CLUSTER ALTA DISPONIBILIDAD

COROSYNC-PACEMAKER-DRBD

Antonio David Tejero Galán
I.E.S. Gonzalo Nazareno

Índice

1. Introducción

1.1 Objetivo

El proyecto consiste en crear un clúster en alta disponibilidad que garantice el servicio web a través de un servidor Apache. También implantaremos un servicio DRBD (Distributed Replicated Block Device).

1.2 ¿Qué es un Clúster de Alta Disponibilidad?

Un **clúster HA** (High Availability) es un sistema orientado a ofrecer y garantizar servicios en Alta Disponibilidad, es decir, con un alto grado de fiabilidad y de continuidad operativa.

Se basa en **máquinas redundantes** (o nodos) que asumen el servicio cuando algún componente del sistema falla.

Un clúster HA debe ser capaz de detectar cualquier fallo de hardware o de software, reiniciar la aplicación en otro nodo y **mantener el servicio** sin intervención de operador alguno, garantizando la **integridad de los datos** del clúster.

1.3 Tipos de configuración

El tamaño más habitual de un clúster HA es de dos nodos, ya que es el mínimo exigido para disponer de redundancia. Las dos configuraciones más comunes en los clusters de dos nodos son:

- **Cluster Activo/Activo**
  - Aprovecha mucho mejor los recursos físicos.
  - Permite repartir mejor la carga entre los nodos.
- **Activo/Pasivo**
  - Configuración más sencilla

1.4 Conceptos básicos

- **Failover**: capacidad de recuperarse de un fallo desplegando los servicios en otro nodo. “Cluster HA” = “failover clusters”
- **Heartbeat:** pulso o “latido” mediante el cual se mantiene la comunicación entre los nodos del cluster. Si el nodo activo no responde al latido, el nodo pasivo toma el control y despliega de inmediato los servicios replicados
- **Split-brain:** se produce cuando los enlaces de red que unen a los nodos entre sí caen, pero los nodos siguen operando. Se dice entonces que el clúster se ha “partido”. Puede causar corrupción de datos en sistemas de almacenamiento compartido.

- **Quorum:** es un mecanismo para prevenir el split-brain. Se asigna un voto a cada nodo y se le permite operar si obtiene mayoría de votos. Con un clúster de dos nodos, la mayoría son dos votos, por lo que no es posible activar el quorum.

2. Configuración Previa

2.1 Nodos

El clúster está formado por dos máquinas virtuales sobre KVM. La configuración de ambas máquinas es la siguiente:

- DRBD1 (Master):
  - Debian Wheezy
  - Dos discos duros. El primero de 6GB con el sistemas y otro de 250M
  - Una tarjeta de Red
  - IP: 10.0.0.8

- DRBD2 (Slave)
  - Debian Wheezy
  - Dos discos duros. El primero de 6GB con el sistemas y otro de 250M
  - Una tarjeta de Red
  - IP: 10.0.0.9

2.2 Esquema de red

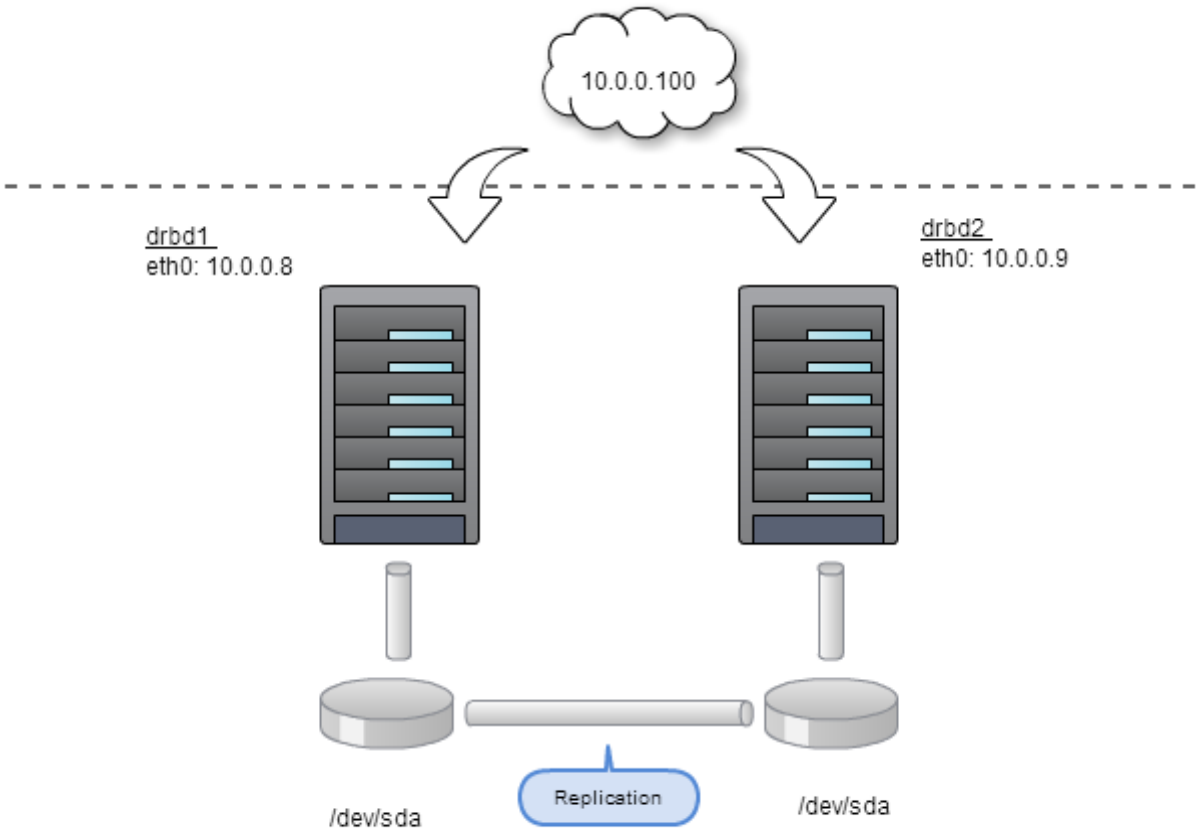
La red en la que están ambos nodos es la 10.0.0.0

Como se ha indicado anteriormente, las ips de los nodos son:

-DRBD1: 10.0.0.8

-DRBD2: 10.0.0.9

La IPV (IP Virtual) es la 10.0.0.100 Por lo tanto el esquema de red será el siguiente:



2.3 Configuración de Red

El fichero de configuración de Debian es ***/etc/network/interfaces***. Por lo tanto para configurar nuestras dos máquinas modificaremos dicho fichero.

