

# Highly Available NFS Storage with DRBD and Pacemaker

## SUSE Linux Enterprise High Availability Extension 11 SP4

June 19, 2015

www.suse.com

This document describes how to set up highly available NFS storage in a 2-node cluster, using the following components that are shipped with SUSE® Linux Enterprise High Availability Extension 11: DRBD (Distributed Replicated Block Device), LVM2 (Logical Volume Manager version 2), and Pacemaker, the cluster resource management framework.

### Introduction

NFS (the Network File System) is one of the most long-lived and ubiquitous networked storage solutions on Linux. NFS is widely deployed and at the time of writing, two NFS versions are of practical relevance to the enterprise: NFSv3 and NFSv4. The solution described in this document is applicable to NFS clients using either version.

### Prerequisites and Installation

Before you proceed with Section “Initial Configuration” (page 2), make sure that the following prerequisites are fulfilled.

#### Software Requirements

- SUSE® Linux Enterprise Server 11 is installed on all nodes that will be part of the cluster. All available online updates for the product are installed.
- SUSE Linux Enterprise High Availability Extension 11 is installed on all nodes that will be part of the cluster. All available online updates for the product are installed.

For instructions on how to install the High Availability Extension as add-on on top of SUSE Linux Enterprise Server, refer to Section “Installation as Add-on” (*Installation and Basic Setup*, ↑High Availability Guide).

In order to create a highly available NFS service, you need to have the following software packages installed:

- `pacemaker`: A cluster resource management framework which you will use to automatically start, stop, monitor, and migrate resources.
- `corosync`: A cluster messaging layer. Pacemaker is capable of using two cluster messaging layers: `Heartbeat` or `Corosync`. Corosync is the only supported Pacemaker messaging layer in SUSE Linux Enterprise 11.
- `drbd-utils` and `drbd-kmp-your_kernel`: Both belong to DRBD, the Kernel block-level synchronous replication facility which serves as an imported shared-nothing cluster building block.
- `lvm2`: Linux Logical Volume Management, version 2, which you may use for easy and flexible data management including online volume expansion and point-in-time snapshots.
- `nfs-kernel-server`: The in-kernel Linux NFS daemon. It serves locally mounted file systems out to clients via the NFS network protocol.

#### IMPORTANT: Product Registration And Updates

Proper registration at Novell Customer Center is mandatory for the system to receive updates. During the registration process, the respective online update repositories are automatically configured.

The repositories for installation and update automatically provide the package versions needed for the setup of highly available NFS storage as described in this document.

#### NOTE: Package Dependencies

You may be required to install packages other than the above-mentioned ones due to package dependencies. However, when using a package management utility such as `zypper` or YaST, these dependencies are automatically taken care of.

## Services at Boot Time

After you have installed the required packages, take care of a few settings applying to your boot process.

Use `chkconfig`, `insserv` or *YaST System Services (Runlevel)* to make sure that:

- OpenAIS *does* start automatically on system boot. This will also start Pacemaker.
- The `drbd` init script *does not* start automatically on system boot. Pacemaker takes care of all DRBD-related functionality—the init script is not needed for this purpose.

## Initial Configuration

This section describes the initial configuration of a highly available NFS exports in the context of the Pacemaker cluster manager. Note that the configuration described here will work for NFS clients using NFS versions 3 or 4.

### Configuring a DRBD Resource

First, it is necessary to configure a DRBD resource to hold your data. This resource will act as the Physical Volume of an LVM Volume Group to be created later. This example assumes that the LVM Volume Group is to be called `nfs`. Hence, the DRBD resource uses that same name.

It is highly recommended that you put your resource configuration in a file whose name is identical to that of the resource. As the file must reside in the `/etc/drbd.d` directory, this example uses the `/etc/drbd.d/nfs` file. Its contents should look similar to this:

```
resource nfs {
    device /dev/drbd0;
    disk /dev/sda1;
    meta-disk internal;
    on alice {
        address 10.0.42.1:7790;
    }
    on bob {
        address 10.0.42.2:7790;
    }
}
```

After you have created this resource, copy the DRBD configuration files to the other DRBD node, using either `scp` or `Csync2`. Proceed with initializing and synchronizing the resource, as specified in Procedure “Manually Configuring DRBD” (↑High Availability Guide). For information about `Csync2`, refer to Section “Transferring the Configuration to All Nodes” (*Installation and Basic Setup*, ↑High Availability Guide).

