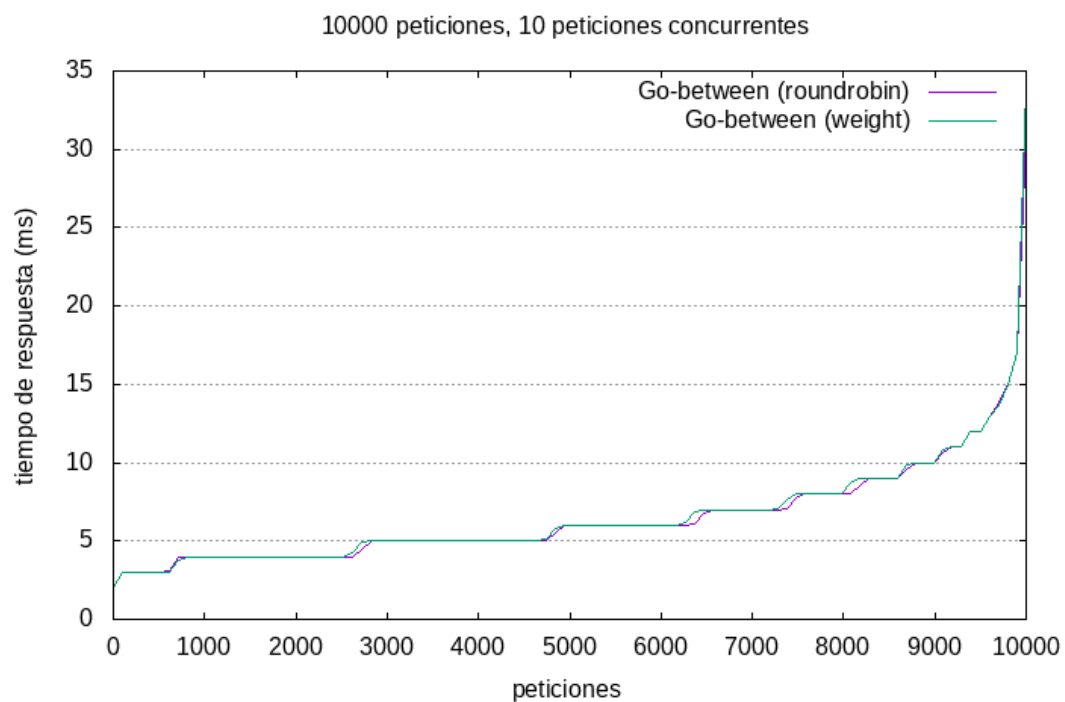
(initramfs)
```

Mi sospecha es que se trata de un problema con la instalación del grub. Pero teniendo en cuenta que el causante de este fallo es un reinicio de la máquina, y esta es la quinta vez que he tenido que reinstalarla, no voy a considerar más este balanceador. Podría ser una combinación de la configuración de instalación, o de su uso con VMWare. En cualquier caso, es el único de los balanceadores que hemos instalado que ha producido tantos problemas. Ha sido una mala experiencia de usuario.

7.2 Tiempos de respuesta

La opción que hemos habilitado a la hora de hacer el benchmark nos permite sacar unas gráficas sobre el tiempo de respuesta:





Estas gráficas son, en mi opinión, bastante más útiles que las medias. Nos muestran el tiempo de respuesta de cada petición con detalle. Es importante destacar que **se ordenan de menor a mayor** tiempo de respuesta. Esto nos permite comparar más a fondo cada balanceador.

Por ejemplo, podemos ver que **Pound** ha tardado bastante más en procesar un 10% de las peticiones, siendo el percentil 1 particularmente elevado. Esto hace que la media se agrande considerablemente.

Por otro lado, **Nginx** mantiene un 80% de las respuestas por debajo de 5 ms, lo cual compensa los 1%.

Lo contrario ocurre con **Haproxy y Go-between**. Sus resultados son más consistentes, pero no mantienen una media especialmente baja.

Estas gráficas también nos permiten ver claramente que **no hay apenas diferencia entre roundrobin y ponderación**. Esto es razonable, pues las dos máquinas usadas son idénticas y sirven el mismo contenido. La única diferencia es la carga que son capaces de soportar, teniendo en cuenta que ambas utilizan el mismo hardware virtualizado.

7.3 Sobre la fiabilidad de estos datos

Primero, dejemos claro cuál es el punto de esta sección: **no creo que estos datos sean de confianza**. Tampoco creo que sirvan para hacer un análisis justo.

A la hora de hacer pruebas, he notado que **el principal bottleneck es el host**. Se ha usado un portátil con un i5-8250U con 8GB de RAM en un NVME SSD. Para sacar los resultados, se han de tener abiertas 3 máquinas virtuales, con un editor (VSCode, basado en Electron) y algún que otro programa. Esto hace que se sature la RAM, y entre la paginación de disco en juego. Eso afecta severamente al benchmark. Por ejemplo, en la primera ejecución de [ab](#), se observaban tiempos de respuesta tan altos como 3 segundos, un valor absurdamente alto comparado con los 6ms de media. Tras varias ejecuciones, estos valores desaparecían. Mi hipótesis es que las máquinas entraban en primer plano, y el OS ponía en reposo algún otro proceso innecesario.

Otro punto importante es la ausencia de solidez del experimento. No creo que un balanceador se pueda analizar basándonos en un par de máquinas en el backend. Estos programas suelen ser muy complejos y están preparados para escalas considerablemente mayores. En contrapartida, este trabajo parece de juguete. Aún así, esto es normal, pues nos encontramos en una asignatura de universidad y no en un ámbito profesional. De todas formas, diferenciar entre roundrobin y ponderación en este caso puede resultar muy difícil.

7.4 Conclusión

Si tuviera que elegir un balanceador basándonos en lo realizado en este documento, **personalmente priorizaría la experiencia de usuario**.

Aunque Nginx claramente ha sido capaz de superar al resto de balanceadores en cuanto a peticiones por segundo, no sería capaz de juzgar cómo se comportaría en una hipotética escalada horizontal. Además, es un servicio preparado para servidores especialmente grandes. Quizás, un programa más compacto como Go-between o Haproxy funcionara mejor en mi caso.

En esencia, la conclusión es que **todos han cumplido su cometido con mayor o menor facilidad**. Destacan negativamente los problemas de Zevenet, que pueden ser debidos a un error por mi parte. Aún así, me parece que es importante destacar los problemas que se han producido, pues es un aspecto importante a considerar.

8 Bibliografía

- <https://www.cyberciti.biz/faq/systemd-systemctl-view-status-of-a-service-on-linux/>
- <https://linuxhint.com/what-is-keepalive-in-nginx/>
- <https://documentation.suse.com/smart/linux/html/reference-systemctl-enable-disable-services/index.html>
- <https://www.haproxy.com/blog/the-four-essential-sections-of-an-haproxy-configuration/#:~:text=There%20are>
- <https://support.ptc.com/help/thingworx/platform/r9/es/index.html#page/ThingWorx/Help/ThingWorxHighAvail>
- <https://www.haproxy.com/blog/exploring-the-haproxy-stats-page/>
- <https://cbonte.github.io/haproxy-dconv/1.7/configuration.html>
- https://www.wikiwand.com/es/Pound_-_Servidor_Proxy_Reverso
- <https://www.apsis.ch/pound.html>
- <https://www.tecmint.com/setting-up-pound-web-server-load-balancing-in-linux/>
- <https://help.ubuntu.com/community/Pound>