→ Convenciones:

```
# En todos los nodos como 'sudo su'.
[root@srv1 ~]# Solo en servidor 'srv1'→ como 'sudo su'.
[root@srv2 ~]# Solo en servidor 'srv2'→ como 'sudo su'.
```

361.2 Load Balanced Clusters (weight: 8)

Weight	8
Description	Candidates should know how to install, configure, maintain and troubleshoot LVS. This includes the configuration and use of keepalived and ldirectord. Candidates should further be able to install, configure, maintain and troubleshoot HAProxy.

Key Knowledge Areas:

- Understand the concepts of LVS / IPVS
- Understand the basics of VRRP
- Configure keepalived
- Configure ldirectord
- Configure backend server networking
- Understand HAProxy
- Configure HAProxy

Partial list of the used files, terms and utilities:

- ipvsadm
- syncd
- LVS Forwarding (NAT, Direct Routing, Tunneling, Local Node)
- connection scheduling algorithms
- keepalived configuration file
- ldirectord configuration file
- genhash
- HAProxy configuration file
- load balancing algorithms
- ACLs

→ Introducción a LVS

Creado por el Dr. Zhang Wenchong de China en 1998, tiene tres modos de trabajo de servicio de equilibrio de carga.

→ **Traducción de direcciones (NAT)**: el planificador de carga sirve como la puerta de entrada y salida de todos los nodos de servicio.

→ **Enrutamiento de enlace directo (DR)**: similar al túnel IP, pero todos los nodos están en la misma LAN.

→ **Túnel de IP (TUN)**: el planificador de carga sirve como entrada al nodo de servicio, y la salida es la IP pública independiente.

Sondeo (rr): envíe el acceso recibido al nodo de servicio circularmente.

Robustez ponderada (wrr): ajusta automáticamente el peso de acuerdo con la carga consultada automáticamente por el servidor, para que el servidor potente pueda procesar la solicitud de manera múltiple.

Menos conexiones (lc): según el estado de la solicitud, la solicitud recibida se asigna al nodo con el menor servicio de procesamiento.

Menos conexiones ponderadas (wlc): los nodos con mayor peso llevan una mayor proporción de solicitudes de servicio.

Instalar servicios LVS

```
# yum -y install ipvsadm      # Instalar herramientas de administración
# modprobe ip_vs              # Módulo de kernel de carga
```

Agregar un nodo virtual

```
ipvsadm -A -t 172.20.10.2:80 -s rr
```

-A agregar un servidor virtual

-t especifica VIP

-s especifica el algoritmo de planificación de equilibrio de carga

Agregar nodo de servicio

```
ipvsadm -a -t 172.20.10.2:80 -r 172.20.10.3:80 -m -w 1
```

-a agregar un servidor real

-t especifica VIP

-r especifica RIP

-m significa usar NAT

-g significa DR

-i significa TUN

-w peso

-p mantener la conexión

Consulta LVS

ipvsadm -ln

ipvsadm -lN

Eliminar nodo/servicio del servidor

ipvsadm -d -r 172.20.10.5:80 -t 172.20.20.2:80

ipvsadm -D 172.20.10.2:80

ipvsadm -C > /etc/sysconfig/ipvsadm-config

-d eliminar un servidor real

-t especifica VIP

-r especifica RIP

-D eliminar todo el servicio VIP

Guardar configuración de LVS

ipvsadm -S -n > /etc/sysconfig/ipvsadm-config

→ Modelo Propuesto para NAT

	director-01	VIP (alias temporal)
enp1s0	192.168.1.11/24	192.168.1.50/32
enp10s0	192.168.10.11/24	192.168.10.50/32
		gw4 192.168.10.5

	real-01	GW	real-02
enp1s0	192.168.1.101/24	gw4 192.168.1.50	192.168.1.102/24
enp10s0	192.168.10.101/24	gw4192.168.10.5	192.168.10.102/24

→ **Configuración Ejemplo**

 # vim /etc/hosts

Directores.

192.168.10.11 director-01.cadilinea.lan director-01

192.168.10.12 director-02.cadilinea.lan director-02

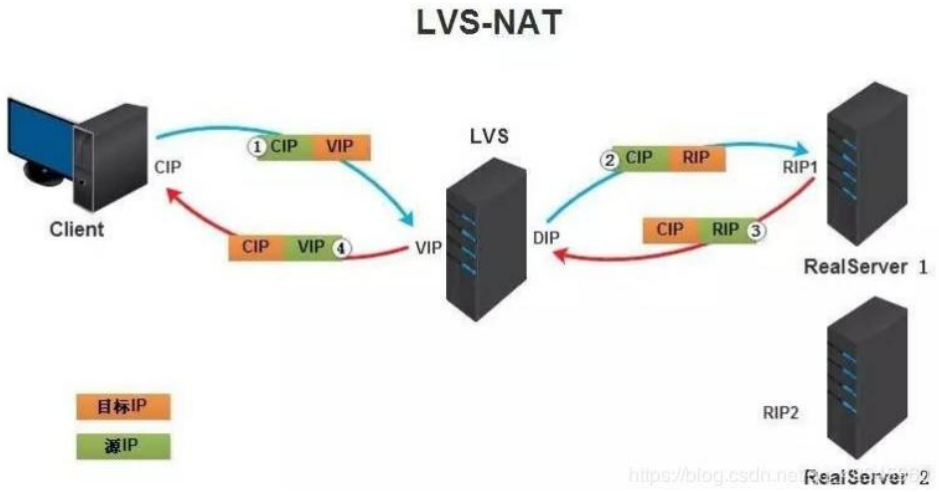
Real Servers.

192.168.10.101 real-01.cadilinea.lan real-01

192.168.10.102 real-02.cadilinea.lan real-02

→ **Sincronizar servidores → chronyd.service**

1 => LVS NAT (Masquerade) → Un solo director → director-01



El cliente envía la solicitud al equilibrador de carga frontal. La dirección de origen del mensaje de

solicitud es CIP (IP del cliente), que se denomina colectivamente CIP más adelante, y la dirección de destino es VIP (la dirección de front-end del equilibrador de carga, que se denomina colectivamente VIP más adelante).

Después de recibir el mensaje, el equilibrador de carga encuentra que la solicitud es la dirección existente en la regla, luego cambia la dirección de destino del mensaje de solicitud del cliente a la dirección RIP del servidor de fondo y envía el mensaje de acuerdo con el algoritmo.

Después de enviar el mensaje a Real Server, dado que la dirección de destino del mensaje es la misma, responderá a la solicitud y devolverá el mensaje de respuesta a LVS.

Luego, lvs cambia la dirección de origen de este mensaje a la máquina local y la envía al cliente.

Nota:

En modo NAT, la puerta de enlace del servidor real debe apuntar a LVS; de lo contrario, el mensaje no se puede entregar al cliente.

Caracteristicas:

1. La tecnología NAT necesita reescribir la dirección del mensaje de solicitud y el mensaje de respuesta a través de LB. Por lo tanto, cuando las visitas al sitio web son relativamente grandes, el planificador de equilibrio de carga LB tiene un cuello de botella relativamente grande y generalmente requiere un máximo de 10-20 unidades Nodo

2. Solo necesita configurar una dirección IP pública en el LB.

3. La dirección de la puerta de enlace de cada servidor interno del servidor real debe ser la dirección de la intranet del planificador LB.

4. El modo NAT admite la conversión de direcciones IP y puertos. Es decir, el puerto solicitado por el usuario y el puerto del servidor real pueden ser diferentes.

Ventajas:

Los servidores físicos en el clúster pueden usar cualquier sistema operativo que admita TCP / IP, y solo el equilibrador de carga necesita una dirección IP legal.

Desventajas:

La escalabilidad es limitada. Cuando el nodo del servidor (servidor de PC común) crece demasiado, el equilibrador de carga se convertirá en el cuello de botella de todo el sistema, porque

todo el flujo de paquetes de solicitud y paquetes de respuesta pasan a través del equilibrador de carga. Cuando hay demasiados nodos de servidor, una gran cantidad de paquetes de datos convergen en el equilibrador de carga, ¡y la velocidad será más lenta!

→ Configuración del director → director-01

director-01 ~ # ssh-keygen

director-01 ~ # ssh-copy-id director-01

director-01 ~ # ssh-copy-id director-02

director-01 ~ # ssh-copy-id real-01

director-01 ~ # ssh-copy-id real-02

→ Instalamos paquetería necesaria → director-01

director-01 ~ # dnf install ipvsadm

→ Activamos forwarding

director-01 ~ # echo 'net.ipv4.ip_forward = 1' | sudo tee -a /etc/sysctl.conf

net.ipv4.ip_forward = 1

director-01 ~ # sysctl -p

net.ipv4.ip_forward = 1

director-01 ~ # mkdir /etc/sysconfig/ipvsadm

director-01 ~ # systemctl enable --now ipvsadm

director-01 ~ # systemctl status ipvsadm

● ipvsadm.service - Initialise the Linux Virtual Server

Loaded: loaded (/usr/lib/systemd/system/ipvsadm.service; enabled; vendor preset: disabled)

Active: active (exited) since Mon 2021-06-07 10:43:52 CEST; 30s ago

Process: 1670 ExecStart=/bin/bash -c exec /sbin/ipvsadm-restore < /etc/sysconfig/ipvsadm (code=exited, status=0/SUCCESS)

Main PID: 1670 (code=exited, status=0/SUCCESS)

jun 07 10:43:52 director-01.cadilinea.lan systemd[1]: Starting Initialise the Linux Virtual Server...

jun 07 10:43:52 director-01.cadilinea.lan systemd[1]: Started Initialise the Linux Virtual Server.

director-01 ~ # lsmod |grep ip_vs

ip_vs 176128 0

nf_conntrack 163840 3 nf_nat,nft_ct,ip_vs

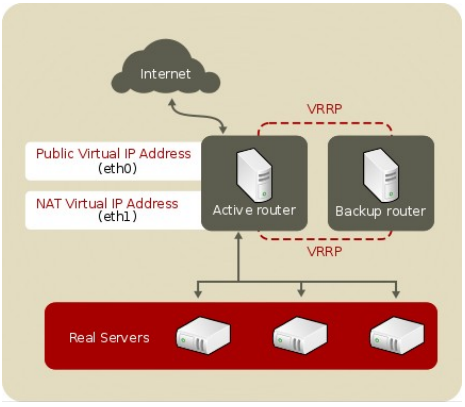
nf_defrag_ipv6 24576 2 nf_conntrack,ip_vs

director-01 ~ # ipvsadm

IP Virtual Server version 1.2.1 (size=4096)

Prot LocalAddress:Port Scheduler Flags

-> RemoteAddress:Port Forward Weight ActiveConn InActConn



 # ipvsadm -A -t (ServiceIP:Port) -s (Distribution method)

-A Add a virtual service. A service address is uniquely defined by a triplet: IP address, port number, and protocol.

