

Práctica: Servicio web en alta disponibilidad

Objetivos

Configurar un servicio web en alta disponibilidad con dos métodos: mediante un balanceador y mediante un proxy inverso.

Preparación

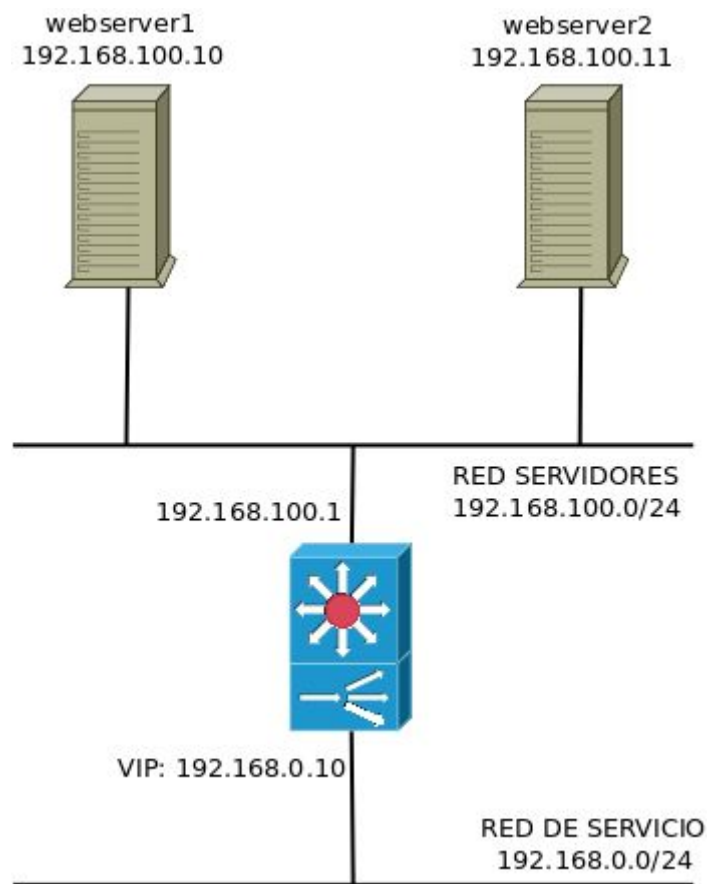
Es necesario el siguiente material:

- Dos servidores web reales o virtuales, sin importar el sistema operativo ni el servidor web (Apache, IIS, etc.)
- Un servidor real o virtual con GNU/Linux, que hará funciones de balanceador y proxy inverso
- Un cliente web
- Documentación de [keepalived](#)

Enunciado

Preparación del escenario

El primer paso es preparar el escenario que se va a usar para configurar un servicio web en alta disponibilidad con una granja de dos servidores web como mínimo. El escenario se muestra en la siguiente figura:



En el escenario se distinguen dos redes: la **red de servidores** (interna) y la **red de servicio** (externa). La red de servicio tendrá el direccionamiento habitual de nuestra red local y en esta red tendrá la IP virtual del servicio (VIP) que vamos a ofrecer en alta disponibilidad. La red de servidores será una red privada interna entre el equipo que hace las funciones de balanceador/proxy inverso y los dos servidores web.

La granja de servidores se encuentra enmascarada por el balanceador y oculta a la red de servicio. Ambas redes tendrán por tanto direccionamiento IP diferente. Se deja la libertad al alumno para que decida el direccionamiento privado de la red de servidores. En el ejemplo, la red de servicio (corresponde con la red local de nuestro domicilio o del centro) es la 192.168.0.0/24, siendo la VIP del servicio la 192.168.0.10. La red interna de servidores es la 192.168.100.0/24, teniendo ambos servidores al balanceador como puerta de enlace con la 192.168.100.1.

El balanceador debe ser por tanto una máquina GNU/Linux de la distribución que desees, pero es necesario habilitar el **ip forwarding** y el **NAT**, para que ambos servidores web puedan alcanzar la red de servicio o Internet. Puedes consultar la segunda práctica del tema 6 donde describe el escenario de prácticas de VPN para ver cómo hacerlo. También es necesario que deshabilites cualquier regla de firewall que pueda interferir con la práctica.

