Red Hat Enterprise Linux 7 High Availability Add-On の管理

High Availability Add-On の設定と管理

Steven Levine

Red Hat Enterprise Linux 7 High Availability Add-On の管理

High Availability Add-On の設定と管理

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

法律上の通知

概要

High Availability Add-On の管理 は、Red Hat Enterprise Linux 7 向けの High Availability Add-On の設定と管理について説明しています。

目次

第1章 Pacemaker を使用した Red Hat High Availability クラスターの作成

本章では、**pcs** コマンドを使用して 2 ノードの Red Hat High Availability クラスターを作成する手順を説明します。クラスターの作成後、必要なリソースやリソースグループを設定できます。

本章で説明しているクラスターを設定する場合には次のコンポーネントが必要になります。

- 2 ノード、クラスターを構成させるノードです。ここでは **z1.example.com** と **z2.example.com** という名前にしています。
- プライベートネットワーク用のネットワークスイッチ、クラスター同士の通信およびネットワーク電源スイッチやファイバーチャンネルスイッチなどのクラスターハードウェアとの通信に必要になります。
- 各ノード用の電源フェンスデバイス、ここでは APC 電源スイッチの 2 ポートを使用しています。ホスト名は **zapc.example.com** にしています。

本章は 3 つの項に分かれています。

- 「クラスターソフトウェアのインストール」では、クラスターソフトウェアのインストール手順を説明します。
- 「クラスターの作成」では、2 ノードクラスターの設定手順を説明します。
- 「排他処理の設定」では、クラスターの各ノードにフェンスデバイスを設定する手順を説明します。

1.1. クラスターソフトウェアのインストール

クラスターのインストールおよび設定手順を以下に示します。

1. クラスターの各ノードに、Red Hat High Availability Add-On ソフトウェアパッケージと使用可能なすべてのフェンスエージェントを High Availability チャンネルからインストールします。

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

2. **firewalld** デーモンを実行している場合は、以下のコマンドを実行して Red Hat High Availability Add-On が必要とするポートを有効にします。

注記

firewalld デーモンがシステムにインストールされているかどうかを確認するには、**rpm -q firewalld** コマンドを実行します。**firewalld** デーモンがインストールされている場合は、**firewall-cmd --state** コマンドを使用して実行されているかどうかを確認できます。

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

3. **pcs** を使ってクラスターの設定やノード間の通信を行うため、**pcs** の管理アカウントとなるユーザー ID **hacluster** のパスワードを各ノードに設定しなければなりません。ユーザー **hacluster** のパスワードは各ノードで同じパスワードにすることを推奨しています。

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

4. クラスターを設定する前に、各ノードで起動時にブートするよう **pcsd** デーモンが起動および有効化されている必要があります。このデーモンは **pcs** コマンドで動作し、クラスターのノード全体で設定を管理します。

   クラスターの各ノードで次のコマンドを実行し **pcsd** サービスを起動、またシステムの起動時に **pcsd** が有効になるよう設定します。

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

5. **pcs** を実行するノードでクラスター内の各ノードの **pcs** ユーザー **hacluster** を認証します。

   次のコマンドでは、**z1.example.com** と **z2.example.com** の 2 ノードで構成されるクラスターの両ノードに対してユーザー **hacluster** の認証を **z1.example.com** 上で行っています。

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

1.2. クラスターの作成

この手順では、**z1.example.com** および **z2.example.com** ノードで構成される Red Hat High Availability Add-On を作成します。

1. 次のコマンドを **z1.example.com** で実行し ノード **z1.example.com** とノード **z2.example.com** で構成される 2 ノードクラスターの **my_cluster** を作成します。これによりクラスター設定ファイルがクラスター内の全ノードに伝搬されます。このコマンドには **--start** オプションが含まれます。このオプションを使用すると、クラスター内の全ノードでクラスターサービスが起動されます。

```
[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. クラスターサービスを有効にし、ノードがブートしたときにクラスターの各ノードでクラスターサービスが実行されるようにします。

注記

使用環境によってクラスターサービスを無効にしておきたい場合などはこの手順を省いて構いません。この手順を行うことで、ノードがダウンした場合にクラスターやリソース関連の問題をすべて解決してからそのノードをクラスターに戻すことができます。クラスターサービスを無効にしている場合には、ノードを再起動する時に **pcs cluster start** コマンドを使って手作業でサービスを起動しなければならないので注意してください。

```
# pcs cluster enable --all
```

pcs cluster status コマンドを使用するとクラスターの現在の状態を表示できます。**pcs cluster setup** コマンドの **--start** オプションを使用してクラスターサービスを起動した場合は、クラスターが稼働する前に若干の遅延が発生することがあるため、クラスターに対する後続アクションと設定を行う前にクラスターが稼働していることを確認する必要があります。

```
[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. 排他処理の設定

クラスターの各ノードにフェンスデバイスを設定する必要があります。フェンスデバイスの設定に関する情報は、『Red Hat Enterprise Linux 7 High Availability Add-On Reference』 を参照してください。