El fichero de configuración de DRBD1 es el siguiente:

```
auto lo
iface lo inet loopback

# The primary network interface
allow-hotplug eth0
iface eth0 inet static
     address 10.0.0.8
     netmask 255.255.255.0
     network 10.0.0.0
     broadcast 10.0.0.255
```

```
     gateway 10.0.0.1
     # dns-* options are implemented by the resolvconf package, if
installed
     dns-nameservers 10.0.0.1
```

Como vemos, la interfaz de red es eth0, se iniciará automáticamente. También está iniciada la interfaz de loopback, que sirve para trabajar localmente.

El fichero de configuración de red de DRBD2 es el siguiente:

```
auto lo
iface lo inet loopback

# The primary network interface
allow-hotplug eth0
iface eth0 inet static
     address 10.0.0.9
     netmask 255.255.255.0
     network 10.0.0.0
     broadcast 10.0.0.255
     gateway 10.0.0.1
     # dns-* options are implemented by the resolvconf package, if
installed
     dns-nameservers 10.0.0.1
```

Algo importante es que ambos nodos puedan resolver sus nombres, y como no contamos con un servidor DNS, vamos a hacer que resuelvan sus nombres gracias al fichero ***/etc/hosts*** que quedaría de la siguiente forma en ambos nodos:

```
127.0.0.1 localhost
10.0.0.8  drbd1.example.com   drbd1
10.0.0.9  drbd2.example.com   drbd2
```

Como vemos escribimos la dirección IP de cada nodo junto con su nombre. Por último, para que los cambios de red se ejecuten, reiniciamos el servicio en las dos máquinas.

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

2.4 Instalación de paquetes

Ahora vamos a instalar los paquetes que necesitaremos para la configuración del clúster. En principio necesitaremos pacemaker y corosync. Así que en ambos nodos ejecutamos las siguientes líneas. Antes de instalar los paquetes debemos actualizar nuestros repositorios:

```
#aptitude update
#aptitude install pacemaker corosync
```

También necesitaremos Apache y php. Por lo que instalamos los paquetes:

```
#aptitude install apache2 php5
```

Por último también necesitaremos las herramientas necesarias para DRBD, para ello instalamos el siguiente paquete:

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

3. Pacemaker y Corosync

3.1 ¿Qué es Pacemaker?

Es un proyecto que parte de Heartbeat, es básicamente su evolución. ¿Y que es Heartbeat?

Es una utilidad que simplemente envía un latido de corazón. De hecho, en español, sería latido de corazón, un simple pulso, o paquete pequeño de datos a uno o más equipos. De ésta manera, permite que en un clúster se **realice un monitoreo automático, para saber si el nodo está vivo y los servicios corriendo en el**, haciendo que en caso de que un nodo "muera", automáticamente el o los nodos vivos tomen ese servicio, lo inicien y continúen trabajando.

3.2 ¿Qué es Corosync?

Básicamente, Corosync permite el **intercambio de mensaje**s entre los nodos.

3.3 Configuración Pacemaker y Corosync

Como hemos indicado anteriormente, nuestro nodo principal es DRBD1, por lo tanto vamos a

crear en ésta máquina el archivo de autenticación de Corosync.

```
root@drbd1:~# corosync-keygen
```

Automáticamente le da permisos, por lo que no es necesario que se los demos. Después hay que pasar ésta llave al nodo cliente (DRBD2) por medio de SSH, por eso debemos darle permiso de escritura en el nodo cliente. Lo hacemos de la siguiente forma:

```
root@drbd1:~# cd /etc/
root@drbd1:/etc# chmod -R 755 corosync/
```

```
root@drbd2:~# cd /etc/
root@drbd2:/etc# chmod -R 755 corosync/
```

Ahora copiamos ese archivo al nodo secundario:

```
root@drbd1:~# scp /etc/corosync/authkey
root@drbd2:/etc/corosync/authkey
authkey 100% 128                 0.1KB/s 00:00
```

El archivo de configuración de Corosync es ***/etc/corosync/corosync.conf***. Lo editaremos en los dos nodos y añadiremos la red que vamos a usar. Para ello en ***bindnetaddr*** indicamos nuestra red (10.0.0.0). El fichero en ambos nodos quedaría de la siguiente forma:

```
root@drbd1:~# cat /etc/corosync/corosync.conf
# Please read the openais.conf.5 manual page

totem {
     version: 2

     # How long before declaring a token lost (ms)
     token: 3000

     # How many token retransmits before forming a new
configuration
```

```
    token_retransmits_before_loss_const: 10

    # How long to wait for join messages in the membership
protocol (ms)
    join: 60

    # How long to wait for consensus to be achieved before
starting a new round of membership configuration (ms)
    consensus: 3600

    # Turn off the virtual synchrony filter
    vsftype: none

    # Number of messages that may be sent by one processor on
receipt of the token
    max_messages: 20

    # Limit generated nodeids to 31-bits (positive signed
integers)
    clear_node_high_bit: yes

    # Disable encryption
    secauth: off

    # How many threads to use for encryption/decryption
    threads: 0

    # Optionally assign a fixed node id (integer)
    # nodeid: 1234

    # This specifies the mode of redundant ring, which may be
none, active, or passive.
    rrp_mode: none

    interface {
```

```
          # The following values need to be set based on your
environment
          ringnumber: 0
          bindnetaddr: 10.0.0.0
          mcastaddr: 226.94.1.1
          mcastport: 5405
     }
}

amf {
     mode: disabled
}

service {
     # Load the Pacemaker Cluster Resource Manager
     ver:       0
     name:      pacemaker
}

aisexec {
        user:   root
        group:  root
}

logging {
        fileline: off
        to_stderr: yes
        to_logfile: no
        to_syslog: yes
     syslog_facility: daemon
        debug: off
        timestamp: on
        logger_subsys {
                subsys: AMF
```

```
                debug: off
                tags: enter|leave|trace1|trace2|trace3|trace4|
trace6
        }
}
```

