**DRBD**

Distributed Replicated Block Device

1. **Descripcion del proyecto**

Este documento describe la instalación del servidor y configuración de los distintos sistemas que implementan los servicios de sincronización de directorios entre dos o más nodos. En el mismo se describen las características más importantes de la instalación y procedimientos de configuración de los paquetes involucrados.

* 1. **Objetivos del proyecto**

El objetivo consiste en poder sincronizar el contenido de uno o varios directorios entre dos o más nodos, esta configuración aplica para cualquier directorio o partición usando LVM2 como manejador de volúmenes de disco.

* 1. **SOLUCION**

La solución consiste en la instalación de un del paquete DRBD en dos servidores, en esta oportunidad se utilizan dos servidores, db2 IP 128.2.105.99 ( testing ) suchugo 128.2.105.199 ( maquina de pruebas en el datacenter ).

**2.** **Conceptos Básicos**

**2.1 Sincronizacion DRBD**

DRBD nos va a permitir sincronizar el contenido de un directorio en particular, para nuestro ejemplo utilizamos **/tecnol** que es un Logical Volume de LVM2 que pertenece al Volume Grupe **rootvg.** El producto funciona en una modalidad de activo-pasivo o primario y secundario, cuando un nodo está configurador como primario solo él puede leer los datos ( montar/escribir ) la partición o directorio que queremos sincronizar.

En modo background mientras tanto automáticamente los datos se están replicando al nodo que está configurado como pasivo o secundario.

Ante la caída de uno de los nodos, o también intencionalmente podemos definir en cualquier momento quien va acceder a los datos, manualmente podemos hacer un “failover” o si uno de los nodos se cae, debemos manualmente indicar que el que esté activo pasa a ser primario y poder acceder a los datos. Decimos manualmente porque no disponemos de un software de clustering que haga el trabajo de hacer el switch entre uno y otro.

**2.2 Kernel y Userland**

Necesitamos un kernel con soporte para DRBD, ya que la versión que disponemos en las sucursales está bastante desactualizado, tenemos que descargar uno más nuevo para tal fin. Siguiendo la recomendación de DRBD en su documentación, descargamos un kernel acorde a la versión de DRBD que vayamos a instalar, en este caso la recomendación es:

**Version 8.3.8.1 de DRBD, utilizar la versión de kernel 2.6.36.**

El kernel nos va a proveer de un modulo que maneja el los dispositivos en **/proc/drbdX,** ya que el paquete de DRBD son las utilidades de userland, o las utilidades administrativas que nos van a permitir manejar el software a nivel de usuario.

1. **INSTALCION Y CONFIGURACION**

A continuación se describen los procedimientos para instalar y compilar el kernel y drbd.

* 1. **Instalación del Kernel**

Descargamos el kernel linux-2.6.36.4 desde [www.kernel.org](http://www.kernel.org) del siguiente link:

<ftp://ftp.kernel.org/pub/linux/kernel/v2.6/linux-2.6.36.4.tar.gz>

Copiar el archivo a /usr/src y descomprimir ahí de la siguiente manera: tar xvfz linux-2.6.36.4.tar.gz lo cual no deja la siguiente estructura de directorio:

**drwxrwxr-x 24 root root 4096 May 18 10:33 linux-2.6.36.4**

Creamos un link simbolico de la siguiente manera:

**ln -s linux-2.6.36.4/ linux**

Si listamos nuevamente el directorio veremos lo siguiente:

**lrwxrwxrwx 1 root root 14 May 16 08:37 linux -> linux-2.6.36.4**

Antes de compilar el kernel debemos tener instaladas las librerias ncurses-devel, para verificarlo ejecutamos el siguiente comando:

**rpm –qa|grep ncurses**

ncurses-devel-5.4-13

ncurses-5.4-13

Ahora estamos en condiciones de poder compilar nuestro kernel, como ya disponemos de un archivo de configuración (**.config** ) vamos a copiar el mismo dentro del directorio linux. Ver abajo que tenemos el archivo Config, copiar el contenido del mismo como .config

****

Una vez copiado el archivo en el directorio, ejecutamos las siguientes sentencias en el siguiente orden.