注記

フェンスデバイスの設定をする際、そのフェンスデバイスで管理を行うノードと電源が共有されていないことを必ず確認してください。

ここでは **zapc.example.com** というホスト名の APC 電源スイッチを使ってノードの排他処理を行います。このスイッチは **fence_apc_snmp** フェンスエージェントを使用します。ノードはすべて同じフェンスエージェントで排他処理されるため、**pcmk_host_map** と **pcmk_host_list** のオプションを使ってすべてのフェンスデバイスを一つのリソースとして設定できます。

pcs stonith create コマンドを使って **stonith** リソースとしてデバイスを設定し、フェンスデバイスを作成します。次のコマンドは、**z1.example.com** および **z2.example.com** ノードの **fence_apc_snmp** フェンスエージェントを使用する **myapc** という名前の **stonith** リソースを設定します。**pcmk_host_map** オプションは **z1.example.com** をポート 1 にマップし、**z2.example.com** をポート 2 にマップします。APC デバイスのログイン値とパスワードはいずれも **apc** です。デフォルトでは、このデバイスは各ノードに対して 60 秒間隔で監視を行います。

ノードのホスト名を指定する場合、IP アドレスを使用することができます。

```
[root@z1 ~]# pcs stonith create myapc fence_apc_snmp params \
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"
```

注記

fence_apc_snmp stonith デバイスを作成するときに次のような警告メッセージが表示されることがありますがこのメッセージは無視して構いません。

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

次のコマンドを使うと既存の STONITH デバイスのパラメーターが表示されます。

```
[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)
```

第2章 Red Hat High Availability クラスターのアクティブ/パッシブ Apache HTTP サーバー

本章では、**pcs** コマンドを使用してクラスターリソースを設定し、2 ノードの Red Hat Enterprise Linux High Availability Add-On クラスターでアクティブ/パッシブな Apache HTTP サーバーを設定する方法を説明します。このユースケースでは、クライアントはフローティング IP アドレスを用いて Apache HTTP サーバーにアクセスします。Web サーバーは 2 つのノードの 1 つで実行されます。Web サーバーが稼働しているノードが正常に動作しなくなった場合、Web サーバーは 2 つ目のノードで再起動され、サービスの中断は最小限になります。

図2.1「2 ノード Red Hat High Availability クラスターの Apache」 はクラスターのハイレベルの概要を示しています。クラスターはネットワーク電源スイッチおよび共有ストレージで設定される 2 ノードの Red Hat High Availability クラスターです。クライアントは仮想 IP を用いて Apache HTTP サーバーへアクセスするため、クラスターノードはパブリックネットワークに接続されます。Apache サーバーはノード 1 またはノード 2 のいずれかで実行されます。いずれのノードも Apache のデータが保持されるストレージへアクセスできます。

