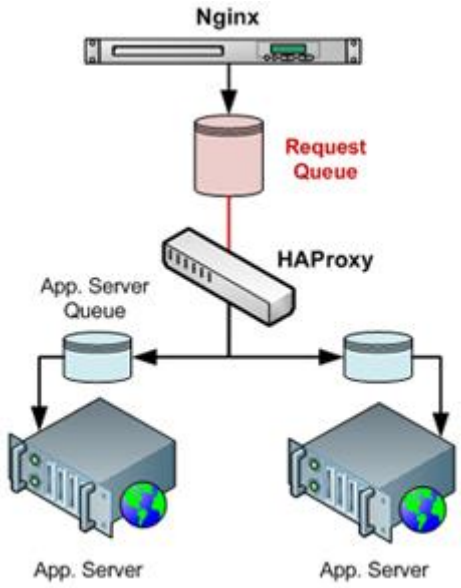
Prácticas HTTP – Balanceador HAProxy



1. Introdución a HPAProxy

HAProxy é un balanceador tcp e http moi rápido e confiable que soporta grandes cantidades de tráfico; e por esta razón, estase a usar nun gran número de sitios webs, entre os que están os máis visitados de Internet (Instagram, Twitter, Alibaba, etc.). Centrándonos nos apectos fundamentais hai que dicir que HAProxy pode traballar como balanceador de tipo L4 ou L7:

- **Balanceador L4**: estamos a falar dun **balanceo a nivel de transporte** (TCP), semellante ó visto con LVS no tema 4. Centrándonos nos sitios web, este tipo de balanceo está pensado para usar con servidores que teñen o mesmo contido, a mesma aplicación.
- **Balanceador L7**: estamos a falar dun **balanceo a nivel de aplicación**. Traballar a nivel de aplicación abre novas posibilidades ó poder acceder á capa máis próxima ó usuario. Centrándonos na web, poderemos tomar decisións en funcións das cabeceiras http dos paquetes, insertar cookies, marcar paquetes, ...; e polo tanto, **poderemos reenviar as solicitudes dos clientes a diferentes servidores en base ó contido da petición do cliente**. Un dos usos máis interesantes é a posibilidade de correr diferentes aplicacións en diferentes servidores reais pero accesibles baixo o mesmo nome de dominio e porto.

Nas seguintes imaxes vese un exemplo de balanceador L4 para dar servizo a dous servidores co mesmo contido e un balanceador L7 que da servizo a dúas aplicacións diferentes baixo o mesmo nome de dominio e porto (se o cliente solicita un recurso baixo /wordpress a súa petición é dirixida ós servidores wordpress-backend, noutro caso vai a web-backend):

