Servicios de RAID en red.

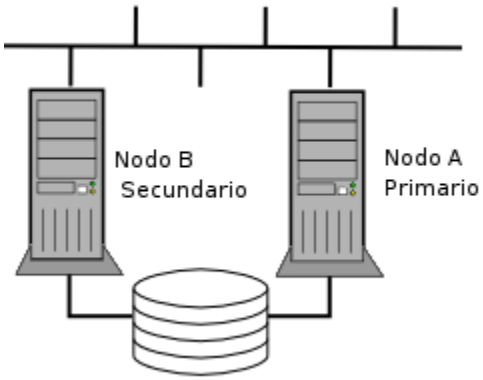
Autor: Enrique V. Bonet Esteban

Introducción.

Cuando disponemos de un solo equipo informático y pretendemos garantizar un cierto grado de redundancia en la información almacenada, podemos montar un sistema RAID (Redundant Array of Independent Disks) mediante dos o más discos, donde la información esta replicada en más de un disco utilizando, de forma más común, RAID-1, RAID-5 o RAID-6. De este modo, un fallo en un disco no supone la pérdida de la información puesto que esta se encuentra en más de un disco. Un punto a tener en cuenta es la sincronización de los datos, en un RAID de discos la sincronización es casi inmediata, las escrituras se pueden realizar en el mismo tiempo, por lo que casi se garantiza que no se producirían perdidas de información si falla un disco del RAID.

Cuando disponemos de más de un equipo y queremos almacenar de forma redundante los datos almacenados, al igual que se pretendía con un RAID, tenemos distintas opciones posibles, pero una solución barata y sencilla de implantar es montar un RAID por red, es decir, disponer de varias copias de la información en distintos nodos de una red garantizando una sincronización lo suficientemente rápida.

Distributed Replicated Block Device (DRBD) es un sistema que permite implementar sistemas de almacenamiento de información redundantes, esto es, existen distintas copias de la misma información en distintas sistemas, lo que posibilita diseñar sistemas de información de alta disponibilidad, entendiendo la disponibilidad como la capacidad de ofrecer un servicio ante posibles fallos.



DRBD es un dispositivo de bloques que permite realizar RAID-1 entre dos dispositivos de bloques situados en dos nodos (ordenadores) diferentes[1], utilizando para ello un enlace de red que suele ser dedicado, permitiendo la salvaguarda de la información y además posibilitando, al utilizar un nodo distinto, el uso de dichos datos por el otro nodo en determinadas situaciones, situaciones que pueden ser un fallo en el nodo principal (sistemas de alta disponibilidad), etc.

[1] Es posible sincronizar dispositivos de bloques entre más de dos nodos, aunque aquí no expondremos la forma de realizarlo por ser una simple extensión de la sincronización entre dos nodos.

El funcionamiento de DRBD se basa en que los dispositivos que forman el RAID-1 poseen un estado (primario o secundario), estando un dispositivo de un nodo en estado primario y el otro en estado secundario[2]. El nodo cuyo dispositivo se encuentra en estado primario puede acceder a los datos para lectura y/o escritura, mientras que el otro nodo no posee acceso a los mismos y se limita a escribir los datos que recibe del otro nodo a su disco.

Cuando el nodo con el dispositivo en estado primario va a realizar una escritura en su dispositivo de bloques, escribe los datos de forma local y envía dichos datos al dispositivo en estado secundario del otro nodo. Este recibe los datos y actualiza la información de su dispositivo de bloques, quedando de esta forma ambos dispositivos sincronizados y por tanto en RAID-1.

Configuración de DRBD.

La configuración de DRBD se realiza mediante el fichero */etc/drbd.conf*, el cual debe ser idéntico en los dos nodos que forman el RAID-1. En la actualidad el citado fichero solo contiene dos líneas:

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

Por lo que toda la información se encuentra dentro de los ficheros que están dentro del directorio */etc/drbd.d*.

La sintaxis de los ficheros es igual (sean extensión conf o extensión res), y consiste en líneas que forman secciones y contienen parámetros, considerándose que el carácter # indica el comienzo de un comentario y por tanto, que toda la línea que sigue a dicho carácter es ignorada.