También tendremos que indicar que Corosync esté iniciado por defecto y de ésta forma se inicie el demonio. Para ello nos dirigimos al fichero ***/etc/default/corosync*** y **START** lo ponemos a **YES**.

```
root@drbd1:/etc# cat /etc/default/corosync
START=yes
```

```
root@drbd2:/etc# cat /etc/default/corosync
START=yes
```

Por último, **reiniciamos los servicios** para que se apliquen los cambios guardados en los dos nodos:

```
root@drbd1:~# service corosync restart
[ ok ] Restarting corosync daemon: corosync.
```

```
root@drbd2:/etc# service corosync restart
[ ok ] Restarting corosync daemon: corosync.
```

Podemos ver el estado del clúster con el comando ***crm status.***

```
root@drbd1:~# crm status
============
Last updated: Thu Dec 11 21:39:50 2014
Last change: Thu Dec 11 21:38:16 2014 via crmd on drbd1
Stack: openais
Current DC: drbd1 - partition with quorum
Version: 1.1.7-ee0730e13d124c3d58f00016c3376a1de5323cff
2 Nodes configured, 2 expected votes
```

```
0 Resources configured.
============
Online: [ drbd2 drbd1 ]
```

```
root@drbd2:~# crm status
============
Last updated: Thu Dec 11 21:39:30 2014
Last change: Thu Dec 11 21:38:16 2014 via crmd on drbd1
Stack: openais
Current DC: drbd1 - partition with quorum
Version: 1.1.7-ee0730e13d124c3d58f00016c3376a1de5323cff
2 Nodes configured, 2 expected votes
0 Resources configured.
============
Online: [ drbd2 drbd1 ]
```

Como podemos apreciar, en éstos momentos en el clúster tenemos dos nodos (drbd1 y drbd2) y ambos están ***Online***.

3.4 IPV (IP Virtual)

IPV es una IP Virtual, es decir, no existe una tarjeta de red física con dicha dirección asignada. El clúster tendrá asignada ésa IP, accediendo a dicha IP tendremos acceso al servidor que en ése momento esté ofreciendo el servicio.

Por lo tanto ahora vamos a configurar la **IP Virtual** como recurso al clúster. La dirección que vamos a configurar es **10.0.0.100,** para ello, en primer lugar, desactivamos el mecanismo STONITH, que se utiliza como se describe al inicio del documento para parar un nodo que esté dando problemas y así evitar comportamientos inadecuados en el clúster.

```
root@drbd1:~# crm configure property stonith-enabled=false
root@drbd2:~# crm configure property stonith-enabled=false
```

Para realizar las siguientes configuraciones usaremos **crm**, por lo tanto las modificaciones que realicemos con crm en un nodo se aplicarán también al otro nodo del clúster.

Ahora si, definiremos un recurso para los dos nodos para indicar la IPV. El comando es el siguiente:

```
root@drbd1:~# crm configure primitive FAILOVER-ADDR
ocf:heartbeat:IPaddr2 params ip="10.0.0.100" nic="eth0" op monitor
interval="10s" meta is-managed="true"
```

Como vemos, el nombre del recurso es FAILOVER-ADDR y la IPV es 10.0.0.100.

Podemos ver la configuración con el comando **crm_mon.** La salida es la siguiente:

```
============
Last updated: Thu Dec 11 21:46:33 2014
Last change: Thu Dec 11 21:46:04 2014 via cibadmin on drbd1
Stack: openais
Current DC: drbd2 - partition with quorum
Version: 1.1.7-ee0730e13d124c3d58f00016c3376a1de5323cff
2 Nodes configured, 2 expected votes
1 Resources configured.
============
Online: [ drbd2 drbd1 ]
FAILOVER-ADDR (ocf::heartbeat:IPaddr2): Started drbd1
```

Sin embargo, drbd1, que es el nodo principal, no pasa a ofrecer inmediatamente el recurso, porque no hay quorum (al inicio de la documentación se define éste termino) en el clúster. El quorum no se puede efectuar con dos máquinas, necesitaríamos como mínimo tres nodos. Por lo tanto debemos ignorar las decisiones basadas en el quorum. Ésto se lo indicamos al clúster con el siguiente comando:

```
root@drbd1:~# crm configure property no-quorum-policy=ignore
```

3.5 Configuración de recurso Apache

Ahora vamos a añadir el recurso de Apache al clúster, para ello escribimos el siguiente comando:

```
root@drbd1:~# crm configure primitive P_APACHE
ocf:heartbeat:apache params configfile="/etc/apache2/apache2.conf"
statusurl="http://localhost/server- status" op monitor
interval="40s"
```

P_APACHE es el nombre del recurso que vamos a compartir.

Lo siguiente que vamos a hacer es indicar a pacemaker y corosync que deben ir comprobando la disponibilidad de los dos nodos y si están “online” mostrar su contenido. Ésto se hace de la siguiente forma:

```
root@drbd1:~# crm configure order START_ORDER inf: FAILOVER-ADDR
P_APACHE
```

FAILOVER-ADDR es el nombre de la IPV y P_APACHE el recurso de apache.

E indicamos el orden de los diferentes nodos. La sintaxis del comando es: *“crm configure location NUMERACION NOMBRE_RECURSO_IP_VIRTUAL 100:NOMBRE_NODO”*.

```
root@drbd1:~# crm configure location L_IP_NODE001 FAILOVER-ADDR
100: drbd1
root@drbd1:~# crm configure location L_IP_NODE002 FAILOVER-ADDR
100: drbd2
```

Ahora, con el comando crm configure show podremos ver las configuraciones que hemos ido realizando y cómo todo está correctamente:

```
root@drbd1:~# crm configure show

node drbd1

node drbd2

primitive FAILOVER-ADDR ocf:heartbeat:IPaddr2 \

params ip="10.0.0.100" nic="eth0" \

op monitor interval="10s" \

meta is-managed="true"
```

```
primitive P_APACHE ocf:heartbeat:apache \
params configfile="/etc/apache2/apache2.conf"
statusurl="http://localhost/server-" status \
op monitor interval="40s"
location L_IP_NODE001 FAILOVER-ADDR 100: drbd1
location L_IP_NODE002 FAILOVER-ADDR 100: drbd2
order START_ORDER inf: FAILOVER-ADDR P_APACHE
property $id="cib-bootstrap-options" \
dc-version="1.1.7-ee0730e13d124c3d58f00016c3376a1de5323cff" \
cluster-infrastructure="openais" \
expected-quorum-votes="2" \
stonith-enabled="false" \
no-quorum-policy="ignore"
```

Con esto ya tendríamos configurado el servidor Apache en HA. Podríamos configurar más opciones en nuestro clúster, pero con los que aquí se muestran ya son necesarios para que funciones correctamente.

