PROYECTO INTEGRADO

CLUSTER DE ALTA DISPONIBILIDAD CON HAPROXY Y KEEPALIVED

Índice de contenido

Introducción

Cada día es más importante y necesario montar nuestros servicios fundamentales en Alta Disponibilidad de manera que si algo falla por causas ajenas, éste siga ofreciendo su función.

Cuando hablamos de **Alta Disponibilidad**, enumeramos servidores que intentan ofrecer servicios sin que éstos dejen de ofrecer su función. Todo el servicio que no se pueda ofrecer se le llama "Tiempo de Inactividad".

La **disponibilidad de un servicio** se suele representar como un porcentaje tiempo/año que lleva prestando servicio ininterrumpidamente.

Normalmente, éste tiempo se expresa según el número de nueves. Lo más común es:

- **99,9%** (tres nueves) => 8,76 horas/año inactivo.
- **99,99 %** (cuatro nueves) => 52,6 minutos/año inactivo.
- **99,999%** (cinco nueves) => 5,26 minutos/año inactivo.

En este **Proyecto Integrado** se redactan los pasos necesarios para montar un Cluster de Alta Disponibilidad utilizando software como **HAProxy y Keepalived**, los cuales consisten en herramientas para montar un proxy inverso, por el cual, y a través de él conectarnos a los servidores web (backend).

Implementa soporte de un solo proceso que mantiene un gran número de conexiones simultaneas y a velocidades muy altas.

He elegido éste tema para mi proyecto ya que ha incitado en mi un **interés creciente** desde el conocimiento proporcionado en clase. Investigando, conocí varias formas de montar los servicios en Alta Disponibilidad, por lo que me pareció importante seguir avanzando en la materia.

Tras varios estudios, elegí montarlo con **LVS (Linux Virtual Servers)**. Éste dispone de mucha cantidad de software para responder a éste fin.

He probado varios como:

- Piranha
- UltraMonkey
- HeartBeat
- Keepalived

Tras familiarizarme con parte de éstos, seleccioné **Keepalived**. El resto de software hay que compilarlo para cada Kernel y esto supone un problema; cada vez que haya una actualización, cada vez que salga un bug en el código, etc. Por lo que, habría que recompilar el código.

Keepalived, al estar compilado y empaquetado en los repositorios de Debian hacen más amena y a la vez menos problemática la instalación de dicho software.

Topología de red

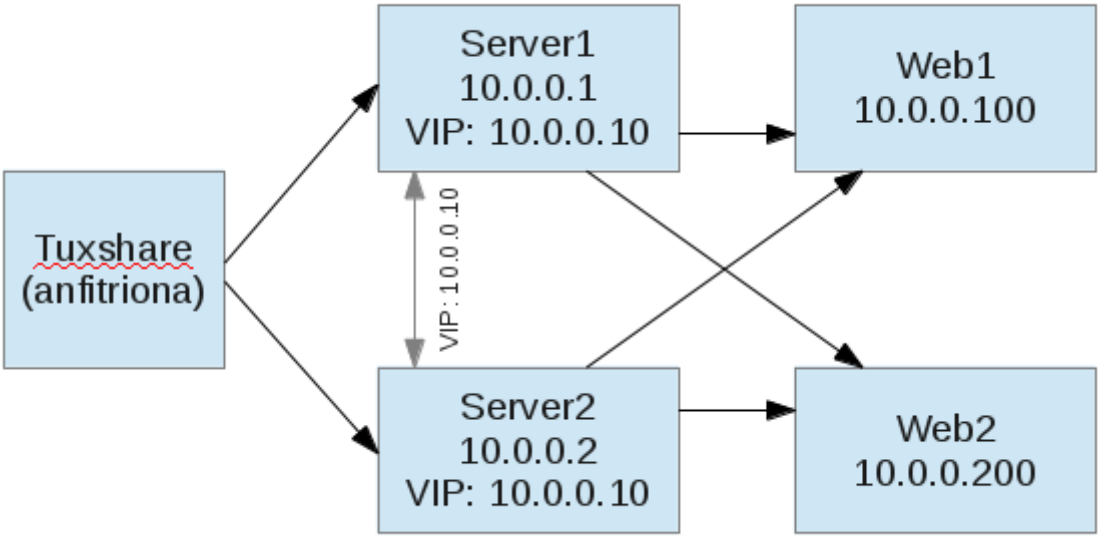
En total, he utilizado cuatro máquinas para mi proyecto. Dos de ellas son las encargadas de ejercer la Alta Disponibilidad y el Balanceo de Carga por Round Robin. Las otras dos son servidores web/backend cuyo contenido está replicado entre las dos máquinas, que de una forma menos directa también aportan Alta Disponibilidad a las web.