Una sección comienza con una palabra clave de inicio de sección, seguida en ocasiones por un nombre adicional y el carácter {, conteniendo en su interior otras secciones y/o los parámetros que configuran esta sección y terminando la misma con el carácter }. Por tanto, su sintaxis puede expresarse como:

```
sección [nombre] { <parámetro> [valor]; [...] }
```

Los parámetros comienzan con el nombre del parámetro seguido por espacios en blanco y a continuación los valores que toma el parámetro dentro de la sección, terminando el parámetro con un punto y como (;). Un caso especial de parámetros son los booleanos, los cuales no contienen un valor, pero deben igualmente terminar con punto y coma. Algunos parámetros tienen unidades por defecto que pueden ser modificadas mediante las terminaciones K (1024), M (1024 K) y G (1024 M).

Veamos más detenidamente las secciones y parámetros existentes.

[2] En la actualidad DRBD posee la posibilidad de que ambos nodos funcionen en modo primario, pero nosotros nos centraremos en su uso como dispositivo de RAID y por tanto con un nodo primario y otro secundario.

Secciones.

Las secciones de DRBD permiten definir valores globales y recursos compartidos y datos de los mismos, utilizando para su configuración los parámetros explicados con anterioridad. Las secciones existentes en DRBD son:

skip. Define una sección que es un comentario que ocupe más de una línea, pues todo el texto comprendido dentro de una sección *skip* es ignorado. Además, el texto comprendido entre la palabra *skip* y el carácter { es también ignorado, por lo que puede utilizarse para comentar cualquier sección existente anteponiendo tan solo esta palabra. Por ejemplo:

```
skip resource drbd0 {
...
}
```

Comenta toda la sección *resource drbd0* sin necesidad de modificar para nada su contenido.

global. Configura parámetros globales, pudiendo existir una sola sección global en el fichero de configuración, preferiblemente la primera sección. Solo los parámetros *minor-count*, *dialog-refresh*, *disable-ip-verification* y *usage-count* pueden ponerse en esta sección.

common. Define valores comunes para todos los recursos utilizados. Estos valores pueden ser modificados posteriormente por cada recurso de DRBD de forma particular. La sección *common* puede contener en su interior otras secciones como *startup*, *syncer*, *handlers*, *net* y *disk*.

resource <nombre>. Configura un recurso de DRBD, siendo obligatoria la presencia en su interior de dos secciones *on <nombre ordenador>*, una para cada uno de los nodos que forman el recurso DRBD. Además, puede tener en su interior, de forma opcional, otras secciones como *startup*, *syncer*, *handlers*, *net* y/o *disk*.

on <nombre ordenador>. Contiene los parámetros necesarios para configurar uno de los nodos del dispositivo DRBD del recurso que contiene esta sección. El *<nombre ordenador>* es obligatorio y debe coincidir con el nombre del ordenador en Linux obtenido ejecutando el comando `uname -n`. Esta sección requiere la especificación obligatoria de los parámetros *device*, *disk*, *address* y *meta-disk*.

disk. Permite especificar el comportamiento del sistema respecto al dispositivo de bloques del disco. Algunos de sus parámetros son *on-io-error* y *resync-rate*.

net. Permite definir opciones de configuración de la red para la transmisión de los bloques entre los nodos. Sus parámetros opcionales son *sndbuf-size*, *rcvbuf-size*, *timeout*, *connect-int*, *ping-int*, *ping-timeout*, *max-buffers*, *max-epoch-size*, *ko-count*.

startup. Permite definir el funcionamiento del sistema en el arranque. Sus parámetros opcionales son *wfc-timeout* y *degr-wfc-timeout* y *outdated-wfc-timeout*.

handlers: Permite definir ejecutables que deben ser arrancados por DRBD en respuesta a cierto eventos.

Parámetros.

DRBD posee un gran número de parámetros, muchos de los cuales toman valores por defecto adecuados para la mayoría de aplicaciones, por lo que tan solo algunos de ellos deben ser configurados en la mayoría de los casos. Resaltar además que no todos los parámetros pueden ser especificados en todas las secciones, como hemos visto con anterioridad. Los parámetros de DRBD son:

minor-count <valor>. De forma general, DRBD carga el número exacto de dispositivos que se indican en el fichero de configuración. Sin embargo, en ciertas ocasiones puede ser necesario definir más dispositivos de DRBD una vez el servicio ha arrancado. Esta posibilidad esta contemplada de forma que por defecto DRBD permite definir 2 nuevos dispositivos si DRBD se ejecuta como un módulo del kernel ó 8 si DRBD se encuentra empotrado en el kernel, esto es, compilado en su interior. En el caso de que DRBD se ejecute como módulo del kernel *minor-count* permite modificar el número de nuevos dispositivos que pueden definirse, mientras que en el segundo caso su valor es ignorado, debiendo pasarse el parámetro *drbd.minor_count=<valor>* en el arranque del kernel para modificar el valor por defecto de 8.