Alternatively, a virtual service may be defined by a firewall-mark.

-t, --tcp-service service-address

Use TCP service. The service-address is of the form host[:port]. Host may be one of a plain IP address or a hostname.

Port may be either a plain port number or the service name of port. The Port may be omitted, in which case zero will be used. A Port of zero is only valid if the service is persistent as the -p|--persistent option, in which case it is a wild-card port, that is connections will be accepted to any port.

-s, --scheduler scheduling-method

Métodos de scheduler.

- **rr** – Round Robin, distribuye trabajos por igual entre los servidores disponibles.
- **wrr** – Weighted Round Robin, asigna trabajos a servidores reales proporcionalmente a la carga real de los servidores. Los servidores con mayor carga reciben nuevos trabajos primero y obtienen más trabajos que los servidores con menos carga. Los servidores con cargas similares obtienen una distribución equitativa de los nuevos trabajos.
- **lc** – Least Connection, asigna más trabajo a servidores con menos trabajo activo.
- **wlc** – Weighted Least Connection asigna más trabajo a los servidores con menos trabajo y en relación con la carga real de los servidores (Ci/Wi). Por defecto.
- **lblc** – Local-Based Least-Connection asigna trabajos destinados a la misma dirección IP del servidor, si el servidor no está sobrecargado y disponible, de lo contrario asigna trabajos a servidores con menos trabajos y lo guarda para futuras asignaciones.
- **lblcr** – La conexión con replicación asigna trabajos destinados a la dirección IP al nodo de conexión mínima en el servidor configurado para la dirección IP. Si todo el nodo en el conjunto de servidores está sobrecargado, recoge un nodo con menos trabajos en el clúster y lo agrega al conjunto de servidores para el destino. Si el conjunto de servidores no se ha modificado durante el tiempo especificado, el nodo más cargado se elimina del conjunto de servidores, para evitar un alto grado de replicación.
- **dh** – Destino Hashing asigna trabajos a los servidores mediante la búsqueda de una tabla hash, asignada estáticamente por las direcciones IP de destino.
- **sh** – Source Hashing asigna trabajos a los servidores mediante la búsqueda de una tabla hash, asignada estáticamente por las direcciones IP de origen.
- **sed** – Asigna un trabajo entrante al servidor con la mínima espera disponible.
- **nq** – Never Queue asigna el trabajo entrante a un servidor inactivo, en vez de esperar al más rapido, si todos los servidores están ocupados, adopta la política de Retardo más corto para asignar el trabajo.

→ Asignamos direcciones VIP (alias temporal interna/externa) → director-01

(No es necesario para NAT crear un alias VIP de red interna. Lo hacemos como práctica para preparar posteriormente keepalived y el protocolo VRRP).

director-01 ~ # ip address add 192.168.1.50/32 dev enp1s0:0 # alias VIP Red Interna.

director-01 ~ # ip a show enp1s0

2: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000

link/ether 52:54:00:eb:ef:dc brd ff:ff:ff:ff:ff:ff

inet 192.168.1.11/24 brd 192.168.1.255 scope global noprefixroute enp1s0

valid_lft forever preferred_lft forever

inet 192.168.1.50/32 scope global enp1s0

valid_lft forever preferred_lft forever

inet6 fe80::5054:ff:feeb:efdc/64 scope link noprefixroute

valid_lft forever preferred_lft forever

(No es necesario para NAT crear un alias VIP de red externa. Lo hacemos como práctica para preparar posteriormente keepalived y el protocolo VRRP)

director-01 ~ # ip address add 192.168.10.50/32 dev enp10s0:0 # alias VIP red Externa.

director-01 ~ # ip a show enp10s0

3: enp10s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000

link/ether 52:54:00:1a:1b:18 brd ff:ff:ff:ff:ff:ff

inet 192.168.10.11/24 brd 192.168.10.255 scope global noprefixroute enp10s0

valid_lft forever preferred_lft forever

inet 192.168.10.50/32 scope global enp10s0

valid_lft forever preferred_lft forever

inet6 fe80::2c2c:1a32:bf74:1b06/64 scope link dadfailed tentative noprefixroute

valid_lft forever preferred_lft forever

inet6 fe80::7f69:78a5:c43b:4344/64 scope link noprefixroute

valid_lft forever preferred_lft forever

→ Para la subred de los servidores reales indicamos el gateway temporal→ 192.168.1.50

(Como comentamos anteriormente no es necesario utilizar una VIP interna. Podriamos haber utilizado directamente y como gateway la IP del balanceador → **director-01 → 192.168.1.11**)

real-01 ~ # nmcli connection modify enp1s0 gw4 192.168.1.50

real-01 ~ # nmcli connection down enp1s0

real-01 ~ # nmcli connection up enp1s0

real-02 ~ # nmcli connection modify enp1s0 gw4 192.168.1.50

real-02 ~ # nmcli connection down enp1s0

real-02 ~ # nmcli connection up enp1s0

→ Asignamos la VIP al servicio para equilibrar la carga de los servidores reales.

director-01 ~ # ipvsadm -A -t 192.168.10.50:80 -s rr

director-01 ~ # ipvsadm -Ln

IP Virtual Server version 1.2.1 (size=4096)

Prot LocalAddress:Port Scheduler Flags

-> RemoteAddress:Port Forward Weight ActiveConn InActConn

TCP 192.168.10.50:80 rr

→ Configuramos los servicios a balancear

→ Desde → real-01 / real-02

 # dnf install httpd

 # systemctl enable --now httpd.service

 # firewall-cmd --permanent --add-service=http

 # firewall-cmd --reload

real-01 ~ # vim /var/www/html/index.html

Servidor Real 1

real-02 ~ # vim /var/www/html/index.html

Servidor Real 2

→ Añadimos los servidores reales con su servicio (masquerade).

director-01 ~ # ipvsadm -a -t 192.168.10.50:80 -r 192.168.1.101:80 -m -w 1

director-01 ~ # ipvsadm -a -t 192.168.10.50:80 -r 192.168.1.102:80 -m -w 1

director-01 ~ # ipvsadm -Ln

IP Virtual Server version 1.2.1 (size=4096)

Prot LocalAddress:Port Scheduler Flags

-> RemoteAddress:Port Forward Weight ActiveConn InActConn

TCP 192.168.10.50:80 rr

-> 192.168.1.101:80 Masq 1 0 0

-> 192.168.1.102:80 Masq 1 0 0

→ Comprobamos el balanceo

director-01 ~ # curl 192.168.10.50:80

Servidor Real 1

director-01 ~ # curl 192.168.10.50:80

Servidor Real 2

director-01 ~ # ipvsadm -Lnc

IPVS connection entries

pro expire state source virtual destination

TCP 01:56 TIME_WAIT 192.168.10.50:49542 192.168.10.50:80 **192.168.1.101:80**

TCP 00:41 SYN_RECV 192.168.10.50:49536 192.168.10.50:80 **192.168.1.102:80**

→ Guardamos la configuración y eliminamos posteriormente.

director-01 ~ # ipvsadm -S -n > /etc/sysconfig/ipvsadm/ipvsadm-NAT-OK

director-01 ~ # cat /etc/sysconfig/ipvsadm/ipvsadm-NAT-OK

-A -t 192.168.10.50:80 -s rr

-a -t 192.168.10.50:80 -r 192.168.1.101:80 -m -w 1

-a -t 192.168.10.50:80 -r 192.168.1.102:80 -m -w 1

director-01 ~ # ipvsadm -C

director-01 ~ # ipvsadm

IP Virtual Server version 1.2.1 (size=4096)

Prot LocalAddress:Port Scheduler Flags

-> RemoteAddress:Port Forward Weight ActiveConn InActConn

→ Creamos de nuevo pero ahora con persistencia → 5 segundos por ejemplo

director-01 ~ # ipvsadm -A -t 192.168.10.50:80 -s rr -p 5

director-01 ~ # ipvsadm -Ln

IP Virtual Server version 1.2.1 (size=4096)

Prot LocalAddress:Port Scheduler Flags

-> RemoteAddress:Port Forward Weight ActiveConn InActConn

TCP 192.168.10.50:80 rr persistent 5

director-01 ~ # ipvsadm -a -t 192.168.10.50:80 -r 192.168.1.101:80 -m -w 1

director-01 ~ # ipvsadm -a -t 192.168.10.50:80 -r 192.168.1.102:80 -m -w 1

→ Modificamos la persistencia a 10 segundos en tiempo de ejecución.

director-01 ~ # ipvsadm -E -t 192.168.10.50:80 -p 10

director-01 ~ # ipvsadm -Ln

IP Virtual Server version 1.2.1 (size=4096)

Prot LocalAddress:Port Scheduler Flags

-> RemoteAddress:Port Forward Weight ActiveConn InActConn

TCP 192.168.10.50:80 **wlc persistent 10 « Nos lo convierte en wlc**

-> 192.168.1.101:80 Masq 1 0 0

-> 192.168.1.102:80 Masq 1 0 0

→ Restauramos, comprobamos, y borramos definitivamente.

director-01 ~ # cat /etc/sysconfig/ipvsadm/ipvsadm-NAT-OK | ipvsadm -R

director-01 ~ # ipvsadm -Ln

IP Virtual Server version 1.2.1 (size=4096)

Prot LocalAddress:Port Scheduler Flags

-> RemoteAddress:Port Forward Weight ActiveConn InActConn

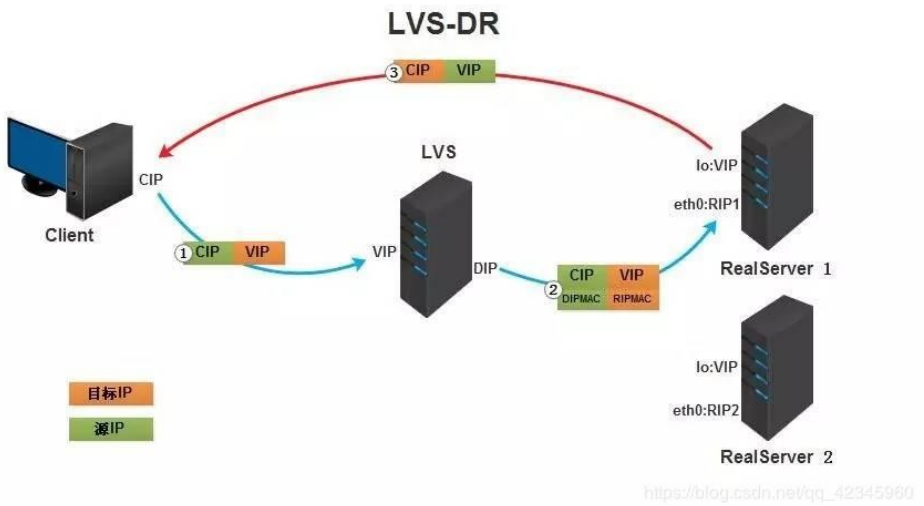
TCP 192.168.10.50:80 **rr**

-> 192.168.1.101:80 Masq 1 0 0

-> 192.168.1.102:80 Masq 1 0 0

director-01 ~ # ipvsadm -C

2 => LVS Direct Routing (Gateway) → Un solo director → director-01



El cliente envía la solicitud al equilibrador de carga frontal, la dirección de origen del mensaje de solicitud es CIP y la dirección de destino es VIP.

