Cluster Web En Alta Disponibilidad Con LVS.

Autor: Juan Luis Sánchez Crespo
Fecha: 22 de junio del 2012

Índice de contenido

1.Introducción....3
2.Objetivos....3
3.Escenario....4
4.Modos de montar un sistema en HA....5
5.DRBD(Distributed Replicated Block Device)....6
5.1.Instalación....6
5.2.Configuración....6
5.3.Puesta en marcha....8
6.OCFS2(Oracle Cluster File System)....9
6.1.Instalación....10
6.2.Configuración....10
6.3.Puesta en marcha....12
7.Balanceo de la carga con LVS....12
7.1.Configuraciones de LVS....13
7.2.Algoritmos....13
7.3.Instalación....15
7.4.Configuración....15
8.Keepalived....16
8.1.Instalación....16
8.2.Configuración....16
8.3.Puesta en marcha....18
9.Apache....18
9.1.Instalación....19
9.2.Configuración....19
9.3.Puesta en marcha....20
10.PHP5....21
10.1.Instalación....21
10.2.Creación de una página de prueba....21
11.PostgreSQL....21
11.1.Instalación....22
11.2.Configuración....22
11.3.Creación de usuario y base de datos drupal....22
12.Drupal....23
12.1.Instalación....23
13.Cosas a tener en cuenta....26
14.Ficheros de configuración....27
14.1.DRBD en protoss y terran....27
14.2.OCFS2 en protoss y terran....27
14.3.LVS en zerg y supermente....28
14.4.Keepalived en zerg....28
14.5.Keepalived en supermente....29
14.6.Apache, servidores virtuales....29
14.7.Fichero fstab de terran y protoss....30
14.8.Ficheros de red....30
14.9.Sincronización al arranque....32
15.Bibliografía....32

1. Introducción.

Alta disponibilidad es la característica de aplicaciones y datos que se encuentren disponibles siempre, debido a su carácter crítico.

Esta característica se mide con el porcentaje que ha estado un sistema activo a lo largo de un año. Las medidas son las siguientes:

- 99,9% (“tres nueves”), ha estado el sistema 8,76 horas inactivo al año.
- 99,99%(“cuatro nueves”), ha estado el sistema 52,6 minutos inactivo al año.
- 99,999%(“cinco nueves”), ha estado el sistema 5,26 minutos inactvo al año.

Que un sistema esté el menor tiempo posible inactivo se puede conseguir de diferentes maneras pero todas tienen una característica, la replicación. Esta replicación la podemos obtener en un mismo sistema informático o mediante varios sistemas informáticos.

Por ejemplo, para la replicación en un mismo sistema podemos ponerle fuentes redundantes o discos en raid. Si queremos obtenerla con sistemas diferentes estos sistemas a parte de tener redundantes sus componentes, son varios que actúan como uno. Esto se llama clúster.

Un clúster son dos o más sistemas informáticos que funciona como si fuesen uno. Estos ordenadores pueden estar unidos mediante una red informática. Los clúster se utilizan para mejorar el rendimiento y la disponibilidad de los servicios ofrecidos.

Un clúster puede tener una o barias de las siguientes características:

- Alto rendimiento.
- Alta disponibilidad.
- Escalabilidad.
- Balanceo o equilibrio de carga.

También tenemos que tener en cuenta que estos sistemas en función de su uso pueden ser:

- Activo/Pasivo: Esto es cuando uno de los servidores está activo mientras el resto esta inactivo esperando que el servidor activo deje de dar servicio para asumirlo alguno de ellos.
- Activo/Activo: Cuando todos los servidores del cluster están dando servicios.

2. Objetivos.

El objetivo de esta practica es instalar un clúster de alta disponibilidad activo/pasivo con keepalived. Estas dos maquinas lo que harán es balancear la carga con LVS(Linux Virtual Server). El balanceador que esté activo en ese momento repartirá las peticiones sobre los diferentes

servidores web que estén activos. Estos servidores tendrán montado un drupal y se comunicarán con el balanceador por una red interna.

Para la sincronización de datos los servidores utilizaran DRBD con OCFS2. Mediante estas herramientas se replicarán los datos de la web y de las bases de datos.

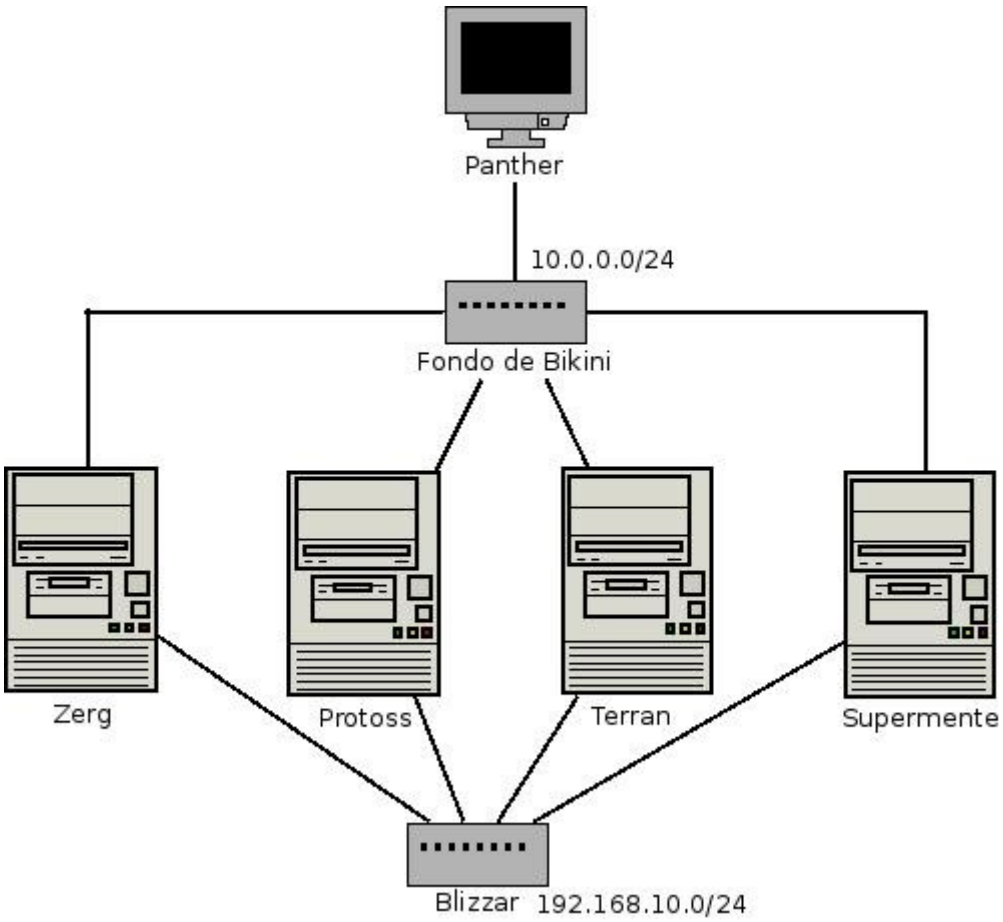
Como servidor web vamos a utilizar apache con php5 y se conectara a las bases de datos que son internas. Este servidor será un drupal.

3. Escenario.

Para este proyecto vamos a utilizar cuatro maquinas virtuales con Debian Squezee amd64. A las máquinas le he puesto Zerg, Protoss, Terran y Supermente.

- Zerg: Será el balanceador de carga principal en modo Direct Routing. Sus direcciones ip son: eth0 10.0.0.1, eth1 192.168.10.1.
- Protoss: Será uno de los servidores web. Sus direcciones ip son: eth0 10.0.0.2, eth1 192.168.10.2, lo:0 10.0.0.4.
- Terran: Es el otro servidor web. Sus direcciones ip son: eth0 10.0.0.3, eth1 192.168.10.3, lo:0 10.0.0.4.
- Supermente: Será el balanceador de carga secundario. Sus direcciones ip son eth0 10.0.0.5, eth1 192.168.10.5.

El esquema de la red sería el siguiente:



4. Modos de montar un sistema en HA.

Un sistema en alta disponibilidad lo podemos montar de diferentes modos, dependiendo de lo que busquemos y del presupuesto. Empezando por los ordenadores, podemos tenerlos de diferentes maneras:

- Virtualizados: Está es la manera más económica y la que voy a utilizar yo. Se utiliza sobre todo para hacer pruebas, ya que se puede hacer una máquina virtual que tenga las características de las dos sumadas y su rendimiento sería mayor. También tiene el inconveniente de que si se estropea la anfitriona, se estropean las dos. Si nos puede salvar si nos cargamos el sistema operativo de uno de ellas.
- En la misma sede: Esta manera es mejor que la anterior, ya que utiliza dos ordenadores como uno. Pero en caso de que ocurra una inundación, corte de luz, terremoto o cualquier desgracia que afecte a la sede, se verán afectados los ordenadores. Otra de las características a favor de este tipo de instalaciones es que la conexión de red es a nivel local, por lo que puede ser más segura y más rápida.
- En diferentes sedes: Esta manera es más difícil, ya que tiene que tener distintas sedes. También tiene el inconveniente de que si el ancho de banda de red no es muy grande, los ordenadores no se podrían comunicar entre sí y también tendríamos que mirar si el protocolo que utilizamos es seguro o no.

Una vez tengamos decidido el tipo de instalación, tenemos que ver para qué queremos la alta disponibilidad. Por ejemplo, si lo que buscamos es que nuestro servicio esté siempre activo, podemos utilizar alta disponibilidad en modo activo/pasivo. Si a parte de eso queremos rendimiento buscaremos alta disponibilidad en modo activo/activo. Si son varios procesos y simplemente buscamos rendimiento, podemos separar los procesos en diferentes servidores y balancear los más pesados. Ejemplos:

- Tenemos un servidor web, con una página estática, pero queremos tener un servidor secundario por si ocurre algo. En este caso utilizaremos alta disponibilidad en modo activo/pasivo.
- Tenemos un servidor web, con un drupal que consume todos los recursos de un ordenador en las horas puntas. En este caso podemos balancear la carga entre dos o más servidores.
- Por último, tenemos un servidor web, con un drupal, pero nos hemos dado cuenta que en las horas puntas las bases de datos no pueden con la carga. En este caso pondremos un servidor web que haga consulta a un balanceador de carga, que balancee a diferentes bases de datos.

Otra cosa que tenemos que mirar es si tenemos que replicar los datos y cómo. Si, por ejemplo, es un servidor DNS que no sufre muchos cambios, o si es un servidor que tiene muchos cambios y de gran carga o si solo tiene pequeños cambios. También tendríamos que mirar el número de nodos que tenemos, ya que puede repercutir en la replicación de datos. Dependiendo de cada necesidad podemos utilizar diferentes opciones, por ejemplo:

- En el caso de un servido DNS, podemos sincronizar los datos con rsync.
- Si la carga es sobre todo de lectura podemos utilizar DRBD, con el cual tenemos los datos sincronizados continuamente y en local, por lo que la lectura es instantánea. Este método si fuera para un sistema con mucha escritura de datos y con muchos nodos, podría ser lento, ya que la información se tendría que replicar en todos los nodos.

