Red Hat Enterprise Linux 8

Configuring and managing high availability clusters

A guide to the configuration and management of the Red Hat High Availability Add-On

Last Updated: 2019-05-08

Red Hat Enterprise Linux 8 Configuring and managing high availability clusters

A guide to the configuration and management of the Red Hat High Availability Add-On

Legal Notice

Abstract

This guide provides information about installing, configuring, and managing the Red Hat High Availability Add-On for Red Hat Enterprise Linux 8.

Table of Contents

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate your input on our documentation. Please let us know how we could make it better. To do so:

- For simple comments on specific passages, make sure you are viewing the documentation in the Multi-page HTML format. Highlight the part of text that you want to comment on. Then, click the **Add Feedback** pop-up that appears below the highlighted text, and follow the displayed instructions.
- For submitting more complex feedback, create a Bugzilla ticket:
  1. Go to the Bugzilla website.
  2. As the Component, use **Documentation**.
  3. Fill in the **Description** field with your suggestion for improvement. Include a link to the relevant part(s) of documentation.
  4. Click **Submit Bug**.

CHAPTER 1. HIGH AVAILABILITY ADD-ON OVERVIEW

The High Availability Add-On is a clustered system that provides reliability, scalability, and availability to critical production services.

A cluster is two or more computers (called *nodes* or *members*) that work together to perform a task. Clusters can be used to provide highly available services or resources. The redundancy of multiple machines is used to guard against failures of many types.

High availability clusters provide highly available services by eliminating single points of failure and by failing over services from one cluster node to another in case a node becomes inoperative. Typically, services in a high availability cluster read and write data (by means of read-write mounted file systems). Therefore, a high availability cluster must maintain data integrity as one cluster node takes over control of a service from another cluster node. Node failures in a high availability cluster are not visible from clients outside the cluster. (High availability clusters are sometimes referred to as failover clusters.) The High Availability Add-On provides high availability clustering through its high availability service management component, **Pacemaker**.

1.1. HIGH AVAILABILITY ADD-ON COMPONENTS

The High Availability Add-On consists of the following major components:

- Cluster infrastructure — Provides fundamental functions for nodes to work together as a cluster: configuration file management, membership management, lock management, and fencing.
- High availability service management — Provides failover of services from one cluster node to another in case a node becomes inoperative.
- Cluster administration tools — Configuration and management tools for setting up, configuring, and managing the High Availability Add-On. The tools are for use with the cluster infrastructure components, the high availability and service management components, and storage.

You can supplement the High Availability Add-On with the following components:

- Red Hat GFS2 (Global File System 2) — Part of the Resilient Storage Add-On, this provides a cluster file system for use with the High Availability Add-On. GFS2 allows multiple nodes to share storage at a block level as if the storage were connected locally to each cluster node. GFS2 cluster file system requires a cluster infrastructure.
- LVM Locking Daemon (**lvmlockd**) — Part of the Resilient Storage Add-On, this provides volume management of cluster storage. **lvmlockd** support also requires cluster infrastructure.
- Load Balancer Add-On — Routing software that provides high availability load balancing and failover in layer 4 (TCP) and layer 7 (HTTP, HTTPS) services. The Load Balancer Add-On runs in a cluster of redundant virtual routers that uses load algorithms to distribute client requests to real servers, collectively acting as a virtual server. It is not necessary to use the Load Balancer Add-On in conjunction with Pacemaker.

1.2. PACEMAKER OVERVIEW

Pacemaker is a cluster resource manager. It achieves maximum availability for your cluster services and resources by making use of the cluster infrastructure's messaging and membership capabilities to deter and recover from node and resource-level failure.

1.2.1. Pacemaker architecture components

A cluster configured with Pacemaker comprises separate component daemons that monitor cluster membership, scripts that manage the services, and resource management subsystems that monitor the disparate resources.

The following components form the Pacemaker architecture:

Cluster Information Base (CIB)

The Pacemaker information daemon, which uses XML internally to distribute and synchronize current configuration and status information from the Designated Coordinator (DC) – a node assigned by Pacemaker to store and distribute cluster state and actions by means of the CIB – to all other cluster nodes.

Cluster Resource Management Daemon (CRMd)

Pacemaker cluster resource actions are routed through this daemon. Resources managed by CRMd can be queried by client systems, moved, instantiated, and changed when needed.
Each cluster node also includes a local resource manager daemon (LRMd) that acts as an interface between CRMd and resources. LRMd passes commands from CRMd to agents, such as starting and stopping and relaying status information.

Shoot the Other Node in the Head (STONITH)

STONITH is the Pacemaker fencing implementation. It acts as a cluster resource in Pacemaker that processes fence requests, forcefully powering down nodes and removing them from the cluster to ensure data integrity. STONITH is configured in the CIB and can be monitored as a normal cluster resource. For a general overview of fencing, see Section 1.3, “Fencing overview”.

corosync

corosync is the component - and a daemon of the same name - that serves the core membership and member-communication needs for high availability clusters. It is required for the High Availability Add-On to function.
In addition to those membership and messaging functions, **corosync** also:

- Manages quorum rules and determination.
- Provides messaging capabilities for applications that coordinate or operate across multiple members of the cluster and thus must communicate stateful or other information between instances.
- Uses the **kronosnet** library as its network transport to provide multiple redundant links and automatic failover.

1.2.2. Configuration and management tools

The High Availability Add-On features two configuration tools for cluster deployment, monitoring, and management.

pcs

The **pcs** command line interface controls and configures Pacemaker and the **corosync** heartbeat daemon. A command-line based program, **pcs** can perform the following cluster management tasks:

- Create and configure a Pacemaker/Corosync cluster
- Modify configuration of the cluster while it is running
- Remotely configure both Pacemaker and Corosync as well as start, stop, and display status information of the cluster

pcsd Web UI

A graphical user interface to create and configure Pacemaker/Corosync clusters.

1.2.3. The cluster and pacemaker configuration files

The configuration files for the Red Hat High Availability Add-On are **corosync.conf** and **cib.xml**.

The **corosync.conf** file provides the cluster parameters used by **corosync**, the cluster manager that Pacemaker is built on. In general, you should not edit the **corosync.conf** directly but, instead, use the **pcs** or **pcsd** interface. However, there may be a situation where you do need to edit this file directly.

The **cib.xml** file is an XML file that represents both the cluster's configuration and the current state of all resources in the cluster. This file is used by Pacemaker's Cluster Information Base (CIB). The contents of the CIB are automatically kept in sync across the entire cluster. Do not edit the **cib.xml** file directly; use the **pcs** or **pcsd** interface instead.

1.3. FENCING OVERVIEW

If communication with a single node in the cluster fails, then other nodes in the cluster must be able to restrict or release access to resources that the failed cluster node may have access to. This cannot be accomplished by contacting the cluster node itself as the cluster node may not be responsive. Instead, you must provide an external method, which is called fencing with a fence agent. A fence device is an external device that can be used by the cluster to restrict access to shared resources by an errant node, or to issue a hard reboot on the cluster node.

Without a fence device configured you do not have a way to know that the resources previously used by the disconnected cluster node have been released, and this could prevent the services from running on any of the other cluster nodes. Conversely, the system may assume erroneously that the cluster node has released its resources and this can lead to data corruption and data loss. Without a fence device configured data integrity cannot be guaranteed and the cluster configuration will be unsupported.

When the fencing is in progress no other cluster operation is allowed to run. Normal operation of the cluster cannot resume until fencing has completed or the cluster node rejoins the cluster after the cluster node has been rebooted.

For more information about fencing, see Fencing in a Red Hat High Availability Cluster.

1.4. QUORUM OVERVIEW

In order to maintain cluster integrity and availability, cluster systems use a concept known as *quorum* to prevent data corruption and loss. A cluster has quorum when more than half of the cluster nodes are online. To mitigate the chance of data corruption due to failure, Pacemaker by default stops all resources if the cluster does not have quorum.

Quorum is established using a voting system. When a cluster node does not function as it should or loses communication with the rest of the cluster, the majority working nodes can vote to isolate and, if needed, fence the node for servicing.

For example, in a 6-node cluster, quorum is established when at least 4 cluster nodes are functioning. If the majority of nodes go offline or become unavailable, the cluster no longer has quorum and Pacemaker stops clustered services.

The quorum features in Pacemaker prevent what is also known as *split-brain*, a phenomenon where the cluster is separated from communication but each part continues working as separate clusters, potentially writing to the same data and possibly causing corruption or loss. For more information on

what it means to be in a split-brain state, and on quorum concepts in general, see Exploring Concepts of RHEL High Availability Clusters - Quorum.

A Red Hat Enterprise Linux High Availability Add-On cluster uses the **votequorum** service, in conjunction with fencing, to avoid split brain situations. A number of votes is assigned to each system in the cluster, and cluster operations are allowed to proceed only when a majority of votes is present.

1.5. RESOURCE OVERVIEW

A *cluster resource* is an instance of program, data, or application to be managed by the cluster service. These resources are abstracted by *agents* that provide a standard interface for managing the resource in a cluster environment.

To ensure that resources remain healthy, you can add a monitoring operation to a resource's definition. If you do not specify a monitoring operation for a resource, one is added by default.

You can determine the behavior of a resource in a cluster by configuring *constraints*. You can configure the following categories of constraints:

- location constraints — A location constraint determines which nodes a resource can run on.
- ordering constraints — An ordering constraint determines the order in which the resources run.
- colocation constraints — A colocation constraint determines where resources will be placed relative to other resources.

One of the most common elements of a cluster is a set of resources that need to be located together, start sequentially, and stop in the reverse order. To simplify this configuration, Pacemaker supports the concept of *groups*.

/ Module included in the following assemblies:

1.6. LVM LOGICAL VOLUMES IN A RED HAT HIGH AVAILABILITY CLUSTER

The Red Hat High Availability Add-On provides support for LVM volumes in two distinct cluster configurations:

- High availability LVM volumes (HA-LVM) in active/passive failover configurations in which only a single node of the cluster accesses the storage at any one time.
- LVM volumes that use the **lvmlockd** daemon to manage storage devices in active/active configurations in which more than one node of the cluster requires access to the storage at the same time. The **lvmlockd** daemon is part of the Resilient Storage Add-On.

1.6.1. Choosing HA-LVM or shared volumes

When to use HA-LVM or shared logical volumes managed by the **lvmlockd** daemon should be based on the needs of the applications or services being deployed.

- If multiple nodes of the cluster require simultaneous read/write access to LVM volumes in an active/active system, then you must use the **lvmlockd** daemon and configure your volumes as shared volumes. The **lvmlockd** daemon provides a system for coordinating activation of and changes to LVM volumes across nodes of a cluster concurrently. The **lvmlockd** daemon's locking service provides protection to LVM metadata as various nodes of the cluster interact

with volumes and make changes to their layout. This protection is contingent upon configuring any volume group that will be activated simultaneously across multiple cluster nodes as a shared volume.

- If the high availability cluster is configured to manage shared resources in an active/passive manner with only one single member needing access to a given LVM volume at a time, then you can use HA-LVM without the **lvmlockd** locking service.

Most applications will run better in an active/passive configuration, as they are not designed or optimized to run concurrently with other instances. Choosing to run an application that is not cluster-aware on shared logical volumes may result in degraded performance. This is because there is cluster communication overhead for the logical volumes themselves in these instances. A cluster-aware application must be able to achieve performance gains above the performance losses introduced by cluster file systems and cluster-aware logical volumes. This is achievable for some applications and workloads more easily than others. Determining what the requirements of the cluster are and whether the extra effort toward optimizing for an active/active cluster will pay dividends is the way to choose between the two LVM variants. Most users will achieve the best HA results from using HA-LVM.

HA-LVM and shared logical volumes using **lvmlockd** are similar in the fact that they prevent corruption of LVM metadata and its logical volumes, which could otherwise occur if multiple machines are allowed to make overlapping changes. HA-LVM imposes the restriction that a logical volume can only be activated exclusively; that is, active on only one machine at a time. This means that only local (non-clustered) implementations of the storage drivers are used. Avoiding the cluster coordination overhead in this way increases performance. A shared volume using **lvmlockd** does not impose these restrictions and a user is free to activate a logical volume on all machines in a cluster; this forces the use of cluster-aware storage drivers, which allow for cluster-aware file systems and applications to be put on top.

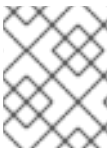
1.6.2. Configuring LVM volumes in a cluster

In Red Hat Enterprise Linux 8, clusters are managed through Pacemaker. Both HA-LVM and shared logical volumes are supported only in conjunction with Pacemaker clusters, and must be configured as cluster resources.

- For examples of procedures for configuring an HA-LVM volume as part of a Pacemaker cluster, see Configuring an active/passive Apache HTTP server in a Red Hat High Availability cluster . and Configuring an active/passive NFS server in a Red Hat High Availability cluster . Note that these procedures include the following steps:
  - Ensuring that only the cluster is capable of activating the volume group
  - Configuring an LVM logical volume
  - Configuring the LVM volume as a cluster resource
- For a procedure for configuring shared LVM volumes that use the **lvmlockd** daemon to manage storage devices in active/active configurations, see Configuring a GFS2 file system in a cluster

CHAPTER 2. GETTING STARTED WITH PACEMAKER

The following procedures provide an introduction to the tools and processes you use to create a Pacemaker cluster. They are intended for users who are interested in seeing what the cluster software looks like and how it is administered, without needing to configure a working cluster.



NOTE

These procedures do not create a supported Red Hat cluster, which requires at least two nodes and the configuration of a fencing device.

2.1. LEARNING TO USE PACEMAKER

This example requires a single node running RHEL 8 and it requires a floating IP address that resides on the same network as one of the node's statically assigned IP addresses.

- The node used in this example is **z1.example.com**.
- The floating IP address used in this example is 192.168.122.120.

NOTE

Ensure that the name of the node on which you are running is in your **/etc/hosts** file.

By working through this procedure, you will learn how to use Pacemaker to set up a cluster, how to display cluster status, and how to configure a cluster service. This example creates an Apache HTTP server as a cluster resource and shows how the cluster responds when the resource fails.