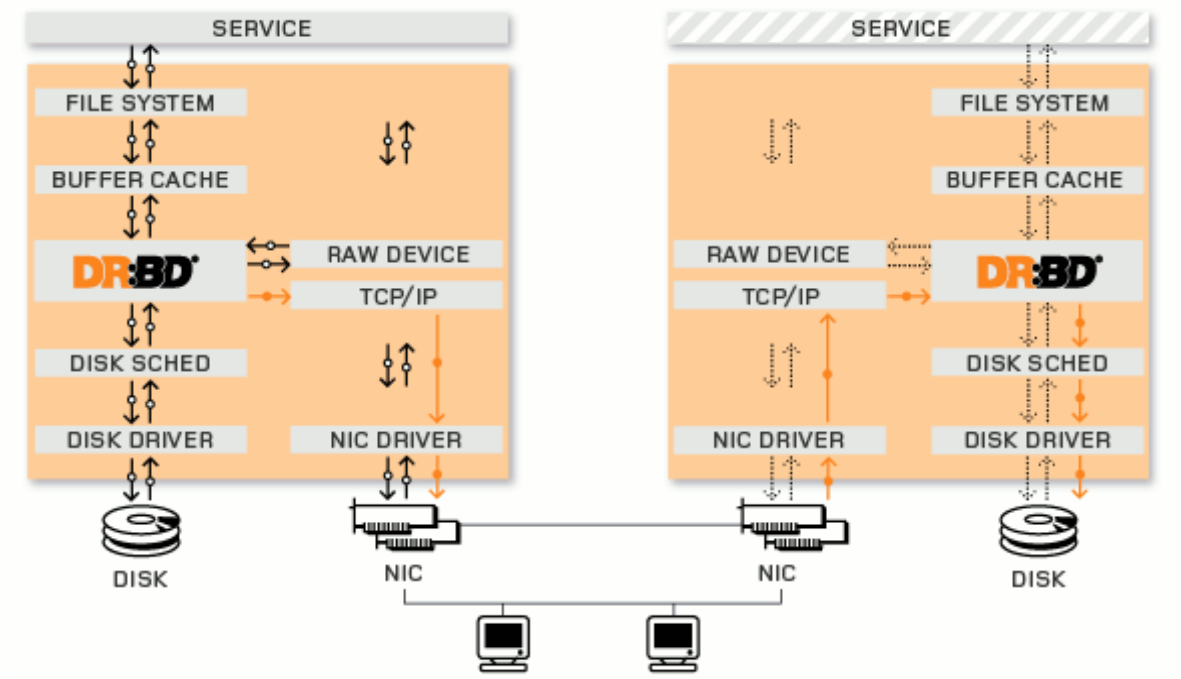
4. DRBD

4.1 ¿Qué es DRBD?

DRBD (Distributed Replicated Block Device) se refiere a los dispositivos de bloque concebido como una edificación en bloque, agrupados para formar un cluster de alta disponibilidad (HA). Esto se hace por medio de reflejo completo de un dispositivo de bloque asignado a través de una red.

4.2 ¿Cómo funciona?

En la siguiente imagen, las dos cajas de color naranja representan dos servidores que forman un clúster de HA. Las cajas contienen los componentes habituales de un núcleo Linux: sistema de archivos, buffer caché, planificador de disco, controladores de disco, etc. Las flechas en negro reflejan el flujo de datos entre estos componentes.



En definitiva, DRBD es un software el cual construye un mirror de discos sobre LAN. Ésto quiere decir que además de nuestra configuración de disco por hardware raid0, raid1, podemos tener nuestros datos respaldados en otro servidor remoto.

Hay varios tipos de configuración de DRBD. En ésta ocasión nosotros vamos a configurar la C. Es la que más se suele usar. La configuración es de las más sencillas.

La gran ventaja de éste tipo de configuración es que tendremos el **disco COMPLETO replicado** en el otro nodo. La desventaja es que este tipo de configuración **no permite que el volumen esté montado en dos nodos a la vez**, para asegurarse de que no se produzca ningún tipo de replicación de datos.

En el caso de nuestro servidor no sería necesario, en principio, que estuviera montado en los dos nodos, ya que nuestra configuración es de Activo/Pasivo.

4.3 Configuración DRBD

Como se indicó al inicio de la documentación, tenemos un disco (aparte del que se encuentra el sistema instalado) de 200M. Éste disco lo usaremos para DRBD.

Si ejecutamos el comando ***fdisk -l*** podremos ver los distintos discos que tenemos en el equipo.

```
root@drbd1:~# fdisk -l

Disk /dev/vda: 6291 MB, 6291456000 bytes

16 heads, 63 sectors/track, 12190 cylinders, total 12288000
sectors
```

```
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00050c6d
 Device Boot Start End Blocks Id System
/dev/vda1 * 2048 11681791 5839872 83 Linux
/dev/vda2 11683838 12285951 301057 5 Extended
/dev/vda5 11683840 12285951 301056 82 Linux swap
/ Solaris
Disk /dev/sda: 264 MB, 264241152 bytes
255 heads, 63 sectors/track, 32 cylinders, total 516096 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000
Disk /dev/sda doesn't contain a valid partition table
```

```
root@drbd2:~# fdisk -l
Disk /dev/vda: 6291 MB, 6291456000 bytes
16 heads, 63 sectors/track, 12190 cylinders, total 12288000
sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00050c6d
 Device Boot Start End Blocks Id System
/dev/vda1 * 2048 11681791 5839872 83 Linux
/dev/vda2 11683838 12285951 301057 5 Extended
/dev/vda5 11683840 12285951 301056 82 Linux swap
/ Solaris
Disk /dev/sda: 264 MB, 264241152 bytes
255 heads, 63 sectors/track, 32 cylinders, total 516096 sectors
Units = sectors of 1 * 512 = 512 bytes
```

```
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000
Disk /dev/sda doesn't contain a valid partition table
```

Como vemos en ambos nodos, el disco es /dev/sda, utiliza éste nombre porque el disco donde se encuentra instalado el sistema es un volumen lógico, entonces nos lo nombra como si fuera el primer disco. También podemos apreciar como ninguno de los dos tienen ninguna partición creada.

