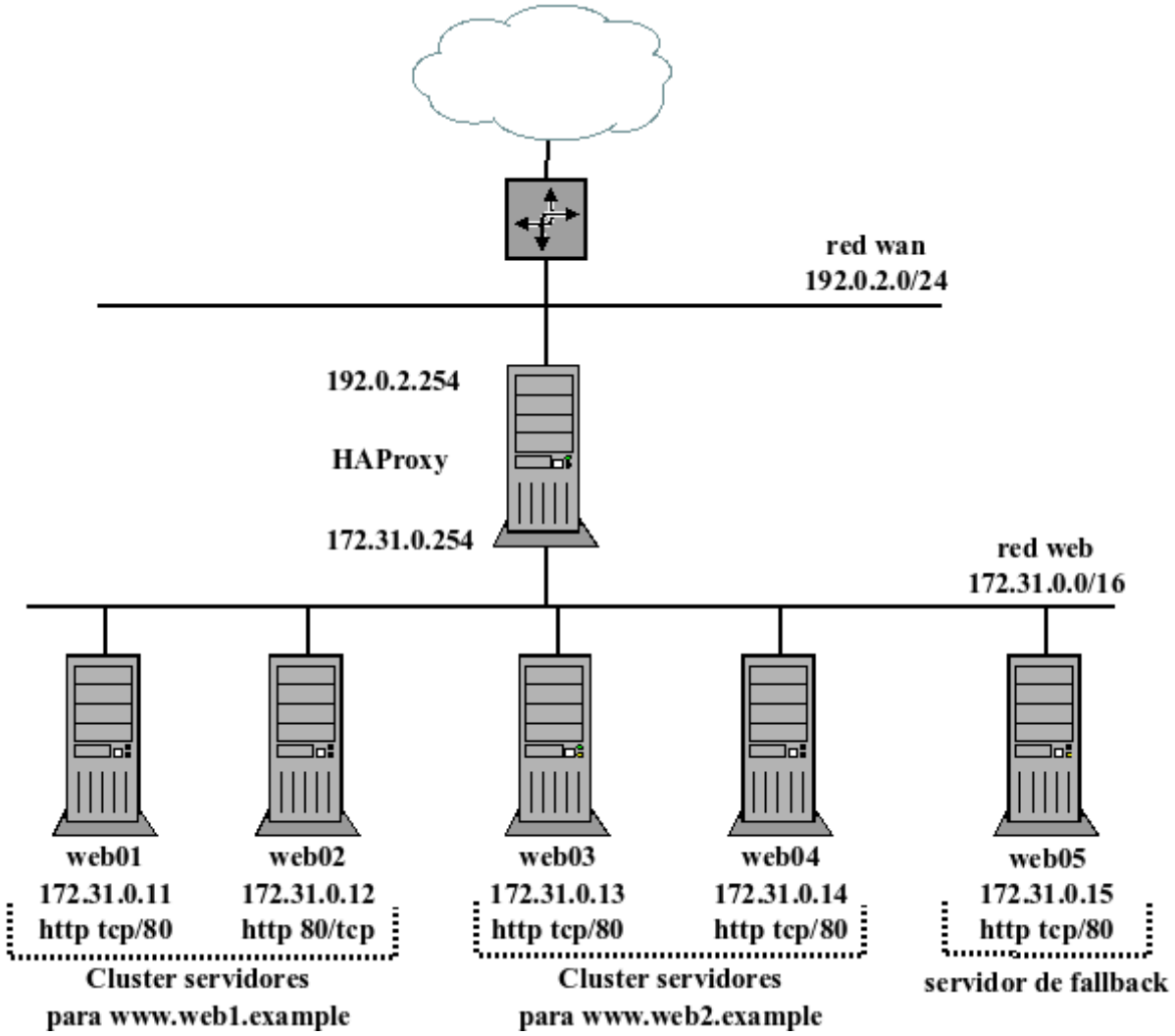
Usaremos HAProxy para dar servicio a la siguiente infraestructura:



CREACIÓN ESCENARIO

Descargar y descomprimir el archivo de creración del escenario: `haproxy_alpine_network.zip`

```
manuel@ideapad5:~/Descargas/haproxy_alpine_network$ ls -lhF
total 16M
-rw-r--r-- 1 manuel manuel  16M abr 27  2022 alpine-php.tar.gz
-rw-r--r-- 1 manuel manuel  531 feb 15 20:45 config.yml_2NICS_PLANTILLA
-rw-r--r-- 1 manuel manuel  466 abr 29  2022 config.yml_PLANTILLA
-rwxr-xr-x 1 manuel manuel 3,2K may 10 10:09 escenario_HA_alpine.sh
-rw-r--r-- 1 manuel manuel  119 jun  9  2020 interfaces_PLANTILLA
-rw-rw-r-- 1 manuel manuel  409 feb 15 20:51 network.yaml_PLANTILLA
-rw-r--r-- 1 manuel manuel  292 abr 29  2022 profile_WAN-WEB
-rw-r--r-- 1 manuel manuel  222 abr 29  2022 profile_WEB
-rw-r--r-- 1 manuel manuel   38 dic  9  2019 resolv.conf
manuel@ideapad5:~/Descargas/haproxy_alpine_network$ chmod 755 escenario_HA_alpine.sh
manuel@ideapad5:~/Descargas/haproxy_alpine_network$ ./escenario_HA_alpine.sh
```

Seleccionar la opicón 1 para crear el escenario. El resultado será un contenedor haproxy01 corriendo Ubuntu 20.04 y 5 contenedores corriendo Alpine Linux con el servidor Apache instalado y preparado para la práctica:

```
manuel@ideapad5:~/Descargas/haproxy_alpine_network$ lxc ls -c n,s,4,P
+-----------+---------+---------------------+----------+
|   NAME    |  STATE  |        IPV4         | PROFILES |
```

```
+-----------+---------+---------------------+----------+
| haproxy01 | RUNNING | 192.0.2.254 (eth0)  | WAN-WEB  |
|           |         | 172.31.0.254 (eth1) |          |
+-----------+---------+---------------------+----------+
| web01     | RUNNING | 172.31.0.11 (eth0)  | WEB      |
+-----------+---------+---------------------+----------+
| web02     | RUNNING | 172.31.0.12 (eth0)  | WEB      |
+-----------+---------+---------------------+----------+
| web03     | RUNNING | 172.31.0.13 (eth0)  | WEB      |
+-----------+---------+---------------------+----------+
| web04     | RUNNING | 172.31.0.14 (eth0)  | WEB      |
+-----------+---------+---------------------+----------+
| web05     | RUNNING | 172.31.0.15 (eth0)  | WEB      |
+-----------+---------+---------------------+----------+
```

HAPROXY

A la hora de instalar HAProxy debemos saber que siempre existen varias versiones sobre las que se está trabajando: las últimas estables y una nueva en desarrollo. En Ubuntu 20.04 para poder acceder a las últimas versiones es necesario añadir un nuevo repositorio:

```
ubuntu@haproxy:~$ sudo add-apt-repository ppa:vbernat/haproxy-2.6
 HAProxy is a free, very fast and reliable solution offering high availability, load balancing, and proxying for TCP and HTTP-based applications. It is particularly suited for web sites crawling under very high loads while needing persistence or Layer7 processing. Supporting tens of thousands of connections is clearly realistic with todays hardware. Its mode of operation makes its integration into existing architectures very easy and riskless, while still offering the possibility not to expose fragile web servers to the Net.


This PPA contains packages for HAProxy 2.6.
 More info: https://launchpad.net/~vbernat/+archive/ubuntu/haproxy-2.6
Press [ENTER] to continue or Ctrl-c to cancel adding it.


Hit:1 http://archive.ubuntu.com/ubuntu focal InRelease
Hit:2 http://security.ubuntu.com/ubuntu focal-security InRelease
Get:3 http://ppa.launchpad.net/vbernat/haproxy-2.6/ubuntu focal InRelease [23.8 kB]
Hit:4 http://archive.ubuntu.com/ubuntu focal-updates InRelease
Hit:5 http://archive.ubuntu.com/ubuntu focal-backports InRelease
Get:6 http://ppa.launchpad.net/vbernat/haproxy-2.6/ubuntu focal/main amd64 Packages [1008 B]
Get:7 http://ppa.launchpad.net/vbernat/haproxy-2.6/ubuntu focal/main Translation-en [704 B]
...
Fetched 25.5 kB in 1s (18.3 kB/s)
Reading package lists... Done~
```