1. Install the Red Hat High Availability Add-On software packages from the High Availability channel, and start and enable the **pcsd** service.

```
# yum install pcs pacemaker fence-agents-all
...
# systemctl start pcsd.service
# systemctl enable pcsd.service
```

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

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

2. Set a password for user **hacluster** on each node in the cluster and authenticate user **hacluster** for each node in the cluster on the node from which you will be running the **pcs** commands. This example is using only a single node, the node from which you are running the commands, but this step is included here since it is a necessary step in configuring a supported Red Hat High Availability multi-node cluster.

```
# passwd hacluster
...
# pcs host auth z1.example.com
```

3. Create a cluster named **my_cluster** with one member and check the status of the cluster. This command creates and starts the cluster in one step.

```
# pcs cluster setup my_cluster --start z1.example.com
...
# pcs cluster status
Cluster Status:
 Stack: corosync
 Current DC: z1.example.com (version 2.0.0-10.el8-b67d8d0de9) - partition with quorum
 Last updated: Thu Oct 11 16:11:18 2018
 Last change: Thu Oct 11 16:11:00 2018 by hacluster via crmd on z1.example.com
 1 node configured
 0 resources configured

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

4. A Red Hat High Availability cluster requires that you configure fencing for the cluster. The reasons for this requirement are described in Fencing in a Red Hat High Availability Cluster . For this introduction, however, which is intended to show only how to use the basic Pacemaker commands, disable fencing by setting the **stonith-enabled** cluster option to **false**.

WARNING

The use of **stonith-enabled=false** is completely inappropriate for a production cluster. It tells the cluster to simply pretend that failed nodes are safely fenced.

```
# pcs property set stonith-enabled=false
```

5. Configure a web browser on your system and create a web page to display a simple text message. If you are running the **firewalld** daemon, enable the ports that are required by **httpd**.

NOTE

Do not use **systemctl enable** to enable any services that will be managed by the cluster to start at system boot.

```
# yum install -y httpd wget
...
# firewall-cmd --permanent --add-service=http
# firewall-cmd --reload

# cat <<-END >/var/www/html/index.html
<html>
<body>My Test Site - $(hostname)</body>
</html>
END
```

In order for the Apache resource agent to get the status of Apache, create the following addition to the existing configuration to enable the status server URL.

```
# cat <<-END > /etc/httpd/conf.d/status.conf
<Location /server-status>
SetHandler server-status
Order deny,allow
Deny from all
Allow from 127.0.0.1
Allow from ::1
</Location>
END
```

6. Create **IPaddr2** and **apache** resources for the cluster to manage. The 'IPaddr2' resource is a floating IP address that 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 address used by the node.
   You can display a list of all available resource types with the **pcs resource list** command. You can use the **pcs resource describe *resourcetype*** command to display the parameters you can set for the specified resource type. For example, the following command displays the parameters you can set for a resource of type **apache**:

   ```
   # pcs resource describe apache
   ...
   ```

   In this example, the IP address resource and the apache resource are both configured as part of a group named **apachegroup**, which ensures that the resources are kept together to run on the same node when you are configuring a working multi-node cluster.

   ```
   # pcs resource create ClusterIP ocf:heartbeat:IPaddr2 ip=192.168.122.120 --group
   apachegroup

   # pcs resource create WebSite ocf:heartbeat:apache
   configfile=/etc/httpd/conf/httpd.conf statusurl="http://localhost/server-status" --group
   apachegroup

   # pcs status
   Cluster name: my_cluster
   Stack: corosync
   Current DC: z1.example.com (version 2.0.0-10.el8-b67d8d0de9) - partition with quorum
   Last updated: Fri Oct 12 09:54:33 2018
   Last change: Fri Oct 12 09:54:30 2018 by root via cibadmin on z1.example.com

   1 node configured
   2 resources configured

   Online: [ z1.example.com ]

   Full list of resources:

   Resource Group: apachegroup
      ClusterIP  (ocf::heartbeat:IPaddr2):       Started z1.example.com
      WebSite    (ocf::heartbeat:apache):        Started z1.example.com
   ```

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

After you have configured a cluster resource, you can use the **pcs resource config** command to display the options that are configured for that resource.

```
# pcs resource config WebSite
Resource: WebSite (class=ocf provider=heartbeat type=apache)
 Attributes: configfile=/etc/httpd/conf/httpd.conf statusurl=http://localhost/server-status
 Operations: start interval=0s timeout=40s (WebSite-start-interval-0s)
             stop interval=0s timeout=60s (WebSite-stop-interval-0s)
             monitor interval=1min (WebSite-monitor-interval-1min)
```

7. Point your browser to the website you created using the floating IP address you configured. This should display the text message you defined.

8. Stop the apache web service and check the cluster status. Using **killall -9** simulates an application-level crash.

```
# killall -9 httpd
```

Check the cluster status. You should see that stopping the web service caused a failed action, but that the cluster software restarted the service and you should still be able to access the website.

```
# pcs status
Cluster name: my_cluster
...
Current DC: z1.example.com (version 1.1.13-10.el7-44eb2dd) - partition with quorum
1 node and 2 resources configured

Online: [ z1.example.com ]

Full list of resources:

Resource Group: apachegroup
    ClusterIP  (ocf::heartbeat:IPaddr2):       Started z1.example.com
    WebSite    (ocf::heartbeat:apache):        Started z1.example.com

Failed Resource Actions:
* WebSite_monitor_60000 on z1.example.com 'not running' (7): call=13, status=complete,
exitreason='none',
    last-rc-change='Thu Oct 11 23:45:50 2016', queued=0ms, exec=0ms

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

You can clear the failure status on the resource that failed once the service is up and running again and the failed action notice will no longer appear when you view the cluster status.

```
# pcs resource cleanup WebSite
```

9. When you are finished looking at the cluster and the cluster status, stop the cluster services on

the node. Even though you have only started services on one node for this introduction, the **--all** parameter is included since it would stop cluster services on all nodes on an actual multi-node cluster.

```
# pcs cluster stop --all
```

2.2. LEARNING TO CONFIGURE FAILOVER

This procedure provides an introduction to creating a Pacemaker cluster running a service that will fail over from one node to another when the node on which the service is running becomes unavailable. By working through this procedure, you can learn how to create a service in a two-node cluster and you can then observe what happens to that service when it fails on the node on which it running.

This example procedure configures a two-node Pacemaker cluster running an Apache HTTP server. You can then stop the Apache service on one node to see how the service remains available.

This procedure requires as a prerequisite that you have two nodes running Red Hat Enterprise Linux 8 that can communicate with each other, and it requires a floating IP address that resides on the same network as one of the node's statically assigned IP addresses.

- The nodes used in this example are **z1.example.com** and **z2.example.com**.
- The floating IP address used in this example is 192.168.122.120.

NOTE

Ensure that the names of the nodes you are using are in the /**etc**/**hosts** file on each node.

1. On both nodes, install the Red Hat High Availability Add-On software packages from the High Availability channel, and start and enable the **pcsd** service.

```
# yum install pcs pacemaker fence-agents-all
...
# systemctl start pcsd.service
# systemctl enable pcsd.service
```

If you are running the **firewalld** daemon, on both nodes enable the ports that are required by the Red Hat High Availability Add-On.

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

2. On both nodes in the cluster, set a password for user **hacluster** .

```
# passwd hacluster
```

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

```
# pcs host auth z1.example.com z2.example.com
```

4. Create a cluster named **my_cluster** with both nodes as cluster members. This command creates and starts the cluster in one step. You only need to run this from one node in the cluster because **pcs** configuration commands take effect for the entire cluster.
On one node in cluster, run the following command.

```
# pcs cluster setup my_cluster --start z1.example.com z2.example.com
```

5. A Red Hat High Availability cluster requires that you configure fencing for the cluster. The reasons for this requirement are described in Fencing in a Red Hat High Availability Cluster . For this introduction, however, to show only how failover works in this configuration, disable fencing by setting the **stonith-enabled** cluster option to **false**

WARNING

The use of **stonith-enabled=false** is completely inappropriate for a production cluster. It tells the cluster to simply pretend that failed nodes are safely fenced.

```
# pcs property set stonith-enabled=false
```

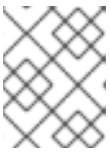
6. After creating a cluster and disabling fencing, check the status of the cluster.

NOTE

When you run the **pcs cluster status** command, it may show output that temporarily differs slightly from the examples as the system components start up.

```
# pcs cluster status
Cluster Status:
 Stack: corosync
 Current DC: z1.example.com (version 2.0.0-10.el8-b67d8d0de9) - partition with quorum
 Last updated: Thu Oct 11 16:11:18 2018
 Last change: Thu Oct 11 16:11:00 2018 by hacluster via crmd on z1.example.com
 2 nodes configured
 0 resources configured

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

7. On both nodes, configure a web browser and create a web page to display a simple text message. If you are running the **firewalld** daemon, enable the ports that are required by **httpd**.



NOTE

Do not use **systemctl enable** to enable any services that will be managed by the cluster to start at system boot.

```
# yum install -y httpd wget
...
# firewall-cmd --permanent --add-service=http
# firewall-cmd --reload

# cat <<-END >/var/www/html/index.html
<html>
<body>My Test Site - $(hostname)</body>
</html>
END
```

In order for the Apache resource agent to get the status of Apache, on each node in the cluster create the following addition to the existing configuration to enable the status server URL.

```
# cat <<-END > /etc/httpd/conf.d/status.conf
<Location /server-status>
SetHandler server-status
Order deny,allow
Deny from all
Allow from 127.0.0.1
Allow from ::1
</Location>
END
```

8. Create **IPaddr2** and **apache** resources for the cluster to manage. The 'IPaddr2' resource is a floating IP address that 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 address used by the node.
You can display a list of all available resource types with the **pcs resource list** command. You can use the **pcs resource describe *resourcetype*** command to display the parameters you can set for the specified resource type. For example, the following command displays the parameters you can set for a resource of type **apache**:

```
# pcs resource describe apache
...
```

In this example, the IP address resource and the apache resource are both configured as part of a group named **apachegroup**, which ensures that the resources are kept together to run on the same node.

Run the following commands from one node in the cluster:

```
# pcs resource create ClusterIP ocf:heartbeat:IPaddr2 ip=192.168.122.120 --group
apachegroup

# pcs resource create WebSite ocf:heartbeat:apache
configfile=/etc/httpd/conf/httpd.conf statusurl="http://localhost/server-status" --group
apachegroup

# pcs status
Cluster name: my_cluster
Stack: corosync
Current DC: z1.example.com (version 2.0.0-10.el8-b67d8d0de9) - partition with quorum
Last updated: Fri Oct 12 09:54:33 2018
```

```
Last change: Fri Oct 12 09:54:30 2018 by root via cibadmin on z1.example.com

2 nodes configured
2 resources configured

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

Full list of resources:

Resource Group: apachegroup
   ClusterIP  (ocf::heartbeat:IPaddr2):       Started z1.example.com
   WebSite    (ocf::heartbeat:apache):        Started z1.example.com

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

Note that in this instance, the **apachegroup** service is running on node z1.example.com.

9. Access the website you created, stop the service on the node on which it is running, and note how the service fails over to the second node.

   a. Point a browser to the website you created using the floating IP address you configured. This should display the text message you defined, displaying the name of the node on which the website is running.

   b. Stop the apache web service. Using **killall -9** simulates an application-level crash.

      ```
      # killall -9 httpd
      ```

      Check the cluster status. You should see that stopping the web service caused a failed action, but that the cluster software restarted the service on the node on which it had been running and you should still be able to access the web browser.

      ```
      # pcs status
      Cluster name: my_cluster
      Stack: corosync
      Current DC: z1.example.com (version 2.0.0-10.el8-b67d8d0de9) - partition with quorum
      Last updated: Fri Oct 12 09:54:33 2018
      Last change: Fri Oct 12 09:54:30 2018 by root via cibadmin on z1.example.com

      2 nodes configured
      2 resources configured

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

      Full list of resources:

      Resource Group: apachegroup
         ClusterIP  (ocf::heartbeat:IPaddr2):       Started z1.example.com
         WebSite    (ocf::heartbeat:apache):        Started z1.example.com

      Failed Resource Actions:
      ```

```
* WebSite_monitor_60000 on z1.example.com 'not running' (7): call=31,
status=complete, exitreason='none',
    last-rc-change='Fri Feb  5 21:01:41 2016', queued=0ms, exec=0ms
```

Clear the failure status once the service is up and running again.

```
# pcs resource cleanup WebSite
```

c. Put the node on which the service is running into standby mode. Note that since we have disabled fencing we can not effectively simulate a node-level failure (such as pulling a power cable) because fencing is required for the cluster to recover from such situations.

```
# pcs node standby z1.example.com
```

d. Check the status of the cluster and note where the service is now running.

```
# pcs status
Cluster name: my_cluster
Stack: corosync
Current DC: z1.example.com (version 2.0.0-10.el8-b67d8d0de9) - partition with quorum
Last updated: Fri Oct 12 09:54:33 2018
Last change: Fri Oct 12 09:54:30 2018 by root via cibadmin on z1.example.com

2 nodes configured
2 resources configured

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

Full list of resources:

Resource Group: apachegroup
    ClusterIP  (ocf::heartbeat:IPaddr2):       Started z2.example.com
    WebSite    (ocf::heartbeat:apache):        Started z2.example.com
```

e. Access the website. There should be no loss of service, although the display message should indicate the node on which the service is now running.

10. To restore cluster services to the first node, take the node out of standby mode. This will not necessarily move the service back to that node.

```
# pcs cluster unstandby z1.example.com
```

11. For final cleanup, stop the cluster services on both nodes.

```
# pcs cluster stop --all
```

CHAPTER 3. THE PCS COMMAND LINE INTERFACE

The **pcs** command line interface controls and configures cluster services such as **corosync**, **pacemaker**,**booth**, and **sbd** by providing an easier interface to their configuration files.

Note that you should not edit the **cib.xml** configuration file directly. In most cases, Pacemaker will reject a directly modified **cib.xml** file.

3.1. PCS HELP DISPLAY

You can use the **-h** option of **pcs** to display the parameters of a **pcs** command and a description of those parameters. For example, the following command displays the parameters of the **pcs resource** command. Only a portion of the output is shown.

```
# pcs resource -h
```

3.2. VIEWING THE RAW CLUSTER CONFIGURATION

Although you should not edit the cluster configuration file directly, you can view the raw cluster configuration with the **pcs cluster cib** command.

You can save the raw cluster configuration to a specified file with the **pcs cluster cib *filename*** command. If you have previously configured a cluster and there is already an active CIB, you use the following command to save the raw xml file.

```
pcs cluster cib filename
```

For example, the following command saves the raw xml from the CIB into a file named **testfile**.

```
pcs cluster cib testfile
```

3.3. SAVING A CONFIGURATION CHANGE TO A WORKING FILE

When configuring a cluster, you can save configuration changes to a specified file without affecting the active CIB. This allows you to specify configuration updates without immediately updating the currently running cluster configuration with each individual update.