dialog-refresh <tiempo (segundos)>. Indica cada cuanto tiempo se refresca el dialogo con el usuario, o no se refresca si *<tiempo>* es 0. El valor por defecto es 1.

disable-ip-verification. Indica que DRBD no verifique la dirección de red del sistema.

usage-count <valor>. Permite enviar un valor a los desarrolladores de DRBD de forma que estos tengan datos sobre el número de equipos que utilizan DRBD. Los valores posibles son *yes*, *no* y *ask*.

protocol <identificador>. En un enlace TCP/IP especifica el protocolo a utilizar mediante el <identificador>. Los protocolos validos son A, B y C, siendo A más rápido en efectuar sus operaciones, pero el menos seguro, mientras que C es el más lento pero el más seguro. En concreto, en el protocolo A toda operación de escritura en el disco se considera efectuada si los datos han sido escritos en el disco local y enviados al buffer de envío TCP/IP del ordenador local, mientras que en el protocolo B toda operación de escritura en el disco se considera efectuada si los datos han sido escritos en el disco local y recibidos en el buffer de recepción TCP/IP del ordenador remoto, mientras que el protocolo C considera efectuada la operación de escritura si todos los datos han sido escritos tanto en el disco local como en el remoto.

device <nombre> minor <número>. Es el nombre del dispositivo de bloques de DRBD. Este es el nombre del dispositivo que debe utilizarse para efectuar operaciones sobre el dispositivo y no el nombre del dispositivo de bloques definido con los parámetros del disco. Es posible omitir *<nombre>*, en cuyo caso se utilizará el dispositivo */dev/drbd<número>* o bien es posible omitir el *minor <número>* en cuyo caso se utilizará el dispositivo indicado por *<nombre>*.

disk <nombre>. Es el nombre del dispositivo de bloques definido con los parámetros del disco que se corresponde con el dispositivo de bloques del nodo.

address <dirección IP:puerto>. Indica la dirección IP y el puerto TCP que es utilizado por uno nodo para esperar conexiones desde el otro nodo, de forma que dos dispositivos DRBD no pueden tener idéntica combinación de IP y puerto.

meta-disk {internal, dispositivo[indice]}. Indica donde se almacenarán los meta-datos[3] necesarios para que DRBD pueda manejar el RAID-1, siendo necesarios unos 128 MB[4] para cada dispositivo de DRBD. El valor *internal* indica que los datos serán almacenados al final del propio dispositivo, por lo que este debe tener un sistema de ficheros que ocupe 128 MB menos que el tamaño físico del dispositivo de bloques del disco. Por su parte, el valor *dispositivo[indice]* indica que los datos del dispositivo se almacenan en el dispositivo indicado por dispositivo en la posición indicada por indice. Así, si se especifica *meta-disk /dev/hda4[0]* y *meta-disk /dev/hda4[1]* los datos de un dispositivo se almacenan en los primeros 128 MB de */dev/hda4*, mientras que los de otro en los siguientes 128 MB, por lo que */dev/hda4* debe tener un tamaño mínimo de 256 MB.

on-io-error {pass_on, call-local-io-error, detach}. Indica la acción a realizar en caso de que el dispositivo de bloques del disco indique un error de lectura o escritura. El valor *pass_on* indica que se informe del error al dispositivo de bloques del nodo, mientras que *call-local-io-error* indica que ejecute el script indicado por *local-io-error* y *detach* indica que el nodo continúe funcionando con el dispositivo de bloques del disco.

sndbuf-size <tamaño>. Especifica el tamaño del buffer de envío del socket TCP. El tamaño por defecto es de 128KBytes, mientras que el valor 0 permite al kernel seleccionar el tamaño de forma automática.

rcvbuf-size <tamaño>. Especifica el tamaño del buffer de recepción del socket TCP. El tamaño por defecto es de 128KBytes, mientras que el valor 0 permite al kernel seleccionar el tamaño de forma automática.

timeout <tiempo (en décimas de segundo)>. Indica el tiempo que se espera la respuesta a un paquete antes de que el otro nodo sea considerado no activo y la conexión TCP/IP abandonada. El valor por defecto es de 60 (6 segundos). El valor de *timeout* debe ser menor que el de *connect-int* y el de *ping-int*.

connect-int <tiempo (en segundos)>. Especifica el tiempo que transcurren entre dos intentos de conexión si la conexión no ha sido posible. El valor por defecto es de 10 segundos.