Comprobamos que ahora está disponible a versión 2.6 y procedemos a instalarla:

```
ubuntu@haproxy:~$  apt show haproxy
Package: haproxy
Version: 2.6.13-1ppa1~focal
Priority: optional
Section: net
Maintainer: Debian HAProxy Maintainers <team+haproxy@tracker.debian.org>
Installed-Size: 3.995 kB
Pre-Depends: dpkg (>= 1.17.14), init-system-helpers (>= 1.54~)
```

```
Depends: libc6 (>= 2.17), libcrypt1 (>= 1:4.1.0), liblua5.3-0, libpcre2-8-0 (>= 10.22), libssl1.1 (>= 1.1.1), libsystemd0, adduser
Suggests: vim-haproxy, haproxy-doc
Download-Size: 1.693 kB
APT-Sources: http://ppa.launchpad.net/vbernat/haproxy-2.6/ubuntu focal/main amd64 Packages
Description: carga equilibrada del proxy rápida y fiable
 HAProxy es un proxy TCP/HTTP  adecuado a entornos de gran disponibilidad.
 Se realiza la conexión a través de HTTP cookies, carga equilibrada,
 encabezado, modificación y borrado en ambos sentidos. Lo que proporciona
 opciones para configurar el estado del servidor.

N: Hay 3 registros adicionales. Utilice la opción «-a» para verlos.
ubuntu@haproxy:~$ sudo apt install haproxy
```

Una vez instalada la versión 2.6 podemos ver que se creó un directorio `/etc/haproxy` que contiene los archivos de configuración:

```
ubuntu@haproxy:~$ ls -lahF /etc/haproxy/
total 9.0K
drwxr-xr-x  3 root root    4 Apr 27 20:56 ./
drwxr-xr-x 95 root root  185 Apr 27 20:56 ../
drwxr-xr-x  2 root root    9 Apr 27 20:56 errors/
-rw-r--r--  1 root root 1.3K Mar 14 22:04 haproxy.cfg
```

Antes de nada, se hará una copia del archivo de configuración original por si tenemos que recuperarlo:

```
ubuntu@haproxy:~$ sudo cp /etc/haproxy/haproxy.cfg /etc/haproxy/haproxy.cfg.original
```

Balanceando http

Para configurar HAproxy hay tres elementos fundamentales:

- **ACLs (Acces Control List)**: permiten comprobar alguna condición y en base al resultado realizar una acción.
- **Secciones Backend**: definen un conjunto de servidores que recibirán las solicitudes reenviadas por HAProxy.
- **Secciones Frontend**: definen como se reenvían las solicitudes a los backends.

Antes de editar el archivo de configuración se explicará como se hace el reenvío del tráfico http:

```
frontend sitio_http
   bind 192.0.2.254:80
   acl host_web1 hdr(host) -i www.web1.example
   acl host_web2 hdr(host) -i www.web2.example
   use_backend sitio_web1 if host_web1
   use_backend sitio_web2 if host_web2
```

Donde:

- `frontend sitio_http` crea una sección *frontend*; es decir, donde atacarán los clientes.
- `bind 192.0.2.254:80` define el socket donde se recibirán las peticiones de los clientes.
- `acl host_web1 hdr(host) -i www.web1.example` crea una ACL llamada `host_web1` donde se verifica que el contenido de la cabecera http Host es `www.web1.example`. Recordad que en las solicitudes http, la cabecera Host se usa para indicar el sitio web en el que está interesado el cliente.

```
GET / HTTP/1.1
```

```
Host: www.web1.example
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:99.0) Gecko/20100101 Firefox/99.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
```