For information on saving the CIB to a file, see Viewing the raw cluster configuration. Once you have created that file, you can save configuration changes to that file rather than to the active CIB by using the **-f** option of the **pcs** command. When you have completed the changes and are ready to update the active CIB file, you can push those file updates with the **pcs cluster cib-push** command.

The following is the recommended procedure for pushing changes to the CIB file. This procedure creates a copy of the original saved CIB file and makes changes to that copy. When pushing those changes to the active CIB, this procedure specifies the **diff-against** option of the **pcs cluster cib-push** command so that only the changes between the original file and the updated file are pushed to the CIB. This allows users to make changes in parallel that do not overwrite each other, and it reduces the load on Pacemaker which does not need to parse the entire configuration file.

1. Save the active CIB to a file. This example saves the CIB to a file named **original.xml**.

```
# pcs cluster cib original.xml
```

2. Copy the saved file to the working file you will be using for the configuration updates.

```
# cp original.xml updated.xml
```

3. Update your configuration as needed. The following command creates a resource in the file **updated.xml** but does not add that resource to the currently running cluster configuration.

```
# pcs -f updated.xml resource create VirtualIP ocf:heartbeat:IPaddr2 ip=192.168.0.120
op monitor interval=30s
```

4. Push the updated file to the active CIB, specifying that you are pushing only the changes you have made to the original file.

```
# pcs cluster cib-push updated.xml diff-against=original.xml
```

Alternately, you can push the entire current content of a CIB file with the following command.

```
pcs cluster cib-push filename
```

When pushing the entire CIB file, Pacemaker checks the version and does not allow you to push a CIB file which is older than the one already in a cluster. If you need to update the entire CIB file with a version that is older than the one currently in the cluster, you can use the **--config** option of the **pcs cluster cib-push** command.

```
pcs cluster cib-push --config filename
```

3.4. DISPLAYING CLUSTER STATUS

You can display the status of the cluster and the cluster resources with the following command.

```
pcs status
```

You can display the status of a particular cluster component with the *commands* parameter of the **pcs status** command, specifying **resources**, **cluster**, **nodes**, or **pcsd**.

```
pcs status commands
```

For example, the following command displays the status of the cluster resources.

```
pcs status resources
```

The following command displays the status of the cluster, but not the cluster resources.

```
pcs cluster status
```

3.5. DISPLAYING THE FULL CLUSTER CONFIGURATION

Use the following command to display the full current cluster configuration.

```
pcs config
```

CHAPTER 4. CREATING A RED HAT HIGH-AVAILABILITY CLUSTER WITH PACEMAKER

The following procedure creates 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 in this example 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. We recommend but do not require a private network for communication among the cluster nodes and other cluster hardware such as network power switches and Fibre Channel switches.
- A 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**.

4.1. INSTALLING CLUSTER SOFTWARE

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

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

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

   Alternatively, you can install the Red Hat High Availability Add-On software packages along with only the fence agent that you require with the following command.

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

   The following command displays a listing of the available fence agents.

```
# rpm -q -a | grep fence
fence-agents-rhevm-4.0.2-3.el7.x86_64
fence-agents-ilo-mp-4.0.2-3.el7.x86_64
fence-agents-ipmilan-4.0.2-3.el7.x86_64
...
```

WARNING

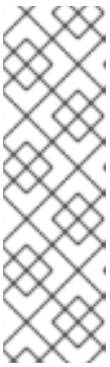
After you install the Red Hat High Availability Add-On packages, you should ensure that your software update preferences are set so that nothing is installed automatically. Installation on a running cluster can cause unexpected behaviors. For further information on performing software updates to a cluster, see Recommended Practices for Applying Software Updates to a RHEL High Availability or Resilient Storage Cluster.

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
```



NOTE

The ideal firewall configuration for cluster components depends on the local environment, where you may need to take into account such considerations as whether the nodes have multiple network interfaces or whether off-host firewalling is present. The example here, which opens the ports that are generally required by a Pacemaker cluster, should be modified to suit local conditions. Enabling ports for the High Availability Add-On shows the ports to enable for the Red Hat High Availability Add-On and provides an explanation for what each port is used for.

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 start up on boot 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 host auth z1.example.com z2.example.com
Username: hacluster
Password:
z1.example.com: Authorized
z2.example.com: Authorized
```

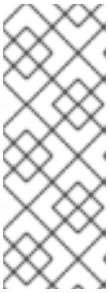
4.2. CREATING A HIGH AVAILABILITY CLUSTER

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 my_cluster --start z1.example.com z2.example.com
```

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



NOTE

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

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

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

```
[root@z1 ~]# pcs cluster status
Cluster Status:
 Stack: corosync
 Current DC: z2.example.com (version 2.0.0-10.el8-b67d8d0de9) - partition with quorum
 Last updated: Thu Oct 11 16:11:18 2018
 Last change: Thu Oct 11 16:11:00 2018 by hacluster via crmd on z2.example.com
 2 Nodes configured
 0 Resources configured

...
```

4.3. CREATING A HIGH AVAILABILITY CLUSTER WITH MULTIPLE LINKS

You can use the **pcs cluster setup** command to create a Red Hat High Availability cluster with multiple links by specifying all of the links for each node.

The format for the command to create a two-node cluster with two links is as follows.

```
pcs cluster setup cluster_name node1_name addr=node1_link0_address addr=node1_link1_address
node2_name addr=node2_link0_address addr=node2_link1_address
```

When creating a cluster with multiple links, you should take the following into account.

- The order of the **addr=*address*** parameters is important. The first address specified after a node name is for **link0**, the second one for **link1**, and so forth.
- It is possible to specify up to eight links using the knet transport protocol, which is the default transport protocol.
- All nodes must have the same number of **addr=** parameters.
- Currently, it is not possible to add, remove, or change links in an existing cluster using the **pcs** command.
- As with single-link clusters, do not mix IPv4 and IPv6 addresses in one link, altough you can have one link running IPv4 and the other running IPv6.
- As with single-link clusters, you can specify addresses as IP addresses or as names as long as the names resolve to IPv4 or IPv6 addresses for which IPv4 and IPv6 addresses are not mixed in one link.

The following example creates a two-node cluster named **my_twolink_cluster** with two nodes, **rh80-node1** and **rh80-node2**. **rh80-node1** has two interfaces, IP address 192.168.122.201 as **link0** and 192.168.123.201 as **link1**. **rh80-node2** has two interfaces, IP address 192.168.122.202 as **link0** and 192.168.123.202 as **link1**.

```
# pcs cluster setup my_twolink_cluster rh80-node1 addr=192.168.122.201
addr=192.168.123.201 rh80-node2 addr=192.168.122.202 addr=192.168.123.202
```

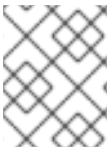
When adding a node to a cluster with multiple links, you must specify addresses for all links. The following example adds the node **rh80-node3** to a cluster, specifying IP address 192.168.122.203 as link0 and 192.168.123.203 as link1.

```
# pcs cluster node add rh80-node3 addr=192.168.122.203 addr=192.168.123.203
```

4.4. CONFIGURING FENCING

You must configure a fencing device for each node in the cluster. For information about the fence configuration commands and options, see Configuring fencing in a Red Hat High Availability cluster .

For general information on fencing and its importance in a Red Hat High Availability cluster, see Fencing in a Red Hat High Availability Cluster.



NOTE

When configuring a fencing device, 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** option.

You create a fencing device by configuring the device as a **stonith** resource with the **pcs stonith create** command. The following command configures a **stonith** resource named **myapc** that uses the

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

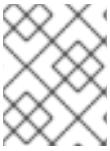
Note that you can use an IP address when specifying the host name for the nodes.

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

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

```
[root@rh7-1 ~]# pcs stonith config 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
login=apc passwd=apc
  Operations: monitor interval=60s (myapc-monitor-interval-60s)
```

After configuring your fence device, you should test the device. For information on testing a fence device, see Testing a fence device .



NOTE

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

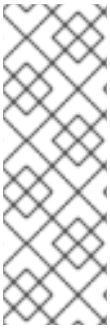
NOTE

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

4.5. BACKING UP AND RESTORING A CLUSTER CONFIGURATION

You can back up the cluster configuration in a tarball with the following command. If you do not specify a file name, the standard output will be used.

```
pcs config backup filename
```



NOTE

The **pcs config backup** command backs up only the cluster configuration itself as configured in the CIB; the configuration of resource daemons is out of the scope of this command. For example if you have configured an Apache resource in the cluster, the resource settings (which are in the CIB) will be backed up, while the Apache daemon settings (as set in`/etc/httpd`) and the files it serves will not be backed up. Similarly, if there is a database resource configured in the cluster, the database itself will not be backed up, while the database resource configuration (CIB) will be.

Use the following command to restore the cluster configuration files on all nodes from the backup. If you do not specify a file name, the standard input will be used. Specifying the **--local** option restores only the files on the current node.

```
pcs config restore [--local] [filename]
```

4.6. ENABLING PORTS FOR THE HIGH AVAILABILITY ADD-ON

The ideal firewall configuration for cluster components depends on the local environment, where you may need to take into account such considerations as whether the nodes have multiple network interfaces or whether off-host firewalling is present.

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. You may need to modify which ports are open to suit local conditions.

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
```

Table 4.1, "Ports to Enable for High Availability Add-On" shows the ports to enable for the Red Hat High Availability Add-On and provides an explanation for what the port is used for.

Table 4.1. Ports to Enable for High Availability Add-On

Port	When Required
TCP 2224	Default **pcsd** port required on all nodes (needed by the pcsd Web UI and required for node-to-node communication). You can configure the **pcsd** port by means of the **PCSD_PORT** parameter in the **/etc/sysconfig/pcsd** file. It is crucial to open port 2224 in such a way that **pcs** from any node can talk to all nodes in the cluster, including itself. When using the Booth cluster ticket manager or a quorum device you must open port 2224 on all related hosts, such as Booth arbiters or the quorum device host.

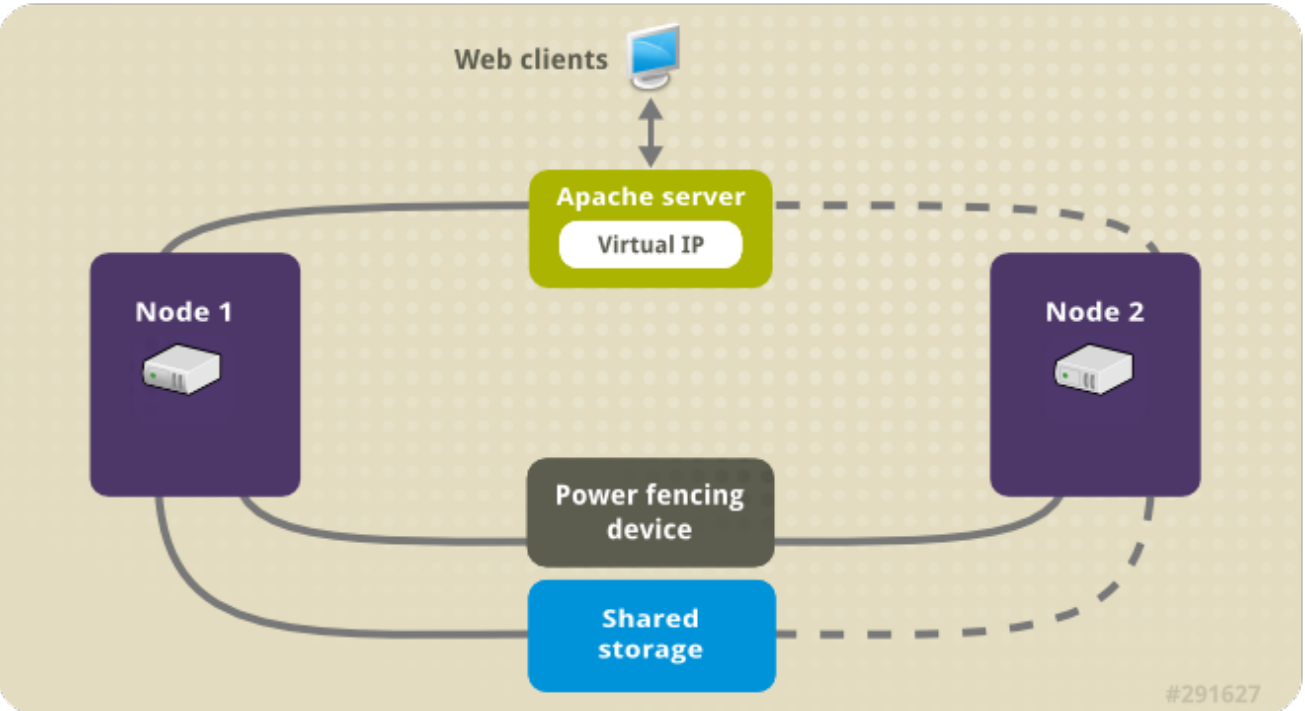
Port	When Required
TCP 3121	Required on all nodes if the cluster has any Pacemaker Remote nodes Pacemaker's **pacemaker-based** daemon on the full cluster nodes will contact the **pacemaker_remoted** daemon on Pacemaker Remote nodes at port 3121. If a separate interface is used for cluster communication, the port only needs to be open on that interface. At a minimum, the port should open on Pacemaker Remote nodes to full cluster nodes. Because users may convert a host between a full node and a remote node, or run a remote node inside a container using the host's network, it can be useful to open the port to all nodes. It is not necessary to open the port to any hosts other than nodes.
TCP 5403	Required on the quorum device host when using a quorum device with **corosync-qnetd**. The default value can be changed with the **-p** option of the **corosync-qnetd** command.
UDP 5404-5412	Required on corosync nodes to facilitate communication between nodes. It is crucial to open ports 5404-5412 in such a way that **corosync** from any node can talk to all nodes in the cluster, including itself.
TCP 21064	Required on all nodes if the cluster contains any resources requiring DLM (such as **GFS2**).

CHAPTER 5. CONFIGURING AN ACTIVE/PASSIVE APACHE HTTP SERVER IN A RED HAT HIGH AVAILABILITY CLUSTER

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

Figure 5.1, "Apache in a Red Hat High Availability Two-Node Cluster" shows a high-level overview of the cluster. The cluster is a two-node Red Hat High Availability cluster which is configured with a network power switch and with shared storage. The cluster nodes are connected to a public network, for client access to the Apache HTTP server through a virtual IP. The Apache server runs on either Node 1 or Node 2, each of which has access to the storage on which the Apache data is kept.