Los dos servidores principales comparten una ip Virtual (VIP). Ésta VIP es por la que accederemos a los servicios web. Cada uno de los backend tienen una ip propia y mediante el software utilizado accederemos a uno u otro dependiendo de la carga que estén soportando.

Máquinas utilizadas:

HOSTNAME	**IP**	**VIP**	**FUNCIÓN**
Server1	10.0.0.1	10.0.0.10	HAProxy – Keepalived
Server2	10.0.0.2	10.0.0.10	HAProxy – Keepalived
Web1	10.0.0.100		Apache
Web2	10.0.0.200		Apache

Las conexiones de las máquinas quedarían así:



Instalación de máquinas

Las máquinas utilizadas están virtualizadas con **KVM y libvirt**. Éstas están tras otra red usando ips del rango 10.0.0.0/24 saliendo hacia mi máquina con NAT.

Su puerta de enlace es la 10.0.0.254, que es la que utiliza mi interfaz virtual virbr1 por la que me conecto a dicha red.

Todas las máquinas que se utilizan son instaladas con Debian Squeeze 2.6.32-5-amd64. Sus **características Hardware** principales son:

- **CPU**: 1 core
- **Memoria RAM**: 256 Mb
- Dos **discos duros** SATA. 8 Gb
- **Tarjeta de red** Virtual usando NAT. Modelo Virtio.

En la **instalación** cabe recordar que he configurado los dos discos como RAID 1, creando así una copia exacta de los dos discos por si uno falla, el otro pueda seguir dando servicio. Éste tipo de configuración también resulta útil cuando el rendimiento en la lectura es más importante que su

capacidad de almacenamiento.

Las máquinas server1 y server2 comparten una **IP Virtual (VIP) [10.0.0.10]**. La configuración para tener ambas ésta ip es la siguiente:

Ésta configuración se hace en los servidores principales [server1 y server2].

```
auto eth0:1
iface eth0:1 inet static
  address 10.0.0.10
  gateway 10.0.0.254
  netmask 255.255.255.0
  network 10.0.0.0
  broadcast 10.0.0.255
```

Software a utilizar

El software que voy a utilizar son HAproxy como **balanceador de carga multiprotocolo** y Keepalived para **manejar el cluster** con una sola herramienta. Los backend al ser servidores web usan Apache.

- **HAproxy** versión 1.4.8-1, actualmente estable.
- **Keepalived** versión 1:1.1.20-1+squeeze1
- **Apache2** versión 2.2.16-6+squeeze8

Instalación y configuración de HAProx

Haproxy es un **proxy inverso** TCP/HTTP adecuado para entornos con Alta Disponibilidad. Cuenta con conexión persistente de cookies y balanceador de carga.

Haproxy se instala desde los repositorios oficiales de Debian.

```
# aptitude install haproxy
```

Ésta instalación se hace en los dos servidores principales [server1 y server2].

Para su configuración tenemos los ficheros en */etc/haproxy*. Solo tiene un fichero donde se configura todo, y un directorio donde se guardan los diferentes errores que se pueden producir; 400,

403, 408,500... y cada uno de ellos con unos parámetros donde se describe el error y unas pautas html que utilizamos para mostrar el error por un navegador.