Después de recibir el mensaje, el equilibrador de carga encuentra que la solicitud es la dirección existente en la regla, luego cambia la dirección MAC de origen del mensaje de solicitud del cliente a su propia dirección MAC DIP y el MAC destino a RIP y envíe este paquete a RS.

RS encuentra que el MAC de destino en el mensaje de solicitud es él mismo y recibirá el mensaje secundario. Después de procesar el mensaje de solicitud, el mensaje de respuesta se enviará a la tarjeta de red a través de la interfaz **lo** y esta enviará directamente al cliente.

Nota:

Los VIP que necesitan configurar la interfaz **lo** no pueden responder a las solicitudes arp en la red local.

Resumen:

1. Reenvíe el paquete modificando la dirección MAC de destino del paquete en el planificador LB. Tenga en cuenta que la dirección de origen sigue siendo CIP, y la dirección de destino sigue siendo una dirección VIP.

2. El mensaje solicitado pasa por el planificador, y el mensaje procesado de respuesta RS no necesita pasar por el planificador LB, por lo que la eficiencia de uso es muy alta cuando el número de acceso concurrente es grande (en comparación con el modo NAT)

3. Debido a que el modo DR se reenvía a través del mecanismo de reescritura de direcciones MAC, todos los nodos RS y el planificador LB solo pueden estar en una LAN

4. El host RS debe vincular la dirección VIP a la interfaz **lo** (enmascarar 32 bits), y la supresión ARP debe configurarse.

5. La puerta de enlace predeterminada del nodo RS no necesita configurarse como LB, sino que se

configura directamente como la puerta de enlace de la ruta de nivel superior, lo que permite que RS salga directamente de la red.

6. Debido a que el planificador en modo DR solo reescribe la dirección MAC, el planificador LB no puede reescribir el puerto de destino, por lo que el servidor RS debe usar el mismo puerto que el VIP para proporcionar servicios.

7. Para servicios externos directos como WEB, la IP de RS es mejor usar IP pública. Para servicios externos, como bases de datos, lo mejor es utilizar la intranet IP.

Ventajas:

Al igual que TUN (Modo de túnel), el equilibrador de carga solo distribuye la solicitud, y el paquete de respuesta se devuelve al cliente a través de un método de enrutamiento separado. En comparación con VS-TUN, la implementación de VS-DR no requiere una estructura de **túnel, por** lo que la mayoría de los sistemas operativos se pueden usar como servidores físicos.

La eficiencia del modo DR es muy alta, pero la configuración es un poco más complicada, por lo que para las empresas que no tienen mucho tráfico, puede usar haproxy / nginx. Puede considerar haproxy / nginx para 1000-2000W PV por día o 10,000 solicitudes simultáneas.

Desventajas

Todos los nodos RS y el planificador LB solo pueden estar en una LAN.

→ Modelo Propuesto

→ Cliente externo

redhat00 ~ # ip a show team0

5: **team0**: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000

link/ether a0:1d:48:97:b6:b4 brd ff:ff:ff:ff:ff:ff

inet 192.168.100.150/24 brd 192.168.100.255 scope global noprefixroute team0

valid_lft forever preferred_lft forever

inet6 fe80::64f6:e3c8:876c:663/64 scope link noprefixroute

valid_lft forever preferred_lft forever

Cliente → **192.168.100.150/24**

director-01 **VIP (alias temporal)**

enp1s0192.168.10.11/24 **192.168.10.50/32**

gw4 192.168.10.5

---

real-01	VIP	real-02
enp1s0192.168.10.101/24	**192.168.10.50/32 (lo:50)**	**192.168.10.102/24**
gw4 192.168.10.5		**gw4 192.168.10.5**

→ Configuración del director → director-01

director-01 ~ # yum install ipvsadm

director-01 ~ # vim /etc/sysctl.conf

net.ipv4.ip_forward = 0

director-01 ~ # sysctl -p

net.ipv4.ip_forward = 0

→ Asignamos la VIP (temporal) para → director-01 → enp1s0:50

director-01 ~ # ifconfig enp1s0:50 192.168.10.50 broadcast 192.168.10.50 netmask 255.255.255.255 up

director-01 ~ # ifconfig enp1s0:50

enp1s0:50: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500

inet 192.168.10.50 netmask 255.255.255.255 broadcast 192.168.10.50

ether 52:54:00:eb:ef:dc txqueuelen 1000 (Ethernet)

director-01 ~ # route -n

Kernel IP routing table

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
0.0.0.0	192.168.10.5	0.0.0.0	UG	100	0	0	enp10s0
192.168.10.0	0.0.0.0	255.255.255.0	U	100	0	0	enp10s0

→ Eliminamos cualquier LVS previo.

director-01 ~ # ipvsadm -C

→ Añadimos un servicio futuro http (80/tcp) a la VIP → 192.168.10.50/32

director-01 ~ # ipvsadm -A -t 192.168.10.50:80 -s rr

→ Añadimos los servidores reales para el servicio http.

director-01 ~ # ipvsadm -a -t 192.168.10.50:80 -r 192.168.10.101:80 -g -w 1

director-01 ~ # ipvsadm -a -t 192.168.10.50:80 -r 192.168.10.102:80 -g -w 1

director-01 ~ # ipvsadm -ln

IP Virtual Server version 1.2.1 (size=4096)

Prot LocalAddress:Port Scheduler Flags

-> RemoteAddress:Port Forward Weight ActiveConn InActConn

TCP 192.168.10.50:80 rr

-> 192.168.10.101:80 Route 1 0 0

-> 192.168.10.102:80 Route 1 0 0

(Ya tenemos el director → **director-01** en modo route/gateway)

→ Configuración de los Servidores Reales → real-01 / real-02

Modificar los parámetros del núcleo en cada Servidor Real para limitar el nivel de respuesta y notificación de un **arp factible**.

→ Límite nivel de respuesta: **arp_ignore ← respuesta**
0: El valor predeterminado, lo que significa que cualquier dirección configurada en cualquier interfaz local se puede utilizar para responder.
=> 1: Solo cuando la IP de destino de la solicitud se configura en la interfaz del host local para recibir el mensaje de solicitud, se dará la respuesta.

→ Nivel de anuncio restringido: **arp_announce ← anuncio**
0: El valor predeterminado, toda la información de todas las interfaces en la máquina se anuncia a la red en cada interfaz.
1: Intente evitar las notificaciones a redes conectadas indirectamente.
=> 2: Debe evitar notificar a las redes no locales.

→ Nos preparamos para el problema ARP indicando al núcleo Linux algunos parámetros.

→ Nota: → → real-0X →significa aplicar para: real-01 / real-02

(Posiblemente indicar al núcleo →net.ipv4.ip_forward = 0 no sea necesario !. Pero para que jugar con fuego !.).

real-0X ~ # vim /etc/sysctl.conf

```
net.ipv4.conf.lo.arp_ignore = 1
net.ipv4.conf.lo.arp_announce = 2
net.ipv4.conf.all.arp_ignore = 1
net.ipv4.conf.all.arp_announce = 2
net.ipv4.ip_forward = 0
```

real-0X ~ # sysctl -p

```
net.ipv4.conf.lo.arp_ignore = 1
net.ipv4.conf.lo.arp_announce = 2
net.ipv4.conf.all.arp_ignore = 1
net.ipv4.conf.all.arp_announce = 2
net.ipv4.ip_forward = 0
```

real-0X ~ # route -n

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.10.5    0.0.0.0         UG    100    0        0 enp10s0
192.168.10.0    0.0.0.0         255.255.255.0   U     100    0        0 enp10s0
```

→ Abordamos el problema ARP creando la VIP con máscara→ /32 → lo:50

Observaciones: La dirección VIP posiblemente pueda crearse en la propia interfaz de red → enp1s0. Debe probarse no obstante. De todas formas la red se verá mas expuesta de esta forma al ser un interfaz no local. Consejo → Utilizar → loopback:alias → local → lo:50

real-0X ~ # ifconfig lo:50 192.168.10.50 broadcast 192.168.10.50 netmask 255.255.255.255 up

real-0X ~ # ifconfig lo:50