Figure 5.1. Apache in a Red Hat High Availability Two-Node Cluster



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

- A two-node Red Hat High Availability cluster with power fencing configured for each node. We recommend but do not require a private network. This procedure uses the cluster example provided in Creating a Red Hat High-Availability cluster with Pacemaker .
- A public virtual IP address, required for Apache.
- Shared storage for the nodes in the cluster, using iSCSI 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:

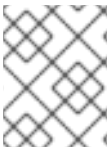
1. Configure an **ext4** file system on the logical volume **my_lv**.

2. Configure a web server.

After performing these procedures, you create the resource group and the resources it contains.

5.1. CONFIGURING AN LVM VOLUME WITH AN EXT4 FILE SYSTEM IN A PACEMAKER CLUSTER

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



NOTE

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

The following procedure creates an LVM logical volume and then creates an **ext4** file system on that volume for use in a Pacemaker cluster. 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.

1. On both nodes of the cluster, perform the following steps to set the value for the LVM system ID to the value of the **uname** for the system. The LVM system ID will be used to ensure that only the cluster is capable of activating the volume group.
   a. Set the **system_id_source** configuration option in the **/etc/lvm/lvm.conf** configuration file to **uname**.

```
# Configuration option global/system_id_source.
system_id_source = "uname"
```

   b. Verify that the LVM system ID on the node matches the **uname** for the node.

```
# lvm systemid
  system ID: z1.example.com
# uname -n
  z1.example.com
```

2. Create the LVM volume and create an **ext4** file system on that volume. Since the **/dev/sdb1** partition is storage that is shared, you perform this part of the procedure on one node only.
   a. Create an LVM physical volume on partition **/dev/sdb1**.

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

   b. 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
```

   c. Verify that the new volume group has the system ID of the node on which you are running and from which you created the volume group.

```
# vgs -o+systemid
  VG    #PV #LV #SN Attr   VSize  VFree  System ID
  my_vg   1   0   0 wz--n- <1.82t <1.82t z1.example.com
```

d. 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
  ...
```

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

```
# mkfs.ext4 /dev/my_vg/my_lv
mke2fs 1.44.3 (10-July-2018)
Creating filesystem with 462848 1k blocks and 115824 inodes
...
```

5.2. CONFIGURING AN APACHE HTTP SERVER

The following procedure configures an Apache HTTP server.

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

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

If you are running the **firewalld** daemon, on each node in the cluster enable the ports that are required by the Red Hat High Availability Add-On.

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

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

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

3. When you use the **apache** resource agent to manage Apache, it does not use **systemd**. Because of this, you must edit the **logrotate** script supplied with Apache so that it does not use **systemctl** to reload Apache.
   Remove the following line in the **/etc/logrotate.d/httpd** file on each node in the cluster.

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

   Replace the line you removed with the following line.

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

4. Create a web page for Apache to serve up. On one node in the cluster, mount the file system you created in Configuring an LVM volume with an ext4 file system , create the file **index.html** on that file system, and 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
   ```

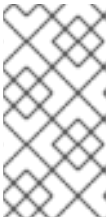
5.3. CREATING THE RESOURCES AND RESOURCE GROUPS

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

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

The following procedure creates the resource group **apachegroup** and the resources that the group contains. The resources will start in the order in which you add them to the group, and they will stop in the reverse order in which they are added to the group. Run this procedure from one node of the cluster only.

1. The following command creates the **LVM-activate** resource **my_lvm**. Because the resource group **apachegroup** does not yet exist, this command creates the resource group.



NOTE

Do not configure more than one **LVM-activate** resource that uses the same LVM volume group in an active/passive HA configuration, as this risks data corruption. Additionally, do not configure an **LVM-activate** resource as a clone resource in an active/passive HA configuration.

```
[root@z1 ~]# pcs resource create my_lvm ocf:heartbeat:LVM-activate vgname=my_vg
vg_access_mode=system_id --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 status
 Resource Group: apachegroup
     my_lvm (ocf::heartbeat:LVM-activate): 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 by default the resources do not start.

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

```
Hello
```

If you find that the resources you configured are not running, you can run the **pcs resource debug-start *resource*** command to test the resource configuration.

5.4. TESTING THE RESOURCE CONFIGURATION

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

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

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

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

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

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

Full list of resources:

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

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

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

```
[root@z1 ~]# pcs 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.

CHAPTER 6. CONFIGURING AN ACTIVE/PASSIVE NFS SERVER IN A RED HAT HIGH AVAILABILITY CLUSTER

The following procedure configures 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:

- A two-node Red Hat High Availability cluster with power fencing configured for each node. We recommend but do not require a private network. This procedure uses the cluster example provided in Creating a Red Hat High-Availability cluster with Pacemaker .
- 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 an existing two-node Red Hat Enterprise Linux High Availability cluster requires that you perform the following steps.

1. Configure an **ext4** file system on the LVM logical volume **my_lv** on the shared storage for the nodes in the cluster.
2. Configure an NFS share on the shared storage on the LVM logical volume.
3. Create the cluster resources.
4. Test the NFS server you have configured.

6.1. CONFIGURING AN LVM VOLUME WITH AN EXT4 FILE SYSTEM IN A PACEMAKER CLUSTER

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

NOTE

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

The following procedure creates an LVM logical volume and then creates an **ext4** file system on that volume for use in a Pacemaker cluster. 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.

1. On both nodes of the cluster, perform the following steps to set the value for the LVM system ID to the value of the **uname** for the system. The LVM system ID will be used to ensure that only the cluster is capable of activating the volume group.
    a. Set the **system_id_source** configuration option in the **/etc/lvm/lvm.conf** configuration file to **uname**.

```
# Configuration option global/system_id_source.
system_id_source = "uname"
```

b. Verify that the LVM system ID on the node matches the **uname** for the node.

```
# lvm systemid
  system ID: z1.example.com
# uname -n
  z1.example.com
```

2. Create the LVM volume and create an **ext4** file system on that volume. Since the **/dev/sdb1** partition is storage that is shared, you perform this part of the procedure on one node only.

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

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

b. 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
```

c. Verify that the new volume group has the system ID of the node on which you are running and from which you created the volume group.

```
# vgs -o+systemid
  VG    #PV #LV #SN Attr   VSize  VFree  System ID
  my_vg   1   0   0 wz--n- <1.82t <1.82t z1.example.com
```

d. 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
  ...
```

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

```
# mkfs.ext4 /dev/my_vg/my_lv
mke2fs 1.44.3 (10-July-2018)
Creating filesystem with 462848 1k blocks and 115824 inodes
...
```

6.2. CONFIGURING AN NFS SHARE

The following procedure configures the NFS share for the NFS daemon failover.

1. On both nodes in the cluster, create the **/nfsshare** directory.

```
# mkdir /nfsshare
```

2. On one node in the cluster, perform the following procedure.

   a. Mount the **ext4** file system that you created in Configuring an LVM volume with an ext4 file system on the **/nfsshare** directory.

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

   b. 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
```

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

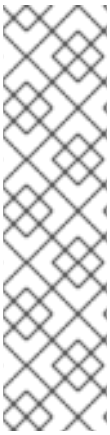
```
[root@z1 ~]# touch /nfsshare/exports/export1/clientdatafile1
[root@z1 ~]# touch /nfsshare/exports/export2/clientdatafile2
```

   d. 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
```

6.3. CONFIGURING THE RESOURCES AND RESOURCE GROUP FOR AN NFS SERVER IN A CLUSTER

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, by default the resources do not start.

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

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-activate resource named **my_lvm**. Because the resource group **nfsgroup** does not yet exist, this command creates the resource group.

NOTE

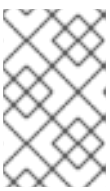
Do not configure more than one **LVM-activate** resource that uses the same LVM volume group in an active/passive HA configuration, as this risks data corruption. Additionally, do not configure an **LVM-activate** resource as a clone resource in an active/passive HA configuration.

```
[root@z1 ~]# pcs resource create my_lvm ocf:heartbeat:LVM-activate vgname=my_vg
vg_access_mode=system_id --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 Configuring an LVM volume with an ext4 file system and will be mounted on the **/nfsshare** directory you created in Configuring an NFS share .

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

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

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

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

NOTE

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

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

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

NOTE

The **fsid=0** option is required only if you want to create a virtual directory for NFSv4 clients. For more information, see How do I configure the fsid option in an NFS server's /etc/exports file?.

```
[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. This resource is part of the resource group **nfsgroup**. For this example deployment, we are using 192.168.122.200 as the floating IP address.

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

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

NOTE

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

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

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

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

6.4. TESTING THE NFS 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 Configuring an NFS share

```
# 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 node 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
```

CHAPTER 7. GFS2 FILE SYSTEMS IN A CLUSTER

This section provideas:

- A procedure to set up a Pacemaker cluster that includes GFS2 file systems
- A procedure to migrate RHEL 7 logical volumes that contain GFS2 file systems to a RHEL 8 cluster

7.1. CONFIGURING A GFS2 FILE SYSTEM IN A CLUSTER

This procedure is an outline of the steps required to set up a Pacemaker cluster that includes GFS2 file systems. This example creates three GFS2 file systems on three logical volumes.

As a prerequisite for this procedure, you must install and start the cluster software on all nodes and create a basic two-node cluster. You must also configure fencing for the cluster. For information on creating a Pacemaker cluster and configuring fencing for the cluster, see Creating a Red Hat High-Availability cluster with Pacemaker.

1. On both nodes of the cluster, install the **lvm2-lockd**, **gfs2-utils**, and **dlm** packages. The **lvm2-lockd** package is part of the AppStream channel and the **gfs2-utils** and **dlm** packages are part of the Resilient Storage channel.

```
# yum install lvm2-lockd gfs2-utils dlm
```

2. Set up a **dlm** resource. This is a required dependency for configuring a GFS2 file system in a cluster. This example creates the **dlm** resource as part of a resource group named **locking**.

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

3. Clone the **locking** resource group so that the resource group can be active on both nodes of the cluster.

```
[root@z1 ~]# pcs resource clone locking interleave=true
```

4. Set up an **lvmlockd** resource as part of the group **locking**.

```
[root@z1 ~]# pcs resource create lvmlockd --group locking ocf:heartbeat:lvmlockd op monitor interval=30s on-fail=fence
```

5. Check the status of the cluster to ensure that the **locking** resource group has started on both nodes of the cluster.

```
[root@z1 ~]# pcs status --full
Cluster name: my_cluster
[...]

Online: [ z1.example.com (1) z2.example.com (2) ]

Full list of resources:

 smoke-apc      (stonith:fence_apc):    Started z1.example.com
 Clone Set: locking-clone [locking]
```

```
    Resource Group: locking:0
      dlm    (ocf::pacemaker:controld):      Started z1.example.com
      lvmlockd       (ocf::heartbeat:lvmlockd):      Started z1.example.com
    Resource Group: locking:1
      dlm    (ocf::pacemaker:controld):      Started z2.example.com
      lvmlockd       (ocf::heartbeat:lvmlockd):      Started z2.example.com
    Started: [ z1.example.com z2.example.com ]
```

6. Verify that the **lvmlockd** daemon is running on both nodes of the cluster.

```
[root@z1 ~]# ps -ef | grep lvmlockd
root     12257     1  0 17:45 ?        00:00:00 lvmlockd -p /run/lvmlockd.pid -A 1 -g dlm
[root@z2 ~]# ps -ef | grep lvmlockd
root     12270     1  0 17:45 ?        00:00:00 lvmlockd -p /run/lvmlockd.pid -A 1 -g dlm
```

7. On one node of the cluster, create two shared volume groups. One volume group will contain two GFS2 file systems, and the other volume group will contain one GFS2 file system.
   The following command creates the shared volume group **shared_vg1** on **/dev/vdb**.

```
[root@z1 ~]# vgcreate --shared shared_vg1 /dev/vdb
  Physical volume "/dev/vdb" successfully created.
  Volume group "shared_vg1" successfully created
  VG shared_vg1 starting dlm lockspace
  Starting locking.  Waiting until locks are ready...
```

   The following command creates the shared volume group **shared_vg2** on **/dev/vdc**.

```
[root@z1 ~]# vgcreate --shared shared_vg2 /dev/vdc
  Physical volume "/dev/vdc" successfully created.
  Volume group "shared_vg2" successfully created
  VG shared_vg2 starting dlm lockspace
  Starting locking.  Waiting until locks are ready...
```

8. On the second node in the cluster, start the lock manager for each of the shared volume groups.

```
[root@z2 ~]# vgchange --lock-start shared_vg1
  VG shared_vg1 starting dlm lockspace
  Starting locking.  Waiting until locks are ready...
[root@z2 ~]# vgchange --lock-start shared_vg2
  VG shared_vg2 starting dlm lockspace
  Starting locking.  Waiting until locks are ready...
```

9. On one node in the cluster, create the shared logical volumes and format the volumes with a GFS2 file system. Ensure that you create enough journals for each of the nodes in your cluster.

```
[root@z1 ~]# lvcreate --activate sy -L5G -n shared_lv1 shared_vg1
  Logical volume "shared_lv1" created.
[root@z1 ~]# lvcreate --activate sy -L5G -n shared_lv2 shared_vg1
  Logical volume "shared_lv2" created.
[root@z1 ~]# lvcreate --activate sy -L5G -n shared_lv1 shared_vg2
  Logical volume "shared_lv1" created.

[root@z1 ~]# mkfs.gfs2 -j2 -p lock_dlm -t my_cluster:gfs2-demo1
/dev/shared_vg1/shared_lv1
```

```
[root@z1 ~]# mkfs.gfs2 -j2 -p lock_dlm -t my_cluster:gfs2-demo2
/dev/shared_vg1/shared_lv2
[root@z1 ~]# mkfs.gfs2 -j2 -p lock_dlm -t my_cluster:gfs2-demo3
/dev/shared_vg2/shared_lv1
```

10. Create an **LVM-activate** resource for each logical volume to automatically activate that logical volume on all nodes.

    a. Create an **LVM-activate** resource named **sharedlv1** for the logical volume **shared_lv1** in volume group **shared_vg1**. This command also creates the resource group **shared_vg1** that includes the resource. In this example, the resource group has the same name as the shared volume group that includes the logical volume.

    ```
    [root@z1 ~]# pcs resource create sharedlv1 --group shared_vg1 ocf:heartbeat:LVM-
    activate lvname=shared_lv1 vgname=shared_vg1 activation_mode=shared
    vg_access_mode=lvmlockd
    ```

    b. Create an **LVM-activate** resource named **sharedlv2** for the logical volume **shared_lv2** in volume group **shared_vg1**. This resource will also be part of the resource group **shared_vg1**.

    ```
    [root@z1 ~]# pcs resource create sharedlv2 --group shared_vg1 ocf:heartbeat:LVM-
    activate lvname=shared_lv2 vgname=shared_vg1 activation_mode=shared
    vg_access_mode=lvmlockd
    ```

    c. Create an **LVM-activate** resource named **sharedlv3** for the logical volume **shared_lv1** in volume group **shared_vg2**. This command also creates the resource group **shared_vg2** that includes the resource.

    ```
    [root@z1 ~]# pcs resource create sharedlv3 --group shared_vg2 ocf:heartbeat:LVM-
    activate lvname=shared_lv1 vgname=shared_vg2 activation_mode=shared
    vg_access_mode=lvmlockd
    ```

11. Clone the two new resource groups.

