Red Hat Enterprise Linux 7

High Availability Add-On Administration

Configuring Red Hat High Availability deployments

Last Updated: 2021-05-26

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

法律通告

摘要

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

目录

第 1 章 使用 PACEMAKER 创建红帽高可用性集群

本章论述了使用 **pcs** 创建红帽高可用性双节点集群的步骤。创建集群后,您可以配置所需的资源和资源组。

配置本章中提供的集群需要您的系统包含以下组件：

- 2 个节点，用于创建集群。在本例中，所用的节点为 **z1.example.com** 和 **z2.example.com**。
- 专用网络的网络交换机,用于集群节点和其它集群硬件（如网络电源交换机和光线通道交换机）间的通信。
- 集群中的每个节点都有一个电源隔离设备。这个示例使用 APC 电源交换机的两个端口，主机名为 **zapc.example.com**。

本章分为三个部分。

- 第 1.1 节 “集群软件安装” 提供安装集群软件的步骤。
- 第 1.2 节 “集群创建” 提供配置双节点集群的步骤。
- 第 1.3 节 “隔离配置” 提供为集群的每个节点配置隔离设备的步骤。

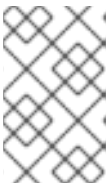
1.1. 集群软件安装

安装和配置集群的步骤如下。

1. 在集群的每个节点中，安装 Red Hat High Availability Add-On 软件包，以及 High Availability 性频道中的所有可用的隔离代理。

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

2. 如果您正在运行 **firewalld** 守护进程,请执行以下命令启用 Red Hat High Availability Add-On 所需的端口。



注意

您可以使用 **rpm -q firewalld** 命令确定您的系统中是否安装了 **firewalld** 守护进程。如果安装了 **firewalld** 守护进程,您可以使用 **firewall-cmd --state** 命令确定它是否在运行。

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

3. 要使用 **pcs** 配置集群并在节点间通信，您必须在每个节点中为用户 ID **hacluster**设置密码，它是 **pcs** 的管理帐户。建议每个节点上的用户 **hacluster** 的密码都相同。

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

4. 在配置集群前,必须启动并启用 **pcsd** 守护进程以便在每个节点启动时引导。这个守护进程和 **pcs** 命令一起工作,可以管理集群中的跨节点的配置。

   在集群的每个节点中执行以下命令启动 **pcsd** 服务并在系统启动时启用 **pcsd**。

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

5. 在要运行 **pcs** 的节点上,为集群中的每个节点验证 **pcs** 用户 **hacluster**。

   以下命令为示例双节点集群、**z1.example.com** 和 **z2.example.com** 中的两个节点在 **z1.example.com** 上验证用户 **hacluster**。

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

1.2. 集群创建

此流程创建由节点 **z1.example.com** 和 **z2.example.com**组成的红帽高可用性附加组件集群。

1. 从 **z1.example.com** 执行以下命令来创建由 **z1.example.com** 和 **z2.example.com** 组成的双节点集群 **my_cluster**。这会将集群配置文件传播到集群中的两个节点。此命令包含 **--start** 选项，该选项将在集群的两个节点上启动集群服务。

   ```
   [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. 在节点引导时，启用集群服务在集群中的每个节点上运行。

   注意

   对于特定环境，您可以跳过这一步来禁用集群服务。这可让您确保在节点重新加入集群前解决集群或您的资源中的任何问题。如果禁用了集群服务,则需要在重新引导节点时手动启动该服务,方法是在该节点上执行 **pcs cluster start** 命令。

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

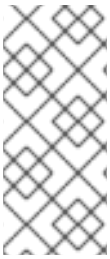
您可以使用 **pcs cluster status** 命令显示集群的当前状态。因为在使用 **pcs cluster setup** 命令的 **--start** 选项启动集群服务时,集群的启动和运行可能会有一些延迟,所以您应该确保在集群及其配置上执行后续操作前已启动并运行了集群。

```
[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. 隔离配置

您必须为集群中的每个节点配置保护设备。有关隔离配置命令和选项的详情请参考 *Red Hat Enterprise Linux 7 High Availability Add-On 参考*。有关隔离的一般信息及其在红帽高可用性集群中的重要程度,请 查看红帽高可用性集群中的隔离。



注意

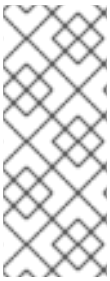
在配置隔离设备时，应该注意该设备是否与集群中的任何节点或者设备共享电源。如果某个节点及其隔离设备共享了电源，那么如果它的电源出现问题，集群可能就无法收到隔离功能的保护。这样的集群应该有冗余电源来保护设备和节点，或者具有没有和节点共享电源的额外的隔离设置。其他替代的隔离方法，比如 SBD 或存储隔离，也可以用来对电源问题提供冗余保护。

这个示例使用主机名为 **zapc.example.com** 的 APC 电源交换机来保护节点，并使用 **fence_apc_snmp** 隔离代理。因为这两个节点都使用同一隔离代理进行隔离,所以您可以使用 **pcmk_host_map** 和 **pcmk_host_list** 选项将这两个隔离设备配置为单一资源。

您可以使用 **pcs stonith create** 命令将该设备配置为 **stonith** 资源来创建隔离设备。以下命令配置名为 **myapc** 的 **stonith** 资源，作为节点 **z1.example.com** 和 **z2.example.com** 的 **fence_apc_snmp** 隔离代理。**pcmk_host_map** 选项将 **z1.example.com** 映射到端口 1，将 **z2.example.com** 映射到端口 2。APC 设备的登录值和密码都是 **apc**。默认情况下，该设备对每个节点都使用 60 秒的监视间隔时间。

请注意，您可以在为节点指定主机名时使用 IP 地址。

```
[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"
```



注意

当您创建 **fence_apc_snmp stonith** 设备时,您可能会看到以下警告信息,您可以安全地忽略：

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

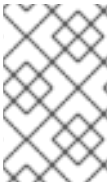
以下命令显示现有 STONITH 设备的参数。

```
[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)
```