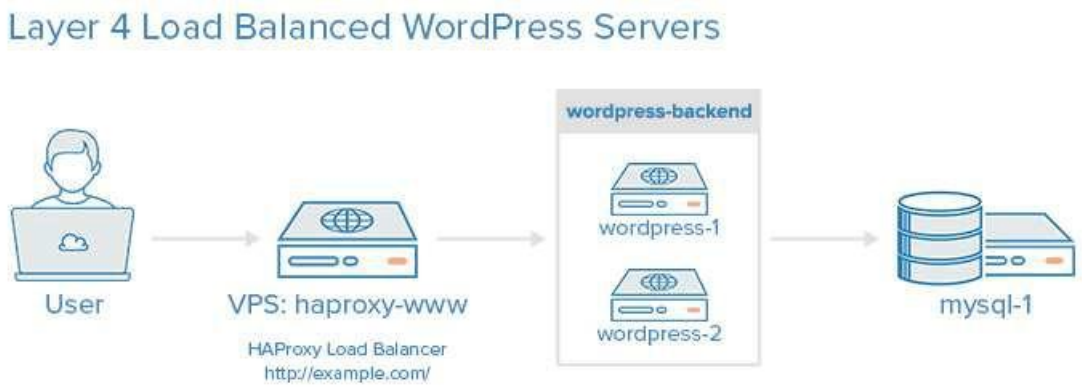


Fig. HAProxy como balanceador L4. Fonte: digitalocean.com

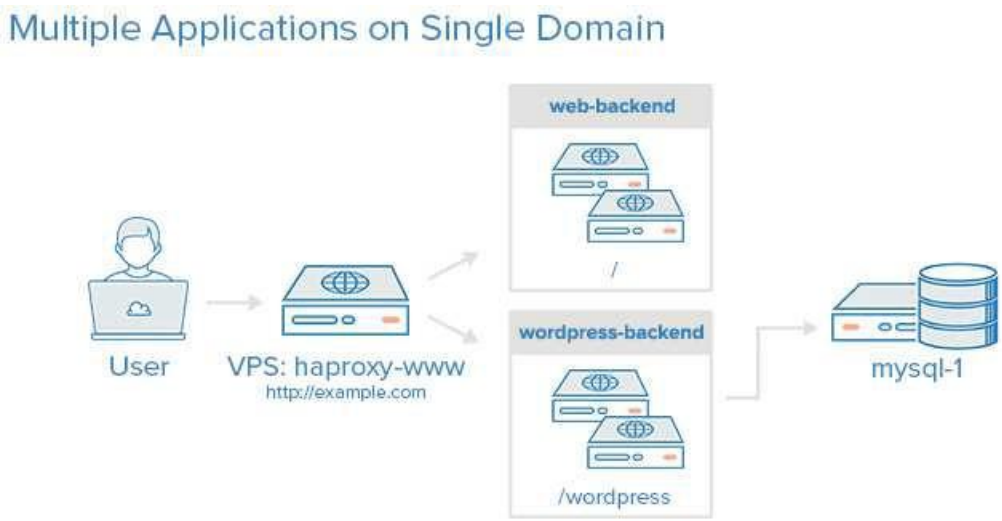


Fig. HAProxy como balanceador L7. Fonte: digitalocean.com

Un dos usos habituais de HAProxy é o de **terminador SSL/TLS**. Os clientes que precisen acceder a un servizo web usando comunicacións https atacan a HAProxy, e éste procede a descifrar a información e contactar cos servidores reais por http. Trátase de aforrar traballo ós servidores internos, que son os que realmente corren a aplicación web. HAProxy soporta SNI polo que podemos traballar con diferentes certificados para dar servizo a diferentes dominios

A seguinte imaxe proporciona unha visión deste concepto:

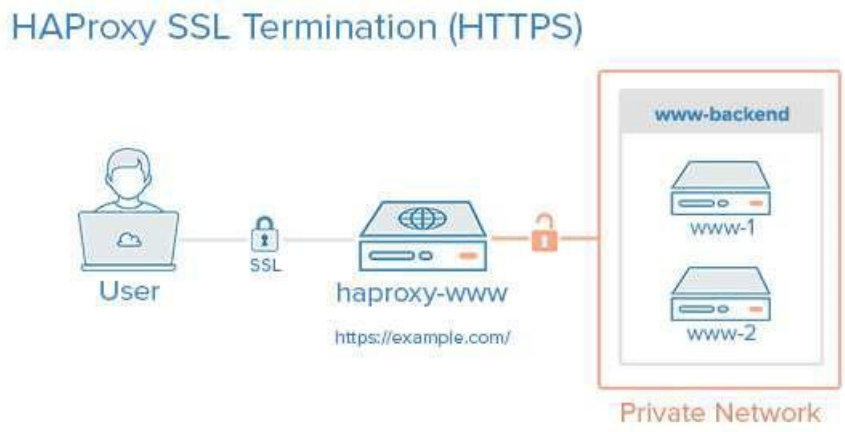


Fig. HAProxy como terminador SSL/TLS. Fonte: digitalocean.com

Á hora de instalar HAProxy debemos saber que sempre existen 3 versións sobre ás que se está a traballar: ás dúas últimas estables e unha nova en desenvolvemento. Nós instalaremos a última versión estable, pero ó non atoparse nos respoitorios oficiais, teremos que engadir aos repositorios da máquina HAProxy un novo repositorio[1]:

uadmin@ubuntu:~$ sudo add-apt-repository ppa:vbernat/haproxy-1.7

[sudo] password for uadmin:

HAProxy is a free, very fast and reliable solution offering high availability, load balancing, and proxy ……

This PPA contains packages for HAProxy 1.7 which is the current stable version. Más información: https://launchpad.net/~vbernat/+archive/ubuntu/haproxy-1.7

Pulse [Intro] para continuar o ctrl-c para cancelar

gpg: anillo «/tmp/tmpd2xo7_7p/secring.gpg» creado gpg: anillo «/tmp/tmpd2xo7_7p/pubring.gpg» creado

gpg: solicitando clave 1C61B9CD de hkp servidor keyserver.ubuntu.com

gpg: /tmp/tmpd2xo7_7p/trustdb.gpg: se ha creado base de datos de confianza

gpg: clave 1C61B9CD: clave pública "Launchpad PPA for Vincent Bernat" importada gpg: Cantidad total procesada: 1

gpg: importadas: 1 (RSA: 1)