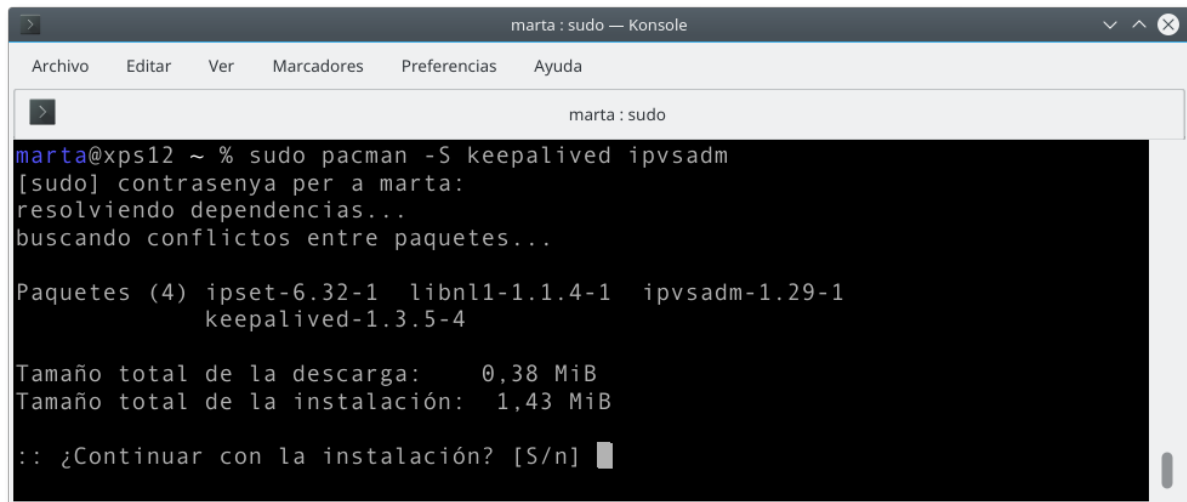
Una vez configurado el escenario, deberías comprobar que las tres máquinas se pueden hacer ping y que los dos servidores web pueden navegar por Internet.

Se deja libertad al alumno para que, en función de los recursos de que disponga, monte el escenario con las máquinas reales o virtuales que quiera. Algunas posibles opciones son:

1. Todos los ordenadores de escenario son máquinas reales. Entendemos que ésta no va a ser la situación habitual, además de requerir dos tarjetas en el equipo que hace de balanceador y que debe

tener instalado GNU/Linux. También es posible hacerlo con una sola tarjeta en el balanceador, pero en este caso, ambas redes deben superponerse en el mismo segmento LAN, teniendo el balanceador doble configuración IP en la tarjeta de red.

2. Tener un anfitrión con GNU/Linux y simular los dos nodos de la granja con una distribución con pocos recursos, como CentOS minimal o Debian con apache instalado. La red interna de servidores es una red virtual sólo anfitrión o host-only. Esta es la configuración que uso yo, host Arch Linux y servidores Debian:



```
marta : sudo — Konsole
Archivo  Editar  Ver  Marcadores  Preferencias  Ayuda

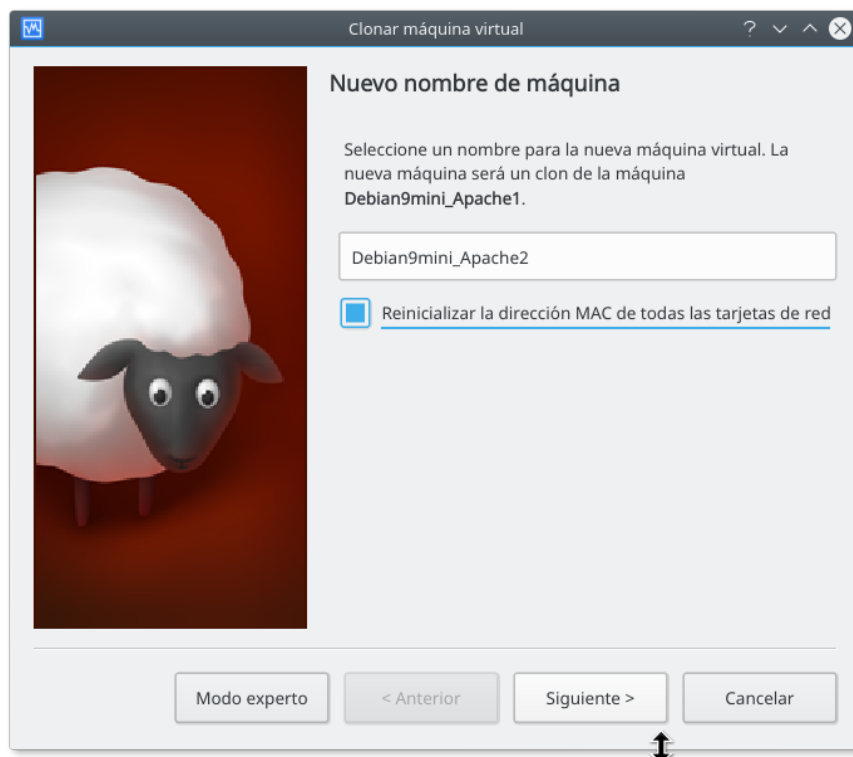
marta : sudo

marta@xps12 ~ % sudo pacman -S keepalived ipvsadm
[sudo] contrasena per a marta:
resolviendo dependencias...
buscando conflictos entre paquetes...

Paquetes (4) ipset-6.32-1  libnl1-1.1.4-1  ipvsadm-1.29-1
              keepalived-1.3.5-4

Tamaño total de la descarga:    0,38 MiB
Tamaño total de la instalación: 1,43 MiB

:: ¿Continuar con la instalación? [S/n]
```