配置了隔离设备后，您应该测试该设备。有关测试隔离设备的详情,请参考 " *高可用性附加组件参考" 中配置* Stonith。

注意

不要通过禁用网络接口来测试您的隔离设备，因为这不会正确测试隔离功能。



注意

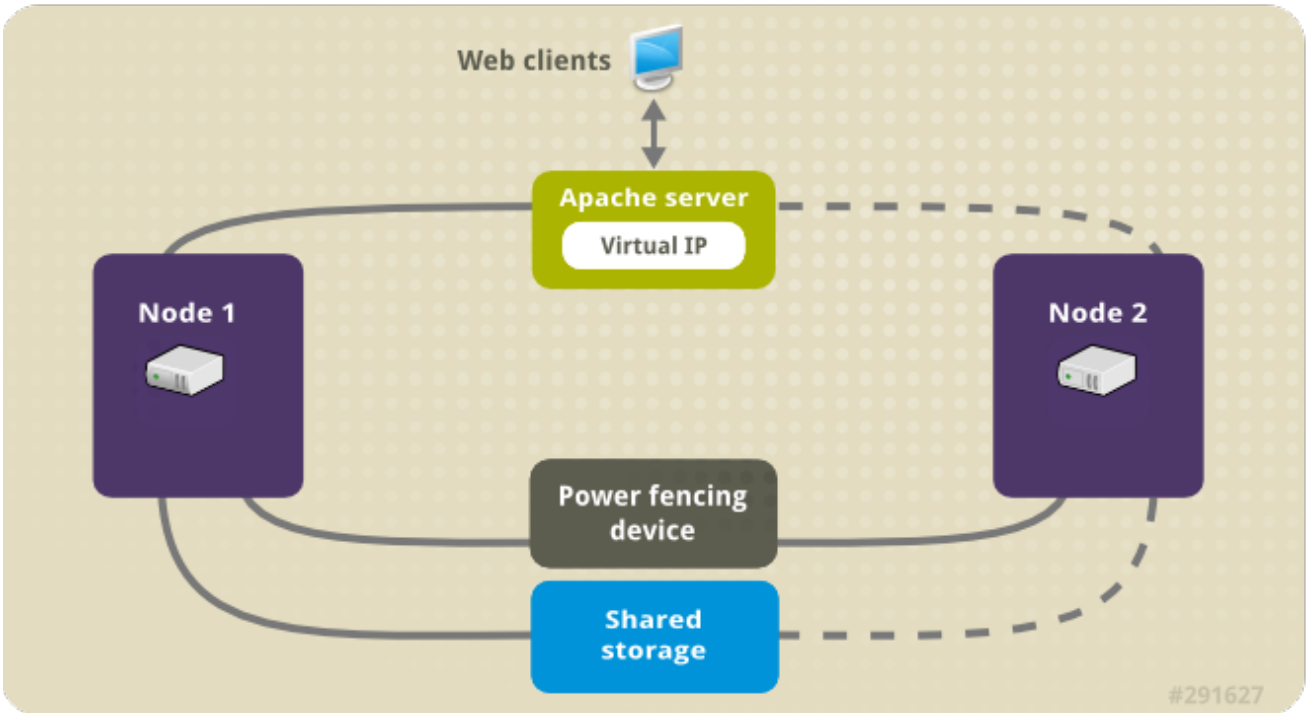
当配置了隔离功能，且启动集群后，网络重启会触发节点的隔离，即使没有超过超时时间也会重启网络。因此，不要在集群服务运行时重启网络服务，因为它将在节点上触发意外隔离。

第 2 章 在红帽高可用性集群中主动/被动 APACHE HTTP 服务器

本章论述了如何使用 **pcs** 在双节点 Red Hat Enterprise Linux High Availability 附加组件集群中配置主动/被动 Apache HTTP 服务器来配置集群资源。在这种情况下，客户端通过浮动 IP 地址访问 Apache HTTP 服务器。Web 服务器在集群的两个节点之一中运行。如果运行 web 服务器的节点出现问题，则 web 服务器会在集群的第二个节点上再次启动，以实现服务中断的最小化。

图 2.1 “Red Hat High Availability 双节点集群中的 Apache” 显示集群的高级别概述。集群是一个双节点 Red Hat High Availability 集群,它使用网络电源交换机和共享存储进行配置。集群节点连接到公用网络，以便客户端通过虚拟 IP 访问 Apache HTTP 服务器。Apache 服务器在 Node 1 或 Node 2 中运行，每个节点都可访问保存 Apache 数据的存储。

图 2.1. Red Hat High Availability 双节点集群中的 Apache



[D]

这个用例需要您的系统包括以下组件：

- 一个双节点 Red Hat High Availability 集群，为每个节点配置了电源隔离功能。此流程使用中提供的集群示例 第 1 章 *使用 Pacemaker 创建红帽高可用性集群*。
- Apache 需要的公共虚拟 IP 地址。
- 集群中节点的共享存储，使用 iSCSI、光纤或其他共享网络块设备。

集群被配置为带有 Apache 资源组，其中包含 web 服务器所需的集群组件：LVM 资源、文件系统资源、IP 地址资源以及 web 服务器资源。这个资源组可以从集群的一个节点切换到另外一个节点，允许其中两个节点运行 web 服务器。在为集群创建资源组前,您将执行以下步骤：

1. 配置挂载到逻辑卷 **my_lv** 中的 **ext4** 文件系统,如所述 第 2.1 节 “使用 ext4 文件系统配置 LVM 卷”。
2. 配置 Web 服务器,如所述 第 2.2 节 “Web 服务器配置”。
3. 确保只有集群可以激活包含 **my_lv** 的卷组,且在启动时不会在集群外激活卷组,如所述 第 2.3 节 “在集群中禁用卷组激活”。