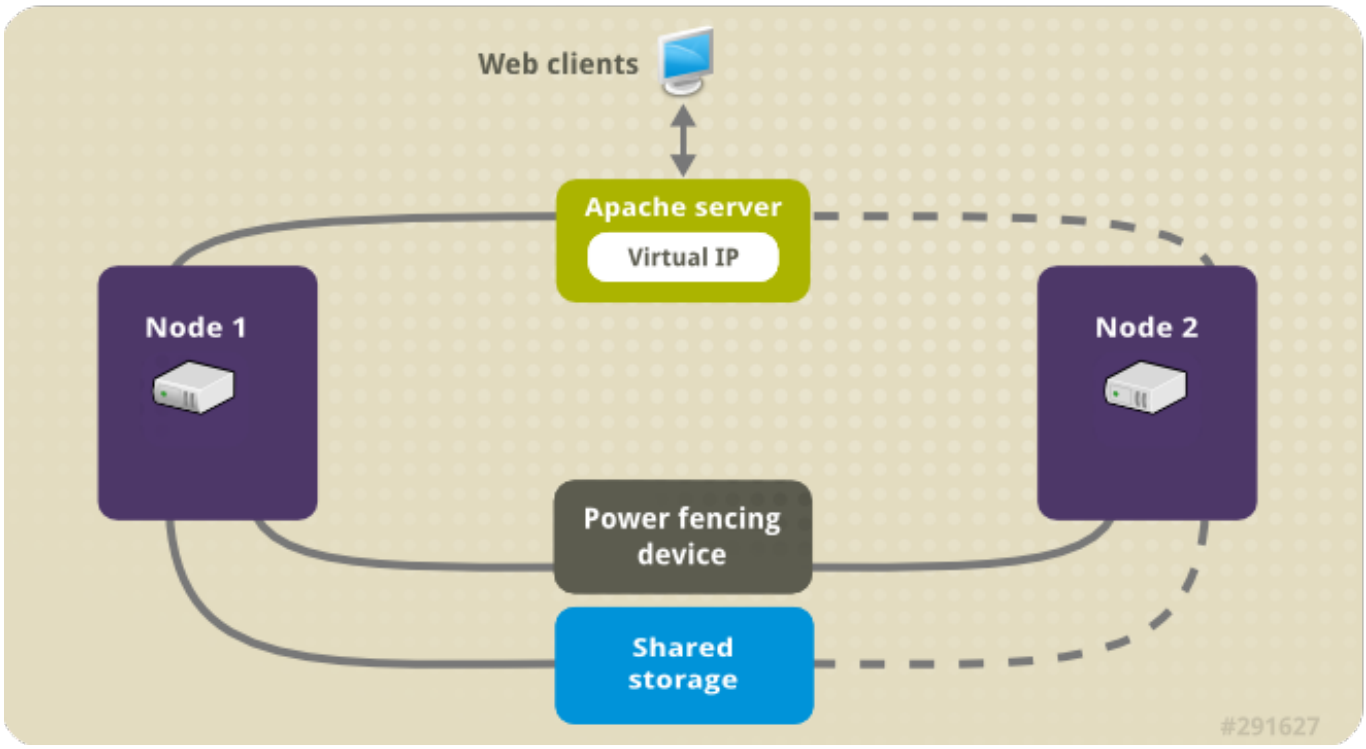


図2.1 2 ノード Red Hat High Availability クラスターの Apache

このユースケースでは、システムに以下のコンポーネントが必要になります。

- 各ノードに電源フェンスが設定されている 2 ノードの Red Hat High Availability クラスター。この手順では 1章*Pacemaker を使用した Red Hat High Availability クラスターの作成* に記載されているクラスターの例を使用します。
- Apache に必要なパブリック仮想 IP アドレス。
- iSCSI またはファイバーチャネルを使用する、クラスターのノードに対する共有ストレージ。

web サーバーで必要とされる LVM リソース、ファイルシステムリソース、IP アドレスリソース、web サーバーリソースなどのクラスターコンポーネントを含ませた Apache リソースグループでクラスターが設定されます。このリソースグループはクラスター内の一つのノードから別のノードへのフェールオーバーが可能なため、いずれのノードでも web サーバーを稼働することができます。クラスターにリソースグループを作成する前に次の手順を行います。

1. 「LVM ボリュームを ext4 ファイルシステムで設定」 の説明に従い **my_lv** 論理ボリュームに **ext4** ファイルシステムを設定します。
2. 「Web サーバーの設定」 の説明に従い web サーバーを設定します。
3. 「ボリュームグループのアクティブ化をクラスター内に限定」 の説明に従い、**my_lv** を含むボリュームグループの作動はクラスターでしか行えないよう限定し、またボリュームグループが起動時にクラスター以外の場所で作動しないようにします。

上記の手順をすべて完了したら、「pcs コマンドを使用したリソースおよびリソースグループの作成」 の説明に従いリソースグループおよびそのグループに含ませるリソースを作成します。

2.1. LVM ボリュームを ext4 ファイルシステムで設定

このユースケースでは、クラスターのノード間で共有されるストレージに LVM 論理ボリュームを作成する必要があります。

次の手順に従い LVM 論理ボリュームを作成しその論理ボリューム上に **ext4** ファイルシステムを作成します。ここでは **/dev/sdb1** 共有パーティションを使って LVM 論理ボリュームの作成元となる LVM 物理ボリュームを格納します。

注記

LVM ボリューム、該当パーティション、クラスターノードで使用するデバイスなどはクラスターノード以外には接続しないでください。

/dev/sdb1 パーティションは共有させるストレージとなるため、この手順は一つのノードでのみ行います。

1. LVM 物理ボリュームを **/dev/sdb1** パーティション上に作成します。

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

2. **/dev/sdb1** 物理ボリュームで構成される **my_vg** ボリュームグループを作成します。

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

3. **my_vg** ボリュームグループを使用する論理ボリュームを作成します。

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

lvs コマンドを使って論理ボリュームを表示してみます。

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

4. **ext4** ファイルシステムを **my_lv** 論理ボリューム上に作成します。

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

2.2. Web サーバーの設定

次の手順に従って Apache HTTP サーバーを設定します。

1. Apache HTTP サーバーがクラスターの各ノードにインストールされているようにしてください。また、Apache HTTP サーバーの状態を確認するには、**wget** ツールをクラスターにインストールする必要があります。

   各ノードで次のコマンドを実行します。

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

2. Apache リソースエージェントが Apache HTTP サーバーの状態を取得できるようにするため、クラスターの各ノードの **/etc/httpd/conf/httpd.conf** ファイルに以下のテキストが含まれ、コメントアウトされていないことを確認してください。このテキストが含まれていない場合は、このファイルの最後に追加します。

```

<Location /server-status>
  SetHandler server-status
  Order deny,allow
  Deny from all
  Allow from 127.0.0.1
</Location>
```

3. Apache で提供する web ページを作成します。クラスター内のいずれかのノードに「LVM ボリュームを ext4 ファイルシステムで設定」で作成したファイルシステムをマウントし、そのファイルシステム上で **index.html** ファイルを作成したら再びファイルシステムをアンマウントします。

```
# 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
```