```
[root@z1 ~]# pcs resource clone shared_vg1 interleave=true
[root@z1 ~]# pcs resource clone shared_vg2 interleave=true
```

12. Configure ordering constraints to ensure that the **locking** resource group that includes the **dlm** and **lvmlockd** resources starts first.

```
[root@z1 ~]# pcs constraint order start locking-clone then shared_vg1-clone
Adding locking-clone shared_vg1-clone (kind: Mandatory) (Options: first-action=start then-
action=start)
[root@z1 ~]# pcs constraint order start locking-clone then shared_vg2-clone
Adding locking-clone shared_vg2-clone (kind: Mandatory) (Options: first-action=start then-
action=start)
```

13. On both nodes in the cluster, verify that the logical volumes are active. There may be a delay of a few seconds.

```
[root@z1 ~]# lvs
  LV        VG         Attr       LSize
  shared_lv1 shared_vg1  -wi-a----- 5.00g
```

```
  shared_lv2 shared_vg1  -wi-a----- 5.00g
  shared_lv1 shared_vg2  -wi-a----- 5.00g

[root@z2 ~]# lvs
  LV         VG         Attr       LSize
  shared_lv1 shared_vg1  -wi-a----- 5.00g
  shared_lv2 shared_vg1  -wi-a----- 5.00g
  shared_lv1 shared_vg2  -wi-a----- 5.00g
```

14. Create a file system resource to automatically mount each GFS2 file system on all nodes. You should not add the file system to the **/etc/fstab** file because it will be managed as a Pacemaker cluster resource. Mount options can be specified as part of the resource configuration with **options=*options***. Run the **pcs resource describe Filesystem** command for full configuration options.

    The following commands create the file system resources. These commands add each resource to the resource group that includes the logical volume resource for that file system.

```
[root@z1 ~]# pcs resource create sharedfs1 --group shared_vg1
ocf:heartbeat:Filesystem device="/dev/shared_vg1/shared_lv1" directory="/mnt/gfs1"
fstype="gfs2" options=noatime op monitor interval=10s on-fail=fence
[root@z1 ~]# pcs resource create sharedfs2 --group shared_vg1
ocf:heartbeat:Filesystem device="/dev/shared_vg1/shared_lv2" directory="/mnt/gfs2"
fstype="gfs2" options=noatime op monitor interval=10s on-fail=fence
[root@z1 ~]# pcs resource create sharedfs3 --group shared_vg2
ocf:heartbeat:Filesystem device="/dev/shared_vg2/shared_lv1" directory="/mnt/gfs3"
fstype="gfs2" options=noatime op monitor interval=10s on-fail=fence
```

15. Verify that the GFS2 file systems are mounted on both nodes of the cluster.

```
[root@z1 ~]# mount | grep gfs2
/dev/mapper/shared_vg1-shared_lv1 on /mnt/gfs1 type gfs2 (rw,noatime,seclabel)
/dev/mapper/shared_vg1-shared_lv2 on /mnt/gfs2 type gfs2 (rw,noatime,seclabel)
/dev/mapper/shared_vg2-shared_lv1 on /mnt/gfs3 type gfs2 (rw,noatime,seclabel)

[root@z2 ~]# mount | grep gfs2
/dev/mapper/shared_vg1-shared_lv1 on /mnt/gfs1 type gfs2 (rw,noatime,seclabel)
/dev/mapper/shared_vg1-shared_lv2 on /mnt/gfs2 type gfs2 (rw,noatime,seclabel)
/dev/mapper/shared_vg2-shared_lv1 on /mnt/gfs3 type gfs2 (rw,noatime,seclabel)
```

16. Check the status of the cluster.

```
[root@z1 ~]# pcs status --full
Cluster name: my_cluster
[...1

Full list of resources:

 smoke-apc      (stonith:fence_apc):    Started z1.example.com
 Clone Set: locking-clone [locking]
     Resource Group: locking:0
         dlm    (ocf::pacemaker:controld):      Started z2.example.com
         lvmlockd       (ocf::heartbeat:lvmlockd):      Started z2.example.com
     Resource Group: locking:1
         dlm    (ocf::pacemaker:controld):      Started z1.example.com
```

```
      lvmlockd      (ocf::heartbeat:lvmlockd):      Started z1.example.com
    Started: [ z1.example.com z2.example.com ]
  Clone Set: shared_vg1-clone [shared_vg1]
    Resource Group: shared_vg1:0
      sharedlv1     (ocf::heartbeat:LVM-activate):  Started z2.example.com
      sharedlv2     (ocf::heartbeat:LVM-activate):  Started z2.example.com
      sharedfs1     (ocf::heartbeat:Filesystem):    Started z2.example.com
      sharedfs2     (ocf::heartbeat:Filesystem):    Started z2.example.com
    Resource Group: shared_vg1:1
      sharedlv1     (ocf::heartbeat:LVM-activate):  Started z1.example.com
      sharedlv2     (ocf::heartbeat:LVM-activate):  Started z1.example.com
      sharedfs1     (ocf::heartbeat:Filesystem):    Started z1.example.com
      sharedfs2     (ocf::heartbeat:Filesystem):    Started example.co
    Started: [ z1.example.com z2.example.com ]
  Clone Set: shared_vg2-clone [shared_vg2]
    Resource Group: shared_vg2:0
      sharedlv3     (ocf::heartbeat:LVM-activate):  Started z2.example.com
      sharedfs3     (ocf::heartbeat:Filesystem):    Started z2.example.com
    Resource Group: shared_vg2:1
      sharedlv3     (ocf::heartbeat:LVM-activate):  Started z1.example.com
      sharedfs3     (ocf::heartbeat:Filesystem):    Started z1.example.com
    Started: [ z1.example.com z2.example.com ]

...
```

7.2. MIGRATING A GFS2 FILE SYSTEM FROM RHEL7 TO RHEL8

In Red Hat Enterprise Linux 8, LVM uses the LVM lock daemon **lvmlockd** instead of **clvmd** for managing shared storage devices in an active/active cluster. This requires that you configure the logical volumes that your active/active cluster will require as shared logical volumes. Additionally, this requires that you use the **LVM-activate** resource to manage an LVM volume and that you use the **lvmlockd** resource agent to manage the **lvmlockd** daemon. See Configuring a GFS2 file system in a cluster for a full procedure for configuring a Pacemaker cluster that includes GFS2 file systems using shared logical volumes.

To use your existing Red Hat Enterprise Linux 7 logical volumes when configuring a RHEL8 cluster that includes GFS2 file systems, perform the following procedure from the RHEL8 cluster. In this example, the clustered RHEL 7 logical volume is part of the volume group **upgrade_gfs_vg**.

NOTE

The RHEL8 cluster must have the same name as the RHEL7 cluster that includes the GFS2 file system in order for the existing file system to be valid.

1. Ensure that the logical volumes containing the GFS2 file systems are currently inactive. This procedure is safe only if all nodes have stopped using the volume group.

2. From one node in the cluster, forcibly change the volume group to be local.

```
[root@rhel8-01 ~]# vgchange --lock-type none --lock-opt force upgrade_gfs_vg
Forcibly change VG lock type to none? [y/n]: y
  Volume group "upgrade_gfs_vg" successfully changed
```

3. From one node in the cluster, change the local volume group to a shared volume group

```
[root@rhel8-01 ~]# vgchange --lock-type dlm upgrade_gfs_vg
  Volume group "upgrade_gfs_vg" successfully changed
```

4. On each node in the cluster, start locking for the volume group.

```
[root@rhel8-01 ~]# vgchange --lock-start upgrade_gfs_vg
  VG upgrade_gfs_vg starting dlm lockspace
  Starting locking.  Waiting until locks are ready...
[root@rhel8-02 ~]# vgchange --lock-start upgrade_gfs_vg
  VG upgrade_gfs_vg starting dlm lockspace
  Starting locking.  Waiting until locks are ready...
```

After performing this procedure, you can create an **LVM-activate** resource for each logical volume.

CHAPTER 8. CONFIGURING FENCING IN A RED HAT HIGH AVAILABILITY CLUSTER

A node that is unresponsive may still be accessing data. The only way to be certain that your data is safe is to fence the node using STONITH. STONITH is an acronym for "Shoot The Other Node In The Head" and it protects your data from being corrupted by rogue nodes or concurrent access. Using STONITH, you can be certain that a node is truly offline before allowing the data to be accessed from another node.

STONITH also has a role to play in the event that a clustered service cannot be stopped. In this case, the cluster uses STONITH to force the whole node offline, thereby making it safe to start the service elsewhere.

For more complete general information on fencing and its importance in a Red Hat High Availability cluster, see Fencing in a Red Hat High Availability Cluster .

You implement STONITH in a Pacemaker cluster by configuring fence devices for the nodes of the cluster.

8.1. DISPLAYING AVAILABLE FENCE AGENTS AND THEIR OPTIONS

Use the following command to view of list of all available STONITH agents. When you specify a filter, this command displays only the STONITH agents that match the filter.

```
pcs stonith list [filter]
```

Use the following command to view the options for the specified STONITH agent.

```
pcs stonith describe stonith_agent
```

For example, the following command displays the options for the fence agent for APC over telnet/SSH.

```
# pcs stonith describe fence_apc
Stonith options for: fence_apc
  ipaddr (required): IP Address or Hostname
  login (required): Login Name
  passwd: Login password or passphrase
  passwd_script: Script to retrieve password
  cmd_prompt: Force command prompt
  secure: SSH connection
  port (required): Physical plug number or name of virtual machine
  identity_file: Identity file for ssh
  switch: Physical switch number on device
  inet4_only: Forces agent to use IPv4 addresses only
  inet6_only: Forces agent to use IPv6 addresses only
  ipport: TCP port to use for connection with device
  action (required): Fencing Action
  verbose: Verbose mode
  debug: Write debug information to given file
  version: Display version information and exit
  help: Display help and exit
  separator: Separator for CSV created by operation list
  power_timeout: Test X seconds for status change after ON/OFF
  shell_timeout: Wait X seconds for cmd prompt after issuing command
```

```
login_timeout: Wait X seconds for cmd prompt after login
power_wait: Wait X seconds after issuing ON/OFF
delay: Wait X seconds before fencing is started
retry_on: Count of attempts to retry power on
```

WARNING

For fence agents that provide a **method** option, a value of **cycle** is unsupported and should not be specified, as it may cause data corruption.

8.2. CREATING A FENCE DEVICE

The following command creates a stonith device. For additional available options, see the **pcs stonith -h** display.

```
pcs stonith create stonith_id stonith_device_type [stonith_device_options] [op operation_action operation_options]
```

```
# pcs stonith create MyStonith fence_virt pcmk_host_list=f1 op monitor interval=30s
```

If you use a single fence device for several nodes, using a different port for each node, you do not need to create a device separately for each node. For example, the following command creates a single fencing device called **myapc-west-13** that uses an APC power switch called **west-apc** and uses port 15 for node **west-13**.

```
# pcs stonith create myapc-west-13 fence_apc pcmk_host_list="west-13" ipaddr="west-apc" login="apc" passwd="apc" port="15"
```

The following example, however, uses the APC power switch named **west-apc** to fence nodes **west-13** using port 15, **west-14** using port 17, **west-15** using port 18, and **west-16** using port 19.