El fichero de configuración de haproxy (***/etc/haproxy/haproxy.cfg***) tiene ésta estructura.

```
root@server1:~# cat /etc/haproxy/haproxy.cfg
global
        log 127.0.0.1   local0
        log 127.0.0.1   local1 notice
        #log loghost    local0 info
        maxconn 4096
        #debug
        #quiet
        user haproxy
        group haproxy
        daemon

defaults
        log     global
        mode    http
        option  httplog
        option  dontlognull
        retries 3
        option redispatch
        maxconn 2000
        contimeout      5000
        clitimeout      50000
        srvtimeout      50000

listen proyectointegrado *:80
       mode http
       stats enable
       stats auth user:pass
       balance roundrobin
       cookie SERVERID insert
       option http-server-close
       option forwardfor
       option httpchk HEAD /check.txt HTTP/1.0
       # servidor web 1
       server webA 10.0.0.100:80 cookie A check
       # servidor web 2
       server webB 10.0.0.200:80 cookie B check
root@server1:~#
```

El fichero de configuración del server2 es exactamente igual.

Otros ficheros de configuración son los del directorio *errors*, que contienen los errores http que se pueden dar al fallar la aplicación. Entre ellos están los errores 400 (cuando el servidor no da respuesta), 403 (no se encuentra el servidor), 500 (Internal Server Error), etc.

La estructura de éstos ficheros es la siguiente:

```
root@server1:/etc/haproxy/errors# cat 500.http
HTTP/1.0 500 Server Error
Cache-Control: no-cache
Connection: close
Content-Type: text/html

<html><body><h1>500 Server Error</h1>
An internal server error occured.
</body></html>

root@server1:/etc/haproxy/errors#
```

Otro fichero que hay que modificar es el ***/etc/default/haproxy***. Éste hay que modificarlo en los dos servidores principales [server1 y server2]. La configuración quedaría así, solo cambiar el **ENABLED** a 1.

```
root@server1:~# cat /etc/default/haproxy
# Set ENABLED to 1 if you want the init script to start haproxy.
ENABLED=1
# Add extra flags here.
#EXTRAOPTS="-de -m 16"
root@server1:~#
```

Instalación y configuración de Keepalived

Keepalived se utiliza para monitorizar servidores dentro de un entorno de cluster **LVS** (Linux Virtual Server). Se puede configurar para eliminar servidores de la cola del cluster si éste deja de responder. También puede configurarse para en el momento que keepalived detecte una máquina con un funcionamiento erróneo éste envíe un correo electrónico al administrador advirtiendo que se ha salido un servidor del cluster.

Además, implementa mediante protocolo **VRRPv2** un módulo para recoger información adicional del cluster.

Su instalación se hace desde los repositorios oficiales de Debian.

```
# aptitude install keepalived
```

Ésta instalación se hace en los dos servidores principales [server1 y server2].

Su configuración varía en ambos. Existen dos **modos de trabajo** en los servidores:

- ***MASTER***: Mientras todo esté correcto éste es el que ofrece servicio.
- ***BACKUP***: Si el servidor maestro cae, éste se convierte en maestro hasta que el MASTER esté recuperado totalmente. Es en éste momento cuando se convierte de nuevo en BACKUP.

En mi configuración he elegido que solo haya un servidor MASTER y uno BACKUP por el simple hecho de que solo van a haber dos, así podremos comprobar cómo al caerse el maestro sigue dando servicio el backup.