- `acl host_web2 hdr(host) -i www.web2.example` crea una ACL llamada `host_web2` donde se verifica que el contenido de la cabecera http Host es `www.web2.example`.
- Las directivas use_backend permiten escoger un backend de servidores concreto en base al cumplimiento de las condiciones creadas con las ACLs definidas:
  - `use_backend sitio_web1 if host_web1` indica que las peticiones serán enviadas al backend llamado `sitio_web1` si es cierta la ACL `host_web1`; es decir, si la solicitud va dirigida al sitio web `www.web1.example`.
  - `use_backend sitio_web2 if host_web2` indica que las peticiones serán enviadas al backend llamado `sitio_web2` si es cierta la ACL `host_web2`; es decir, si la solicitud va dirigida al sitio web `www.web2.example`.

Como se indica en la documentación de HAProxy:

"*There may be as many 'use_backend' rules as desired. All of these rules are evaluated in their declaration order, and the first one which matches will assign the backend.*"

Para definir los servidores de backends tendríamos que crear unas secciones backends:

```
backend sitio_web1
  balance roundrobin
  server web01 172.31.0.11:80 check
  server web02 172.31.0.12:80 check
  server web05 172.31.0.15:80 backup check
backend sitio_web2
   balance roundrobin
   server web03 172.31.0.13:80 check
   server web04 172.31.0.14:80 check
   server web05 172.31.0.15:80 backup check
```

Tomando como ejemplo la configuración del `backend sitio_web1` tendríamos:

- `balance roundrobin` indica que los servidores que forman parte del backend estarán balanceados usando el algoritmo RoundRobin.
- `server web01 172.31.0.11:80 check` define al equipo 172.31.0.11 puerto tcp/80 como miembro del backend llamado `sitio_web1`. Además, su estado será monitorizado por HAProxy[1].
- `server web02 172.31.0.12:80 check` define al equipo 172.31.0.12 puerto tcp/80 como miembro del backend llamado `sitio_web1`. Además, su estado será monitorizado por HAProxy.
- `server web05 172.31.0.15:80 backup check` define al equipo 172.31.0.15 puerto tcp/80 como un servidor de backup o *fallback* para el cluster. La idea de un servidor de backup es que cuando todos los servidores del cluster estén caídos, web05 atenderá las peticiones de los clientes mostrando una página de error o una versión del sitio web degradada.

[1] https://www.haproxy.com/blog/how-to-enable-health-checks-in-haproxy/

El frontend junto con los backends definidos permiten satisfacer los requisitos en relación al acceso por http. Posibles modificaciones en la configuración serían:

- `default_backend sitio_web1` permitiría definir a que backend irían las solicitudes recibidas en el frontend y que no casan con ninguna acl.
- `option httpchk` permite configurar comprobaciones más avanzadas del estado de los servidores del cluster lanzando consultas http. Si el código de respuesta recibida es de tipo 2xx o 3xx se considera que el servidor del backend está operativo, en otro caso se le considera caído.

Editamos el archivo de configuración para añadir las opciones de frontend y backend. Las secciones global y defaults las dejamos, añadiendo únicamente las directivas:

- `option forwardfor` para añadir la cabecera `X-Forwarded-For` para permitir identificar a los servidores internos la IP del cliente real. Pensad que en los logs de los servidores web internos siempre aparecerá la IP del proxy como la IP del equipo que le solicita un recurso; de esta forma podemos saber la IP del equipo cliente real.
- `option http-server-close` permite mejorar el rendimiento de las conexiones al mantenerse el http keep-alive.

