# NFS High Availability Clustering Using DRBD and Pacemaker on RHEL 9

Matt Kereczman, David Thomas, Michael Troutman, Dennis Hartinger

Version 1.4, 2022-10-25

# Table of Contents

# Chapter 1. Abstract

This guide can be used to deploy a high-availability (HA) two node NFS cluster on a LAN. To achieve this, we will use DRBD, Pacemaker, and Corosync on a Red Hat 9/AlmaLinux server. With this solution set up, data transfer between a client system and an NFS share should not be interrupted by failure of a node. In this guide, we will set up the HA NFS cluster, simulate a node failure, and verify that our data transfer continues.

For the testing environment in this guide, we will use RHEL/AlmaLinux 9 and DRBD as packaged by LINBIT. Please contact the LINBIT sales team (sales@linbit.com) for access to this software.

# Chapter 2. Assumptions

This guide assumes the following:

## 2.1. System Configuration

| Hostname | LVM Device | Volume Group | DRBD Device | External Interface | External IP | Crossover Interface | Crossover IP |
|---|---|---|---|---|---|---|---|
| node-a | /dev/sdb | vg_drbd | lv_r0 | eth0 | 192.168.10.201 | eth1 | 172.16.0.201 |
| node-b | /dev/sdb | vg_drbd | lv_r0 | eth0 | 192.168.10.202 | eth1 | 172.16.0.202 |

We'll need a virtual IP for services to run on. For this guide we will use 192.168.10.200

## 2.2. Firewall Configuration

Refer to your firewall documentation for how to open ports. You will need the following ports open in order for your cluster to function properly.

| Component | Protocol | Port |
|---|---|---|
| DRBD | TCP | 7788 |
| Corosync | UDP | 5404, 5405 |

## 2.3. System Registration Requirements

If you're using Red Hat Enterprise Linux (RHEL) then the target systems must be registered with Red Hat before you continue. If you're using AlmaLinux or some other RHEL 9 variant, no registration is required.

## 2.4. SELinux

If you have SELinux enabled and you're having issues consult your distribution's documentation for how to properly configure it, or disable it (not recommended).

# Chapter 3. Installation and Configuration

All target systems are registered with Red Hat prior to continuing.

## 3.1. Registering Nodes and Configuring Package Repositories

You will install DRBD and dependencies that you may need from LINBIT's repositories. To access those repositories you will need to have been set up in LINBIT's system and have access to the LINBIT Customer Portal. If you have not been set up in LINBIT's system, or if you want an evaluation account, you can contact a sales team member: sales@linbit.com.

Once you have access to the LINBIT Customer Portal, you can register your cluster nodes and configure repository access by using LINBIT's Python helper script. See the Register Nodes section of the Customer Portal for details about this script.

To download and run the LINBIT helper script, enter the following commands on all nodes, one node at a time:

```
# curl -O https://my.linbit.com/linbit-manage-node.py
# chmod +x ./linbit-manage-node.py
# ./linbit-manage-node.py
```

The script must be run as superuser.

If the error message `no python interpreter found :-(` is displayed when running `linbit-manage-node.py`, enter the command `dnf -y install python3` to install Python 3.

The script will prompt you to enter your LINBIT Customer Portal username and password. After validating your credentials, the script will list clusters and nodes (if you have any already registered) that are associated with your account.

### 3.1.1. Joining Nodes to a Cluster and Enabling Repositories

Select the cluster that you want to register the current node with. If you want the node to be the first node in a new cluster, select the "new cluster" option. The script will then ask you a series of questions about which repositories you want to enable. For RHEL 9, the possible repositories will be:

```
   1) pacemaker-2(Disabled)
   2) drbd-proxy-3.2(Disabled)
   3) drbd-9(Disabled)
```

The `drbd-9` repository includes the latest DRBD 9 version. It also includes other LINBIT software packages, including LINSTOR®, DRBD Reactor, LINSTOR GUI, and others.

Be sure to respond **yes** to the questions about installing LINBIT's public key to your keyring and writing the repository configuration file.

### 3.1.2. Verifying LINBIT Repositories

After the script completes, verify that you enabled LINBIT repositories.

```
# dnf info kmod-drbd
```

Output from the command should show the DNF package manager pulling package information from LINBIT repositories.