```
lo:50: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 192.168.10.50  netmask 255.255.255.255
```

loop txqueuelen 1000 (Local Loopback)

→ **Desde → real-01 / real-02**

 # dnf install httpd

 # systemctl enable --now httpd.service

 # firewall-cmd --permanent --add-service=http

 # firewall-cmd --reload

→ **Incorporamos testigos en ambos servidores reales → real-01 / real-02**

real-01 ~ # vim /var/www/html/index.html

Servidor Real 1

real-02 ~ # vim /var/www/html/index.html

Servidor Real 2

→ **Pruebas**

director-01 ~ # ipvsadm -lnc

IPVS connection entries

pro expire state source virtual destination

→ **Desde nuestro cliente:**

redhat00 ~ # curl 192.168.10.50:80

Servidor Real 1

redhat00 ~ # curl 192.168.10.50:80

Servidor Real 2

→ Después de algunas conexiones, ...

director-01 ~ # ipvsadm -lnc

IPVS connection entries

pro expire state source virtual destination

TCP 01:54 FIN_WAIT 192.168.10.5:48750 192.168.10.50:80 192.168.10.101:80

TCP 01:55 FIN_WAIT 192.168.10.5:48752 192.168.10.50:80 192.168.10.102:80

(Persistencia por defecto → 120 segundos).

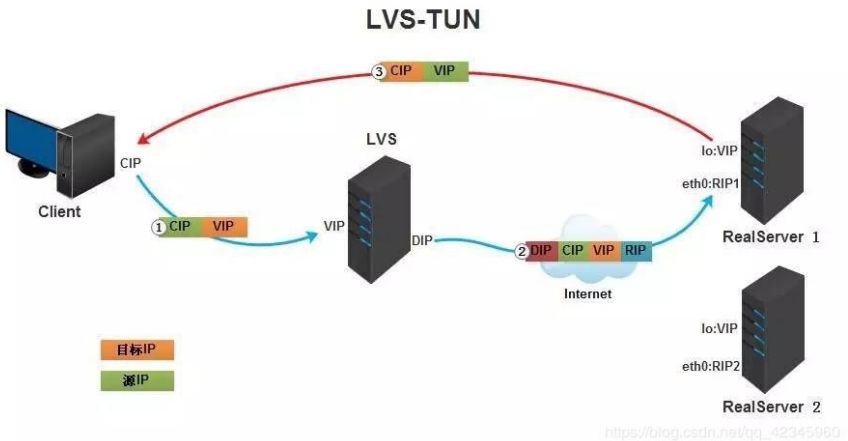
3 => LVS Tunneling → Un solo director → director-01

Cliente → 192.168.100.150/24

director-01	tunl0
enp1s0192.168.10.11/24	192.168.10.50/32
gw4 192.168.10.5	

---

real-01	tunl0	real-02
enp1s0192.168.10.101/24	192.168.10.50/32	192.168.10.102/24
gw4 192.168.10.5		gw4 192.168.10.5



El cliente envía la solicitud al equilibrador de carga frontal, la dirección de origen del mensaje de solicitud es CIP y la dirección de destino es VIP.

Después de recibir el mensaje, el equilibrador de carga descubre que la solicitud es la dirección existente en la regla, luego encapsulará otra capa de mensaje IP en el encabezado del mensaje de solicitud del cliente, cambiará la dirección de origen a DIP y la dirección de destino Cambie a RIP y envíe este paquete a RS.

Después de recibir el mensaje de solicitud, RS primero desempaquetará la primera capa de encapsulación y luego encontrará que también hay una capa de encabezado IP cuya dirección de destino es el VIP en su propia interfaz lo, por lo que procesará el segundo mensaje de solicitud e informará la respuesta El texto se envía a la tarjeta de red eth0 a través de la interfaz lo y se envía directamente al cliente.

Nota:

Los VIP que necesitan configurar la interfaz lo no pueden aparecer en la red pública.

Resumen:

1.El modo TUNNEL debe vincular la dirección IP VIP en todas las máquinas de servidor real.

2. Vip en modo TUNNEL → Comunicación de paquetes Real Server a través del modo TUNNEL, las redes internas y externas pueden comunicarse, por lo que no hay necesidad de lvs vip y real server en el mismo segmento de red.

3.El modo real TUNNEL enviará el paquete directamente al cliente, no a lvs.

4. El modo de túnel es un modo de túnel, por lo que es difícil de operar y mantener, por lo que generalmente no es necesario.

Ventajas:

El equilibrador de carga solo es responsable de distribuir el paquete de solicitud al servidor de nodo de fondo, mientras que el RS envía directamente el paquete de respuesta al usuario. Por lo tanto, se reduce una gran cantidad de flujo de datos del equilibrador de carga. El equilibrador de carga ya no es el cuello de botella del sistema y puede manejar una gran cantidad de solicitudes. De esta manera, un equilibrador de carga puede distribuir muchos RS. Y ejecutar en la red pública se puede distribuir en diferentes regiones.

Desventajas:

Los nodos RS en modo túnel requieren IP legal. Este método requiere que todos los servidores admitan el protocolo "IP Tunneling" (Encapsulación IP). El servidor puede estar limitado a algunos sistemas Linux.

→ Asignamos direcciones Tunneling → tun10 → director-01

director-01 ~ # modprobe ipip

director-01 ~ # lsmod | grep ipip

```
ipip                   16384  0
tunnel4                16384  1 ipip
ip_tunnel              28672  1 ipip
```

director-01 ~ # ifconfig tunl0 192.168.10.50 netmask 255.255.255.255 broadcast 192.168.10.50 up

director-01 ~ # ifconfig tunl0

```
tunl0: flags=193<UP,RUNNING,NOARP>  mtu 1480
        inet 192.168.10.50  netmask 255.255.255.255
        tunnel   txqueuelen 1000  (IPIP Tunnel)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

director-01 ~ # route add -host 192.168.10.50 dev tunl0

director-01 ~ # route -n

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.10.5    0.0.0.0         UG    100    0        0 enp1s0
192.168.10.0    0.0.0.0         255.255.255.0   U     100    0        0 enp1s0
192.168.10.50   0.0.0.0         255.255.255.255 UH    0      0        0 tunl0
```

→ Desde → real-01 / real-02

real-0X ~ # modprobe ipip

real-0X ~ # lsmod | grep ipip

```
ipip                   16384  0
tunnel4                16384  1 ipip
```

ip_tunnel 28672 1 ipip

→ **Problema ARP**

real-0X ~ # sysctl -p

net.ipv4.conf.all.rp_filter = 0

net.ipv4.conf.all.arp_filter = 0

net.ipv4.conf.default.rp_filter = 0

net.ipv4.conf.default.arp_filter = 0

net.ipv4.conf.lo.rp_filter = 0

net.ipv4.conf.lo.arp_filter = 0

net.ipv4.conf.enp1s0.rp_filter = 0

net.ipv4.conf.enp1s0.arp_filter = 0

net.ipv4.conf.tunl0.rp_filter = 0

net.ipv4.conf.tunl0.arp_filter = 0

net.ipv4.ip_forward = 0

→ **Configurar de forma temporal el interfaz tunnel → tunl0**

real-0X ~ # ifconfig tunl0 192.168.10.50 netmask 255.255.255.255 broadcast 192.168.10.50 up

→ **Desde → director-01**

director-01 ~ # ipvsadm

IP Virtual Server version 1.2.1 (size=4096)

Prot LocalAddress:Port Scheduler Flags

-> RemoteAddress:Port Forward Weight ActiveConn InActConn

director-01 ~ # ipvsadm -A -t 192.168.10.50:80 -s rr

director-01 ~ # ipvsadm -a -t 192.168.10.50:80 -r 192.168.10.101:80 -i -w 1

director-01 ~ # ipvsadm -a -t 192.168.10.50:80 -r 192.168.10.102:80 -i -w 1

director-01 ~ # ipvsadm -ln

IP Virtual Server version 1.2.1 (size=4096)

Prot LocalAddress:Port Scheduler Flags

-> RemoteAddress:Port Forward Weight ActiveConn InActConn

TCP 192.168.10.50:80 rr

-> 192.168.10.101:80 Tunnel 1 0 0

-> 192.168.10.102:80 Tunnel 1 0 0

→ Pruebas

director-01 ~ # ipvsadm -lnc

IPVS connection entries

pro expire state source virtual destination

redhat00 ~ # curl 192.168.10.50:80

Servidor Real 1

redhat00 ~ # curl 192.168.10.50:80

Servidor Real 2

director-01 ~ # ipvsadm -lnc

IPVS connection entries

pro expire state source virtual destination

TCP 01:53 FIN_WAIT 192.168.10.5:49710 192.168.10.50:80 192.168.10.102:80

TCP 01:52 FIN_WAIT 192.168.10.5:49708 192.168.10.50:80 192.168.10.101:80

4 => Load Balancer → Visión general → keepalived → haproxy

Un Load Balancer es un conjunto de componentes integrados en un software que proporcionan balanceo de tráfico IP en varios servidores reales. Todo consiste en dos principales tecnologías que monitorizan el cluster y los servicios: keepalived y haproxy.

keepalived => utiliza **lvs** → Linux Virtual Server, para controlar el balanceo de carga y sus fallos en routers activo/pasivo . **Capa 4 OSI** → TRANSPORTE.

haproxy => Se encarga del balanceo de carga y de la disponibilidad de los servicios para las aplicaciones TCP y HTTP. **Capa 7 OSI** → APLICACIÓN.

→ keepalived

Es un daemon ejecutado en activo/pasivo para controlar los routers utilizando el protocolo **VRRP**. El router activo envia advertencias periódicas a systemctl el cual interpreta la configuración estandar desde → **/etc/keepalived/keepalived.conf**

→ Actua en la capa 4 OSI → TRANSPORTE

→ haproxy

Su misión es balancear servicios basados en **tcp** y **http** basandose en un algoritmo de scheduling.

Todo se define para su frontend a traves de una dirección VIP que escucha al backend solicitado.

→ Actua en la capa 7 OSI → APLICACIÓN

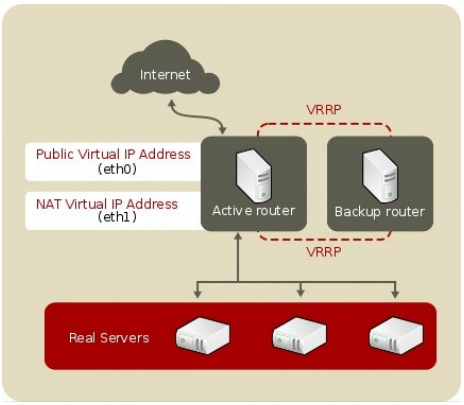
→ → keepalived configuración básica → /etc/keepalived/keepalived.conf

==> En el momento de escribir esta configuración el servicio → **NetworkManager** no es compatible para RedHat, por tanto debe utilizarse el servicio → **network**

1 → Planteamiento de keepalived para → NAT → 2 directores

	director-01	VIP (alias temporal)	director-02
enp10s0	192.168.10.11/24	192.168.10.50/32 gw4 192.168.10.5	192.168.10.12/24

	real-01	GW	real-02
enp10s0	192.168.10.101/24	gw 192.168.10.5	192.168.10.102/24



→ Instalar keepalived → director-01 / director-02

director-0X ~ # dnf install keepalived ipvsadm

→ Configuraciones de ejemplo:

director-0X ~ # ls /usr/share/doc/keepalived/

AUTHOR keepalived.conf.PING_CHECK keepalived.conf.vrrp

ChangeLog keepalived.conf.quorum keepalived.conf.vrrp.localcheck

CONTRIBUTORS keepalived.conf.sample keepalived.conf.vrrp.lvs_syncd

COPYING keepalived.conf.SMTP_CHECK keepalived.conf.vrrp.routes

keepalived.conf.conditional_conf keepalived.conf.SSL_GET keepalived.conf.vrrp.rules

keepalived.conf.fwmark keepalived.conf.status_code keepalived.conf.vrrp.scripts

keepalived.conf.HTTP_GET.port keepalived.conf.SYNOPSIS
keepalived.conf.vrrp.static_ipaddress

keepalived.conf.inhibit keepalived.conf.track_interface keepalived.conf.vrrp.sync

keepalived.conf.IPv6 keepalived.conf.UDP_CHECK README

keepalived.conf.misc_check keepalived.conf.virtualhost TODO

keepalived.conf.misc_check_arg keepalived.conf.virtual_server_group

→ La configuración de keepalived se plantea basicamente en 3 grupos o instancias:

1→ Definiciones Globales.

notification_email: Quien recibe el email de alerta.

notification_email_from: Remite del email de alerta.

lvs_id: Identificador del director.

2→ vrrp instancias.

state: MASTER / BACKUP

priority: MASTER mayor prioridad.

3→ virtual servers.

lb_algo: Algoritmo de balanceo de carga rr|wrr|lc|wlc|lblc|sh|dh

lb_kind: Tipo de Balanceo de carga NAT|DR|TUN

real_server: Servidor Real del pool.

weight: Peso del servidor para balancear la carga.

TCP_CHECK: Método para chequear la disponibilidad del servidor.

* → Health Checking

Para el *health-checking* de los servicios balanceados *keepalived* puede realizar la comprobación de varias formas:

→ ***TCP_CHECK*** se hará una petición TCP y si no es respondida se eliminará el servidor de la lista de servidores activos.

