drbdsetup

Name

`drbdsetup` — Setup tool for DRBD

Synopsis

drbdsetup {*device*} {disk} {*lower_device*} [-d {*size*}] [-p]

drbdsetup {*device*} {net} {*local_addr*} [*:port*] {*remote_addr*} [*:port*] {*protocol*} [-r {*rate*}] [-s {*size*}] [-c {*time*}] [-i {*time*}] [-k]

drbdsetup {*device*} {disconnect}

drbdsetup {*device*} {down}

drbdsetup {*device*} {primary}

drbdsetup {*device*} {secondary}

drbdsetup {*device*} {secondary_remote}

drbdsetup {*device*} {replicate}

drbdsetup {*device*} {wait_connect} [-t {*connect_timeout*}]

drbdsetup {*device*} {wait_sync} [-t {*connect_timeout*}]

drbdsetup {*device*} {show}

Description

drbdsetup is used to associate drbd devices with their lower level block devices, to set up drbd device pairs to mirror their lower level block devices and to inspect the configuration of running drbd devices.

Note

drbdsetup is a low level tool of the drbd program suite. It is used by the drbd, datadisk and drbdc scripts to communicate with the device driver.

Commands

Each drbd sub-command might require arguments and bring its own set of options. All values have default units which might be overruled by K, M or G. These units are defined in the usual way (eg. K = 2^10).

disk

Associates *device* with *lower_device* to store its data blocks on. The *device* is only ready for immediate use when also using the `-d` (or `-disk-size`) option. When not specifying the size of the *device*, it will become ready for use as soon as it is successfully connected to its partner device with the `net` command.

`-d`, `-disk-size` *size*

If you need to use the device before connecting, use this option to pass the *size* of the drbd device to the driver. Default unit is KB.

`-p`, `-do-panic`

If the driver of the *lower_device* reports an error to drbd, drbd passes this error on to the upper layers of the operating system by default. With the `-p` or `-do-panic` you can request that drbd triggers a kernel panic instead.

net

Sets up the *device* to listen on *local_addr:port* for incoming connections and to try to connect to *remote_addr:port*. If *port* is omitted, 7788 is used as default. On the tcp/ip link the specified *protocol* is used. Valid protocol spezifiers are A, B and C.

`-r`, `-sync-rate` *rate*

To ensure smooth operation of the application on top of drbd, it is possible to limit the bandwidth which may be used by background synchronisations. The default is 250 KB/sec, the default unit is KB.

`-c`, `-connect-int` *time*

In case it is not possible to connect to the remote drbd device immediately, drbd keeps on trying to connect. With this option you can set the time between two tries. The default value is 10 seconds, the unit is 1 second.

`-k`, `-skip-sync`

This option supresses the automatic start of the resynchronisation process, which is triggered as soon as two drbd devices are connected.

`-i`, `-ping-int` *time*

If the tcp/ip connection linking a drbd device pair is idle for more than *time* seconds, drbd will generate a keep-alive packet to check if its partner is still alive. The default is 10 seconds.

`-t`, `-timeout` *val*

If the partner node failes to send an expected response packet within *val* 10^{ths} of a second, the partner node is considered dead and therefore the tcp/ip connection is abandoned. The default value is 60 = 6 seconds.

`-s`, `-tl-size` *size*

The driver uses a cyclic data structure to keep track of sent data blocks. When using protocol A over a network with high latency, it might be necessary to increase the size of this data structure. If this is the case, the driver will write messages to the syslog saying that the transfer-log is too small. The default *size* is 256 entries.

primary

Sets the *device* into primary state, this means that applications (e.g. a file system) may open the *device* for read and write access. Data written to the *device* in primary state is mirrored to the device in secondary state.

If you set both devices of a connected drbd device pair to primary state, they will disconnect immediately.

secondary

Sets the *device* into secondary state, this operation fails as long as at least one application (or file system) has the device open for write access.

It is however, possible that both devices of a connected drbd device pair are in secondary state.

secondary_remote

Tries to set the partner of *device* into secondary state. This command will return successfully as long as it is possible to communicate with the partner, Its return value does not indicate if the partner device could change its state.

replicate

This forces the pair of connected drbd devices into SyncAll state, which means that all data blocks of the device in primary state are copied to the device in secondary state.

This command will fail if the *device* is not part of a connected device pair or if both devices of the pair are in secondary state.

wait_connect

Returns as soon as the *device* can communicate with its partner device.

`-t`, `-time` *connect_timeout*

This command will fail if the *device* can not communicate with its partner for *connect_timeout* seconds. The default value is 0 which disables the timeout machanism.

wait_sync

Returns as soon as the *device* leaves any synchronisation state and returns into connected state.

`-t,-time` *connect_timeout*

This command will fail if the *device* stays for *connect_timeout* seconds in unconnected state. The default value is 8 seconds. If it is set to 0, the command will forever wait for a connection.

disconnect

Removes the information set by the `net` command from the *device*. This means that the *device* goes into unconnected states and that it will no longer listen for incomming connections.

down

Removes all configuration information from the *device* and forces it back to unconfigured state.

show

Shows all available configuration information of the *device*.

Examples

Example 1. Setting up a device pair

In this example the hosts, *tc1* and *tc2*, are connected by a crossover cable via the interfaces *192.168.37.2* (on tc1) and *192.168.37.3* (on tc2). We want to turn /dev/hda6 into a virtually shared disk.

On tc2 we need:

```
$ drbdsetup /dev/nb0 disk /dev/hda6
$ drbdsetup /dev/nb0 net 192.168.37.2 192.168.37.3 B
```

On tc1 we need:

```
$ drbdsetup /dev/nb0 disk /dev/hda6
$ drbdsetup /dev/nb0 net 192.168.37.3 192.168.37.2 B
$ drbdsetup /dev/nb0 primary
$ cat /proc/drbd
version      : 58

0: cs:Connected st:Primary/Secondary ns:0 nr:0 dw:0 dr:0 of:0
1: cs:WFConnection st:Secondary/Unknown ns:0 nr:0 dw:0 dr:0 of:0
```

From /prod/drbd we can see, that our device pair is connected, and that the device is ready for use on tc1.

We can now run our application on top of it:

```
$ mkfs -b 4096 /dev/nb0
$ mount /dev/nb0 /mnt/mountpoint
```

Example 2. Snapshot a device to a third machine

In this example the hosts, *tc1* and *tc2*, are connected and tc2 is primary on the /dev/nb0 device. The resource should be snapshotted to *tc3's /dev/hda6*.

We need to prepare tc3:

```
$ drbdsetup /dev/nb0 disk /dev/hda6
$ drbdsetup /dev/nb0 net tc3 tc2 B
```

On tc2 we have to issue:

```
$ drbdsetup /dev/nb0 disconnect
$ drbdsetup /dev/nb0 net tc2 tac3 B -sync-rate 4M
$ #drbdsetup /dev/nb0 replicate###TODO describe why this is not needed.
$ drbdsetup /dev/nb0 wait_sync
$ drbdsetup /dev/nb0 disconnect
$ drbdsetup /dev/nb0 net tc2 tc1 B
```

Since the snapshot was taken without bringing the on disk file system into a consisten state we need theese commands on tc3:

```
$ drbdsetup /dev/nb0 down
$ fsck /dev/hda6
$ mount /dev/hda6 /some/mountpint
```

Author

Written by Philipp Reisner <philipp.reisner@gmx.at>.

Reporting Bugs

Report bugs to <drbd-devel@lists.sourceforge.net>.

Copyright