### Configuring LVM

To use LVM with DRBD, it is necessary to change some options in the LVM configuration file (`/etc/lvm/lvm.conf`) and to remove stale cache entries on the nodes:

1. Open `/etc/lvm/lvm.conf` in a text editor.
2. Search for the line starting with `filter` and edit it as follows:

```
filter = [ "r|/dev/sda1|" ]
```

This masks the underlying block device from the list of devices LVM scans for Physical Volume signatures. This way, LVM is instructed to read Physical Volume signatures from DRBD devices, rather than from the underlying backing block devices.

However, if you are using LVM *exclusively* on your DRBD devices, then you may also specify the LVM filter as such:

```
filter = [ "a|/dev/drbd.*|", "r|.*)" ]
```

3. In addition, disable the LVM cache by setting:

```
write_cache_state = 0
```

4. Save your changes to the file.

5. Delete `/etc/lvm/cache/.cache` to remove any stale cache entries.
6. Repeat the above steps on the peer node.

Now you can prepare the Physical Volume, create an LVM Volume Group with Logical Volumes and create file systems on the Logical Volumes. Before you start with the following steps, make sure to have activated the initial synchronization of your DRBD resource.

### **IMPORTANT: Automatic Synchronization**

Execute all of the following steps only on the node where your resource is currently in the primary role. It is *not* necessary to repeat the commands on the DRBD peer node as the changes are automatically synchronized.

1. To be able to create an LVM Volume Group, first initialize the DRBD resource as an LVM Physical Volume. To do so, issue the following command:

```
pvcreate /dev/drbd/by-res/nfs
```

2. Create an LVM Volume Group that includes this Physical Volume:

```
vgcreate nfs /dev/drbd/by-res/nfs
```

3. Create Logical Volumes in the Volume Group. This example assumes two Logical Volumes of 20 GB each, named `sales` and `engineering`:

```
lvcreate -n sales -L 20G nfs
lvcreate -n engineering -L 20G nfs
```

4. Activate the Volume Group and create file systems on the new Logical Volumes. This example assumes `ext3` as the file system type:

```
vgchange -ay nfs
mkfs -t ext3 /dev/nfs/sales
mkfs -t ext3 /dev/nfs/engineering
```

## **Initial Cluster Setup**

Use the YaST cluster module for the initial cluster setup. The process is documented in Section “Manual Cluster Setup (YaST)” (*Installation and Basic Setup*, ↑High Availability Guide) and includes the following basic steps:

1. Section “Defining the Communication Channels” (*Installation and Basic Setup*, ↑High Availability Guide)
2. Section “Defining Authentication Settings” (*Installation and Basic Setup*, ↑High Availability Guide)
3. Section “Transferring the Configuration to All Nodes” (*Installation and Basic Setup*, ↑High Availability Guide)

Then start the OpenAIS/Corosync service as described in Section “Bringing the Cluster Online” (*Installation and Basic Setup*, ↑High Availability Guide).

## **Creating a Basic Pacemaker Configuration**

Configure a STONITH device as described in *Fencing and STONITH* (↑High Availability Guide). Then use the following basic configuration.

For a highly available NFS server configuration that involves a 2-node cluster, you need to adjust the following global cluster options:

- `no-quorum-policy`: Must be set to `ignore`.
- `default-resource-stickiness`: Must be set to `200`.

For more information about global cluster options, refer to Section “Global Cluster Options” (*Configuration and Administration Basics*, ↑High Availability Guide).

To adjust the options, open the CRM shell as `root` (or any non-`root` user that is part of the `haclient` group) and issue the following commands:

```
crm(live) # configure
```

```
crm(live)configure# property no-quorum-policy="ignore"
crm(live)configure# rsc_defaults resource-stickiness="200"
crm(live)configure# commit
```

## Cluster Resources for a Highly Available NFS Service

A highly available NFS service consists of the following cluster resources:

### DRBD Master/Slave Resource (page 4)

The resource is used to replicate data and is switched from and to the Primary and Secondary roles as deemed necessary by the cluster resource manager.

### NFS Kernel Server Resource (page 4)

With this resource, Pacemaker ensures that the NFS server daemons are always available.

### LVM and File System Resources (page 5)

The LVM Volume Group is made available on whichever node currently holds the DRBD resource in the Primary role. Apart from that, you need resources for one or more file systems residing on any of the Logical Volumes in the Volume Group. They are mounted by the cluster manager wherever the Volume Group is active.

### NFSv4 Virtual File System Root (page 6)

A virtual NFS root export. (Only needed for NFSv4 clients).

### Non-root NFS Exports (page 6)

One or more NFS exports, typically corresponding to the file system mounted from LVM Logical Volumes.

### Resource for Floating IP Address (page 7)

A virtual, floating cluster IP address, allowing NFS clients to connect to the service no matter which physical node it is running on.

How to configure these resources (using the `crm` shell) is covered in detail in the following sections.

### **Example 1:** NFS Scenario

The following configuration examples assume that `10.9.9.180` is the virtual IP address to use for a NFS server which serves clients in the `10.9.9.0/24` subnet.

The service is to host an NFSv4 virtual file system root hosted from `/srv/nfs`, with exports served from `/srv/nfs/sales` and `/srv/nfs/engineering`.

Into these export directories, the cluster will mount `ext3` file systems from Logical Volumes named `sales` and `engineering`, respectively. Both of these Logical Volumes will be part of a highly available Volume Group, named `nfs`, which is hosted on a DRBD device.

## DRBD Master/Slave Resource

To configure this resource, issue the following commands from the `crm` shell:

```
crm(live)# configure
crm(live)configure# primitive p_drbd_nfs \
  ocf:linbit:drbd \
    params drbd_resource="nfs" \
    op monitor interval="15" role="Master" \
    op monitor interval="30" role="Slave"
crm(live)configure# ms ms_drbd_nfs p_drbd_nfs \
  meta master-max="1" master-node-max="1" clone-max="2" \
  clone-node-max="1" notify="true"
crm(live)configure# commit
```

This will create a Pacemaker Master/Slave resource corresponding to the DRBD resource `nfs`. Pacemaker should now activate your DRBD resource on both nodes, and promote it to the Master role on one of them.

Check this with the `crm_mon` command, or by looking at the contents of `/proc/drbd`.

## NFS Kernel Server Resource

In the `crm` shell, the resource for the NFS server daemons must be configured as a *clone* of an `lsb` resource type, as follows:

```
crm(live)configure# primitive p_lsb_nfsserver \
  lsb:nfsserver \
  op monitor interval="30s"
crm(live)configure# clone cl_lsb_nfsserver p_lsb_nfsserver
crm(live)configure# commit
```

#### NOTE: Resource Type Name and NFS Server init Script

The name of the `lsb` resource type must be *exactly* identical to the file name of the NFS server init script, installed under `/etc/init.d`. SUSE Linux Enterprise ships the init script as `/etc/init.d/nfsserver` (package `nfs-kernel-server`). Hence the resource must be of type `lsb:nfsserver`.

After you have committed this configuration, Pacemaker should start the NFS Kernel server processes on both nodes.

## LVM and File System Resources

1. Configure LVM and file system type resources as follows (but do *not commit* this configuration yet):

```
crm(live)configure# primitive p_lvm_nfs \
  ocf:heartbeat:LVM \
  params volgrpname="nfs" \
  op monitor interval="30s"
crm(live)configure# primitive p_fs_engineering \
  ocf:heartbeat:Filesystem \
  params device=/dev/nfs/engineering \
  directory=/srv/nfs/engineering \
  fstype=ext3 \
  op monitor interval="10s"
crm(live)configure# primitive p_fs_sales \
  ocf:heartbeat:Filesystem \
  params device=/dev/nfs/sales \
  directory=/srv/nfs/sales \
  fstype=ext3 \
  op monitor interval="10s"
```

2. Combine these resources into a Pacemaker resource *group*:

```
crm(live)configure# group g_nfs \
  p_lvm_nfs p_fs_engineering p_fs_sales
```

3. Add the following constraints to make sure that the group is started on the same node where the DRBD Master/Slave resource is in the Master role:

```
crm(live)configure# order o_drbd_before_nfs inf: \
  ms_drbd_nfs:promote g_nfs:start
crm(live)configure# colocation c_nfs_on_drbd inf: \
  g_nfs ms_drbd_nfs:Master
```

4. Commit this configuration:

```
crm(live)configure# commit
```

After these changes have been committed, Pacemaker does the following:

- It activates all Logical Volumes of the `nfs` LVM Volume Group on the same node where DRBD is in the Primary role. Confirm this with `vgdisplay` or `lvs`.
- It mounts the two Logical Volumes to `/srv/nfs/sales` and `/srv/nfs/engineering` on the same node. Confirm this with `mount` (or by looking at `/proc/mounts`).

## NFS Export Resources

Once your DRBD, LVM, and file system resources are working properly, continue with the resources managing your NFS exports. To create highly available NFS export resources, use the `exportfs` resource type.

## NFSv4 Virtual File System Root

If clients exclusively use NFSv3 to connect to the server, you do not need this resource. In this case, continue with Section “Non-root NFS Exports” (page 6).

1. To enable NFSv4 support, configure one—and only one—NFS export whose `fsid` option is either 0 (as used in the example below) or the string `root`. This is the root of the virtual NFSv4 file system.

```
crm(live)configure# primitive p_exportfs_root \  
  ocf:heartbeat:exportfs \  
  params fsid=0 \  
    directory="/srv/nfs" \  
    options="rw,crossmnt" \  
    clientspec="10.9.9.0/255.255.255.0" \  
  op monitor interval="30s" \  
crm(live)configure# clone cl_exportfs_root p_exportfs_root
```

This resource does not hold any actual NFS-exported data, merely the empty directory (`/srv/nfs`) that the other NFS exports are mounted into. Since there is no shared data involved here, we can safely *clone* this resource.

2. Since any data should be exported only on nodes where this clone has been properly started, add the following constraints to the configuration:

```
crm(live)configure# order o_root_before_nfs inf: \  
  cl_exportfs_root g_nfs:start \  
crm(live)configure# colocation c_nfs_on_root inf: \  
  g_nfs cl_exportfs_root \  
crm(live)configure# commit
```

After this, Pacemaker should start the NFSv4 virtual file system root on both nodes.

3. Check the output of the `exportfs -v` command to verify this.

## Non-root NFS Exports

All NFS exports that do *not* represent an NFSv4 virtual file system root must set the `fsid` option to either a unique positive integer (as used in the example), or a UUID string (32 hex digits with arbitrary punctuation).

1. Create NFS exports with the following commands:

```
crm(live)configure# primitive p_exportfs_sales \  
  ocf:heartbeat:exportfs \  
  params fsid=1 \  
    directory="/srv/nfs/sales" \  
    options="rw,mountpoint" \  
    clientspec="10.9.9.0/255.255.255.0" \  
    wait_for_lease_time_on_stop=true \  
  op monitor interval="30s" \  
crm(live)configure# primitive p_exportfs_engineering \  
  ocf:heartbeat:exportfs \  
  params fsid=2 \  
    directory="/srv/nfs/engineering" \  
    options="rw,mountpoint" \  
    clientspec="10.9.9.0/255.255.255.0" \  
    wait_for_lease_time_on_stop=true \  
  op monitor interval="30s"
```

2. After you have created these resources, add them to the existing `g_nfs` resource group:

```
crm(live)configure# edit g_nfs
```

3. Edit the group configuration so it looks like this:

```
group g_nfs \  
  p_lvm_nfs p_fs_engineering p_fs_sales \  
  p_exportfs_engineering p_exportfs_sales
```

#### 4. Commit this configuration:

```
crm(live)configure# commit
```

Pacemaker will export the NFS virtual file system root and the two other exports

#### 5. Confirm that the NFS exports are set up properly:

```
exportfs -v
```

### Resource for Floating IP Address

To enable smooth and seamless failover, your NFS clients will be connecting to the NFS service via a floating cluster IP address, rather than via any of the hosts' physical IP addresses.

#### 1. Add the following resource to the cluster configuration:

```
crm(live)configure# primitive p_ip_nfs \  
    ocf:heartbeat:IPaddr2 \  
    params ip=10.9.9.180 \  
    cidr_netmask=24 \  
    op monitor interval="30s"
```

#### 2. Add the IP address to the resource group (like you did with the `exportfs` resources):

```
crm(live)configure# edit g_nfs
```

This is the final setup of the resource group:

```
group g_nfs \  
    p_lvm_nfs p_fs_engineering p_fs_sales \  
    p_exportfs_engineering p_exportfs_sales \  
    p_ip_nfs
```

#### 3. Complete the cluster configuration:

```
crm(live)configure# commit
```

At this point Pacemaker will set up the floating cluster IP address.

#### 4. Confirm that the cluster IP is running correctly:

```
ip address show
```

The cluster IP should be added as a *secondary* address to whatever interface is connected to the 10.9.9.0/24 subnet.

#### **NOTE: Connection of Clients**

There is no way to make your NFS exports bind to *just* this cluster IP address. The Kernel NFS server always binds to the wildcard address (0.0.0.0 for IPv4). However, your clients must connect to the NFS exports through the floating IP address *only*, otherwise the clients will suffer service interruptions on cluster failover.

### Using the NFS Service

This section outlines how to use the highly available NFS service from an NFS client. It covers NFS clients using NFS versions 3 and 4.

#### Connecting with NFS Version 3

To connect to the highly available NFS service with an NFS version 3 client, make sure to use the *virtual IP address* to connect to the cluster, rather than a physical IP configured on one of the cluster nodes' network interfaces. NFS version 3 requires that you specify the *full* path of the NFS export on the server.

In its simplest form, the command to mount the NFS export with NFSv3 looks like this:

```
mount -t nfs 10.9.9.180:/srv/nfs/engineering /home/engineering
```

For selecting a specific transport protocol (`proto`) and maximum read and write request sizes (`rsize` and `wsize`):



```
mount -t nfs -o proto=udp,rsiz=32768,wsiz=32768 \  
10.9.9.180:/srv/nfs/engineering /home/engineering
```

For further NFSv3 mount options, consult the `nfs` man page.

## Connecting with NFS Version 4

### IMPORTANT

Connecting to the NFS server with NFSv4 *will not work* unless you have configured an NFSv4 virtual file system root. For details on how to set this up, see Section “NFSv4 Virtual File System Root” (page 6).

To connect to a highly available NFS service, NFSv4 clients must use the floating cluster IP address (as with NFSv3), rather than any of the cluster nodes' physical NIC addresses. NFS version 4 requires that you specify the NFS export path *relative* to the root of the virtual file system. Thus, to connect to the `engineering` export, you would use the following `mount` command (note the `nfs4` file system type):

```
mount -t nfs4 10.9.9.180:/engineering /home/engineering
```

As with NFSv3, there are a multitude of mount options for NFSv4. For example, selecting larger-than-default maximum request sizes may be desirable:

```
mount -t nfs4 -o rsiz=32768,wsiz=32768 \  
10.9.9.180:/engineering /home/engineering
```

## Ensuring Smooth Cluster Failover

For NFSv4 connections, the NFSv4 lease time can influence cluster failover.

### NFSv4 Lease Time

For exports used by NFSv4 clients, the NFS server hands out *leases* which the client holds for the duration of file access and then relinquishes. If the NFS resource shuts down or migrates while one of its clients is holding a lease on it, the client never lets go of the lease — at which point the server considers the lease expired after the expiration of a certain grace period.

While any clients are still holding unexpired leases, the NFS server maintains an open file handle on the underlying file system, preventing it from being unmounted. The `exportfs` resource agent handles this through the `wait_for_lease_time_on_stop` resource parameter. When set, it simply rides out the grace period before declaring the resource properly stopped. At that point, the cluster can proceed by unmounting the underlying file system (and subsequently, bringing both the file system and the NFS export up on the other cluster node).

On some systems, this grace period is set to 90 seconds by default, which may cause a prolonged period of NFS lock-up that clients experience during an NFS service transition on the cluster.

### Modifying NFSv4 Lease Time

To allow for faster failover, you may decrease the grace period.

#### NOTE: Modification and Restart

To modify the file, the NFS Kernel server must be stopped. Any modification only becomes active after the NFS Kernel server is restarted.

To set the grace period to 10 seconds, edit `/etc/sysconfig/nfs` and adjust the value of the following parameter:

```
NFSD_V4_GRACE=10
```

Restart the NFS services.

## Legal Notice

Copyright © 2011 Novell Inc., doc-team@suse.de. Used with permission.

Copyright © 2010, 2011 LINBIT HA-Solutions GmbH



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

**License information** The text and illustrations in this document are licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported license ("CC BY-NC-ND-3.0").

A summary of CC BY-NC-ND-3.0 is available at <http://creativecommons.org/licenses/by-nc-nd/3.0/>.

The full license text is available at <http://creativecommons.org/licenses/by-nc-nd/3.0/legalcode>.

In accordance with CC BY-NC-ND-3.0, if you distribute this document, you must provide the URL for the original version: <http://www.linbit.com/en/education/tech-guides/highly-available-nfs-with-drbd-and-pacemaker/>.



Created by SUSE® with XSL-FO