El fichero de configuración de DRBD es /etc/drbd.conf. En éste fichero encontramos el siguiente contenido:

```
root@drbd1:~# cat /etc/drbd.conf
# You can find an example in
/usr/share/doc/drbd.../drbd.conf.example
include "drbd.d/global_common.conf";
include "drbd.d/*.res";
```

Como vemos nos está haciendo un include del archivo global_common.conf y de todos los archivos que estén dentro de drbd.d/ con extensión .res.

Por lo tanto lo que haremos será crear un fichero “prueba.res” en el directorio /etc/drbd.d/ con la configuración del recurso que creemos para DRBD.

El contenido de prueba.res es el siguiente:

```
resource prueba
{
  device    /dev/drbd0;            #Dispositivo DRBD
  meta-disk internal;

  protocol  C;               #Protocolo que vamos a usar

  on drbd1               #Configuración en drbd1
  {
    address   10.0.0.2:7789;  #IP y puerto que se utilizará en
drbd1
```

```
    disk      /dev/sda;       #Dispositivo donde se encuentra el
bloque DRBD
  }
  on drbd2              #Configuración en drbd2
  {
    address   10.0.0.3:7789;  #Dispositivo donde se encuentra el
bloque DRBD
    disk      /dev/sda;
  }
}
```

Como vemos la configuración es básica. Podríamos usar muchas más opciones, pero con estas funciona corrrectamente DRBD.

*NOTA: Como se puede apreciar, ha habido un cambio de direcciones IP. Esto es debido a que he montado en varias ocasiones DRBD y la última cambié de IP.

La configuración del fichero prueba.res también se tiene que **realizar exactamente igual** en drbd2.

También vamos a cargar el módulo DRBD en el kernel, en las dos máquinas, de la siguiente forma:

```
root@drbd1:~# modprobe drbd
root@drbd1:~# lsmod | grep drbd
drbd 193891 0
lru_cache 12969 1 drbd
```

```
root@drbd2:~# modprobe drbd
root@drbd2:~# lsmod | grep drbd
drbd 193891 0
lru_cache 12969 1 drbd
```

Ahora, en uno de los nodos iniciamos los metadatos del disco usando el siguiente comando:

```
#drbdadm create-md prueba
md_offset 263188480
al_offset 263155712
```

```
bm_offset 263147520
Found ext3 filesystem
 5839872 kB data area apparently used
 256980 kB left usable by current configuration
Device size would be truncated, which would corrupt data and
result in
'access beyond end of device' errors.
You need to either
 * use external meta data (recommended)
 * shrink that filesystem first
 * zero out the device (destroy the filesystem)
Operation refused.
Command 'drbdmeta 0 v08 /dev/sda1 internal create-md' terminated
with exit code 40
drbdadm create-md prueba: exited with code 40
```

Luego también lo hacemos en el otro nodo, pero como ya tenemos iniciado los metadatos, nos avisa de ello y nos pedirá confirmación para modificar el sistema de ficheros, como se mueestra a continuación:

```
root@drbd2:~# drbdadm create-md prueba
md_offset 263188480
al_offset 263155712
bm_offset 263147520
Found some data
 ==> This might destroy existing data! <==
Do you want to proceed?
[need to type 'yes' to confirm] yes
Writing meta data...
initializing activity log
NOT initialized bitmap
New drbd meta data block successfully created.
```

Ahora vamos a levantar el recurso DRBD:

```
#drbdadm up prueba
```

Los discos aún no están sincronizados. A continuación si tenemos datos en uno de los discos debemos ejecutar en el nodo que tenga los datos lo siguiente:

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

De esta forma se copiarán los bloques del disco del equipo donde lo ejecutemos al disco del equipo secundario.

En el caso que en ninguno de los discos de los dos equipos tengamos datos podemos simplemente marcar los discos como iguales con:

```
#drbdadm – --clear-bitmap new-current-uuid r0
```

De esta forma nos ahorramos traspasar unos bloques que realmente no contienen nada y a medida que se modifiquen se irán sincronizando realmente.

Ahora, en el nodo secundario procedemos a levantar también el recurso DRBD, procediendo a la sincronización de los datos con el nodo principal:

```
#drbdadm up prueba
```

En el fichero ***/proc/drbd*** podemos ver el proceso de sincronización:

```
root@drbd1:/home/usuario# cat /proc/drbd
version: 8.3.11 (api:88/proto:86-96)
srcversion: F937DCB2E5D83C6CCE4A6C9
 0: cs:SyncSource ro:Primary/Secondary ds:UpToDate/Inconsistent C
r-----
 ns:7808 nr:0 dw:0 dr:8472 al:0 bm:0 lo:0 pe:1 ua:0 ap:0 ep:1
wo:f oos:3063164
[>....................] sync'ed: 0.4% (3063164/3070844)K
finish: 0:38:17 speed: 1,280 (1,280) K/sec
```

```
root@drbd2:/home/usuario# cat /proc/drbd
version: 8.3.11 (api:88/proto:86-96)
srcversion: F937DCB2E5D83C6CCE4A6C9
 0: cs:SyncTarget ro:Secondary/Primary ds:Inconsistent/UpToDate C
r-----
 ns:0 nr:39808 dw:39680 dr:0 al:0 bm:2 lo:1 pe:0 ua:1 ap:0 ep:1
wo:f oos:3031164
[>....................] sync'ed: 1.4% (3031164/3070844)K
finish: 0:37:15 speed: 1,336 (1,280) want: 250 K/sec
```