执行这些流程后,您可以创建资源组及其包含的资源,如所述 第 2.4 节 “使用 pcs 命令创建资源和资源组”。

2.1. 使用 EXT4 文件系统配置 LVM 卷

这个用例要求您在集群节点之间共享的存储中创建 LVM 逻辑卷。

下面的步骤创建了 LVM 逻辑卷,然后在该卷中创建 **ext4** 文件系统。在这个示例中，使用共享分区 **/dev/sdb1** 来存储创建 LVM 逻辑卷的 LVM 物理卷。

注意

LVM 卷以及集群节点使用的对应分区和设备必须只能连接到集群节点。

因为 **/dev/sdb1** 分区是共享的存储,因此您只在一个节点中执行这个步骤,

1. 在分区 **/dev/sdb1** 中创建 LVM 物理卷。

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

2. 创建由物理卷 **my_vg** 组成的卷组 **/dev/sdb1**。

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

3. 使用卷组 **my_vg** 创建逻辑卷。

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

   您可以使用 **lvs** 命令来显示逻辑卷。

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

4. 在逻辑卷 **my_lv** 中创建 **ext4** 文件系统。

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

2.2. WEB 服务器配置

以下步骤配置 Apache HTTP 服务器。

1. 确定在集群的每个节点上安装了 Apache HTTP 服务器。您还需要安装 **wget** 工具才能检查 Apache HTTP 服务器的状态。

在每个节点上执行以下命令。

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

2. 要让 Apache 资源代理获得 Apache HTTP 服务器的状态,确保在集群中每个节点的 **/etc/httpd/conf/httpd.conf** 文件中有以下文本,并确保没有注释掉它。如果这个内容不存在，在文件的末尾添加这个内容。

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

3. 当使用 **apache** 资源代理来管理 Apache 时，它不使用 **systemd**。因此,您必须编辑 Apache 提供的 **logrotate** 脚本,使其不使用 **systemctl** 重新载入 Apache。

   删除集群中每个节点的 **/etc/logrotate.d/httpd** 文件中的以下行。

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

   使用以下三行替换您删除的行。

```
/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. 为 Apache 创建网页供服务。在集群的一个节点中挂载您创建的文件系统 第 2.1 节 “使用 ext4 文件系统配置 LVM 卷”,在该文件系统中创建文件 **index.html**,然后卸载该文件系统。

```
# 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. 在集群中禁用卷组激活

下面的步骤配置卷组的方式是,确保只有集群可以激活卷组,且在启动时不会在集群外激活卷组。如果卷组由集群外的系统激活,则有破坏卷组元数据的风险。

此流程修改 **/etc/lvm/lvm.conf** 配置文件中的 **volume_list** 条目。**volume_list** 条目中列出的卷组可以在集群管理器控制之外的本地节点中自动激活。与节点本地根目录和主目录相关的卷组应包含在此列表中。由集群管理器管理的所有卷组都必须从 **volume_list** 条目中排除。请注意,这个过程不需要使用 **clvmd**。

在集群的每个节点上执行以下步骤。

1. 执行以下命令,确保 **locking_type** 在 **/etc/lvm/lvm.conf** 文件中被设置为 1,且 **use_lvmetad** 被设置为 0。这个命令还会立即禁用和停止任何 **lvmetad** 进程。

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

2. 使用以下命令确定当前在本地存储中配置哪些卷组。这会输出当前配置的卷组列表。如果您在单独的卷组中为 root 分配了空间,并且为这个节点的主目录分配了空间,您会在输出中看到这些卷,如下例所示。

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

3. 将 **my_vg** 以外的卷组（您刚刚为集群定义的卷组）添加为 **/etc/lvm/lvm.conf** 配置文件中的 **volume_list** 的条目。

   例如：如果您在单独的卷组中为 root 和您的主目录分配了空间,您可以取消对 **lvm.conf** 文件的 **volume_list** 行的注释,并将这些卷组作为条目添加到 **volume_list** 中,如下所示。请注意,您刚刚为集群定义的卷组（本例中为(**my_vg**)没有在这个列表中。

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

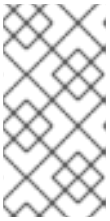
注意

如果节点上没有在集群管理器之外激活的本地卷组,您仍需要将 **volume_list** 条目初始化为 **volume_list = []**。

4. 重建 **initramfs** 引导镜像,以确保引导镜像不会尝试激活集群控制的卷组。使用以下命令更新 **initramfs** 设备。这个命令可能需要一分钟时间完成。

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

5. 重新引导节点。



注意

如果您在引导节点后安装了一个新的 Linux 内核,新的 **initrd** 镜像将适用于创建该节点时运行的内核,而不是重启节点时运行的新内核。您可以通过在重启前和重启后运行 **uname -r** 命令确定使用正确的 **initrd** 设备,以确定正在运行的内核发行版本。如果发行版本不同,请在重启新内核后更新 **initrd** 文件,然后重新引导节点。

6. 当节点重新引导时,通过在该节点上执行 **pcs cluster status** 命令来检查集群服务是否在该节点上再次启动。如果生成了信息 **Error: cluster is not currently running on this node**,请输入以下命令。

```
# pcs cluster start
```

   另外,您可以等待集群中的每个节点重新引导,并使用以下命令在每个节点上启动集群服务。

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

2.4. 使用 PCS 命令创建资源和资源组

这个用例需要您创建四个集群资源。为确保这些资源在同一节点上运行，它们都配置为资源组 **apachegroup** 的一部分。要创建的资源如下，按其启动顺序列出。