```
# pcs stonith create myapc fence_apc pcmk_host_list="west-13,west-14,west-15,west-16" ipaddr="west-apc" login="apc" passwd="apc"
```

After configuring a fence device, it is imperative that you test the device to ensure that it is working correctly. For information on testing a fence device, see Testing a fence device.

8.3. GENERAL PROPERTIES OF FENCING DEVICES

Any cluster node can fence any other cluster node with any fence device, regardless of whether the fence resource is started or stopped. Whether the resource is started controls only the recurring monitor for the device, not whether it can be used, with the following exceptions:

- You can disable a fencing device by running the **pcs stonith disable *stonith_id*** command. This will prevent any node from using that device.
- To prevent a specific node from using a fencing device, you can configure location constraints for the fencing resource with the **pcs constraint location ... avoids** command.

- Configuring **stonith-enabled=false** will disable fencing altogether. Note, however, that Red Hat does not support clusters when fencing is disabled, as it is not suitable for a production environment.

Table 8.1, "General Properties of Fencing Devices" describes the general properties you can set for fencing devices.

Table 8.1. General Properties of Fencing Devices

Field	Type	Default	Description
pcmk_host_map	string		A mapping of host names to ports numbers for devices that do not support host names. For example: **node1:1;node2:2,3** tells the cluster to use port 1 for node1 and ports 2 and 3 for node2
pcmk_host_list	string		A list of machines controlled by this device (Optional unless **pcmk_host_check=static-list**).
pcmk_host_check	string	* **static-list** if either **pcmk_host_list** or **pcmk_host_map** is set * Otherwise, **dynamic-list** if the fence device supports the **list** action * Otherwise, **status** if the fence device supports the **status** action *Otherwise, **none**.	How to determine which machines are controlled by the device. Allowed values: **dynamic-list** (query the device), **static-list** (check the **pcmk_host_list** attribute), none (assume every device can fence every machine)

8.4. ADVANCED FENCING CONFIGURATION OPTIONS

Table 8.2, "Advanced Properties of Fencing Devices" summarizes additional properties you can set for fencing devices. Note that these properties are for advanced use only.

Table 8.2. Advanced Properties of Fencing Devices

Field	Type	Default	Description
pcmk_host_argument	string	port	An alternate parameter to supply instead of port. Some devices do not support the standard port parameter or may provide additional ones. Use this to specify an alternate, device-specific parameter that should indicate the machine to be fenced. A value of **none** can be used to tell the cluster not to supply any additional parameters.

Field	Type	Default	Description
pcmk_reboot_action	string	reboot	An alternate command to run instead of **reboot**. Some devices do not support the standard commands or may provide additional ones. Use this to specify an alternate, device-specific, command that implements the reboot action.
pcmk_reboot_timeout	time	60s	Specify an alternate timeout to use for reboot actions instead of **stonith-timeout**. Some devices need much more/less time to complete than normal. Use this to specify an alternate, device-specific, timeout for reboot actions.
pcmk_reboot_retries	integer	2	The maximum number of times to retry the **reboot** command within the timeout period. Some devices do not support multiple connections. Operations may fail if the device is busy with another task so Pacemaker will automatically retry the operation, if there is time remaining. Use this option to alter the number of times Pacemaker retries reboot actions before giving up.
pcmk_off_action	string	off	An alternate command to run instead of **off**. Some devices do not support the standard commands or may provide additional ones. Use this to specify an alternate, device-specific, command that implements the off action.
pcmk_off_timeout	time	60s	Specify an alternate timeout to use for off actions instead of **stonith-timeout**. Some devices need much more or much less time to complete than normal. Use this to specify an alternate, device-specific, timeout for off actions.
pcmk_off_retries	integer	2	The maximum number of times to retry the off command within the timeout period. Some devices do not support multiple connections. Operations may fail if the device is busy with another task so Pacemaker will automatically retry the operation, if there is time remaining. Use this option to alter the number of times Pacemaker retries off actions before giving up.

Field	Type	Default	Description
pcmk_list_action	string	list	An alternate command to run instead of **list**. Some devices do not support the standard commands or may provide additional ones. Use this to specify an alternate, device-specific, command that implements the list action.
pcmk_list_timeout	time	60s	Specify an alternate timeout to use for list actions. Some devices need much more or much less time to complete than normal. Use this to specify an alternate, device-specific, timeout for list actions.
pcmk_list_retries	integer	2	The maximum number of times to retry the **list** command within the timeout period. Some devices do not support multiple connections. Operations may fail if the device is busy with another task so Pacemaker will automatically retry the operation, if there is time remaining. Use this option to alter the number of times Pacemaker retries list actions before giving up.
pcmk_monitor_action	string	monitor	An alternate command to run instead of **monitor**. Some devices do not support the standard commands or may provide additional ones. Use this to specify an alternate, device-specific, command that implements the monitor action.
pcmk_monitor_timeout	time	60s	Specify an alternate timeout to use for monitor actions instead of **stonith-timeout**. Some devices need much more or much less time to complete than normal. Use this to specify an alternate, device-specific, timeout for monitor actions.
pcmk_monitor_retries	integer	2	The maximum number of times to retry the **monitor** command within the timeout period. Some devices do not support multiple connections. Operations may fail if the device is busy with another task so Pacemaker will automatically retry the operation, if there is time remaining. Use this option to alter the number of times Pacemaker retries monitor actions before giving up.

Field	Type	Default	Description
pcmk_status_action	string	status	An alternate command to run instead of **status**. Some devices do not support the standard commands or may provide additional ones. Use this to specify an alternate, device-specific, command that implements the status action.
pcmk_status_timeout	time	60s	Specify an alternate timeout to use for status actions instead of **stonith-timeout**. Some devices need much more or much less time to complete than normal. Use this to specify an alternate, device-specific, timeout for status actions.
pcmk_status_retries	integer	2	The maximum number of times to retry the status command within the timeout period. Some devices do not support multiple connections. Operations may fail if the device is busy with another task so Pacemaker will automatically retry the operation, if there is time remaining. Use this option to alter the number of times Pacemaker retries status actions before giving up.
pcmk_delay_base	time	0s	Enable a base delay for stonith actions and specify a base delay value. In a cluster with an even number of nodes, configuring a delay can help avoid nodes fencing each other at the same time in an even split. A random delay can be useful when the same fence device is used for all nodes, and differing static delays can be useful on each fencing device when a separate device is used for each node The overall delay is derived from a random delay value adding this static delay so that the sum is kept below the maximum delay. If you set **pcmk_delay_base** but do not set **pcmk_delay_max**, there is no random component to the delay and it will be the value of **pcmk_delay_base**.

Field	Type	Default	Description
pcmk_delay_max	time	0s	Enable a random delay for stonith actions and specify the maximum of random delay. In a cluster with an even number of nodes, configuring a delay can help avoid nodes fencing each other at the same time in an even split. A random delay can be useful when the same fence device is used for all nodes, and differing static delays can be useful on each fencing device when a separate device is used for each node The overall delay is derived from this random delay value adding a static delay so that the sum is kept below the maximum delay. If you set **pcmk_delay_max** but do not set **pcmk_delay_base** there is no static component to the delay.
pcmk_action_limit	integer	1	The maximum number of actions that can be performed in parallel on this device. The cluster property **concurrent-fencing=true** needs to be configured first. A value of -1 is unlimited.
pcmk_on_action	string	on	For advanced use only: An alternate command to run instead of **on**. Some devices do not support the standard commands or may provide additional ones. Use this to specify an alternate, device-specific, command that implements the **on** action.
pcmk_on_timeout	time	60s	For advanced use only: Specify an alternate timeout to use for **on** actions instead of **stonith-timeout**. Some devices need much more or much less time to complete than normal. Use this to specify an alternate, device-specific, timeout for **on** actions.
pcmk_on_retries	integer	2	For advanced use only: The maximum number of times to retry the **on** command within the timeout period. Some devices do not support multiple connections. Operations may **fail** if the device is busy with another task so Pacemaker will automatically retry the operation, if there is time remaining. Use this option to alter the number of times Pacemaker retries **on** actions before giving up.

8.5. TESTING A FENCE DEVICE

Fencing is a fundamental part of the Red Hat Cluster infrastructure and it is therefore important to validate or test that fencing is working properly.

Use the following procedure to test a fence device.

1. Use ssh, telnet, HTTP, or whatever remote protocol is used to connect to the device to manually log in and test the fence device or see what output is given. For example, if you will be configuring fencing for an IPMI-enabled device, then try to log in remotely with **ipmitool**. Take note of the options used when logging in manually because those options might be needed when using the fencing agent.
   If you are unable to log in to the fence device, verify that the device is pingable, there is nothing such as a firewall configuration that is preventing access to the fence device, remote access is enabled on the fencing device, and the credentials are correct.

2. Run the fence agent manually, using the fence agent script. This does not require that the cluster services are running, so you can perform this step before the device is configured in the cluster. This can ensure that the fence device is responding properly before proceeding.

NOTE

The examples in this section use the **fence_ipmilan** fence agent script for an iLO device. The actual fence agent you will use and the command that calls that agent will depend on your server hardware. You should consult the man page for the fence agent you are using to determine which options to specify. You will usually need to know the login and password for the fence device and other information related to the fence device.

The following example shows the format you would use to run the **fence_ipmilan** fence agent script with **-o status** parameter to check the status of the fence device interface on another node without actually fencing it. This allows you to test the device and get it working before attempting to reboot the node. When running this command, you specify the name and password of an iLO user that has power on and off permissions for the iLO device.

```
# fence_ipmilan -a ipaddress -l username -p password -o status
```

The following example shows the format you would use to run the **fence_ipmilan** fence agent script with the **-o reboot** parameter. Running this command on one node reboots the node managed by this iLO device.

```
# fence_ipmilan -a ipaddress -l username -p password -o reboot
```

If the fence agent failed to properly do a status, off, on, or reboot action, you should check the hardware, the configuration of the fence device, and the syntax of your commands. In addition, you can run the fence agent script with the debug output enabled. The debug output is useful for some fencing agents to see where in the sequence of events the fencing agent script is failing when logging into the fence device.

```
# fence_ipmilan -a ipaddress -l username -p password -o status -D /tmp/$(hostname)-fence_agent.debug
```

When diagnosing a failure that has occurred, you should ensure that the options you specified when manually logging in to the fence device are identical to what you passed on to the fence agent with the fence agent script.

For fence agents that support an encrypted connection, you may see an error due to certificate validation failing, requiring that you trust the host or that you use the fence agent's **ssl-insecure** parameter. Similarly, if SSL/TLS is disabled on the target device, you may need to account for this when setting the SSL parameters for the fence agent.

NOTE

If the fence agent that is being tested is a **fence_drac**, **fence_ilo**, or some other fencing agent for a systems management device that continues to fail, then fall back to trying **fence_ipmilan**. Most systems management cards support IPMI remote login and the only supported fencing agent is **fence_ipmilan**.

3. Once the fence device has been configured in the cluster with the same options that worked manually and the cluster has been started, test fencing with the **pcs stonith fence** command from any node (or even multiple times from different nodes), as in the following example. The **pcs stonith fence** command reads the cluster configuration from the CIB and calls the fence agent as configured to execute the fence action. This verifies that the cluster configuration is correct.

```
# pcs stonith fence node_name
```

If the **pcs stonith fence** command works properly, that means the fencing configuration for the cluster should work when a fence event occurs. If the command fails, it means that cluster management cannot invoke the fence device through the configuration it has retrieved. Check for the following issues and update your cluster configuration as needed.

- Check your fence configuration. For example, if you have used a host map you should ensure that the system can find the node using the host name you have provided.
- Check whether the password and user name for the device include any special characters that could be misinterpreted by the bash shell. Making sure that you enter passwords and user names surrounded by quotation marks could address this issue.
- Check whether you can connect to the device using the exact IP address or host name you specified in the **pcs stonith** command. For example, if you give the host name in the stonith command but test by using the IP address, that is not a valid test.
- If the protocol that your fence device uses is accessible to you, use that protocol to try to connect to the device. For example many agents use ssh or telnet. You should try to connect to the device with the credentials you provided when configuring the device, to see if you get a valid prompt and can log in to the device.
  If you determine that all your parameters are appropriate but you still have trouble connecting to your fence device, you can check the logging on the fence device itself, if the device provides that, which will show if the user has connected and what command the user issued. You can also search through the **/var/log/messages** file for instances of stonith and error, which could give some idea of what is transpiring, but some agents can provide additional information.

4. Once the fence device tests are working and the cluster is up and running, test an actual failure. To do this, take an action in the cluster that should initiate a token loss.
   - Take down a network. How you take a network depends on your specific configuration. In many cases, you can physically pull the network or power cables out of the host. For information on simulating a network failure, see What is the proper way to simulate a network failure on a RHEL Cluster?.

NOTE

Disabling the network interface on the local host rather than physically disconnecting the network or power cables is not recommended as a test of fencing because it does not accurately simulate a typical real-world failure.

- Block corosync traffic both inbound and outbound using the local firewall.
  The following example blocks corosync, assuming the default corosync port is used, **firewalld** is used as the local firewall, and the network interface used by corosync is in the default firewall zone:

```
# firewall-cmd --direct --add-rule ipv4 filter OUTPUT 2 -p udp --dport=5405 -j DROP
# firewall-cmd --add-rich-rule='rule family="ipv4" port port="5405" protocol="udp" drop'
```

- Simulate a crash and panic your machine with **sysrq-trigger**. Note, however, that triggering a kernel panic can cause data loss; it is recommended that you disable your cluster resources first.

```
# echo c > /proc/sysrq-trigger
```

8.6. CONFIGURING FENCING LEVELS

Pacemaker supports fencing nodes with multiple devices through a feature called fencing topologies. To implement topologies, create the individual devices as you normally would and then define one or more fencing levels in the fencing topology section in the configuration.

- Each level is attempted in ascending numeric order, starting at 1.
- If a device fails, processing terminates for the current level. No further devices in that level are exercised and the next level is attempted instead.
- If all devices are successfully fenced, then that level has succeeded and no other levels are tried.
- The operation is finished when a level has passed (success), or all levels have been attempted (failed).

Use the following command to add a fencing level to a node. The devices are given as a comma-separated list of stonith ids, which are attempted for the node at that level.

```
pcs stonith level add level node devices
```

The following command lists all of the fencing levels that are currently configured.

```
pcs stonith level
```

In the following example, there are two fence devices configured for node **rh7-2**: an ilo fence device called **my_ilo** and an apc fence device called **my_apc**. These commands set up fence levels so that if the device **my_ilo** fails and is unable to fence the node, then Pacemaker will attempt to use the device **my_apc**. This example also shows the output of the **pcs stonith level** command after the levels are configured.

```
# pcs stonith level add 1 rh7-2 my_ilo
```

```
# pcs stonith level add 2 rh7-2 my_apc
# pcs stonith level
 Node: rh7-2
  Level 1 - my_ilo
  Level 2 - my_apc
```

The following command removes the fence level for the specified node and devices. If no nodes or devices are specified then the fence level you specify is removed from all nodes.

```
pcs stonith level remove level [node_id] [stonith_id] ... [stonith_id]
```

The following command clears the fence levels on the specified node or stonith id. If you do not specify a node or stonith id, all fence levels are cleared.

```
pcs stonith level clear [node|stonith_id(s)]
```

If you specify more than one stonith id, they must be separated by a comma and no spaces, as in the following example.

```
# pcs stonith level clear dev_a,dev_b
```

The following command verifies that all fence devices and nodes specified in fence levels exist.

```
pcs stonith level verify
```

You can specify nodes in fencing topology by a regular expression applied on a node name and by a node attribute and its value. For example, the following commands configure nodes **node1**, **node2**, and **`node3** to use fence devices **apc1** and **`apc2**, and nodes **`node4**, **node5**, and **`node6** to use fence devices **apc3** and **`apc4**.

```
pcs stonith level add 1 "regexp%node[1-3]" apc1,apc2
pcs stonith level add 1 "regexp%node[4-6]" apc3,apc4
```

The following commands yield the same results by using node attribute matching.

```
pcs node attribute node1 rack=1
pcs node attribute node2 rack=1
pcs node attribute node3 rack=1
pcs node attribute node4 rack=2
pcs node attribute node5 rack=2
pcs node attribute node6 rack=2
pcs stonith level add 1 attrib%rack=1 apc1,apc2
pcs stonith level add 1 attrib%rack=2 apc3,apc4
```

8.7. CONFIGURING FENCING FOR REDUNDANT POWER SUPPLIES

When configuring fencing for redundant power supplies, the cluster must ensure that when attempting to reboot a host, both power supplies are turned off before either power supply is turned back on.

If the node never completely loses power, the node may not release its resources. This opens up the possibility of nodes accessing these resources simultaneously and corrupting them.

You need to define each device only once and to specify that both are required to fence the node, as in the following example.