La configuración del MASTER [10.0.0.1] es la siguiente (***/etc/keepalived/keepalived.conf***):

```
root@server1:/etc/keepalived# cat keepalived.conf
vrrp_script chk_haproxy {
        script "killall -0 haproxy"
        interval 2
        weight 2
}

vrrp_instance VI_1 {
        interface eth0
        state MASTER
        virtual_router_id 51
        priority 101            # 101 en master, 100 en backup
        virtual_ipaddress {
            10.0.0.10           # IP virtual
        }
        track_script {
            chk_haproxy
        }
}
root@server1:/etc/keepalived#
```

Y la configuración del BACKUP [10.0.0.2] es la siguiente (***/etc/keepalived/keepalived.conf***):

Como se observa en las imágenes, la configuración de los dos estados es muy similar. Lo único que cambia en ésta configuración son el **estado *[MASTER/BACKUP]*** y la **prioridad *[101 en MASTER / 100 en BACKUP]***.

```
root@server2:/etc/keepalived# cat keepalived.conf
vrrp_script chk_haproxy {
        script "killall -0 haproxy"
        interval 2
        weight 2
}

vrrp_instance VI_1 {
        interface eth0
        state BACKUP
        virtual_router_id 51
        priority 100            # 101 en master, 100 en backup
        virtual_ipaddress {
                10.0.0.10       # IP virtual
        }
        track_script {
                chk_haproxy
        }
}
root@server2:/etc/keepalived#
```

Se puede observar que el MASTER tiene la prioridad 101. Uno más alto que el BACKUP.

También se puede comprobar que ambos comparten la IP Virtual.

Configuración de sysctl

Tenemos que activar el **bit de forwarding** y el **non local bind**.

Ésto puede hacerse de dos maneras diferentes: modificando el fichero en sí o añadiendolo al fichero sysctl y aplicando los cambios. Acostumbrados a hacerlo siempre desde los propios ficheros he decidido hacerlo con ***sysctl***.

Para cambiar ésta configuración lo hacemos con un echo de la siguiente forma:

```
# echo "net.ipv4.ip_forward = 1" >> /etc/sysctl.conf

# echo "net.ipv4.ip_nonlocal_bind = 1" >> /etc/sysctl.conf

# sysctl -p
```

(que muestra lo siguiente)

```
root@server1:~# sysctl -p
net.ipv4.ip_forward = 1
net.ipv4.ip_nonlocal_bind = 1
root@server1:~#
```

Ésto hay que hacerlo en los dos servidores principales [server1 y server2].

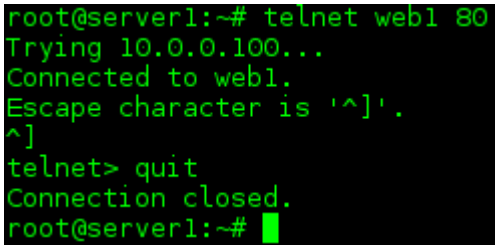
Configuración de servidores web

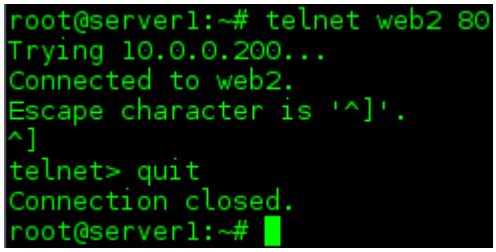
Los dos **servidores web** que he montado, para no hacerlos muy pesados ya que tienen poca RAM, lo único que tienen es un Apache corriendo en el puerto 80.

Para ello instalamos el servidor web **apache2** desde los repositorios:

```
# aptitude install apache2
```

Primero hay que asegurarse que responden al puerto 80. Ésto podemos saberlo haciendo un telnet a dicho puerto:





Para que los servidores web sirvan mucho más rápido las páginas he instalado **varnish**. Éste consiste en una simple **caché** que acelera el proceso de búsqueda de la web.

Para la instalación hay que hacerlo desde respositorios:

```
# aptitude install varnish
```

No hace falta configurarlo puesto que viene configurado para que tenga un correcto funcionamiento.