**make**

**make modules\_install**

**make install**

Con esto completamos la instalación del nuestro nuevo kernel, lo que este proceso hace es, compila los fuentes y los modulos ( **make** ), instala los modulos en /lib/modules/kernel-version ( **make modules\_install** ), instala el nuevo kernel en /boot y modifica grub.conf añadiendo la nueva entrada de nuestro kernel ( **make install** ). El instalador no modifica que kernel queremos bootear por defecto en grub.conf, asi que debemos modificar lo siguiente en **/boot/grub/menú.lst**

**default=1** cambiar a **default=0**

Con este ya estamos en condiciones de reiniciar nuestro servidor, probar que los servicios levanten e instalar drbd.

* 1. **Instalación de DRBD**

Descargamos el paquete de alguno de los repositorios de CentOS, <http://pkgs.org/centos-4-rhel-4/centos-rhel-extras-i386/drbd83-8.3.8-1.el4_8.i386.rpm.html>

Copiamos el paquete a nuestro servidor e instalamos de la siguiente manera:

**rpm –Uvh drbd83-8.3.8-1.el4\_8.i386.rpm**

* + 1. **Configuración**

Básicamente DRBD consta de 3 archivos de configuración, ellos son:

**/etc/ drbd.conf** donde se define el archivo de parámetros globales y de donde y se carga nuestro archivo de configuración del recurso.

**include "drbd.d/global\_common.conf";**

**include "drbd.d/\*.res";**

Dentro de **/etc/drbd.d** tenemos **global\_common.conf** el archivo de configuración global y el archivo de la configuración de nuestro recurso **recurso.res**.

Dentro de **global\_common.conf**  podemos destacar las siguientes opciones:

**usage-count no;**

Cuando instalamos el producto no pide si deseamos participar de una estadística de cantidad de servidores con DRBD instalados, a veces aunque decimos que no, no se actualiza correctamente la variable arriba mencionada y cada vez que iniciamos el servicio nos da las gracias por participar y trata de conectarse el servidor de DRBD, para deshabilitar esa característica configuramos usage-count en no.

**protocol C;**

Es uno de los parámetros más importantes de la configuración ya que va a indicar como va a ser la sincronización entre nodos, C indica que es sincrónica.

**wfc-timeout 10;**

Por defecto cuando iniciamos el servicio si su par no está levantado va a espera indefinidamente, como no queremos eso, porque por ejemplo el otro nodo está caído, le decimos que espere 10 segundos, si no obtiene respuesta, continúa después de los 10 segundos.

**recurso.res**

Es nuestro archive de configuración, en el definimos que nodos conforman nuestra réplica:

# Nuestro recurso se llama drbdtest

resource drbdtest {

# A que velocidad va a syncronizar, depende de la tarjeta de red

syncer {

rate 100M;

}

on suchugo {

**# Dispositivo que vamos a usar para montar luego el recurso**

device /dev/drbd1;

**# El volume lógico que vamos a replicar**

disk /dev/rootvg/lvu02;

address 128.2.105.199:7789;

# Volumen lógico para los metadatos

**meta-disk /dev/rootvg/drbdmetadata[0];**

}

on db2 {

device /dev/drbd1;

disk /dev/rootvg/lvu02;

address 128.2.105.99:7789;

meta-disk /dev/rootvg/drbdmetadata[0];

}

}

**meta-disk**

Meta disk en donde DRBD va a escribir información que el necesita, cuando creamos un volumen nuevo sin datos, no necesitamos utilizar un volumen externo como en este caso **drbdmetadata**, porque DRBD va a escribir los metadatos al final del volumen.

Si bien podríamos haber extendido el mismo, es peligroso porque se pueden perder datos si no se hace correctamente, así que en esta ocasión como ya tenemos datos en los volúmenes, decidimos crear otro volumen lógico de 512 megas para tal fin.

**ACTIVE '/dev/rootvg/drbdmetadata' [512.00 MB] inherit**

* + 1. **Iniciando el servicio por primera vez**

Tenemos casi terminado nuestro sistema, falta crear los dispositivos e iniciar el servicio, antes que nada debemos crear nuestro recurso drbdtest como se indica a continuación:

[root@db2 drbd.d]# **drbdadm create-md drbdtest**

md\_offset 11383320576

al\_offset 11383287808

bm\_offset 11382939648

Found LVM2 physical volume signature

11112448 kB data area apparently used

11116152 kB left usable by current configuration

Even though it looks like this would place the new meta data into

unused space, you still need to confirm, as this is only a guess.

