



Red Hat Enterprise Linux 7 High Availability Add-On Administration

Configuring and Managing the High Availability Add-On

Configuring and Managing the High Availability Add-On

Legal Notice

Copyright © 2015 Red Hat, Inc. and others.

This document is licensed by Red Hat under the [Creative Commons Attribution-ShareAlike 3.0 Unported License](#). If you distribute this document, or a modified version of it, you must provide attribution to Red Hat, Inc. and provide a link to the original. If the document is modified, all Red Hat trademarks must be removed.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, MetaMatrix, Fedora, the Infinity Logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux® is the registered trademark of Linus Torvalds in the United States and other countries.

Java® is a registered trademark of Oracle and/or its affiliates.

XFS® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js® is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack® Word Mark and OpenStack Logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

High Availability Add-On Administration describes the configuration and management of 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	2
1.1. Cluster Software Installation	2
1.2. Cluster Creation	3
1.3. Fencing Configuration	4
Chapter 2. An active/passive Apache Web Server in a Red Hat High Availability Cluster	6
2.1. Configuring an LVM Volume with an ext4 File System	7
2.2. Web Server Configuration	8
2.3. Exclusive Activation of a Volume Group in a Cluster	8
2.4. Creating the Resources and Resource Groups with the pcs Command	10
2.5. Testing the Resource Configuration	12
Chapter 3. An active/passive NFS Server in a Red Hat High Availability Cluster	14
3.1. Creating the NFS Cluster	14
3.2. Configuring an LVM Volume with an ext4 File System	15
3.3. NFS Share Setup	16
3.4. Exclusive Activation of a Volume Group in a Cluster	16
3.5. Configuring the Cluster Resources	18
3.6. Testing the Resource Configuration	21
Appendix A. Revision History	24

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 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.

```
# 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 general information about configuring fencing devices, see the *Red Hat Enterprise Linux 7 High Availability Add-On Reference*.

**Note**

When configuring a fencing device, you should ensure that your fencing device does not share power with the node that it controls.

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 60s 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 params \
ipaddr="z1pc.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=z1pc.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)
```

Chapter 2. An active/passive Apache Web Server in a Red Hat High Availability Cluster

This chapter describes how to configure an active/passive Apache web 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 web 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 Web Server 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 web 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.