[3] Los meta-datos contienen información relativa a los bloques que se van a sincronizar, contadores y eventos de entrada/salida, dependiendo del protocolo empleado y la información almacenada en este área DRBD determina si debe o no sincronizar

[4] Una aproximación bastante exacta del tamaño que realmente ocupan los metadatos puede obtenerse con la formula $M_{MB} = 1 + C_{MB}/32768$, pudiendo observarse que 128 Mbytes son suficientes para particiones inferiores a 4 Tbytes.

ping-int <tiempo (en segundos)>. Especifica que si en el tiempo indicado por <tiempo> no se ha recibido ningún paquete del otro nodo, se envíe un paquete para comprobar que el otro nodo sigue activo. El valor por defecto es de 10 segundos.

ping-timeout <tiempo (en milésimas de segundo)>. Especifica el tiempo de espera a la respuesta de un paquete generado por *ping-int*. Si la respuesta no se recibe en este tiempo se considerará al otro nodo muerto. El tiempo por defecto es de 500 milisegundos.

max-buffers <número>. Indica el número de buffers que serán reservados por DRBD. El valor por defecto es de 32 buffers de 4 KB.

ko-count <valor>. Indica que en caso de que un nodo secundario falle <valor> veces en realizar una escritura, se excluya del cluster. El valor por defecto 0 deshabilita esta opción.

max-epoch-size <número>. Indica el máximo número de bloques de datos entre dos escrituras.

wfc-timeout <tiempo (segundos)>. Indica el tiempo de espera en el arranque antes de dar un timeout. El arranque de DRBD bloquea el arranque de todo el nodo hasta que los recursos de DRBD se han conectado, por lo que si no se desea una espera infinita (valor 0, que es el valor por defecto), debe especificarse un valor máximo de espera.

degr-wfc-timeout <tiempo (segundos)>. Indica el tiempo de espera en el arranque cuando se rearranca un sistema que estaba degradado con anterioridad (con un solo nodo) en lugar del valor de wfc-timeout.

outdated-wfc-timeout <tiempo (segundos)>. Indica el tiempo de espera si el otro nodo esta desactualizado.

rate <valor>. Es un alias del parámetro *resync-rate <valor>*.

resync-rate <valor>. Especifica el máximo ancho de banda que utilizará DRBD en la transmisión de datos. El valor por defecto es de 250 KBytes/s., pudiendo especificarse los valores con los sufijos K, M y G.

local-io-error <comando>. Comando a ejecutar si se produce un error en la entrada/salida del subsistema local.

Ejemplo de configuración de DRBD.

Vamos a ver un ejemplo completo de configuración de un DRBD formado por dos nodos de una red con un disco en cada nodo. Pretendemos crear un dispositivo drbd0 en ambos nodos que contenga una partición de cada uno de los nodos, siendo la partición */dev/sda2* con un tamaño de 2000 MB.

El primer dato a tener en cuenta es donde se va a almacenar los meta-datos (parámetro *meta-disk*). En nuestro caso optaremos por una configuración interna al dispositivo (opción *internal*), por requerir un mayor esfuerzo de configuración del sistema.

En efecto, debido a que la opción *internal* almacena al final del dispositivo de bloques de la partición los 128 MB que utiliza para los meta-datos, pero no comprueba si dicha sección del disco se encuentra ocupada por un sistema de ficheros, es posible estar escribiendo datos en el área utilizada para los meta-datos, produciéndose un error. Para evitar este problema lo primero que debemos hacer es redimensionar el tamaño del sistema de ficheros de la partición, reduciéndolo de forma que el nuevo tamaño del sistema de ficheros deje libre los últimos 128 MB. Para ello, ejecutamos el comando[5]:

```
resize2fs /dev/sda2 1872M
```

Una vez redimensionado el sistema de ficheros en ambos nodos configuramos el fichero de configuración de DRBD (*/etc/drbd.d/global_common.conf* y */etc/drbd.d/midrbd.conf*). En concreto, el fichero global.common.conf deberá contener los valores:

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

Mientras que el fichero *midrbd.conf* contendría los siguientes valores:

```
resource drbd0 {
      net {
            protocol C;
      }

      syncer {
            resync-rate 10M;
      }

      startup {
            wfc-timeout 300;
            degr-wfc-timeout 150;
            outdated-wfc-timeout 150;
      }

      on nodo1 {
            device /dev/drbd0;
            disk /dev/sda2;
            address 192.168.0.1:7780;
            meta-disk internal;
      }

      on nodo2 {
            device /dev/drbd0;
            disk /dev/sda2;
```