1. **LVM** 资源名为 **my_lvm**,它使用您在其中创建的 LVM 卷组 第 2.1 节 “使用 ext4 文件系统配置 LVM 卷”。
2. **Filesystem** 资源名为 **my_fs**,它使用您在其中创建的文件系统设备 **/dev/my_vg/my_lv** 第 2.1 节 “使用 ext4 文件系统配置 LVM 卷”。
3. **IPaddr2** 资源，它是 **apachegroup** 资源组的浮动 IP 地址。IP 地址不能是一个已经与物理节点关联的 IP 地址。如果没有指定 **IPaddr2** 资源的 NIC 设备,浮动 IP 必须位于与静态分配的 IP 地址相同的网络中,否则无法正确检测到分配浮动 IP 地址的 NIC 设备。
4. 一个名为 **Website** 的 **apache** 资源,它使用 **index.html** 文件和您在中定义的 Apache 配置 第 2.2 节 “Web 服务器配置”。

以下流程创建资源组 **apachegroup** 以及组包含的资源。资源将以您添加到组的顺序启动，并按照添加到组中的相反顺序停止。仅从集群的一个节点运行此步骤。

1. 以下命令创建 LVM 资源 **my_lvm**。这个命令指定 **exclusive=true** 参数以确保只有集群可以激活 LVM 逻辑卷。由于资源组 **apachegroup** 尚不存在，这个命令会创建资源组。

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

   当您创建资源时，会自动启动该资源。您可以使用以下命令确认资源已创建并启动。

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

   您可以使用 **pcs resource disable** 和 **pcs resource enable** 命令手动停止并启动单独的资源。

2. 以下命令为配置创建剩余的资源，并将其添加到现有资源组 **apachegroup** 中。

   ```
   [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. 创建资源和包含这些资源的资源组后，您可以检查集群的状态。请注意，所有四个资源都在同一个节点上运行。

   ```
   [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
```

请注意,如果您还没有如所述为集群配置隔离设备 第 1.3 节 “隔离配置”,默认情况下资源不会启动。

4. 当集群启动并运行后，您可以将浏览器指向定义为 **IPaddr2** 资源，查看示例显示，包括一个简单的单词 "Hello"。

```
Hello
```

如果发现您配置的资源没有运行,您可以运行 **pcs resource debug-start *resource*** 命令来测试资源配置。有关 **pcs resource debug-start** 命令的详情请参考 *高可用性附加组件参考手册*。

2.5. 测试资源配置

在显示的集群状态中 第 2.4 节 “使用 pcs 命令创建资源和资源组”,所有资源都在节点 **z1.example.com** 中运行。您可以按照以下流程将第一个节点设置为 **standby** 模式来测试资源组是否切换到节点 **z2.example.com**，之后该节点将不再能够托管资源。

1. 以下命令将节点 **z1.example.com** 置于 **standby** 模式。

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

2. 将节点 **z1** 置于待机模式后,检查集群状态。请注意，这些资源现在都应在 **z2** 中运行。