- Por ultimo si tenemos varios nodos con mucha escritura, podemos montar un sistema de archivos centralizados con una San. Esta opción es el mejor de los tres métodos, pero también es la mas cara.

5. DRBD(Distributed Replicated Block Device).

DRBD es un sistema de sincronización de datos, lo que viene haciendo es un raid1 por red.

Debian Squeeze trae soporte para drbd en el kernel, por lo que no necesita cargar ningún módulo, que en versiones más antiguas si hacia falta.

DRBD crea un dispositivo de bloque drbdX accesible desde los dos servidores. El servidor primario es el que tiene acceso de lectura y escritura mientras que el secundario solo tiene de lectura. A partir de la versión 0.8, DRBD nos permite tener dos servidores primarios, por lo que los dos pueden leer y escribir.

Cuando escribimos algo en alguno de los dispositivos drbdX, se escriben en la partición física y esos mismos datos son enviados mediante TCP/IP al otro servidor.

5.1. Instalación.

En mi caso como he utilizado Debian Squezee, ya trae soporte de DRBD en el núcleo, así que solo tendremos que instalar las herramientas necesarias para gestionar DRBD:

```
# aptitude install drbd8-utils
```

Con esto se instalarán los siguientes comandos:

- drbdmeta: se utiliza para mostrar, crear y modificar el sistema de metadatos. Se encarga de la sincronización.
- drbdsetup: permite asociar el dispositivo drbd con los ficheros de bloques y sirve para cambiar casi todos los parámetros de configuración.
- drbdadm: esta herramienta es la que realmente se utiliza ya que en función de los parámetros que se le introduzcan, ejecuta un drbdmeta o un drbdsetup.

5.2. Configuración.

El fichero de configuración principal se encuentra en /etc/drbd.conf. Antes, se incluía la configuración completa en este fichero, pero ahora contiene dos includes:

```
include "drbd.d/global_common.conf";
include "drbd.d/*.res";
```

Esto lo que vine a hacer es agregar todos los ficheros que se llamen global_common.conf o que terminen en .res y se encuentre en el directorio /etc/drbd.d. Con esta separación podemos separar la

configuración global de la de los RAID-1 que vamos a crear.

En el fichero global_common.conf encontramos las opciones que afectarán a todos los dispositivos que creemos, si en la definición de este dispositivo no pone lo contrario. En este fichero podemos ver dos secciones: global, donde se configuran los parámetros globales de DRBD y common, donde se configuran los parámetros de configuración comunes.

Los ficheros de configuración .res solo contienen la sección resource, donde se definen los recursos. Dentro de esta sección podemos poner las mismas opciones en la sección common del fichero de configuración común para todos los dispositivos. La parte más importante dentro de este fichero, es la que define al cliente(on cliente). Todo lo que esté en ese apartado es exclusivo para ese cliente, mientras lo que esté afuera afectará a todos los clientes.

Con este esquema de ficheros, podemos definir los parámetros comunes en la sección common del fichero global_common.conf y los parámetros específicos de un recurso en su fichero de configuración. A continuación voy a describir mi fichero de configuración.

En la sección global, le he dicho que no utilice el contador. Este contador lo que hace es conectarse a la página de DRBD y formar parte de las estadísticas.

```
global {
      usage-count no;
}
```

En la parte common he definido todos los parámetros de configuración para que los nodos estén en primario los dos. Estas opciones también se podría haber puesto en la parte resource de un recurso concreto. Lo primero que he configurado en esta parte es el protocolo de sincronización C, el cual es necesario si queremos tener dos nodos primarios. Los otros protocolos son A y B.

```
protocol C;
```

Estos protocolos tienen las siguientes características:

- A: Protocolo de replicación asíncrona.
- B: Protocolo de replicación síncrona de memoria(semi-síncrono).
- C: Protocolo de replicación síncrona.

En esta sección se le dice a DRBD que cuando arranque, lo haga en modo primario.

```
startup {
      become-primary-on both;
}
```

En net se definen los parámetros de red y qué hacer en caso de fallo, en este caso se ha configurado para que se permitan dos primarios y que cuando haya un fallo porque nadie tenga el recurso, lo tenga uno solo, o lo tengan los dos, y que no hagan nada para evitar la corrupción de datos, de eso ya se encargará OCFS2.

```
net {
            allow-two-primaries;
            after-sb-0pri discard-zero-changes;
            after-sb-1pri discard-secondary;
            after-sb-2pri disconnect;
      }
```

Por último ponemos la velocidad de sincronización en la sección syncer.

```
syncer {
      rate 1000M;
}
```

Ya en el fichero de configuración del recurso, el cual he llamado drbd.res, solo se encuentra la sección resource y en ella se define el nombre del recurso, el disco físico que va a utilizar, el nombre del dispositivo que nos va a crear DRBD. También le vamos a decir que los metadatos se guarden junto con los datos y también pondremos la dirección ip y el puerto tcp por el que se hará la sincronización.

```
resource drbd {
device    /dev/drbd1; # Dispositivo que nos va a crear.
meta-disk internal; # Como guardar los metadatos.
  on protoss { # Nombre del nodo.
    disk      /dev/vdb1; # Disco físico.
    address   10.0.0.2:7789; #Dirección y puerto.
  }
  on terran {
    disk      /dev/vdb1;
    address   10.0.0.3:7789;
  }
}
```

Los elemento que tengan en común se pueden sacar a resource. He puesto los discos dentro, pero se podrían haber puesto bajo meta-disk ya que en los dos servidores son el vdb1.

5.3. Puesta en marcha.

Tras la configuración tendremos que iniciar el servicio con los siguientes comandos. Donde he puesto drbd, tenemos que poner el nombre del recurso, también podemos poner “all” y serán todos los recursos que tengamos. Primero para crear los dispositivos de bloques que se van a replicar a los cuales se les asigna el nombre que se le haya dado en el fichero de configuración:

```
# drbdadm create-md drbd
```

Ahora vamos a asociar el dispositivo drbd con el dispositivo físico:

```
# drbdadm attach drbd
```

Configuramos los parámetros de sincronización que definimos en el fichero drbd.conf o global_common.conf.

```
# drbdadm syncer drbd
```

Por último conectamos los dos servidores entre sí.

```
# drbdadm connect drbd
```

Los últimos tres pasos(asociación del dispositivo, configuración de parámetros y conexión) los

podemos hacer con un parámetro.

```
# drbdadm up drbd
```

Estos comandos se tendrían que hacer en las dos máquinas, tras esto tendríamos que pasar a replicar los datos de un servidor. Se haría con el siguiente comando

```
# drbdadm -- -- overwrite-data-of-peer primary all
```

Con todo esto los datos ya empiezan a sincronizarse. Podemos verlo de varias maneras yo lo hice de la siguiente:

```
# /etc/init.d/drbd status
drbd driver loaded OK; device status:
version: 8.3.7 (api:88/proto:86-91)
srcversion: EE47D8BF18AC166BE219757
m:res   cs          ro                 ds                     p  mounted
fstype
...     sync'ed:    14.9%              (895420/1048220)K
1:drbd  SyncSource  Primary/Secondary  UpToDate/Inconsistent  C
```

Otra manera sería viendo el fichero /proc/drbd.

Como podemos observar el segundo servidor aparece como secundario, para solucionarlo ejecutamos este comando en el segundo servidor.

```
drbdadm primary all
```

Si tras esto volvemos a mirar como esta nuestro drbd:

```
# /etc/init.d/drbd status
drbd driver loaded OK; device status:
version: 8.3.7 (api:88/proto:86-91)
srcversion: EE47D8BF18AC166BE219757
m:res   cs          ro               ds                     p  mounted
fstype
...     sync'ed:    49.3%            (533404/1048220)K
1:drbd  SyncSource  Primary/Primary  UpToDate/Inconsistent  C
```

Esto no es suficiente para la sincronización, ahora tendremos que instalar OCFS2, para darle formato a nuestro disco.

6. OCFS2(Oracle Cluster File System).

OCFS2 es un sistema de fichero en clúster que permite el acceso simultaneo de varios nodos. Está desarrollado por Oracle y tiene licencia GNU.

Cada nodo dispone de un sistema de ficheros montados, normalmente escribe en un fichero meta-data permitiendo a los otros nodos saber que se encuentra disponible.

Limitaciones:

- Permite hasta 32000 directorios.
- El tamaño máximo de un volumen es de 4PB.
- El tamaño máximo de un fichero es de 4PB.
- La longitud del nombre del fichero es de 255 bytes.

Se ha escogido OCFS por ser más adecuado para distribuciones basadas en Debian. En contraposición tenemos a GFS2 más indicado para distribuciones Red Hat y derivados.

6.1. Instalación.

La instalación en Debian de este software es sencilla, ya que está disponible en los repositorios:

```
# apt-get install ocfs2-tools ocfs2console
```

6.2. Configuración.

Tras la instalación se nos instalarán dos servicios:

- ocfs2: se encarga de los recursos que están montados en el clúster. Su fichero de configuración se encuentra en /etc/ocfs2/cluster.conf y contiene la siguiente información:

```
node:
    ip_port = 7777 # Puerto por el que queremos que los nodos se
comuniquen.
    ip_address = 192.168.0.2 # Dirección Ip del nodo
    number = 0 #Número de identificación que se le quiere dar a el nodo.
    name = protoss #Nombre del nodo
    cluster = ocfs2 #Nombre del clúster
cluster:
    node_count = 2 #Nodos que tiene el clúster
    name = ocfs2 #Nombre del clúster
```

- o2cb: se encarga de los parámetros técnicos del clúster y su fichero de configuración se encuentra en /etc/default/o2cb. En este fichero encontramos lo siguiente:

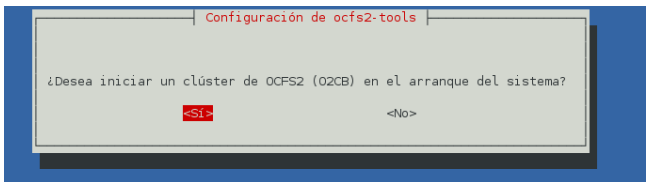
```
# O2CB_ENABLED: True si quiere que se arranque el servicio con el sistema.
O2CB_ENABLED=true
# O2CB_BOOTCLUSTER: El nombre que se le quiere dar a el cluster.
O2CB_BOOTCLUSTER=ocfs2
# O2CB_HEARTBEAT_THRESHOLD:Iteraciones necesarias para considerar un nodo
caído.
O2CB_HEARTBEAT_THRESHOLD=31
# O2CB_IDLE_TIMEOUT_MS: Tiempo en milisegundos a partir del cual la red se
considera caida.
O2CB_IDLE_TIMEOUT_MS=20000
# O2CB_KEEPALIVE_DELAY_MS: Tiempo máximo para enviar un paquete para
comprobar si un nodo esta activo.
O2CB_KEEPALIVE_DELAY_MS=2000
# O2CB_RECONNECT_DELAY_MS: Tiempo mínimo entre intentos de conexión.
O2CB_RECONNECT_DELAY_MS=2000
```