Una vez finalice el replicado de los datos, podemos volver a consultar el fichero ***/proc/drbd:***

```
root@drbd1:~# cat /proc/drbd
version: 8.3.11 (api:88/proto:86-96)
srcversion: F937DCB2E5D83C6CCE4A6C9
 0: cs:Connected ro:Primary/Secondary ds:UpToDate/UpToDate C
r-----
    ns:1 nr:4 dw:5 dr:475 al:0 bm:1 lo:0 pe:0 ua:0 ap:0 ep:1 wo:f
oos:0
```

```
root@drbd2:~# cat /proc/drbd
version: 8.3.11 (api:88/proto:86-96)
srcversion: F937DCB2E5D83C6CCE4A6C9
 0: cs:Connected ro:Secondary/Primary ds:UpToDate/UpToDate C
r-----
    ns:4 nr:1 dw:1 dr:4 al:0 bm:1 lo:0 pe:0 ua:0 ap:0 ep:1 wo:f
oos:0
```

Como vemos, el fragmento Primary/Secondary indica que es el nodo principal y el Secondary/Primary es el secundario. También UpToDate/UpToDate indica que ambos están sincronizados.

Por decirlo así tenemos una red 'RAID1', un **dispositivo de bloques /dev/drbd0** (que está formado por /dev/sda1 de drbd1 y /dev/sda1 de drbd2).

Ahora vamos a darle el sistema de ficheros ext3 y **montarlo en /var/www**. Esta operación **sólo se puede realizar en el nodo principal**.

```
root@drbd1:~# mkfs.ext3 /dev/drbd0
mke2fs 1.42.5 (29-Jul-2012)
Etiqueta del sistema de ficheros=
OS type: Linux
Tamaño del bloque=1024 (bitácora=0)
Tamaño del fragmento=1024 (bitácora=0)
Stride=0 blocks, Stripe width=0 blocks
64256 inodes, 256980 blocks
12849 blocks (5.00%) reserved for the super user
Primer bloque de datos=1
Número máximo de bloques del sistema de ficheros=67371008
32 bloque de grupos
8192 bloques por grupo, 8192 fragmentos por grupo
2008 nodos-i por grupo
Respaldo del superbloque guardado en los bloques:
8193, 24577, 40961, 57345, 73729, 204801, 221185
Allocating group tables: hecho
Escribiendo las tablas de nodos-i: hecho
Creating journal (4096 blocks): hecho
Escribiendo superbloques y la información contable del sistema de
ficheros: 0/3hecho
```

Y ahora lo montamos en /var/www.

```
root@drbd1:~# mount /dev/drbd0 /var/www/
root@drbd1:~# mount
/dev/drbd0 on /var/www type ext3
(rw,relatime,errors=continue,user_xattr,acl,barrier=1,data=ordered
)
```

Como vemos en la última línea de la salida del comando **mount**, nos muestra como efectivamente ya está montado.

Para comprobar que la replicación se efectúa correctamente, vamos a crear unos ficheros en /var/www de drbd1 y veremos como se replican en drbd2.

Primero creamos los ficheros de prueba:

```
root@drbd1:~# touch /var/www/prueba1.txt
root@drbd1:~# echo
'<html><head></head><body><h1>prueba</h1></body></html>' >
/var/www/drbd2.html
root@drbd1:~# ls -l /var/www/
total 13
-rw-r--r-- 1 root root 55 dic 11 19:42 drbd2.html
drwx------ 2 root root 12288 dic 11 19:34 lost+found
-rw-r--r-- 1 root root 0 dic 11 19:41 prueba1.txt
```

Ahora desmontamos /dev/drbd0

```
root@drbd1:~# umount /var/www
```

Pasamos drbd1 a secundario. Y en ***/proc/drbd*** podemos comprobar como ahora drbd1 también es nodo secundario:

```
root@drbd1:~# drbdadm secondary prueba
root@drbd1:~# cat /proc/drbd
version: 8.3.11 (api:88/proto:86-96)
srcversion: F937DCB2E5D83C6CCE4A6C9
 0: cs:Connected ro:Secondary/Secondary ds:UpToDate/Diskless C
r-----
 ns:0 nr:0 dw:25423 dr:2949 al:34 bm:0 lo:0 pe:0 ua:0 ap:0 ep:1
wo:f oos:256980
```

A continuación pasamos a DRBD2 a nodo primario con el siguiente comando:

```
root@drbd2:~# drbdadm primary prueba
```

Y comprobamos en ambos nodos como la configuración es correcta:

```
root@drbd2:~# cat /proc/drbd
version: 8.3.11 (api:88/proto:86-96)
srcversion: F937DCB2E5D83C6CCE4A6C9
 0: cs:Connected ro:Primary/Secondary ds:Diskless/UpToDate C
r-----
 ns:0 nr:664 dw:0 dr:0 al:0 bm:0 lo:0 pe:0 ua:0 ap:0 ep:1 wo:f
oos:0

root@drbd1:~# cat /proc/drbd
version: 8.3.11 (api:88/proto:86-96)
srcversion: F937DCB2E5D83C6CCE4A6C9
 0: cs:Connected ro:Secondary/Primary ds:UpToDate/Diskless C
r-----
 ns:664 nr:0 dw:25423 dr:3613 al:34 bm:0 lo:0 pe:0 ua:0 ap:0
ep:1 wo:f oos:256980
```

Y por último montamos el dispositivo:

```
root@drbd2:~# mount /dev/drbd0 /var/www/
root@drbd2:~# ls -l /var/www/
total 13
-rw-r--r-- 1 root root 55 dic 11 19:42 drbd2.html
drwx------ 2 root root 12288 dic 11 19:34 lost+found
-rw-r--r-- 1 root root 0 dic 11 19:41 prueba1.txt
```

Como vemos ahora tenemos el contenido perfectamente replicado en drbd2, por lo que el funcionamiento de DRBD es correcto.

5. Añadiendo DRBD como recurso al clúster

DRBD sólo puede estar montado en el nodo principal como ya hemos visto. Por lo tanto, para que DRBD fuera práctico en nuestro clúster tendríamos que añadirlo al clúster como recurso, de forma que cuando el nodo principal se "caiga", automáticamente DRBD se configure como principal en el segundo nodo y se monte en el directorio indicado.