```
# pcs stonith create apc1 fence_apc_snmp ipaddr=apc1.example.com login=user
passwd='7a4D#1j!pz864' pcmk_host_map="node1.example.com:1;node2.example.com:2"

# pcs stonith create apc2 fence_apc_snmp ipaddr=apc2.example.com login=user
passwd='7a4D#1j!pz864' pcmk_host_map="node1.example.com:1;node2.example.com:2"

# pcs stonith level add 1 node1.example.com apc1,apc2
# pcs stonith level add 1 node2.example.com apc1,apc2
```

8.8. DISPLAYING CONFIGURED FENCE DEVICES

The following command shows all currently configured fence devices. If a *stonith_id* is specified, the command shows the options for that configured stonith device only. If the **--full** option is specified, all configured stonith options are displayed.

```
pcs stonith config [stonith_id] [--full]
```

8.9. MODIFYING AND DELETING FENCE DEVICES

Use the following command to modify or add options to a currently configured fencing device.

```
pcs stonith update stonith_id [stonith_device_options]
```

Use the following command to remove a fencing device from the current configuration.

```
pcs stonith delete stonith_id
```

8.10. MANUALLY FENCING A CLUSTER NODE

You can fence a node manually with the following command. If you specify **--off** this will use the **off** API call to stonith which will turn the node off instead of rebooting it.

```
pcs stonith fence node [--off]
```

In a situation where no stonith device is able to fence a node even if it is no longer active, the cluster may not be able to recover the resources on the node. If this occurs, after manually ensuring that the node is powered down you can enter the following command to confirm to the cluster that the node is powered down and free its resources for recovery.

WARNING

If the node you specify is not actually off, but running the cluster software or services normally controlled by the cluster, data corruption/cluster failure will occur.

-

```
pcs stonith confirm node
```

8.11. DISABLING A FENCE DEVICE

To disable a fencing device/resource, you run the **pcs stonith disable** command.

The following command disables the fence device **myapc**.

```
# pcs stonith disable myapc
```

8.12. PREVENTING A NODE FROM USING A FENCE DEVICE

To prevent a specific node from using a fencing device, you can configure location constraints for the fencing resource.

The following example prevents fence device **node1-ipmi** from running on **node1**.

```
# pcs constraint location node1-ipmi avoids node1
```

8.13. CONFIGURING ACPI FOR USE WITH INTEGRATED FENCE DEVICES

If your cluster uses integrated fence devices, you must configure ACPI (Advanced Configuration and Power Interface) to ensure immediate and complete fencing.

If a cluster node is configured to be fenced by an integrated fence device, disable ACPI Soft-Off for that node. Disabling ACPI Soft-Off allows an integrated fence device to turn off a node immediately and completely rather than attempting a clean shutdown (for example, **shutdown -h now**). Otherwise, if ACPI Soft-Off is enabled, an integrated fence device can take four or more seconds to turn off a node (see the note that follows). In addition, if ACPI Soft-Off is enabled and a node panics or freezes during shutdown, an integrated fence device may not be able to turn off the node. Under those circumstances, fencing is delayed or unsuccessful. Consequently, when a node is fenced with an integrated fence device and ACPI Soft-Off is enabled, a cluster recovers slowly or requires administrative intervention to recover.

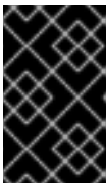
NOTE

The amount of time required to fence a node depends on the integrated fence device used. Some integrated fence devices perform the equivalent of pressing and holding the power button; therefore, the fence device turns off the node in four to five seconds. Other integrated fence devices perform the equivalent of pressing the power button momentarily, relying on the operating system to turn off the node; therefore, the fence device turns off the node in a time span much longer than four to five seconds.

- The preferred way to disable ACPI Soft-Off is to change the BIOS setting to "instant-off" or an equivalent setting that turns off the node without delay, as described in Section 8.13.1, “Disabling ACPI Soft-Off with the BIOS”.

Disabling ACPI Soft-Off with the BIOS may not be possible with some systems. If disabling ACPI Soft-Off with the BIOS is not satisfactory for your cluster, you can disable ACPI Soft-Off with one of the following alternate methods:

- Setting **HandlePowerKey=ignore** in the **/etc/systemd/logind.conf** file and verifying that the node node turns off immediately when fenced, as described in Section 8.13.2, "Disabling ACPI Soft-Off in the logind.conf file". This is the first alternate method of disabling ACPI Soft-Off.
- Appending **acpi=off** to the kernel boot command line, as described in Section 8.13.3, "Disabling ACPI completely in the GRUB 2 File". This is the second alternate method of disabling ACPI Soft-Off, if the preferred or the first alternate method is not available.



IMPORTANT

This method completely disables ACPI; some computers do not boot correctly if ACPI is completely disabled. Use this method *only* if the other methods are not effective for your cluster.

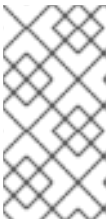
8.13.1. Disabling ACPI Soft-Off with the BIOS

You can disable ACPI Soft-Off by configuring the BIOS of each cluster node with the following procedure.

NOTE

The procedure for disabling ACPI Soft-Off with the BIOS may differ among server systems. You should verify this procedure with your hardware documentation.

1. Reboot the node and start the **BIOS CMOS Setup Utility** program.
2. Navigate to the Power menu (or equivalent power management menu).
3. At the Power menu, set the **Soft-Off by PWR-BTTN** function (or equivalent) to **Instant-Off** (or the equivalent setting that turns off the node by means of the power button without delay). **BIOS CMOS Setup Utility**: shows a Power menu with **ACPI Function** set to **Enabled** and **Soft-Off by PWR-BTTN** set to **Instant-Off**.



NOTE

The equivalents to **ACPI Function**, **Soft-Off by PWR-BTTN**, and **Instant-Off** may vary among computers. However, the objective of this procedure is to configure the BIOS so that the computer is turned off by means of the power button without delay.

4. Exit the **BIOS CMOS Setup Utility** program, saving the BIOS configuration.
5. Verify that the node turns off immediately when fenced. For information on testing a fence device, see Testing a fence device.

BIOS CMOS Setup Utility:

```
`Soft-Off by PWR-BTTN` set to
`Instant-Off`
```

```
+---------------------------------------------|-------------------+
|   ACPI Function             [Enabled]       |   Item Help       |
|   ACPI Suspend Type         [S1(POS)]       |-------------------|
```

```
| x Run VGABIOS if S3 Resume   Auto          |   Menu Level   * |
|   Suspend Mode             [Disabled]     |                  |
|   HDD Power Down           [Disabled]     |                  |
|   Soft-Off by PWR-BTTN     [Instant-Off   |                  |
|   CPU THRM-Throttling      [50.0%]        |                  |
|   Wake-Up by PCI card      [Enabled]      |                  |
|   Power On by Ring         [Enabled]      |                  |
|   Wake Up On LAN           [Enabled]      |                  |
| x USB KB Wake-Up From S3    Disabled      |                  |
|   Resume by Alarm          [Disabled]     |                  |
| x  Date(of Month) Alarm     0             |                  |
| x  Time(hh:mm:ss) Alarm     0 :  0 :      |                  |
|   POWER ON Function        [BUTTON ONLY   |                  |
| x KB Power ON Password      Enter         |                  |
| x Hot Key Power ON         Ctrl-F1        |                  |
|                                  |                  |
|                                  |                  |
+----------------------------------------|------------------+
```

This example shows **ACPI Function** set to **Enabled**, and **Soft-Off by PWR-BTTN** set to **Instant-Off**.

8.13.2. Disabling ACPI Soft-Off in the logind.conf file

To disable power-key handing in the **/etc/systemd/logind.conf** file, use the following procedure.

1. Define the following configuration in the **/etc/systemd/logind.conf** file:

   ```
   HandlePowerKey=ignore
   ```

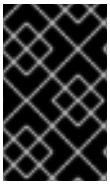
2. Reload the **systemd** configuration:

   ```
   # systemctl daemon-reload
   ```

3. Verify that the node turns off immediately when fenced. For information on testing a fence device, see Testing a fence device.

8.13.3. Disabling ACPI completely in the GRUB 2 File

You can disable ACPI Soft-Off by appending **acpi=off** to the GRUB menu entry for a kernel.



IMPORTANT

This method completely disables ACPI; some computers do not boot correctly if ACPI is completely disabled. Use this method *only* if the other methods are not effective for your cluster.

Use the following procedure to disable ACPI in the GRUB 2 file:

1. Use the **--args** option in combination with the **--update-kernel** option of the **grubby** tool to change the **grub.cfg** file of each cluster node as follows:

   ```
   # grubby --args=acpi=off --update-kernel=ALL
   ```

2. Reboot the node.

3. Verify that the node turns off immediately when fenced. For information on testing a fence device, see Testing a fence device.

CHAPTER 9. CONFIGURING CLUSTER RESOURCES

The format for the command to create a cluster resource is as follows:

```
pcs resource create resource_id [standard:[provider:]]type [resource_options] [op operation_action operation_options [operation_action operation options]...] [meta meta_options...] [clone [clone_options] | master [master_options] | --group group_name [--before resource_id | --after resource_id] | [bundle bundle_id] [--disabled] [--wait[=n]]
```

Key cluster resource creation options include the following:

- When you specify the **--group** option, the resource is added to the resource group named. If the group does not exist, this creates the group and adds this resource to the group.
- The **--before** and **--after** options specify the position of the added resource relative to a resource that already exists in a resource group.
- Specifying the **--disabled** option indicates that the resource is not started automatically.

You can determine the behavior of a resource in a cluster by configuring constraints for that resource.

Resource creation examples

The following command creates a resource with the name **VirtualIP** of standard **ocf**, provider **heartbeat**, and type **IPaddr2**. The floating address of this resource is 192.168.0.120, and the system will check whether the resource is running every 30 seconds.

```
# pcs resource create VirtualIP ocf:heartbeat:IPaddr2 ip=192.168.0.120 cidr_netmask=24 op monitor interval=30s
```

Alternately, you can omit the *standard* and *provider* fields and use the following command. This will default to a standard of **ocf** and a provider of **heartbeat**.

```
# pcs resource create VirtualIP IPaddr2 ip=192.168.0.120 cidr_netmask=24 op monitor interval=30s
```

Deleting a configured resource

Use the following command to delete a configured resource.

```
pcs resource delete resource_id
```

For example, the following command deletes an existing resource with a resource ID of **VirtualIP**.

```
# pcs resource delete VirtualIP
```

9.1. RESOURCE AGENT IDENTIFIERS

The identifiers that you define for a resource tell the cluster which agent to use for the resource, where to find that agent and what standards it conforms to. Table 9.1, “Resource Agent Identifiers”, describes these properties.

Table 9.1. Resource Agent Identifiers

Field	Description
standard	The standard the agent conforms to. Allowed values and their meaning: * **ocf** - The specified *type* is the name of an executable file conforming to the Open Cluster Framework Resource Agent API and located beneath **/usr/lib/ocf/resource.d/*provider*** * **lsb** - The specified *type* is the name of an executable file conforming to Linux Standard Base Init Script Actions. If the type does not specify a full path, the system will look for it in the **/etc/init.d** directory. * **systemd** - The specified *type* is the name of an installed **systemd** unit * **service** - Pacemaker will search for the specified *type*, first as an **lsb** agent, then as a **systemd** agent * **nagios** - The specified *type* is the name of an executable file conforming to the Nagios Plugin API and located in the **/usr/libexec/nagios/plugins** directory, with OCF-style metadata stored separately in the **/usr/share/nagios/plugins-metadata** directory (available in the **nagios-agents-metadata** package for certain common plugins).
type	The name of the resource agent you wish to use, for example **IPaddr** or **Filesystem**
provider	The OCF spec allows multiple vendors to supply the same resource agent. Most of the agents shipped by Red Hat use **heartbeat** as the provider.

Table 9.2, "Commands to Display Resource Properties" summarizes the commands that display the available resource properties.

Table 9.2. Commands to Display Resource Properties

pcs Display Command	Output
pcs resource list	Displays a list of all available resources.
pcs resource standards	Displays a list of available resource agent standards.
pcs resource providers	Displays a list of available resource agent providers.
pcs resource list *string*	Displays a list of available resources filtered by the specified string. You can use this command to display resources filtered by the name of a standard, a provider, or a type.

9.2. DISPLAYING RESOURCE-SPECIFIC PARAMETERS

For any individual resource, you can use the following command to display a description of the resource, the parameters you can set for that resource, and the default values that are set for the resource.

```
pcs resource describe [standard:[provider:]]type
```

For example, the following command displays information for a resource of type **apache**.

```
# pcs resource describe ocf:heartbeat:apache
This is the resource agent for the Apache Web server.
This resource agent operates both version 1.x and version 2.x Apache
servers.

...
```

9.3. CONFIGURING RESOURCE META OPTIONS

In addition to the resource-specific parameters, you can configure additional resource options for any resource. These options are used by the cluster to decide how your resource should behave.

Table 9.3, "Resource Meta Options" describes the resource meta options.

Table 9.3. Resource Meta Options

Field	Default	Description
priority	**0**	If not all resources can be active, the cluster will stop lower priority resources in order to keep higher priority ones active.
target-role	**Started**	What state should the cluster attempt to keep this resource in? Allowed values: * *Stopped* - Force the resource to be stopped * *Started* - Allow the resource to be started (and in the case of promotable clones, promoted to master role if appropriate) * *Master* - Allow the resource to be started and, if appropriate, promoted * *Slave* - Allow the resource to be started, but only in Slave mode if the resource is promotable
is-managed	**true**	Is the cluster allowed to start and stop the resource? Allowed values: **true**, **false**

Field	Default	Description
resource-stickiness	0	Value to indicate how much the resource prefers to stay where it is.
requires	Calculated	Indicates under what conditions the resource can be started. Defaults to **fencing** except under the conditions noted below. Possible values: * **nothing** - The cluster can always start the resource. * **quorum** - The cluster can only start this resource if a majority of the configured nodes are active. This is the default value if **stonith-enabled** is **false** or the resource's **standard** is **stonith**. * **fencing** - The cluster can only start this resource if a majority of the configured nodes are active *and* any failed or unknown nodes have been fenced. * **unfencing** - The cluster can only start this resource if a majority of the configured nodes are active *and* any failed or unknown nodes have been fenced *and* only on nodes that have been *unfenced*. This is the default value if the **provides=unfencing stonith** meta option has been set for a fencing device.
migration-threshold	**INFINITY**	How many failures may occur for this resource on a node, before this node is marked ineligible to host this resource. A value of 0 indicates that this feature is disabled (the node will never be marked ineligible); by contrast, the cluster treats **INFINITY** (the default) as a very large but finite number. This option has an effect only if the failed operation has **on-fail=restart** (the default), and additionally for failed start operations if the cluster property **start-failure-is-fatal** is **false**.

Field	Default	Description
failure-timeout	**0** (disabled)	Used in conjunction with the **migration-threshold** option, indicates how many seconds to wait before acting as if the failure had not occurred, and potentially allowing the resource back to the node on which it failed.
multiple-active	**stop_start**	What should the cluster do if it ever finds the resource active on more than one node. Allowed values: * **block** - mark the resource as unmanaged * **stop_only** - stop all active instances and leave them that way * **stop_start** - stop all active instances and start the resource in one location only

9.3.1. Changing the default value of a resource option

To change the default value of a resource option, use the following command.