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

      # See: https://ssl-config.mozilla.org/#server=haproxy&server-
version=2.0.3&config=intermediate
        ssl-default-bind-ciphers ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-
SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-
POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384
        ssl-default-bind-ciphersuites
TLS_AES_128_GCM_SHA256:TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256
        ssl-default-bind-options ssl-min-ver TLSv1.2 no-tls-tickets

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
      option forwardfor
```

```
      option  http-server-close

frontend sitio_http
   bind 192.0.2.254:80
   acl host_web1 hdr(host) -i www.web1.example
   acl host_web2 hdr(host) -i www.web2.example
   use_backend sitio_web1 if host_web1
   use_backend sitio_web2 if host_web2

backend sitio_web1
  balance roundrobin
  server web01 172.31.0.11:80 check
  server web02 172.31.0.12:80 check
  server web05 172.31.0.15:80 backup check

backend sitio_web2
   balance roundrobin
   server web03 172.31.0.13:80 check
   server web04 172.31.0.14:80 check
   server web05 172.31.0.15:80 backup check
```

Reiniciamos el servicio HAProxy y verificamos en el archivo de log el estado de los backends:

```
ubuntu@haproxy:~$ sudo systemctl restart haproxy.service
ubuntu@haproxy:~$ cat /var/log/haproxy.log
Apr 29 10:31:49 haproxy01 haproxy[1079]: [WARNING]  (1079) : Exiting Master process...
Apr 29 10:31:49 haproxy01 haproxy[1079]: [NOTICE]   (1079) : haproxy version is 2.6.15-1ppa1~focal
Apr 29 10:31:49 haproxy01 haproxy[1079]: [NOTICE]   (1079) : path to executable is /usr/sbin/haproxy
Apr 29 10:31:49 haproxy01 haproxy[1079]: [ALERT]    (1079) : Current worker #1 (1081) exited with code 143 (Terminated)
Apr 29 10:31:49 haproxy01 haproxy[1079]: [WARNING]  (1079) : All workers exited. Exiting... (0)
Apr 29 10:31:49 haproxy01 haproxy[1356]: [NOTICE]   (1356) : New worker #1 (1358) forked
```

PRUEBAS FUNCIONAMIENTO

Balanceo

Para que los clientes puedan acceder a `www.web1.example` y `www.web2.example` es necesario que se produzca una resolución DNS que traduzca esos nombres a la dirección IP pública del balanceados (192.0.2.254). En nuestro entorno de pruebas configuraremos la resolución DNS usando el fichero hosts del equipo cliente, donde añadimos la siguiente línea:

```
ubuntu@cliente:~$ sudo nano /etc/hosts
127.0.0.1 localhost
# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
192.0.2.254 www.web1.example www.web2.example
```

Se puede comprobar que la resolución funciona:

```
ubuntu@cliente:~$ ping -c 2 www.web1.example
PING www.web1.example (192.0.2.254) 56(84) bytes of data.
64 bytes from www.web1.example (192.0.2.254): icmp_seq=1 ttl=64 time=0.218 ms
64 bytes from www.web1.example (192.0.2.254): icmp_seq=2 ttl=64 time=0.110 ms


--- www.web1.example ping statistics ---
```

```
2 packets transmitted, 2 received, 0% packet loss, time 1013ms
rtt min/avg/max/mdev = 0.110/0.164/0.218/0.054 ms
ubuntu@cliente:~$ ping -c 2 www.web2.example
PING www.web1.example (192.0.2.254) 56(84) bytes of data.
64 bytes from www.web1.example (192.0.2.254): icmp_seq=1 ttl=64 time=0.146 ms
64 bytes from www.web1.example (192.0.2.254): icmp_seq=2 ttl=64 time=0.111 ms


--- www.web1.example ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1021ms
rtt min/avg/max/mdev = 0.111/0.128/0.146/0.017 ms
ubuntu@cliente:~$
```

Si probamos a acceder con un navegador podremos comprobar como nuestras peticiones terminan en los servidores reales (con el balanceo correspondiente).