A continuación, he montado una simple página web estática con código ***html y css*** en la que he

puesto un enlace para comprobar qué servidor web nos está respondiendo. Esto es una simple forma de comprobar cuando un servidor web cae el otro asumen el mando.

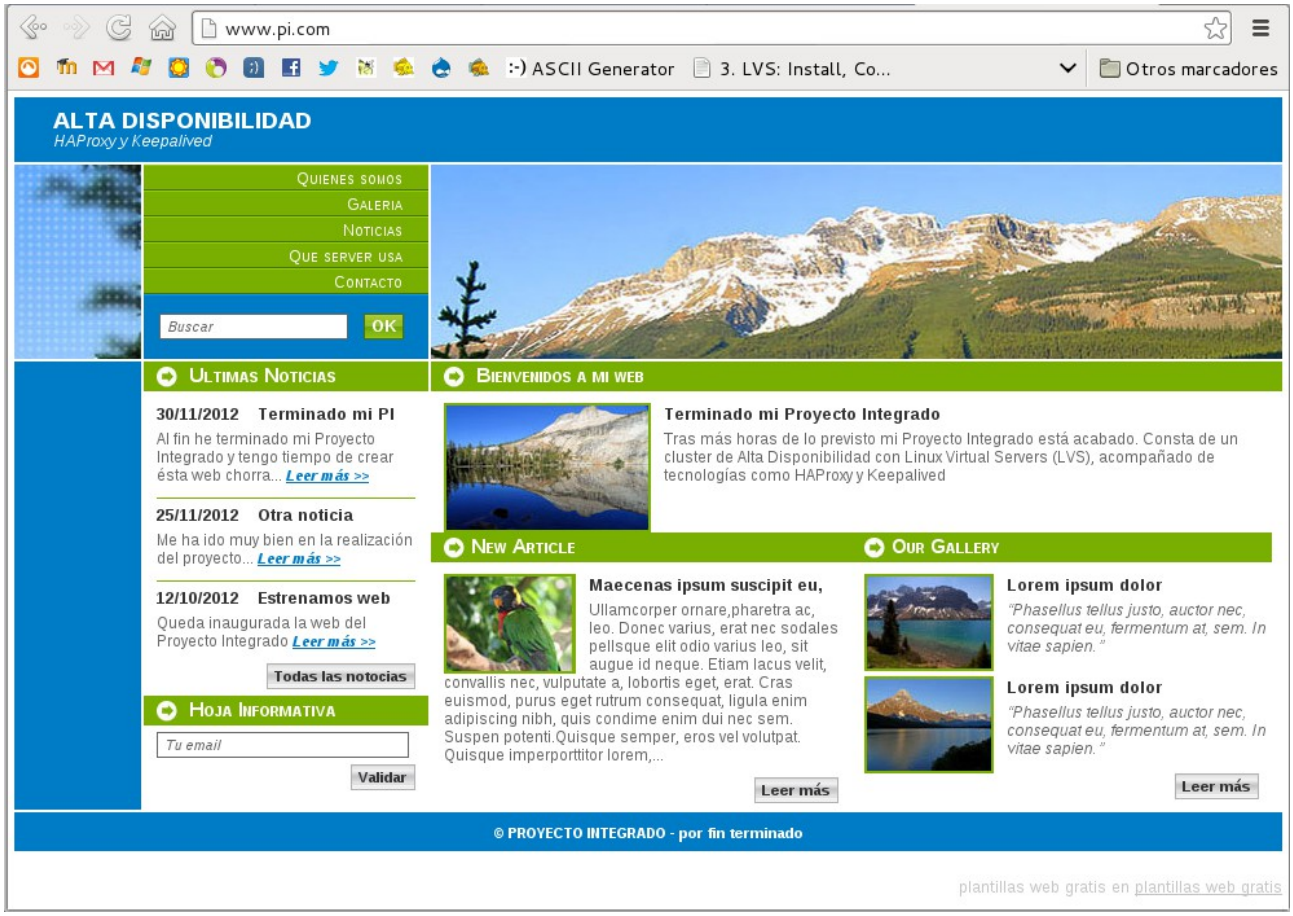
Otra cosa más o menos importante es crear un fichero *check.txt* en los directorios web donde incluya el texto “check ok”. Esto sirve porque cuando haproxy se conecta con los servidores web pide confirmación, y mira si existe este archivo. Si no existe lo escribe en el log, y así evitamos que el log se llene de cosas inservibles.