```
[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
```

定义的 IP 地址的网页仍会显示，而不中断。

3. 要从 **standby** 模式中删除 **z1**，请输入以下命令。

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

注意

从 **standby** 模式中删除节点不会自动导致资源恢复到该节点。这将依赖于资源的 **resource-stickiness** 值。有关 **resource-stickiness** meta 属性的详情,请参考 *红帽高可用性附加组件参考* 中的配置资源以引用其当前节点。

第 3 章 在红帽高可用性集群中有一个主动/被动 NFS 服务器

本章论述了如何使用共享存储在双节点 Red Hat Enterprise Linux High Availability 附加组件集群中配置高度可用的主动/被动 NFS 服务器。以下操作过程使用 **pcs** 来配置 Pacemaker 集群资源。在这种情况下，客户端会通过一个浮动 IP 地址访问 NFS 文件系统。NFS 服务器在集群中的两个节点中的一个上运行。如果运行 NFS 服务器的节点出现问题，则 NFS 服务器会在集群的第二个节点上再次启动，以实现服务中断的最小化。

这个用例需要您的系统包括以下组件：

- 两个节点,用于创建运行 Apache HTTP 服务器的集群。在本例中，所用的节点为 **z1.example.com** 和 **z2.example.com**。
- 集群中的每个节点都有一个电源隔离设备。这个示例使用 APC 电源交换机的两个端口，主机名为 **zapc.example.com**。
- NFS 服务器需要的一个公共虚拟 IP 地址。
- 集群中节点的共享存储，使用 iSCSI、光纤或其他共享网络块设备。

在双节点 Red Hat Enterprise Linux High 中配置高可用性的主动/被动 NFS 服务器需要执行以下步骤。

1. 创建运行 NFS 服务器并为集群中的每个节点配置隔离的集群,如所述 第 3.1 节 “创建 NFS 集群”。
2. 在集群中节点的共享存储中配置在 LVM 逻辑卷 **my_lv** 中挂载的 **ext4** 文件系统,如所述 第 3.2 节 “使用 ext4 文件系统配置 LVM 卷”。
3. 在 LVM 逻辑卷的共享存储中配置 NFS 共享,如所述 第 3.3 节 “NFS 共享设置”。
4. 确定只有集群可以激活包含逻辑卷 **my_lv** 的 LVM 卷组,且在启动时不会在集群外激活该卷组,如所述 第 3.4 节 “在集群中禁用卷组激活”。
5. 如所述创建集群资源 第 3.5 节 “配置集群资源”。
6. 测试您配置的 NFS 服务器,如所述 第 3.6 节 “测试资源配置”。

3.1. 创建 NFS 集群

使用以下步骤来安装和创建 NFS 集群。

1. 使用 **z1.example.com** 和 **z2.example.com** 中提供的流程在节点 和 上安装集群软件 第 1.1 节 “集群软件安装”。
2. 使用提供的流程,创建由 **z1.example.com** 和 **z2.example.com** 组成的双节点集群 第 1.2 节 “集群创建”。如该示例中一样,这使用案例命名集群 **my_cluster**。
3. 使用中介绍的步骤为集群的每个节点配置隔离设备 第 1.3 节 “隔离配置”。这个示例使用 APC 电源交换机的两个端口配置隔离,主机名为 **zapc.example.com**。

3.2. 使用 EXT4 文件系统配置 LVM 卷

这个用例要求您在集群节点之间共享的存储中创建 LVM 逻辑卷。

下面的步骤创建了 LVM 逻辑卷,然后在该卷中创建 **ext4** 文件系统。在这个示例中，使用共享分区 **/dev/sdb1** 来存储创建 LVM 逻辑卷的 LVM 物理卷。

注意

LVM 卷以及集群节点使用的对应分区和设备必须只能连接到集群节点。

因为 **/dev/sdb1** 分区是共享的存储,因此您只在一个节点中执行这个步骤,

1. 在分区 **/dev/sdb1** 中创建 LVM 物理卷。

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

2. 创建由物理卷 **my_vg** 组成的卷组 **/dev/sdb1**。

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

3. 使用卷组 **my_vg** 创建逻辑卷。

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

您可以使用 **lvs** 命令来显示逻辑卷。

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

4. 在逻辑卷 **my_lv** 中创建 **ext4** 文件系统。

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

3.3. NFS 共享设置

以下步骤为 NFS 守护进程故障转移配置 NFS 共享。您只需要在集群中的一个节点中执行这个步骤。

1. 创建 **/nfsshare** 目录。

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

2. 挂载您在 **/nfsshare** 目录中创建 第 3.2 节 “使用 ext4 文件系统配置 LVM 卷” 的 **ext4** 文件系统。

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

3. 在 **/nfsshare** 目录中创建 **exports** 目录树。

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

4. 将文件放在 **exports** 目录中用于 NFS 客户端访问。在本例中,我们创建名为 **clientdatafile1** 和 **clientdatafile2** 的测试文件。

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

5. 卸载 ext4 文件系统，并取消激活 LVM 卷组。

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

3.4. 在集群中禁用卷组激活

下面的步骤配置 LVM 卷组的方式是,确保只有集群可以激活卷组,且在启动时不会在集群外激活卷组。如果卷组由集群外的系统激活,则有破坏卷组元数据的风险。

此流程修改 **/etc/lvm/lvm.conf** 配置文件中的 **volume_list** 条目。**volume_list** 条目中列出的卷组可以在集群管理器控制之外的本地节点中自动激活。与节点本地根目录和主目录相关的卷组应包含在此列表中。由集群管理器管理的所有卷组都必须从 **volume_list** 条目中排除。请注意,这个过程不需要使用 **clvmd**。

在集群的每个节点上执行以下步骤。

1. 执行以下命令,确保 **locking_type** 在 **/etc/lvm/lvm.conf** 文件中被设置为 1,且 **use_lvmetad** 被设置为 0。这个命令还会立即禁用和停止任何 **lvmetad** 进程。

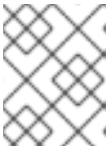
```
# lvmconf --enable-halvm --services --startstopservices
```

2. 使用以下命令确定当前在本地存储中配置哪些卷组。这会输出当前配置的卷组列表。如果您在单独的卷组中为 root 分配了空间,并且为这个节点的主目录分配了空间,您会在输出中看到这些卷,如下例所示。

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

3. 将 **my_vg** 以外的卷组（您刚刚为集群定义的卷组）添加为 **/etc/lvm/lvm.conf** 配置文件中的 **volume_list** 的条目。例如：如果您在不同的卷组中为 root 和您的主目录分配了空间,您可以取消对 **lvm.conf** 文件的 **volume_list** 行的注释,并将这些卷组作为条目添加到 **volume_list**,如下所示：

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



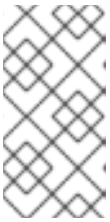
注意

如果节点上没有在集群管理器之外激活的本地卷组,您仍需要将 **volume_list** 条目初始化为 **volume_list = []**。

4. 重建 **initramfs** 引导镜像,以确保引导镜像不会尝试激活集群控制的卷组。使用以下命令更新 **initramfs** 设备。这个命令可能需要一分钟时间完成。

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

5. 重新引导节点。



注意

如果您在引导节点后安装了一个新的 Linux 内核,新的 **initrd** 镜像将适用于创建该节点时运行的内核,而不是重启节点时运行的新内核。您可以通过在重启前和重启后运行 **uname -r** 命令确定使用正确的 **initrd** 设备,以确定正在运行的内核发行版本。如果发行版本不同,请在重启新内核后更新 **initrd** 文件,然后重新引导节点。

6. 当节点重新引导时,通过在该节点上执行 **pcs cluster status** 命令来检查集群服务是否在该节点上再次启动。如果生成了信息 **Error: cluster is not currently running on this node**,请输入以下命令。

```
# pcs cluster start
```

另外,您可以等到集群中每个节点重新引导,并使用以下命令在集群中的所有节点上启动集群服务。

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

3.5. 配置集群资源

本节提供了为这个用例配置集群资源的步骤。

注意

建议您使用 **pcs resource create** 创建集群资源时,请立即执行 **pcs status** 命令验证资源是否正在运行。请注意,如果您还没有如所述为集群配置隔离设备 第 1.3 节 “隔离配置”,默认情况下资源不会启动。

如果发现您配置的资源没有运行,您可以运行 **pcs resource debug-start *resource*** 命令来测试资源配置。这会在集群控制之外启动服务。当配置的资源再次运行时,运行 **pcs resource cleanup *resource*** 使集群了解更新。有关 **pcs resource debug-start** 命令的详情请参考 *高可用性附加组件参考手册*。

以下步骤配置系统资源。为确保这些资源在同一节点上运行，它们都配置为资源组 **nfsgroup** 的一部分。资源将以您添加到组的顺序启动，并按照添加到组中的相反顺序停止。仅从集群的一个节点运行此步骤。

1. 以下命令创建名为 **my_lvm** 的 LVM 资源。这个命令指定 **exclusive=true** 参数以确保只有集群可以激活 LVM 逻辑卷。由于资源组 **nfsgroup** 尚不存在，这个命令会创建资源组。

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

检查集群的状态，以验证资源是否在运行。

