PROYECTO INTEGRADO

CLUSTER DE ALTA DISPONIBILIDAD

Alejandro González Pérez
Instituto Gonzalo Nazareno
Ciclo Superior de Administración de Sistemas Informaticos

Indice General

1.- Introducción y Objetivos

Este proyecto integrado consiste en el montaje de un cluster de alta disponibilidad que garantice el servicio web. El cluster está formado por dos ordenadores y dos routers, que garantizan siempre un respaldo de garantía en caso de fallo grave del sistema. Pero antes, para empezar, vamos a colocar alguna definición

¿Qué es un cluster?

- En términos informáticos, entendemos cluster como conjuntos o conglomerados de computadoras construidos mediante la utilización de componentes de hardware comunes y que se comportan como si fuesen una única computadora. (fuente *http://es.wikipedia.org/wiki/Cluster_(inform%C3%A1tica))*

Existen varios tipos de clusters. Alto rendimiento, Alta disponibilidad, Escalabilidad y Balanceo de carga. En el proyecto hemos elegido la opción de montar un cluster de alta disponibilidad. Pero...

¿Qué es un cluster de alta disponibilidad?

- Un cluster de alta disponibilidad es un conjunto de dos o más máquinas que se caracterizan por mantener una serie de servicios compartidos y por estar constantemente monitorizándose entre sí.
  (fuente http://es.wikipedia.org/wiki/Cluster_de_alta_disponibilidad)

El cluster de alta disponibilidad se utilizá para garantizar el que un servicio siempre pueda ser ofrecido, en este caso, nuestro cluster va a proporcionar el servicio WEB a través de un servidor apache. El hecho de usar dos salidas a Internet es ficticio, porque esta opción no es real en la red que dispongo para el proyecto,por lo tanto, usaré dos routers que se conecten al router principal de mi red, que da salida a Internet. Los routers funcionaran cambiando de una red local (192.168.1.0) a otra (192.168.2.0) Y se conectarán a la puerta de enlace principal que es la que provee de Internet (192.168.1.1).

Estaba presente la posibilidad de hacer el sistema de forma simulada a través de herramientas de virtualización como xen (http://www.xen.org/) o vmware (http://www.vmware.com/es/) pero con intención de darle un poco más de *realidad* y consistencia al proyecto, todo se hará con equipos reales.

2.- Configuración del Hardware

2.1 – Hardware de los Servidores

Los dos servidores que formaran el cluster de alta disponibilidad son dos máquinas antiguas. Están recicladas y montadas a base de piezas de varias máquinas. La configuración final de los dos ordenadores es la siguiente:

Corman (Nodo Primario) :

- Procesador Intel Pentium III a 550 Mhz
- 256 MB de RAM
- Dos discos duros, ambos IDE, uno de 8 GB y otro de 4 GB
- Tres tarjetas ethernet 10/100

Rabius (Nodo Secundario) :

- Procesador Intel Pentium III a 700 Mhz
- 256MB de RAM
- Dos discos duros, ambos IDE de 20 GB cada uno.
- Tres tarjetas ethernet 10/100.

2.2 – Routers

Utilizaremos dos routers LINKSYS muy parecidos. Los modelos y su configuración de red es la siguiente:

Router 1 *CLASE* WRT54GL.

IP PUBLICA: 192.168.1.251

IP INTERNA: 192.168.2.1

PUERTA DE ENLACE 192.168.1.1

DNS: 192.168.1.1

- Router 2 *RAFA* WRT54G

IP PUBLICA 192.168.1.252

IP INTERNA: 192.168.2.2

PUERTA DE ENLACE: 192.168.1.1

DNS: 192.168.1.1

En los dos routers hay que habilitar el NAT de puertos para que el servicio web pueda pasar a través del cortafuegos que lleva implementado el router. En ambos, habilitaremos NAT para los puertos 22 y 80 para la IP 192.168.2.10. El puerto 22 es para el SSH y posibilita el control desde un equipo remoto, el 80, permite que la web que se ofrezca pueda ser vista. La IP 192.168.2.10 no corresponde a ningún equipo, si no a una interfaz virtual que explicaremos más adelante.

Para configurar los routers los conectaremos directamente con un cable de red al equipo propio que utilizamos para monitorizar. Dejamos pulsado el botón de reset del router durante 10 segundos para que vuelva a los valores por defectos, y en nuestro navegador escribimos la dirección http://192.168.1.1 Ahora elegimos que el tipo de conexión a Internet sea estática y ponemos los valores de arriba. Luego iremos a la parte de NAT y permitiremos el paso a los puertos 22 y 80 para la IP 192.168.2.10.

2.3 – Esquema de red y conectividad

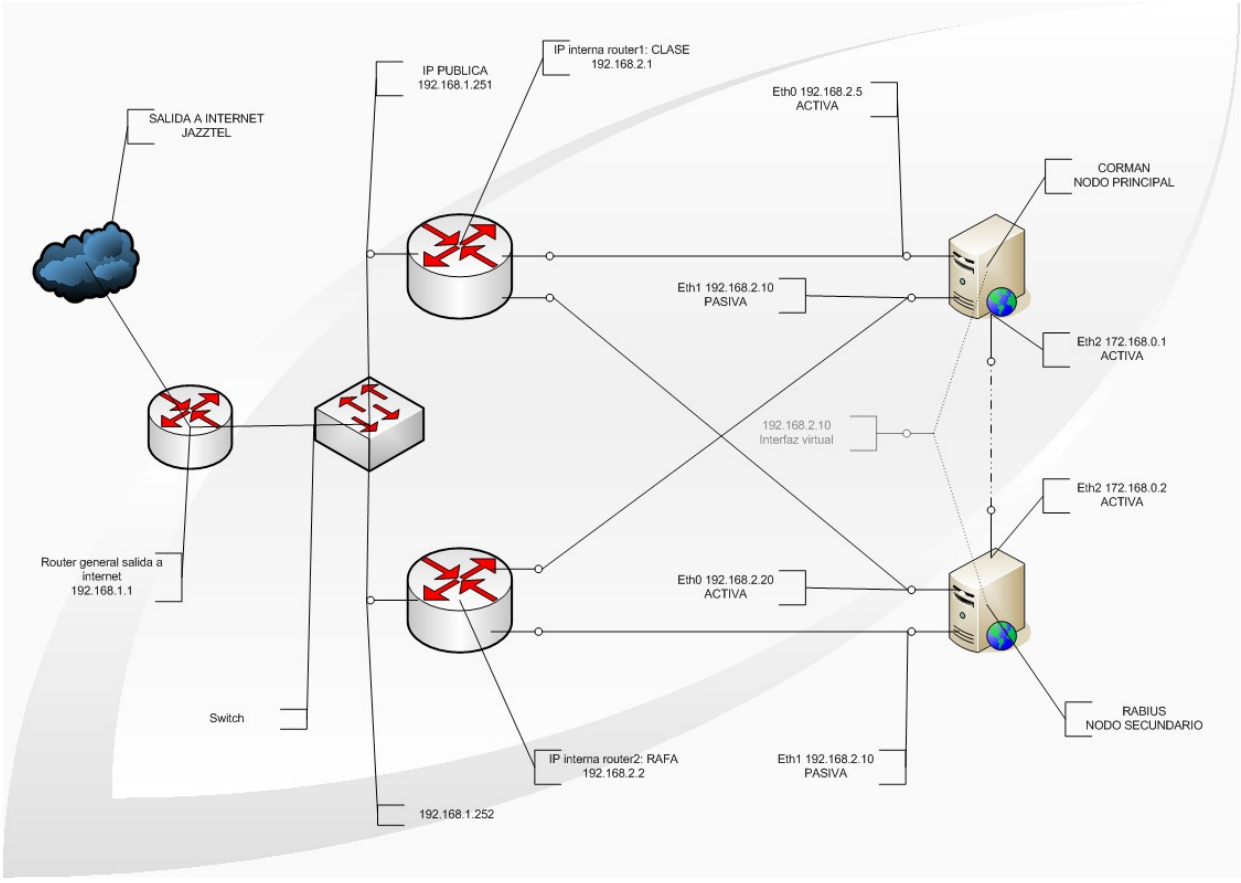
Para la conectividad entre routers y equipos, hemos de utilizar varios cables de red, la forma mas barata y cómoda es crimpar tus propios cables. En este caso, utilizamos 15 metros de cable de red.

Necesitamos varios cables que ahora mostraremos en los esquemas. Estos saldrán de los servidores a los routers y de los routers a un switch que los conecta a la red principal.

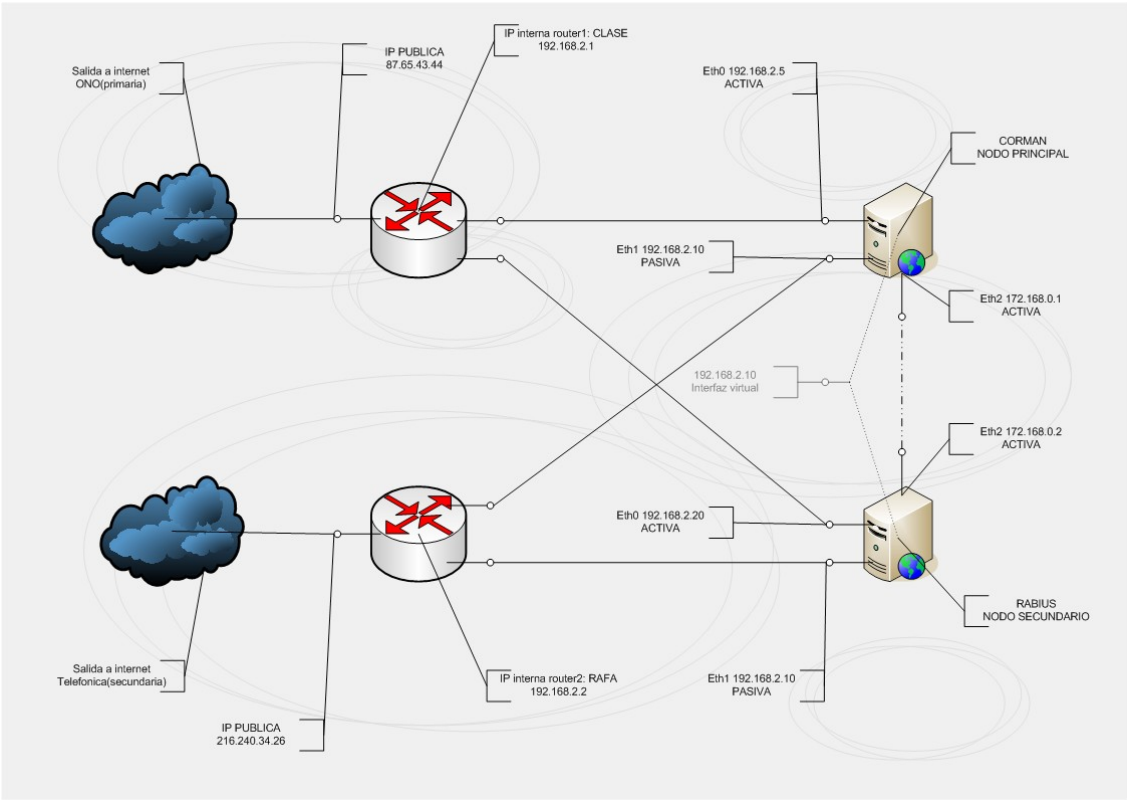
Hay dos tipos de cable de red, el paralelo y el cruzado. Utilizaremos siempre el paralelo salvo en el cable de red que conecta a los dos equipos. Las normas para el crimpado de cables se pueden sacar de varias webs. Entre ellas http://www.configurarequipos.com/doc297.html.

Ahora mostramos el esquema de red. Este lo configuraremos después de la instalación del sistema operativo. Vamos a mostrar dos esquemas de red; el real, que es el que hemos usado en este proyecto, y el ficticio, que es el que usaríamos en una empresa real, con dos salidas a Internet. La única diferencia entre estos esquemas es la conexión a Internet, que en el caso del esquema real es a través de una sola salida, a la que se conectan los dos routers, y en el caso del esquema ficticio es a través de dos salidas diferentes (que garantizaría la alta disponibilidad. En el esquema hay una interfaz VIRTUAL entre los dos servidores, es la 192.168.2.10, que será la interfaz que ofrecerá el servicio, pero ya hablaremos de ella durante la configuración de heartbeat.

El esquema real que hemos usado es el siguiente:



Esquema utilizado en el proyecto



Esquema FICTICIO pensado para dos salidas a Internet

3.- Instalación de Software

En este punto trataremos la instalación de todo el software necesario para el funcionamiento del servidor y las máquinas. En esta sección no trataremos la instalación de los paquetes necesarios para heartbeat o DRBD.