Do you want to proceed?

[need to type 'yes' to confirm]

Writing meta data...

initializing activity log

NOT initialized bitmap

New drbd meta data block successfully created.

success

Una vez creado nuestro recurso, debemos hacer lo siguiente en el nodo que va a oficiar de primario:

**drbdadm -- --overwrite-data-of-peer primary all**

Ahora sí estamos en condiciones de iniciar el servicio.

**service drbd start**

Asi se muestra cuando está realizando una sincronización, en esta ocación, por primera vez con el nodo secundario:

[root@db2 drbd.d]# **service drbd status**

drbd driver loaded OK; device status:

version: 8.3.8.1 (api:88/proto:86-94)

srcversion: 70A36576ED3A03229A2AC68

m:res cs ro ds p mounted fstype

... sync'ed: 24.3% (104032/131072)K

1:drbdtest SyncSource Primary/Secondary UpToDate/Inconsistent C

Como podemos ver a continuación, el recurso está sincronizado **( UpToDate )** y este nodo en este momento está como primario:

[root@db2 ~]# **service drbd status**

drbd driver loaded OK; device status:

version: 8.3.8.1 (api:88/proto:86-94)

srcversion: 70A36576ED3A03229A2AC68

m:res cs ro ds p mounted fstype

1:drbdtest Connected Primary/Secondary UpToDate/UpToDate C

**4.0 Failover Manual**

Es importante saber que solo un nodo va a tener acceso a los datos por lo cual solo uno puede oficiar de Primary, para poder acceder a los datos desde el nodo secundario debemos hacer lo siguiente:

4.0.1 En el nodo que es Primary

Bajar la base de datos

**drbdadm secondary drbdtest**

**umount /u02/oradata/SF300**

4.0.2 En el nodo que es Secondary

**drbdadm primary drbdtest**

**mount /dev/drbd1 /u02/oradata/SF300**

Subir la base de datos

**4.1 Incrementar un sistema de archivos.**

Tenemos la posibilidad de incrementar un sistema de archivos en caliente, el procedimiento varía de lo que lo hacemos habitualmente, en condiciones normales, solo incrementamos el Volumen Lógico en un solo nodo, con DRBD tenemos que ejecutar los pasos en ambos nodos mas otros pasos adicionales en el nodo primario para que recién ahí comience la réplica.

**4.1.2 Pasos**

Tomemos como ejemplo nuestros nodos de prueba e incrementemos el sistema de archivos tecnol, con drbd-overview vemos el estado de las conexiones y el tamaño actual de los sistemas de archivos, tecnol en este momento tiene su valor original de 124MB. El procedimiento consiste en dos pasos, extender el volumen a nivel de bloque y despúes a nivel de sistema de archivos como veremos a continuación. Esto es, extender el volumen lógico **/dev/rootvg/lvtecnol** y luego el sistema de archivos **/dev/drbd3.**

**4.1.2.3 Ejecución**

Primeramente vemos el estado actual de nuestro sistema de archivos, como podemos ver inicialmente dispone de 124MB de espacio asignado.

[root@db2 ~]# **drbd-overview**

1:database Connected Secondary/Secondary UpToDate/UpToDate C r----

2:archive Connected Secondary/Secondary UpToDate/UpToDate C r----

**3:tecnol Connected Primary/Secondary UpToDate/UpToDate C r---- /tecnol ext3 124M 91M 28M 78%**

4:sfctrl Connected Secondary/Secondary UpToDate/UpToDate C r----

5:sts Connected Secondary/Secondary UpToDate/UpToDate C r----

Ahora extendemos el sistema de archivos a nivel de bloque, este es el único paso que debemos realizar en ambos nodos:

[root@db2 ~]# **lvextend -L +500M /dev/rootvg/lvtecnol**

/dev/drbd1: open failed: Read-only file system

/dev/drbd2: open failed: Read-only file system

/dev/drbd4: open failed: Read-only file system

/dev/drbd5: open failed: Read-only file system

Rounding up size to full physical extent 512.00 MB

Extending logical volume lvtecnol to 640.00 MB

Logical volume lvtecnol successfully resized

suc999:/root> **lvextend -L +500M /dev/rootvg/lvtecnol**

/dev/drbd3: open failed: Read-only file system

Rounding up size to full physical extent 512.00 MB

Extending logical volume lvtecnol to 640.00 MB

Logical volume lvtecnol successfully resized

El siguiente paso debemos ejecutarlo solamente en el nodo que oficia de primario, en nuestro caso db2, esto va a indicarle a DRBD que ejecute su rutina de réplica como vemos en los logs.