```
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. 为集群配置 **Filesystem** 资源。

注意

您可以使用 **options=*options*** 参数指定挂载选项作为 **Filesystem** 资源资源配置的一部分。运行 **pcs resource describe Filesystem** 命令获取完整配置选项。

以下命令将名为 **nfsshare** 的 ext4 **Filesystem** 资源配置为 **nfsgroup** 资源组的一部分。这个文件系统使用您在中创建的 LVM 卷组和 ext4 文件系统, 第 3.2 节 “使用 ext4 文件系统配置 LVM 卷” 并将挂载到您创建的 **/nfsshare** 目录中 第 3.3 节 “NFS 共享设置”。

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

检查 **my_lvm** 和 **nfsshare** 资源是否正在运行。

```
[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. 创建资源组 **nfsgroup** 的名为 **nfs-daemon** 的 **nfsserver** 资源。

注意

nfsserver 资源允许您指定一个 **nfs_shared_infodir** 参数,该参数是 NFS 守护进程用来存储与 NFS 相关的有状态信息的目录。建议将此属性设置为您在这个导出集合中创建的 **Filesystem** 资源的子目录。这样可确保 NFS 守护进程将其有状态的信息存储在需要重新定位资源组时可供另一个节点使用的设备中。在这个示例中, **/nfsshare** 是由 **Filesystem** 资源管理的共享存储目录, **/nfsshare/exports/export1** 和 **/nfsshare/exports/export2** 是导出的目录, **/nfsshare/nfsinfo** 是 **nfsserver** 资源的共享目录。

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

4. 添加 **exportfs** 资源来导出 **/nfsshare/exports** 目录。这些资源是资源组 **nfsgroup** 的一部分。这为 NFSv4 客户端构建了一个虚拟目录。NFSv3 客户端也可以访问这些导出。

```
[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. 添加 NFS 客户端用来访问 NFS 共享的浮动 IP 地址资源。您指定的浮动 IP 地址需要反向 DNS 查找,或者必须在集群中所有节点上的 **/etc/hosts** 中指定。这个资源是资源组 **nfsgroup** 的一部分。在本示例部署中，我们使用 192.168.122.200 作为浮动 IP 地址。

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

6. 在完成整个 NFS 部署后，添加用于发送 NFSv3 重启通知的 **nfsnotify** 资源。这个资源是资源组 **nfsgroup** 的一部分。

注意

为了正确处理 NFS 通知，浮动 IP 地址必须具有与其关联的主机名，在 NFS 服务器和 NFS 客户端中都一致。

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

在创建资源和资源限制后，您可以检查集群的状态。请注意，所有资源都在同一个节点上运行。

```
[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. 测试资源配置

您可以按照以下步骤验证您的系统配置。您应该可以使用 NFSv3 或 NFSv4 挂载导出的文件系统。

1. 在与部署位于同一个网络中的、位于集群以外的一个节点中，通过挂载 NFS 共享来确定 NFS 共享。在本例中，我们使用 192.168.122.0/24 网络。

   ```
   # 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. 要验证您可以用 NFSv4 挂载 NFS 共享，将 NFS 共享挂载到客户端节点的目录中。挂载后，请确定导出目录的内容是可见的。测试后卸载共享。

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

3. 确定您可以用 NFSv3 挂载 NFS 共享。挂载后,请确定测试文件 **clientdatafile1** 是可见的。与 NFSv4 不同,因为 NFSV3 不使用虚拟文件系统,所以您必须挂载一个特定的导出。测试后卸载共享。

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

4. 要测试故障切换,请执行以下步骤。

   a. 在集群外的节点中挂载 NFS 共享,并确认可以访问我们中创建的 **clientdatafile1** 第 3.3 节 “NFS 共享设置”。

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

b. 在集群的一个节点中，决定集群中的哪个节点正在运行 **nfsgroup**。在本例中，**nfsgroup** 在 **z1.example.com** 上运行。

```
[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. 在集群的一个节点中，将运行 **nfsgroup** 的节点设置为待机模式。

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

d. 验证 **nfsgroup** 是否在另外一个集群节点上成功启动。

```
[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. 在您挂载了 NFS 共享的集群之外的节点中，确认这个外部节点仍然可以访问 NFS 挂载中的测试文件。

```
# ls nfsshare
clientdatafile1
```

在短暂的故障转移过程中,服务可能会在短暂时间内不可用,但可以在没有用户干预的情况下恢复。默认情况下，使用 NFSv4 的客户端可能最多需要 90 秒恢复该挂载。这个 90 秒代表服务器启动时观察到的 NFSv4 文件租期的宽限期。NFSv3 客户端应该在几秒钟内就可以恢复对该挂载的访问。

f. 从集群的一个节点中删除最初运行 **nfsgroup** 的节点，从待机模式中删除最初运行的节点。这本身不会将集群资源移至此节点。

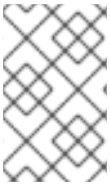
```
[root@z1 ~]# pcs node unstandby z1.example.com
```

注意

从 **standby** 模式中删除节点不会自动导致资源恢复到该节点。这将依赖于资源的 **resource-stickiness** 值。有关 **resource-stickiness** meta 属性的详情，请参考 *红帽高可用性附加组件参考* 中的配置资源以引用其当前节点。

第 4 章 RED HAT HIGH AVAILABILITY CLUSTER（RED HAT ENTERPRISE LINUX 7.4 和 LATER）中的主动/主动 SAMBA SERVER

自 Red Hat Enterprise Linux 7.4 发行版本起,Red Hat Resilient Storage 附加组件支持使用 Pacemaker 在主动/主动集群配置中运行 Samba。Red Hat Resilient Storage Add-On 包括 High Availability Add-On。