Este fichero se puede configurar de forma automática con el comando:

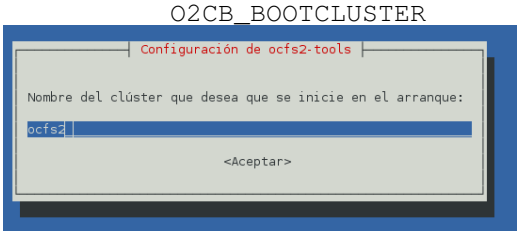
```
# dpkg-reconfigure ocfs2-tools
```

Y siguiendo los pasos, los cuales se corresponden con las opciones anteriores:

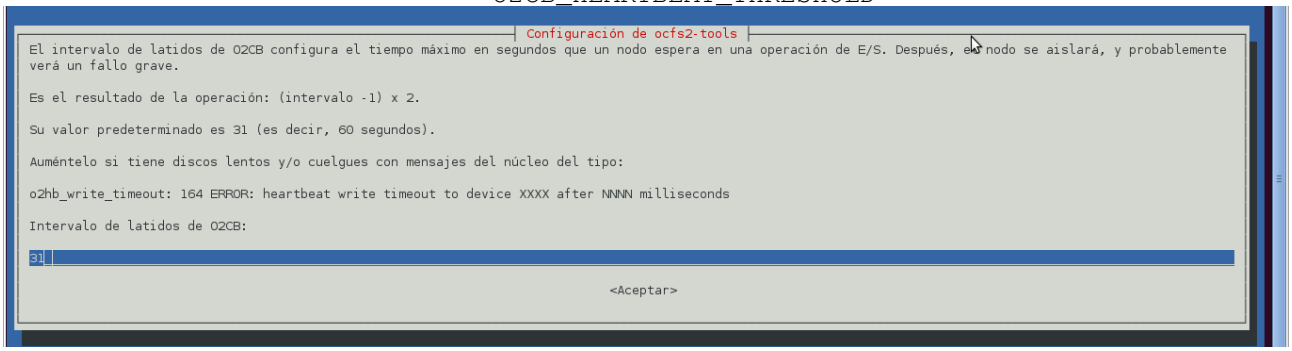
```
O2CB_ENABLED
```



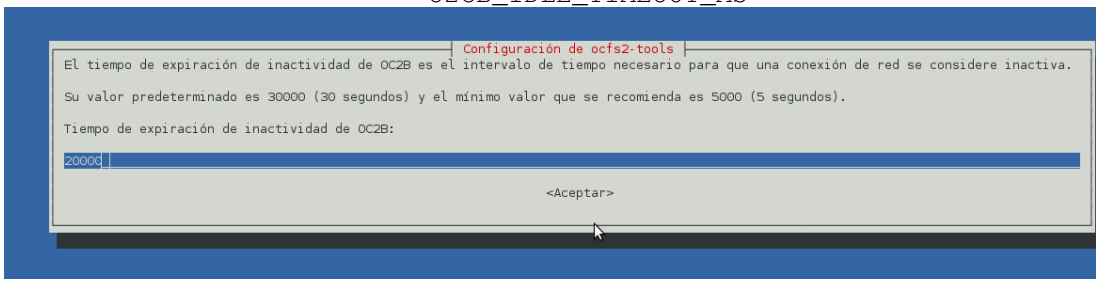
O2CB_BOOTCLUSTER



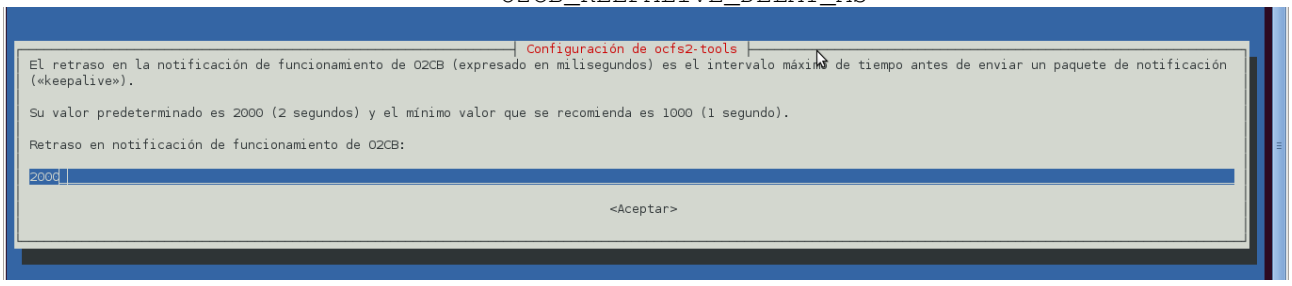
O2CB_HEARTBEAT_THRESHOLD



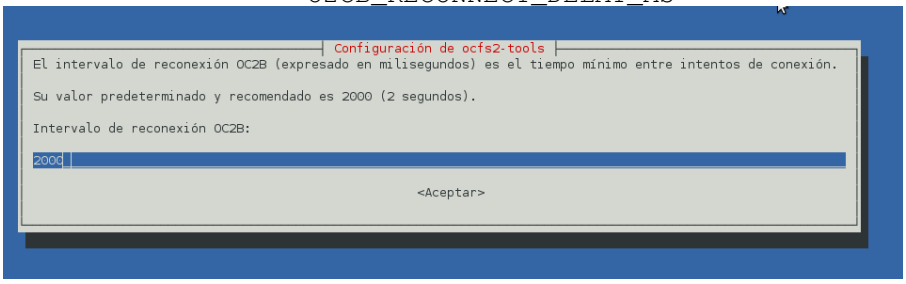
O2CB_IDLE_TIMEOUT_MS



O2CB_KEEPALIVE_DELAY_MS



O2CB_RECONNECT_DELAY_MS



6.3. Puesta en marcha.

Tras tenerlo configurado, reiniciamos los servicios:

```
# /etc/init.d/ocfs2 restart
# /etc/init.d/o2cb restart
```

Con esta herramienta se instalarán los siguientes comandos:

- mount.ocfs2: Para montar el sistema de ficheros en un directorio.
- mkfs.ocfs2: Crea el sistema de ficheros en un dispositivo. El servicio o2cb debe estar parado.
- fsck.ocfs2: Chequea el sistema de ficheros.
- mounted.ocfs2: Detecta los volúmenes ocfs2 que haya montados.
- tunefs.ocfs2: Sirve para cambiar parámetros del sistema de ficheros
- debugfs.ocfs2: Es un depurador del sistema de ficheros y sirve para ver las estructuras del dispositivo.
- ocfs2cdsl: Permite crear enlaces simbólicos a un fichero o directorio

La que nosotros utilizaremos con el dispositivo de bloque será mkfs.ocfs2, para formatearla en ocfs2:

```
# mkfs.ocfs2 /dev/drbd1
```

Para montar la partición una vez formateada:

```
#  mount /dev/drbd1 /opt
```

Y como queremos que se monte con la máquina tenemos que añadir la siguiente linea al fichero fstab:

```
/dev/drbd1 /opt ocfs2 rw,_netdev,heartbeat=local 0 0
```

Para que monte la partición correctamente no me valió con eso, así que añadí una línea en el fichero /etc/rc.local, para que cargara de nuevo el fichero fstab. Con esto tenemos que tener cuidado, ya que todos los programas que necesiten los ficheros de /opt(va a ser donde voy a colocar los ficheros que se tengan que sincronizar, como los de apache) tienen que arrancar después de que se monte la partición. La línea que añadí al fichero fue:

```
mount –a
```

7. Balanceo de la carga con LVS.

El balanceo de la carga es una técnica que se utiliza para compartir el trabajo entre varios procesos, ordenadores, discos u otros recursos. El balanceo de carga se puede hacer mediante diferentes algoritmos. En nuestro caso vamos a utilizar LVS.

LVS es un software que nos permite balancear la carga entre varios servidores. LVS está integrado dentro del kernel de Linux con lo cual se consigue un mayor rendimiento, además existe gran cantidad de documentación en la red. LVS también es la opción más genérica, con LVS se puede balancear tanto HTTP ,FTP ,MySQL o cualquier otro protocolo de red.

7.1. Configuraciones de LVS.

LVS nos permite diferentes configuraciones de red. Las configuraciones que disponemos son:

- NAT: Con esta técnica se modifica el datagrama IP. En esta técnica hay un dispositivo intermedio actuando como puerta de enlace. Este dispositivo recibe un paquete, modifica el datagrama, y lo envía a uno de los servidores que tenga asignado para ese servicio. El servidor procesa esa petición y se la envía a la puerta de enlace de nuevo. La puerta de enlace vuelve a cambiar el datagrama y se lo envía al cliente. La configuración de un servidor virtual mediante NAT es muy transparente, ya que no requiere modificaciones ni configuraciones especiales en los sistemas operativos de los servidores reales, lo cual facilita su implementación. Una de las principales desventajas de esta técnica es que los servidores deben de encontrarse en la misma red que el balanceador pero esto no es un gran impedimento. La principal desventaja de esta técnica es el cuello de botella que puede causar el tener que pasar todos los paquetes tanto de ida como de vuelta por la puerta de enlace. En este caso el balanceador tiene que cambiar tanto la dirección IP como la MAC de los paquetes de los clientes y de los servidores reales.

- Direct Routing(Enrutamiento directo): En este caso el balanceador de carga sigue siendo el que recibe las peticiones pero son los servidores los que envían la respuesta. El balanceador y los servidores deben de estar conectados por una LAN. En esta técnica el principal inconveniente es el arp, ya que todos los servidores reales tienen que saber que la ip virtual es suya, pero si lo ponemos en una interfaz física con arp, todos dirán que la dirección virtual es la suya. Para arreglar esto tenemos dos métodos, unos es desactivar el arp de los servidores, por lo que ya solo el balanceador contestará y otra es asignarle la dirección a la interfaz loopback, por lo que el servidor real sabrá que pertenece a él pero no contestará a las peticiones arp ya que son en otra red. Para que funcione bien, los servidores no deben responder a los mensajes ARP. En este caso el balanceador de carga solo tiene que cambiar la dirección MAC para direccionar las peticiones al servidor real.