Para añadir DRBD como recurso al clúster utilizamos el siguiente comando:

#crm configure primitive drbdprueba ocf:linbit:drbd params drbd_resource="prueba"

A continuación deberemos indicar las limitaciones de DRBD:

- **clone-max**="2": Sólo puede estar en dos instancias
- **clone-node-max**="1" Una instancia por nodo
- **master-max**="1": Máximo 1 solo master
- **master-node-max**="1": Máximo 1 master por nodo
- **notify**="true": Se debe notificar al recurso que ocurre al peer

Así que el comando queda de la siguiente forma:

```
#crm configure primitive drbdprueba ocf:linbit:drbd params
drbd_resource="prueba"
```

Ahora **limpiamos los mensajes de error** que se hayan generado mientras el DRBD no pertenecía a un **set master/slave** con:

```
#crm resource cleanup drbdprueba
```

Con el comando crm status podemos ver el estado del clúster actualmente y ver como está configurado el set master/slave:

```
root@drbd1:~# crm status
============
Last updated: Tue Dec 16 21:06:08 2014
Last change: Tue Dec 16 21:05:11 2014 via crmd on drbd1
Stack: openais
```

```
Current DC: drbd1 - partition with quorum
Version: 1.1.7-ee0730e13d124c3d58f00016c3376a1de5323cff
2 Nodes configured, 2 expected votes
4 Resources configured.
============

Online: [ drbd1 drbd2 ]

 FAILOVER-ADDR (ocf::heartbeat:IPaddr2):    Started drbd1
 Master/Slave Set: ms_drbdprueba [drbdprueba]
     Masters: [ drbd1 ]
     Slaves: [ drbd2 ]
```

```
root@drbd2:~# crm status
============
Last updated: Tue Dec 16 21:07:42 2014
Last change: Tue Dec 16 21:05:11 2014 via crmd on drbd1
Stack: openais
Current DC: drbd1 - partition with quorum
Version: 1.1.7-ee0730e13d124c3d58f00016c3376a1de5323cff
2 Nodes configured, 2 expected votes
4 Resources configured.
============

Online: [ drbd1 drbd2 ]

 FAILOVER-ADDR (ocf::heartbeat:IPaddr2):    Started drbd1
 Master/Slave Set: ms_drbdprueba [drbdprueba]
     Masters: [ drbd1 ]
     Slaves: [ drbd2 ]
```

```
root@drbd2:~# cat /proc/drbd
```

```
version: 8.3.11 (api:88/proto:86-96)
srcversion: F937DCB2E5D83C6CCE4A6C9
 0: cs:Connected ro:Secondary/Primary ds:UpToDate/UpToDate C
r-----
    ns:0 nr:204756 dw:204756 dr:0 al:0 bm:13 lo:0 pe:0 ua:0 ap:0
ep:1
```

Por lo tanto, ya tendríamos DRBD implementado en el clúster, pero además de que en caso de fallo en el Master, el Slave pasára a ser primario, también tendría que montarse el dispositivo en el directorio /var/www.

Por lo tanto vamos a indicar al clúster que debe gestionar también el montaje y desmontaje del sistema de ficheros mediante **heartbeat:Filesystem.**

Con estas opciones añadimos el **recurso** para que **monte el sistema de ficheros**:

```
#crm configure primitive fsprueba ocf:heartbeat:Filesystem params
device="/dev/drbd0" directory="/var/www" fstype="ext3"
```

Podemos ver con el comando **crm status** como la última configuración se ha realizado correctamente:

```
# crm status
============
Last updated: Tue Dec 16 21:12:03 2014
Last change: Tue Dec 16 21:11:59 2014 via cibadmin on drbd1
Stack: openais
Current DC: drbd1 - partition with quorum
Version: 1.1.7-ee0730e13d124c3d58f00016c3376a1de5323cff
2 Nodes configured, 2 expected votes
5 Resources configured.
============

Online: [ drbd1 drbd2 ]

 FAILOVER-ADDR (ocf::heartbeat:IPaddr2):     Started drbd1
 Master/Slave Set: ms_drbdprueba [drbdprueba]
```

```
     Masters: [ drbd1 ]
     Slaves: [ drbd2 ]
 fsprueba (ocf::heartbeat:Filesystem):  Started drbd1
```

En este caso se ha levantado bien, pero podría haber generado algún error si se hubiera levantado en el otro nodo. Por lo tanto, debemos indicar explícitamente que siempre debe **montar el sistema de ficheros después del master/slave set de DRB**:

```
#crm configure order drbd-fs mandatory: ms_drbdprueba:promote
fsprueba:start
```

Además, siempre debe estar en el mismo nodo el **montaje del sistema de ficheros** y el **master del recurso DRBD**:

```
#crm configure colocation fsprueba-drbdprueba INFINITY: fsprueba
ms_drbdprueba:Master
```

A continuación limpiamos los errores del recurso que monta el sistema de ficheros por si se ha producido algún error anteriormente:

```
#crm resource cleanup fsprueba
```

Y volvemos a comprobar el estado del clúster:

```
# crm status
============
Last updated: Tue Dec 16 21:12:03 2014
Last change: Tue Dec 16 21:11:59 2014 via cibadmin on drbd1
Stack: openais
Current DC: drbd1 - partition with quorum
Version: 1.1.7-ee0730e13d124c3d58f00016c3376a1de5323cff
2 Nodes configured, 2 expected votes
5 Resources configured.
============
```

```
Online: [ drbd1 drbd2 ]

 FAILOVER-ADDR (ocf::heartbeat:IPaddr2):     Started drbd1
 Master/Slave Set: ms_drbdprueba [drbdprueba]
     Masters: [ drbd1 ]
     Slaves: [ drbd2 ]
 fsprueba (ocf::heartbeat:Filesystem):  Started drbd1
```

Llegados a este punto, ya tenemos montado nuestro clúster HA con DRBD. Ahora si en el nodo principal se produce algún tipo de problema, automáticamente en el segundo nodo pasaría a ser el primario y se montaría el sistema de ficheros. En el momento en el que el nodo principal volviera a estar disponible, se configuraría como primario de nuevo.

Ahora haremos algunas comprobaciones.