OK

uadmin@ubuntu:~$ sudo apt-get update
uadmin@ubuntu:~$ sudo apt-get install haproxy

[1] Consultar https://haproxy.debian.net

Unha vez instalada a versión 1.7 podemos ver que se creou un directorio **`/etc/haproxy`** que contén os arquivos de configuración:

uadmin@ubuntu:~$ ls -lahF /etc/haproxy/

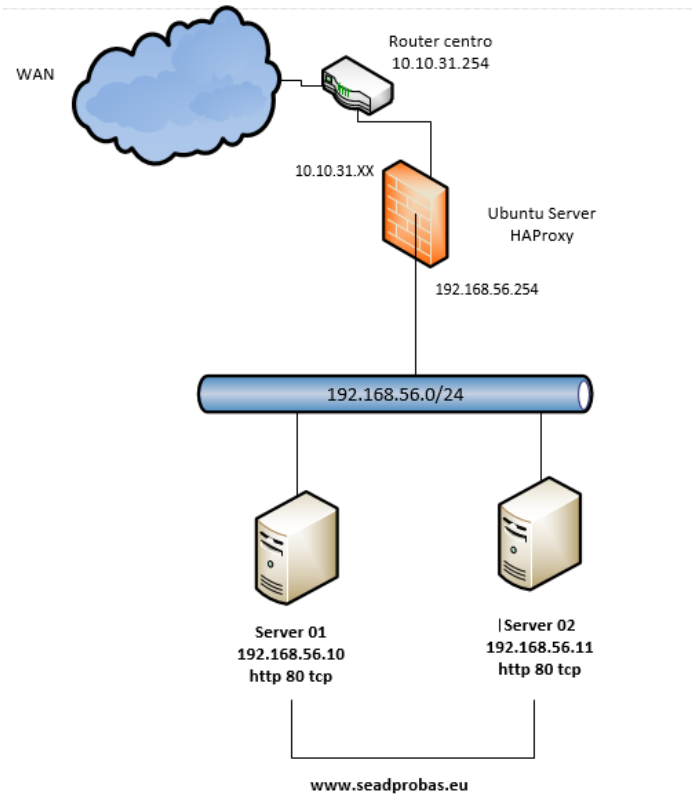
total 16K
drwxr-xr-x 3 root root 4,0K mar 7 00:56 ./ drwxr-xr-x 90 root root 4,0K mar 7 00:56 ../
drwxr-xr-x 2 root root 4,0K mar 7 00:56 errors/

- rw-r—r-- 1 root root 1,2K ene 1 12:38 haproxy.cfg

Antes de nada, faremos unha copia do arquivo de configuración orixinal por se temos que recuperalo:

uadmin@ubuntu:~$ sudo cp /etc/haproxy/haproxy.cfg /etc/haproxy/haproxy.cfg.orixinal

2. Escenario da práctica



Temos dous servidores web que supostamente son accesibles ao escribir www.seadprobas.eu na barra de localización dun navegador e que deberían de servir o mesmo contido.

No **FW** temos instalado o balanceador **HAProxy**.

A nivel de configuración de rede temos o seguinte:

- equipo FW:
  - Sistema Operativo Ubuntu Server 16.04 (32 bits).
  - Memoria RAM: 512 MBytes
  - NIC para a WAN:
    - En modo ponte (bridge):
    - Configurada coa IP **10.10.31.XX/16**. Esta IP terá a consideración de pública para o escenario. Interface configurada para ter saída a Internet.
  - NIC para a DMZ.
    - Rede **sólo-anfitrión**: empregaremos unha rede sólo-anfitrión para a DMZ; de xeito que, poderemos administrar vía rede os servidores directamente dende o equipo real.
    - Configurada coa IP 192.168.56.254/24 e sen gateway nin dns
  - Se queremos que os equipos da DMZ se conecten a Internet para descargar a paquetería debemos facer o seguinte:
    - para activar o enrutamento en FW debemos editar o arquivo /etc/sysctl.conf e modificar o valor da liña **net.ipv4.ip_forward = 1**. Aplicamos o comando **sysctl –p** para recargar a configuración e listo.
    - Executar a regra `iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -j MASQUERADE`