- IP Tunneling(Encapsulación IP): En el IP Tunneling, los servidores no tienen que estar conectados directamente a una red donde se encuentre el balanceador. En este caso el balanceador recibe el datagrama del cliente con destino el servicio virtual. El datagrama lo reenvía con dirección Ip del cliente como origen y dirección IP del servicio virtual como destino, este datagrama va metido dentro de otro datagrama con dirección Ip del balanceador como origen y dirección Ip del servidor como destino. El paquete pueden ser enviados por dispositivos intermedios como routers.

7.2. Algoritmos.

Estas configuraciones de red pueden utilizar diferentes algoritmos para balancear las cargas. Los algoritmos que podemos utilizar son:

Round-Robin(rr):

La clásica cola Round Robin o FIFO: cada petición se envía a un servidor, y la siguiente petición al siguiente servidor de la lista, hasta llegar al último tras lo cual se vuelve a enviar al primero.

Es la solución más sencilla y que menos recursos consume, a pesar de que no es la más justa, es posible que toda la carga “pesada” vaya a parar al mismo servidor mientras que el resto sólo reciban

peticiones triviales.

Un problema de este método es que todos los servidores recibirán el mismo número de peticiones, independientemente de si su potencia de cálculo es la misma o no.

Weighted Round-Robin(wrr):

Este algoritmo es igual que el anterior, pero añadiendo un “peso” a cada servidor. Este peso no es mas que un entero que indica la potencia de cálculo del servidor, de forma que la cola Round Robin se modificará para que aquellos servidores con mayor potencia de calculo reciban peticiones más a menudo que el resto. Por ejemplo, si tenemos tres servidores A, B y C, con una cola Round Robin normal la secuencia de distribución tendrá tres pasos y será ABC. Si usamos una Round Robin Ponderada y asignamos pesos 4, 3 y 2 respectivamente a cada servidor, la cola ahora distribuirá en nueve pasos (4+3+2) y una posible planificación de acuerdo a estos pesos sería AABABCABC.

El problema de este método es que, si bien asegura que los servidores más capaces reciban mas carga, también por probabilidad acabarán recibiendo más peticiones “pesadas”, con lo que a pesar de todo podrían llegar a sobrecargarse.

Least-Connection(lc):

Este mecanismo de distribución consulta a los servidores para ver en cada momento cuántas conexiones abiertas tiene cada uno con los clientes, y envía cada petición al servidor que menos conexiones tenga en ese momento. Es una forma de distribuir las peticiones hacia los servidores con menos carga.

A pesar de que sobre el papel parece que este método si que será capaz de repartir la carga sobre todos los servidores de una forma equitativa. En la práctica falla cuando la potencia de los servidores no es la misma: si todos tienen más o menos las mismas características, este algoritmo funciona como se espera, si hay diferencias en las prestaciones de los equipos, lo que ocurre en la práctica es que debido a la espera en TIME_WAIT de las conexiones perdidas (alrededor de 2 minutos por lo general), los servidores rápidos tendrán en un momento dado una gran cantidad de conexiones activas siendo atendidas, y otra cantidad también grande de conexiones realmente inactivas, pero aún abiertas en TIME_WAIT, mientras que los servidores lentos tendrán muchas menos conexiones tanto activas como en TIME_WAIT, de forma que se enviará más carga a los servidores lentos.

Weighted Least-Connection(wlc):

Al igual que la estrategia Round Robin Ponderada, en este algoritmo se coge el anterior y se le añaden unos pesos a los servidores que de alguna forma midan su capacidad de cálculo, para modificar la preferencia a la hora de escoger uno u otro según este peso.

Locality-Bassed Least-Connection(lblc):

Este algoritmo intenta asignar conexiones a la misma IP hacia un servidor real. Este método es

frecuentemente empleado en compañía de proxys http.

Locality-Bassed Least-Connection with Replication(lblcr):

Se trata de una variante de Locality-Bassed Least-Connection la cual permite a un grupo de servidores mantener una dirección IP dada en situaciones con mucha carga.

Source-Hashing/Destination-Hashing(sh/dh):

En estos dos últimos métodos se dispone de una tabla de asignaciones fijas, en las que bien por la IP de origen o de destino, se indica qué servidor deberá atender la petición.

El balanceador compara las direcciones de los paquetes TCP/IP que reciba con estas tablas y actúa en consecuencia.

Shortest Expected Delay(sed):

Dirige las conexiones al servidor real cuya respuesta sea la más rápida.

Never Queue(nq):

Dirige una conexión al servidor real que esté libre si es que hay alguno, si esto no fuese posible emplea el algoritmo Shortest Expected Delay altgorithm.

7.3. Instalación.

Para la instalación simplemente tenemos que instalar la herramienta administrativa, ya que LVS viene integrado en el núcleo de linux.

```
# aptitude install ipvsadm
```

7.4. Configuración.

Su fichero de configuración es /etc/default/ipvsadm y el fichero de configuración de sus reglas es etc/ipvsadm.rules. En mi caso no me va a hacer falta este fichero ya que keepalived será el encargado de manejarlos.

A parte de por ficheros de configuración se puede configurar con comandos. La primera vez lo configuré por comandos para ver si tenía bien configurado los clientes, ya que resultaba más fácil. Para configurar LVS por comando, lo primero que tenemos que definir es el servicio que vamos a balancear, en mi caso el puerto 80 y el algoritmo, si no ponemos ninguno coge el que trae por defecto(wlc) yo he puesto wrr. El comando es el siguiente:

```
ipvsadm –A –t 10.0.0.4:80 –s wrr
```

Tras esto tendremos que ir añadiendo todos los servidores reales:

```
ipvsadm –a –t 10.0.0.4:80 –r 192.168.10.2:80 –g
ipvsadm –a –t 10.0.0.4:80 –r 192.168.10.3:80 –g
```

8. Keepalived.

Keepalived es un demonio que monitoriza el estado de los servicios que se le dicen en su fichero de configuración. Keepalived implementa un framework basado en tres tipos de chequeo: nivel 3, nivel 4 y nivel 5/7. Este Framework da al demonio la habilidad de chequear el estado de un pool de Linux Virtual Server (LVS). Cuando un servidor del pool está parado, keepalived informa al kernel de Linux para borrar a este servidor de la entrada de la topología de LVS. Además keepalived implementa una pila VRRPv2 (es un protocolo de redundancia que significa Virtual Router Redundancy Protocol version2) para manejar los nodos del balanceador.

En resumen, controla en el propio balanceador (que es un clúster activo/pasivo) el nodo que es el activo y el pasivo, de modo que si uno cae el otro siga dando el servicio de balanceo. Y por otro lado, monitoriza a los servidores reales viendo si tienen los servicios levantados o no, para sacarlos de la tabla de candidatos a recibir peticiones del servicio monitorizado.

8.1. Instalación.

Viene en los repositorios de Debian para instalarlo:

```
aptitude install keepalived
```

8.2. Configuración.

Su fichero de configuración es /etc/keepalived/keepalived.conf y contiene tres partes:

global_defs; en esta primera parte tenemos las definiciones globales. En mi caso solo tengo el identificador del director. En este apartado se puede configurar el envío de correos si se tiene un servidor de correos. Mi apartado queda así.

```
global_defs {
      lvs_id LVS1
}
```

virtual_server; directivas de cada servidor virtual. Aquí se define la ip virtual y el puerto. Se define tantas como ipvirtuales o servicios tengamos. En este apartado encontramos las opciones:

- *delay_loop*, intervalo de chequeo en segundos.
- *lb_algo*, algoritmo de reparto.
- *lb_kind*, método de redirección.
- *protocol*, protocolo que va a redirigir.
- *sorry_server*, donde redirigir las peticiones en caso de que todos los servidores reales estén caídos.
- *real_server*, especifica un servidor real

- *wight*, peso que tiene el servidor real a la hora del balanceo.
- *TCP_CHECK*, en este apartado se especifica que chequee el puerto 80 TCP con un tiempo de espera.

```
virtual_server 10.0.0.4 80 {
   delay_loop 1
   lb_algo wrr
   lb_kind DR
   protocol TCP
   sorry_server 10.0.0.2 80
  real_server 192.168.10.2 80 {
    weight 1
    TCP_CHECK {
       connect_port 80
       connect_timeout 1
    }
  }
  real_server 192.168.10.3 80 {
    weight 1
    TCP_CHECK {
       connect_port 80
       connect_timeout 1
    }
  }
}
```

vrrp_instance; éstas son las directivas de los directores. Y sus opciones son:

- state, rol que va a tomar(*MASTER* o *BACKUP)*
- interface, interfaz de red de ese nodo.
- lvs_sync_daemon_inteface, interfaz para la sincronización de lvs
- virtual_router_id 51, router virtual de la instancia.
- priority, prioridad del nodo, el nodo principal tiene que tener un número mayor.
- advert_int, intervalo entre chequeos en segundos.
- smtp_alert, para que nos alerte por SMTP, en mi caso no hace nada.
- Authentication, en esta sección se configura el tipo de autenticación entre los balanceadores.
- auth_type, tipo de contraseña.
- auth_pass, contraseña.
- virtual_ipaddress, en esta sección se define la dirección ip del cluster

```
vrrp_instance VI_1 {
  state MASTER
  interface eth0
  lvs_sync_daemon_inteface eth0
  virtual_router_id 51
  priority 150
  advert_int 1
  smtp_alert
  authentication {
     auth_type PASS
     auth_pass example
  }
  virtual_ipaddress {
```

```
        10.0.0.4
    }
}
```

En el fichero de configuración del nodo secundario solo cambian las opciones: lvs_id, priority y states.

8.3. Puesta en marcha.

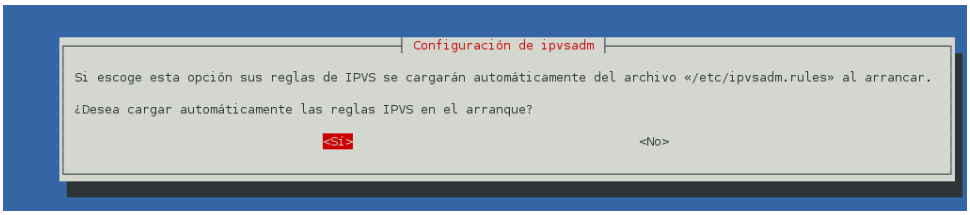
Tras la configuración solo tenemos que reiniciar el demonio.