Para acceder a la web he añadido una entrada en el /etc/hosts de mi máquina anfitriona.

```
user@tuxshare:~$ cat /etc/hosts
127.0.0.1       localhost
127.0.1.1       tuxshare
10.0.0.1        server1
10.0.0.2        server2
10.0.0.100      web1
10.0.0.200      web2
10.0.0.10       www.pi.com      # IP Virtual en Servers

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
user@tuxshare:~$
```

La página web está replicada en los dos servidores. Ésto podría hacerse configurando por ejemplo ***rsync***, pero al ser una web estática y para no forzar más las máquinas lo que he hecho ha sido colocar el mismo contenido en los dos servidores.



Inicio de servicios

Cuando ya tengamos todo configurado, procedemos a ***iniciar servicios***.

En server1 y en server2:

```
# /etc/init.d/haproxy start
```

```
# /etc/init.d/keepalived start
```

*"Hay que iniciar los dos servicios **primero en el MASTER** y segundo en el BACKUP. Si se hace de forma alternativa puede que de fallos al no poder sincronizar archivos."*

Comprobaciones

Antes que nada tenemos que comprobar que haproxy y keepalived están funcionando. Para ello desde server1 y server2 ejecutamos éste comando:

```
# ip addr sh eth0
```

Se puede comprobar en cada máquina al hacer éste *comando* cuál tiene la ip virtual, y cual es el MASTER y el BACKUP.

[MASTER]

[BACKUP]

En la siguiente imagen, haciendo una simulación como si hubiera caido el master, podemos comprobar como el BACKUP comienza a ofrecer el servicio.

Para comprobar si una máquina ha caído y ver el estado [MASTER/BACKUP] podemos mirar el **/var/log/messages**, donde se aprecia los mensajes del tipo “Transition to MASTER STATE”, etc...

```
root@server1:~# tailf /var/log/messages
Dec  2 11:04:50 server1 Keepalived_healthcheckers: Registering Kernel netlink reflector
Dec  2 11:04:50 server1 Keepalived_healthcheckers: Registering Kernel netlink command channel
Dec  2 11:04:50 server1 Keepalived_healthcheckers: Opening file '/etc/keepalived/keepalived.conf'.
Dec  2 11:04:50 server1 Keepalived_vrrp: Configuration is using : 60972 Bytes
Dec  2 11:04:50 server1 Keepalived_vrrp: Using LinkWatch kernel netlink reflector...
Dec  2 11:04:50 server1 Keepalived_healthcheckers: Configuration is using : 4225 Bytes
Dec  2 11:04:50 server1 Keepalived_healthcheckers: Using LinkWatch kernel netlink reflector...
Dec  2 11:04:50 server1 Keepalived_vrrp: VRRP_Script(chk_haproxy) succeeded
Dec  2 11:04:51 server1 Keepalived_vrrp: VRRP_Instance(VI_1) Transition to MASTER STATE
Dec  2 11:04:52 server1 Keepalived_vrrp: VRRP_Instance(VI_1) Entering MASTER STATE
```