4. **apache** リソースエージェントを使用して Apache を管理する場合、**systemd** は使用されません。そのため、Apache のリロードに **systemctl** が使用されないようにするため、Apache によって提供される **logrotate** スクリプトを編集する必要があります。

/etc/logrotate.d/httpd ファイルにある以下の行を削除します。

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

削除した行を以下の行に置き換えます。

```
/usr/sbin/httpd -f /etc/httpd/conf/httpd.conf -c "PidFile
/var/run/httpd.pid" -k graceful > /dev/null 2>/dev/null || true
```

2.3. ボリュームグループのアクティブ化をクラスター内に限定

次の手順でボリュームグループを設定すると、クラスターでしかボリュームグループを作動することができなくなり、またボリュームグループは起動時にクラスター以外の場所では作動しなくなります。クラスター以外のシステムでボリュームグループが作動されるとボリュームグループのメタデータが破損する恐れがあります。

この手順では **/etc/lvm/lvm.conf** 設定ファイル内の **volume_list** のエントリーを編集します。**volume_list** のエントリーに記載されているボリュームグループはクラスターマネージャーの管轄外となるローカルノードでの自動作動が許可されます。ノードのローカルな root ディレクトリやホームディレクトリに関連するボリュームグループはこのリストに含ませてください。クラスターマネージャーで管理するボリュームグループは **volume_list** のエントリーには入れないでください。ここでの手順に **clvmd** を使用する必要はありません。

クラスター内の各ノードで以下の手順を行います。

1. 次のコマンドを実行して、**/etc/lvm/lvm.conf** ファイルで **locking_type** が 1 に設定されていることと **use_lvmetad** が 0 に設定されていることを確認します。また、このコマンドを実行すると、すべての **lvmetad** プロセスがすぐに無効になり、停止します。

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

2. 次のコマンドでローカルストレージに現在設定されているボリュームグループを確認します。現在設定されているボリュームグループ一覧が出力されます。このノード上に root ディレクトリ用のボリュームグループとホームディレクトリ用のボリュームグループを別々に用意している場合は各ボリュームが以下のように出力されます。

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

3. **/etc/lvm/lvm.conf** 設定ファイルの **volume_list** のエントリーとして **my_vg** (クラスター用として定義したボリュームグループ) 以外のボリュームグループを追加します。例えば、root ディレクトリ用のボリュームグループ、ホームディレクトリ用のボリュームグループを別々に用意している場合は、**lvm.conf** ファイルの **volume_list** の行のコメントを外して以下のように root ディレクトリ用、ホームディレクトリ用の各ボリュームグループを **volume_list** のエントリーとして追加します。

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

注記

クラスターマネージャーの管轄外で作動させるローカルボリュームグループがノードにない場合でも **volume_list** のエントリーは **volume_list = []** と指定して初期化する必要があります。

4. 起動イメージがクラスターで制御しているボリュームグループを作動させないよう **initramfs** 起動イメージを再構築します。次のコマンドで **initramfs** デバイスを更新します。このコマンドは完了に 1 分ほどかかる場合があります。

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

5. ノードを再起動します。

注記

起動イメージを作成したノードに新しい Linux カーネルをインストールした場合、**initrd** イメージはそれを作成したときに実行していたカーネル用であってノードを再起動したら実行される新しいカーネル用ではありません。再起動の前後で **uname -r** コマンドを使って実行しているカーネルリリースを確認し必ず正しい **initrd** デバイスを使用するよう注意してください。リリースが異なる場合は新しいカーネルで再起動した後、**initrd** ファイルを更新しノードをもう一度再起動します。

6. ノードが再起動したら **pcs cluster status** コマンドを実行し、クラスターサービスがそのノードで再度開始されたかどうかを確認します。**Error: cluster is not currently running on this node** というメッセージが出力される場合は次のコマンドを実行します。

```
# pcs cluster start
```

または、クラスター内の各ノードの再起動が完了するのを待ってから次のコマンドで各ノードでのクラスターサービスの起動を行います。

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

2.4. pcs コマンドを使用したリソースおよびリソースグループの作成

この事例の場合、クラスターリソースを 4 つ作成する必要があります。すべてのリソースが必ず同じノードで実行されるよう **apachegroup** というリソースグループの一部として構成させます。作成するリソースを起動する順序で以下に示します。

1. 「LVM ボリュームを ext4 ファイルシステムで設定」 の手順で作成した LVM ボリュームグループを使用する、**my_lvm** という名前の **LVM** リソース。

2. 「LVM ボリュームを ext4 ファイルシステムで設定」 の手順で作成したファイルシステムデバイス **/dev/my_vg/my_lv** を使用する、**my_fs** という名前の **Filesystem** リソース。