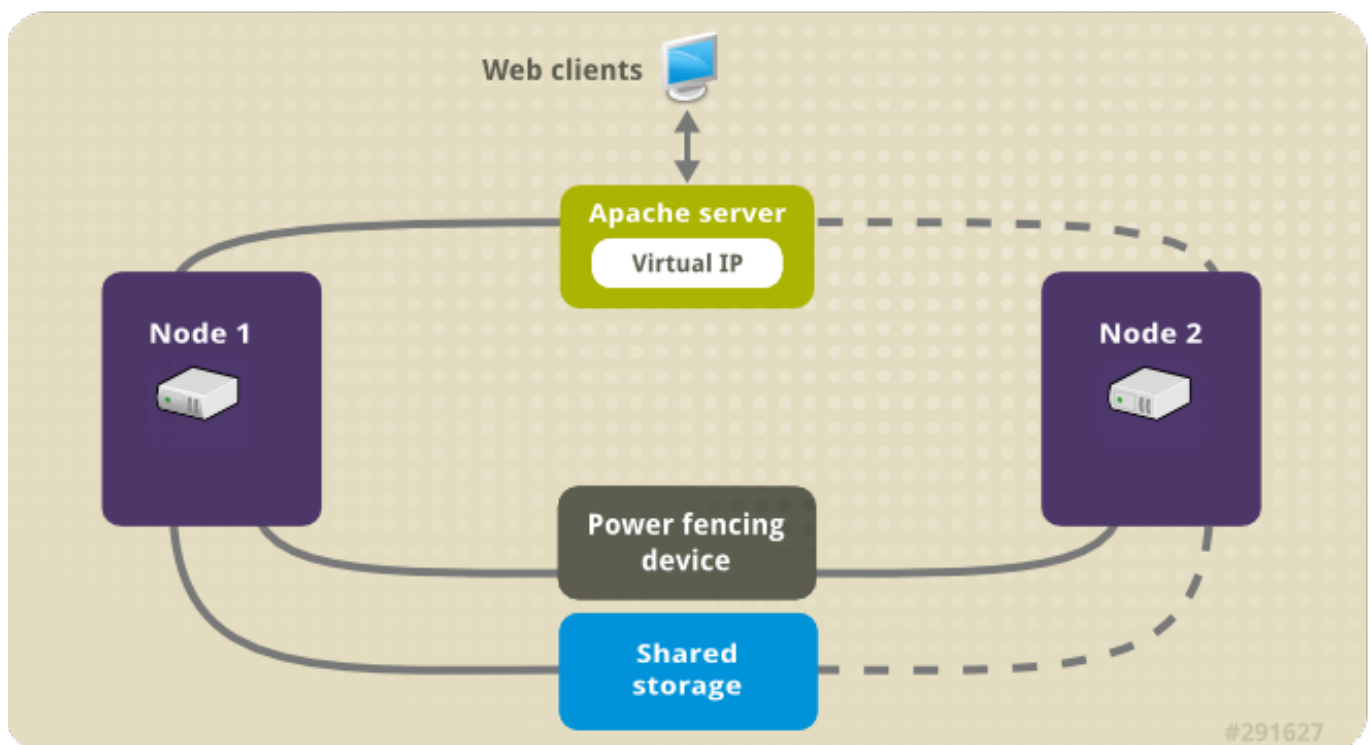


Figure 2.1. Apache Web Server in a Red Hat High Availability Two-Node Cluster

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

- ✦ A 2-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 the Apache web server.
- ✦ Shared storage for the nodes in the cluster, using iSCSI or Fibre Channel.

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 web server.

1. Ensure that the Apache HTTPD 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 web 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 web 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
  Order deny,allow
  Deny from all
  Allow from 127.0.0.1
</Location>
```

3. 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:

```
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.

- 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 run 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.

- 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”](#).
- A **Filesystem** resource named **my_fs**, that uses the filesystem device **/dev/my_vg/my_lv** you created in [Section 2.1, “Configuring an LVM Volume with an ext4 File System”](#).
- 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 can not be properly detected.
- 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.

- 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.

- 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.

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

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

- 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.

- To remove **z1** from **standby** mode, run the following command.


```
root@z1 ~]# pcs cluster 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. For information on controlling which node resources can run on, see the chapter on configuring cluster resources 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 web server. In this example, the nodes used are **z1.example.com** and **z2.example.com**.
- A power fencing device for each node of the webfarm 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 or Fibre Channel.

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 webfarm 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. Configuring fencing devices for each node of the webfarm 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 Set up

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
```

- 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
```

- 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 = []**.

- 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)
```

- 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.

- 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 run 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 cluster 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**.

```

[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.



Note

For the NFS notification to be processed correctly, the floating IP address must have a hostname 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
```

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.
 - a. 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
```

- b. 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
...
```

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

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

- d. 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
...
```

- e. 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.

- f. 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 cluster unstandby z1.example.com
```

Appendix A. Revision History

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 1.1-9	Tue Dec 9 2014	Steven Levine
Added nfs cluster configuration procedure		
Revision 1.1-6	Mon Dec 8 2014	Steven Levine
Updating load balancer cluster procedure.		
Revision 0.1-34	Fri Dec 5 2014	Steven Levine
Updating to implement new sort order on the Red Hat Enterprise Linux splash page.		
Revision 1.1-5	Thu Dec 04 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		
Revision 0.1-31	Wed May 21 2014	Steven Levine
Resolves: #886235 Document volume_list usage		
Revision 0.1-29	Tue May 20 2014	Steven Levine
Rebuild for style changes and updated draft		
Revision 0.1-20	Wed Apr 9 2014	Steven Levine
Updated Beta draft		
Revision 0.1-8	Fri Dec 6 2013	Steven Levine
Beta draft		
Revision 0.0-1	Wed Jan 16 2013	Steven Levine
First version for Red Hat Enterprise Linux 7		