[root@db2 ~]# **drbdadm resize tecnol**

[root@db2 ~]# **tail -f /var/log/messages**

May 24 07:41:55 db2 kernel: [76128.471973] EXT3-fs (drbd3): using internal journal

May 24 07:41:55 db2 kernel: [76128.471977] EXT3-fs (drbd3): mounted filesystem with writeback data mode

**May 24 07:47:07 db2 kernel: [76439.818663] block drbd3: drbd\_bm\_resize called with capacity == 1310720**

May 24 07:47:07 db2 kernel: [76439.818673] block drbd3: resync bitmap: bits=163840 words=5120

May 24 07:47:07 db2 kernel: [76439.818677] **block drbd3: size = 640 MB (655360 KB)**

May 24 07:47:07 db2 kernel: [76439.818682] block drbd3: Writing the whole bitmap, size changed

May 24 07:47:07 db2 kernel: [76439.841926] block drbd3: 512 MB (131072 bits) marked out-of-sync by on disk bit-map.

May 24 07:47:07 db2 kernel: [76439.874803] block drbd3: Resync of new storage after online grow

May 24 07:47:07 db2 kernel: [76439.874826] block drbd3: conn( Connected -> SyncSource ) pdsk( UpToDate -> Inconsistent )

May 24 07:47:07 db2 kernel: [76439.874832] block drbd3: **Began resync as SyncSource (will sync 524288 KB [131072 bits set]).**

En suc999 vemos que comienza la sincronización, **SyncTarget** nos indica que estamos recibiendo la actualización de nuestro par db2:

suc999:/root>**watch drbd-overview**

**Every 2.0s: drbd-overview Thu May 24 11:56:40 2012**

1:database Connected Secondary/Secondary UpToDate/UpToDate C r----

2:archive Connected Secondary/Secondary UpToDate/UpToDate C r----

3:tecnol **SyncTarget** Secondary/Primary Inconsistent/UpToDate C r----

**[=========>..........] sync'ed: 52.0% (254304/524288)K**

4:sfctrl Connected Secondary/Secondary UpToDate/UpToDate C r----

5:sts Connected Secondary/Secondary UpToDate/UpToDate C r----

Una vez sincronizado vemos que el estado pasa de Inconsistent a UpToDate, lo cual indica que la actualización finalizó correctamente.

**May 24 07:47:53 db2 kernel: [76485.848428] block drbd3: Resync done (total 45 sec; paused 0 sec; 11648 K/sec)**

**May 24 07:47:53 db2 kernel: [76485.848437] block drbd3: conn( SyncSource -> Connected ) pdsk( Inconsistent -> UpToDate )**

Como podemos ver el tamaño no ha cambiado, esto es porque nos falta un paso final, que es extenderlo a nivel de sistema de archivos, anteriormente lo hicimos a nivel de bloques.

[root@db2 ~]# **drbd-overview**

1:database Connected Secondary/Secondary UpToDate/UpToDate C r----

2:archive Connected Secondary/Secondary UpToDate/UpToDate C r----

**3:tecnol Connected Primary/Secondary UpToDate/UpToDate C r---- /tecnol ext3 124M 91M 28M 78%**

4:sfctrl Connected Secondary/Secondary UpToDate/UpToDate C r----

5:sts Connected Secondary/Secondary UpToDate/UpToDate C r----

[root@db2 ~]# **resize2fs /dev/drbd3**

resize2fs 1.39 (29-May-2006)

Filesystem at /dev/drbd3 is mounted on /tecnol; on-line resizing required

Performing an on-line resize of /dev/drbd3 to 655360 (1k) blocks.

The filesystem on /dev/drbd3 is now 655360 blocks long.