```
TCP_CHECK {
        connect_timeout 10
        port_connect 80
    }
```

→ ***HTTP_GET*** Se solicitará una página del servidor y se comprobará con una que es la correcta mediante un hashing MD5, previamente calculado, en caso de no coincidir el servidor será eliminado de la lista de servidores activos. Para general el hashing MD5 utilizaremos la utilidad **genhash** que nos permitirá generar el hashing directamente desde el servidor.

```
HTTP_GET {
        url {
        path check/200.jsp  #check uri address
        # Generated by genhash that  comes with keepalived: /usr/bin/genhash -s rs_IP -p port -u uri
        digest 1362a91278f0806aa1d33e1e26d67763
    }
```

→ ***SSL_GET*** igual que *HTTP_GET* pero la página será solicitada bajo una conexión SSL por el puerto 443 (https).

→ ***MISC_CHECK*** esta opción nos permitirá comprobar la disponibilidad de los servidores mediante un script creado por nosotros. Si la situación lo requiere podremos comprobar la disponibilidad de los servidores de forma personalizada.

```
MISC_CHECK {
        # External program or script  path and parameters
        misc_path "/etc/keepalived/misc_check.sh   http://192.168.2.222:6500/check/200.jsp"
        misc_timeout 10        # Script execution timeout
        # If the script returns 0, the success weight is unchanged. If 1 is     returned, the failure
        weight is set to 0.
        misc_dynamic
   }
```

```
#!/bin/bash
# ./misc_check.sh http://192.168.2.222:6500/check/200.jsp

if [ $# -ne 1 ]; then
```

```
    echo "Warning: command param error."
    exit 1
else
    CHECK_URL=$1
    CMD=`/usr/bin/curl -I ${CHECK_URL} 2>/dev/null | grep "200 OK" | wc -l`

    if [ ${CMD} -eq 1 ]; then
         echo "Succ: Check proxy ${CHECK_URL} is succeed."
       exit 0

    else
       echo "Fail: check proxy ${CHECK_URL} is failed."
       exit 1
    fi
fi
```

→ **Proceso de Configuración:**

director-01 ~ # mv /etc/keepalived/keepalived.conf /etc/keepalived/keepalived.conf.original

director-02 ~ # mv /etc/keepalived/keepalived.conf /etc/keepalived/keepalived.conf.original

director-01 ~ # vim /etc/keepalived/ keepalived.conf

```
## Definiciones Globales:
global_defs {
   notification_email {
     admin@cadilinea.lan
   }
   notification_email_from admin@cadilinea.lan
   smtp_server 127.0.0.1
   smtp_connect_timeout 30
   router_id LVS_50
   lvs_id DIR-11              # Identificador del director  DIR-11 /  DIR-12
}
## VRRP Instances:
vrrp_instance VI_50 {
    state MASTER              # BACKUP    → director-02
    interface enp10s0
```

```
    virtual_router_id 50
    priority 100                # 90            → director-02
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 123456
    }
    virtual_ipaddress {
        192.168.10.50/32
    }
}
## Virtual Servers:
virtual_server 192.168.10.50 80 {
    delay_loop 6
    lb_algo rr
    lb_kind NAT
    #persistence_timeout 50
    protocol TCP
    #sorry_server 192.168.10.103 80
    real_server 192.168.10.101 80 {
        weight 1
        TCP_CHECK {
            connect_timeout 10
        }
        retry 3
        delay_before_retry 3
    }
    real_server 192.168.10.102 80 {
```

```
        weight 1
        TCP_CHECK {
                connect_timeout 10
        }
        retry 3
        delay_before_retry 3
    }
}
```

director-01 ~ # **rsync -avrh /etc/keepalived/keepalived.conf director-02:/etc/keepalived/keepalived.conf**

→ Tener en cuenta el cambio para → director-02

```
state MASTER        # BACKUP    → director-02
priority 100        # 90        → director-02
```

director-0X ~ # **vim /etc/sysctl.conf**

```
net.ipv4.ip_forward = 1
```

director-0X ~ # **sysctl -p**

```
net.ipv4.ip_forward = 1
```

director-0X ~ # **systemctl restart keepalived.service**

director-0X ~ # **systemctl status keepalived.service**

```
● keepalived.service - LVS and VRRP High Availability Monitor
   Loaded: loaded (/usr/lib/systemd/system/keepalived.service; disabled; vendor preset: disabled)
   Active: active (running) since Wed 2021-06-16 11:50:53 CEST; 8s ago
  Process: 4291 ExecStart=/usr/sbin/keepalived $KEEPALIVED_OPTIONS (code=exited, status=0/SUCCESS)
 Main PID: 4293 (keepalived)
    Tasks: 3 (limit: 17418)
   Memory: 1.9M
```

CGroup: /system.slice/keepalived.service

├─4293 /usr/sbin/keepalived -D

├─4294 /usr/sbin/keepalived -D

└─4295 /usr/sbin/keepalived -D

jun 16 11:50:56 director-01.cadilinea.lan Keepalived_vrrp[4295]: (VI_50) Receive advertisement timeout

jun 16 11:50:56 director-01.cadilinea.lan Keepalived_vrrp[4295]: (VI_50) Entering MASTER STATE

jun 16 11:50:56 director-01.cadilinea.lan Keepalived_vrrp[4295]: (VI_50) setting VIPs.

jun 16 11:50:56 director-01.cadilinea.lan Keepalived_vrrp[4295]: (VI_50) Sending/queueing gratuitous ARPs on enp10s0 for 192.168.10.50

jun 16 11:50:56 director-01.cadilinea.lan Keepalived_vrrp[4295]: Sending gratuitous ARP on enp10s0 for 192.168.10.50

jun 16 11:50:56 director-01.cadilinea.lan Keepalived_vrrp[4295]: Sending gratuitous ARP on enp10s0 for 192.168.10.50

jun 16 11:50:56 director-01.cadilinea.lan Keepalived_vrrp[4295]: Sending gratuitous ARP on enp10s0 for 192.168.10.50

jun 16 11:50:56 director-01.cadilinea.lan Keepalived_vrrp[4295]: Sending gratuitous ARP on enp10s0 for 192.168.10.50

jun 16 11:50:56 director-01.cadilinea.lan Keepalived_vrrp[4295]: Sending gratuitous ARP on enp10s0 for 192.168.10.50

jun 16 11:50:59 director-01.cadilinea.lan Keepalived_healthcheckers[4294]: TCP connection to [192.168.10.102]:tcp:80 success.

→ Configurar Servidores Reales → real-01 / real-02

dnf install httpd

systemctl enable --now httpd.service

firewall-cmd --permanent --add-service=http

firewall-cmd --reload

real-01 ~ # vim /var/www/html/index.html

Servidor Real 1

real-02 ~ # vim /var/www/html/index.html

Servidor Real 2

→ Test para keepalived en modo → NAT

director-01 ~ # ipvsadm -ln

IP Virtual Server version 1.2.1 (size=4096)

Prot LocalAddress:Port Scheduler Flags

-> RemoteAddress:Port Forward Weight ActiveConn InActConn

TCP 192.168.10.50:80 rr

-> 192.168.10.101:80 Masq 1 0 0

-> 192.168.10.102:80 Masq 1 0 0

director-01 ~ # curl 192.168.10.50:80

Servidor Real 2

director-01 ~ # curl 192.168.10.50:80

Servidor Real 1

director-01 ~ # ipvsadm -lnc

IPVS connection entries

pro expire state source virtual destination

TCP 01:55 TIME_WAIT 192.168.10.50:52950 192.168.10.50:80 192.168.10.101:80

TCP 01:53 TIME_WAIT 192.168.10.50:52948 192.168.10.50:80 192.168.10.102:80

2→ Planteamiento de keepalived para → DR (Direct Routing) → 2 directores

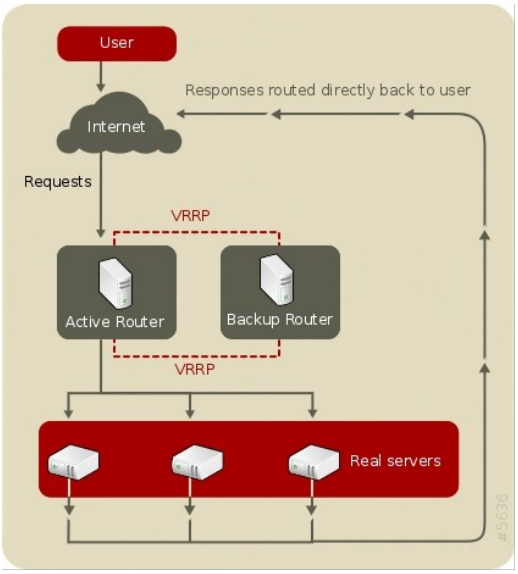
---

Cliente → **192.168.100.150/24**

director-01	VIP (alias temporal)	director-02
enp1s0**192.168.10.11/24**	**192.168.10.50/32**	enp1s0**192.168.10.12/24**
gw4 192.168.10.5		

---

real-01	VIP	real-02
enp1s0192.168.10.101/24	192.168.10.50/32 (lo:50)	192.168.10.102/24
gw4 192.168.10.5		gw4 192.168.10.5



El problema ARP

Una situación a resolver en esta configuración es el "Problema ARP". Como varios servidores tienen la misma IP, si la publican ante un requerimiento ARP habrá una competencia por el paquete, llamado "Race Condition". Para evitar eso, debemos asegurarnos que el único que publica la dirección IP Virtual es el director, por lo que debemos deshabilitar la publicación ARP en los servidores reales.

director-0X ~ # sysctl -p

net.ipv4.ip_forward = 0

real-0X ~ # sysctl -p

net.ipv4.conf.lo.arp_ignore = 1

net.ipv4.conf.lo.arp_announce = 2

net.ipv4.conf.all.arp_ignore = 1

net.ipv4.conf.all.arp_announce = 2

net.ipv4.ip_forward = 0