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

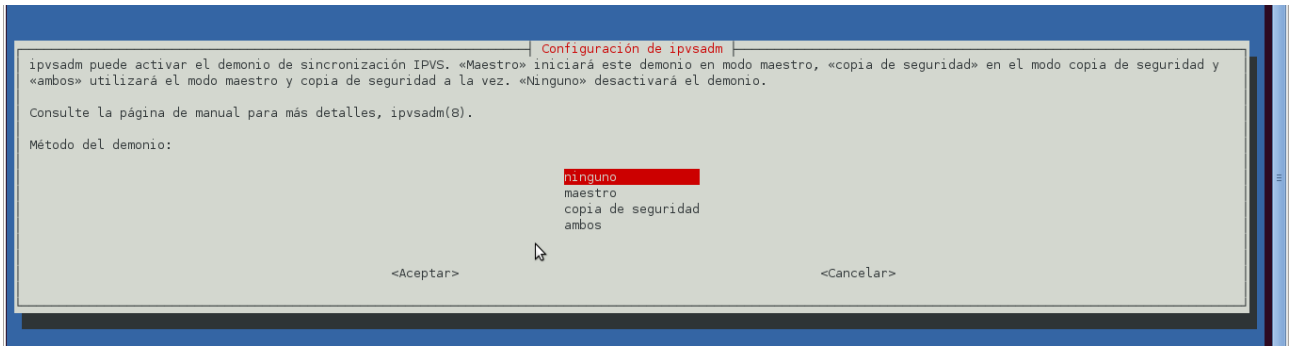
Por último, keepalived necesita del módulo ip_vs. Para que cargue este módulo lo podemos hacer de varias maneras, crear un script y ponerlo en rc2.d, poner un comando /etc/rc.local o hacer que ipvsadm cargue el módulo automáticamente. Yo opté por la última, y es tan fácil como:

```
# dpkg-reconfigure ipvsadm
```

Esta es la opción que nos permite arrancar el módulo ip_vs.



Esto es por si tenemos la configuración con ipvsadm, pero en nuestro caso está con keepalived.



9. Apache.

Un servidor web es un programa que implementa el protocolo HTTP. Se ejecuta continuamente en un ordenador manteniéndose a la espera de recibir peticiones por parte de un cliente.

Como servidor web se ha escogido apache. Apache es un servidor weg HTTP de código abierto que implementa el protocolo HTTP. Es el servidor web más utilizado y es altamente configurable.

9.1. Instalación.

Para la instalación, la podemos hacer mediante los repositorios:

```
# aptitude install apache2
```

En los repositorios también tenemos muchos módulos que podemos ir añadiendo a apache para hacerlo más configurable.

9.2. Configuración.

Los ficheros de configuración de apache se encuentran en /etc/apache2, pero yo utilizaré solo los que se encuentran en el directorio /etc/apache2/sites-available/. Tras crear el fichero en ese directorio tendremos que crear un enlace a ../sites-enabled, apache incorpora una herramienta que lo hace automáticamente llamada **a2ensite <nombre del sitio>**. Estos ficheros de configuración son para configurar diferentes sitios virtuales.

Tras instalar LVS, para probar que funcionaba correctamente, simplemente modifiqué el archivo /var/www/index.html y puse el nombre de la máquina. Con eso podía saber a que máquina me había enviado el balanceador. Tras saber la máquina en la que estaba, le quitaba la red y volvía a cargar la página para confirmar que el balanceador me dirigía al otro servidor real.

Cuando ya estaba seguro que los balanceadores me funcionaban correctamente, me puse a modificar apache para que estuvieran los archivos sincronizados y cree varios sitios virtuales. Lo primero que hice para sincronizarlo, fue parar el servicio y copiar los directorios en /opt, que es donde se encuentra montado el disco drbd1. Después borre los originales.

```
# /etc/init.d/apache2 stop
# mkdir /opt/etc
# mkdir /opt/var/
# cp –pr /etc/apache2 /opt/etc/
# cp –pr /var/www /opt/var
# rm –r /etc/apache2/*
# rm –r /var/www/*
```

Ahora tenemos que hacer que los ficheros sigan en sus directorios. Para esto decidí utilizar el mount con la opción --bind, para hacerlo permanente añadimos las siguientes líneas a el fichero /etc/fstab. Estas líneas tienen que ir después que la que añadimos para /opt:

```
/opt/etc/apache2 /etc/apache2 none rw,bind 0 0
/opt/var/www /var/www none rw,bind 0 0
```

Por último montamos las particiones e iniciamos el servicio.

```
# mount –a
# /etc/init.d/apache2 start
```

La parte de borrar y montar las particiones, tendremos que hacerla en los dos servidores, la de la copia, solo en uno. Tras esto pasamos a la configuración de los sitios virtuales. Para ello creamos dos ficheros de configuración. El primero es para saber en que máquina nos encontramos y el segundo para que nos direccione a drupal.

Antes que nada eliminé el fichero de configuración por defecto y creé el fichero ha. Este fichero nos dirige a una página donde nos informa de en qué servidor estamos, mediante php. No tenemos instalado php, podemos instalarlo para ver su correcto funcionamiento o esperar a la explicación de php. El fichero de configuración de ha es muy sencillo y tiene lo siguiente:

```
<VirtualHost *:80>
      DocumentRoot /var/www/ha
      <Directory />
            Options FollowSymLinks
            AllowOverride None
      </Directory>
</VirtualHost>
```

El segundo fichero de configuración que tenemos que crear en /etc/apache2/sites-available/ es el de drupal. Es muy parecido al anterior, solo especificamos que si entramos con la dirección drupal.ha nos dirija al directorio /var/www/drupal:

```
<VirtualHost *:80>
      ServerName drupal.ha
      DocumentRoot /var/www/drupal
      <Directory />
            Options FollowSymLinks
            AllowOverride None
      </Directory>
</VirtualHost>
```

Tras tener los ficheros de configuración tenemos que añadirlos, y eliminar el que trae por defecto. También tenemos que añadir los nuevos directorios.

```
# a2dissite 000-default
# a2ensite drupal
# a2ensite ha
# mkdir /var/www/drupal
# mkdir /var/www/ha
```

Para que el fichero de configuración ha sea el de por defecto entramos en /etc/apache2/sites-enabled/ y le ponemos 000 delante, así apache lo leera primero.

```
# cd /etc/apache2/sites-enabled/
# mv ha 000-ha
```

9.3. Puesta en marcha.

Tras esto iniciamos el apache.

```
# /etc/init.d/apache2 start
```

Con esto tenemos corriendo el apache, pero si reiniciamos la máquina, como los directorios se montan después de que arranque apache, dará error. Esto lo podemos solucionar parando el servicio del arranque y poniendolo en rc.local, después del mount -a. Para desactivar el apache del arranque yo lo realice con una herramienta llamada chkconfig. Los pasos son los siguientes:

```
# aptitude install chkconfig
# chkconfig –d apache2
```

Y por último añadimos la siguiente línea en el fichero /etc/rc.local, después del mount -a.

```
/etc/init.d/apache2
```

10. PHP5.

PHP es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas y puede ser incrustado en páginas HTML. Es usado normalmente para que el servidor ejecute un código. El cliente le envía la petición, el servidor la procesa y le envía la respuesta.

10.1. Instalación.

Para la instalación podemos hacerla desde los repositorios. En nuestro caso, como vamos a utilizar drupal tenemos que instalar un php específico. También tendremos que instalar php para que pueda leer de bases de datos postgresql.

```
# aptitude install php5 php5-gd php5-pgsql
```

10.2. Creación de una página de prueba.

Tras esto, creamos la página que nos sirve para comprobar que php está funcionando y poder saber en qué servidor nos encontramos. Está es la página:

```
cat /var/www/ha/index.php
<html>
 <head>
  <title>Servidor</title>
 </head>
 <body>
<?php phpinfo(); ?>
 </body>
</html>
```

Por último, reiniciamos apache

```
# /etc/init.d/apache2 restart
```

11. PostgreSQL.

Un servidor de base de datos es un programa que provee servicios de base de datos a otros programas u otras computadoras, como es definido por el modelo cliente-servidor.

En aplicaciones web, lo más utilizado es MySQL, aunque en algunas también podemos utilizar PostgreSQL. En mi caso para salir de lo más común he utilizado PostgreSQL.

PostgreSQL es un sistema de gestión de base de datos relacional orientada a objetos y libre, publicado bajo la licencia BSD.

11.1. Instalación.

La instalación la he realizado desde los repositorios.

```
# aptitude install postgresql
```

11.2. Configuración.

Los ficheros de configuración de postgresql, se encuentran en /etc/postgresql y las bases de datos en /var/lib/postgresql. En nuestro caso no tenemos que tocar ningún fichero de configuración.

Cuando tengamos instalado postgresql, tenemos que, al igual que con apache, sincronizarlos. Es lo mismo pero con diferentes directorios. Primero paramos la base de datos, copiamos los ficheros necesarios, borramos los antiguos, montamos lo nuevos e iniciamos el servicio.

```
# /etc/init.d/postgresql stop
# cp -pr /etc/postgresql /opt/etc
# cp -pr /etc/postgresql-common/ /opt/etc
# mkdir /opt/var/lib
# cp -pr /var/lib/postgresql/ /opt/var/lib
# rm -r /etc/postgresql/*
# rm -r /etc/postgresql-common/*
# rm -r /var/lib/postgresql/*
# echo '/opt/etc/postgresql /etc/postgresql none rw,bind 0 0'>>/etc/fstab
# echo '/opt/etc/postgresql-common /etc/postgresql-common none rw,bind 0
0'>>/etc/fstab
# echo '/opt/var/lib/postgresql /var/lib/postgresql none rw,bind 0
0'>>/etc/fstab
# mount -a
# /etc/init.d/postgresql start
```

Este proceso, al igual que con apache, tenemos que desactivarlo y añadirlo al rc.local:

```
# aptitude install chkconfig
# chkconfig -d apache2
```

Y añadimos la siguiente línea a /etc/rc.local, después del mount -a.

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

11.3. Creación de usuario y base de datos drupal.

Tras esto vamos a crear un usuario drupal y una base de datos drupal. Para crear el usuario, nos tenemos que loguear como usuario postgres y después entrar en la base de datos. Una vez creado el usuario, nos salimos y volvemos a iniciar postgres, esta vez con el usuario drupal, y creamos la base de datos.

```
# su postgres
$ psql
postgres=# CREATE USER drupal with
postgres-# password 'drupal'
postgres-# createdb;
\q
$ psql -U drupal -h localhost -d template1
template1=> CREATE DATABASE drupal;
```

12. Drupal.

Drupal es un sistema de gestión de contenido modular y muy configurable.

Es un programa de código abierto, con licencia GNU/GPL, escrito en PHP, desarrollado y mantenido por una activa comunidad de usuarios. Destaca por la calidad de su código y de las páginas generadas, el respeto de los estándares de la web, y un énfasis especial en la usabilidad y consistencia de todo el sistema.

El diseño de Drupal es especialmente idóneo para construir y gestionar comunidades en Internet. No obstante, su flexibilidad y adaptabilidad, así como la gran cantidad de módulos adicionales disponibles, hace que sea adecuado para realizar muchos tipos diferentes de sitios web.