### 3.1.3. Including HA Packages from the Red Hat Subscription Repository

If you are on RHEL 9, then use the `dnf config-manager` command to enable `rhel-9-for-x86_64-highavailability-rpms`. If you are on AlmaLinux 9, then use the `dnf config-manager` to enable `HighAvailability`.

RHEL:

```
# dnf config-manager --set-enabled rhel-9-for-x86_64-highavailability-rpms
```

AlmaLinux:

```
# dnf config-manager --set-enabled highavailability
```

For our NFS cluster setup, we will only need to enable the, `drbd-9`, LINBIT repository.

## 3.2. Install DRBD

Install DRBD, the DRBD kernel module, and the Pacemaker configuration system, using the following command:

```
# dnf install drbd kmod-drbd pcs
```

As Pacemaker will be responsible for starting the DRBD service, prevent DRBD from starting at boot:

```
# systemctl disable drbd
```

## 3.3. Configure DRBD

Now that we've installed DRBD, we'll need to create our resource configuration file. To do this, create `/etc/drbd.d/r0.res` with the following contents:

```
resource r0 {
        protocol C;
        device    /dev/drbd0;
        disk      /dev/vg_drbd/lv_r0;
        meta-disk internal;
        on node-a {
            address 172.16.0.201:7788;
        }
        on node-b {
            address 172.16.0.202:7788;
        }
```

```
}
```

This is a minimal configuration file. There are many settings you can add and adjust to increase performance. See the *DRBD User's Guide* for more information.

Create the resource metadata by issuing the following command:

```
# drbdadm create-md r0
initializing activity log
initializing bitmap (32 KB) to all zero
Writing meta data...
New drbd meta data block successfully created.
success
```

This command should complete without any warnings - if you get messages about data being detected, and choose to proceed, you will lose data.

If you get any error messages when running the above `drbdadm` command, you can verify your DRBD configuration by entering the following command: `drbdadm dump all`.

Bring the device up on both nodes and verify their states by entering the following commands:

```
# drbdadm up r0
# drbdadm status
r0 role:Secondary
  disk:Inconsistent
  node-b role:Secondary
    peer-disk:Inconsistent
```

You can see in the above output that the resource is connected, but in an inconsistent state. To have your data replicated, you'll need to put the resource into a consistent state. There are two options:

1. Do a full sync of the device, which could potentially take a long time depending upon the size of the disk.
2. Skip the initial sync.

As we know this is a new setup with "just created" metadata and without any existing data on our device, we'll use option 2 and skip the initial sync. Enter the following commands on **node-a**:

```
# drbdadm new-current-uuid --clear-bitmap r0/0
# drbdadm status
r0 role:Secondary
  disk:UpToDate
  node-b role:Secondary
    peer-disk:UpToDate
```

## 3.4. Create a File System on Our DRBD Backed Device

After DRBD is initialized, we'll need to make 'node-a' Primary and create a file system on the DRBD device that we configured earlier as resource `r0`.

First, make the `node-a` Primary for resource `r0` by entering the following command on `node-a`:

```
# drbdadm primary r0
```

Next, on `node-a`, create a file system on the backing device.

```
# mkfs.xfs /dev/drbd0
```

You only need to create the file system on 'node-a', and **not** on 'node-b'. DRBD will take care of the replication.

Create the following directory on **both** 'node-a' and 'node-b':

```
# mkdir /mnt/drbd
```

This concludes the initial setup of the DRBD device. We will next configure Corosync on each of our nodes.

## 3.5. Install Pacemaker and Corosync

This section will cover installing Pacemaker and Corosync. We will use Pacemaker as our cluster resource manager (CRM). Corosync acts a messaging layer, providing information to Pacemaker about the state of our cluster nodes.

Enter the following commands to install the necessary packages and then enable the Pacemaker and Corosync services to start when the system boots:

```
# dnf install pacemaker corosync

# systemctl enable pacemaker corosync
Created symlink /etc/systemd/system/multi-user.target.wants/pacemaker.service to
/usr/lib/systemd/system/pacemaker.service.
Created symlink /etc/systemd/system/multi-user.target.wants/corosync.service to
/usr/lib/systemd/system/corosync.service.
```