Para implementar os **servidores** da DMZ podemos usar máquinas virtuais correndo Ubuntu Server ou usar contenedores LXC. Usaranse servidores conectados á rede **sólo-anfitrión** para a DMZ cunha única tarxeta de rede rede configurada para ter saída a Internet a través de Ubuntu Linux HAProxy. A continuación aparece un exemplo do arquivo de configuración **`/etc/network/interfaces`** para **server01**:

```
# The loopback network interface auto lo
iface lo inet loopback

# The primary network interface auto eth0
iface eth0 inet static
address 192.168.56.10 netmask 255.255.255.0 gateway 192.168.56.254
dns-nameservers 8.8.8.8 8.8.4.4
```

Para instalar o servidor web Apache con soporte para PHP habería que executar os seguintes comandos:

julyov@server01:~$ sudo apt-get update

julyov@server01:~$ sudo apt-get install apache2 php libapache2-mod-php

Configurar a resolución DNS dende os clientes:

Para poder acceder ós sitios web dende os clientes hai que configurar a resolución DNS. Hai dúas solucións:

- Instalar un servidor dns cunha zona para o dominio seadprobas.eu
- Nos equipos cliente poden engadirse as entradas correspondentes aos hosts www.seadprobas.eu.

Aínda que non é a mellor solución, por ser un escenario de probas escolleuse a segunda por ser a máis rápida e sinxela. A modo de exemplo así quedaría o arquivo hosts dun cliente linux:

julyov@lubuntu:~$ cat /etc/hosts

```
127.0.0.1       localhost
127.0.1.1       ubuntu
10.10.31.XX     www.seadprobas.eu
```

3. Configuración dos servidores web

Virtualhosts

Como poderiamos configurar HAProxy como terminador SSL/TLS nos servidores Apache únicamente crearemos virtualhost para http. A continuación crearemos os virtualhost necesarios para cada servidor:

Server01 e server02

As accións indicadas a continuación hai que executalas en server01 e server02. A continuación crease o arquivo de configuración do sitio web para www.seadprobas.e u tomando como punto de partida o arquivo de configuración do sitio por defecto:

julyov@server:~$ sudo cp /etc/apache2/sites-available/000-default.conf /etc/apache2/sites-available/www.seadprobas.eu.conf

julyov@server:~$ sudo nano /etc/apache2/sites-available/www.seadprobas.eu.conf

<VirtualHost *:80>

ServerName www.seadprobas.eu

ServerAdmin webmaster@localhost

DocumentRoot /var/www/www.seadprobas.eu

ErrorLog ${APACHE_LOG_DIR}/www.seadprobas.eu_error.log

CustomLog ${APACHE_LOG_DIR}/wwww.seadprobas.eu_access.log combined

</VirtualHost>

Personalízanse as directivas:

- **Virtualhost**: indícase a IP e porto asociado ó sitio web.
- **ServerName**: nome de dominio asociado ó virtualhost.
- **DocumentRoot**: carpeta onde estarán os arquivos do sitio web.
- **ErrorLog** e **CustomLog**: para indicar os arquivos de logs do sitio web. Especificáronse uns propios para o sitio www.probas1.com para non compartilos con outros dominios.

Créase o directorio e a páxina principal do sirio www.seadprobas.eu:

julyov@server:~$ sudo mkdir /var/www/www.seadprobas.eu
julyov@server:~$ sudo nano /var/www/www. seadprobas.eu/index.php
<?php
echo “
<html>
<head>
<title>Sitio web en ${_SERVER[”SERVER_ADDR”]}</title>
</head>

```
<body>
<h1>Sitio Web www.seadprobas.eu</h1>

Est&aacute;s en ${_SERVER[”SERVER_ADDR”]}
</body>
</html>”;
?>
```

Activamos o sitio www.seadprobas.eu:

julyov@server:~$ sudo a2ensite www.seadprobas.eu.conf

Enabling site www.seadprobas.eu.