注意

有关 Samba 支持政策的更多信息, 请参阅 RHEL Resilient Storage 支持政策 - ctdb General 策略及 RHEL Resilient Storage 支持政策 - 通过红帽客户门户网站上的其他协议导出 gfs2 内容。

本章论述了如何使用共享存储在双节点 Red Hat Enterprise Linux High Availability 附加组件集群中配置高可用性主动/主动 Samba 服务器。以下操作过程使用 **pcs** 来配置 Pacemaker 集群资源。

这个用例需要您的系统包括以下组件：

- 两个节点,用于创建运行 Clustered Samba 的集群。在本例中,所用的节点为 **z1.example.com** 和 **z2.example.com**,它们的 IP 地址为 **192.168.1.151** 和 **192.168.1.152**。
- 集群中的每个节点都有一个电源隔离设备。这个示例使用 APC 电源交换机的两个端口，主机名为 **zapc.example.com**。
- 集群中节点的共享存储,使用 iSCSI 或光纤频道。

在双节点 Red Hat Enterprise Linux High Availability Add-On 集群中配置高可用性主动/主动 Samba 服务器需要执行以下步骤。

1. 创建导出 Samba 共享并为集群中的每个节点配置隔离的集群,如所述 第 4.1 节 “创建集群”。
2. 在集群的 LVM 逻辑卷 **my_clv** 中为集群中的节点配置共享存储中的 **gfs2** 文件系统,如所述 第 4.2 节 “使用 GFS2 文件系统配置集群 LVM 卷”。
3. 在集群的每个节点上配置 Samba 第 4.3 节 “配置 Samba”。
4. 如所述创建 Samba 集群资源 第 4.4 节 “配置 Samba 集群资源”。
5. 测试您配置的 Samba 共享,如所述 第 4.5 节 “测试资源配置”。

4.1. 创建集群

使用以下步骤安装和创建用于 Samba 服务的集群：

1. 使用 **z1.example.com** 和 **z2.example.com** 中提供的流程在节点 和 上安装集群软件 第 1.1 节 “集群软件安装”。
2. 使用提供的流程,创建由 **z1.example.com** 和 **z2.example.com** 组成的双节点集群 第 1.2 节 “集群创建”。如该示例中一样,这使用案例命名集群 **my_cluster**。
3. 使用中介绍的步骤为集群的每个节点配置隔离设备 第 1.3 节 “隔离配置”。这个示例使用 APC 电源交换机的两个端口配置隔离,主机名为 **zapc.example.com**。

4.2. 使用 GFS2 文件系统配置集群 LVM 卷

这个用例要求您在集群节点间共享的存储中创建集群的 LVM 逻辑卷。

这部分论述了如何使用那个卷中的 GFS2 文件系统创建集群 LVM 逻辑卷。在这个示例中,使用共享分区 **/dev/vdb** 来存储创建 LVM 逻辑卷的 LVM 物理卷。

注意

LVM 卷以及集群节点使用的对应分区和设备必须只能连接到集群节点。

在开始此步骤前,在集群的两个节点上安装 **lvm2-cluster** 和 **gfs2-utils** 软件包,它们是 Resilient Storage 频道的一部分。

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

因为 **/dev/vdb** 分区是共享的存储,因此您只在一个节点中执行这个步骤,

1. 将全局 Pacemaker 参数 **no_quorum_policy** 设置为 **freeze**。这可防止在每次 quorum 丢失时整个集群被隔离。有关设置该策略的更多信息,请参阅 *全局文件系统 2*。

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

2. 设置 **dlm** 资源。这是 **clvmd** 服务和 GFS2 文件系统所需的依赖项。

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

3. 将 **clvmd** 设置为集群资源。

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

   请注意,作为启动过程的一部分, **ocf:heartbeat:clvm** 资源代理将 **/etc/lvm/lvm.conf** 文件中的 **locking_type** 参数设置为 **3**,并禁用 **lvmetad** 守护进程。

4. 设置 **clvmd** 和 **dlm** 依赖项并启动顺序。**clvmd** 资源必须在 **dlm** 资源后启动,且必须与 **dlm** 资源在同一节点上运行。

   ```
   [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. 验证 **dlm** 和 **clvmd** 资源是否在所有节点上运行。

   ```
   [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. 创建集群逻辑卷

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

7. 另外,要验证卷是否已成功创建,您可以使用 **lvs** 命令显示逻辑卷。

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

8. 使用 GFS2 文件系统格式化卷。在本例中, **my_cluster** 是集群名称。本例指定 **-j 2** 指定两个日志,因为您配置的日志数量必须等于集群中的节点数量。

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

9. 创建 **Filesystem** 资源,它将 Pacemaker 配置为挂载和管理文件系统。这个示例创建了名为 **fs** 的 **Filesystem** 资源,并在集群的两个节点上创建 **/mnt/gfs2share**。

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

10. 为 GFS2 文件系统和 **clvmd** 服务配置依赖关系和启动顺序。GFS2 必须在 **clvmd** 后启动,且必须与 **clvmd** 在同一个节点上运行。

```
[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. 验证 GFS2 文件系统是否如预期挂载。

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

4.3. 配置 SAMBA

以下流程初始化 Samba 环境并在集群节点上配置 Samba。

1. 在集群的两个节点上执行以下步骤：

   a. 安装 **samba**、**ctdb** 和 **cifs-utils** 软件包。

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

   b. 如果您正在运行 **firewalld** 守护进程,请运行以下命令启用 **ctdb** 和 **samba** 服务所需的端口。

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

c. 输入以下命令以确保这些守护进程没有运行且不会在引导时启动。请注意,不是所有这些守护进程都可能存在或者在您的系统中运行。

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

d. 在 **/etc/samba/smb.conf** 文件中,配置 Samba 服务器并设置 **[public]** 共享定义。例如：