### 3.5.1. Configure Corosync

Create and edit the file `/etc/corosync/corosync.conf`. It should look like this:

```
totem {
  version: 2
  secauth: off
  cluster_name: cluster
  transport: knet
  rrp_mode: passive
}

nodelist {
  node {
    ring0_addr: 172.16.0.201
    ring1_addr: 192.168.10.201
    nodeid: 1
    name: node-a
  }
  node {
    ring0_addr: 172.16.0.202
```

```
    ring1_addr: 192.168.10.202
    nodeid: 2
    name: node-b
  }
}

quorum {
  provider: corosync_votequorum
  two_node: 1
}

logging {
  to_syslog: yes
}
```

Now that Corosync has been configured, start the Corosync and Pacemaker services:

```
# systemctl start corosync pacemaker
```

Repeat the preceding Pacemaker and Corosync installation and configuration steps on each cluster node. Verify that everything has been started and is working correctly by entering the following `crm_mon` command. You should get output similar to this:

```
# crm_mon -rf -n1
Cluster Summary:
    * Stack: corosync
    * Current DC: node-a (version 2.0.5.linbit-1.0.el8-ba59be712) - partition with quorum
    * Last updated: Mon Feb 14 00:03:44 2022
    * Last change: Sun Feb 13 04:18:45 2022 by root crmd on node-a
    * 2 nodes configured

Node List:
    * Online: [ node-a node-b ]

Inactive resources:

Migration Summary:
    * Node node-a:
    * Node node-b:
```

## 3.6. Additional Cluster-Level Configuration

Since we only have two nodes, we will need to tell Pacemaker to ignore quorum. Run the following commands from either cluster node (but not both):

```
# pcs property set no-quorum-policy=ignore
```

Furthermore, we will not be configuring node level fencing (aka STONITH) in this guide. Disable STONITH using the following command:

```
# pcs property set stonith-enabled=false
```

Fencing/STONITH is an important part of HA clustering and should be used whenever possible. Disabling STONITH will lend the cluster vulnerable to split-brains and potential data corruption or loss.

For more information on Fencing and STONITH, you can review the ClusterLabs page on STONITH or contact the experts at LINBIT.

# Chapter 4. Configure Pacemaker for HA NFS

Now that we have initialized our cluster, we can begin configuring Pacemaker to manage our resources. Using `pcs cluster cib <file name>` pcs will create a configuration file. This is where we will write configuration changes before pushing the changes into the running cluster.

## 4.1. Configure DRBD Resources

DRBD is the first resource we will configure in Pacemaker. Use the following commands on 'node-a' as the commands below will be implemented throughout Pacemaker. This will pull a working version of the Cluster Information Base (CIB), configure the DRBD primitive (`p_drbd_r0`) and the promotable clone for the DRBD resource, and finally verify and commit the configuration changes:

```
# pcs cluster cib drbdconf

# pcs -f drbdconf resource create p_drbd_r0 ocf:linbit:drbd \
drbd_resource=r0 \
op start interval=0s timeout=240s \
stop interval=0s timeout=100s \
monitor interval=31s timeout=20s \
role=Unpromoted monitor interval=29s timeout=20s role=Promoted

# pcs -f drbdconf resource promotable p_drbd_r0 \
promoted-max=1 promoted-node-max=1 clone-max=2 clone-node-max=1 notify=true

# pcs cluster cib-push drbdconf
```

Now, if we run `crm_mon` to view the cluster state, we should see that Pacemaker is managing our DRBD device:

```
# crm_mon
Cluster Summary:
  * Stack: corosync
  * Current DC: node-a (version 2.1.2-4.el9-ada5c3b36e2) - partition with quorum
  * Last updated: Tue May 31 16:19:36 2022
  * Last change:  Tue May 31 16:18:22 2022 by root via cibadmin on node-a
  * 2 nodes configured
  * 2 resource instances configured

Node List:
  * Online: [ node-a node-b ]

Active Resources:
  * Clone Set: p_drbd_r0-clone [p_drbd_r0] (promotable):
    * Promoted: [ node-a ]
    * Unpromoted: [ node-b ]
```

## 4.2. Configure the File System Primitive

With DRBD running, we can now configure our file system within Pacemaker. We will need to configure `colocation` and `order` constraints to ensure that the file system is mounted where DRBD is Promoted, and only after DRBD has been promoted:

```
# pcs -f drbdconf resource create p_fs_drbd0 ocf:heartbeat:Filesystem \
device=/dev/drbd0 directory=/mnt/drbd fstype=xfs \
options=noatime,nodiratime \
op start interval="0" timeout="60s" \
```

```
stop interval="0" timeout="60s" \
monitor interval="20" timeout="40s"

# pcs -f drbdconf constraint order promote p_drbd_r0-clone then start p_fs_drbd0

# pcs -f drbdconf constraint colocation \
add p_fs_drbd0 with p_drbd_r0-clone INFINITY with-rsc-role=Promoted

# pcs cluster cib-push drbdconf

# crm_mon
```