```
ubuntu@cliente:~$ curl http://www.web1.example
<html>
<head>
       <title>Sitio web en 172.31.0.11</title>
</head>
<body>
Est&aacute;s en el sitio www.web1.example con la IP: 172.31.0.11
</body>
</html>
ubuntu@cliente:~$ curl http://www.web1.example
<html>
<head>
       <title>Sitio web en 172.31.0.12</title>
</head>
<body>
Est&aacute;s en el sitio www.web1.example con la IP: 172.31.0.12
</body>
</html>
ubuntu@cliente:~$ curl http://www.web1.example
<html>
<head>
       <title>Sitio web en 172.31.0.11</title>
</head>
<body>
Est&aacute;s en el sitio www.web1.example con la IP: 172.31.0.11
</body>
</html>
ubuntu@cliente:~$ curl http://www.web1.example
<html>
<head>
       <title>Sitio web en 172.31.0.12</title>
</head>
<body>
Est&aacute;s en el sitio www.web1.example con la IP: 172.31.0.12
</body>
</html>
ubuntu@cliente:~$ curl http://www.web2.example
<html>
<head>
       <title>Sitio web en 172.31.0.13</title>
</head>
```

```
<body>
Est&aacute;s en el sitio www.web2.example con la IP: 172.31.0.13
</body>
</html>
ubuntu@cliente:~$ curl http://www.web2.example
<html>
<head>
	<title>Sitio web en 172.31.0.14</title>
</head>
<body>
Est&aacute;s en el sitio www.web2.example con la IP: 172.31.0.14
</body>
</html>
ubuntu@cliente:~$ curl http://www.web2.example
<html>
<head>
	<title>Sitio web en 172.31.0.13</title>
</head>
<body>
Est&aacute;s en el sitio www.web2.example con la IP: 172.31.0.13
</body>
</html>
ubuntu@cliente:~$ curl http://www.web2.example
<html>
<head>
	<title>Sitio web en 172.31.0.14</title>
</head>
<body>
Est&aacute;s en el sitio www.web2.example con la IP: 172.31.0.14
</body>
```

Estas solicitudes quedan reflejadas también en los logs de HAProxy:

```
ubuntu@haproxy01:~$ tail -f /var/log/haproxy.log
Apr  29  10:38:06  haproxy01  haproxy[1358]:  192.0.2.156:48538  [29/Apr/2022:10:38:06.449]  sitio_http
sitio_web1/web01 0/0/1/2/3 200 320 - - ---- 1/1/0/0/0 0/0 "GET / HTTP/1.1"
Apr  29  10:38:07  haproxy01  haproxy[1358]:  192.0.2.156:48540  [29/Apr/2022:10:38:07.363]  sitio_http
sitio_web1/web02 0/0/0/1/1 200 320 - - ---- 1/1/0/0/0 0/0 "GET / HTTP/1.1"
Apr  29  10:39:17  haproxy01  haproxy[1358]:  192.0.2.156:48542  [29/Apr/2022:10:39:17.343]  sitio_http
sitio_web1/web01 0/0/0/0/0 200 320 - - ---- 1/1/0/0/0 0/0 "GET / HTTP/1.1"
Apr  29  10:39:17  haproxy01  haproxy[1358]:  192.0.2.156:48544  [29/Apr/2022:10:39:17.875]  sitio_http
sitio_web1/web02 0/0/0/0/0 200 320 - - ---- 1/1/0/0/0 0/0 "GET / HTTP/1.1"
Apr  29  10:39:36  haproxy01  haproxy[1358]:  192.0.2.156:48546  [29/Apr/2022:10:39:36.931]  sitio_http
sitio_web2/web03 0/0/0/1/1 200 320 - - ---- 1/1/0/0/0 0/0 "GET / HTTP/1.1"
Apr  29  10:39:37  haproxy01  haproxy[1358]:  192.0.2.156:48550  [29/Apr/2022:10:39:37.602]  sitio_http
sitio_web2/web04 0/0/0/1/1 200 320 - - ---- 1/1/0/0/0 0/0 "GET / HTTP/1.1"
Apr  29  10:39:38  haproxy01  haproxy[1358]:  192.0.2.156:48552  [29/Apr/2022:10:39:38.242]  sitio_http
sitio_web2/web03 0/0/25/0/25 200 320 - - ---- 1/1/0/0/0 0/0 "GET / HTTP/1.1"
Apr  29  10:39:38  haproxy01  haproxy[1358]:  192.0.2.156:48554  [29/Apr/2022:10:39:38.835]  sitio_http
sitio_web2/web04 0/0/0/0/0 200 320 - - ---- 1/1/0/0/0 0/0 "GET / HTTP/1.1"
```

Caída de servidores

Si cae el servidor web01 del primer cluster, HAProxy lo va a detectar y enviará todas las peticiones a www.web1.exampe al servidor web02:

```
ubuntu@haproxy01:~$ tail -f /var/log/haproxy.log
```

Apr 29 10:55:16 haproxy01 haproxy[1358]: **Server sitio_web1/web01 is DOWN**, reason: Layer4 timeout, check duration: 2000ms. 1 active and 1 backup servers left. 0 sessions active, 0 requeued, 0 remaining in queue.

Apr 29 10:55:16 haproxy01 haproxy[1358]: [WARNING] (1358) : **Server sitio_web1/web01 is DOWN**, reason: Layer4 timeout, check duration: 2000ms. 1 active and 1 backup servers left. 0 sessions active, 0 requeued, 0 remaining in queue.

Apr 29 10:55:16 haproxy01 haproxy[1358]: S**erver sitio_web1/web01 is DOWN,** reason: Layer4 timeout, check duration: 2000ms. 1 active and 1 backup servers left. 0 sessions active, 0 requeued, 0 remaining in queue.

Apr 29 10:55:37 haproxy01 haproxy[1358]: 192.0.2.156:48980 [29/Apr/2022:10:55:37.904] sitio_http **sitio_web1/web02** 0/0/0/0/0 200 320 - - ---- 1/1/0/0/0 0/0 "GET / HTTP/1.1"

Apr 29 10:55:39 haproxy01 haproxy[1358]: 192.0.2.156:48982 [29/Apr/2022:10:55:39.119] sitio_http **sitio_web1/web02** 0/0/0/0/0 200 320 - - ---- 1/1/0/0/0 0/0 "GET / HTTP/1.1"

Apr 29 10:55:42 haproxy01 haproxy[1358]: 192.0.2.156:48984 [29/Apr/2022:10:55:42.256] sitio_http **sitio_web1/web02** 0/0/0/0/0 200 320 - - ---- 1/1/0/0/0 0/0 "GET / HTTP/1.1"

Apr 29 10:55:43 haproxy01 haproxy[1358]: 192.0.2.156:48986 [29/Apr/2022:10:55:43.473] sitio_http **sitio_web1/web02** 0/0/0/0/0 200 320 - - ---- 1/1/0/0/0 0/0 "GET / HTTP/1.1"

En cuanto web01 vuelva a estar opertativo, HAProxy lo detecta y procede a enviarle peticiones retomando el balanceo:

Apr 29 10:57:16 haproxy01 haproxy[1358]: S**erver sitio_web1/web01 is UP**, reason: Layer4 check passed, check duration: 0ms. 2 active and 1 backup servers online. 0 sessions requeued, 0 total in queue.

Apr 29 10:57:16 haproxy01 haproxy[1358]: [WARNING] (1358) : **Server sitio_web1/web01 is UP**, reason: Layer4 check passed, check duration: 0ms. 2 active and 1 backup servers online. 0 sessions requeued, 0 total in queue.

Apr 29 10:57:16 haproxy01 haproxy[1358]: **Server sitio_web1/web01 is UP**, reason: Layer4 check passed, check duration: 0ms. 2 active and 1 backup servers online. 0 sessions requeued, 0 total in queue.

Apr 29 10:58:03 haproxy01 haproxy[1358]: 192.0.2.156:48988 [29/Apr/2022:10:58:03.840] sitio_http **sitio_web1/web02** 0/0/0/1/1 200 320 - - ---- 1/1/0/0/0 0/0 "GET / HTTP/1.1"

Apr 29 10:58:04 haproxy01 haproxy[1358]: 192.0.2.156:48990 [29/Apr/2022:10:58:04.348] sitio_http **sitio_web1/web01** 0/0/0/0/0 200 320 - - ---- 1/1/0/0/0 0/0 "GET / HTTP/1.1"

Apr 29 10:58:16 haproxy01 haproxy[1358]: 192.0.2.156:48992 [29/Apr/2022:10:58:16.864] sitio_http **sitio_web1/web02** 0/0/0/0/0 200 320 - - ---- 1/1/0/0/0 0/0 "GET / HTTP/1.1"