[root@db2 ~]# **drbd-overview**

1:database Connected Secondary/Secondary UpToDate/UpToDate C r----

2:archive Connected Secondary/Secondary UpToDate/UpToDate C r----

**3:tecnol Connected Primary/Secondary UpToDate/UpToDate C r---- /tecnol ext3 620M 92M 497M 16%**

4:sfctrl Connected Secondary/Secondary UpToDate/UpToDate C r----

5:sts Connected Secondary/Secondary UpToDate/UpToDate C r----

Ahora si podemos comprobar que nuestro sistema de archivos tecnol se encuentra montado y reflejando el nuevo espacio asignado.

**Rev 1:**

Hay que tener en cuenta que cuando se monta por primera vez en el secundario, va a tomar el user ID efectivo, es decir, si en el nodo que es primario el usuario oracle tiene uid 505, cuando monte va a tratar de mapear los mismos datos, si 505 corresponde a otro usuario, los permisos van a quedar para ese usuario, asçi que es importante que el usuario y grupo estén iguales en ambos servidores.

**Rev 2:**

Se actualiza la configuración para que replique dos recursos, tenemos que dividir la configuración en dos archivos separados ya que la versión de DRBD que tenemos instalada no soporta la configuración de de volúmenes múltiples en un mismo archivo de configuración. Cabe agregar que tampoco soporta los dos recursos sobre un mismo link, lo cual implica la utilización de otra tarjeta de red.

Los volúmenes lógicos para los metadatos:

**ACTIVE '/dev/rootvg/drbdmetadata1' [512.00 MB] inherit**

**ACTIVE '/dev/rootvg/drbdmetadata2' [512.00 MB] inherit**

**Recurso 1:**

[root@db2 drbd.d]# cat res1.res

resource database {

syncer {

rate 100M;

}

on suchugo {

device /dev/drbd1;

disk /dev/rootvg/lvu02;

address 128.2.105.199:7789;

meta-disk /dev/rootvg/drbdmetadata1[0];

}

on db2 {

device /dev/drbd1;

disk /dev/rootvg/lvu02;

address 128.2.105.99:7789;

meta-disk /dev/rootvg/drbdmetadata1[0];

}

}

}

**Recurso 2:**

[root@db2 drbd.d]# cat res2.res

resource archive {

syncer {

rate 100M;

}

on suchugo {

device /dev/drbd2;

disk dev/rootvg/lvarch;

address 3.1.1.2:7789;

meta-disk /dev/rootvg/drbdmetadata2[0];

}

on db2 {

device /dev/drbd2;

disk dev/rootvg/lvarch;

address 3.1.1.1:7789;

meta-disk /dev/rootvg/drbdmetadata2[0];

}

}

**Rev 3:**

Para el failover manual actualizamos también el procedimiento:

Nodo1:

Bajar la base de datos

**drbdadm secondary database**

**drbdadm secondary archive**

**umount /u02/oradata/SF300**

**umount /dbfs\_arc\_SF300**

Nodo 2:

**drbdadm primary database**

**drbdadm primary archive**

**mount /u02/oradata/SF300**

**mount /dbfs\_arc\_SF300**

Subir la base de datos

En suchugo como Secundario, SyncTarget indica que somos el nodo que recibe la actualización del Primario, en este caso db2.

version: 8.3.8.1 (api:88/proto:86-94)

srcversion: 70A36576ED3A03229A2AC68

1: cs:**SyncTarget** ro:Secondary/Primary ds:Inconsistent/UpToDate C r----

ns:0 nr:1519328 dw:1519296 dr:0 al:0 bm:92 lo:2 pe:14344 ua:1 ap:0 ep:1 wo:b oos:17355072

[>...................] sync'ed: 8.1% (16948/18432)M

finish: 0:17:51 speed: 16,000 (11,508) K/sec

2: cs:Connected ro:Secondary/Primary ds:UpToDate/UpToDate C r----

ns:0 nr:1048576 dw:1048576 dr:0 al:0 bm:64 lo:0 pe:0 ua:0 ap:0 ep:1 wo:b oos:0

En db2, SyncSource nos indica que somos la fuente de actualización.