Our `crm_mon` output should now look similar to this:

```
Cluster Summary:
  * Stack: corosync
  * Current DC: node-a (version 2.1.2-4.el9-ada5c3b36e2) - partition with quorum
  * Last updated: Wed Jun  1 07:10:07 2022
  * Last change:  Wed Jun  1 07:08:36 2022 by root via cibadmin on node-a
  * 2 nodes configured
  * 3 resource instances configured

Node List:
  * Online: [ node-a node-b ]

Active Resources:
  * Clone Set: p_drbd_r0-clone [p_drbd_r0] (promotable):
    * Promoted: [ node-b ]
    * Unpromoted: [ node-a ]
  * p_fs_drbd0  (ocf:heartbeat:Filesystem):  Started node-a
```

If you execute the `mount` command on 'node-b', you should see the DRBD device mounted at `/mnt/drbd`.

## 4.3. Configure the NFS Service and Exports

We will now configure our NFS server and the exported file system.

Verify that the 'nfs-utils' and 'rpcbind' packages are installed on both nodes, and that 'rpcbind' is enabled to start at boot:

```
# dnf install nfs-utils rpcbind
# systemctl enable rpcbind --now
```

Since the file system needs to be mounted locally before the NFS server can export it, we will need to set the appropriate ordering and colocation constraints for our resources: NFS will start on the node where the file system is mounted, and only after it's been mounted; then the 'exportfs' resources can start on the node where the file system is mounted.

In the example below, pcs will automatically give meaningful names to both our `order` and `colocation` constraints, with `order-` and `colocation-` being automatically prefixed to the resource name.

First, we will need to define the primitive for the NFS server. The NFS server requires a directory to store its special files. This needs to be placed on our DRBD device, since it needs to be present where and when the NFS server starts, to allow for smooth failover.

Run the following commands on one of the nodes:

```
# pcs -f drbdconf resource create p_nfsserver ocf:heartbeat:nfsserver \
nfs_shared_infodir=/mnt/drbd/nfs_shared_infodir nfs_ip=192.168.10.200 \
op start interval=0s timeout=40s \
stop interval=0s timeout=20s \
monitor interval=10s timeout=20s

# pcs -f drbdconf constraint colocation add p_nfsserver with p_fs_drbd0 INFINITY

# pcs -f drbdconf constraint order p_fs_drbd0 then p_nfsserver

# pcs cluster cib-push drbdconf

# crm_mon
```

After a few seconds, you should see the NFS server resource start on 'node-b':

```
# crm_mon
Cluster Summary:
...
Cluster Summary:
  * Stack: corosync
  * Current DC: node-a (version 2.1.2-4.el9-ada5c3b36e2) - partition with quorum
  * Last updated: Wed Jun  1 07:30:36 2022
  * Last change:  Wed Jun  1 07:30:33 2022 by root via cibadmin on node-a
  * 2 nodes configured
  * 4 resource instances configured

Node List:
  * Online: [ node-a node-b ]

Active Resources:
  * Clone Set: p_drbd_r0-clone [p_drbd_r0] (promotable):
    * Promoted: [ node-a ]
    * Unpromoted: [ node-b ]
  * p_fs_drbd0  (ocf:heartbeat:Filesystem):  Started node-a
  * p_nfsserver (ocf:heartbeat:nfsserver):   Started node-b

    * 2 nodes configured
    * 4 resources configured
```

With the NFS server running, we can create and configure our exports (from whichever node has 'p_nfsserver' started in your cluster):

```
# mkdir -p /mnt/drbd/exports/dir1
# chown nobody:nobody /mnt/drbd/exports/dir1

# pcs -f drbdconf resource create p_exportfs_dir1 ocf:heartbeat:exportfs \
clientspec=192.168.10.0/24 directory=/mnt/drbd/exports/dir1 fsid=1 \
unlock_on_stop=1 options=rw,sync \
op start interval=0s timeout=40s \
stop interval=0s timeout=120s \
monitor interval=10s timeout=20s

# pcs -f drbdconf constraint order p_nfsserver then p_exportfs_dir1

# pcs -f drbdconf constraint colocation add p_exportfs_dir1 with p_nfsserver INFINITY

# pcs cluster cib-push drbdconf
```