3. **apachegroup** リソースグループのフローティング IP アドレスである **IPaddr2** リソース。すでに物理ノードに関連付けされた IP アドレスは使用できません。**IPaddr2** リソースの NIC デバイスが指定されていない場合、クラスターノードによって使用される静的に割り当てられた IP アドレスと

同じネットワーク上にフローティング IP が存在しないと、フローティング IP アドレスを割り当てる NIC デバイスが適切に検出されません。

4. **Website** と言う名前の **apache** リソース、「Web サーバーの設定」 の手順で定義した **index.html** ファイルと Apache 設定を使用します。

次の手順で **apachegroup** リソースグループとこのグループに含ませるリソースを作成します。リソースはグループに追加した順序で起動し、またその逆順で停止します。次の手順はクラスター内いずれか一つのノードだけで行います。

1. 次のコマンドでは **my_lvm** LVM リソースを作成しています。LVM 論理ボリュームの作動がクラスター以外では行えないよう **exclusive=true** パラメーターを指定しています。この時点で **apachegroup** リソースグループはまだ存在していないため、このコマンドにより作成されることになります。

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

リソースを作成するとそのリソースは自動的に起動されます。次のコマンドを使ってリソースが確かに作成、起動されたことを確認します。

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

pcs resource disable と **pcs resource enable** のコマンドを使用すると手作業によるリソースの停止と起動をリソースごと個別に行うことができます。

2. 次のコマンドでは構成に必要な残りのリソースを作成し、**apachegroup** リソースグループに追加しています。

```
[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. リソースおよびそのリソースを含ませるリソースグループの作成が完了したらクラスターの状態を確認します。4 つのリソースすべてが同じノードで実行していることを確認してください。

```
[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
```

「排他処理の設定」 の手順でクラスターにフェンスデバイスを設定していないとリソースはデフォルトでは起動しないので注意してください。

4. クラスターが起動し稼働し始めたら、 ブラウザで **IPaddr2** リソースとして定義した IP アドレスをポイントし "Hello" のテキストで構成されるサンプル表示が正しく表示されるか確認します。

```
Hello
```

設定したリソースが実行していない場合には **pcs resource debug-start *resource*** コマンドを実行してリソースの設定をテストしてみます。**pcs resource debug-start** コマンドの詳細については 『High Availability Add-On Reference (High Availability Add-On リファレンス)』 のガイドを参照してください。

2.5. リソース設定のテスト

「pcs コマンドを使用したリソースおよびリソースグループの作成」 で示すようにクラスターの状態表示では全リソースが **z1.example.com** ノードで実行しています。次の手順に従い 1 番目のノードを **standby** モードにしてリソースグループが **z2.example.com** ノードにフェールオーバーするかどうかテストします。1 番目のノードをスタンバイモードにするとこのノードではリソースをホストできなくなります。

1. 次のコマンドでは **z1.example.com** を **standby** モードにしています。

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

2. **z1** をスタンバイモードにしたらクラスターの状態を確認します。リソースがすべて **z2** で実行しているはずです。