3. Teniendo un anfitrión Windows con suficiente potencia y simular los tres equipos en él. Se recomienda usar en las tres máquinas una distribución con pocos recursos, como CentOS minimal. La máquina que hace de balanceador tendrá dos tarjetas: la interna en modo sólo anfitrión o host-only y con visibilidad hacia los servidores web, y la tarjeta externa en modo puente con la red local del anfitrión. Los

servidores web sólo tendrán una tarjeta en modo sólo anfitrión y teniendo al balanceador como puerta de enlace. Esta será probablemente la opción de la mayoría de vosotros.

4. Cualquier otra que se le ocurra al alumno y sirva para simular el escenario.

Parte 1: Instalación y configuración de un balanceador con keepalived

Keepalived es un paquete software que implementa el protocolo VRRP de alta disponibilidad que habilita el enrutamiento tolerante a fallos para un par o más de routers que funcionan como balanceadores (mediante el software LVS: Linux Virtual Server) y el demonio keepalived que realiza el chequeo de los nodos de la granja de servidores para realizar el balanceo entre ellos según el algoritmo elegido. En esta práctica sólo usaremos la parte de balanceo de carga que nos permite utilizarlo para montar un servicio web en alta disponibilidad balanceado.

Para comenzar con la práctica, realiza los siguientes pasos:

1. Instala el paquete **keepalived** e **ipvsadm** utilizando el sistema de gestión de paquetes de tu distribución. En algunas distribuciones como CentOS o Fedora, no se encuentra en los repositorios oficiales, sino en epel.

2. Como root, crea en tu balanceador el fichero **/etc/keepalived/keepalived.conf** con el siguiente contenido:

```
global_defs {
    lvs_id LVS_PRACTICA
}

virtual_server ip_publica_balanceador 80 {
    delay_loop 3
    # ver man ipvsadm para ver los tipos de algoritmos
    lb_algo rr
    lb_kind NAT
    persistence_timeout 0
    protocol TCP

    real_server ip_webserver1 80 {
        weight 1
        TCP_CHECK {
            connect_port 80
            connect_timeout 3
        }
    }

    real_server ip_webserver2 80 {
        weight 1
        TCP_CHECK {
            connect_port 80
            connect_timeout 3
        }
    }
}
```

Donde debes reemplazar los valores en negrita, por las IP de tu escenario.

Para comprobar el funcionamiento de esta configuración, asegúrate de que se han cargado en el kernel los módulos **ip_vs_rr**, **ip_vs_wrr** y **ip_vs**. Para ello puedes usar desde consola el comando **lsmod**

para verificar si están cargados y si no lo están, los puedes cargar como root con **modprobe ip_vs_rr** y **modprobe ip_vs_wrr** (carga por dependencia el módulo ip_vs).

3. Después crea una página HTML por defecto (archivo /var/www/html/index.html) en cada servidor que muestre el texto “Bienvenido a webserverX” en cada uno de ellos, donde X será 1 o 2 en función del servidor. En un entorno real de producción, ambos sistemas de archivos estarán sincronizados utilizando algún software como **csync2**, **DRBD** o bien un directorio montado por **NFS** o un disco en una **SAN** montado por Fiber Channel Protocol o iSCSI (en estos casos es habitual usar un sistema de ficheros en cluster como GFS2 o OCFS para evitar inconsistencias en el sistema de archivos cuando dos o más nodos acceden al mismo dispositivo por bloques en una SAN).
4. Finalmente, asegúrate de que el servicio web está iniciado en los dos servidores (service httpd/apache2 start o systemctl start https/apache2, consulta tu distribución) e inicia el demonio keepalived en el balanceador mediante el comando correspondiente (service keepalived start o /etc/init.d/keepalived start). Recuerda que en un entorno real de producción, debes asegurarte de que arranca automáticamente este servicio en el inicio del sistema y que se cargan los módulos ip_vs necesarios para que funcione.

Comprueba que el balanceador funciona conectándote con un cliente web (puede ser el propio balanceador) a la VIP de tu balanceador.. Actualiza la página varias veces **¿Qué ocurre?** Modifica el parámetro weight en la configuración de ambos servidores para cambiar el peso de cada servidor web. También tendrás que modificar el algoritmo de balanceo en la configuración de keepalived para que haga caso de los pesos (consulta man ipvsadm) y observa el resultado.

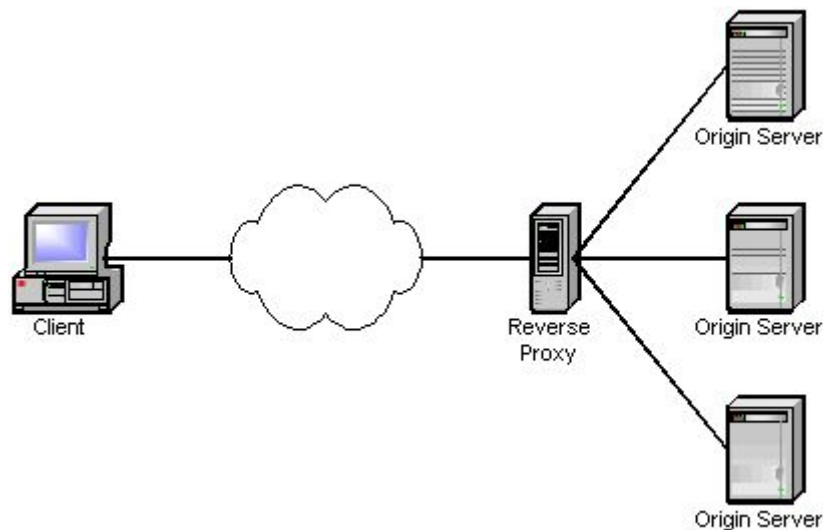
5. Modifica la configuración anterior cambiando el valor de **persistence_timeout** (en segundos) para que haya **persistencia** en las conexiones HTTP de forma que un cliente web siempre se conecte al mismo servidor (afinidad de sesiones web) durante un tiempo de 20 minutos. De esta forma, siempre que un cliente solicite una página al balanceador antes de que pasen esos 20 minutos (tiempo que hemos definido como la duración de una sesión web), se va a conectar al mismo servidor. **¿Qué utilidad le ves a esta persistencia?** Reinicia el servicio keepalived y observa los cambios.
6. Mira en los ejemplos de /usr/share/doc/keepalived-version/samples o en la documentación, como modificar el chequeo que realiza el balanceador (TCP_CHECK) de protocolo y puerto, por uno más sofisticado de forma que si se produce un error de apache en la página html, el texto que muestra el servidor es diferente del correcto y entonces el balanceador considerará que el servidor no funciona bien. Pista: **HTTP_GET** y herramienta **genhash** que instala keepalived de serie. Implementa este chequeo en la configuración del balanceador.

Es posible que tengas dificultades con la configuración anterior. Por ejemplo, yo tuve problemas con el keepalived en su día porque estaba mal empaquetado en mi distribución y posteriormente funcionó bien sin hacer nada cuando actualizaron el paquete. También puedes tener dificultades con la configuración de red (fíjate bien en las direcciones que usas, el NAT, el ip_forward o que no tengas ningún filtrado de iptables). Si es así, prueba a cambiar la máquina que hace de balanceador y configúrala en limpio de nuevo.

No te desanimes si no te ha funcionado bien. La configuración de la parte 2 es más fácil:

Parte 2: Instalación y configuración de un proxy inverso con Apache

Un **proxy inverso**, **proxy reverso** o **reverse proxy** tiene una funcionalidad en concepto opuesta al uso habitual de un proxy, más conocido como proxy directo. Un reverse proxy es un servidor proxy instalado en el domicilio de uno o más servidores web, de manera que todo el tráfico entrante de Internet y con el destino de uno de esos servidores web pasa a través del servidor proxy:



Un proxy inverso proporciona:

- **Seguridad:** el servidor proxy es una capa adicional de defensa y por lo tanto protege los servidores web.
- **Encriptación / Aceleración SSL/TLS:** cuando se crea un sitio web seguro, la encriptación SSL/TLS es realizada por el proxy inverso, el cual suele estar equipado con un hardware de aceleración SSL/TLS liberando la CPU de los servidores que hay detrás.
- **Balanceo de carga:** el proxy reverso puede distribuir la carga entre varios servidores web según distintos algoritmos (round robin, el menos utilizado, el menos ocupado, el más rápido, etc.)
- **Caché** de contenido estático: un proxy reverso puede descargar los servidores web almacenando contenido estático como imágenes y otro contenido gráfico

Hay mucho software libre para poder implementar un proxy inverso, como Squid, nginx o el propio Apache, que es el que usaremos en esta práctica.

Para continuar con la práctica, realiza los siguientes pasos:

1. Detén el servicio keepalived de la práctica anterior para que no interfiera con el proxy inverso que vamos a configurar (service keepalived stop o equivalente) y verifica que tienes instalado el servidor web apache en tu ordenador anfitrión. Una vez verificado, vas a añadir en el directorio /etc/httpd/conf.d un fichero al que puedes llamar proxy-inverso.conf con el siguiente contenido:

```
<VirtualHost ip_publica_balanceador:80>
    ProxyPass / balancer://mcluster/ lbmethod=byrequests
    ProxyPassReverse / balancer://mcluster/
</VirtualHost>
<Proxy balancer://mcluster>
    Order deny,allow
    Allow from all
```

```

        BalancerMember http://ip_webserver1 loadfactor=1
        BalancerMember http://ip_webserver2 loadfactor=1
    </Proxy>
</VirtualHost>

```

Debes reemplazar el texto en **negrita** con los valores correctos. Utilizando la documentación de directivas de Apache o consultando Internet, averigua qué función tiene cada una de las directivas anteriores e intenta comprender que hace función realiza esta configuración en el escenario de la práctica.

2. Habilita los módulos necesarios. Por ejemplo, en un balanceador Debian o similar:

```
# a2enmod proxy proxy_http proxy_balancer lbmethod_byrequests
```

3. Reinicia Apache y comprueba que el balanceador proxy inverso funciona conectándote a la VIP del balanceador (192.168.0.10 en el ejemplo) desde un navegador. Actualiza la página varias veces **¿Qué ocurre?** Modifica el parámetro **loadfactor** en la configuración de ambos servidores y observa el resultado.
4. Modifica la configuración anterior para que haya **persistencia** en las conexiones HTTP de forma que un cliente web siempre se conecte al mismo servidor (afinidad de sesiones web). Esto lo realizaremos mediante cookies ya que Apache dispone de esta facilidad. Para ello modifica el fichero de configuración que creaste para que tenga este aspecto:

```

<VirtualHost ip_publica_balanceador:80>
    Header add Set-Cookie "ROUTEID=.%{BALANCER_WORKER_ROUTE}e; path=/"
    env=BALANCER_ROUTE_CHANGED
    ProxyPass / balancer://micluster/ lbmethod=byrequests
    stickysession=ROUTEID
    ProxyPassReverse / balancer://micluster/
    <Proxy balancer://micluster>
        Order deny,allow
        Allow from all
        BalancerMember http://ip_webserver1 loadfactor=1 route=wsrv1
        BalancerMember http://ip_webserver2 loadfactor=1 route=wsrv2
    </Proxy>
</VirtualHost>

```

5. Consulta en la documentación de Apache la función de estas directivas que has añadido. Reinicia Apache y observa el resultado al actualizar la página. **¿Siempre te aparece el mismo servidor web?** **¿Qué algoritmos de balanceo permite la directiva ProxyPass?**

Sube a Classroom en la actividad correspondiente, una memoria con capturas describiendo el proceso realizado.