You should now be able to use the `showmount` command from a client system to see the exported

directories on the current Primary node ('node-a' in our example):

```
# showmount -e node-a
Export list for node-a:
/mnt/drbd/exports/dir1 192.168.122.0/24
```

## 4.4. Configure the Virtual IP

The Virtual IP (VIP) will provide consistent client access to the NFS export if one of our cluster nodes should fail. To do this, we will need to define a `colocation` constraint so that the VIP resource will always start on the node where the NFS export is currently active:

```
# pcs -f drbdconf resource create p_virtip_dir1 ocf:heartbeat:IPaddr2 \
ip=192.168.10.200 cidr_netmask=24 \
op monitor interval=20s timeout=20s \
start interval=0s timeout=20s \
stop interval=0s timeout=20s

# pcs -f drbdconf constraint order p_exportfs_dir1 then  p_virtip_dir1

# pcs -f drbdconf constraint colocation add p_virtip_dir1 with p_exportfs_dir1 INFINITY

# pcs cluster cib-push drbdconf
```

Now we should be able to use the `showmount` command from a client system, specifying the Virtual IP, and see the same output as we saw above:

```
# showmount -e 192.168.10.200
Export list for 192.168.10.200:
/mnt/drbd/exports/dir1 192.168.10.0/24
```

# Chapter 5. Test Cluster Failover

There are many ways we could test the persistence of our cluster's NFS export in a failover scenario. We could start a failover while copying a file from a client system to our NFS export. Or we could have a client system play an audio file stored on the NFS export during a failover, and so on.

In this guide, we'll describe how to simulate a file copy during a failover scenario. First, we'll create a "large" file in the mounted export directory using the `dd` utility, while failing over.

Mount the exported file system on a **client system**, and use `dd` to create a 1GiB file named 'write_test.out':

```
# mkdir /mnt/test_nfs_mount
# mount 192.168.10.200:/mnt/drbd/exports/dir1 /mnt/test_nfs_mount
# dd if=/dev/zero of=/mnt/test_nfs_mount/write_test.out bs=1M count=1024
```

Before the `dd` command on the client completes, reboot the Primary node with the following command:

```
# echo b > /proc/sysrq-trigger
```

The node should reboot immediately, causing the cluster to migrate all services to the peer node. The failover should not interrupt the `dd` command on the client system and the command should complete the file write without error.

To further verify our failover scenario, we can use the `drbdadm status` command on the node that was just rebooted:

```
# drbdadm status
r0 role:Secondary
  disk:UpToDate
    node-b role:Primary
        peer-disk:UpToDate
```

This output confirms that during the failover, our DRBD, Corosync, and Pacemaker HA solution promoted 'node-b' to a Primary role when we rebooted 'node-a'.

Congratulations! You've set up a high-availability NFS cluster using DRBD, Corosync, and Pacemaker. We'll leave configuring additional exports as an exercise for the reader. You may also want to try other scenarios where a client system is accessing an NFS cluster export during a failover.

# Chapter 6. Conclusion

This guide was created with RHEL/AlmaLinux systems in mind. If you have any questions regarding setting up a high-availability NFS cluster in your environment, you can contact the experts at LINBIT.

# Appendix A: Additional Information and Resources

- LINBIT's GitHub Organization: https://github.com/LINBIT/
- Join LINBIT's Community on Slack: https://www.linbit.com/join-the-linbit-drbd-linstor-slack/
- The DRBD® and LINSTOR® User's Guide: https://docs.linbit.com/
- The DRBD® and LINSTOR® Mailing Lists: https://lists.linbit.com/
    - drbd-announce: Announcements of new releases and critical bugs found
    - drbd-user: General discussion and community support
    - drbd-dev: Coordination of development

# Appendix B: Legalese

## B.1. Trademark Notice

LINBIT®, the LINBIT logo, DRBD®, the DRBD logo, LINSTOR®, and the LINSTOR logo are trademarks or registered trademarks of LINBIT in Austria, the EU, the United States, and many other countries. Other names mentioned in this document may be trademarks or registered trademarks of their respective owners.

## B.2. License Information