Drupal es un sistema dinámico: en lugar de almacenar sus contenidos en archivos estáticos en el sistema de ficheros del servidor de forma fija, el contenido textual de las páginas y otras configuraciones son almacenados en una base de datos y se editan utilizando un entorno Web.

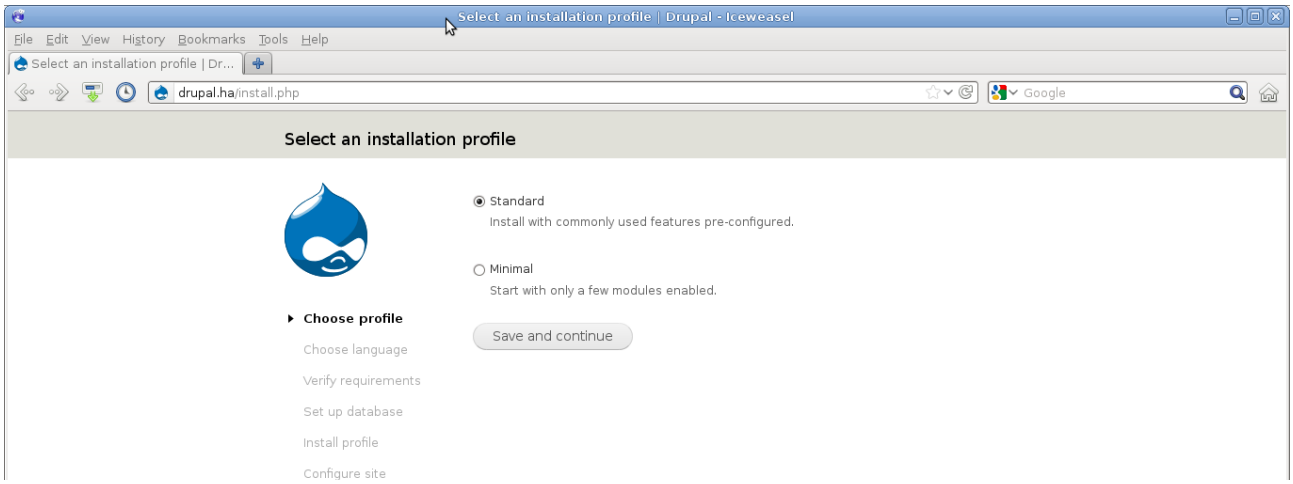
12.1. Instalación.

Para la instalación, me he descargado de la página principal, un archivo comprimido que descomprimiremos en /var/www. Se descomprime como drupal-7.14, lo renombraremos a drupal, ya que es como configuramos el apache. También nos descargaremos el idioma en español.

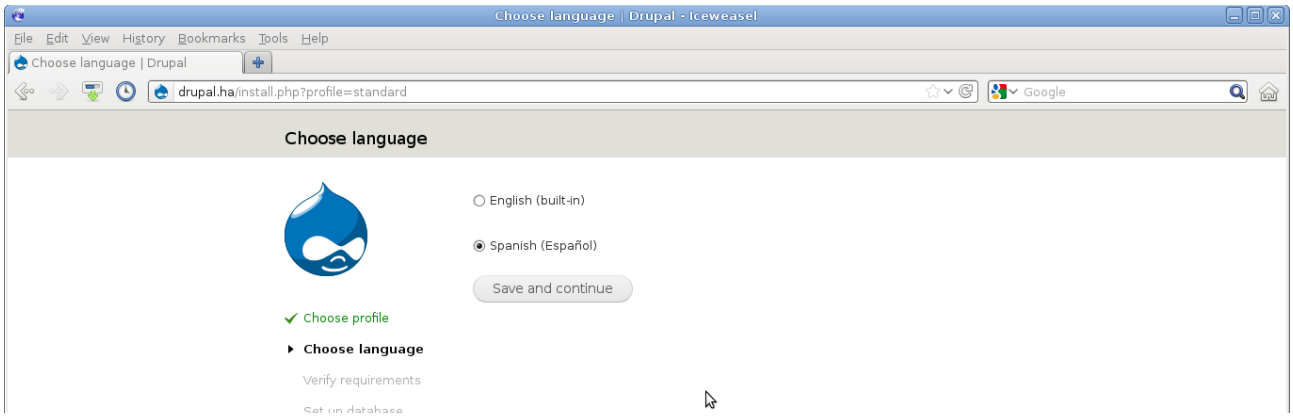
```
# wget “http://ftp.drupal.org/files/projects/drupal-7.14.tar.gz”
# wget “http://ftp.drupal.org/files/translations/7.x/drupal/drupal-
7.13.es.po”
# cd /var/www
# tar -zxvf /tmp/drupal-7.14.tar.gz
# mv /drupal-7.14 drupal
# cp /tmp/drupal-7.13.es.po /var/www/drupal/profiles/standard/translations/
# chown www-data.www-data /var/www
```

Ahora nos metemos en el navegador, ponemos la dirección de nuestro sitio y seguimos los pasos.

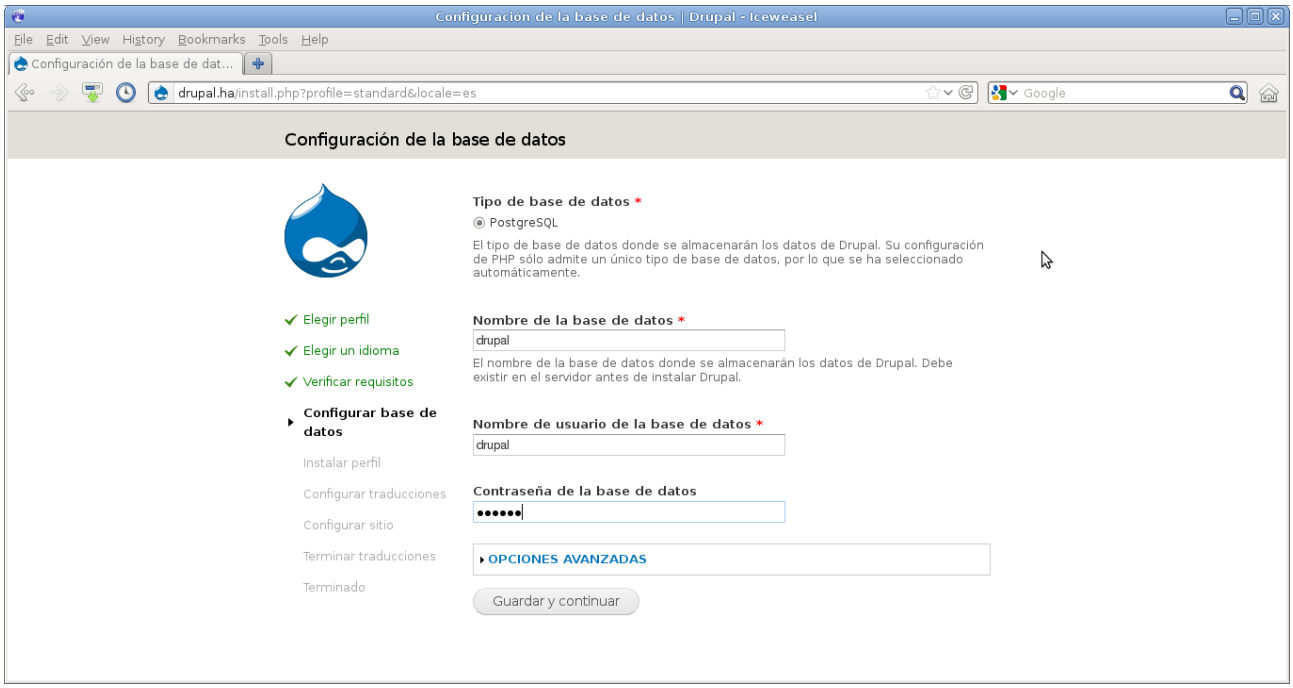
Instalación estándar.



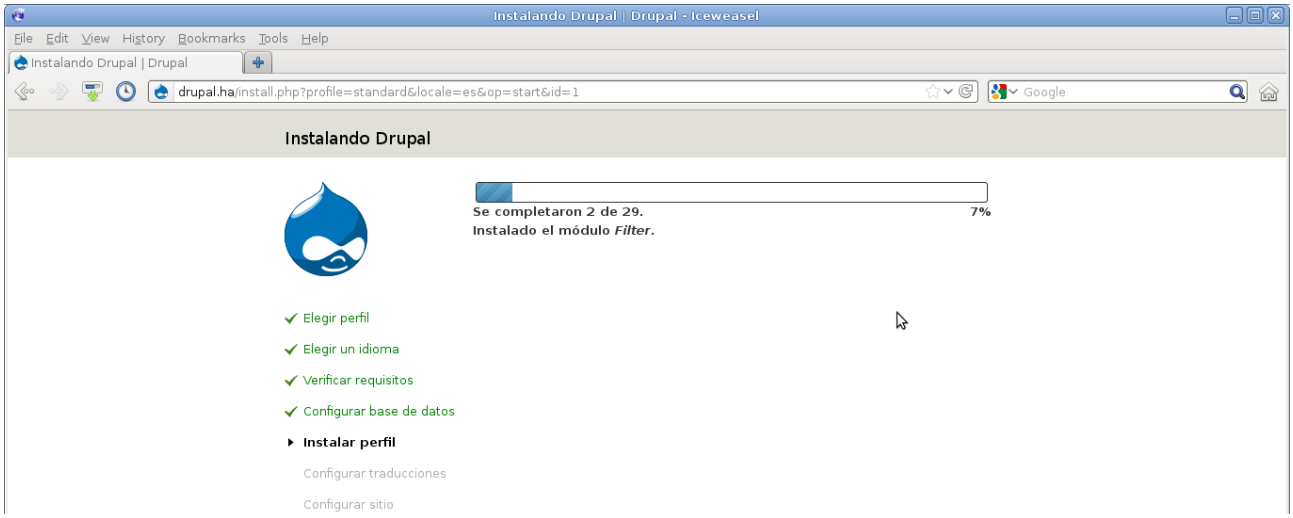
Como pusimos el módulo de español, podemos elegir español.