```
real-0X ~ # ifconfig lo:50 192.168.10.50 broadcast 192.168.10.50 netmask 255.255.255.255 up
real-0X ~ # ifconfig lo:50
lo:50: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 192.168.10.50  netmask 255.255.255.255
        loop  txqueuelen 1000  (Local Loopback)
real-0X ~ # route add -host 192.168.10.50 dev lo

director-0X ~ # vim /etc/keepalived/keepalived.conf
…
## Virtual Servers:
virtual_server 192.168.10.50 80 {
    delay_loop 6
    lb_algo rr
    lb_kind DR
…
director-0X ~ # systemctl restart keepalived.service
director-0X ~ # ipvsadm -ln
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  192.168.10.50:80 rr
  -> 192.168.10.101:80            Route   1      0          0
  -> 192.168.10.102:80            Route   1      0          0

redhat00 ~ # curl 192.168.10.50:80
Servidor Real 1
redhat00 ~ # curl 192.168.10.50:80
Servidor Real 2
```

director-01 ~ # ipvsadm -lnc

IPVS connection entries

pro expire state source virtual destination

TCP 01:50 FIN_WAIT 192.168.10.5:48418 192.168.10.50:80 192.168.10.101:80

TCP 01:52 FIN_WAIT 192.168.10.5:48420 192.168.10.50:80 192.168.10.102:80

3→ Planteamiento de keepalived para → TUN (Direct Tunneling)→ 2 directores

(Planteamiento similar a DR).

Cliente → 192.168.100.150/24

director-01	Tunnel (VIP/tunlo)	director-02
enp1s0192.168.10.11/24	**192.168.10.50/32**	**enp1s0192.168.10.12/24**
gw4 192.168.10.5		

---

real-01	Tunnel (tunl0)	real-02
enp1s0192.168.10.101/24	**192.168.10.50/32**	**192.168.10.102/24**
gw4 192.168.10.5		**gw4 192.168.10.5**

---

director-0X ~ # sysctl -p

net.ipv4.ip_forward = 0

director-0X ~ # modprobe ipip

director-0X ~ # lsmod | grep ipip

ipip 16384 0

tunnel4 16384 1 ipip

ip_tunnel 28672 1 ipip

director-0X ~ # ifconfig tunl0 192.168.10.50 netmask 255.255.255.255 broadcast 192.168.10.50 up

director-0X ~ # ifconfig tunl0

```
tunl0: flags=193<UP,RUNNING,NOARP>  mtu 1480
        inet 192.168.10.50  netmask 255.255.255.255
        tunnel   txqueuelen 1000  (IPIP Tunnel)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

director-0X ~ # route add -host 192.168.10.50 dev tunl0

director-0X ~ # route -n

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.10.5    0.0.0.0         UG    100    0        0 enp1s0
192.168.10.0    0.0.0.0         255.255.255.0   U     100    0        0 enp1s0
192.168.10.50   0.0.0.0         255.255.255.255 UH    0      0        0 tunl0
```

→ Desde→ real-01 / real-02

real-0X ~ # modprobe ipip

real-0X ~ # lsmod | grep ipip

```
ipip                   16384  0
tunnel4                16384  1 ipip
ip_tunnel              28672  1 ipip
```

→ Problema ARP

real-0X ~ # sysctl -p

```
net.ipv4.conf.all.rp_filter = 0
net.ipv4.conf.all.arp_filter = 0
net.ipv4.conf.default.rp_filter = 0
net.ipv4.conf.default.arp_filter = 0
```

net.ipv4.conf.lo.rp_filter = 0

net.ipv4.conf.lo.arp_filter = 0

net.ipv4.conf.enp10s0.rp_filter = 0

net.ipv4.conf.enp10s0.arp_filter = 0

net.ipv4.conf.tunl0.rp_filter = 0

net.ipv4.conf.tunl0.arp_filter = 0

net.ipv4.ip_forward = 0

→ Configurar de forma temporal el interfaz → tunl0

real-0X ~ # ifconfig tunl0 192.168.10.50 netmask 255.255.255.255 broadcast 192.168.10.50 up

real-01 ~ # ip addr add 192.168.10.50/32 dev tunl0 brd 192.168.10.50

real-0X ~ # route add -host 192.168.10.50 dev tunl0

director-0X ~ # vim /etc/keepalived/keepalived.conf

…

Virtual Servers:

virtual_server 192.168.10.50 80 {

   delay_loop 6

   lb_algo rr

   lb_kind TUN

…

director-0X ~ # systemctl restart keepalived.service

director-0X ~ # ipvsadm -ln

IP Virtual Server version 1.2.1 (size=4096)

Prot LocalAddress:Port Scheduler Flags

  -> RemoteAddress:Port   Forward Weight ActiveConn InActConn

TCP 192.168.10.50:80 rr

  -> 192.168.10.101:80   Tunnel 1   0   0

  -> 192.168.10.102:80   Tunnel 1   0   0

redhat00 ~ # curl 192.168.10.50:80

Servidor Real 1

redhat00 ~ # curl 192.168.10.50:80

Servidor Real 2

director-01 ~ # ipvsadm -lnc

IPVS connection entries

pro expire state source virtual destination

TCP 01:53 FIN_WAIT 192.168.10.5:48650 192.168.10.50:80 192.168.10.101:80

TCP 01:54 FIN_WAIT 192.168.10.5:48652 192.168.10.50:80 192.168.10.102:80

→ haproxy

→ Conceptos Teóricos.

HAProxy es un software que nos permite balancear carga entre varios servidores. Se suele usar para balancear servicios web, pero puede balancear cualquier servicio que funcione bajo protocolo TCP. Veremos varias configuraciones de HAProxy incluso teniendo en cuenta las cookies. Si tienes 2 servidores web -en adelante frontales- cuando el usuario llega por primera vez a la web lo hace en el frontal-01. Cuando el usuario se registra crea una cookie, el usuario siguen navegando y de repente HAProxy envía el tráfico al frontal-02, en ese servidor no existe la cookie y por tanto le cerrará la sesión. (Configurando adecuadamente HAProxy esto no sucederá).

→ HAProxy Scheduling Algorithms

Los algoritmos de programación de HAProxy para el equilibrio de carga se pueden editar en el parámetro **balance** en la sección de backend del archivo de configuración /etc/haproxy/haproxy.cfg. Tenga en cuenta que HAProxy admite la configuración con múltiples backends, y cada backend se puede configurar con un algoritmo de programación.

Round-Robin (***roundrobin***)

Distribuye cada solicitud secuencialmente alrededor del grupo de servidores reales. Con este algoritmo, todos los servidores reales se tratan como iguales sin tener en cuenta la capacidad o la carga. Este modelo de programación se parece al DNS por turnos, pero es más granular debido al hecho de que se basa en la conexión de red y no en el host. La programación **roundrobin** de Load Balancer tampoco sufre los desequilibrios causados por las consultas de DNS almacenadas en caché. Sin embargo, en HAProxy, dado que la configuración de los pesos de los servidores se puede realizar sobre la marcha utilizando este programador, la cantidad de servidores activos está limitada a 4095 por back-end.

Static Round-Robin (***static-rr***)

Distribuye cada solicitud secuencialmente alrededor de un grupo de servidores reales como lo hace Round-Robin, pero no permite la configuración dinámica del peso del servidor. Sin embargo, debido a la naturaleza estática del peso del servidor, no hay limitación en el número de servidores activos en el back-end.

Least-Connection (***leastconn***)
Distribuye más solicitudes a servidores reales con menos conexiones activas. Los administradores con un entorno dinámico con diferentes longitudes de sesión o conexión pueden encontrar que este programador se adapta mejor a sus entornos. También es ideal para un entorno en el que un grupo de servidores tiene diferentes capacidades, ya que los administradores pueden ajustar el peso sobre la marcha utilizando este programador.

Source (***source***)
Distribuye las solicitudes a los servidores mediante el hash de la dirección IP de origen y la divide por el peso de todos los servidores en ejecución para determinar qué servidor recibirá la solicitud. En un escenario en el que todos los servidores se estén ejecutando, la solicitud de IP de origen será atendida de manera consistente por el mismo servidor real. Si hay un cambio en el número o el peso de los servidores en ejecución, la sesión se puede mover a otro servidor porque el resultado de hash/peso ha cambiado.

URI (***uri***)
Distribuye las solicitudes a los servidores mediante el hash de todo el URI (o una parte configurable de un URI) y divide por el peso de todos los servidores en ejecución para determinar qué servidor realizará la solicitud. En un escenario en el que se estén ejecutando todos los servidores activos, la solicitud de IP de destino será atendida constantemente por el mismo servidor real. Este programador se puede configurar adicionalmente por la longitud de caracteres al comienzo de una parte de directorio de un URI para calcular el resultado hash y la profundidad de directorios en un URI (designado por barras diagonales en el URI) para calcular el resultado hash.

URL Parameter (***url_param***)
Distribuye las solicitudes a los servidores buscando una cadena de parámetro particular en una solicitud de URL de origen y realizando un cálculo hash dividido por el peso de todos los servidores en ejecución. Si el parámetro no se encuentra en la URL, el programador utiliza de forma predeterminada la programación por turnos. Se pueden usar modificadores basados en parámetros POST, así como límites de espera basados en el número máximo de octetos que un administrador asigna al peso de un determinado parámetro antes de calcular el resultado hash.

Header Name (***hdr***)
Distribuye solicitudes a los servidores verificando un nombre de encabezado particular en cada solicitud HTTP de origen y realizando un cálculo hash dividido por el peso de todos los servidores en ejecución. Si el encabezado está ausente, el programador utiliza de forma predeterminada la programación por turnos.

RDP Cookie (***rdp-cookie***)
Distribuye las solicitudes a los servidores buscando la cookie RDP para cada solicitud TCP y realizando un cálculo hash dividido por el peso de todos los servidores en ejecución. Si el encabezado está ausente, el programador utiliza de forma predeterminada la programación por turnos. Este método es ideal para la persistencia, ya que mantiene la integridad de la sesión.

→ Secciones HAProxy

1 → global

La configuración global configura los parámetros que se aplican a todos los servidores que ejecutan HAProxy. Una sección global típica puede tener el siguiente aspecto:

```
global
    log 127.0.0.1 local2
    maxconn 4000
    user haproxy
    group haproxy
    daemon
```

En la configuración anterior, el administrador ha configurado el servicio para registrar todas las entradas en el servidor syslog local. De forma predeterminada, esto podría ser: **/var/log/syslog** o alguna ubicación designada por el usuario. El parámetro **maxconn** especifica el número máximo de conexiones simultáneas para el servicio. De forma predeterminada, el máximo es 2000. Los parámetros de **user** y **group** especifican el nombre de usuario y el nombre de grupo al que pertenece el proceso haproxy. Finalmente, el parámetro **daemon** especifica que haproxy se ejecuta como un proceso en segundo plano.

2→ defaults

La sección **defaults** configura los parámetros que se aplican por defecto a todas las subsecciones del proxy en una configuración (frontend, backend y listen). Una sección predeterminada típica puede tener el siguiente aspecto:

```
defaults
    mode                    http
    log                     global
    option                  httplog
    option                  dontlognull
    retries                 3
    timeout http-request    10s
    timeout queue           1m
    timeout connect         10s
    timeout client          1m
    timeout server          1m
```

Cualquier parámetro configurado en la subsección del proxy (frontend, backend o listen) tiene prioridad sobre el valor del parámetro defaults.

mode especifica el protocolo para la instancia de HAProxy. El uso del modo http conecta las solicitudes de origen a servidores reales basados en HTTP, ideal para equilibrar la carga de servidores web. Para otras aplicaciones, use el modo tcp.
log especifica la dirección de registro y las funciones de syslog en las que se escriben las entradas de registro. El valor global hace referencia a la instancia de HAProxy a lo que se especifique en el parámetro de registro en la sección global.
option httplog permite el registro de varios valores de una sesión HTTP, incluidas las solicitudes HTTP, el estado de la sesión, los números de conexión, la dirección de origen y los temporizadores de conexión, entre otros valores.

option dontlognull deshabilita el registro de conexiones nulas, lo que significa que HAProxy no registrará conexiones en las que no se hayan transferido datos. Esto no se recomienda para entornos como aplicaciones web a través de Internet donde las conexiones nulas podrían indicar actividades maliciosas como la exploración de puertos abiertos en busca de vulnerabilidades.
retries especifica el número de veces que un servidor real reintentará una solicitud de conexión después de no poder conectarse en el primer intento.
Los distintos valores de **timeout** especifican el tiempo de inactividad para una solicitud, conexión o respuesta determinada. Estos valores se expresan generalmente en milisegundos (a menos que se indique explícitamente lo contrario) pero pueden expresarse en cualquier otra unidad añadiendo el sufijo de la unidad al valor numérico. Las unidades admitidas son us (microsegundos), ms (milisegundos), s (segundos), m (minutos), h (horas) yd (días). http-request 10s da 10 segundos para esperar una solicitud HTTP completa de un cliente. queue 1m establece un minuto como la cantidad de tiempo de espera antes de que se interrumpa una conexión y un cliente reciba un error 503 o "Servicio no disponible". connect 10s especifica el número de segundos de espera para una conexión exitosa a un servidor. client 1m especifica la cantidad de tiempo (en minutos) que un cliente puede permanecer inactivo (no acepta ni envía datos). server 1m especifica la cantidad de tiempo (en minutos) que se le da a un servidor para aceptar o enviar datos antes de que se agote el tiempo de espera.

3→ frontend

La configuración **frontend** configura los sockets de escucha de los servidores para las solicitudes de conexión del cliente. Una configuración típica de HAProxy de la interfaz puede verse así:

```
frontend  main
  bind 192.168.0.10:80
  default_backend app
```

El ***frontend*** llamado ***main*** está configurado para la dirección IP 192.168.0.10 y escucha en el puerto 80 usando el parámetro ***bind***. Una vez conectado, el backend de uso especifica que todas las sesiones se conectan al backend de la app.

4→ backend

La configuración del backend especifica las direcciones IP reales del servidor, así como el algoritmo de programación del balanceador de carga. El siguiente ejemplo muestra una sección de back-end típica:

```
backend app
   balance     roundrobin
   server  app1 192.168.1.1:80 check
   server  app2 192.168.1.2:80 check
   server  app3 192.168.1.3:80 check inter 2s rise 4 fall 3
   server  app4 192.168.1.4:80 backup
```

El servidor back-end se llama app. El balance especifica el algoritmo de programación del balanceador de carga que se utilizará, que en este caso es Round Robin (roundrobin), pero puede ser cualquier programador compatible con HAProxy.

Las líneas de servidor especifican los servidores disponibles en el back-end app1 a app4 son los nombres asignados internamente a cada servidor real. Los archivos de registro especificarán los mensajes del servidor por nombre. La dirección es la dirección IP asignada. El valor después de los dos puntos en la dirección IP es el número de puerto al que se produce la conexión en el servidor en particular. La opción de verificación marca un servidor para verificaciones de estado periódicas para garantizar que esté disponible y pueda recibir y enviar datos y tomar solicitudes de sesión.

El servidor app3 también configura el intervalo de verificación de estado en dos segundos (entre 2s), la cantidad de verificaciones que debe pasar app3 para determinar si el servidor se considera healthy (rise 4) y la cantidad de veces que un servidor falla consecutivamente en una verificación antes de que sea considerado failed (fall 3).

5 → listen

listen stats

bind *:8083

mode http

stats enable

stats uri /stats

stats realm HAProxy\ Statistics

stats auth carlos:123456

→ Configurar → keepalived/haproxy → modo NAT → director-01 / director-02

---

	director-01		director-02
	(keepalived/haproxy)	**VIP**	**(keepalived/haproxy)**
enp10s0	**192.168.10.11/24**	**192.168.10.50/32**	**192.168.10.12/24**
		gw4 192.168.10.5	

---

	real-01	GW	real-02
enp10s0	**192.168.10.101/24**	**gw4192.168.10.5**	**192.168.10.102/24**

---

director-0X ~ # subscription-manager repos --enable codeready-builder-for-rhel-8-x86_64-rpms

director-0X~ # dnf install lynx

director-0X ~ # dnf install keepalived haproxy ipvsadm

director-0X ~ # cp /etc/haproxy/haproxy.cfg /etc/haproxy/haproxy.cfg.original

→ Configuramos keepalived en modo NAT → director-01 / director-02

director-0X ~ # vim /etc/keepalived/keepalived.conf

```
## Definiciones Globales:
global_defs {
  notification_email {
    admin@cadilinea.lan
  }
  notification_email_from admin@cadilinea.lan
  smtp_server 127.0.0.1
  smtp_connect_timeout 30
  router_id LVS_50
}
## VRRP Instances:
vrrp_instance VI_50 {
   state MASTER            # BACKUP para director-02
   interface enp10s0
   virtual_router_id 50
   priority 100            # 90 para director-02
   advert_int 1
   authentication {
      auth_type PASS
```

```
        auth_pass 123456
    }
    virtual_ipaddress {
        192.168.10.50/32
    }
}
## Virtual Servers:
```

 # => Configuración en haproxy.cfg

→ Configuración → haproxy

director-01 ~ # vim /etc/haproxy/haproxy.cfg

```
#---------------------------------------------------------------------
# Example configuration for a possible web application.  See the
# full configuration options online.
#
#   https://www.haproxy.org/download/1.8/doc/configuration.txt
#
#---------------------------------------------------------------------


#---------------------------------------------------------------------
# Global settings
#---------------------------------------------------------------------
global
    # to have these messages end up in /var/log/haproxy.log you will
    # need to:
    #
    # 1) configure syslog to accept network log events.  This is done
    #    by adding the '-r' option to the SYSLOGD_OPTIONS in
    #    /etc/sysconfig/syslog
```

```
   #
   # 2) configure local2 events to go to the /var/log/haproxy.log
   #   file. A line like the following can be added to
   #   /etc/sysconfig/syslog
   #
   #    local2.*                       /var/log/haproxy.log
   #
   log         127.0.0.1 local0
   log         127.0.0.1 local0 notice
   maxconn     256
   user        haproxy
   group       haproxy
   daemon
   stats socket /var/run/haproxy.sock mode 600 level admin           # socket para hatop
#---------------------------------------------------------------------
# common defaults that all the 'listen' and 'backend' sections will
# use if not designated in their block
#---------------------------------------------------------------------
defaults
   mode                    http
   log                     global
   option                  httplog
   option                  dontlognull
   option http-server-close
   option forwardfor       except 127.0.0.0/8
   option                  redispatch
   retries                 3
   timeout http-request    10s
```

```
    timeout queue           1m
    timeout connect         10s
    timeout client          1m
    timeout server          1m
    timeout http-keep-alive 10s
    timeout check           10s
    maxconn                 3000