```
[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
```

定義している IP アドレスの web サイトは中断されることなく表示されているはずです。

3. **standby** モードから **z1** を削除するには、以下のコマンドを実行します。

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

注記

ノードを **standby** モードから削除しても、リソースがそのノードにフェイルバックされることはありません。実行できるノードリソースの制御に関する詳細は、『Red Hat High Availability Add-On Reference リファレンス』 のクラスターリソースの設定に関する章を参照してください。

第3章 Red Hat High Availability クラスターのアクティブ/パッシブな NFS サーバー

本章では、共有ストレージを使用して 2 ノードの Red Hat Enterprise Linux High Availability Add-On クラスターで高可用性アクティブ/パッシブ NFS サーバーを設定する方法について説明します。手順では **pcs** コマンドを使用して Pacemaker クラスターリソースを設定します。このユースケースでは、クライアントはフローティング IP アドレスより NFS ファイルシステムにアクセスします。NFS サービスは、クラスターの 2 つのノードの 1 つで実行されます。NFS サーバーが稼働しているノードが正常に動作しなくなった場合、NFS サーバーはクラスターの 2 つ目のノードで再起動され、サービスの中断は最小限になります。

このユースケースでは、システムに以下のコンポーネントが必要になります。

- Apache HTTP サーバーを実行するクラスターを作成するために使用される 2 つのノード。この例では、**z1.example.com** と **z2.example.com** の 2 つのノードが使用されます。
- 各ノード用の電源フェンスデバイス、ここでは APC 電源スイッチの 2 ポートを使用しています。ホスト名は **zapc.example.com** にしています。
- NFS サーバーに必要なパブリック仮想 IP アドレス。
- クラスター内のノードで使用する共有ストレージ、iSCSI または Fibre チャンネルを使用します。

2 ノード Red Hat Enterprise Linux で高可用性アクティブ/パッシブ NFS サーバーを設定するには、以下の手順を実行する必要があります。

1. 「NFS クラスターの作成」 の説明に従って、NFS サーバーを実行するクラスターを作成し、クラスターの各ノードにフェンシングを設定します。
2. 「LVM ボリュームを ext4 ファイルシステムで設定」 の説明に従って、クラスターのノードに対する共有ストレージの LVM 論理ボリューム **my_lv** にマウントされた **ext4** ファイルシステムを設定します。
3. 「NFS 共有の設定」 の説明に従って、LVM 論理ボリュームの共有ストレージで NFS 共有を設定します。
4. 「ボリュームグループのアクティブ化をクラスター内に限定」 の説明に従って、論理ボリューム **my_lv** が含まれる LVM ボリュームグループをクラスターのみがアクティブ化できるようにし、ボリュームグループが起動時にクラスターの外部でアクティブ化されないようにします。
5. 「クラスターリソースの設定」 の説明に従って、クラスターリソースを作成します。
6. 「リソース設定のテスト」 に従って、設定した NFS サーバーをテストします。

3.1. NFS クラスターの作成

以下の手順に従って、NFS クラスターをインストールおよび作成します。

1. 「クラスターソフトウェアのインストール」 の手順に従って、**z1.example.com** および **z2.example.com** ノードにクラスターソフトウェアをインストールします。
2. 「クラスターの作成」 の手順に従って、**z1.example.com** と **z2.example.com** で構成される 2 ノードクラスターを作成します。この手順の例と同様に、クラスターには **my_cluster** という名前が付けられます。

3. 「排他処理の設定」 の説明に従って、クラスターの各ノードにフェンスデバイスを設定します。この例では、ホスト名が **zapc.example.com** という APC 電源スイッチの 2 つのポートを使用してフェンシングが設定されます。

3.2. LVM ボリュームを ext4 ファイルシステムで設定

このユースケースでは、クラスターのノード間で共有されるストレージに LVM 論理ボリュームを作成する必要があります。

次の手順に従い LVM 論理ボリュームを作成しその論理ボリューム上に **ext4** ファイルシステムを作成します。ここでは **/dev/sdb1** 共有パーティションを使って LVM 論理ボリュームの作成元となる LVM 物理ボリュームを格納します。

注記

LVM ボリューム、該当パーティション、クラスターノードで使用するデバイスなどはクラスターノード以外には接続しないでください。

/dev/sdb1 パーティションは共有されるストレージであるため、この手順は 1 つのノードでのみ実行します。

1. LVM 物理ボリュームを **/dev/sdb1** パーティション上に作成します。

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

2. **/dev/sdb1** 物理ボリュームで構成される **my_vg** ボリュームグループを作成します。

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

3. **my_vg** ボリュームグループを使用する論理ボリュームを作成します。

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

   lvs コマンドを使用すると論理ボリュームを表示できます。

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

4. **ext4** ファイルシステムを **my_lv** 論理ボリューム上に作成します。

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

```
...
```

3.3. NFS 共有の設定

以下の手順では、NFS デーモンフェールオーバーに対して NFS 共有を設定します。この手順は、クラスターの 1 つのノードのみで行う必要があります。

1. **/nfsshare** ディレクトリーを作成します。

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

2. 「LVM ボリュームを ext4 ファイルシステムで設定」 で作成した **ext4** ファイルシステムを **/nfsshare** ディレクトリーにマウントします。

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

3. **/nfsshare** ディレクトリー内に **exports** ディレクトリーツリーを作成します。

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

4. NFS クライアントがアクセスするファイルを **exports** ディレクトリーに置きます。この例では、**clientdatafile1** および **clientdatafile2** という名前のテストファイルを作成します。

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

5. ext4 ファイルをアンマウントし、LVM ボリュームグループを非アクティブ化します。

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

3.4. ボリュームグループのアクティブ化をクラスター内に限定

次の手順では、LVM ボリュームグループを設定して、クラスターのみがボリュームグループをアクティブ化でき、ボリュームグループが起動時にクラスターの外部でアクティブ化されないようにします。ボリュームグループがクラスター外部のシステムによってアクティブ化されると、ボリュームグループのメタデータが破損することがあります。

この手順では **/etc/lvm/lvm.conf** 設定ファイル内の **volume_list** のエントリーを編集します。**volume_list** のエントリーに記載されているボリュームグループはクラスターマネージャーの管轄外となるローカルノードでの自動作動が許可されます。ノードのローカルな root ディレクトリやホームディレクトリに関連するボリュームグループはこのリストに含めてください。クラスターマネージャーで管理するボリュームグループは **volume_list** のエントリーには入れないでください。ここでの手順に **clvmd** を使用する必要はありません。

クラスター内の各ノードで以下の手順を行います。

1. 次のコマンドを実行して、**/etc/lvm/lvm.conf** ファイルで **locking_type** が 1 に設定されていることと **use_lvmetad** が 0 に設定されていることを確認します。また、このコマンドを実行すると、すべての **lvmetad** プロセスがすぐに無効になり、停止します。

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

2. 次のコマンドでローカルストレージに現在設定されているボリュームグループを確認します。現在設定されているボリュームグループ一覧が出力されます。このノード上に root ディレクトリ用のボリュームグループとホームディレクトリ用のボリュームグループを別々に用意している場合は各ボリュームが以下のように出力されます。

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

3. **/etc/lvm/lvm.conf** 設定ファイルの **volume_list** のエントリーとして **my_vg** (クラスター用として定義したボリュームグループ) 以外のボリュームグループを追加します。例えば、root ディレクトリ用のボリュームグループ、ホームディレクトリ用のボリュームグループを別々に用意している場合は、**lvm.conf** ファイルの **volume_list** の行のコメントを外して以下のように root ディレクトリ用、ホームディレクトリ用の各ボリュームグループを **volume_list** のエントリーとして追加します。

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

注記

クラスターマネージャーの管轄外で作動させるローカルボリュームグループがノードにない場合でも **volume_list** のエントリーは **volume_list = []** と指定して初期化する必要があります。

4. 起動イメージがクラスターで制御しているボリュームグループを作動させないよう **initramfs** 起動イメージを再構築します。次のコマンドで **initramfs** デバイスを更新します。このコマンドは完了に 1 分ほどかかる場合があります。

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

5. ノードを再起動します。

注記

起動イメージを作成したノードに新しい Linux カーネルをインストールした場合、**initrd** イメージはそれを作成したときに実行していたカーネル用であってノードを再起動したら実行される新しいカーネル用ではありません。再起動の前後で **uname -r** コマンドを使って実行しているカーネルリリースを確認し必ず正しい **initrd** デバイスを使用するよう注意してください。リリースが異なる場合は新しいカーネルで再起動した後、**initrd** ファイルを更新しノードをもう一度再起動します。

6. ノードが再起動したら **pcs cluster status** コマンドを実行し、クラスターサービスがそのノードで再度開始されたかどうかを確認します。**Error: cluster is not currently running on this node** というメッセージが出力される場合は次のコマンドを実行します。