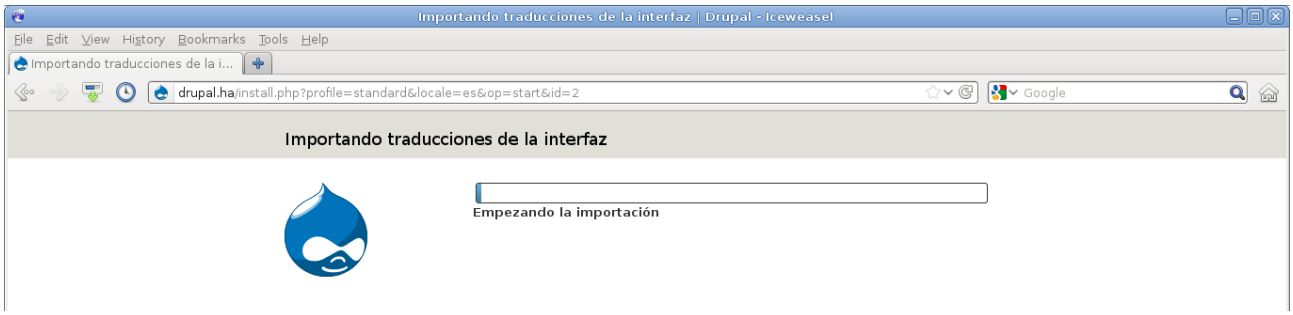
Rellenamos los datos de nuestra base de datos.



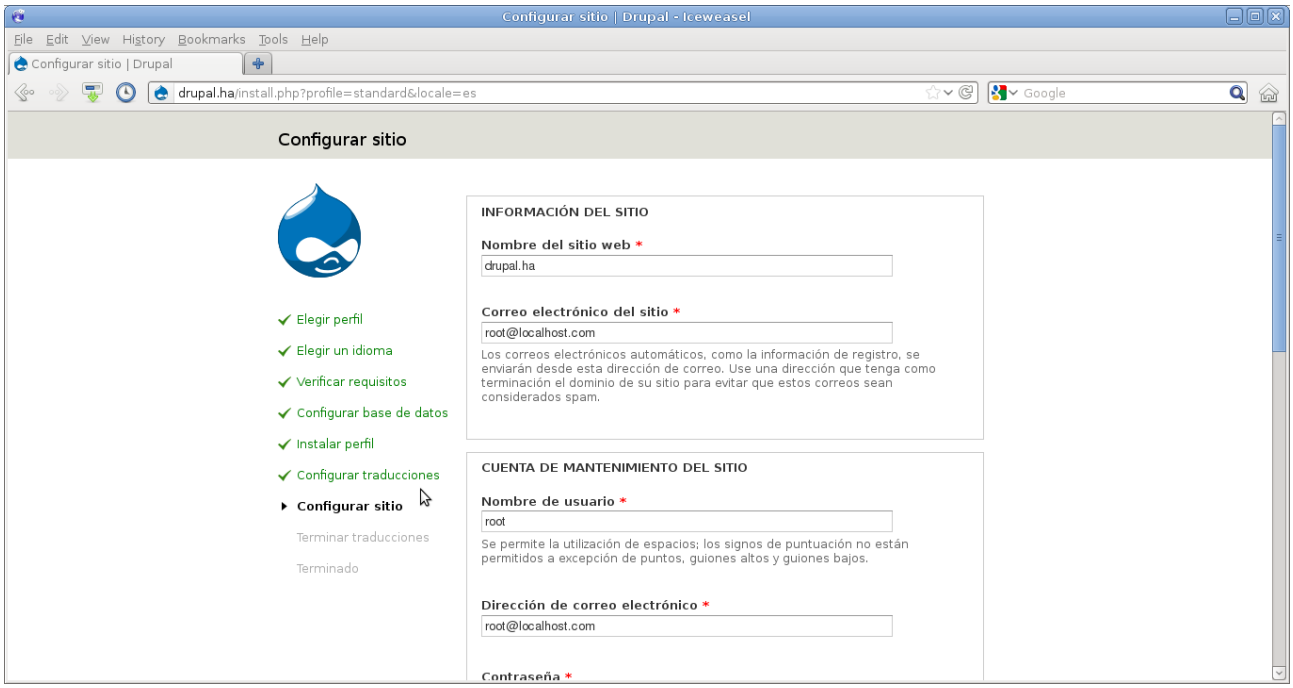
Y comienza la instalación.



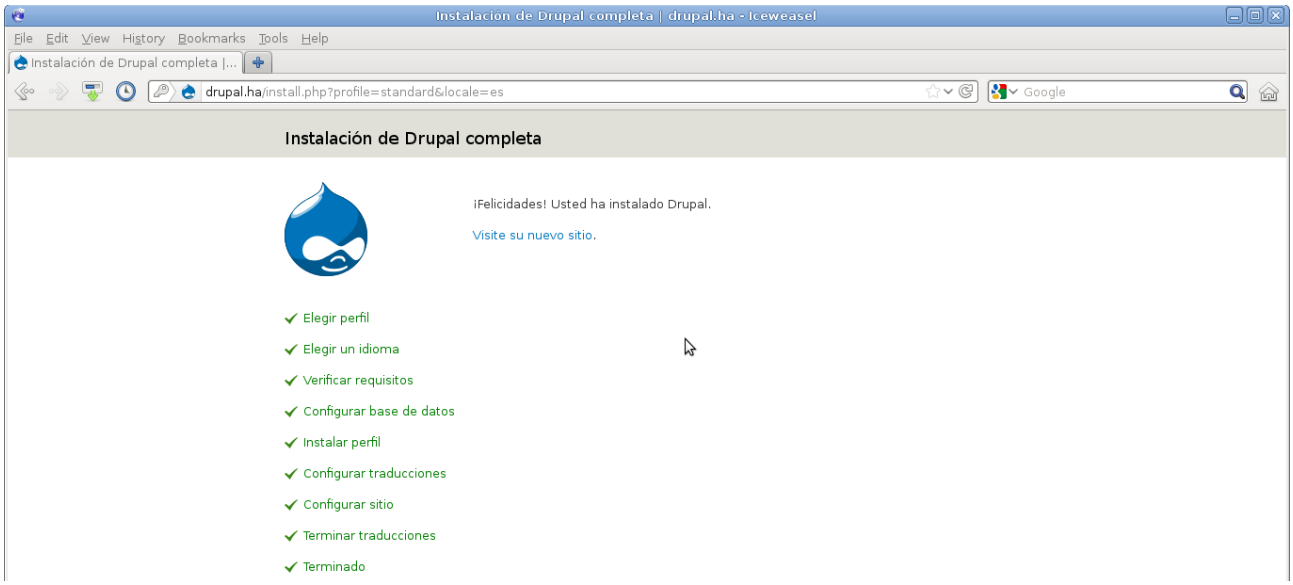
Después exporta el idioma.



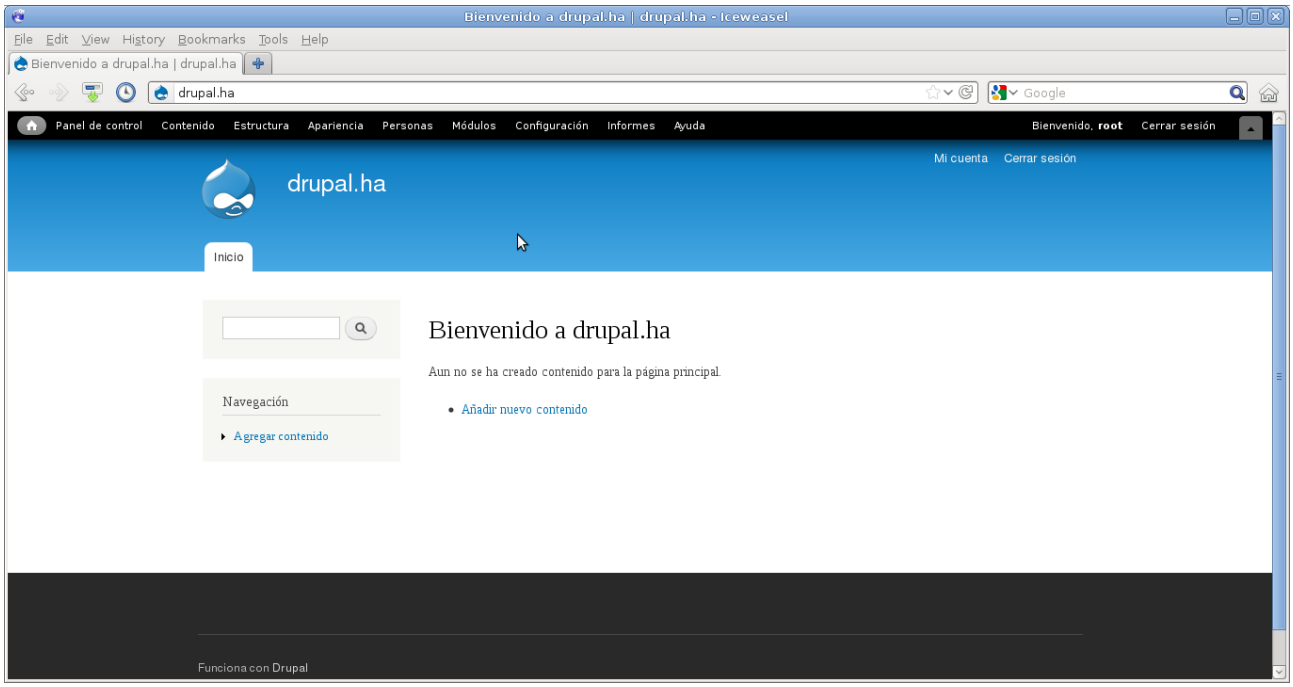
Ahora nos pide unos datos básicos para nuestra página. El usuario que pongamos, será administrador.



Y ya podemos ir a nuestro sitio.



Éste es el resultado.



13. Cosas a tener en cuenta.

A la hora de elaborar este documento he dejado a un lado los comandos y configuraciones básicas. Una de las cosas que tenemos que tener configurada bien es el /etc/hosts, para que apache no dé el error al no tener el FQDN correcto. Los ficheros de configuración de mis máquinas tienen el siguiente aspecto:

```
127.0.0.1   nombre.ha   nombre      localhost
192.168.10.1      zerg
192.168.10.2      protoss
192.168.10.3      terran
192.168.10.5      supermente
```

La configuración de la red no la he explicado. Podéis ver los ficheros de configuración en el último punto del proyecto.

Lo que mas dolor de cabeza me ha dado ha sido la sincronización de ficheros. Me ha dado muchos errores y los he ido solucionando todos. Tras solucionarlos todos, un día se me puso la sincronización en StandAlone. Buscando vi que cuando estaba en este estado podía ser por diferentes motivos. La descripción era:

> *"No tiene configuración de red disponible. El recurso aún no se ha conectado, ha sido desconectado, se ha caído de su conexión debido a un error de autenticación o el cerebro esta dividido(diferente información en los discos)."*

Tras leer por internet muchos foros, en uno decían que si en el inicio no se le daba tiempo a los discos de sincronizarse y apache o postgresql hacían alguna modificación, podía provocar esto.

En este proyecto no he probado esta solución, pero he puesto un script modificado, que

comprueba si los discos están sincronizados antes de arrancar los servicios. Si no están sincronizados enviar un correo a root@localhost y no arrancan los servicios.