6. Comprobaciones

Vamos a poner a prueba a nuestro clúser. Lo que vamos a hacer es crear una página en el volumen DRBD en PHP donde nos muestre un mensaje con el nombre del host en el que se está ejecutando ese código.

Indicar que la IPV en esta ocasión es la 10.0.0.20 ya que como expuse anteriormente he realizado el proyecto en diferentes máquinas, y la última utilicé otra IPV.

El contenido de index.php es el siguiente:

```
root@drbd1:~# cat /var/www/index.php
<html>
<head></head>
<body>
<?$hostname= exec('hostname');
?>
<h2><?echo $hostname?></h2>
</body>
</html>
```

Y el estado de DRBD es el siguiente:

```
root@drbd1:~# cat /proc/drbd
version: 8.3.11 (api:88/proto:86-96)
srcversion: F937DCB2E5D83C6CCE4A6C9
 0: cs:Connected ro:Primary/Secondary ds:UpToDate/UpToDate C
r-----
    ns:1 nr:4 dw:5 dr:477 al:0 bm:1 lo:0 pe:0 ua:0 ap:0 ep:1 wo:f
oos:0
```

Vemos como DRBD1 es el nodo primario y como el secundario también está disponible y bien configurado:

```
root@drbd2:~# cat /proc/drbd
version: 8.3.11 (api:88/proto:86-96)
srcversion: F937DCB2E5D83C6CCE4A6C9
 0: cs:Connected ro:Secondary/Primary ds:UpToDate/UpToDate C
r-----
    ns:4 nr:1 dw:1 dr:4 al:0 bm:1 lo:0 pe:0 ua:0 ap:0 ep:1 wo:f
oos:0
```

También comprobamos el estado del cluster:

```
# crm status
============
Last updated: Thu Dec 18 01:54:36 2014
Last change: Wed Dec 17 20:54:33 2014 via crmd on drbd1
Stack: openais
Current DC: drbd2 - partition with quorum
Version: 1.1.7-ee0730e13d124c3d58f00016c3376a1de5323cff
2 Nodes configured, 2 expected votes
5 Resources configured.
============

Online: [ drbd1 drbd2 ]
```

```
FAILOVER-ADDR (ocf::heartbeat:IPaddr2):     Started drbd1
Master/Slave Set: ms_drbdprueba [drbdprueba]
    Masters: [ drbd1 ]
    Slaves: [ drbd2 ]
fsprueba (ocf::heartbeat:Filesystem):  Started drbd1
```

Podemos ver como los dos nodos **están "online"**

Una vez comprobado el estado de los dos nodos, vamos abrir en el navegador la IPV y nos muestra el siguiente contenido:

Como vemos se está ejecutando en DRBD1, ya que están "online" los dos nodos y DRBD1 es el primario.

Ahora vamos a apagar DRBD1 y vemos de nuevo el estado del clúster.

```
root@drbd2:~# crm status
============
Last updated: Thu Dec 18 01:58:13 2014
Last change: Wed Dec 17 20:54:33 2014 via crmd on drbd1
Stack: openais
Current DC: drbd2 - partition WITHOUT quorum
Version: 1.1.7-ee0730e13d124c3d58f00016c3376a1de5323cff
2 Nodes configured, 2 expected votes
5 Resources configured.
============
```

```
Online: [ drbd2 ]
OFFLINE: [ drbd1 ]

 FAILOVER-ADDR (ocf::heartbeat:IPaddr2):     Started drbd2
 Master/Slave Set: ms_drbdprueba [drbdprueba]
     Masters: [ drbd2 ]
     Stopped: [ drbdprueba:0 ]
 fsprueba (ocf::heartbeat:Filesystem):  Started drbd2
```

Se puede apreciar como **DRBD2 está "online"** mientras que **DRBD1 está "offline".**

También vemos el estado de DRBD en ***/proc/drbd***

```
root@drbd2:~# cat /proc/drbd
version: 8.3.11 (api:88/proto:86-96)
srcversion: F937DCB2E5D83C6CCE4A6C9
 0: cs:WFConnection ro:Primary/Unknown ds:UpToDate/DUnknown C
r-----
    ns:4 nr:2 dw:3 dr:462 al:0 bm:1 lo:0 pe:0 ua:0 ap:0 ep:1 wo:f
oos:4
```

Nos muestra como drbd2 ahora pasa a ser primario y como no hay nodo secundario muestra "Unknown".

Por lo tanto se evidencia como el recurso DRBD en el clúster está funcionando correctamente. Pero también realizamos la configuración para que el sistema de ficheros se montara automáticamente en /var/www. Así que vamos a comprobar que así sea.

```
root@drbd2:~# ls /var/www/
index.php  lost+found  prueba
```

Vemos también como se ha montado perfectamente el sistema de ficheros. Por último vamos a abrir la IPV en el navegador:

De ésta forma queda patente como el clúster está funcionando correctamente junto con DRBD.

7. Conclusiones

Las conclusiones una vez finalizado es que, algo que ya sabemos, es que la HA es algo básico para sistemas que estén en producción.

DRBD es bastante útil, y fácil de configurar. Podemos evitar grandes problemas con la simple replicación.

Me ha quedado claro el funcionamiento del clúster y la herramienta DRBD, que al principio me costó entender, pero una vez entendido su funcionalidad es mucho más fácil llegar a configurarlo.

El objetivo con DRBD era comprenderlo e iniciarme un poco con el, el objetivo se ha cumplido.

Finalmente, creo que el proyecto ha sido provechoso, tanto por aprender cosas nuevas, como por que éstas cosas sean de máxima utilidad.

Para finalizar me gustaría resaltar que al realizar el proyecto, no sólo he podido aprender sobre alta disponibilidad y DRBD, aunque anteriormente en cierta forma otros proyectos hayan fracasado, he tenido la oportunidad de conocer más a fondo Openstack y herramientas como Vagrant. Así que en general lo tomo como algo muy positivo.

8. Referencias

- http://www.drbd.org/
- http://www.wikipedia.com
- http://www.drbd.org/
- http://www.linux-party.com/index.php/component/content/article/99-cloudcomputing/9045-
- http://clusterlabs.org/doc/en-US/Pacemaker/1.1-pcs/html-single/Clusters_from_Scratch/index.html#idm254658409520