Apr 29 10:58:17 haproxy01 haproxy[1358]: 192.0.2.156:48994 [29/Apr/2022:10:58:17.406] sitio_http **sitio_web1/web01** 0/0/0/2/2 200 320 - - ---- 1/1/0/0/0 0/0 "GET / HTTP/1.1"

Si los dos servidores del cluster caen, al estar definido web05 como servidor de backup atenderá todas las peticiones a www.web1.example. En cuanto uno de los servidores del cluster vuelva a estar operativo, el servidor de backup deja de atender solicitudes que serán enviadas al servidor activo.

Apr 29 11:00:30 haproxy01 haproxy[1358]: **Server sitio_web1/web01 is DOWN**, reason: Layer4 timeout, check duration: 2001ms. 1 active and 1 backup servers left. 0 sessions active, 0 requeued, 0 remaining in queue.

Apr 29 11:00:30 haproxy01 haproxy[1358]: [WARNING] (1358) : **Server sitio_web1/web01 is DOWN**, reason: Layer4 timeout, check duration: 2001ms. 1 active and 1 backup servers left. 0 sessions active, 0 requeued, 0 remaining in queue.

Apr 29 11:00:30 haproxy01 haproxy[1358]: **Server sitio_web1/web01 is DOWN**, reason: Layer4 timeout, check duration: 2001ms. 1 active and 1 backup servers left. 0 sessions active, 0 requeued, 0 remaining in queue.

Apr 29 11:00:34 haproxy01 haproxy[1358]: **Server sitio_web1/web02 is DOWN**, reason: Layer4 timeout, check duration: 2001ms. 0 active and 1 backup servers left. Running on backup. 0 sessions active, 0 requeued, 0 remaining in queue.

Apr 29 11:00:34 haproxy01 haproxy[1358]: [WARNING] (1358) : **Server sitio_web1/web02 is DOWN**, reason: Layer4 timeout, check duration: 2001ms. 0 active and 1 backup servers left. Running on backup. 0 sessions active, 0 requeued, 0 remaining in queue.

Apr 29 11:00:34 haproxy01 haproxy[1358]: **Server sitio_web1/web02 is DOWN**, reason: Layer4 timeout, check duration: 2001ms. 0 active and 1 backup servers left. Running on backup. 0 sessions active, 0 requeued, 0 remaining in queue.

Apr 29 11:00:51 haproxy01 haproxy[1358]: 192.0.2.156:48996 [29/Apr/2022:11:00:51.119] sitio_http **sitio_web1/web05** 0/0/0/2/2 200 320 - - ---- 1/1/0/0/0 0/0 "GET / HTTP/1.1"

Apr 29 11:00:53 haproxy01 haproxy[1358]: 192.0.2.156:48998 [29/Apr/2022:11:00:53.247] sitio_http **sitio_web1/web05** 0/0/0/0/0 200 320 - - ---- 1/1/0/0/0 0/0 "GET / HTTP/1.1"

Apr 29 11:00:54 haproxy01 haproxy[1358]: 192.0.2.156:49000 [29/Apr/2022:11:00:54.752] sitio_http **sitio_web1/web05** 0/0/0/0/0 200 320 - - ---- 1/1/0/0/0 0/0 "GET / HTTP/1.1"

Apr 29 11:00:55 haproxy01 haproxy[1358]: 192.0.2.156:49002 [29/Apr/2022:11:00:55.519] sitio_http **sitio_web1/web05** 0/0/0/0/0 200 320 - - ---- 1/1/0/0/0 0/0 "GET / HTTP/1.1"