[root@db2 ~]# service drbd status

drbd driver loaded OK; device status:

version: 8.3.8.1 (api:88/proto:86-94)

srcversion: 70A36576ED3A03229A2AC68

m:res cs ro ds p mounted fstype

... sync'ed: 16.3% (15432/18432)M

1:database **SyncSource** Primary/Secondary UpToDate/Inconsistent C

2:archive Connected Primary/Secondary UpToDate/UpToDate C

**Rev 4:**

Para la sucursal 61 en la cual se utilizó al servicio se le modifico de la siguiente manera en el host que hace de master:

suc61:/etc/drbd.d>cat global\_common.conf |grep become

**become-primary-on suc61;**

Esto hace que cada vez que levanten los recursos se configuren como primaries y no haya que intervenir manualmente. Tambien se modificó el archivo fstab comentando las líneas que corresponden a tecnol y sfctrl que son los recursos que se van a compartir y los mismos se levantan desde el archivo rc.local agregando la siguiente línea:

**echo "Montando Recursos DRBD"**

**mount /dev/drbd0 /tecnol**

**mount /dev/drbd1 /sfctrl**

**echo "Hecho"**

La configuración de los servicios quedó de la siguiente manera:

**sfctrl.res**

**tecnol.res**

suc61:/etc/drbd.d>cat sfctrl.res

resource drbd-sfctrl {

device minor 2;

# A que velocidad va a syncronizar, depende de la tarjeta de red

syncer {

rate 100M;

}

on suc61 {

# Dispositivo que vamos a usar para montar luego el recurso

device /dev/drbd1;

# El volume logico que vamos a replicar

disk /dev/vgcoto/sfctrl;

address 3.1.1.12:7789;

# Volumen logico para los metadatos

meta-disk /dev/vgcoto/drbdmetadata2[0];

}

on suc61b {

device /dev/drbd1;

disk /dev/vgcoto/sfctrl;

address 3.1.1.13:7789;

meta-disk /dev/vgcoto/drbdmetadata2[0];

}

}

suc61:/etc/drbd.d>cat tecnol.res

resource drbd-tecnol {

device minor 1;

# A que velocidad va a syncronizar, depende de la tarjeta de red

syncer {

rate 100M;

}

on suc61 {

# Dispositivo que vamos a usar para montar luego el recurso

device /dev/drbd0;

# El volume logico que vamos a replicar

disk /dev/vgcoto/tecnol;

address 3.1.1.10:7789;

# Volumen logico para los metadatos

meta-disk /dev/vgcoto/drbdmetadata1[0];

}

on suc61b {

device /dev/drbd0;

disk /dev/vgcoto/tecnol;

address 3.1.1.11:7789;

meta-disk /dev/vgcoto/drbdmetadata1[0];

}

}

Las IP quedaron de la siguiente manera, recordar que estas IP las utiliza DRBD para comunicarse con el host esclavo y en esta versión de DRBD no se soportan varios recursos en un solo link, por eso se soluciona con un alias por cada recurso extra que agregamos:

eth1 **Link encap:Ethernet HWaddr 5C:F3:FC:F1:F2:30**

**inet addr:3.1.1.1 Bcast:3.1.1.255 Mask:255.255.255.0**

eth1:0 Link encap:Ethernet HWaddr 5C:F3:FC:F1:F2:30

**inet addr:3.1.1.10 Bcast:3.1.1.255 Mask:255.255.255.0**

eth1:1 Link encap:Ethernet HWaddr 5C:F3:FC:F1:F2:30

i**net addr:3.1.1.12 Bcast:3.1.1.255 Mask:255.255.255.0**

Tambien se agregó lo siguiente en /etc/hosts en ambos servidores:

3.1.1.1 nodo1

3.1.1.2 nodo2

Con el servicio heartbeat manejamos el failover de la IP, el servicio está siempre bajo ya que queremos que el failover sea manual el cual levantamos siempre en el host secundario y se encuentra en:

**suc61:/etc/init.d>./heartbeat**

**NOTA1**

Para la versión 8.4 de DRBD el procedimiento de inicialización inicial cambia, por favor ver la documentación oficial:

**http://www.drbd.org/users-guide-8.4/s-first-time-up.html**

Los pasos serian en ambos nodos:

drbdadm create-md resource

drbdadm up resource

Luego en un solo nodo, el que hace de master:

drbdadm primary --force resource