To activate the new configuration, you need to run:

service apache2 reload

julyov@server:~$ sudo service apache2 reload

- Reloading web server apache2 julyov@server:~$

Virtualhosts

Como configuraremos HAProxy como terminador SSL/TLS nos servidores Apache únicamente crearemos virtualhost para http. A continuación crearemos os virtualhost necesarios para cada servidor:

Server01 e server02

As accións indicadas a continuación hai que executalas en server01 e server02. A continuación crease o arquivo de configuración do sitio web para www.probas1.com tomando como punto de partida o arquivo de configuración do sitio por defecto:

julyov@server:~$ sudo cp /etc/apache2/sites-available/000-default.conf /etc/apache2/sites-available/www.seadprobas.eu.conf

julyov@server:~$ sudo nano /etc/apache2/sites-available/www.seadprobas.eu.conf

<VirtualHost *:80>

ServerName www.seadprobas.eu

ServerAdmin webmaster@localhost

DocumentRoot /var/www/www.seadprobas.eu

ErrorLog ${APACHE_LOG_DIR}/www.seadprobas.eu_error.log

CustomLog ${APACHE_LOG_DIR}/wwww.seadprobas.eu_access.log combined

</VirtualHost>

Personalízanse as directivas:

- **Virtualhost**: indícase a IP e porto asociado ó sitio web.
- **ServerName**: nome de dominio asociado ó virtualhost.
- **DocumentRoot**: carpeta onde estarán os arquivos do sitio web.
- **ErrorLog** e **CustomLog**: para indicar os arquivos de logs do sitio web. Especificáronse uns propios para o sitio www.probas1.com para non compartilos con outros dominios.

Créase o directorio e a páxina principal do sirio www.seadprobas.eu:

julyov@server:~$ sudo mkdir /var/www/www.seadprobas.eu
julyov@server:~$ sudo nano /var/www/www. seadprobas.eu/index.php
<?php
echo “
<html>
<head>
<title>Sitio web en ${_SERVER[”SERVER_ADDR”]}</title>
</head>
<body>
<h1>Sitio Web www.seadprobas.eu</h1>

Estás en ${_SERVER[”SERVER_ADDR”]}
</body>
</html>”;

```
?>
```

Activamos o sitio www.probas1.com:

julyov@server:~$ sudo a2ensite www.probas1.com.conf

```
Enabling site www.seadprobas.com.
```

To activate the new configuration, you need to run:

service apache2 reload

julyov@server:~$ sudo service apache2 reload

- Reloading web server apache2 julyov@server:~$

4. Configuración de HA_PROXY

Para configurar HAproxy hai tres elementos fundamentais:

- **ACLs (Acces Control List)**: permiten comprobar algunha condición e en base ao resultado realizarase unha acción.
- **Seccións Backend**: definen un conxunto de servidores que recibirán as solicitudes reenviadas por HAProxy.
- **Seccións Frontend**: definen como se reenvían as solicitudes ós backends.

Nesta primeira parte configuraremos o reenvío do tráfico http aos servidores internos e antes de editar o arquivo de configuración explicaremos como o faremos:

frontend sitio_http

```
bind 10.10.31.XX:80
acl host_www.seadprobas.eu_ hdr(host) -i www.seadprobas.eu
acl blog path_beg /blog
use_backend sitio_probas1_back if host_www.seadprobas.eu
```

- **`frontend sitio_http`** crea unha sección frontend; é dicir, onde atacarán os clientes.
- **`bind 10.10.200.10:80`** define o socket onde se recibirán as peticións dos clientes.
- **`acl host_www.seadprobas.eu hdr(host) -i www.seadprobas.eu`** crea unha ACL chamada host_www.seadprobas.eu onde se verifica que o contido da cabeceira http host é www.seadprobas.eu. Lembrar que nas solictudes http, a cabeceira Host úsase para indicar o sitio web no que está interesado o cliente.
- **`acl blog path_beg /blog`** crea unha ACL chamada blog que verifica que o Path da solicitude do cliente comeza por "/blog".
- As directivas **use_backend** permiten escoller un backend de servidores concreto en base ó cumprimento das condicións creadas coas ACLs definidas:
  - **`use_backend sitio_probas1_back if host_www.seadprobas.eu`** indica que as peticións serán enviadas ó backend chamado sitio_probas1_back se é certa a ACL `host_www.seadprobas.eu`; é dicir, se a solicitude vai dirixida ó sitio web www.seadprobas.eu.