3.1- Instalación del Sistema Operativo

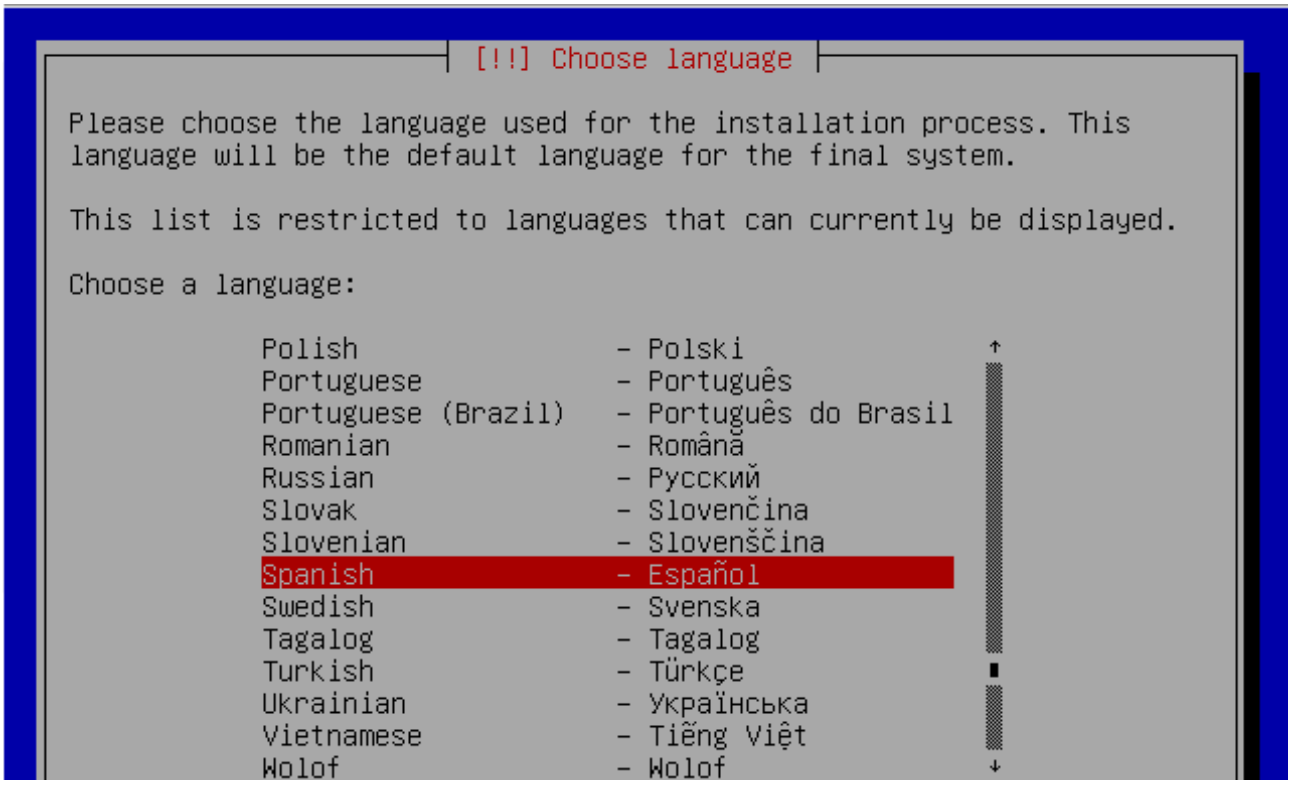
Los dos servidores van a utilizar LINUX de sistema operativo, más concretamente Debian en su versión Etch (4.0). Hemos elegido este sistema y esta versión de debian por varios motivos. Linux, por su seguridad, disponibilidad de buen software para el proyecto, y su licencia GPL (*http://es.wikipedia.org/wiki/GPL*). Hemos escogido debian por su seguridad en el aspecto de servidores. La versión Etch es la llamada en el momento de realización del proyecto *Oldstable*. Hay una nueva versión , lenny, que es la *stable*, pero al final nos hemos decantado por etch.

Esta versión nos la podemos bajar de la pagina web de debian, en su versión CD de instalación: http://www.debian.org/releases/etch/ . Posteriormente grabamos la ISO en un CD y lo introducimos en el lector del servidor y procedemos a arrancar.

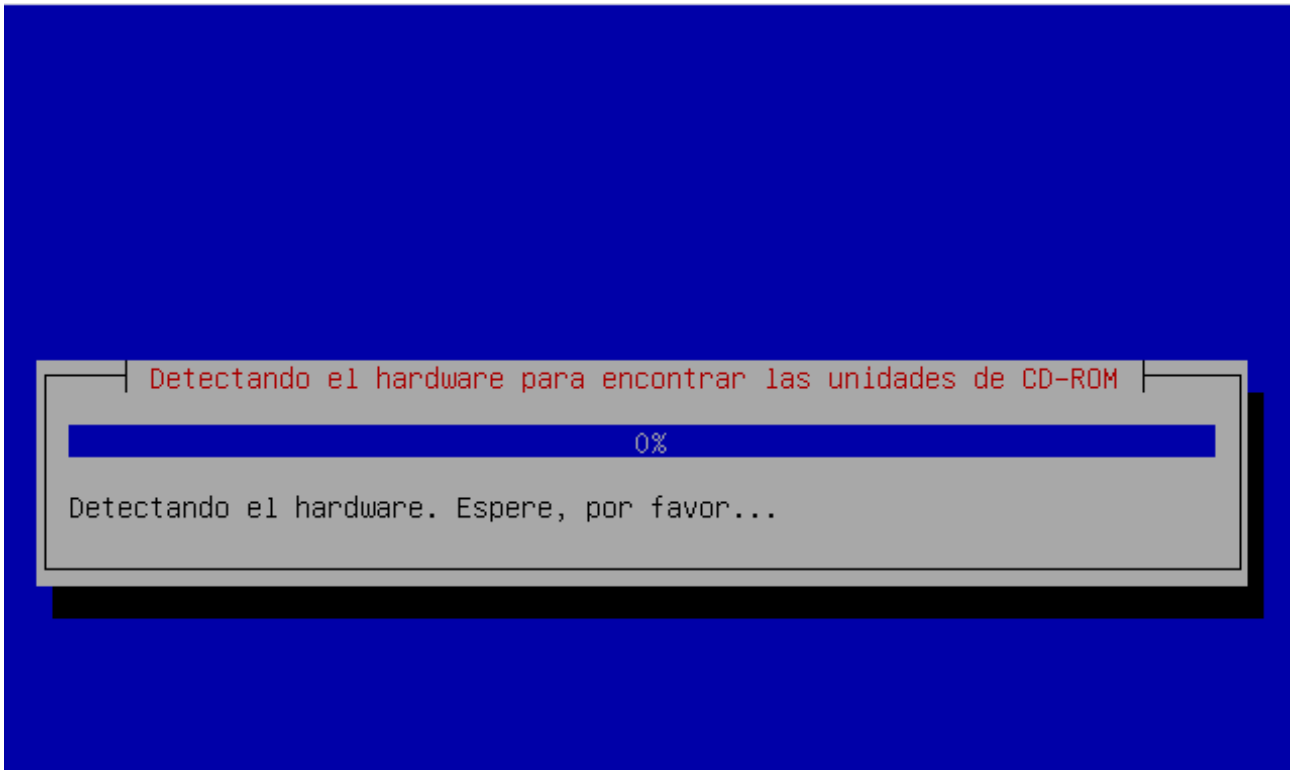
Para que arranque desde el CD hay que retocar el orden de arranque del sistema. Esto se hace entrando en la BIOS. En el caso de Corman, pulsando la tecla DEL , en el caso de Rabius, pulsando F12. Una vez configurada esta opción, comenzamos con la instalación.

En esta pantalla pulsamos ENTER y accedemos a la instalación

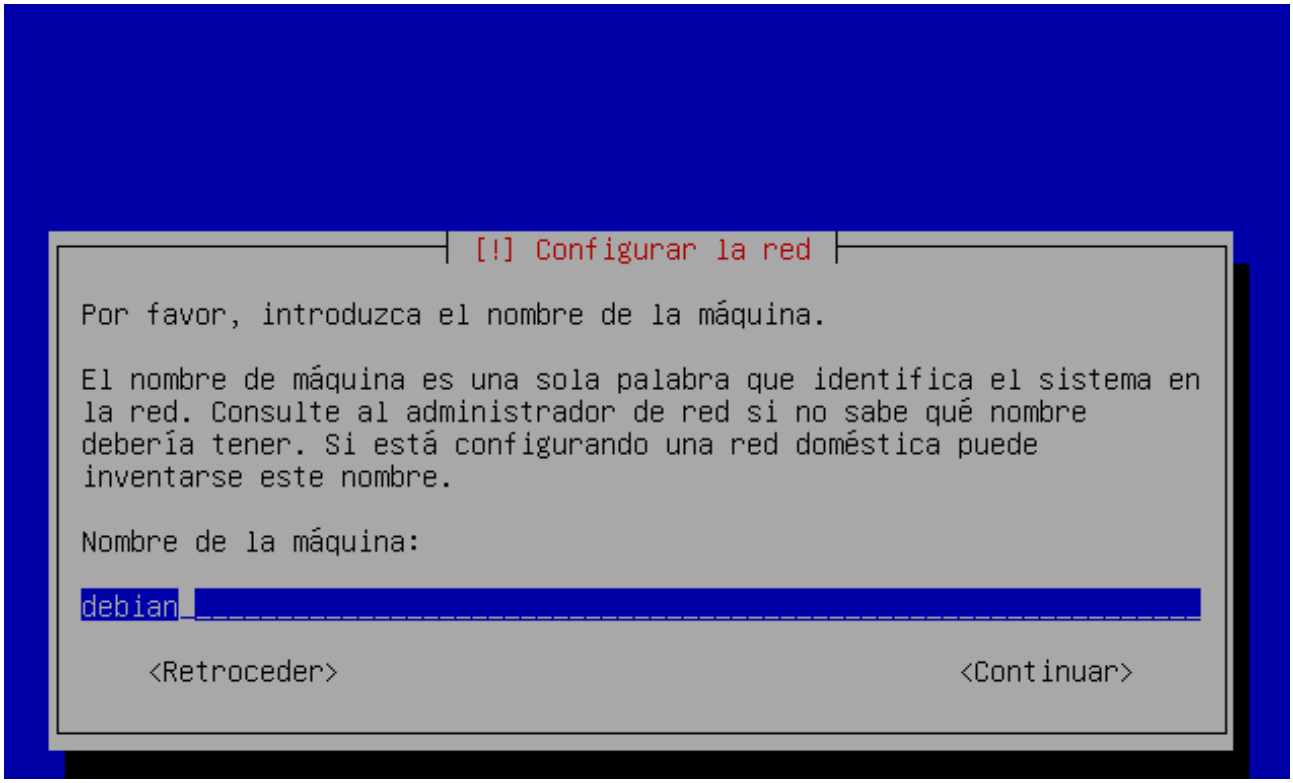
En esta pantalla, seleccionamos nuestro idioma y pulsamos ENTER.



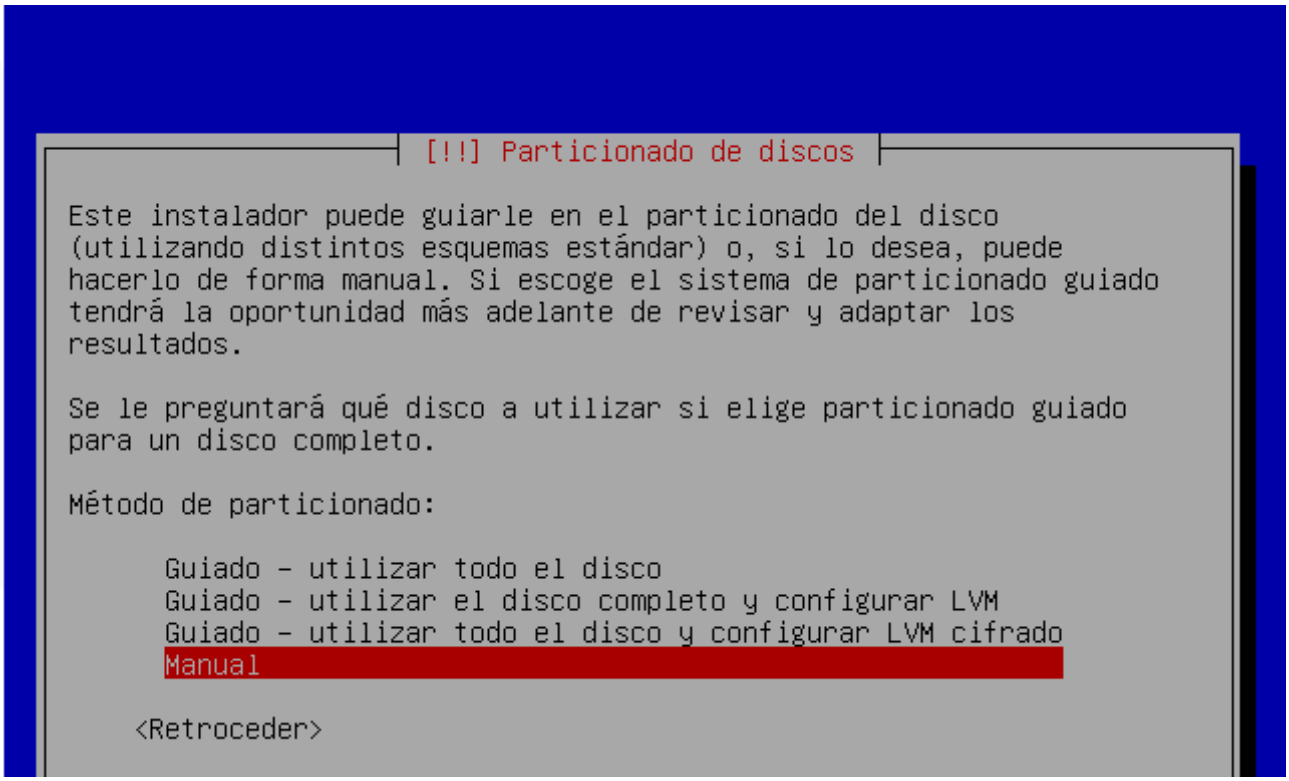
Esperamos a que encuentre todo el hardware del equipo. Durante la instalación, los servidores solo disponían de un disco duro, el segundo disco duro lo añadimos durante la configuración de DRBD.