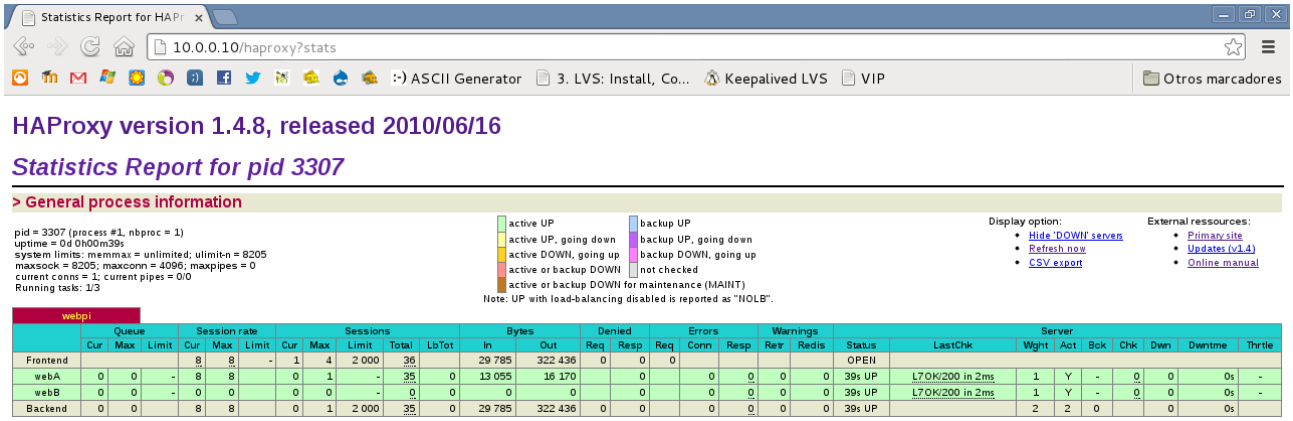
También podemos ver el **/var/log/messages de server2**, y comprobar que está funcionando como BACKUP.

```
root@server2:~# tailf /var/log/messages
Dec  2 11:04:58 server2 Keepalived_vrrp: Registering Kernel netlink reflector
Dec  2 11:04:58 server2 Keepalived_vrrp: Registering Kernel netlink command channel
Dec  2 11:04:58 server2 Keepalived_vrrp: Registering gratutious ARP shared channel
Dec  2 11:04:58 server2 Keepalived_vrrp: Opening file '/etc/keepalived/keepalived.conf'.
Dec  2 11:04:58 server2 Keepalived_vrrp: Configuration is using : 60972 Bytes
Dec  2 11:04:58 server2 Keepalived_vrrp: Using LinkWatch kernel netlink reflector...
Dec  2 11:04:58 server2 Keepalived_vrrp: VRRP_Instance(VI_1) Entering BACKUP STATE
Dec  2 11:04:58 server2 Keepalived_healthcheckers: Configuration is using : 4225 Bytes
Dec  2 11:04:58 server2 Keepalived_healthcheckers: Using LinkWatch kernel netlink reflector...
Dec  2 11:04:58 server2 Keepalived_vrrp: VRRP_Script(chk_haproxy) succeeded
```

Estadísticas con HAProxy

Una de las funciones que contiene HAProxy es poder contemplar las estadísticas mediante una consola web.

Para ello accedemos desde el navegador a la dirección http://10.0.0.10/haproxy?stats donde observamos lo siguiente:



Si hacemos otra simulación apagando uno de los servidores web, éste se ve reflejado en la interfaz web. Primero con el estado "***active UP, going down***" que significa que se ha caido pero aún no se ha sacado del cluster, y luego "**active or backup DOWN**" que es cuando lo saca del cluster.

Referencias web

Páginas Oficiales:

- **LVS**: http://www.linuxvirtualserver.org/
- **Keepalived**: http://www.keepalived.org/
- **HAProxy**: http://haproxy.1wt.eu/

HOWTO y manuales:

- http://www.linuxvirtualserver.org/docs/ha/keepalived.html
- http://www.danieldemichele.com.ar/alta-disponibilidad-load-balancer-con-haproxy/
- http://www.howtoforge.com/setting-up-haproxy-keepalived-on-debian-lenny-p2

Información sobre varias definiciones en www.wikipedia.org.