```
# pcs cluster start
```

この代わりに、クラスターの各ノードを再起動し、再起動が完了してから次のコマンドを実行してクラスターのすべてのノードでクラスターサービスを開始することもできます。

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

3.5. クラスターリソースの設定

このユースケースでクラスターリソースを設定する手順について説明します。

注記

pcs resource create コマンドを使用してクラスターリソースを作成する場合、作成直後に **pcs status** コマンドを実行してリソースが稼働していることを検証することが推奨されます。「排他処理の設定」 の説明に従ってクラスターのフェンスデバイスを設定していない場合、デフォルトではリソースが起動しません。

設定したリソースが実行されていない場合は、**pcs resource debug-start *resource*** コマンドを実行してリソースの設定をテストできます。このコマンドは、クラスターの制御や認識の範囲外でサービスを起動します。設定したリソースが再度実行されたら、**pcs cluster cleanup *resource*** コマンドを実行してクラスターが更新を認識するようにします。**pcs resource debug-start** コマンドの詳細は 『High Availability Add-On リファレンス』 を参照してください。

以下の手順では、システムリソースを設定します。これらのリソースがすべて同じノードで実行されるようにするため、これらのリソースはリソースグループ **nfsgroup** の一部として設定されます。リソースは、グループに追加された順序で起動し、その逆の順序で停止します。この手順は、クラスターの 1 つのノードのみで実行してください。

1. 以下のコマンドは **my_lvm** という名前の LVM リソースを作成します。このコマンドは、**exclusive=true** パラメーターを指定し、クラスターのみが LVM 論理ボリュームをアクティブ化できるようにします。この時点ではリソースグループ **nfsgroup** は存在しないため、このコマンドによって作成されます。

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

   クラスターの状態を確認し、リソースが実行されていることを確認します。

   ```
   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. クラスターの **Filesystem** リソースを設定します。

注記

options=*options* パラメーターを使用すると、**Filesystem** リソースのリソース設定の一部としてマウントオプションを指定できます。完全な設定オプションを表示するには、**pcs resource describe Filesystem** コマンドを実行します。

以下のコマンドは、**nfsshare** という名前の ext4 **Filesystem** リソースを **nfsgroup** リソースグループの一部として設定します。このファイルシステムは、「LVM ボリュームを ext4 ファイルシステムで設定」で作成された LVM ボリュームグループと ext4 ファイルシステムを使用します。このファイルシステムは「NFS 共有の設定」で作成された **/nfsshare** ディレクトリーにマウントされます。

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

my_lvm および **nfsshare** リソースが実行されていることを確認します。

```
[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. **nfsgroup** リソースグループの一部である **nfs-daemon** という名前の **nfsserver** リソースを作成します。

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

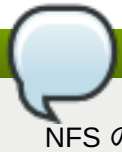
4. **exportfs** リソースを追加して **/nfsshare/exports** ディレクトリーをエクスポートします。これらのリソースは **nfsgroup** リソースグループの一部です。これにより、NFSv4 クライアントの仮想ディレクトリーが構築されます。NFSv3 クライアントもこれらのエクスポートにアクセスできます。