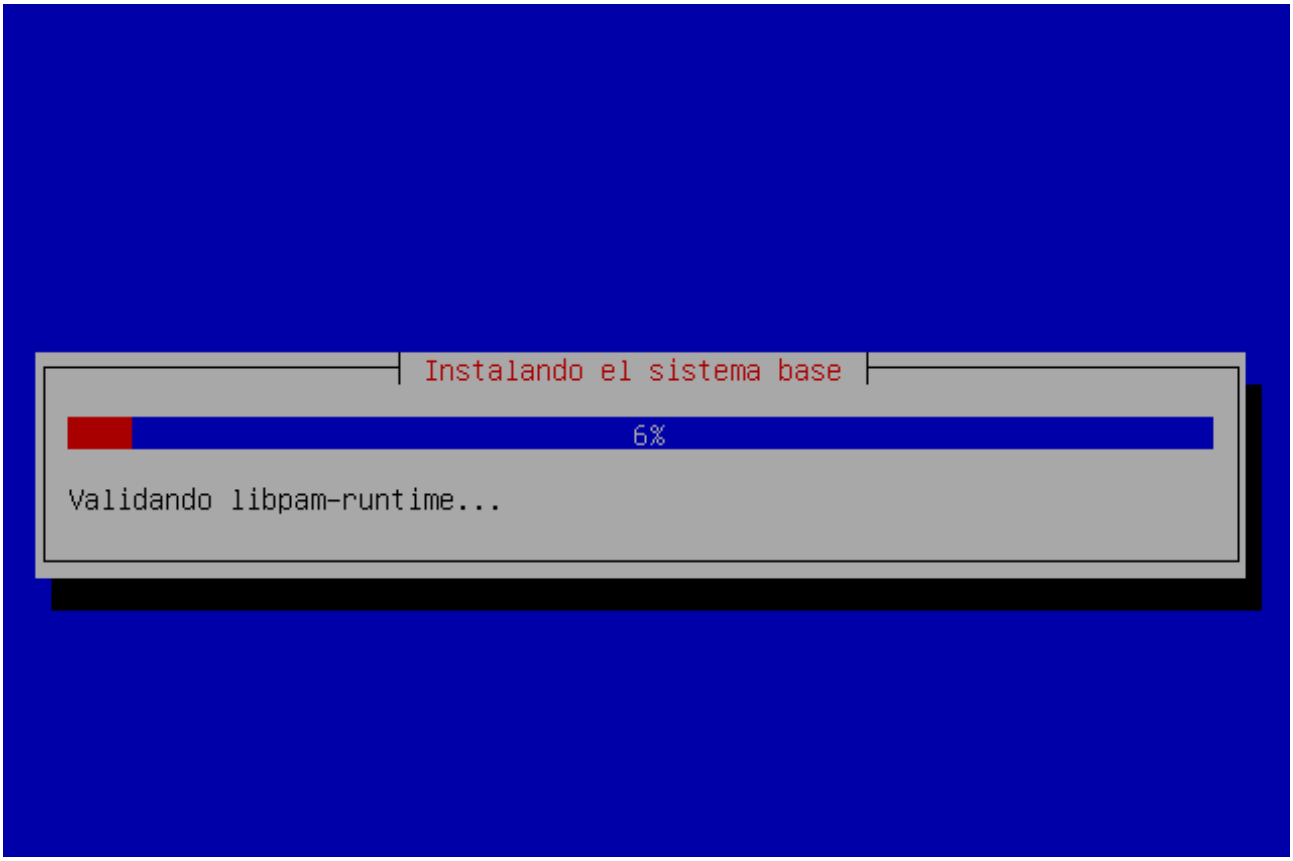
Elegimos el nombre de la máquina, Corman primero y Rabius después.



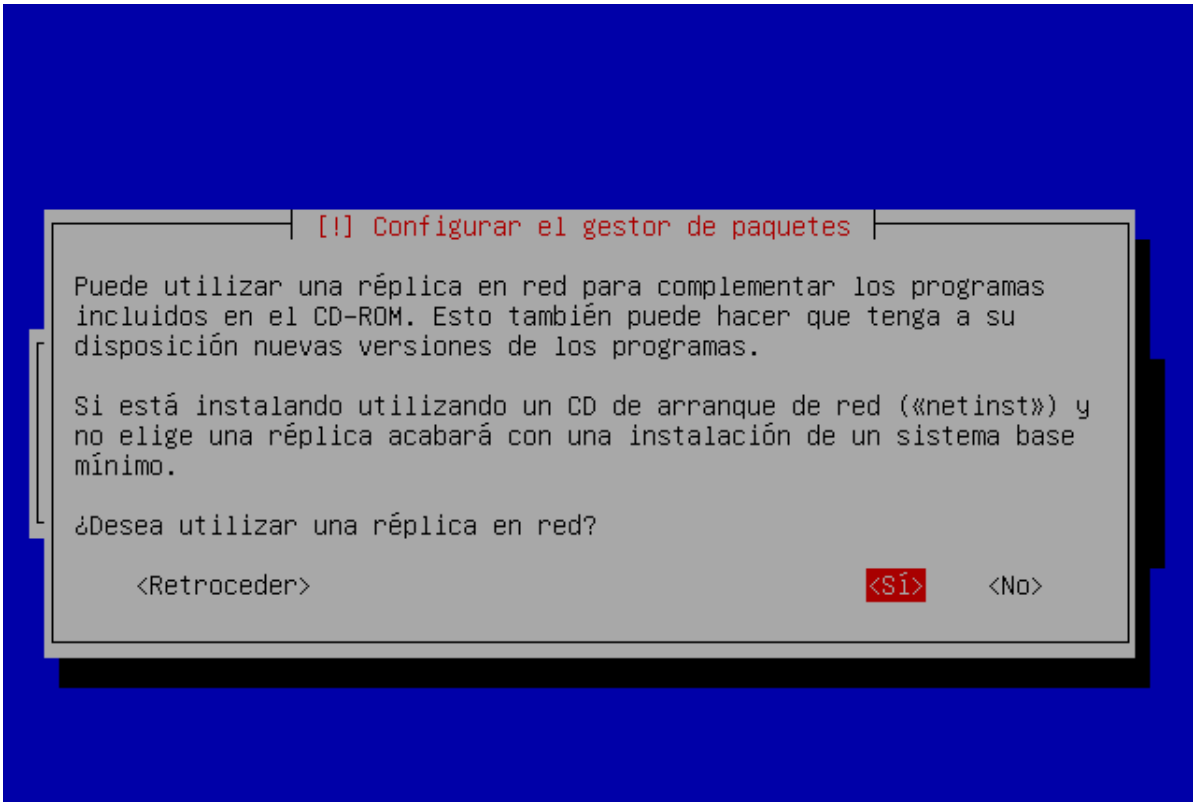
En esta opción, al no saber como se utilizaba DRBD escogimos la opción de Guiado – utilizar todo el disco. De haber conocido el funcionamiento del DRBD hubiera hecho más particiones.



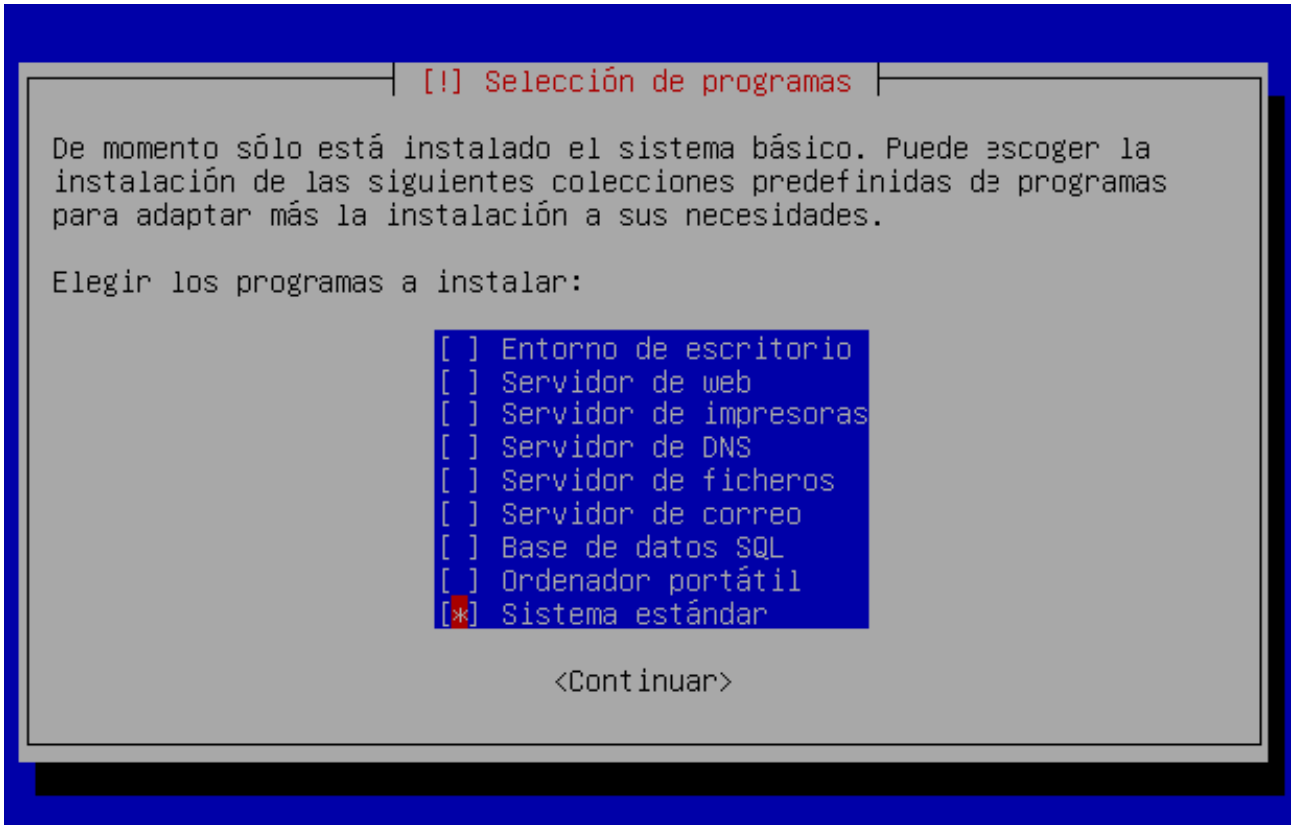
Esperamos que instale el sistema base



Ahora, no utilizaremos una replica en red

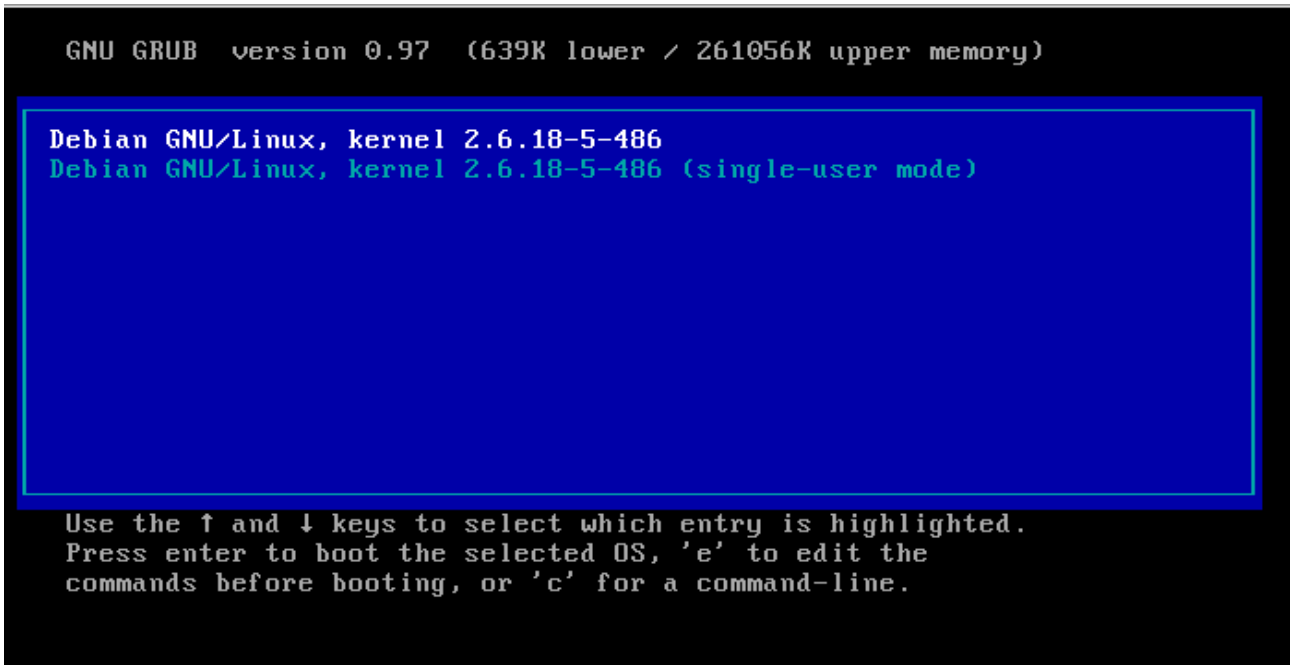


Para el servidor solo necesitamos montar el sistema estándar. Aunque te de la opción de implementar ya el servidor web, no escogemos esa opción. El entorno de escritorio no es necesario, ya que manejaremos las máquinas de forma remota



Esperamos a que lo instale, y luego elegimos que SÍ instale el GRUB (gestor de arranque)

Una vez hecho esto, ya podemos acceder a nuestra máquina y ver como es el GRUB



3.2 – Configuración de red

La configuración de red en debian se hace a través del archivo /etc/network/interfaces. Este archivo de configuración controla todas las interfaces de red, y tendemos que editarlo. Para ello, y siempre a partir de este momento, cada vez que haya que editar un fichero de configuración , nosotros utilizaremos el editor de texto *nano*.

Ejecutamos:

```
# nano /etc/network/interfaces
```

En el archivo escribiremos los detalles de nuestros interfaces de red. El archivo del servidor Corman quedará así :

```
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet static
        address 192.168.2.5
        netmask 255.255.255.0
        broadcast 192.168.2.255
        gateway 192.168.2.1
auto eth2
iface eth2 inet static
```

```
        address 172.168.0.1
        netmask 255.255.255.0
```

Las interfaces de loopback(sirve para trabajar en modo local) eth0 y eth2 están iniciadas automáticamente, definidas por "auto ethX"(donde X es el número). La interfaz eth1existe pero no la tenemos definida, puesto que se utiliza para la segunda salida a Internet. Como de momento no se usa esa salida, no configuraremos esta interfaz, ya que llevaría a problemas de encaminamiento. Eth0 en ambos servidores nos conecta con el router 1 (clase) y eth2 es un cable cruzado entre ellos.

La configuración del mismo archivo para el servidor Rabius sería la siguiente:

```
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet static
        address 192.168.2.20
        netmask 255.255.255.0
        broadcast 192.168.2.255
        gateway 192.168.2.1
auto eth2
iface eth2 inet static
        address 172.168.0.2
        netmask 255.255.255.0
```

Para hacer que esta configuración sea valida, ejecutaremos el comando:

```
# /etc/init.d/networking restart
```

Analizando el comando, lo que estamos haciendo es ejecutar el script situado en /etc/init.d/networking y pasarle el parámetro "restart". Al recibir ese parámetro reiniciara el servicio de red del servidor.

Ahora, vamos a añadir un par de líneas al fichero /etc/hosts . Mediante esta edición, identificaremos los hosts corman y rabius con su IP, de forma que haremos mucho más cómodo el mantenimiento al no tener que escribir la dirección, solamente el nombre.

Ejecutamos:

```
# nano /etc/hosts
```

Y añadiremos las siguientes líneas en ambos servidores:

```
172.168.0.2     rabius
172.168.0.1     corman
```

3.3- Configurando APT

APT viene del ingles **A**dvanced **P**ackaging **T**ool, que podría traducirse al español como herramienta avanzada de gestión de paquetes. Pero primero:

¿Qué es un paquete?

- Un paquete es un archivo o un grupo de archivos que son necesarios tanto para la ejecución de un programa o como para agregar características a un programa ya instalado.
  (fuente: http://www.alegsa.com.ar/Dic/paquete%20de%20software.php)

APT es una herramienta que se encarga de la descarga y gestión de estos paquetes en debian. Su utilización básica es sencilla, pero a niveles avanzados es una utilidad compleja y profunda. Para empezar, debemos saber que APT trabaja con repositorios. Un repositorio es un lugar centralizado donde se archivan los paquetes con los que trabajan herramientas como APT. Lo primero que tenemos que hacer es editar el archivo de configuración de apt en el cual se encuentran los repositorios. Para ello, y siempre a partir de este momento, cada vez que haya que editar un fichero de configuración , nosotros utilizaremos *nano*.

Desde la consola, y como root, escribimos:

```
# nano /etc/apt/sources.list
```

La ruta del archivo es /etc/apt/sources.list. La mayoría de los archivos de configuración instalados desde paquetes en debian se encontrarán en /etc/elnombredelpaquete/archivo.conf. Una vez abierto, nos mostrará los repositorios que contiene nuestro APT. La primera línea que sale es la del CDROM, puesto que lo utiliza como repositorio, comentamos la línea añadiendo " # " . Al comentarla, el programa no leerá la línea, así que es como borrarla, pero pudiendo conservarla.

Comentamos las líneas del CD-ROM, y añadimos las siguientes debajo:

```
deb http://http.us.debian.org/debian etch main contrib non-free
deb-src http://http.us.debian.org/debian etch main contrib non-free
```

Estos son los repositorios oficiales de debian, una vez terminado esto, si estamos utilizando el nano pulsaremos CTRL+O para guardar, y CTRL+X para salir. Después, hay que decirle a APT que actualice su lista de paquetes, esto se hace mediante la instrucción:

```
# apt-get update
```

Una vez actualizada la lista de paquetes haremos una actualización de todos los paquetes instalados, después empezaremos a descargar los paquetes necesarios para el servidor. Para actualizar los paquetes instalados ejecutamos:

```
# apt-get ugrade
```

3.4- Instalando paquetes necesarios

Lo primero que vamos a instalar es el servidor de SSH. SSH **S**ecure **SH**ell es el nombre de un protocolo y un programa que lo implementa que permite acceder a máquinas remotas a través de la red de forma segura. Permite la conexión cifrada, de manera que nadie pueda leer nuestras contraseñas o datos privados, aparte, permite también la copia de archivos de esta manera mediante SCP.

Para instalar el servidor de SSH simplemente ejecutamos en la consola

```
# apt-get install openssh-server
```

Automáticamente el servidor de SSH se instala y se lanza. No hace falta tocar nada en los archivos de configuración, directamente funciona. Aún así, estos archivos están en: /etc/ssh/ssh_config para el cliente y /etc/ssh/sshd_config para el servidor.

A partir de ahora, para conectarnos desde una máquina de la red usaremos la instrucción "ssh 'nombredeusuario@' nombredehost'". Si hacemos desde nuestra maquina para monitorizar, que se encuentra en la red 192.168.1.0 "ssh root@192.168.1.251" nos conectará directamente con Corman, ya que hicimos NAT al router para que trabajara así.

Lo siguiente a instalar es el servidor web apache y el php. Para instalarlo, utilizamos

```
# apt-get install apache2 php5
```

En una sola orden metemos los dos paquete, y APT los instala y configura. Desde nuestro navegador, ponemos la dirección del router 1 (IP: 192.168.1.251) y nos mostrará el mensaje por defecto de apache:

It Works!

Ahora, para nuestras pruebas, editamos la página por defecto de apache, para ello, en cada servidor hacemos

```
# nano /etc/apache2/apache-default/index.html
```

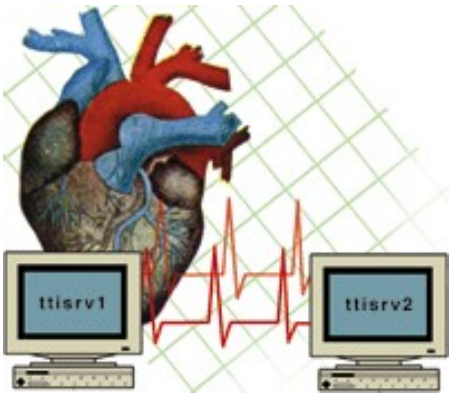
Simplemente borraremos el "It Works" por un "It Working Corman!", o en el caso de Rabius, "It Working Rabius". De esta forma, sabremos con facilidad cual de los dos nodos es el que esta mostrando la página web.

Con esto se ha terminado por ahora con la instalación de paquetes, volveremos a hacer uso de apt-get para instalar posteriormente los paquetes necesarios para DRBD y heartbeat.

4.- Heartbeat

4.1- Que es y como funciona Heartbeat

Traducido del inglés significa *Latido del corazón*



Heartbeat es un software encargado de comprobar continuamente que existe comunicación entre equipos de una red y actuar en caso de fallo. Se utiliza para servidores de alta disponibilidad, monitorizando la conexión entre el nodo primario y el secundario. Si el nodo principal falla, heartbeat lanza una serie de scripts y recursos para que el servicio compartido por el cluster siga siendo ofrecido.

Cada cierto lapso de tiempo(esta variable es definible en el archivo de configuración, como veremos posteriormente) envía un paquete entre el nodo primario y el secundario. Si no recibe respuesta, y hace comprobación de que no esta funcionando, se encarga de hacer que el segundo nodo funcione. Lee en un archivo de configuración donde se incluyen los servicios y recursos(*resources*) y los ejecuta. Entonces, el latido del heartbeat sigue funcionando para comprobar si el nodo primario vuelve a funcionar. Si este se levanta, heartbeat devuelve los recursos al nodo principal y devuelve todo a la normalidad.

Para instalar heartbeat escribimos:

```
# apt-get install heartbeat
```

Esto instalara heartbeat, pero el proceso no arrancará porque no existen los archivos de configuración.

4.2- Archivos de configuración

Heartbeat tiene tres archivos de configuración básicos:

ha.cf :

- Este es el archivo de configuración principal donde definimos las variables principales del servicio.

haresources:

- En este archivo definiremos los servicios que tienen que tener alta disponibilidad (en nuestro caso apache)

authkeys:

- Este archivo contiene una clave que sera la que utilizara heartbeat para comprobar que el otro nodo es realmente el, y nadie lo suplanta.

Los archivos de configuración han de ser **idénticos** en los dos servidores, estos, se encuentran en /etc/ha.d/. También se puede acceder a este directorio desde un enlace que hay en /etc/heartbeat

Abrimos el primero de ellos mediante:

```
# nano /etc/ha.d/ha.cf
```

ha.cf

```
logfacility daemon                          # El demonio que heartbeat usara
                                            # para escribir los logs será
                                            # "daemon"
                                            #
logfile /var/log/ha-log                     # Indicamos el archivo donde se
                                            # escribirá el log
                                            #
node corman rabius                          # La lista de los nodos del
                                            # servidor, en orden, primero el
                                            # primario,después el secundario.
                                            #
keepalive 2                                 # Tiempo entre latidos.
                                            #
warntime  6                                 # Espera una vez que los latidos
                                            # no llegan
                                            #
deadtime 10                                 # Declara el nodo muerto una vez
                                            # pasados 10 segundos desde que
                                            # acabó el tiempo de espera
                                            #
bcast eth0 eth2                             # Lanza los latidos del heartbeat
                                            # por medio de broadcast a través
                                            # de las interfaces eth0 y eth2
                                            #
ping 192.168.1.1                            # Hacemos ping a la puerta de
                                            # enlace para comprobar
                                            # conectividad.
auto_failback yes                           # Devuelve los valores al nodo
                                            # principal cuando se restaure
                                            #
respawn hacluster /usr/lib/heartbeat/ipfail     # Script que se usará en
                                            # caso de fallo de la red
```

haresources

```
# nano /etc/ha.d/haresources
```

Este archivo contiene el servicio o servicios a respaldar.

Es una sola línea. Aquí es donde hacemos que el servidor sea activo/pasivo. Si quisiésemos hacer un balanceo de carga, habría que respaldar ambos servidores, en ese caso habría que añadir otra linea debajo con el grupo de servicios del servidor Rabius.

Este archivo ira cambiando a lo largo del documento, pero a este punto, es así. El contenido de este archivo es:

```
#archivo de configuración haresources
corman  IPaddr2::192.168.2.10/24/eth0:0 apache2
```

Decimos que el nodo que comparte los recursos es Corman, y le damos la siguientes órdenes: con IPaddr2 le decimos que tiene que compartir arrancar una interfaz de red, con :: le damos parámetros al script IPaddrr2. Los parámetros dados son la IP, la máscara y la interfaz a usar. Esta será la eth0:0 . IPaddrr2 es una aplicación que se encuentra dentro de los recursos incluidos por heartbeat. Estos están en /etc/ha.d/resource.d/ . También le pasamos la orden de que ejecute el servicio apache2, que lo buscará en /etc/init.d/apache2 automáticamente. Este recurso compartido es la interfaz de red virtual de la que hablábamos en el esquema, es la IP que tiene hecho NAT en el router.

authkeys

```
# nano /etc/ha.d/authkeys
```

Este archivo es simple, le decimos que use la clave 1, y en la clave 1 definimos el método de encriptación “sha” y la clave será “clavesecreta”

```
#archivo de configuración authkeys
auth 1
1 sha1 clavesecreta
```

Ahora, enviaremos los archivos de configuración mediante SCP(copia segura sobre protocolo SSH) desde Corman hacia Rabius.

```
# cd /etc/ha.d/
# scp ha.cf haresources authkeys rabius:/etc/ha.d/
```

Nos pedirá la contraseña de root, y copiará los tres archivos. Después, en ambas máquinas reiniciaremos el servicio mediante:

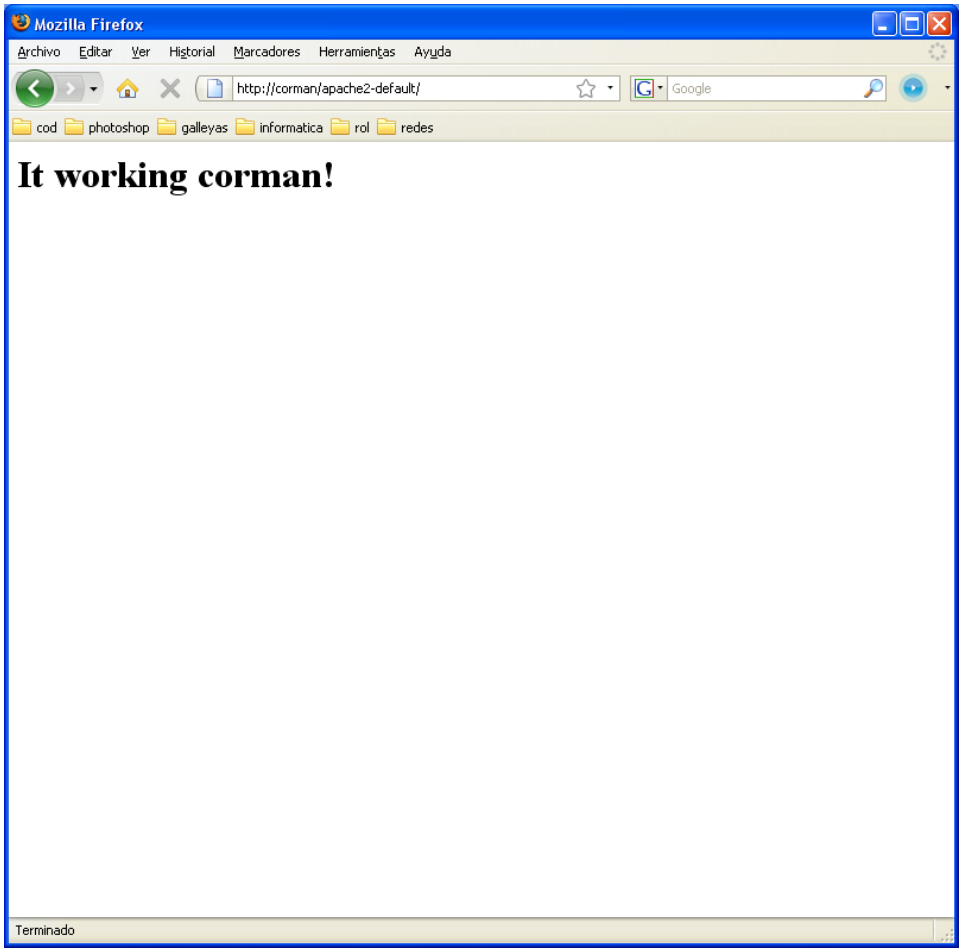
```
# /etc/init.d/heartbeat restart
```

4.3 – Comprobando funcionamiento

Ahora vamos a comprobar que todo lo que hemos hecho funciona. Para simplificar las cosas, desde el PC con el que trabajemos añadiremos al /etc/hosts si se trata de Linux, y si te trata de Windows editaremos el archivo que se encuentra en la ruta C:\WINDOWS\system32\drivers\etc . Sea cual sea, añadiremos la siguiente línea que identificará el servidor con la dirección del ROUTER1:

192.168.1.251 corman

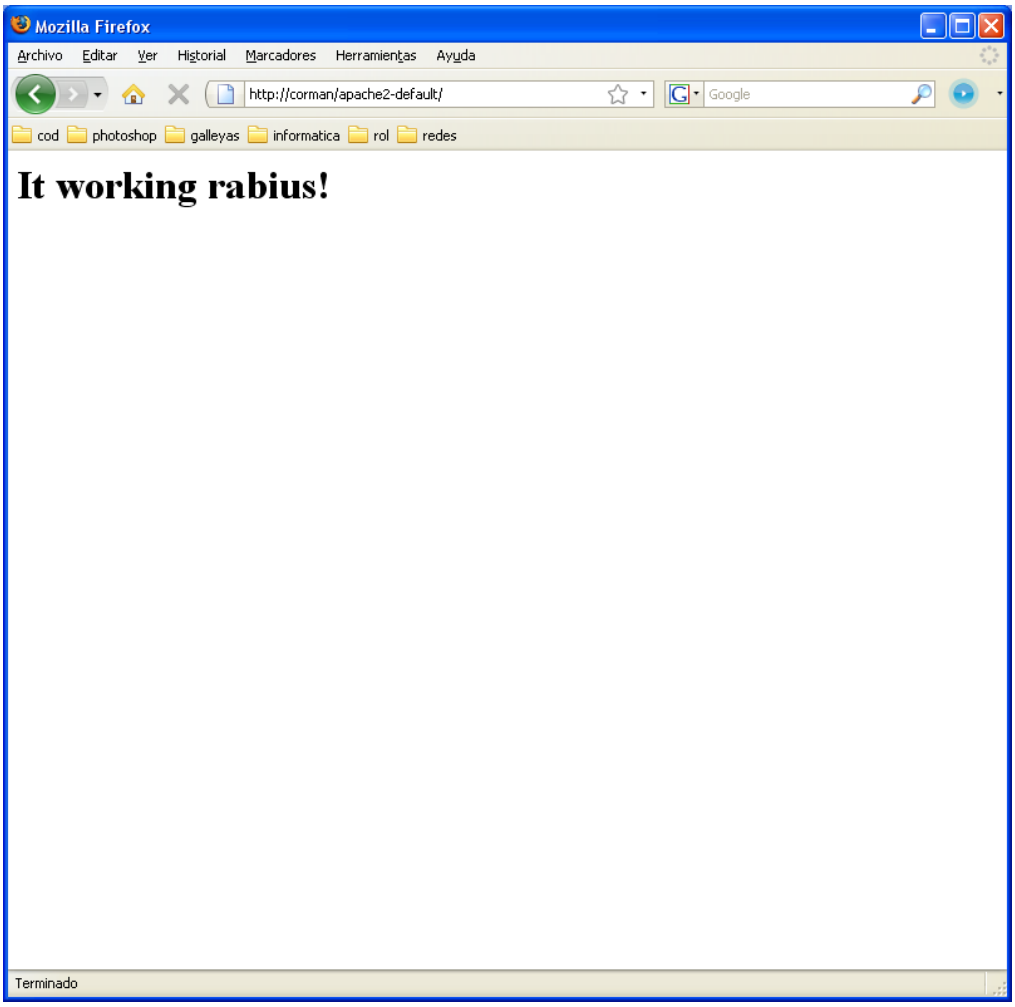
Ahora, escribimos corman en nuestro navegador para comprobar el funcionamiento



Ahora vamos a comprobar que heartbeat funciona. Primero, abriremos una consola en rabius e introduciremos el siguiente comando:

```
# tail -f /var/log/ha-log
```

Con esto, veremos las líneas que se vayan escribiendo en este archivo de log sobre la marcha. Ya estamos listos, así que procedemos a tirar del cable de corriente del servidor Corman, para comprobar el funcionamiento de heartbeat. Esperamos unos segundos y recargamos la página



Como antes retocamos los index.html vemos claramente que a la dirección de Corman esta respondiendo Rabius.

Vemos la salida del log, que comentaremos línea a línea:

```
heartbeat: 2009/06/18_05:55:46 WARN: node corman: is dead
```

Tras varios intentos, el nodo corman no responde al ping

```
heartbeat: 2009/06/18_05:55:46 WARN: No STONITH device configured.
```

No hay ninguna orden para el protocolo STONITH, que explicaremos después.

```
heartbeat: 2009/06/18_05:55:46 info: Resources being acquired from corman.
heartbeat: 2009/06/18_05:55:46 info: Link corman:eth0 dead.
heartbeat: 2009/06/18_05:55:46 info: Link corman:eth2 dead.
heartbeat: 2009/06/18_05:55:46 info: Running /etc/ha.d/rc.d/status status
```

```
heartbeat: 2009/06/18_05:55:46 info: No local resources
[/usr/lib/heartbeat/ResourceManager listkeys rabius] to acquire.
```

En estas líneas comprueba el fichero haresources y los recursos que tiene que servir.

```
heartbeat: 2009/06/18_05:55:46 info: Taking over resource group
IPaddr2::192.168.2.10/24/eth0:0
heartbeat: 2009/06/18_05:55:46 info: Acquiring resource group: corman
IPaddr2::192.168.2.10/24/eth0:0 apache2 pingcheck
```

Ya ha encontrado los recursos necesarios, y se dispone a lanzarlos.

```
heartbeat: 2009/06/18_05:55:46 info: Running /etc/ha.d/resource.d/IPaddr2
192.168.2.10/24/eth0:0 start
heartbeat: 2009/06/18_05:55:46 info: /sbin/ip -f inet addr add 192.168.2.10/24 brd
192.168.2.255 dev eth0 label eth0:0
heartbeat: 2009/06/18_05:55:46 info: /sbin/ip link set eth0 up
heartbeat: 2009/06/18_05:55:46 /usr/lib/heartbeat/send_arp -i 200 -r 5 -p
/var/lib/heartbeat/rsctmp/send_arp/send_arp-192.168.2.10 eth0 192.168.2.10 auto
192.168.2.10 ffffffffffff
heartbeat: 2009/06/18_05:55:46 info: Running /etc/init.d/apache2  start
```

Lanza los servicios del archivo haresources

```
heartbeat: 2009/06/18_05:55:52 info: /usr/lib/heartbeat/mach_down: nice_failback: foreign
resources acquired
heartbeat: 2009/06/18_05:55:52 info: mach_down takeover complete.
heartbeat: 2009/06/18_05:55:52 info: mach_down takeover complete for node corman.
```

Termina de completar el paso, y se convierte en nodo primario.

Para comprobar la configuración de red de Rabius en este punto, ejecutamos lo siguiente:

```
# ifconfig
```

La salida nos muestra las interfaces y sus ips. Destaca la interfaz eth0:0 que es la que ofrece el servicio de IP que le ha pasado el heartbeat.

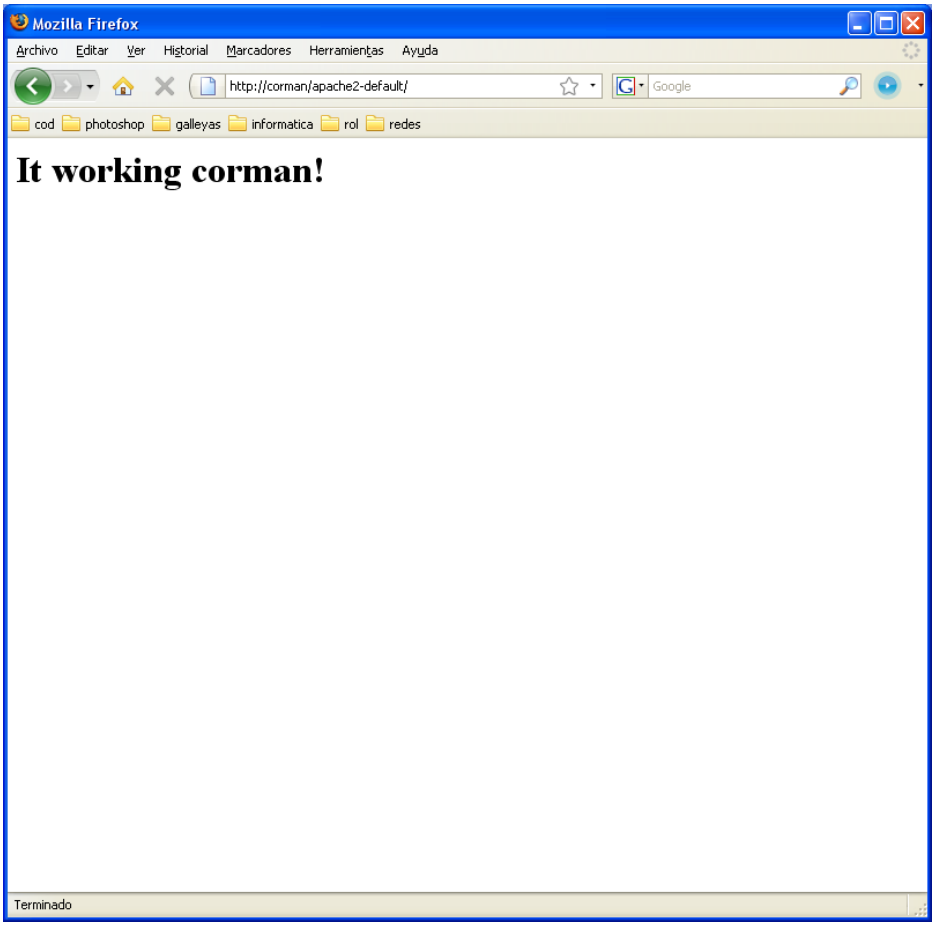
```
eth0      Link encap:Ethernet  HWaddr 00:22:F7:15:54:B8
          inet addr:192.168.1.20  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::222:f7ff:fe15:54b8/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2499 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2360 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:474304 (463.1 KiB)  TX bytes:498916 (487.2 KiB)
          Interrupt:177 Base address:0x1000

eth0:0    Link encap:Ethernet  HWaddr 00:22:F7:15:54:B8
          inet addr:192.168.1.10  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          Interrupt:177 Base address:0x1000

eth2      Link encap:Ethernet  HWaddr 00:80:5A:68:15:27
          inet addr:172.168.0.2  Bcast:172.168.0.255  Mask:255.255.255.0
```

```
inet6 addr: fe80::280:5aff:fe68:1527/64 Scope:Link
UP BROADCAST MULTICAST  MTU:1500  Metric:1
RX packets:2245 errors:0 dropped:0 overruns:0 frame:0
TX packets:2315 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:529569 (517.1 KiB)  TX bytes:544199 (531.4 KiB)
Interrupt:193 Base address:0x1800
```

Ahora, probamos a enchufar el cable de corriente a Corman, y esperar un par de minutos, cargamos de nuevo la página y muestra lo siguiente:



Volvemos a revisar la salida de los logs que dejamos antes funcionando y ahora muestran:

```
heartbeat: 2009/06/18_05:59:07 info: Heartbeat restart on node corman
heartbeat: 2009/06/18_05:59:07 info: Link corman:eth0 up.
heartbeat: 2009/06/18_05:59:07 info: Status update for node corman: status up
heartbeat: 2009/06/18_05:59:07 info: Link corman:eth2 up.
```

Heartbeat ha detectado que Corman vuelve a funcionar

```
heartbeat: 2009/06/18_05:59:07 info: Running /etc/ha.d/rc.d/status status
heartbeat: 2009/06/18_05:59:08 info: Status update for node corman: status active
```

Vuelve a comprobar que Corman está vivo.

```
heartbeat: 2009/06/18_05:59:08 info: remote resource transition completed.
heartbeat: 2009/06/18_05:59:08 info: rabius wants to go standby [foreign]
heartbeat: 2009/06/18_05:59:08 info: standby: corman can take our foreign resources
heartbeat: 2009/06/18_05:59:08 info: give up foreign HA resources (standby).
```

Una vez comprobado, se dispone a apagar los servicios en rabius y encenderlos en corman.

```
heartbeat: 2009/06/18_05:59:08 info: Running /etc/ha.d/rc.d/status status
heartbeat: 2009/06/18_05:59:08 info: Releasing resource group: corman
IPaddr2::192.168.2.10/24/eth0:0 apache2
heartbeat: 2009/06/18_05:59:08 info: Running /etc/init.d/apache2  stop
heartbeat: 2009/06/18_05:59:09 info: Running /etc/ha.d/resource.d/IPaddr2
192.168.2.10/24/eth0:0 stop
heartbeat: 2009/06/18_05:59:09 info: /sbin/ip -f inet addr delete 192.168.2.10 dev eth0
heartbeat: 2009/06/18_05:59:09 info: /sbin/ip -o -f inet addr show eth0
heartbeat: 2009/06/18_05:59:09 info: IP Address 192.168.2.10 released
heartbeat: 2009/06/18_05:59:09 info: foreign HA resource release completed (standby).
heartbeat: 2009/06/18_05:59:09 info: Local standby process completed [foreign].
```

Los recursos han sido totalmente devueltos

```
heartbeat: 2009/06/18_05:59:12 WARN: 1 lost packet(s) for [corman] [11:13]
```

Heartbeat ha detectado la perdida de un paquete, posiblemente por algún problema de red, pero como para declarar a un nodo muerto hacen falta varios segundos, no pasa nada.

```
heartbeat: 2009/06/18_05:59:12 info: remote resource transition completed.
heartbeat: 2009/06/18_05:59:12 info: No pkts missing from corman!
heartbeat: 2009/06/18_05:59:12 info: Other node completed standby takeover of foreign
resources.
```

Esta es nuestra configuración de heartbeat. Tiene más opciones, pero es suficiente con esto para que funcione correctamente. No hemos definido ningún protocolo STONITH(**S**hoot **T**he **O**ther **N**ode **I**n **T**he **H**ead) que sirve para cerciorarse que el otro nodo ha caído. Es un extra que permite implementar heartbeat en su configuración, pero nosotros no lo hemos usado.

5.- Scripts de Control de Conexión

5.1- Planteamiento del problema

A este punto del documento hemos conseguido montar la alta disponibilidad en dos equipos, pero este proyecto va más allá. La intención ahora es ofrecer servicio de alta disponibilidad en salidas a Internet, a través de dos routers. Las interfaces que conectan los servidores con los routers deberán permanecer inactivas, si estas estuvieran activas los equipos dispondrían de dos puertas de enlace para conectarse a la red externa, lo que deriva en problemas de encaminamiento.

Lo primero que hay que hacer a la hora de programar es pensar en el problema. Nuestro problema a solucionar es comprobar que nuestro servidor tenga siempre conexión a Internet y si esta conexión falla automatizar el cambio de salida a internet. Disponemos de dos routers que se encargan de proveernos de este servicio, pero solo podemos usar uno a la vez. La IP interna del router principal (clase) es 192.168.2.1 y la IP externa de este router es la 192.168.1.251. Desde nuestro servidor haremos constantemente a través de un script propio pings de comprobación de red a la IP 192.168.1.1, que es la que nos provee de Internet. En un esquema real con salidas a Internet, este ping debería hacerse primero a la DNS de nuestro proveedor, y luego añadir al mismo script mas comprobaciones, como ping a www.google.com o www.debian.org . Si este ping falla, entonces nuestra conexión primaria de Internet no funciona.

En ese caso, levantaremos la interfaz eth1 del servidor y le añadiremos los datos necesarios tanto de IP como de puerta de enlace. Además, al hacer esto, lanzaremos otro script que compruebe si el router 1 (CLASE) vuelve a tener Internet. Si la conexión vuelve, este script levantar la interfaz eth0 y desconectará la eth1, y devolverá todos los datos de red necesarios. Además, volverá a lanzar el script principal para que vuelva a comprobar si la conexión vuelve a fallar, convirtiéndolo todo en una tarea completamente automatizada.

Para la comprobación después de que han ido bien, vamos a hacer una función de LOG de los scripts . Guardaremos la información de su funcionamiento en un archivo.

Utilizaremos cinco Scripts para esta tarea, estos son:

pingcheck → Encargado de comprobar que la primera salida a Internet funciona.
failcase → El trozo de código que actuará en caso de fallo de la salida primaria a Internet
newdns → fuerza el script de NO-IP con la configuración nueva de IP.
checkoldping → Comprueba si la conexión antigua ha sido restaurada.
backnormal → Si la conexión primaria es restaurada la vuelve a colocar como activa y desactiva la otra.

5.2- Código y explicación de los Scripts

5.2.1- pingcheck

Utilizaremos el nano para escribir estos scripts, desde la consola hacemos:

```
# nano /etc/ha.d/resource.d/pingcheck.sh
```

Situaremos todos los scripts en esta carpeta, pues es la predefinida por heartbeat para los recursos, y pingcheck, será un recurso de heartbeat.

El código es el siguiente:

```
#!/bin/bash                          # este es el inicio de todos los scripts

PINGCOUNT=1                          # aquí definimos unas cuantas variables
PINGMAXWAIT=1                        # de forma que nos será mas fácil
PINGHOST="192.168.1.1"               # identificar luego cada una
WAITTIME=5                           # en el código

while /bin/true                      # hacemos un while con bucle infinito
do                                   # hasta que se le ordene salir
   if ping -c ${PINGCOUNT} -w ${PINGMAXWAIT} ${PINGHOST} 2>&1 >/dev/null
      # en el ping le decimos que lo haga hacia Internet( en la aplicación real,
      #cambiaríamos el valor de pinghost por varios direcciones de Internet como
      # mencionamos arriba, nosotros chequeamos la salida a Internet
      then
      sleep ${WAITTIME} # si funciona, esperamos 5 segundos antes de volver a
                        # chequear
   else
        ping -c 3 ${PINGHOST} 2>&1 >/dev/null # si no funciona volvemos a
                                              # chequear el fallo, pero tres
                                              #veces, para obtener confirmación
        if [ $? -gt 0 ]                       # utilizamos ahora otro método,
                                              # igual de valido que el anterior
                                              # para comprobar si el ping no ha
                                              # tenido éxito
                then    # si devuelve error, significa que la conexión primaria
                        # falla lanzamos un mensaje de error a un archivo de log
```

```
echo "`date` Conexión primaria a Internet fallida, lanzando segunda conexión"
2>&1 >> /etc/cron.mio/ping.log
                        # ahora, nos situamos en la carpeta donde están el resto
                        # de scripts y lanzamos el script failcase.sh
                cd /etc/ha.d/resource.d/
                 sh ./failcase.sh
                 exit # le decimos al programa que se salga del bucle
        fi
   fi
done
```

Una cosa que no está comentada arriba es el 2>&1 >/dev/null . Esto significa que tras ejecutar el comando (de ping en este caso) redireccione tanto errores como notificaciones a la /dev/null/, o lo que viene a ser lo mismo, que no los muestre por ningún lado, queremos que el script sea trasparente a ojos del usuario. Después, con el "echo", hacemos algo parecido. Utilizamos echo "mensaje" >> archivode.log para enviar mediante ">>" lo que salga por el echo a un log.

5.2.2- failcase

Al igual que antes, utilizamos el nano

```
# nano /etc/ha.d/resource.d/failcase.sh
```

El código es:

```
#!/bin/bash

ifconfig eth0 down 2>&1 /dev/null
        echo "`date` desconectando interfaz de red eth0" >> /var/log/ping.log

      # en las líneas de arriba desconectamos la primera interfaz de red y
      # redireccionamos la salida por ningún sitio, para que no se muestre en
      # pantalla nada, para eso, hacemos un archivo de log donde nos mostrará la
      # hora y el mensaje de desconexión. Automáticamente, al hacer esto, se
      # borrará cualquier entrada de la tabla de encaminamiento relacionada con
      # la eth0. Evitando así cualquier problema.

ifconfig eth1 up 192.168.2.10 netmask 255.255.255.0 2>&1 /dev/null
        echo "`date` levantando interfaz de red eth1" >> /var/log/ping.log

      # Volvemos a hacer lo mismo de antes, pero en vez de desconectar la
```

```
      # interfaz levantamos la interafz eth1 y le damos su IP y máscara, además
      # de escribir lo necesario en los archivos de log
route add default gw 192.168.2.2 dev eth1
    echo "`date` seleccionando 192.168.2.2 como nueva puerta de enlace" >>
/var/log/ping.log

      # añadimos a la tabla de encaminamiento como puerta de enlace
      # predeterminada la ip interna del segundo router, pero con el parámetro
      # dev, para que la relacione con eth1, para así poderla desconectar luego
      # facilmente.

cd /etc/ha.d/resource.d/
        ./newdns rafa
        ./checkoldping.sh
      # ejecutamos el script newdns para actualizar el dominio de NO-IP con el
      # parámetro rafa, que explicaremos después, y ejecutamos checkoldping para
      # comprobar si la salida primaria a Internet vuelve a funcionar.
```

Es importante destacar que la IP que levantamos con este script es la 192.168.2.10. Esto es así porque al desconectar la eth0, tiramos también la eth0:0. Siempre que lancemos este script será porque la conexión del router 1 ha fallado, y este script solo se ejecutará en el servidor que este ofreciendo el servicio, por lo tanto levanta esa IP en esa interfaz.

5.2.3- newdns

Para las DNS vamos a utilizar NO-IP. NO-IP es un proveedor de DNS dinámico. Esta parte del proyecto la haremos de forma educativa, porque realmente nuestros routers no apuntan a Internet, si no a la red interna. Por ello, para poder acceder a la web que esta en el servicio deberemos poner en nuestro navegador la IP del router que está sirviendo en ese momento. Para la presentación, se implementará en un sistema de DNS(bind) el cual automáticamente balanceará la carga de la IP de un router a otro, como solo uno de los dos ofrecerá el servicio en cada momento, asociará el nombre con la IP del router que ofrezca servicio.

Para seguir con NO-IP nos registramos en su página web y seguimos las instrucciones para registrar el dominio que queremos, en nuestro caso “elric.no-ip.info”. Después, desde nuestra consola, ejecutamos:

```
# apt-get install no-ip
```

Esto instala el cliente, pero no arrancará porque no tiene configuración, para configurar el cliente y hacer que funcione escribimos:

```
# no-ip -C
```

Nos preguntará que interfaz de red queremos usar para el servicio. Esto supone un problema porque queremos usar dos diferentes y esa opción no existe, la solución es crear dos archivos de configuración. Rellenamos los datos para la segunda conexión, eligiendo la salida a Internet eth1 y los datos de nuestra cuenta, y el intervalo de refresco, en nuestro caso 5 minutos. Ahora, renombramos el archivo de configuración y volvemos a crear otro. El archivo de configuración se crea en /etc/no-ip.conf. Escribimos:

```
#mv /etc/no-ip.conf /etc/no-ip.conf.rafa
```

Ahora, volvemos a crear un archivo de configuración con las instrucciones de arriba pero diciéndole que utilice la interfaz eth0 y arrancamos el servicio con la configuración inicial, simplemente usamos

```
#/etc/init.d/no-ip start
```

Ahora, con la configuración preparada, ejecutamos para escribir el script:

```
      # nano /etc/ha.d/resource.d/newdns.sh
```

```
#!/bin/bash

case "$1" in                              # utilizamos la instrucción "case" que
      clase)                              # recibe un parámetro, si recibe
      /etc/init.d/no-ip stop              # el parámetro "clase" para el
                                          # servicio y lo relanza con la
                                          # configuración predeterminada para
      /usr/bin/no-ip -c /etc/no-ip.conf   # el router clase

        ;;
        rafa)                             # si el parámetro que recibe es rafa
        /etc/init.d/no-ip stop            # para el servicio y lo lanza con la
        /usr/bin/no-ip -c /etc/no-ip.conf.rafa # configuración para el router
        ;;                                # rafa
esac
exit 0                                    # cerramos la instrucción y salimos de
                                          # el programa.
```

5.2.4- checkoldping

Este script es el encargado de comprobar si la conexión primaria a Internet a través del router clase vuelve a la vida. Es muy parecido al script pingcheck pero cambiando los valores. Escribimos:

```
# nano /etc/ha.d/resource.d/checkoldping.sh
```

El código de este script es:

```
#!/bin/bash

PINGCOUNT=1                          # Al igual que antes definimos las
                                     # variables. Ahora destacamos
PINGMAXWAIT=1                        # que el host al cual hacemos ping es la ip
                                     # pública del router
PINGHOST="192.168.1.251"             # de esta forma comprobamos que vuelve a
                                     # tener Internet. para esto, la ip publica
WAITTIME=60                          # tiene que ser estática.  el tiempo de
                                     # espera es 60, aunque esa variable cambiará
while /bin/true                      # para la presentación, pero en el proyecto
                                     # real no tiene que ser un ping tan
do                                   # constante como en otros casos
   if ping -c ${PINGCOUNT} -w ${PINGMAXWAIT} ${PINGHOST} 2>&1 >/dev/null
   then
        cd /etc/ha.d/resource.d/     # volvemos a hacer un bucle infinito, y
                                     # ahora, si devuelve positivo es que
                ./backnormal.sh      # la conexión primaria vuelve a funcionar,
                                     # con lo que lanzamos el script
        exit                         # backnormal.sh que se encarga de restaurar
                                     # las conexiones.
   else
        echo "`date` Conexion primaria a Internet caída, utilizando segunda
conexión" >> /var/log/ping.log
        sleep ${WAITTIME}            # si el ping falla, lo escribimos en el
                                     # archivo de log de nuestros scripts
   fi
done
```

5.2.5- backnormal

Este es el último de nuestros scripts, se encarga de devolver a la normalidad las conexiones cuando se restaura la conexión a Internet del primer router y a reactivar las comprobaciones originales. Al reiniciar los servicios de red volverá a funcionar el eth0:0 y así podrá seguir funcionando el servicio web.

```
#!/bin/bash

ifconfig eth1 down            # el script es bastante sencillo, desconectamos la
```

```
cd /etc/init.d/                 # interfaz eth1 que apunta al router 2
./networking restart            # y ejecutamos el reinicio de las conexiones de
                                # red al estado en el que se encuentran en el
                                # archivo /etc/network/interfaces

echo "`date` volviendo a la normalidad" >> /var/log/ping.log
                                # registramos la actividad en nuestro log
cd /etc/ha.d/resource.d/        # lanzamos newdns con parámetro clase
 ./newdns clase                 # y relanzamos de nuevo el script pingcheck que
 ./pingcheck.sh                 # comprueba si la conexión deja de funcionar.
```

5.3- Implementación de los scripts en heartbeat

Lo primero para conseguir que los scripts funcionen es darles permisos de ejecución. En linux, hay tres tipos de permisos: Escritura, Lectura y Ejecución. Al ser un programa, le añadiremos los de ejecución. Nos situamos en la carpeta de resources de heartbeat mediante:

```
# cd /etc/ha.d/resource.d/
```

Y le damos los permisos de ejecución a todos los archivos que acaben en .sh

```
# chmod +x *.sh
```

Copiamos los scripts al servidor Rabius y le damos los permisos allí también. Como seguimos en la carpeta resources ejecutamos:

```
# scp *.sh rabius:/etc/ha.d/resource.d/
```

Y copiará todos los archivos, allí, volvemos a ejecutar la orden de dar permisos y listo. Ahora, necesitamos copiar para que el archivo haresources y el propio heartbeat reconozcan nuestros scripts como un recurso a compartir en /usr/sbin/.

```
#mv /etc/ha.d/resource.d/pingcheck.sh /usr/sbin/pingcheck
```

Hacemos lo mismo en las dos máquinas y le damos permisos de ejecución al archivo en /usr /sbin/. Para funcionar, heartbeat nos pide que pingcheck este incluido en el arranque del sistema y tenga un script de arranque y parada del demonio en /etc/init.d/. Así que nos disponemos a escribir este script.

```
#nano /etc/init.d/pingcheck
```

El contenido de este script es:

```
#!/bin/sh

# /etc/init.d/pingcheck
```

```
# que hacer cuando salga cada caso


case "$1" in
        start)                # en caso de start, mostramos un mensaje por
                              # pantalla y utilizamos el comando
                              # start-stop-daemon para lanzar nuestro script.
                              # es importante el uso del parámetro –background,
                              # o el sistema no arrancará
        echo "iniciando servicio de comprobación de ping"
       start-stop-daemon --start --background --exec /sbin/pingcheck
        ;;
        stop)
        echo "parando servicio de comprobacio de ping"     # en caso de que el
                                                           # parámetro recibido
        start-stop-daemon --stop --exec /sbin/pingcheck    # sea stop, paramos
        ;;                                                 # el servicio
        *)
        echo "usar: start|stop"     # si no nos dan parámetros, mostramos un
        exit 1                      # mensaje de como se debe usar el script.
              ;;
esac
exit 0
```

Ahora le damos permisos de ejecución a este script

```
#chmod +x /etc/init.d/pingcheck
```

Ahora, queda añadirlo al arranque del sistema. Esto se hace mediante la orden update-rc.d. Ejecutamos **solo** en Corman:

```
#update-rc.d pingcheck defaults
```

Y el script quedará añadido al arranque. Es importante decir, que en el segundo servidor, en Rabius, el script de /etc/init.d/pingcheck debe estar, pero no se debe ejecutar desde el inicio al igual que apache, será heartbeat el que diga cuando se ejecuta. Para ello, volvemos a editar el fichero de servicios de heartbeat, haresources y añadimos el pingcheck a los servicios.

```
#nano /etc/ha.d/haresources
```

Y la línea quedaría:

```
corman  IPaddr2::192.168.2.10/24/eth0:0 apache2 pingcheck
```

5.4- Comprobando funcionamiento de Scripts

Ahora vamos a probar que todo lo que hemos hecho funciona correctamente. Reiniciamos los equipos, y durante el arranque, si estamos atentos podemos ver el mensaje "iniciando servicio de comprobación de ping". Este mensaje es lanzado por el script que esta en /etc/init.d/pingcheck. Esperamos a que ambos ordenadores terminen de arrancar comprobamos que desde Internet se ve la página. Entonces, escribimos en una consola:

```
# tail -f /var/log/ping.log
```

Y desconectamos el cable de red que surte de Internet al router 1, o en nuestro caso, el que lo conecta a la red de mi casa.. El resultado del log es el siguiente:

```
vie jun 19 18:00:18 CEST 2009 desconectando interfaz de red eth0
Fri Jun 19 18:00:18 CEST 2009 levantando interfaz de red eth1
Fri Jun 19 18:00:18 CEST 2009 seleccionando 192.168.2.2 como nueva puerta de enlace
Fri Jun 19 18:00:19 CEST 2009 Conexión primaria a Internet caída, utilizando segunda
conexion
Fri Jun 19 18:00:29 CEST 2009 Conexión primaria a Internet caída, utilizando segunda
conexion
```

Abrimos otra terminal pulsando ctrl+alt+f2, nos logueamos y escribimos ifconfig para ver la configuración actual de red, que es la siguiente:

```
eth1      Link encap:Ethernet  HWaddr 00:50:BF:5D:37:0D
          inet addr:192.168.2.10  Bcast:192.168.2.255  Mask:255.255.255.0
          inet6 addr: fe80::250:bfff:fe5d:370d/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:53 errors:0 dropped:0 overruns:0 frame:0
          TX packets:107 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:5289 (5.1 KiB)  TX bytes:9478 (9.2 KiB)
          Interrupt:11 Base address:0xec00


eth2      Link encap:Ethernet  HWaddr 00:E0:7D:C5:B2:24
          inet addr:172.168.0.1  Bcast:172.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::2e0:7dff:fec5:b224/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:10787 errors:0 dropped:0 overruns:0 frame:0
          TX packets:10382 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1252255 (1.1 MiB)  TX bytes:1695814 (1.6 MiB)
          Interrupt:12 Base address:0xe800
```

También queremos ver la tabla de enrutamiento, que se ve escribiendo el comando

```
# route -n
```

Y nos muestra:

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.2.0     0.0.0.0         255.255.255.0   U     0      0        0 eth1
172.168.0.0     0.0.0.0         255.255.255.0   U     0      0        0 eth2
0.0.0.0         192.168.2.2     0.0.0.0         UG    0      0        0 eth1
```

Las dos salidas de los archivos nos muestran que el script funciona perfectamente, la puerta de enlace es el router 2 y las interfaces de red estan bien configuradasAhora volvemos al terminal 1 pulsando ctrl+alt+f1 y conectamos el cable de red. Entonces, el log que estábamos monitorizando mediante el comando tail nos dirá:

```
Fri Jun 19 18:00:34 CEST 2009 Conexión principal restaurada, volviendo a la normalidad
Fri Jun 19 18:00:36 CEST 2009 Conexión principal restaurada, volviendo a la normalidad
```

Terminamos el comando pulsando ctrl+c y volvemos a comprobar que muestra la salida de ifconfig:

```
eth0      Link encap:Ethernet  HWaddr 00:01:02:12:4D:81
          inet addr:192.168.2.5  Bcast:192.168.2.255  Mask:255.255.255.0
          inet6 addr: fe80::201:2ff:fe12:4d81/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:10238 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8497 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1271458 (1.2 MiB)  TX bytes:910519 (889.1 KiB)
          Interrupt:10 Base address:0xa100

eth0:0    Link encap:Ethernet  HWaddr 00:01:02:12:4D:81
          inet addr:192.168.2.10  Bcast:192.168.2.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          Interrupt:10 Base address:0xa100

eth2      Link encap:Ethernet  HWaddr 00:E0:7D:C5:B2:24
          inet addr:172.168.0.1  Bcast:172.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::2e0:7dff:fec5:b224/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:10974 errors:0 dropped:0 overruns:0 frame:0
          TX packets:10519 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1270102 (1.2 MiB)  TX bytes:1713768 (1.6 MiB)
          Interrupt:12 Base address:0xe800
```

Y la de la tabla de encaminamiento con route -n otra vez

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.2.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
172.168.0.0     0.0.0.0         255.255.255.0   U     0      0        0 eth2
0.0.0.0         192.168.2.1     0.0.0.0         UG    0      0        0 eth0
```

Como se ha visto, el script levanta la eth1 que tiene la dirección 192.168.2.10, la que ofrece el servicio. Levanta esa IP, porque al desconectar la eth0 automáticamente deja de ofrecer el servicio de ip que tiene en haresources para la eth0:0. Al realizar un /etc/init.d/networking restart reinicia los servicios de red, y como eth0 funciona, eth0:0 vuelve a funcionar.

Ahora, la siguiente prueba consiste en hacer la prueba en Rabius. Para eso, desconectamos el servidor Corman y Rabius arrancará. Después, apagamos el router 1 y vemos exactamente la misma salida que hemos puesto antes, Rabius funciona por la segunda salida.

6.-Trabajando con DRBD

6.1- Qué es y como funciona DRBD

DRBD es un sistema distribuido de almacenamiento para Linux. Es un módulo para el Kernel que funciona integrando dos particiones, o dos discos completos de dos equipos distintos. Cada vez que se guarda información en el DRBD esta se escribe en el nodo principal, entonces, cada vez que pasa un lapso de tiempo definido el DRBD del nodo primario propaga la información con el DRBD del nodo secundario que la escribe en el dispositivo real. En resumen, DRBD nos sirve para que la web y los datos que queramos de nuestros dos servidores sea exactamente igual.

Para hacer la sincronización es necesario que DRBD trabaje con un proceso que maneje el cluster de alta disponibilidad, o lo que es lo mismo en nuestro caso, con heartbeat.

6.2- Instalando y configurando DRBD

Lo primero que necesitamos para instalar DRBD son dos particiones preparadas para esto. Al principio de empezar este proyecto yo no se sabía este hecho, así que la instalación de debian fue sobre el disco completo. Para tener que evitar problemas de reparticionado decidí instalar dos discos duros, uno por cada equipo.

Después de la instalación física y de arrancar el servidor, escribimos en la consola:

```
# fdisk -l
```

Esto nos muestra los discos presentes y sus particiones.

```
Disco /dev/hda: 8455 MB, 8455200768 bytes
255 cabezas, 63 sectores/pista, 1027 cilindros
Unidades = cilindros de 16065 * 512 = 8225280 bytes

Disposit. Inicio    Comienzo      Fin      Bloques  Id  Sistema
/dev/hda1   *           1         978     7855753+  83  Linux
/dev/hda2             979        1027      393592+   5  Extendida
/dev/hda5             979        1027      393561   82  Linux swap / Solaris

Disco /dev/hdd: 4303 MB, 4303272960 bytes
255 cabezas, 63 sectores/pista, 523 cilindros
```

```
Unidades = cilindros de 16065 * 512 = 8225280 bytes
```

El nuevo disco es el /dev/hdd, ahora hay que crear una nueva partición en el disco y darle formato. Para ello escribimos en la línea de comandos:

```
# fdisk /dev/hdd
```

Después, elegiremos borrar todas las particiones existentes y crearemos una nueva de tipo linux y le daremos a escribir, ahora volvemos a hacer otra vez el fdisk -l y vemos que tenemos una nueva línea:

```
Disposit. Inicio    Comienzo       Fin      Bloques   Id  Sistema
/dev/hdd1               1         460     3694918+  83  Linux
```

Hacemos lo mismo en Rabius. No hace falta añadir el dispositivo a el fichero /etc/fstab, que es el encargado de montar las particiones en el arranque, pues de esto se encargara DRBD con ayuda de heartbeat. Ahora, para instalar DRBD escribiremos desde la consola de ambas máquinas:

```
# apt-get install drbd0.7-module-source drbd0.7-utils
```

Esto nos descargará el módulo para el kernel y un paquete con utilidades. Para que funcione el módulo en el kernel hay que compilarlo e instalarlo, para ello nos valdremos del comando que trae debian; “module-assistant”.

Ejecutamos las siguientes instrucciones desde la consola en ambas máquinas de nuevo:

```
# module-assistant prepare
# m-a auto-install drbd0.7-module-source
```

Ahora ya tenemos instalado el DRBD. El paso siguiente es la configuración, esta se modifica retocando el fichero /etc/drbd.conf. Este archivo debe ser idéntico en ambos servidores.

```
# nano /etc/drbd.conf
```

El archivo se configura de la siguiente forma:

```
resource data {  # le decimos el recurso a compartir y el protocolo a usar

  protocol C;
```

```
  startup {
    degr-wfc-timeout 10;    # al arrancar debe comprobar que el otro nodo existe
  }                          # o esperar

  disk {
    on-io-error   detach;    #le decimos que si el dispositivo tiene un error
  }                          # no lo arranque

  syncer {                   # le decimos que sincronice los dispositivos drbd
    rate 1M;                 # cada mega.
    Group 1;                 # y cada segundo
    al-extents 257;
  }

  on corman {                 # le damos los datos de corman y rabius, donde
    device     /dev/drbd0;    # esta el dispositivo a montar y la dirección ip
    disk       /dev/hdd1;     # de este
    address    172.168.0.1:7788;
    meta-disk  internal;
  }

  on rabius {
    device    /dev/drbd0;
    disk      /dev/hdb1;
    address   172.168.0.2:7788;
    meta-disk internal;
  }
}
```

Después de copiarlo en la segunda máquina reiniciamos. Entonces, mostramos el contenido del fichero /proc/drbd que nos dirá si drbd esta funcionando. Desde Corman, la salida sería:

```
# cat /proc/drbd

version: 0.7.21 (api:79/proto:74)
SVN Revision: 2326 build by root@corman, 2009-06-17 00:05:03
 0: cs:Connected st:Primary/Secondary ld:Consistent
    ns:8 nr:0 dw:4 dr:21 al:0 bm:1 lo:0 pe:0 ua:0 ap:0
```

Este archivo nos muestra que DRBD esta funcionando y actuando como primario en este nodo. Ahora hay que sincronizar los discos, desde el nodo primario ejecutamos:

```
#drbdsetup /dev/drbd0 primary -do-what-I-say
```

Creamos un punto de montaje, y montamos el sistema de ficheros de DRBD para meter los datos necesarios

```
#mkdir /data
#mount /dev/drbd0 /data
```

Copiamos los directorios que queramos compartir y ya tenemos el DRBD configurado. Ahora, hay que añadir el servicio al heartbeat, para que se clusterice mediante el haresources.

```
# nano /etc/haresources
```

La línea de configuración quedaría así al final:

corman IPaddr2::192.168.2.10/24/eth0:0 apache2 pingcheck drbddisk::data Filesystem::/dev/drbd0::/data::ext3

Es una sola línea a la que hemos añadido el servicio drbddisk con los parámetros de montaje necesarios, la unidad a montar, el destino, y el tipo.

Reiniciamos heartbeat, y tenemos nuestro cluster en funcionamiento.

6.3- Dándole utilidad real a DRBD

Queremos usar DRBD para que nuestros dos servidores tengan la misma web. Lo primero que tenemos que hacer para eso, es crear un directorio y una web en el. Esto lo haremos dentro de /data en Corman, que es el recurso activo.

```
# cd /data/
# mkdir web
```

El código en php que haremos para esta web es bastante sencillo, simplemente queremos una web que nos muestre en que servidor esta siendo ejecutada y la hora en la que se accede a la web. La creamos mediante el comando:

```
# nano web/index.php
```

Y escribimos:

```
<?$hostname= exec('hostname');  # abrimos las líneas de php con "<?" y las
$reloj= exec('date');?>          # cerramos con "?>". Mediante exec podemos
                                 # utilizar comandos de bash. Usamos esto para
                                 # meter dentro de las variables fecha y host.
        Esto funciona, este servidor es  <h2> <?echo $hostname?> </h2>

        <?echo $reloj?>             # ahora en html escribimos la frase, y
                                    # volvemos a abrir el php para mostrar las
```

```
                                    # variables de reloj y hostname
```

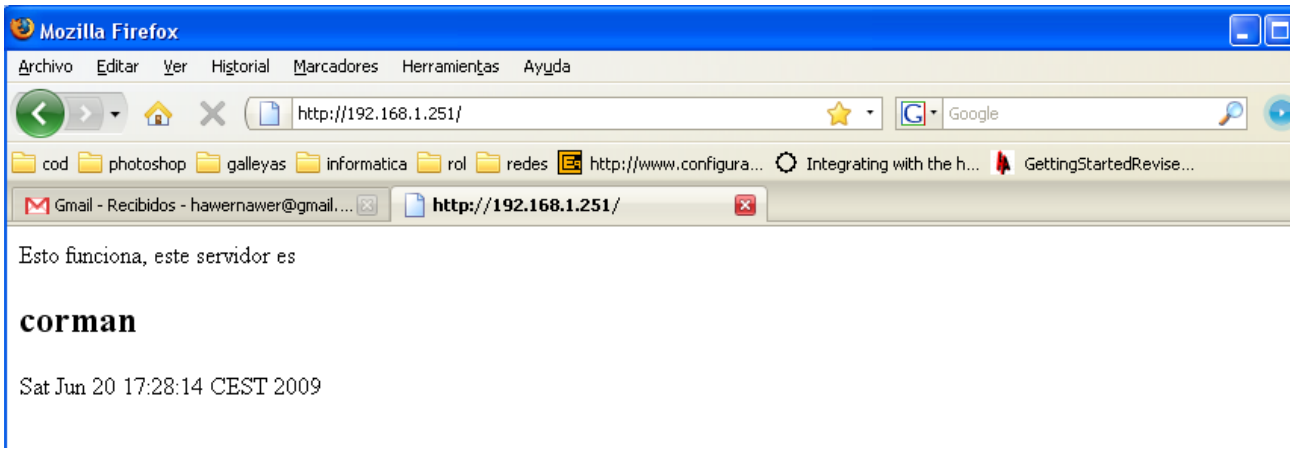
Ahora le damos la posesión de la carpeta web y a las subcarpetas al usuario apache haciendo lo siguiente:

```
# cd /data/
# chown -R www-data:www-data web/
```

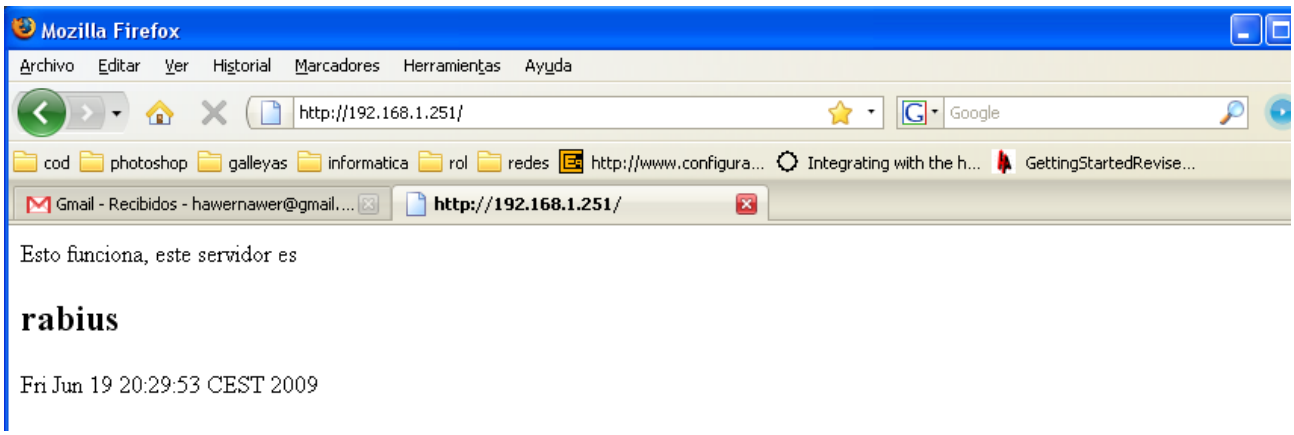
Lo siguiente es decirle a apache que la web que tiene que abrir por defecto es la que hemos creado. Nos vamos a la carpeta sites-enabled dentro de la configuración de apache, y retocamos el fichero 000-default que contiene los valores de la web por defecto.

```
# cd /etc/apache2/sites-enabled/
# nano 000-default
```

El valor que debemos modificar es el de "DocumentRoot" cambiaremos lo que pone tras esta directiva por el valor "/data/web/". Con esto, solo queda abrir el explorador y poner la dirección de nuestra web, el resultado es el siguiente:



Desconectamos el cable de corriente de corman, y actualizamos la we



7.- Conclusiones finales

El proyecto se ha llevado a cabo de una forma totalmente satisfactoria. Ha sido un proyecto complicado, no por ya la intención en si, si no porque he tenido muchos problemas derivado de el uso de equipos antiguos. Más de un formateo, alguna placa base rota, y un puerto PCI que no para de dar problemas han minado el avance de un proyecto, que aunque con objetivos cumplidos, sin tanto contratiempo podría haber llegado a implementar más opciones.

Mi contribución a los cluster de alta disponibilidad no es más que una guía de buena configuración, con el añadido de un sistema de comprobación para las conexiones a Internet, ya que eso, lo he programado yo completamente. Me queda un regusto amargo, porque hace poco descubrí más herramientas que pone a nuestra distribución el paquete de heartbeat, que es bastante extenso, el crm_manager. Un gestor de recursos para gestionar los recursos de una manera mas profunda que el propio haresources, y además, ofrecer alta disponibilidad de servicios, que no de hardware/conectividad. Aún así, creo que el proyecto cumple con todo lo esperado, y a mi personalmente me ha servido para ampliar mis conocimientos de una forma extensa, en el uso de scripts y de manejo de Linux en general.

En resumen , y dándolo por terminado, este proyecto ha sido muy instructivo y estoy bastante contento de haberlo escogido en su día.

8.- Linkografía

Página oficial de clusters de alto rendimiento de linux

www.linux-ha.org

Página oficial de DRBD

www.drbdo.org

Wikipedia usada para definiciones

http://es.wikipedia.org y http://en.wikipedia.org

Página para la configuración de los cables de red

http://www.configurarequipos.com/doc297.html

Y, en general:

www.google.es

Canales de irc:

#debian

#linux

#linux-ha

Todos ellos en chat.freenode.net