#---------------------------------------------------------------------
# main frontend which proxys to the backends
#---------------------------------------------------------------------
frontend LOADBALANCER
    bind  192.168.10.50:80
    mode http
    default_backend WEBSERVERS
#---------------------------------------------------------------------
# round robin balancing between the various backends
#---------------------------------------------------------------------
backend WEBSERVERS
    balance roundrobin
    server real-01 192.168.10.101:80 weight 1 maxconn 512 check
    server real-02 192.168.10.102:80 weight 1 maxconn 512 check
    option httpchk
    # Activamos estadísticas.
    stats enable
    stats auth carlos:123456


#---------------------------------------------------------------------
```

Visualizar estadísticas haproxy usuario:password -> **carlos:123456**

#---

listen stats

bind *:8083

mode http

stats enable

stats uri /stats

stats realm HAProxy\ Statistics

stats auth carlos:123456

director-01 ~ # haproxy -f /etc/haproxy/haproxy.cfg -c

Configuration file is valid

director-01 ~ # rsync -avrh /etc/haproxy/haproxy.cfg director-02:/etc/haproxy/haproxy.cfg

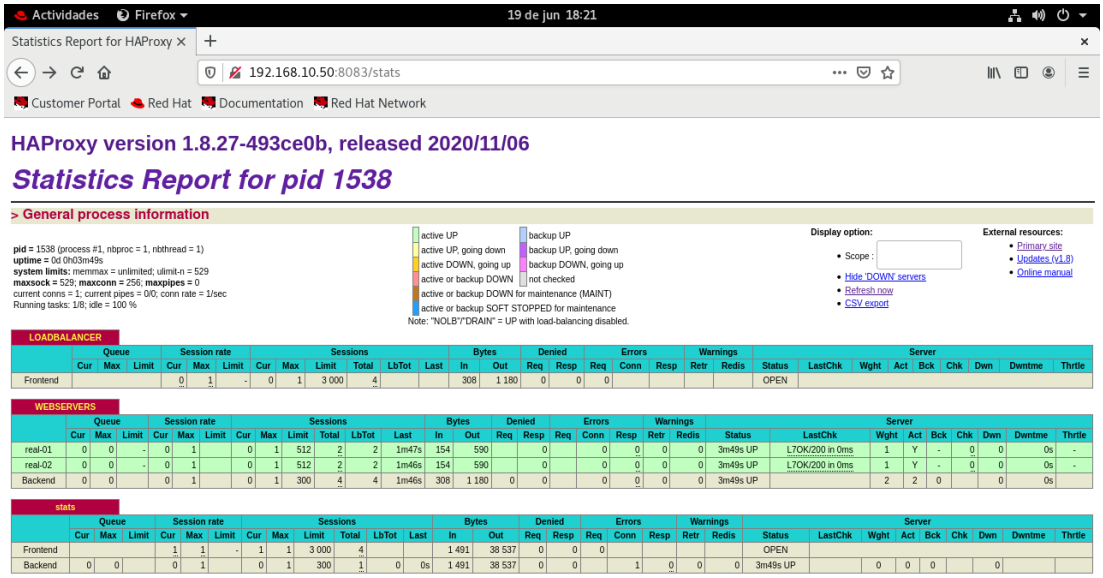
director-0X ~ # sysctl -p

net.ipv4.ip_forward = 1

net.ipv4.ip_nonlocal_bind = 1

director-0X ~ # systemctl start keepalived.service haproxy.service

http://192.168.10.50:8083/stats



→ Instalamos → HATop

director-0X ~ # dnf install git

director-0X ~ # git clone https://github.com/jhunt/hatop.git /opt/hatop

director-0X ~ # vim /etc/haproxy/haproxy.cfg

global

…

stats socket /var/run/haproxy.sock mode 600 level admin

…

director-01 ~ # rsync -avrh /etc/haproxy/haproxy.cfg director-02:/etc/haproxy/haproxy.cfg

director-0X ~ # systemctl restart keepalived.service haproxy.service

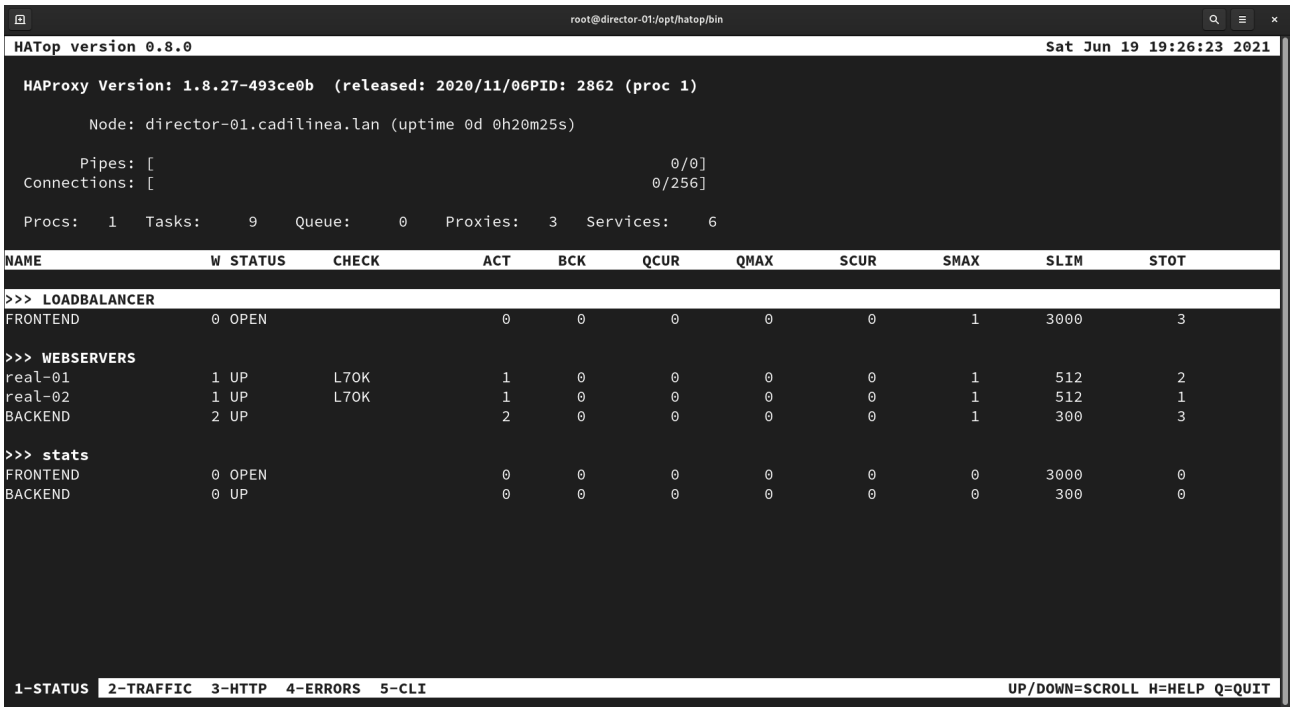
redhat00 ~ # curl 192.168.10.50:80

Servidor Real 1

redhat00 ~ # curl 192.168.10.50:80

Servidor Real 2

director-0X ~ # /opt/hatop/bin/hatop -s /var/run/haproxy.sock



→ Controlar cookies y estados del servidor

director-0X ~ # vim /etc/haproxy/haproxy.cfg

…

#---

round robin balancing between the various backends

#---

backend WEBSERVERS

balance roundrobin

server real-01 192.168.10.101:80 weight 1 maxconn 512 check fall 3 cookie L1

server real-02 192.168.10.102:80 weight 1 maxconn 512 check fall 3 cookie L2

option httpchk

stats enable

stats auth carlos:123456

```
#--------------------------------------------------------------------
# Visualizar estadísticas haproxy usuario:password -> carlos:123456
#--------------------------------------------------------------------
listen stats
    bind *:8083
    mode http
    stats enable
    stats uri /stats
    stats realm HAProxy\ Statistics
    stats refresh 10s                # Refresca estado cada 10 segundos.
# Control de cookies y estado de los servidores
    option httpchk GET /index.php HTTP/1.0
    http-check expect rstatus (2|3)[0-9][0-9]|503
    cookie PHPSESSID prefix indirect nocache
```

director-0X ~ # systemctl restart haproxy.service

BIBLIOGRAFIA:

LIBRO: ==> Linux Solutions de Haute Disponibilité – Sébastian Rohaut – ENI Editions (2010)

https://superuser.openstack.org/articles/ipvs-direct-routing-openstack/

https://docs.oracle.com/en/operating-systems/oracle-linux/6/admin/section_wkd_ys2_4r.html

https://computingforgeeks.com/setup-linux-virtual-server-load-balancer-on-centos-rhel/
https://www.natsav.com/blog/lvs-deployement-centos7/
https://www.osradar.com/how-to-setup-lvs-linux-virtual-server-load-balancer-on-centos-8-rhel-8/
http://www.austintek.com/LVS/LVS-HOWTO/HOWTO/LVS-HOWTO.install.html
https://programmer.group/centos-7-builds-lvs-keepalived-high-available-web-services-cluster.html

https://www.lisenet.com/2015/setting-up-a-load-balancing-lvs-direct-routing-cluster-with-piranha/
https://www.programmersought.com/article/74404005296/
https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/load_balancer_administration/s1-lvs-direct-vsa
https://programming.vip/docs/setup-of-linux-high-availability-lvs-load-balancing-cluster-keepalived-lvs-dr.html
https://sysadmins.co.za/setup-piranha-loadbalancer-on-centos/
https://www.sololinux.es/instalar-configurar-un-lvs-server-load-balancer/
https://debugged.it/blog/ipvs-the-linux-load-balancer/
https://linuxize.com/post/how-to-configure-and-manage-firewall-on-centos-8/
https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html-single/load_balancer_administration/index#s2-lvs-nat-VSA
http://www.iakovlev.org/index.html?p=1184

https://programmerclick.com/article/8358352121/

http://www.austintek.com/LVS/linuxexpo99/

http://www.linuxvirtualserver.org/

https://programmer.help/blogs/lvs-mode-2-tun-tunnel-mode.html

https://extensions.libreoffice.org/en/extensions/show/vrt-network-equipment

https://docs.oracle.com/cd/E37670_01/E41138/html/section_jyh_zhz_4r.html

https://clouding.io/hc/es/articles/360010289000-Balancear-servicio-web-con-HAProxy-con-certificado-SSL-en-Ubuntu-18-04