14. Ficheros de configuración.

14.1. DRBD en protoss y terran

/etc/drbd.conf

```
include "drbd.d/global_common.conf";
include "drbd.d/*.res";
```

/etc/drbd.d/global_common.conf

```
global {
      usage-count no;
}
common {
      protocol C;
      handlers {
            pri-on-incon-degr "/usr/lib/drbd/notify-pri-on-incon-
degr.sh; /usr/lib/drbd/notify-emergency-reboot.sh; echo b > /proc/sysrq-
trigger ; reboot -f";
            pri-lost-after-sb "/usr/lib/drbd/notify-pri-lost-after-
sb.sh; /usr/lib/drbd/notify-emergency-reboot.sh; echo b > /proc/sysrq-
trigger ; reboot -f";
            local-io-error "/usr/lib/drbd/notify-io-error.sh;
/usr/lib/drbd/notify-emergency-shutdown.sh; echo o > /proc/sysrq-trigger ;
halt -f";
      }
      startup {
            become-primary-on both;
      }
      net {
            allow-two-primaries;
            after-sb-0pri discard-zero-changes;
            after-sb-1pri discard-secondary;
            after-sb-2pri disconnect;
      }
      syncer {
            rate 1000M;
      }
}
```

14.2. OCFS2 en protoss y terran.

/etc/ocfs2/cluster.conf

```
node:
      ip_port = 7777
      ip_address = 192.168.10.2
      number = 0
      name = protoss
      cluster = ocfs2
node:
      ip_port = 7777
      ip_address = 192.168.10.3
      number = 1
```

```
        name = terran
        cluster = ocfs2
cluster:
        node_count = 2
        name = ocfs2
```

/etc/default/o2cb

```
O2CB_ENABLED=true
O2CB_BOOTCLUSTER=ocfs2
O2CB_HEARTBEAT_THRESHOLD=1
O2CB_IDLE_TIMEOUT_MS=5000
O2CB_KEEPALIVE_DELAY_MS=1000
O2CB_RECONNECT_DELAY_MS=2000
```

14.3. LVS en zerg y supermente.

/etc/default/ipvsadm

```
# Do not edit! Use 'dpkg-reconfigure ipvsadm'.
AUTO="true"
DAEMON="none"
```

14.4. Keepalived en zerg.

/etc/keepalived/keepalived.conf

```
global_defs {
      lvs_id LVS1
}

virtual_server 10.0.0.4 80 {
   delay_loop 1
   lb_algo wrr
   lb_kind DR
   protocol TCP
   sorry_server 10.0.0.6 80
  real_server 192.168.10.2 80 {
    weight 1
    TCP_CHECK {
       connect_port 80
       connect_timeout 1
    }
  }
  real_server 192.168.10.3 80 {
    weight 1
    TCP_CHECK {
       connect_port 80
       connect_timeout 1
    }
  }
}

vrrp_instance VI_1 {
  state MASTER
  interface eth0
  lvs_sync_daemon_inteface eth0
  virtual_router_id 51
  priority 150
  advert_int 1
```

```
  smtp_alert
  authentication {
     auth_type PASS
     auth_pass example
  }
  virtual_ipaddress {
     10.0.0.4
  }
}
```

14.5. Keepalived en supermente.

/etc/keepalived/keepalived.conf

```
global_defs {
      lvs_id LVS2
}

virtual_server 10.0.0.4 80 {
   delay_loop 1
   lb_algo wrr
   lb_kind DR
   protocol TCP
   sorry_server 10.0.0.6 80
  real_server 192.168.10.2 80 {
    weight 1
    TCP_CHECK {
       connect_port 80
       connect_timeout 1
    }
  }
  real_server 192.168.10.3 80 {
    weight 1
    TCP_CHECK {
       connect_port 80
       connect_timeout 1
    }
  }
}

vrrp_instance VI_1 {
  state BACKUP
  interface eth0
  lvs_sync_daemon_inteface eth0
  virtual_router_id 51
  priority 100
  advert_int 3
  smtp_alert
  authentication {
     auth_type PASS
     auth_pass example
  }
  virtual_ipaddress {
     10.0.0.4
  }
}
```

14.6. Apache, servidores virtuales.

/etc/apache2/sites-available/ha

```
<VirtualHost *:80>
      ServerAdmin webmaster@localhost

      DocumentRoot /var/www/ha
      <Directory />
            Options FollowSymLinks
            AllowOverride None
      </Directory>
</VirtualHost>
```

/etc/apache2/sites-available/drupal

```
<VirtualHost *:80>
      ServerAdmin webmaster@localhost
      ServerName drupal.ha
      DocumentRoot /var/www/drupal
      <Directory />
            Options FollowSymLinks
            AllowOverride None
      </Directory>
</VirtualHost>
```

14.7. Fichero fstab de terran y protoss.

/etc/fstab

```
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
proc            /proc           proc    defaults        0       0
# / was on /dev/vda1 during installation
UUID=63476114-ae72-43f0-b9d5-b062e68f7d7c /               ext3
errors=remount-ro 0       1
# swap was on /dev/vda5 during installation
UUID=6fa962ce-3cd6-4164-95d5-3e7297fc4875 none            swap    sw
0       0
/dev/scd0       /media/cdrom0   udf,iso9660 user,noauto     0       0
## HA ##
/dev/drbd1 /opt ocfs2 rw,_netdev,heartbeat=local 0 0
/opt/etc/postgresql /etc/postgresql none rw,bind 0 0
/opt/etc/postgresql-common /etc/postgresql-common none rw,bind 0 0
/opt/etc/apache2 /etc/apache2 none rw,bind 0 0
/opt/var/lib/postgresql /var/lib/postgresql none rw,bind 0 0
/opt/var/www /var/www none rw,bind 0 0
```

14.8. Ficheros de red.

Zerg:
/etc/network/interfaces

```
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet static
      address 10.0.0.1
      netmask 255.255.255.0
      network 10.0.0.0
      broadcast 10.0.0.255
      gateway 10.0.0.128
      dns-nameservers 10.0.0.128
auto eth1
```

```
iface eth1 inet static
      address 192.168.10.1
      netmask 255.255.255.0
      network 192.168.10.0
      broadcast 192.168.10.255
```

Protos:

/etc/network/interfaces

```
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet static
      address 10.0.0.2
      netmask 255.255.255.0
      network 10.0.0.0
      broadcast 10.0.0.255
      gateway 10.0.0.128
      dns-nameservers 10.0.0.128
auto eth1
iface eth1 inet static
      address 192.168.10.2
      netmask 255.255.255.0
      network 192.168.10.0
      broadcast 192.168.10.255
auto lo:0
iface lo:0 inet static
      address 10.0.0.4
      netmask 255.255.255.255
```

Terran:

/etc/network/interfaces

```
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet static
      address 10.0.0.3
      netmask 255.255.255.0
      network 10.0.0.0
      broadcast 10.0.0.255
      gateway 10.0.0.128
      dns-nameservers 10.0.0.128
auto eth1
iface eth1 inet static
      address 192.168.10.3
      netmask 255.255.255.0
      network 192.168.10.0
      broadcast 192.168.10.255
auto lo:0
iface lo:0 inet static
        address 10.0.0.4
        netmask 255.255.255.255
```

Supermente:

/etc/network/interfaces

```
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet static
      address 10.0.0.5
      netmask 255.255.255.0
```

```
      network 10.0.0.0
      gateway 10.0.0.128
      dns-nameservers 10.0.0.128
auto eth1
iface eth1 inet static
      address 192.168.10.5
      netmask 255.255.255.0
      network 192.168.10.0
```

14.9. Sincronización al arranque.

```
#!/bin/sh
contador=0;
estado=$(cat /proc/drbd|grep cs|cut -d : -f 3|cut -d ' ' -f 1|sed -n -e
'1p');
estado2=$(cat /proc/drbd|grep cs|cut -d : -f 3|cut -d ' ' -f 1|sed -n -e
'2p');
while [ "$estado" != "Connected" -o "$estado2" != "Connected" ];
  do
   if [ $contador -eq 300 ]; then break
   else
   sleep 1;
   contador=$(($contador+1));
   echo $contador;
   estado=$(cat /proc/drbd|grep cs|cut -d : -f 3|cut -d ' ' -f 1|sed -n -e
'1p');
   estado2=$(cat /proc/drbd|grep cs|cut -d : -f 3|cut -d ' ' -f 1|sed -n -e
'2p')
   fi
  done
if [ $estado = "Connected" -a $estado2 = "Connected" ]; then
  echo "Discos replicados y servicios levantados"
  mount -a;
  /etc/init.d/apache2 start;
  /etc/init.d/postgresql start;
 else
  echo "Ha habido un problema en el arranque de servicios. El modo
automatico no ha
funcionado"|sendmail root@localhost;
fi
```

15. Bibliografía.

http://es.wikipedia.org

http://luismido.wikidot.com/lvs-keepalived-ubuntu

http://www.estrellateyarde.org/discover/virtualizacion/clusters/clusters-ha-con-lvs/cluster-lvs-keepalived-en-linux

http://www.drbd.org/

http://httpd.apache.org/docs/2.0/mod/directives.html

http://www.postgresql.org/

http://www.elastix.org/dokuwiki/doku.php?id=cluster_con_elastix_with_english

http://www.ipcorp.com.ar/blog/2009/10/01/cluster-de-alta-disponibilidad-de-servidores-postgresql-con-drbd/

http://linuxsilo.net/articles/postgresql-pgpool.html

http://www.ibiblio.org/pub/linux/docs/LuCaS/Manuales-LuCAS/doc-curso-salamanca-clustering/html/clustering-ha.html#id2499627

http://ubanov.wordpress.com/2009/08/10/instalacion-servidores-con-ocfs2-drbd-lvm2-raid1-sobre-debian-howto-install/

http://www.austintek.com/LVS/LVS-HOWTO/mini-HOWTO/LVS-mini-HOWTO.html

http://www.go2linux.org/lvs-linux-virtual-server

http://www.ultramonkey.org/papers/lvs_tutorial/html/

http://linuxsan.wordpress.com/2008/02/04/cluster-de-alta-disponibilidad-heartbeat2-xen-cluster-con-drbd8-y-ocfs2/

http://realtechtalk.com/UbuntuDebian_DRBD_80_Setup_Guide-1085-articles

http://gnudocs.dyndns.info/?q=blogs/%5B3/drbd-aoe-ocfs2

http://padmavyuha.blogspot.com.es/2011/07/configuring-drbd-with-ocfs2-on-suse.html

http://gcharriere.com/blog/?p=339

http://www.linuxvirtualserver.org/docs/ha/keepalived.html

http://www.howtoforge.com/setting-up-a-high-availability-load-balancer-with-haproxy-keepalived-on-debian-lenny