Para definir ós servidores de backends teríamos que crear unhas seccións backends:

backend sitio_probas1_back

```
balance roundrobin
server server01 192.168.56.10:80 check
server server02 192.168.56.11:80 check
```

Escolleuse unha configuración moi sinxela para facela rápidamente comprensible:

backend sitio_probas1_back:

- **`balance roundrobin`** indica que os servidores que forma parte do backend estarán balanceados usando o algoritmo RoundRobin.
- **`server server01`** `192.168.56.10:80` **`check`** define o equipo `192.168.56.10` porto tcp/80 como membro do backend chamado sitio_probas1_back. Además, o seu estado será monitorizado por HAProxy
- **`server server02`** `192.168.56.11:80`**`check`** define o equipo `192.168.56.11` porto tcp/80 como membro do backend chamado sitio_probas1_back. Además, o seu estado será monitorizado por HAProxy.

Podemos comprobar que o frontend xunto cos backends definidos permiten satisfacer os requisitos en relación ao acceso por http. Posibles modificacións na configuración serían:

- `default_backend sitio_probas1_back` permitiría definir a que backend irían as solicitudes recibidas no frontend e que non casan con ningunha acl.
- `server server06 192.168.56.186:80 backup check` permitiría definir servidores de fallback engadindo nos backend unha liña deste tipo.

Editamos o arquivo de configuración para engadir as opcións de frontend e backend. As seccións global e defaults as deixamos, engadindo únicamente as directivas:

- **`option forwardfor`** para engadir a cabeceira X-Forwarded-For para permitir identificar ós servidores internos a IP do cliente real. Pensade que nos logs dos servidores web internos sempre aparecerá a IP do proxy como a IP do equipo que lle solicita un recurso; desta forma podemos saber a IP do equipo cliente real.
- **`option http-server-close`** permite mellorar o rendemento das conexións ó manterse o http keep-alive.

julyov@ubuntu:~$ sudo nano /etc/haproxy/haproxy.cfg

```
global
    log /dev/log local0
    log /dev/loglocal1 notice chroot /var/lib/haproxy
    stats socket /run/haproxy/admin.sock  mode 660 level admin
    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/
    ssl-default-bind-ciphers ECDH+AESGCM:DH+AESGCM:ECDH+AES256:DH+AES256:ECDH+AES128:DH+AES:ECDH+3DES:DH+
    ssl-default-bind-options no-sslv3
defaults
    log             global
    mode            http
    option          httplog
    option          dontlognull
contimeout 5000
clitimeout 50000
srvtimeout 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
retries 3
option forwardfor
option http-server-close

frontend sitio_http
    bind 10.10.200.10:80
    acl host_www.seadprobas.eu hdr(host) -i www.seadprobas.eu
    acl blog path_beg /blog
    use_backend sitio_probas1_back if host_www.seadprobas.eu
backend sitio_probas1_back balance roundrobin
server server01 192.168.56.10:80 check
server server02 192.168.56.11:80 check
```

Antes de reiniciar o servizo de HAProxy modificaremos o syslog para gardar os rexistros do noso balanceador. Editamos o arquivo `/etc/rsyslog.conf` e habilitamos o syslog por UDP:

```
julyov@ubuntu:~$ sudo nano /etc/rsyslog.conf
...
# provides UDP syslog reception
$ModLoad imudp
$UDPServerRun 514
$UDPServerAddress 127.0.0.1
...
julyov@ubuntu:~$ sudo service rsyslog restart
```

Unha vez feito esto, reiniciamos o servizo HAProxy e verificamos no arquivo de log o estado dos backends:

```
julyov@ubuntu:~$ sudo service haproxy restart
julyov@ubuntu:~$ tail /var/log/haproxy.log
```

```
Mar 7 23:55:40 ubuntu haproxy[2332]: Proxy sitio_http started. Mar 7 23:55:40 ubuntu haproxy[2332]: Proxy sitio_http started.

Mar 7 23:55:40 ubuntu haproxy[2332]: Proxy sitio_probas1_back started. Mar 7 23:55:40 ubuntu haproxy[2332]: Proxy sitio_probas1_back started. Mar 7 23:55:40 ubuntu haproxy[2332]: Proxy sitio_probas2_back started. Mar 7 23:55:40 ubuntu haproxy[2332]: Proxy sitio_probas2_back started. Mar 7 23:55:40 ubuntu haproxy[2332]: Proxy sitio_probas3_back started. Mar 7 23:55:40 ubuntu haproxy[2332]: Proxy sitio_probas3_back started.

Mar 7 23:55:40 ubuntu haproxy[2332]: Proxy sitio_probas3_blog_back started.
```

Se imos cun navegador ou coa utilidade curl poderemos comprobar o correcto funcionamento do balanceo: