Red Hat Enterprise Linux 7

High Availability Add-On Administration

Configuring Red Hat High Availability deployments

Last Updated: 2020-11-18

Red Hat Enterprise Linux 7 High Availability Add-On Administration

Configuring Red Hat High Availability deployments

Steven Levine
Red Hat Customer Content Services
slevine@redhat.com

Legal Notice

Abstract

High Availability Add-On Administration provides sample cluster configurations that utilize the High Availability Add-On for Red Hat Enterprise Linux 7.

Table of Contents

CHAPTER 1. CREATING A RED HAT HIGH-AVAILABILITY CLUSTER WITH PACEMAKER

This chapter describes the procedure for creating a Red Hat High Availability two-node cluster using **pcs**. After you have created a cluster, you can configure the resources and resource groups that you require.

Configuring the cluster provided in this chapter requires that your system include the following components:

- 2 nodes, which will be used to create the cluster. In this example, the nodes used are **z1.example.com** and **z2.example.com**.
- Network switches for the private network, required for communication among the cluster nodes and other cluster hardware such as network power switches and Fibre Channel switches.
- A power fencing device for each node of the cluster. This example uses two ports of the APC power switch with a host name of **zapc.example.com**.

This chapter is divided into three sections.

- Section 1.1, "Cluster Software Installation" provides the procedure for installing the cluster software.
- Section 1.2, "Cluster Creation" provides the procedure for configuring a two-node cluster.
- Section 1.3, "Fencing Configuration" provides the procedure for configuring fencing devices for each node of the cluster.

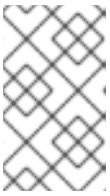
1.1. CLUSTER SOFTWARE INSTALLATION

The procedure for installing and configuring a cluster is as follows.

1. On each node in the cluster, install the Red Hat High Availability Add-On software packages along with all available fence agents from the High Availability channel.

```
# yum install pcs pacemaker fence-agents-all
```

2. If you are running the **firewalld** daemon, execute the following commands to enable the ports that are required by the Red Hat High Availability Add-On.



NOTE

You can determine whether the **firewalld** daemon is installed on your system with the **rpm -q firewalld** command. If the **firewalld** daemon is installed, you can determine whether it is running with the **firewall-cmd --state** command.

```
# firewall-cmd --permanent --add-service=high-availability
# firewall-cmd --add-service=high-availability
```

3. In order to use **pcs** to configure the cluster and communicate among the nodes, you must set a password on each node for the user ID **hacluster**, which is the **pcs** administration account. It is recommended that the password for user **hacluster** be the same on each node.

```
# passwd hacluster
Changing password for user hacluster.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
```

4. Before the cluster can be configured, the **pcsd** daemon must be started and enabled to boot on startup on each node. This daemon works with the **pcs** command to manage configuration across the nodes in the cluster.

   On each node in the cluster, execute the following commands to start the **pcsd** service and to enable **pcsd** at system start.

```
# systemctl start pcsd.service
# systemctl enable pcsd.service
```

5. Authenticate the **pcs** user **hacluster** for each node in the cluster on the node from which you will be running **pcs**.

   The following command authenticates user **hacluster** on **z1.example.com** for both of the nodes in the example two-node cluster, **z1.example.com** and **z2.example.com**.

```
[root@z1 ~]# pcs cluster auth z1.example.com z2.example.com
Username: hacluster
Password:
z1.example.com: Authorized
z2.example.com: Authorized
```

1.2. CLUSTER CREATION

This procedure creates a Red Hat High Availability Add-On cluster that consists of the nodes **z1.example.com** and **z2.example.com**.

1. Execute the following command from **z1.example.com** to create the two-node cluster **my_cluster** that consists of nodes **z1.example.com** and **z2.example.com**. This will propagate the cluster configuration files to both nodes in the cluster. This command includes the **--start** option, which will start the cluster services on both nodes in the cluster.

```
[root@z1 ~]# pcs cluster setup --start --name my_cluster \
z1.example.com z2.example.com
z1.example.com: Succeeded
z1.example.com: Starting Cluster...
z2.example.com: Succeeded
z2.example.com: Starting Cluster...
```

2. Enable the cluster services to run on each node in the cluster when the node is booted.

NOTE

For your particular environment, you may choose to leave the cluster services disabled by skipping this step. This allows you to ensure that if a node goes down, any issues with your cluster or your resources are resolved before the node rejoins the cluster. If you leave the cluster services disabled, you will need to manually start the services when you reboot a node by executing the **pcs cluster start** command on that node.

```
[root@z1 ~]# pcs cluster enable --all
```

You can display the current status of the cluster with the **pcs cluster status** command. Because there may be a slight delay before the cluster is up and running when you start the cluster services with the **--start** option of the **pcs cluster setup** command, you should ensure that the cluster is up and running before performing any subsequent actions on the cluster and its configuration.

```
[root@z1 ~]# pcs cluster status
Cluster Status:
 Last updated: Thu Jul 25 13:01:26 2013
 Last change: Thu Jul 25 13:04:45 2013 via crmd on z2.example.com
 Stack: corosync
 Current DC: z2.example.com (2) - partition with quorum
 Version: 1.1.10-5.el7-9abe687
 2 Nodes configured
 0 Resources configured
```

1.3. FENCING CONFIGURATION

You must configure a fencing device for each node in the cluster. For information about the fence configuration commands and options, see the *Red Hat Enterprise Linux 7 High Availability Add-On Reference*. For general information on fencing and its importance in a Red Hat High Availability cluster, see Fencing in a Red Hat High Availability Cluster .

NOTE

When configuring a fencing device, attention should be given to whether that device shares power with any nodes or devices in the cluster. If a node and its fence device do share power, then the cluster may be at risk of being unable to fence that node if the power to it and its fence device should be lost. Such a cluster should either have redundant power supplies for fence devices and nodes, or redundant fence devices that do not share power. Alternative methods of fencing such as SBD or storage fencing may also bring redundancy in the event of isolated power losses.

This example uses the APC power switch with a host name of **zapc.example.com** to fence the nodes, and it uses the **fence_apc_snmp** fencing agent. Because both nodes will be fenced by the same fencing agent, you can configure both fencing devices as a single resource, using the **pcmk_host_map** and **pcmk_host_list** options.

You create a fencing device by configuring the device as a **stonith** resource with the **pcs stonith create** command. The following command configures a **stonith** resource named **myapc** that uses the **fence_apc_snmp** fencing agent for nodes **z1.example.com** and **z2.example.com**. The

pcmk_host_map option maps **z1.example.com** to port 1, and **z2.example.com** to port 2. The login value and password for the APC device are both **apc**. By default, this device will use a monitor interval of sixty seconds for each node.

Note that you can use an IP address when specifying the host name for the nodes.

```
[root@z1 ~]# pcs stonith create myapc fence_apc_snmp \
ipaddr="zapc.example.com" pcmk_host_map="z1.example.com:1;z2.example.com:2" \
pcmk_host_check="static-list" pcmk_host_list="z1.example.com,z2.example.com" \
login="apc" passwd="apc"
```

NOTE

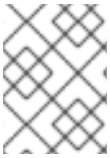
When you create a **fence_apc_snmp stonith** device, you may see the following warning message, which you can safely ignore:

```
Warning: missing required option(s): 'port, action' for resource type:
stonith:fence_apc_snmp
```

The following command displays the parameters of an existing STONITH device.

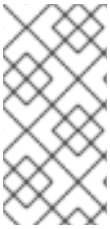
```
[root@rh7-1 ~]# pcs stonith show myapc
 Resource: myapc (class=stonith type=fence_apc_snmp)
  Attributes: ipaddr=zapc.example.com pcmk_host_map=z1.example.com:1;z2.example.com:2
pcmk_host_check=static-list pcmk_host_list=z1.example.com,z2.example.com login=apc
passwd=apc
  Operations: monitor interval=60s (myapc-monitor-interval-60s)
```

After configuring your fence device, you should test the device. For information on testing a fence device, see Fencing: Configuring Stonith in the *High Availability Add-On Reference* .



NOTE

Do not test your fence device by disabling the network interface, as this will not properly test fencing.



NOTE

Once fencing is configured and a cluster has been started, a network restart will trigger fencing for the node which restarts the network even when the timeout is not exceeded. For this reason, do not restart the network service while the cluster service is running because it will trigger unintentional fencing on the node.

CHAPTER 2. AN ACTIVE/PASSIVE APACHE HTTP SERVER IN A RED HAT HIGH AVAILABILITY CLUSTER

This chapter describes how to configure an active/passive Apache HTTP server in a two-node Red Hat Enterprise Linux High Availability Add-On cluster using **pcs** to configure cluster resources. In this use case, clients access the Apache HTTP server through a floating IP address. The web server runs on one of two nodes in the cluster. If the node on which the web server is running becomes inoperative, the web server starts up again on the second node of the cluster with minimal service interruption.

Figure 2.1, "Apache in a Red Hat High Availability Two-Node Cluster" shows a high-level overview of the cluster. The cluster is a two-node Red Hat High Availability cluster which is configured with a network power switch and with shared storage. The cluster nodes are connected to a public network, for client access to the Apache HTTP server through a virtual IP. The Apache server runs on either Node 1 or Node 2, each of which has access to the storage on which the Apache data is kept.

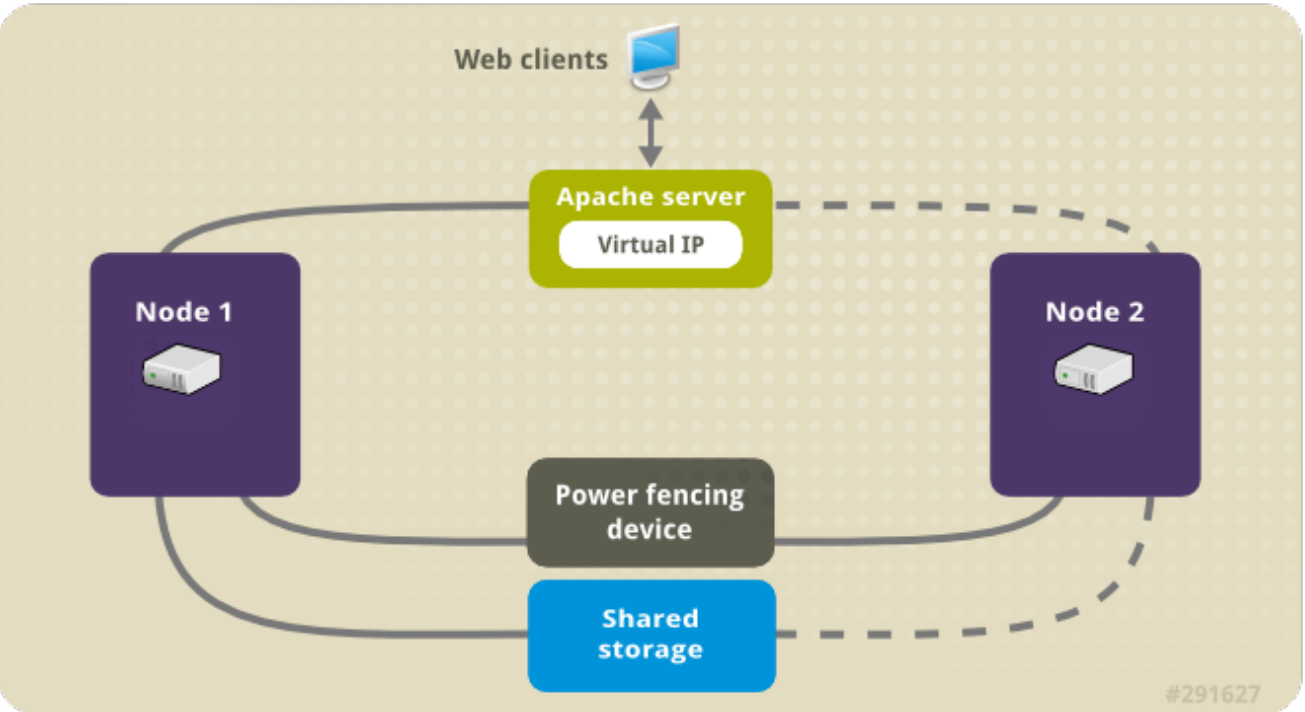


Figure 2.1. Apache in a Red Hat High Availability Two-Node Cluster

This use case requires that your system include the following components:

- A two-node Red Hat High Availability cluster with power fencing configured for each node. This procedure uses the cluster example provided in Chapter 1, *Creating a Red Hat High-Availability Cluster with Pacemaker*.
- A public virtual IP address, required for Apache.
- Shared storage for the nodes in the cluster, using iSCSI, Fibre Channel, or other shared network block device.

The cluster is configured with an Apache resource group, which contains the cluster components that the web server requires: an LVM resource, a file system resource, an IP address resource, and a web server resource. This resource group can fail over from one node of the cluster to the other, allowing either node to run the web server. Before creating the resource group for this cluster, you will perform the following procedures:

1. Configure an **ext4** file system mounted on the logical volume **my_lv**, as described in Section 2.1, "Configuring an LVM Volume with an ext4 File System".
2. Configure a web server, as described in Section 2.2, "Web Server Configuration".
3. Ensure that only the cluster is capable of activating the volume group that contains **my_lv**, and that the volume group will not be activated outside of the cluster on startup, as described in Section 2.3, "Exclusive Activation of a Volume Group in a Cluster" .

After performing these procedures, you create the resource group and the resources it contains, as described in Section 2.4, "Creating the Resources and Resource Groups with the pcs Command" .

2.1. CONFIGURING AN LVM VOLUME WITH AN EXT4 FILE SYSTEM

This use case requires that you create an LVM logical volume on storage that is shared between the nodes of the cluster.

The following procedure creates an LVM logical volume and then creates an **ext4** file system on that volume. In this example, the shared partition **/dev/sdb1** is used to store the LVM physical volume from which the LVM logical volume will be created.

NOTE

LVM volumes and the corresponding partitions and devices used by cluster nodes must be connected to the cluster nodes only.

Since the **/dev/sdb1** partition is storage that is shared, you perform this procedure on one node only,

1. Create an LVM physical volume on partition **/dev/sdb1**.

```
# pvcreate /dev/sdb1
  Physical volume "/dev/sdb1" successfully created
```

2. Create the volume group **my_vg** that consists of the physical volume **/dev/sdb1**.

```
# vgcreate my_vg /dev/sdb1
  Volume group "my_vg" successfully created
```

3. Create a logical volume using the volume group **my_vg**.

```
# lvcreate -L450 -n my_lv my_vg
  Rounding up size to full physical extent 452.00 MiB
  Logical volume "my_lv" created
```

   You can use the **lvs** command to display the logical volume.

```
# lvs
  LV      VG      Attr      LSize   Pool Origin Data%  Move Log Copy%  Convert
  my_lv   my_vg   -wi-a---- 452.00m
  ...
```

4. Create an **ext4** file system on the logical volume **my_lv**.

```
# mkfs.ext4 /dev/my_vg/my_lv
mke2fs 1.42.7 (21-Jan-2013)
Filesystem label=
OS type: Linux
...
```

2.2. WEB SERVER CONFIGURATION

The following procedure configures an Apache HTTP server.

1. Ensure that the Apache HTTP server is installed on each node in the cluster. You also need the **wget** tool installed on the cluster to be able to check the status of the Apache HTTP server.

   On each node, execute the following command.

   ```
   # yum install -y httpd wget
   ```

2. In order for the Apache resource agent to get the status of the Apache HTTP server, ensure that the following text is present in the **/etc/httpd/conf/httpd.conf** file on each node in the cluster, and ensure that it has not been commented out. If this text is not already present, add the text to the end of the file.

   ```
   <Location /server-status>
      SetHandler server-status
      Require local
   </Location>
   ```

3. When you use the **apache** resource agent to manage Apache, it does not use **systemd**. Because of this, you must edit the **logrotate** script supplied with Apache so that it does not use **systemctl** to reload Apache.

   Remove the following line in the **/etc/logrotate.d/httpd** file on each node in the cluster.

   ```
   /bin/systemctl reload httpd.service > /dev/null 2>/dev/null || true
   ```

   Replace the line you removed with the following three lines.

   ```
   /usr/bin/test -f /run/httpd.pid >/dev/null 2>/dev/null &&
   /usr/bin/ps -q $(/usr/bin/cat /run/httpd.pid) >/dev/null 2>/dev/null &&
   /usr/sbin/httpd -f /etc/httpd/conf/httpd.conf -c "PidFile /run/httpd.pid" -k graceful > /dev/null
   2>/dev/null || true
   ```

4. Create a web page for Apache to serve up. On one node in the cluster, mount the file system you created in Section 2.1, “Configuring an LVM Volume with an ext4 File System” , create the file **index.html** on that file system, then unmount the file system.

   ```
   # mount /dev/my_vg/my_lv /var/www/
   # mkdir /var/www/html
   # mkdir /var/www/cgi-bin
   # mkdir /var/www/error
   # restorecon -R /var/www
   # cat <<-END >/var/www/html/index.html
   <html>
   ```

```
<body>Hello</body>
</html>
END
# umount /var/www
```

2.3. EXCLUSIVE ACTIVATION OF A VOLUME GROUP IN A CLUSTER

The following procedure configures the volume group in a way that will ensure that only the cluster is capable of activating the volume group, and that the volume group will not be activated outside of the cluster on startup. If the volume group is activated by a system outside of the cluster, there is a risk of corrupting the volume group's metadata.

This procedure modifies the **volume_list** entry in the **/etc/lvm/lvm.conf** configuration file. Volume groups listed in the **volume_list** entry are allowed to automatically activate on the local node outside of the cluster manager's control. Volume groups related to the node's local root and home directories should be included in this list. All volume groups managed by the cluster manager must be excluded from the **volume_list** entry. Note that this procedure does not require the use of **clvmd**.

Perform the following procedure on each node in the cluster.

1. Execute the following command to ensure that **locking_type** is set to 1 and that **use_lvmetad** is set to 0 in the **/etc/lvm/lvm.conf** file. This command also disables and stops any **lvmetad** processes immediately.

   ```
   # lvmconf --enable-halvm --services --startstopservices
   ```

2. Determine which volume groups are currently configured on your local storage with the following command. This will output a list of the currently-configured volume groups. If you have space allocated in separate volume groups for root and for your home directory on this node, you will see those volumes in the output, as in this example.

   ```
   # vgs --noheadings -o vg_name
     my_vg
     rhel_home
     rhel_root
   ```

3. Add the volume groups other than **my_vg** (the volume group you have just defined for the cluster) as entries to **volume_list** in the **/etc/lvm/lvm.conf** configuration file.

   For example, if you have space allocated in separate volume groups for root and for your home directory, you would uncomment the **volume_list** line of the **lvm.conf** file and add these volume groups as entries to **volume_list** as follows. Note that the volume group you have just defined for the cluster (**my_vg** in this example) is not in this list.

   ```
   volume_list = [ "rhel_root", "rhel_home" ]
   ```

   NOTE

   If no local volume groups are present on a node to be activated outside of the cluster manager, you must still initialize the **volume_list** entry as **volume_list = []**.

4. Rebuild the **initramfs** boot image to guarantee that the boot image will not try to activate a volume group controlled by the cluster. Update the **initramfs** device with the following command. This command may take up to a minute to complete.

```
# dracut -H -f /boot/initramfs-$(uname -r).img $(uname -r)
```

5. Reboot the node.

NOTE

If you have installed a new Linux kernel since booting the node on which you created the boot image, the new **initrd** image will be for the kernel that was running when you created it and not for the new kernel that is running when you reboot the node. You can ensure that the correct **initrd** device is in use by running the **uname -r** command before and after the reboot to determine the kernel release that is running. If the releases are not the same, update the **initrd** file after rebooting with the new kernel and then reboot the node.

6. When the node has rebooted, check whether the cluster services have started up again on that node by executing the **pcs cluster status** command on that node. If this yields the message **Error: cluster is not currently running on this node** then enter the following command.

```
# pcs cluster start
```

Alternately, you can wait until you have rebooted each node in the cluster and start cluster services on each of the nodes with the following command.

```
# pcs cluster start --all
```

2.4. CREATING THE RESOURCES AND RESOURCE GROUPS WITH THE PCS COMMAND

This use case requires that you create four cluster resources. To ensure these resources all run on the same node, they are configured as part of the resource group **apachegroup**. The resources to create are as follows, listed in the order in which they will start.

1. An **LVM** resource named **my_lvm** that uses the LVM volume group you created in Section 2.1, "Configuring an LVM Volume with an ext4 File System".
2. A **Filesystem** resource named **my_fs**, that uses the file system device **/dev/my_vg/my_lv** you created in Section 2.1, "Configuring an LVM Volume with an ext4 File System" .
3. An **IPaddr2** resource, which is a floating IP address for the **apachegroup** resource group. The IP address must not be one already associated with a physical node. If the **IPaddr2** resource's NIC device is not specified, the floating IP must reside on the same network as the statically assigned IP addresses used by the cluster nodes, otherwise the NIC device to assign the floating IP address cannot be properly detected.
4. An **apache** resource named **Website** that uses the **index.html** file and the Apache configuration you defined in Section 2.2, "Web Server Configuration".

The following procedure creates the resource group **apachegroup** and the resources that the group contains. The resources will start in the order in which you add them to the group, and they will stop in

the reverse order in which they are added to the group. Run this procedure from one node of the cluster only.

1. The following command creates the LVM resource **my_lvm**. This command specifies the **exclusive=true** parameter to ensure that only the cluster is capable of activating the LVM logical volume. Because the resource group **apachegroup** does not yet exist, this command creates the resource group.

```
[root@z1 ~]# pcs resource create my_lvm LVM volgrpname=my_vg \
exclusive=true --group apachegroup
```

   When you create a resource, the resource is started automatically. You can use the following command to confirm that the resource was created and has started.

```
# pcs resource show
 Resource Group: apachegroup
     my_lvm (ocf::heartbeat:LVM): Started
```

   You can manually stop and start an individual resource with the **pcs resource disable** and **pcs resource enable** commands.

2. The following commands create the remaining resources for the configuration, adding them to the existing resource group **apachegroup**.

```
[root@z1 ~]# pcs resource create my_fs Filesystem \
device="/dev/my_vg/my_lv" directory="/var/www" fstype="ext4" --group \
apachegroup

[root@z1 ~]# pcs resource create VirtualIP IPaddr2 ip=198.51.100.3 \
cidr_netmask=24 --group apachegroup

[root@z1 ~]# pcs resource create Website apache \
configfile="/etc/httpd/conf/httpd.conf" \
statusurl="http://127.0.0.1/server-status" --group apachegroup
```

3. After creating the resources and the resource group that contains them, you can check the status of the cluster. Note that all four resources are running on the same node.

```
[root@z1 ~]# pcs status
Cluster name: my_cluster
Last updated: Wed Jul 31 16:38:51 2013
Last change: Wed Jul 31 16:42:14 2013 via crm_attribute on z1.example.com
Stack: corosync
Current DC: z2.example.com (2) - partition with quorum
Version: 1.1.10-5.el7-9abe687
2 Nodes configured
6 Resources configured


Online: [ z1.example.com z2.example.com ]

Full list of resources:
 myapc (stonith:fence_apc_snmp): Started z1.example.com
 Resource Group: apachegroup
     my_lvm (ocf::heartbeat:LVM): Started z1.example.com
```

```
    my_fs (ocf::heartbeat:Filesystem): Started z1.example.com
    VirtualIP (ocf::heartbeat:IPaddr2): Started z1.example.com
    Website (ocf::heartbeat:apache): Started z1.example.com
```

Note that if you have not configured a fencing device for your cluster, as described in Section 1.3, "Fencing Configuration", by default the resources do not start.

4. Once the cluster is up and running, you can point a browser to the IP address you defined as the **IPaddr2** resource to view the sample display, consisting of the simple word "Hello".

```
Hello
```

If you find that the resources you configured are not running, you can run the **pcs resource debug-start *resource*** command to test the resource configuration. For information on the **pcs resource debug-start** command, see the *High Availability Add-On Reference* manual.

2.5. TESTING THE RESOURCE CONFIGURATION

In the cluster status display shown in Section 2.4, "Creating the Resources and Resource Groups with the pcs Command", all of the resources are running on node **z1.example.com**. You can test whether the resource group fails over to node **z2.example.com** by using the following procedure to put the first node in **standby** mode, after which the node will no longer be able to host resources.

1. The following command puts node **z1.example.com** in **standby** mode.

```
root@z1 ~]# pcs node standby z1.example.com
```

2. After putting node **z1** in standby mode, check the cluster status. Note that the resources should now all be running on **z2**.

```
[root@z1 ~]# pcs status
Cluster name: my_cluster
Last updated: Wed Jul 31 17:16:17 2013
Last change: Wed Jul 31 17:18:34 2013 via crm_attribute on z1.example.com
Stack: corosync
Current DC: z2.example.com (2) - partition with quorum
Version: 1.1.10-5.el7-9abe687
2 Nodes configured
6 Resources configured


Node z1.example.com (1): standby
Online: [ z2.example.com ]

Full list of resources:

 myapc (stonith:fence_apc_snmp): Started z1.example.com
 Resource Group: apachegroup
     my_lvm (ocf::heartbeat:LVM): Started z2.example.com
     my_fs (ocf::heartbeat:Filesystem): Started z2.example.com
     VirtualIP (ocf::heartbeat:IPaddr2): Started z2.example.com
     Website (ocf::heartbeat:apache): Started z2.example.com
```

The web site at the defined IP address should still display, without interruption.

3. To remove **z1** from **standby** mode, enter the following command.

```
root@z1 ~]# pcs node unstandby z1.example.com
```

NOTE

Removing a node from **standby** mode does not in itself cause the resources to fail back over to that node. This will depend on the **resource-stickiness** value for the resources. For information on the **resource-stickiness** meta attribute, see Configuring a Resource to Prefer its Current Node in the *Red Hat High Availability Add-On Reference*.

CHAPTER 3. AN ACTIVE/PASSIVE NFS SERVER IN A RED HAT HIGH AVAILABILITY CLUSTER

This chapter describes how to configure a highly available active/passive NFS server on a two-node Red Hat Enterprise Linux High Availability Add-On cluster using shared storage. The procedure uses **pcs** to configure Pacemaker cluster resources. In this use case, clients access the NFS file system through a floating IP address. The NFS server runs on one of two nodes in the cluster. If the node on which the NFS server is running becomes inoperative, the NFS server starts up again on the second node of the cluster with minimal service interruption.

This use case requires that your system include the following components:

- Two nodes, which will be used to create the cluster running the Apache HTTP server. In this example, the nodes used are **z1.example.com** and **z2.example.com**.
- A power fencing device for each node of the cluster. This example uses two ports of the APC power switch with a host name of **zapc.example.com**.
- A public virtual IP address, required for the NFS server.
- Shared storage for the nodes in the cluster, using iSCSI, Fibre Channel, or other shared network block device.

Configuring a highly available active/passive NFS server on a two-node Red Hat Enterprise Linux High requires that you perform the following steps.

1. Create the cluster that will run the NFS server and configure fencing for each node in the cluster, as described in Section 3.1, “Creating the NFS Cluster” .
2. Configure an **ext4** file system mounted on the LVM logical volume **my_lv** on the shared storage for the nodes in the cluster, as described in Section 3.2, “Configuring an LVM Volume with an ext4 File System”.
3. Configure an NFS share on the shared storage on the LVM logical volume, as described in Section 3.3, “NFS Share Setup” .
4. Ensure that only the cluster is capable of activating the LVM volume group that contains the logical volume **my_lv**, and that the volume group will not be activated outside of the cluster on startup, as described in Section 3.4, “Exclusive Activation of a Volume Group in a Cluster” .
5. Create the cluster resources as described in Section 3.5, “Configuring the Cluster Resources” .
6. Test the NFS server you have configured, as described in Section 3.6, “Testing the Resource Configuration”.

3.1. CREATING THE NFS CLUSTER

Use the following procedure to install and create the NFS cluster.

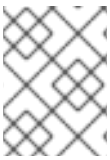
1. Install the cluster software on nodes **z1.example.com** and **z2.example.com**, using the procedure provided in Section 1.1, “Cluster Software Installation” .
2. Create the two-node cluster that consists of **z1.example.com** and **z2.example.com**, using the procedure provided in Section 1.2, “Cluster Creation”. As in that example procedure, this use case names the cluster **my_cluster**.

3. Configure fencing devices for each node of the cluster, using the procedure provided in Section 1.3, "Fencing Configuration". This example configures fencing using two ports of the APC power switch with a host name of **zapc.example.com**.

3.2. CONFIGURING AN LVM VOLUME WITH AN EXT4 FILE SYSTEM

This use case requires that you create an LVM logical volume on storage that is shared between the nodes of the cluster.

The following procedure creates an LVM logical volume and then creates an **ext4** file system on that volume. In this example, the shared partition **/dev/sdb1** is used to store the LVM physical volume from which the LVM logical volume will be created.



NOTE

LVM volumes and the corresponding partitions and devices used by cluster nodes must be connected to the cluster nodes only.

Since the **/dev/sdb1** partition is storage that is shared, you perform this procedure on one node only,

1. Create an LVM physical volume on partition **/dev/sdb1**.

```
[root@z1 ~]# pvcreate /dev/sdb1
  Physical volume "/dev/sdb1" successfully created
```

2. Create the volume group **my_vg** that consists of the physical volume **/dev/sdb1**.

```
[root@z1 ~]# vgcreate my_vg /dev/sdb1
  Volume group "my_vg" successfully created
```

3. Create a logical volume using the volume group **my_vg**.

```
[root@z1 ~]# lvcreate -L450 -n my_lv my_vg
  Rounding up size to full physical extent 452.00 MiB
  Logical volume "my_lv" created
```

You can use the **lvs** command to display the logical volume.

```
[root@z1 ~]# lvs
  LV      VG      Attr      LSize   Pool Origin Data%  Move Log Copy%  Convert
  my_lv   my_vg   -wi-a---- 452.00m
  ...
```

4. Create an **ext4** file system on the logical volume **my_lv**.

```
[root@z1 ~]# mkfs.ext4 /dev/my_vg/my_lv
mke2fs 1.42.7 (21-Jan-2013)
Filesystem label=
OS type: Linux
...
```

3.3. NFS SHARE SETUP

The following procedure configures the NFS share for the NFS daemon failover. You need to perform this procedure on only one node in the cluster.

1. Create the **/nfsshare** directory.

```
[root@z1 ~]# mkdir /nfsshare
```

2. Mount the **ext4** file system that you created in Section 3.2, "Configuring an LVM Volume with an ext4 File System" on the **/nfsshare** directory.

```
[root@z1 ~]# mount /dev/my_vg/my_lv /nfsshare
```

3. Create an **exports** directory tree on the **/nfsshare** directory.

```
[root@z1 ~]# mkdir -p /nfsshare/exports
[root@z1 ~]# mkdir -p /nfsshare/exports/export1
[root@z1 ~]# mkdir -p /nfsshare/exports/export2
```

4. Place files in the **exports** directory for the NFS clients to access. For this example, we are creating test files named **clientdatafile1** and **clientdatafile2**.

```
[root@z1 ~]# touch /nfsshare/exports/export1/clientdatafile1
[root@z1 ~]# touch /nfsshare/exports/export2/clientdatafile2
```

5. Unmount the ext4 file system and deactivate the LVM volume group.

```
[root@z1 ~]# umount /dev/my_vg/my_lv
[root@z1 ~]# vgchange -an my_vg
```

3.4. EXCLUSIVE ACTIVATION OF A VOLUME GROUP IN A CLUSTER

The following procedure configures the LVM volume group in a way that will ensure that only the cluster is capable of activating the volume group, and that the volume group will not be activated outside of the cluster on startup. If the volume group is activated by a system outside of the cluster, there is a risk of corrupting the volume group's metadata.

This procedure modifies the **volume_list** entry in the **/etc/lvm/lvm.conf** configuration file. Volume groups listed in the **volume_list** entry are allowed to automatically activate on the local node outside of the cluster manager's control. Volume groups related to the node's local root and home directories should be included in this list. All volume groups managed by the cluster manager must be excluded from the **volume_list** entry. Note that this procedure does not require the use of **clvmd**.

Perform the following procedure on each node in the cluster.

1. Execute the following command to ensure that **locking_type** is set to 1 and that **use_lvmetad** is set to 0 in the **/etc/lvm/lvm.conf** file. This command also disables and stops any **lvmetad** processes immediately.

```
# lvmconf --enable-halvm --services --startstopservices
```

2. Determine which volume groups are currently configured on your local storage with the following command. This will output a list of the currently-configured volume groups. If you have space allocated in separate volume groups for root and for your home directory on this node, you will see those volumes in the output, as in this example.

```
# vgs --noheadings -o vg_name
  my_vg
  rhel_home
  rhel_root
```

3. Add the volume groups other than **my_vg** (the volume group you have just defined for the cluster) as entries to **volume_list** in the **/etc/lvm/lvm.conf** configuration file. For example, if you have space allocated in separate volume groups for root and for your home directory, you would uncomment the **volume_list** line of the **lvm.conf** file and add these volume groups as entries to **volume_list** as follows:

```
volume_list = [ "rhel_root", "rhel_home" ]
```

NOTE

If no local volume groups are present on a node to be activated outside of the cluster manager, you must still initialize the **volume_list** entry as **volume_list = []**.

4. Rebuild the **initramfs** boot image to guarantee that the boot image will not try to activate a volume group controlled by the cluster. Update the **initramfs** device with the following command. This command may take up to a minute to complete.

```
# dracut -H -f /boot/initramfs-$(uname -r).img $(uname -r)
```

5. Reboot the node.

NOTE

If you have installed a new Linux kernel since booting the node on which you created the boot image, the new **initrd** image will be for the kernel that was running when you created it and not for the new kernel that is running when you reboot the node. You can ensure that the correct **initrd** device is in use by running the **uname -r** command before and after the reboot to determine the kernel release that is running. If the releases are not the same, update the **initrd** file after rebooting with the new kernel and then reboot the node.

6. When the node has rebooted, check whether the cluster services have started up again on that node by executing the **pcs cluster status** command on that node. If this yields the message **Error: cluster is not currently running on this node** then enter the following command.

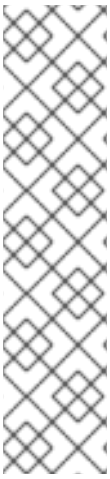
```
# pcs cluster start
```

Alternately, you can wait until you have rebooted each node in the cluster and start cluster services on all of the nodes in the cluster with the following command.

```
# pcs cluster start --all
```

3.5. CONFIGURING THE CLUSTER RESOURCES

This section provides the procedure for configuring the cluster resources for this use case.



NOTE

It is recommended that when you create a cluster resource with the **pcs resource create**, you execute the **pcs status** command immediately afterwards to verify that the resource is running. Note that if you have not configured a fencing device for your cluster, as described in Section 1.3, "Fencing Configuration", by default the resources do not start.

If you find that the resources you configured are not running, you can run the **pcs resource debug-start *resource*** command to test the resource configuration. This starts the service outside of the cluster's control and knowledge. At the point the configured resources are running again, run **pcs resource cleanup *resource*** to make the cluster aware of the updates. For information on the **pcs resource debug-start** command, see the *High Availability Add-On Reference* manual.

The following procedure configures the system resources. To ensure these resources all run on the same node, they are configured as part of the resource group **nfsgroup**. The resources will start in the order in which you add them to the group, and they will stop in the reverse order in which they are added to the group. Run this procedure from one node of the cluster only.

1. The following command creates the LVM resource named **my_lvm**. This command specifies the **exclusive=true** parameter to ensure that only the cluster is capable of activating the LVM logical volume. Because the resource group **nfsgroup** does not yet exist, this command creates the resource group.

```
[root@z1 ~]# pcs resource create my_lvm LVM volgrpname=my_vg \
exclusive=true --group nfsgroup
```

Check the status of the cluster to verify that the resource is running.

```
root@z1 ~]#  pcs status
Cluster name: my_cluster
Last updated: Thu Jan  8 11:13:17 2015
Last change: Thu Jan  8 11:13:08 2015
Stack: corosync
Current DC: z2.example.com (2) - partition with quorum
Version: 1.1.12-a14efad
2 Nodes configured
3 Resources configured

Online: [ z1.example.com z2.example.com ]

Full list of resources:
 myapc  (stonith:fence_apc_snmp):       Started z1.example.com
 Resource Group: nfsgroup
     my_lvm     (ocf::heartbeat:LVM):   Started z1.example.com

PCSD Status:
  z1.example.com: Online
  z2.example.com: Online
```

```
Daemon Status:
  corosync: active/enabled
  pacemaker: active/enabled
  pcsd: active/enabled
```

2. Configure a **Filesystem** resource for the cluster.

NOTE

You can specify mount options as part of the resource configuration for a **Filesystem** resource with the **options=*options*** parameter. Run the **pcs resource describe Filesystem** command for full configuration options.

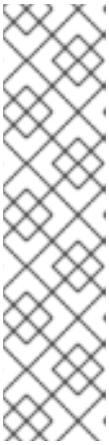
The following command configures an ext4 **Filesystem** resource named **nfsshare** as part of the **nfsgroup** resource group. This file system uses the LVM volume group and ext4 file system you created in Section 3.2, "Configuring an LVM Volume with an ext4 File System" and will be mounted on the /**nfsshare** directory you created in Section 3.3, "NFS Share Setup" .

```
[root@z1 ~]# pcs resource create nfsshare Filesystem \
device=/dev/my_vg/my_lv directory=/nfsshare \
fstype=ext4 --group nfsgroup
```

Verify that the **my_lvm** and **nfsshare** resources are running.

```
[root@z1 ~]# pcs status
...
Full list of resources:
 myapc  (stonith:fence_apc_snmp):       Started z1.example.com
 Resource Group: nfsgroup
     my_lvm     (ocf::heartbeat:LVM):   Started z1.example.com
     nfsshare   (ocf::heartbeat:Filesystem):    Started z1.example.com
...
```

3. Create the **nfsserver** resource named **nfs-daemon** part of the resource group **nfsgroup**.



NOTE

The **nfsserver** resource allows you to specify an **nfs_shared_infodir** parameter, which is a directory that NFS daemons will use to store NFS-related stateful information. It is recommended that this attribute be set to a subdirectory of one of the **Filesystem** resources you created in this collection of exports. This ensures that the NFS daemons are storing their stateful information on a device that will become available to another node if this resource group should need to relocate. In this example, /**nfsshare** is the shared-storage directory managed by the **Filesystem** resource, /**nfsshare/exports/export1** and /**nfsshare/exports/export2** are the export directories, and /**nfsshare/nfsinfo** is the shared-information directory for the **nfsserver** resource.

```
[root@z1 ~]# pcs resource create nfs-daemon nfsserver \
nfs_shared_infodir=/nfsshare/nfsinfo nfs_no_notify=true \
--group nfsgroup
[root@z1 ~]# pcs status
...
```

4. Add the **exportfs** resources to export the **/nfsshare/exports** directory. These resources are part of the resource group **nfsgroup**. This builds a virtual directory for NFSv4 clients. NFSv3 clients can access these exports as well.

```
[root@z1 ~]# pcs resource create nfs-root exportfs \
clientspec=192.168.122.0/255.255.255.0 \
options=rw,sync,no_root_squash \
directory=/nfsshare/exports \
fsid=0 --group nfsgroup

[root@z1 ~]# # pcs resource create nfs-export1 exportfs \
clientspec=192.168.122.0/255.255.255.0 \
options=rw,sync,no_root_squash directory=/nfsshare/exports/export1 \
fsid=1 --group nfsgroup

[root@z1 ~]# # pcs resource create nfs-export2 exportfs \
clientspec=192.168.122.0/255.255.255.0 \
options=rw,sync,no_root_squash directory=/nfsshare/exports/export2 \
fsid=2 --group nfsgroup
```

5. Add the floating IP address resource that NFS clients will use to access the NFS share. The floating IP address that you specify requires a reverse DNS lookup or it must be specified in the **/etc/hosts** on all nodes in the cluster. This resource is part of the resource group **nfsgroup**. For this example deployment, we are using 192.168.122.200 as the floating IP address.

```
[root@z1 ~]# pcs resource create nfs_ip IPaddr2 \
ip=192.168.122.200 cidr_netmask=24 --group nfsgroup
```

6. Add an **nfsnotify** resource for sending NFSv3 reboot notifications once the entire NFS deployment has initialized. This resource is part of the resource group **nfsgroup**.

NOTE

For the NFS notification to be processed correctly, the floating IP address must have a host name associated with it that is consistent on both the NFS servers and the NFS client.

```
[root@z1 ~]# pcs resource create nfs-notify nfsnotify \
source_host=192.168.122.200 --group nfsgroup
```

After creating the resources and the resource constraints, you can check the status of the cluster. Note that all resources are running on the same node.

```
[root@z1 ~]# pcs status
...
Full list of resources:
 myapc  (stonith:fence_apc_snmp):       Started z1.example.com
 Resource Group: nfsgroup
     my_lvm     (ocf::heartbeat:LVM):   Started z1.example.com
     nfsshare   (ocf::heartbeat:Filesystem):    Started z1.example.com
     nfs-daemon (ocf::heartbeat:nfsserver):     Started z1.example.com
     nfs-root   (ocf::heartbeat:exportfs):      Started z1.example.com
```

```
    nfs-export1        (ocf::heartbeat:exportfs):      Started z1.example.com
    nfs-export2        (ocf::heartbeat:exportfs):      Started z1.example.com
    nfs_ip     (ocf::heartbeat:IPaddr2):       Started  z1.example.com
    nfs-notify (ocf::heartbeat:nfsnotify):     Started z1.example.com
...
```

3.6. TESTING THE RESOURCE CONFIGURATION

You can validate your system configuration with the following procedure. You should be able to mount the exported file system with either NFSv3 or NFSv4.

1. On a node outside of the cluster, residing in the same network as the deployment, verify that the NFS share can be seen by mounting the NFS share. For this example, we are using the 192.168.122.0/24 network.

```
# showmount -e 192.168.122.200
Export list for 192.168.122.200:
/nfsshare/exports/export1 192.168.122.0/255.255.255.0
/nfsshare/exports         192.168.122.0/255.255.255.0
/nfsshare/exports/export2 192.168.122.0/255.255.255.0
```

2. To verify that you can mount the NFS share with NFSv4, mount the NFS share to a directory on the client node. After mounting, verify that the contents of the export directories are visible. Unmount the share after testing.

```
# mkdir nfsshare
# mount -o "vers=4" 192.168.122.200:export1 nfsshare
# ls nfsshare
clientdatafile1
# umount nfsshare
```

3. Verify that you can mount the NFS share with NFSv3. After mounting, verify that the test file **clientdatafile1** is visible. Unlike NFSv4, since NFSV3 does not use the virtual file system, you must mount a specific export. Unmount the share after testing.

```
# mkdir nfsshare
# mount -o "vers=3" 192.168.122.200:/nfsshare/exports/export2 nfsshare
# ls nfsshare
   clientdatafile2
# umount nfsshare
```

4. To test for failover, perform the following steps.

    1. On a node outside of the cluster, mount the NFS share and verify access to the **clientdatafile1** we created in Section 3.3, “NFS Share Setup” .

```
# mkdir nfsshare
# mount -o "vers=4" 192.168.122.200:export1 nfsshare
# ls nfsshare
clientdatafile1
```

    2. From a node within the cluster, determine which node in the cluster is running **nfsgroup**. In this example, **nfsgroup** is running on **z1.example.com**.

```
[root@z1 ~]# pcs status
...
Full list of resources:
 myapc  (stonith:fence_apc_snmp):       Started z1.example.com
 Resource Group: nfsgroup
     my_lvm     (ocf::heartbeat:LVM):   Started z1.example.com
     nfsshare   (ocf::heartbeat:Filesystem):    Started z1.example.com
     nfs-daemon (ocf::heartbeat:nfsserver):     Started z1.example.com
     nfs-root   (ocf::heartbeat:exportfs):      Started z1.example.com
     nfs-export1        (ocf::heartbeat:exportfs):      Started z1.example.com
     nfs-export2        (ocf::heartbeat:exportfs):      Started z1.example.com
     nfs_ip     (ocf::heartbeat:IPaddr2):       Started  z1.example.com
     nfs-notify (ocf::heartbeat:nfsnotify):     Started z1.example.com
...
```

3. From a node within the cluster, put the node that is running **nfsgroup** in standby mode.

```
[root@z1 ~]# pcs node standby z1.example.com
```

4. Verify that **nfsgroup** successfully starts on the other cluster node.

```
[root@z1 ~]# pcs status
...
Full list of resources:
 Resource Group: nfsgroup
     my_lvm     (ocf::heartbeat:LVM):   Started z2.example.com
     nfsshare   (ocf::heartbeat:Filesystem):    Started z2.example.com
     nfs-daemon (ocf::heartbeat:nfsserver):     Started z2.example.com
     nfs-root   (ocf::heartbeat:exportfs):      Started z2.example.com
     nfs-export1        (ocf::heartbeat:exportfs):      Started z2.example.com
     nfs-export2        (ocf::heartbeat:exportfs):      Started z2.example.com
     nfs_ip     (ocf::heartbeat:IPaddr2):       Started  z2.example.com
     nfs-notify (ocf::heartbeat:nfsnotify):     Started z2.example.com
...
```

5. From the node outside the cluster on which you have mounted the NFS share, verify that this outside node still continues to have access to the test file within the NFS mount.

```
# ls nfsshare
clientdatafile1
```

Service will be lost briefly for the client during the failover briefly but the client should recover in with no user intervention. By default, clients using NFSv4 may take up to 90 seconds to recover the mount; this 90 seconds represents the NFSv4 file lease grace period observed by the server on startup. NFSv3 clients should recover access to the mount in a matter of a few seconds.

6. From a node within the cluster, remove the node that was initially running running **nfsgroup** from standby mode. This will not in itself move the cluster resources back to this node.

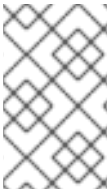
```
[root@z1 ~]# pcs node unstandby z1.example.com
```

NOTE

Removing a node from **standby** mode does not in itself cause the resources to fail back over to that node. This will depend on the **resource-stickiness** value for the resources. For information on the **resource-stickiness** meta attribute, see Configuring a Resource to Prefer its Current Node in the *Red Hat High Availability Add-On Reference*.

CHAPTER 4. AN ACTIVE/ACTIVE SAMBA SERVER IN A RED HAT HIGH AVAILABILITY CLUSTER (RED HAT ENTERPRISE LINUX 7.4 AND LATER)

As of the Red Hat Enterprise Linux 7.4 release, the Red Hat Resilient Storage Add-On provides support for running Samba in an active/active cluster configuration using Pacemaker. The Red Hat Resilient Storage Add-On includes the High Availability Add-On.



NOTE

For further information on support policies for Samba, see Support Policies for RHEL Resilient Storage - ctdb General Policies and Support Policies for RHEL Resilient Storage - Exporting gfs2 contents via other protocols on the Red Hat Customer Portal.

This chapter describes how to configure a highly available active/active Samba server on a two-node Red Hat Enterprise Linux High Availability Add-On cluster using shared storage. The procedure uses **pcs** to configure Pacemaker cluster resources.

This use case requires that your system include the following components:

- Two nodes, which will be used to create the cluster running Clustered Samba. In this example, the nodes used are **z1.example.com** and **z2.example.com** which have IP address of **192.168.1.151** and **192.168.1.152**.
- A power fencing device for each node of the cluster. This example uses two ports of the APC power switch with a host name of **zapc.example.com**.
- Shared storage for the nodes in the cluster, using iSCSI or Fibre Channel.

Configuring a highly available active/active Samba server on a two-node Red Hat Enterprise Linux High Availability Add-On cluster requires that you perform the following steps.

1. Create the cluster that will export the Samba shares and configure fencing for each node in the cluster, as described in Section 4.1, "Creating the Cluster".
2. Configure a **gfs2** file system mounted on the clustered LVM logical volume **my_clv** on the shared storage for the nodes in the cluster, as described in Section 4.2, "Configuring a Clustered LVM Volume with a GFS2 File System".
3. Configure Samba on each node in the cluster, Section 4.3, "Configuring Samba".
4. Create the Samba cluster resources as described in Section 4.4, "Configuring the Samba Cluster Resources".
5. Test the Samba share you have configured, as described in Section 4.5, "Testing the Resource Configuration".

4.1. CREATING THE CLUSTER

Use the following procedure to install and create the cluster to use for the Samba service:

1. Install the cluster software on nodes **z1.example.com** and **z2.example.com**, using the procedure provided in Section 1.1, "Cluster Software Installation" .

2. Create the two-node cluster that consists of **z1.example.com** and **z2.example.com**, using the procedure provided in Section 1.2, "Cluster Creation". As in that example procedure, this use case names the cluster **my_cluster**.

3. Configure fencing devices for each node of the cluster, using the procedure provided in Section 1.3, "Fencing Configuration". This example configures fencing using two ports of the APC power switch with a host name of **zapc.example.com**.

4.2. CONFIGURING A CLUSTERED LVM VOLUME WITH A GFS2 FILE SYSTEM

This use case requires that you create a clustered LVM logical volume on storage that is shared between the nodes of the cluster.

This section describes how to create a clustered LVM logical volume with a GFS2 file system on that volume. In this example, the shared partition /**dev**/**vdb** is used to store the LVM physical volume from which the LVM logical volume will be created.

NOTE

LVM volumes and the corresponding partitions and devices used by cluster nodes must be connected to the cluster nodes only.

Before starting this procedure, install the **lvm2-cluster** and **gfs2-utils** packages, which are part of the Resilient Storage channel, on both nodes of the cluster.

```
# yum install lvm2-cluster gfs2-utils
```

Since the /**dev**/**vdb** partition is storage that is shared, you perform this procedure on one node only,

1. Set the global Pacemaker parameter **no_quorum_policy** to **freeze**. This prevents the entire cluster from being fenced every time quorum is lost. For further information on setting this policy, see *Global File System 2*.

   ```
   [root@z1 ~]# pcs property set no-quorum-policy=freeze
   ```

2. Set up a **dlm** resource. This is a required dependency for the **clvmd** service and the GFS2 file system.

   ```
   [root@z1 ~]# pcs resource create dlm ocf:pacemaker:controld op monitor interval=30s on-fail=fence clone interleave=true ordered=true
   ```

3. Set up **clvmd** as a cluster resource.

   ```
   [root@z1 ~]# pcs resource create clvmd ocf:heartbeat:clvm op monitor interval=30s on-fail=fence clone interleave=true ordered=true
   ```

   Note that the **ocf:heartbeat:clvm** resource agent, as part of the start procedure, sets the **locking_type** parameter in the **/etc/lvm/lvm.conf** file to **3** and disables the **lvmetad** daemon.

4. Set up **clvmd** and **dlm** dependency and start up order. The **clvmd** resource must start after the **dlm** resource and must run on the same node as the **dlm** resource.

```
[root@z1 ~]# pcs constraint order start dlm-clone then clvmd-clone
Adding dlm-clone clvmd-clone (kind: Mandatory) (Options: first-action=start then-action=start)
[root@z1 ~]# pcs constraint colocation add clvmd-clone with dlm-clone
```

5. Verify that the **dlm** and **clvmd** resources are running on all nodes.

```
[root@z1 ~]# pcs status
...
Full list of resources:
...
 Clone Set: dlm-clone [dlm]
     Started: [ z1 z2 ]
 Clone Set: clvmd-clone [clvmd]
     Started: [ z1 z2 ]
```

6. Create the clustered logical volume

```
[root@z1 ~]# pvcreate /dev/vdb
[root@z1 ~]# vgcreate -Ay -cy cluster_vg /dev/vdb
[root@z1 ~]# lvcreate -L4G -n cluster_lv cluster_vg
```

7. Optionally, to verify that the volume was created successfully you can use the **lvs** command to display the logical volume.

```
[root@z1 ~]# lvs
  LV         VG         Attr       LSize ...
  cluster_lv cluster_vg -wi-ao---- 4.00g
  ...
```

8. Format the volume with a GFS2 file system. In this example, **my_cluster** is the cluster name. This example specifies **-j 2** to indicate two journals because the number of journals you configure must equal the number of nodes in the cluster.

```
[root@z1 ~]# mkfs.gfs2 -p lock_dlm -j 2 -t my_cluster:samba /dev/cluster_vg/cluster_lv
```

9. Create a **Filesystem** resource, which configures Pacemaker to mount and manage the file system. This example creates a **Filesystem** resource named **fs**, and creates the **/mnt/gfs2share** on both nodes of the cluster.

```
[root@z1 ~]# pcs resource create fs ocf:heartbeat:Filesystem
device="/dev/cluster_vg/cluster_lv" directory="/mnt/gfs2share" fstype="gfs2" --clone
```

10. Configure a dependency and a startup order for the GFS2 file system and the **clvmd** service. GFS2 must start after **clvmd** and must run on the same node as **clvmd**.

```
[root@z1 ~]# pcs constraint order start clvmd-clone then fs-clone
Adding clvmd-clone fs-clone (kind: Mandatory) (Options: first-action=start then-action=start)
[root@z1 ~]# pcs constraint colocation add fs-clone with clvmd-clone
```

11. Verify that the GFS2 file system is mounted as expected.

```
[root@z1 ~]# mount |grep /mnt/gfs2share
/dev/mapper/cluster_vg-cluster_lv on /mnt/gfs2share type gfs2 (rw,noatime,seclabel)
```

4.3. CONFIGURING SAMBA

The following procedure initializes the Samba environment and configures Samba on the cluster nodes.

1. On both nodes of the cluster, perform the following steps:

    1. Install the **samba**, **ctdb**, and **cifs-utils** packages.

    ```
    # yum install samba ctdb cifs-utils
    ```

    2. If you are running the **firewalld** daemon, run the following commands to enable the ports that are required by the **ctdb** and **samba** services.

    ```
    # firewall-cmd --add-service=ctdb --permanent
    # firewall-cmd --add-service=samba --permanent
    # firewall-cmd --reload
    ```

    3. Enter the following commands to ensure that these daemons are not running and do not start at bootup. Note that not all of these daemons may be present or running on your system.

    ```
    # systemctl disable ctdb
    # systemctl disable smb
    # systemctl disable nmb
    # systemctl disable winbind
    # systemctl stop ctdb
    # systemctl stop smb
    # systemctl stop nmb
    # systemctl stop winbind
    ```

    4. In the **/etc/samba/smb.conf** file, configure the Samba server and set up the **[public]** share definition. For example:

    ```
    # cat << END > /etc/samba/smb.conf
    [global]
    netbios name = linuxserver
    workgroup = WORKGROUP
    server string = Public File Server
    security = user
    map to guest = bad user
    guest account = smbguest
    clustering = yes
    ctdbd socket = /tmp/ctdb.socket
    [public]
    path = /mnt/gfs2share/public
    guest ok = yes
    read only = no
    END
    ```

For information on configuring Samba as a standalone server, as in this example, as well as information on verifying the **smb.conf** file with the **testparm** utility, see the File and Print Servers section of the *System Administrator's Guide*

5. Add the IP address of the cluster nodes to the **/etc/ctdb/nodes** file.

```
# cat << END > /etc/ctdb/nodes
192.168.1.151
192.168.1.152
END
```

6. For load balancing between the nodes of the cluster, you can add two or more IP addresses that can be used to access the Samba shares exported by this cluster to the **/etc/ctdb/public_addresses** file. These are the IP addresses that you should configure in DNS for the name of the Samba server and are the addresses that SMB clients will connect to. Configure the name of the Samba server as one DNS type A record with multiple IP addresses and let round-robin DNS distribute the clients across the nodes of the cluster.

   For this example, the DNS entry **linuxserver.example.com** has been defined with both the addresses listed under the **/etc/ctdb/public_addresses** file. With this in place, DNS will distribute the Samba clients across the cluster nodes in a round-robin fashion. Please note that when implementing this scenario, the DNS entries should match your needs.

   Add the IP addresses that can be used to access the Samba shares exported by this cluster to the **/etc/ctdb/public_addresses** file.

```
# cat << END > /etc/ctdb/public_addresses
192.168.1.201/24 eth0
192.168.1.202/24 eth0
END
```

7. Create a Samba group, then add a local user for the public test share directory, setting the previously created group as the primary group.

```
# groupadd smbguest
# adduser smbguest -g smbguest
```

8. Make sure that the SELinux context are correct in the CTDB-related directories.

```
# mkdir /var/ctdb/
# chcon -Rv -u system_u -r object_r -t ctdbd_var_lib_t /var/ctdb/
changing security context of '/var/ctdb/'
# chcon -Rv -u system_u -r object_r -t ctdbd_var_lib_t /var/lib/ctdb/
changing security context of '/var/lib/ctdb/'
```

2. On one node of the cluster, perform the following steps:

   1. Set up the directories for the CTDB lock file and public share.

```
[root@z1 ~]# mkdir -p /mnt/gfs2share/ctdb/
[root@z1 ~]# mkdir -p /mnt/gfs2share/public/
```

   2. Update the SELinux contexts on the GFS2 share.

```
[root@z1 ~]# chown smbguest:smbguest /mnt/gfs2share/public/
[root@z1 ~]# chmod 755 /mnt/gfs2share/public/
[root@z1 ~]# chcon -Rv -t ctdbd_var_run_t /mnt/gfs2share/ctdb/
changing security context of ‘/mnt/gfs2share/ctdb/’
[root@z1 ~]# chcon -Rv -u system_u -r object_r -t samba_share_t /mnt/gfs2share/public/
changing security context of ‘/mnt/gfs2share/public’
```

4.4. CONFIGURING THE SAMBA CLUSTER RESOURCES

This section provides the procedure for configuring the Samba cluster resources for this use case.

The following procedure creates a snapshot of the cluster's **cib** file named **samba.cib** and adds the resources to that test file rather then configuring them directly on the running cluster. After the resources and constraints are configured, the procedure pushes the contents of **samba.cib** to the running cluster configuration file.

On one node of the cluster, run the following procedure.

1. Create a snapshot of the **cib** file, which is the cluster configuration file.

```
[root@z1 ~]# pcs cluster cib samba.cib
```

2. Create the CTDB resource to be used by Samba. Create this resource as a cloned resource so that it will run on both cluster nodes.

```
[root@z1 ~]# pcs -f samba.cib resource create ctdb ocf:heartbeat:CTDB \
ctdb_recovery_lock="/mnt/gfs2share/ctdb/ctdb.lock" \
ctdb_dbdir=/var/ctdb ctdb_socket=/tmp/ctdb.socket \
ctdb_logfile=/var/log/ctdb.log \
op monitor interval=10 timeout=30 op start timeout=90 \
op stop timeout=100 --clone
```

3. Create the cloned Samba server.

```
[root@z1 ~]# pcs -f samba.cib resource create samba systemd:smb --clone
```

4. Create the colocation and order constraints for the cluster resources. The startup order is Filesystem resource, CTDB resource, then Samba resource.

```
[root@z1 ~]# pcs -f samba.cib constraint order fs-clone then ctdb-clone
Adding fs-clone ctdb-clone (kind: Mandatory) (Options: first-action=start then-action=start)
[root@z1 ~]# pcs -f samba.cib constraint order ctdb-clone then samba-clone
Adding ctdb-clone samba-clone (kind: Mandatory) (Options: first-action=start then-
action=start)
[root@z1 ~]# pcs -f samba.cib constraint colocation add ctdb-clone with fs-clone
[root@z1 ~]# pcs -f samba.cib constraint colocation add samba-clone with ctdb-clone
```

5. Push the content of the **cib** snapshot to the cluster.

```
[root@z1 ~]# pcs cluster cib-push samba.cib
CIB updated
```

6. Check the status of the cluster to verify that the resource is running.

Note that in Red Hat Enterprise Linux 7.4 it can take a couple of minutes for CTDB to start Samba, export the shares, and stabilize. If you check the cluster status before this process has completed, you may see a message that the CTDB status call failed. Once this process has completed, you can clear this message from the display by running the **pcs resource cleanup ctdb-clone** command.

```
[root@z1 ~]# pcs status
Cluster name: my_cluster
Stack: corosync
Current DC: z1.example.com (version 1.1.16-12.el7_4.2-94ff4df) - partition with quorum
Last updated: Thu Oct 19 18:17:07 2017
Last change: Thu Oct 19 18:16:50 2017 by hacluster via crmd on z1.example.com

2 nodes configured
11 resources configured

Online: [ z1.example.com z2.example.com ]

Full list of resources:

 myapc  (stonith:fence_apc_snmp):       Started z1.example.com
 Clone Set: dlm-clone [dlm]
     Started: [ z1.example.com z2.example.com ]
 Clone Set: clvmd-clone [clvmd]
     Started: [ z1.example.com z2.example.com ]
 Clone Set: fs-clone [fs]
     Started: [ z1.example.com z2.example.com ]
 Clone Set: ctdb-clone [ctdb]
     Started: [ z1.example.com z2.example.com ]
 Clone Set: samba-clone [samba]
     Started: [ z1.example.com z2.example.com ]
```

NOTE

If you find that the resources you configured are not running, you can run the **pcs resource debug-start *resource*** command to test the resource configuration. This starts the service outside of the cluster's control and knowledge. If the configured resources are running again, run **pcs resource cleanup *resource*** to make the cluster aware of the updates. For information on the **pcs resource debug-start** command, see the Enabling, Disabling, and Banning Cluster Resources section in the *High Availability Add-On Reference* manual.

4.5. TESTING THE RESOURCE CONFIGURATION

If the Samba configuration was successful, you should be able to mount the Samba share on a node in the cluster. The following example procedure mounts a Samba share.

1. Add an existing user in the cluster node to the **smbpasswd** file and assign a password. In the following example, there is an existing user **smbuser**.

```
[root@z1 ~]# smbpasswd -a smbuser
New SMB password:
Retype new SMB password:
Added user smbuser
```

2. Mount the Samba share:

```
[root@z1 ~]# mkdir /mnt/sambashare
[root@z1 ~]# mount -t cifs -o user=smbuser //198.162.1.151/public /mnt/sambashare
Password for smbuser@//198.162.1.151/public:  ********
```

3. Check whether the file system is mounted:

```
[root@z1 ~]# mount | grep /mnt/sambashare
//198.162.1.151/public on /mnt/sambashare type cifs
(rw,relatime,vers=1.0,cache=strict,username=smbuser,domain=LINUXSERVER,uid=0,noforceu
id,gid=0,noforcegid,addr=10.37.167.205,unix,posixpaths,serverino,mapposix,acl,rsize=1048576
wsize=65536,echo_interval=60,actimeo=1)
```

To check for Samba recovery, perform the following procedure.

1. Manually stop the CTDB resource with the following command:

```
[root@z1 ~]# pcs resource debug-stop ctdb
```

2. After you stop the resource, the system should recover the service. Check the cluster status with the **pcs status** command. You should see that the **ctdb-clone** resource has started, but you will also see a **ctdb_monitor** failure.

```
[root@z1 ~]# pcs status
...
Clone Set: ctdb-clone [ctdb]
     Started: [ z1.example.com z2.example.com ]
...
Failed Actions:
* ctdb_monitor_10000 on z1.example.com 'unknown error' (1): call=126, status=complete,
exitreason='CTDB status call failed: connect() failed, errno=111',
    last-rc-change='Thu Oct 19 18:39:51 2017', queued=0ms, exec=0ms
...
```

To clear this error from the status, enter the following command on one of the cluster nodes:

```
[root@z1 ~]# pcs resource cleanup ctdb-clone
```

APPENDIX A. REVISION HISTORY

Revision 6-1 **Wed Aug 7 2019** **Steven Levine**
Preparing document for 7.7 GA publication.

Revision 5-2 **Thu Oct 4 2018** **Steven Levine**
Preparing document for 7.6 GA publication.

Revision 4-2 **Wed Mar 14 2018** **Steven Levine**
Preparing document for 7.5 GA publication.

Revision 4-1 **Thu Dec 14 2017** **Steven Levine**
Preparing document for 7.5 Beta publication.

Revision 3-4 **Wed Aug 16 2017** **Steven Levine**
Updated version for 7.4.

Revision 3-3 **Wed Jul 19 2017** **Steven Levine**
Document version for 7.4 GA publication.

Revision 3-1 **Wed May 10 2017** **Steven Levine**
Preparing document for 7.4 Beta publication.

Revision 2-6 **Mon Apr 17 2017** **Steven Levine**
Update for 7.3

Revision 2-4 **Mon Oct 17 2016** **Steven Levine**
Version for 7.3 GA publication.

Revision 2-3 **Fri Aug 12 2016** **Steven Levine**
Preparing document for 7.3 Beta publication.

Revision 1.2-3 **Mon Nov 9 2015** **Steven Levine**
Preparing document for 7.2 GA publication.

Revision 1.2-2 **Tue Aug 18 2015** **Steven Levine**
Preparing document for 7.2 Beta publication.

Revision 1.1-19 **Mon Feb 16 2015** **Steven Levine**
Version for 7.1 GA release

Revision 1.1-10 **Thu Dec 11 2014** **Steven Levine**
Version for 7.1 Beta release

Revision 0.1-33 **Mon Jun 2 2014** **Steven Levine**
Version for 7.0 GA release