```
# 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
```

有关将 Samba 配置为独立服务器的详情,如本例中所示,以及使用 **testparm** 实用程序验证 **smb.conf** 文件的详情,请参考 *系统管理员指南中的 File and Print Servers 部分。*

e. 将集群节点的 IP 地址添加到 **/etc/ctdb/nodes** 文件中。

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

f. 要在集群节点之间进行负载平衡,您可以将两个或者多个 IP 地址添加到 **/etc/ctdb/public_addresses** 文件中,可用于访问此集群导出的 Samba 共享。这些是在 DNS 中为 Samba 服务器的名称配置的 IP 地址,也是 SMB 客户端要连接的地址。将 Samba 服务器的名称配置为具有多个 IP 地址的 DNS 类型 A 记录,并允许轮询 DNS 在集群节点中分发客户端。

在本例中,DNS 条目 **linuxserver.example.com** 是由 **/etc/ctdb/public_addresses** 文件中列出的地址定义的。在这个版本中,DNS 会以轮循(round-robin)方式在集群节点中分发 Samba 客户端。请注意,在执行这种情况时,DNS 条目应该与您的需要匹配。

将可用于访问此集群导出的 Samba 共享的 IP 地址添加到 **/etc/ctdb/public_addresses** 文件中。

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

g. 创建 Samba 组,然后为公共测试共享目录添加本地用户,将之前创建的组设置为主组群。

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

h. 请确定在 CTDB 相关目录中 SELinux 上下文正确。

```
# 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. 在集群的一个节点上执行以下步骤：

a. 为 CTDB 锁定文件和公共共享设置目录。

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

b. 更新 GFS2 共享中的 SELinux 上下文。

```
[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. 配置 SAMBA 集群资源

本节提供了为这个用例配置 Samba 集群资源的步骤。

以下流程创建了名为 **samba.cib** 的集群 **cib** 文件的快照,并将该资源添加到该测试文件中,然后直接在正在运行的集群中配置这些资源。配置资源和约束后,流程将 **samba.cib** 的内容推送到正在运行的集群配置文件。

在集群的一个节点上运行以下步骤。

1. 创建 **cib** 文件的快照,即集群配置文件。

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

2. 创建 Samba 使用 CTDB 资源。将这个资源作为克隆的资源创建,以便它在两个集群节点上运行。

```
[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. 创建克隆的 Samba 服务器。

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

4. 为集群资源创建 colocation 和顺序限制。启动顺序是 Filesystem 资源、CTDB 资源,然后是 Samba 资源。

```
[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. 将 **cib** 快照的内容推送到集群。

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

6. 检查集群的状态，以验证资源是否在运行。

   请注意,在 Red Hat Enterprise Linux 7.4 中, CTDB 可能需要几分钟时间来启动 Samba、导出共享和稳定。如果在这个过程完成前检查集群状态,您可能会看到一个 CTDB 状态调用失败的消息。这个过程完成后,您可以运行 **pcs resource cleanup ctdb-clone** 命令从显示中清除这个信息。

```
[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 ]
```

注意

如果发现您配置的资源没有运行,您可以运行 **pcs resource debug-start *resource*** 命令来测试资源配置。这会在集群控制之外启动服务。如果配置的资源再次运行,运行 **pcs resource cleanup *resource*** 使集群了解更新。有关 **pcs resource debug-start** 命令的详情请参考 *High Availability Add-On Reference* 手册中的 Enabling、Disabling 和 Banning Cluster Resources 部分。

4.5. 测试资源配置

如果 Samba 配置成功,应该可以在集群的节点中挂载 Samba 共享。以下示例步骤挂载了一个 Samba 共享。

1. 在集群节点中添加现有用户到 **smbpasswd** 文件并分配密码。在以下示例中,有一个现有用户 **smbuser**。

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

2. 挂载 Samba 共享：

```
[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. 检查是否挂载文件系统：

```
[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)
```

要检查 Samba 恢复,请执行以下步骤。

1. 使用以下命令手动停止 CTDB 资源：

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

2. 停止资源后,系统应该恢复该服务。使用 **pcs status** 命令检查集群状态。您应该看到 **ctdb-clone** 资源已启动,但您也会看到 **ctdb_monitor** 失败。

```
[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
...
```

要从状态中删除这个错误,请在其中一个集群节点上输入以下命令：

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

附录 A. 修订记录

修订 6-1 为 7.7 GA 发布版本准备文件。	Wed Aug 7 2019	Steven Levine
修订 5-2 为 7.6 GA 发布准备文件。	Thu Oct 4 2018	Steven Levine
修订 4-2 为 7.5 GA 发布准备文件。	Wed Mar 14 2018	Steven Levine
修订 4-1 为 7.5 Beta 发布编写文件。	Thu Dec 14 2017	Steven Levine
修订 3-4 更新版本 7.4。	Wed Aug 16 2017	Steven Levine
修订 3-3 发布 7.4 GA 的文件版本。	Wed Jul 19 2017	Steven Levine
修订 3-1 为 7.4 发布版本准备文件。	Wed May 10 2017	Steven Levine
修订 2-6 7.3 的更新	Mon Apr 17 2017	Steven Levine
修订 2-4 7.3 GA 发布的版本。	Mon Oct 17 2016	Steven Levine
修订 2-3 为 7.3 Beta 发布准备文件。	Fri Aug 12 2016	Steven Levine
修订 1.2-3 为 7.2 GA 发布版本准备文件。	Mon Nov 9 2015	Steven Levine
修订 1.2-2 为 7.2 Beta 发布准备文件。	Tue Aug 18 2015	Steven Levine
修订 1.1-19 7.1 GA 版本	Mon Feb 16 2015	Steven Levine
修订 1.1-10 7.1 Beta 发行版本的版本	Thu Dec 11 2014	Steven Levine
修订 0.1-33 7.0 GA 发行版本	Mon Jun 2 2014	Steven Levine