```
[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. NFS 共有にアクセスするために NFS クライアントが使用するフローティング IP アドレスリソースを追加します。指定するフローティング IP アドレスには DNS リバースルックアップが必要になりますが、クラスターのすべてのノードで **/etc/hosts** にフローティング IP アドレスを指定して対処することもできます。このリソースは **nfsgroup** リソースグループの一部です。このデプロイ例では、192.168.122.200 をフローティング IP アドレスとして使用します。

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

6. NFS デプロイメント全体が初期化されたら、NFSv3 の再起動通知を送信するため **nfsnotify** リソースを追加します。このリソースは、リソースグループ **nfsgroup** の一部です。



注記

NFS の通知が適切に処理されるようにするには、フローティング IP アドレスとホスト名が関連付けられ、NFS サーバーと NFS クライアントの両方で一貫性を保つ必要があります。

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

リソースとリソースの制約を作成したら、クラスターの状態をチェックできます。すべてのリソースは同じノードで実行されていることに注意してください。

```
[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. リソース設定のテスト

以下の手順を使用するとシステムの設定を検証できます。NFSv3 または NFSv4 のいずれかでエクスポートされたファイルシステムをマウントできるはずです。

1. デプロイメントと同じネットワークにあるクラスター外部のノードで、NFS 共有をマウントすると NFS 共有が表示されることを確認します。この例では、192.168.122.0/24 ネットワークを使用します。

```
# 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. NFSv4 で NFS 共有をマウントできることを確認するには、NFS 共有をクライアントノード上のディレクトリーにマウントします。マウントした後、エクスポートディレクトリーの内容が表示されることを確認します。テスト後に共有をアンマウントします。

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

3. NFSv3 で NFS 共有をマウントできることを確認します。マウント後、テストファイル **clientdatafile1** が表示されることを確認します。NFSv4 とは異なり NFSv3 は仮想ファイルシステムを使用しないため、特定のエクスポートをマウントする必要があります。テスト後に共有をアンマウントします。

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

4. フェイルオーバーをテストするには、以下の手順を実行します。

    a. クラスター外部のノードに NFS 共有をマウントし、「NFS 共有の設定」 で作成した **clientdatafile1** にアクセスできることを確認します。

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

b. クラスター内のノードより、**nfsgroup** を実行しているノードを判断します。この例では、**nfsgroup** は **z1.example.com** で実行されています。

```
[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. クラスター内のノードより、**nfsgroup** を実行しているノードをスタンバイモードにします。

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

d. **nfsgroup** が別のクラスターノードで正常に起動することを確認します。

```
[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. NFS 共有をマウントしたクラスターの外部のノードより、この外部ノードが NFS マウント

内のテストファイルにアクセスできることを確認します。

```
# ls nfsshare
clientdatafile1
```

ファイルオーバー中、クライアントに対するサービスは一時的に失われますが、クライアントはユーザーが介入しなくても回復するはずです。デフォルトでは、NFSv4 を使用するクライアントはマウントのリカバリーに最大 90 秒かかることがあります。この 90 秒は、起動時にサーバーによって確認される NFSv4 ファイルのリースの猶予期間です。NFSv3 クライアントでは、数秒でマウントへのアクセスが回復するはずです。

f. クラスター内のノードより、最初に **nfsgroup** を実行していたノードをスタンバイノードから削除します。これだけでは、クラスターリソースはこのノードに戻されません。

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

付録A 改訂履歴

改訂 2-4.4	**Mon Jul 3 2017**	**Junko Ito**
翻訳完了		
改訂 2-4.3	**Wed Mar 22 2017**	**Terry Chuang**
翻訳ファイルを XML ソースバージョン 2-4 と同期		
改訂 2-4.2	**Wed Mar 1 2017**	**Terry Chuang**
翻訳ファイルを XML ソースバージョン 2-4 と同期		
改訂 2-4.1	**Mon Nov 28 2016**	**Terry Chuang**
翻訳ファイルを XML ソースバージョン 2-4 と同期		
改訂 2-4	**Mon Oct 17 2016**	**Steven Levine**
7.3 GA 公開用バージョン		
改訂 2-3	**Fri Aug 12 2016**	**Steven Levine**
7.3 ベータ公開用ドキュメントの準備		
改訂 1.2-3	**Mon Nov 9 2015**	**Steven Levine**
7.2 GA 公開用ドキュメントの準備。		
改訂 1.2-2	**Tue Aug 18 2015**	**Steven Levine**
7.2 ベータ公開用ドキュメントの準備 。		
改訂 1.1-19	**Mon Feb 16 2015**	**Steven Levine**
7.1 GA リリース向けのバージョン		
改訂 1.1-10	**Thu Dec 11 2014**	**Steven Levine**
7.1 ベータリリース向けのバージョン		
改訂 0.1-33	**Mon Jun 2 2014**	**Steven Levine**
7.0 GA リリース向けバージョン		