[5] Tengase en cuenta que en nuestro ejemplo suponemos un sistema de ficheros de 2000 MB, de hay el valor de 1872M que ponemos en el comando resize2fs.

```
            address 192.168.0.2:7780;
            meta-disk internal;
      }
}
```

Como podemos ver, hemos establecido que el protocolo de sincronización que se utiliza sea el C y que el ancho de banda máximo se encuentre limitado a 10 MBytes, que es casi el límite físico del enlace entre dos nodos conectados por un cable cruzado a 100 Mbits. Además, los parámetros *wfc-timeout*, *degr-wfc-timeout* y *outdated-wfc-timeout* se han establecido a 300, 150 y 150 segundos, respectivamente.

Por otra parte, indicamos que el DRBD se compone de dos nodos llamados *nodo1* y *nodo2* siendo el dispositivo de bloques */dev/drbd0* y las particiones físicas a montar */dev/sda2* en ambos nodos. Ambos nodos se comunican con sus respectivas direcciones IP utilizando el puerto 7780 y, tal y como se comentó previamente, el área de meta-datos se encuentra dentro de cada una de las particiones.

Para finalizar la configuración debemos inicializar el área de metadatos en cada uno de los nodos. Para ello, ejecutamos el comando:

```
drbdadm create-md drbd0
```

Donde *drbd0* es el nombre del recurso de DRBD que hemos definido en el fichero de configuración.

Arranque y comprobación del estado de DRBD.

Una vez se ha configurado el servicio de DRBD, éste se arranca siempre con el comando:

```
service drbd start
```

Pudiendo comprobarse el estado de los nodos mediante la ejecución del comando:

```
service drbd status
```

En el primer arranque del servicio de DRBD en ambos nodos, la salida del comando anterior es:

```
m:res    cs         st                 ds                         p  mounted  fstype
0:drbd0  Connected  Secondary/Secondary  Inconsistent/Inconsistent   C
```

Donde se puede observar como al arrancar el servicio ambos nodos se encuentran en estado secundario (Secondary) e inconsistente (Inconsistent), pues nunca se han sincronizado entre ellos y por tanto tienen los mismos datos. Conviene resaltar que si el sistema de seguridad *selinux* se encuentra activado y no esta configurado para manejar de forma correcta los dispositivos que utiliza DRBD, la salida del comando anterior es:

```
m:res    cs         st                  ds                 p  mounted  fstype
0:drbd0  Connected  Secondary/Secondary  Diskless/Diskless  C
```

Indicando que el sistema no ha podido asignar al DRBD los discos de forma automática, en cuyo caso se puede hacer manualmente ejecutando el comando:

```
drbdadm attach drbd0
```

Aunque es conveniente arreglar el problema para garantizar que el servicio de DRBD pueda arrancar automáticamente con el sistema.

Una vez arrancado el servicio de DRBD, se puede forzar la primera sincronización convirtiendo uno de los nodos en primario y forzando que su estado sea el correcto mediante la ejecución del comando:

```
drbdadm primary --force drbd0
```

De forma que convertimos en primario ese nodo e indicamos, con la opción *--force*, que se considerén sus datos como correctos. Si ahora ejecutamos el comando:

```
service drbd status
```

Obtenemos como salida:

```
m:res    cs         st               ds                      p   mounted   fstype
0:drbd0  Connected  Primary/Secondary UpToDate/Inconsistent   C
...      sync'ed    3.4%             (10132/10476)Mfinish: 0:01:58 87,808 (87,808) K/sec
```

Y una vez terminada la sincronización la salida sería, en el nodo que hemos puesto en estado primario:

```
m:res    cs         st                  ds                 p   mounted       fstype
0:drbd0  Connected  Primary/Secondary   UpToDate/UpToDate  C
```

Y en el nodo que se encuentra en estado secundario:

```
m:res    cs         st                  ds                 p   mounted       fstype
0:drbd0  Connected  Secondary/Primary   UpToDate/UpToDate  C
```

Si los nodos ya se encuentran sincronizados entre ellos, podemos poner uno de ellos en estado primario ejecutando el comando:

```
drbdadm primay drbd0
```

Sin necesidad de indicar que su contenido es el correcto.

Una vez un nodo esta en estado primario, podemos proceder a montarlo en un directorio del sistema de ficheros ejecutando sencillamente el comando[6]:

```
mount [-t <tipo de sistema>] /dev/drbd0 <punto de montaje>
```

Obteniendo como salida del estado la siguiente información en el nodo cuyo estado es primario y se encuentra montado:

[6] Obviamente el comando debe ejecutarse en el nodo que tiene el estado de primario.

```
m:res    cs         st                  ds                 p   mounted       fstype
0:drbd0  Connected  Primary/Secundary   UpToDate/UpToDate  C   /directorio   ext3
```

Y la siguiente en el nodo cuyo estado es secundario:

```
m:res    cs         st                  ds                 p   mounted       fstype
0:drbd0  Connected  Secondary/Primary   UpToDate/UpToDate  C
```

A partir de este momento, cualquier cambio que se produzca en este dispositivo de bloques montado en el nodo primario se replicara por red en el dispositivo de bloques del otro nodo.

Sincronización de los nodos de DRBD.

Además de en el instante inicial, que hemos visto como sincronizar ambos nodos forzando a elegir sus datos como los correctos, pueden surgir ocasiones en que los nodos de DRBD queden en un estado inconsistente al no estar sincronizados por una pérdida temporal de comunicación. Generalmente DRBD es capaz de resolver el problema en estos casos, pues en los metadatos almacena que sectores han sido modificados desde la pérdida de comunicación y por tanto, que sectores deben ser sincronizados.

Sin embargo, existe un caso especial en que este proceso automático no es posible, y es cuando ambos nodos quedan sin comunicación entre sí, ambos adquieren el papel de primarios y realizan modificaciones en los sistemas de ficheros, pues en estos casos puede suceder que ambos discos tengan sectores modificados que coincidan, desconociendo DRBD que datos son los que debe considerar como correctos y por tanto usar para la sincronización[7]. Si en uno de estos casos comprobamos el estado del sistema obtenemos:

```
m:res    cs         st                  ds                 p   mounted       fstype
0:drbd0  StandAlone Secondary/Unknown   UpToDate/DUnknow   r-----
```

Debiendo el administrador decidir qué nodo es el que posee la información correcta, e informar de ellos al sistema, forzando al nodo cuyos datos son erróneos (o se decide que son erróneos) a tomar el valor de erróneos, ejecutando para ello el comando:

```
drbdadm invalidate drbd0
```

Y proceder con posterioridad a forzar la conexión entre ambos nodos mediante el comando:

```
drbdadm connect drbd0
```

Con ello, observaremos como el estado en el nodo cuya información es correcta aparece como:

[7] En el siguiente tema veremos el uso de DRBD para sistemas de alta disponibilidad y como en ciertos casos podemos llegar a tener dos nodos en estado primario. En el tema actual puede forzarse esta situación poniendo reglas en los cortafuegos de ambos nodos que impidan la comunicación de un nodo con el otro.

```
m:res    cs         st                    ds                      p   mounted       fstype
0:drbd0  SyncSource Secondary/Secondary   UpToDate/Inconsistent   C
```

Y en el nodo con información incorrecta aparece como:

```
m:res    cs         st                    ds                      p   mounted       fstype
0:drbd0  SyncTarget Secondary/Secondary   Inconsistent/UpToDate   C
```

Quedando al cabo de un cierto tiempo ambos nodos en el estado:

```
m:res    cs         st                    ds                 p   mounted       fstype
0:drbd0  Connected  Secondary/Sedondary   UpToDate/UpToDate  C
```

Donde podemos ver que ambos nodos están conectados y en estado consistente.

Ejercicios.

1- Dos ordenadores, de direcciones IP 192.168.0.1 y 192.168.0.2 e interfaz de 1 Gbit entre ellos, poseen particiones /dev/sda1 y /dev/sda2 de igual tamaño en ambos nodos. Configurar DRBD para que la particiones /dev/sda1 sean sincronizada por la red, utilizando para almacenar los metadatos las particiones /dev/sda2.

2- Dos ordenadores, de direcciones IP 192.168.0.1 y 192.168.0.2 e interfaz de 1 Gbit entre ellos, poseen particiones /dev/sda1 y /dev/sda2, ambas iguales en ambos nodos. Configurar DRBD para que ambas particiones se sincronicen por la red suponiendo que los metadatos se almacenan internamente en cada partición.

3- Completar el ejercicio anterior ejecutando todos los comandos necesarios para que el sistema pueda funcionar de forma correcta, sincronizarse inicialmente, etc. Como dato adicional se os indica que las particiones /dev/sda1 y /dev/sda2 tienen un tamaño de 20 Gbytes.