```
pcs resource defaults options
```

For example, the following command resets the default value of **resource-stickiness** to 100.

```
# pcs resource defaults resource-stickiness=100
```

9.3.2. Displaying currently configured resource defaults

Omitting the *options* parameter from the **pcs resource defaults** displays a list of currently configured default values for resource options. The following example shows the output of this command after you have reset the default value of **resource-stickiness** to 100.

```
# pcs resource defaults
resource-stickiness: 100
```

9.3.3. Setting meta options on resource creation

Whether you have reset the default value of a resource meta option or not, you can set a resource option for a particular resource to a value other than the default when you create the resource. The following shows the format of the **pcs resource create** command you use when specifying a value for a resource meta option.

```
pcs resource create resource_id [standard:[provider:]]type [resource options] [meta meta_options...]
```

For example, the following command creates a resource with a **resource-stickiness** value of 50.

```
# pcs resource create VirtualIP ocf:heartbeat:IPaddr2 ip=192.168.0.120 meta resource-stickiness=50
```

You can also set the value of a resource meta option for an existing resource, group, cloned resource, or master resource with the following command.

```
pcs resource meta resource_id | group_id | clone_id meta_options
```

In the following example, there is an existing resource named **dummy_resource**. This command sets the **failure-timeout** meta option to 20 seconds, so that the resource can attempt to restart on the same node in 20 seconds.

```
# pcs resource meta dummy_resource failure-timeout=20s
```

After executing this command, you can display the values for the resource to verify that **failure-timeout=20s** is set.

```
# pcs resource config dummy_resource
 Resource: dummy_resource (class=ocf provider=heartbeat type=Dummy)
  Meta Attrs: failure-timeout=20s
  ...
```

9.4. CONFIGURING RESOURCE GROUPS

One of the most common elements of a cluster is a set of resources that need to be located together, start sequentially, and stop in the reverse order. To simplify this configuration, Pacemaker supports the concept of resource groups.

9.4.1. Creating a resource group

You create a resource group with the following command, specifying the resources to include in the group. If the group does not exist, this command creates the group. If the group exists, this command adds additional resources to the group. The resources will start in the order you specify them with this command, and will stop in the reverse order of their starting order.

```
pcs resource group add group_name resource_id [resource_id] ... [resource_id] [--before resource_id | --after resource_id]
```

You can use the **--before** and **--after** options of this command to specify the position of the added resources relative to a resource that already exists in the group.

You can also add a new resource to an existing group when you create the resource, using the following command. The resource you create is added to the group named *group_name*. If the group *group_name* does not exist, it will be created.

```
pcs resource create resource_id [standard:[provider:]]type [resource_options] [op operation_action operation_options] --group group_name
```

There is no limit to the number of resources a group can contain. The fundamental properties of a group are as follows.

- Resources are colocated within a group.
- Resources are started in the order in which you specify them. If a resource in the group cannot run anywhere, then no resource specified after that resource is allowed to run.
- Resources are stopped in the reverse order in which you specify them.

The following example creates a resource group named **shortcut** that contains the existing resources **IPaddr** and **Email**.

```
# pcs resource group add shortcut IPaddr Email
```

In this example:

- The **IPaddr** is started first, then **Email**.
- The **Email** resource is stopped first, then **IPAddr**.
- If **IPaddr** cannot run anywhere, neither can **Email**.
- If **Email** cannot run anywhere, however, this does not affect **IPaddr** in any way.

9.4.2. Removing a resource group

You remove a resource from a group with the following command. If there are no remaining resources in the group, this command removes the group itself.

```
pcs resource group remove group_name resource_id...
```

9.4.3. Displaying resource groups

The following command lists all currently configured resource groups.

```
pcs resource group list
```

9.4.4. Group options

You can set the following options for a resource group, and they maintain the same meaning as when they are set for a single resource: **priority**, **target-role**, **is-managed**. For information on resource meta options, see Configuring resource meta options.

9.4.5. Group stickiness

Stickiness, the measure of how much a resource wants to stay where it is, is additive in groups. Every active resource of the group will contribute its stickiness value to the group's total. So if the default **resource-stickiness** is 100, and a group has seven members, five of which are active, then the group as a whole will prefer its current location with a score of 500.

9.5. DETERMINING RESOURCE BEHAVIOR

You can determine the behavior of a resource in a cluster by configuring constraints for that resource. You can configure the following categories of constraints:

- **location** constraints — A location constraint determines which nodes a resource can run on. For information on configuring location constraints, see Determining which nodes a resource can run on.
- **order** constraints — An ordering constraint determines the order in which the resources run. For information on configuring ordering constraints, see Determining the order in which cluster resources are run.
- **colocation** constraints — A colocation constraint determines where resources will be placed relative to other resources. For information on colocation constraints, see Colocating cluster resources.

As a shorthand for configuring a set of constraints that will locate a set of resources together and ensure that the resources start sequentially and stop in reverse order, Pacemaker supports the concept of resource groups. After you have created a resource group, you can configure constraints on the group itself just as you configure constraints for individual resources. For information on resource groups, see Configuring resource groups.

CHAPTER 10. DETERMINING WHICH NODES A RESOURCE CAN RUN ON

Location constraints determine which nodes a resource can run on. You can configure location constraints to determine whether a resource will prefer or avoid a specified node.

10.1. CONFIGURING LOCATION CONSTRAINTS

You can configure a basic location constraint to specify whether a resource prefers or avoids a node, with an optional **score** value to indicate the relative degree of preference for the constraint.

The following command creates a location constraint for a resource to prefer the specified node or nodes. Note that it is possible to create constraints on a particular resource for more than one node with a single command.

```
pcs constraint location rsc prefers node[=score] [node[=score]] ...
```

The following command creates a location constraint for a resource to avoid the specified node or nodes.

```
pcs constraint location rsc avoids node[=score] [node[=score]] ...
```

Table 10.1, "Location Constraint Options" summarizes the meanings of the basic options for configuring location constraints.

Table 10.1. Location Constraint Options

Field	Description
rsc	A resource name
node	A node's name
score	Positive integer value to indicate the degree of preference for whether the given resource should prefer or avoid the given node. **INFINITY** is the default **score** value for a resource location constraint. A value of **INFINITY** for *score* in a command that configures a resource to prefer a node indicates that the resource will prefer that node if the node is available, but does not prevent the resource from running on another node if the specified node is unavailable. A value of **INFINITY** in a command that configures a resource to avoid a node indicates that the resource will never run on that node, even if no other node is available. A numeric score (that is, not **INFINITY**) means the constraint is optional, and will be honored unless some other factor outweighs it. For example, if the resource is already placed on a different node, and its **resource-stickiness** score is higher than a**prefers** location constraint's score, then the resource will be left where it is.

The following command creates a location constraint to specify that the resource **Webserver** prefers node **node1**.

```
pcs constraint location Webserver prefers node1
```

pcs supports regular expressions in location constraints on the command line. These constraints apply to multiple resources based on the regular expression matching resource name. This allows you to configure multiple location contraints with a single command line.

The following command creates a location constraint to specify that resources **dummy0** to **dummy9** prefer **node1**.

```
pcs constraint location 'regexp%dummy[0-9]' prefers node1
```

Since Pacemaker uses POSIX extended regular expressions as documented at http://pubs.opengroup.org/onlinepubs/9699919799/basedefs/V1_chap09.html#tag_09_04, you can specify the same constraint with the following command.

```
pcs constraint location 'regexp%dummy[[:digit:]]' prefers node1
```

10.2. LIMITING RESOURCE DISCOVERY TO A SUBSET OF NODES

Before Pacemaker starts a resource anywhere, it first runs a one-time monitor operation (often referred to as a "probe") on every node, to learn whether the resource is already running. This process of resource discovery can result in errors on nodes that are unable to execute the monitor.

When configuring a location constraint on a node, you can use the **resource-discovery** option of the **pcs constraint location** command to indicate a preference for whether Pacemaker should perform resource discovery on this node for the specified resource. Limiting resource discovery to a subset of nodes the resource is physically capable of running on can significantly boost performance when a large set of nodes is present. When **pacemaker_remote** is in use to expand the node count into the hundreds of nodes range, this option should be considered.

The following command shows the format for specifying the **resource-discovery** option of the **pcs constraint location** command. In this command, a positive value for *score* corresponds to a basic location constraint that configures a resource to prefer a node, while a negative value for *score* corresponds to a basic location`constraint that configures a resource to avoid a node. As with basic location constraints, you can use regular expressions for resources with these constraints as well.

```
pcs constraint location add id rsc node score [resource-discovery=option]
```

Table 10.2, “Resource Discovery Constraint Parameters” summarizes the meanings of the basic parameters for configuring constraints for resource discovery.

Table 10.2. Resource Discovery Constraint Parameters

Field	Description
id	A user-chosen name for the constraint itself.
rsc	A resource name
node	A node’s name

score	Integer value to indicate the degree of preference for whether the given resource should prefer or avoid the given node. A positive value for score corresponds to a basic location constraint that configures a resource to prefer a node, while a negative value for score corresponds to a basic location constraint that configures a resource to avoid a node. A value of **INFINITY** for *score* indicates that the resource will prefer that node if the node is available, but does not prevent the resource from running on another node if the specified node is unavailable. A value of **-INFINITY** in a command that configures a resource to avoid a node indicates that the resource will never run on that node, even if no other node is available. A numeric score (that is, not **INFINITY** or **-INFINITY**) means the constraint is optional, and will be honored unless some other factor outweighs it. For example, if the resource is already placed on a different node, and its **resource-stickiness** score is higher than a **prefers** location constraint's score, then the resource will be left where it is.
resource-discovery options	* **always** - Always perform resource discovery for the specified resource on this node. This is the default **resource-discovery** value for a resource location constraint. * **never** - Never perform resource discovery for the specified resource on this node. * **exclusive** - Perform resource discovery for the specified resource only on this node (and other nodes similarly marked as **exclusive**). Multiple location constraints using **exclusive** discovery for the same resource across different nodes creates a subset of nodes **resource-discovery** is exclusive to. If a resource is marked for **exclusive** discovery on one or more nodes, that resource is only allowed to be placed within that subset of nodes.

WARNING

Setting **resource-discovery** to **never** or **exclusive** removes Pacemaker's ability to detect and stop unwanted instances of a service running where it is not supposed to be. It is up to the system administrator to make sure that the service can never be active on nodes without resource discovery (such as by leaving the relevant software uninstalled).

10.3. CONFIGURING A LOCATION CONSTRAINT STRATEGY

When using location constraints, you can configure a general strategy for specifying which nodes a resource can run on:

- Opt-In Clusters — Configure a cluster in which, by default, no resource can run anywhere and then selectively enable allowed nodes for specific resources.
- Opt-Out Clusters — Configure a cluster in which, by default, all resources can run anywhere and then create location constraints for resources that are not allowed to run on specific nodes.

Whether you should choose to configure your cluster as an opt-in or opt-out cluster depends on both your personal preference and the make-up of your cluster. If most of your resources can run on most of the nodes, then an opt-out arrangement is likely to result in a simpler configuration. On the other hand, if most resources can only run on a small subset of nodes an opt-in configuration might be simpler.

10.3.1. Configuring an "Opt-In" Cluster

To create an opt-in cluster, set the **symmetric-cluster** cluster property to **false** to prevent resources from running anywhere by default.

```
# pcs property set symmetric-cluster=false
```

Enable nodes for individual resources. The following commands configure location constraints so that the resource **Webserver** prefers node **example-1**, the resource **Database** prefers node **example-2**, and both resources can fail over to node **example-3** if their preferred node fails. When configuring location constraints for an opt-in cluster, setting a score of zero allows a resource to run on a node without indicating any preference to prefer or avoid the node.

```
# pcs constraint location Webserver prefers example-1=200
# pcs constraint location Webserver prefers example-3=0
# pcs constraint location Database prefers example-2=200
# pcs constraint location Database prefers example-3=0
```

10.3.2. Configuring an "Opt-Out" Cluster

To create an opt-out cluster, set the **symmetric-cluster** cluster property to **true** to allow resources to run everywhere by default. This is the default configuration if **symmetric-cluster** is not set explicitly.

```
# pcs property set symmetric-cluster=true
```

The following commands will then yield a configuration that is equivalent to the example in Section 10.3.1, “Configuring an "Opt-In" Cluster”. Both resources can fail over to node **example-3** if their preferred node fails, since every node has an implicit score of 0.

```
# pcs constraint location Webserver prefers example-1=200
# pcs constraint location Webserver avoids example-2=INFINITY
# pcs constraint location Database avoids example-1=INFINITY
# pcs constraint location Database prefers example-2=200
```

Note that it is not necessary to specify a score of INFINITY in these commands, since that is the default value for the score.

CHAPTER 11. DETERMINING THE ORDER IN WHICH CLUSTER RESOURCES ARE RUN

To determine the order in which the resources run, you configure an ordering constraint.

The following shows the format for the command to configure an ordering constraint.

```
pcs constraint order [action] resource_id then [action] resource_id [options]
```

Table 11.1, "Properties of an Order Constraint" , summarizes the properties and options for configuring ordering constraints.

Table 11.1. Properties of an Order Constraint

Field	Description
resource_id	The name of a resource on which an action is performed.
action	The action to perform on a resource. Possible values of the *action* property are as follows: * **start** - Start the resource. * **stop** - Stop the resource. * **promote** - Promote the resource from a slave resource to a master resource. * **demote** - Demote the resource from a master resource to a slave resource. If no action is specified, the default action is **start**.
kind option	How to enforce the constraint. The possible values of the **kind** option are as follows: * **Optional** - Only applies if both resources are executing the specified action. For information on optional ordering, see Configuring advisory ordering. * **Mandatory** - Always (default value). If the first resource you specified is stopping or cannot be started, the second resource you specified must be stopped. For information on mandatory ordering, see Configuring mandatory ordering. * **Serialize** - Ensure that no two stop/start actions occur concurrently for the resources you specify. The first and second resource you specify can start in either order, but one must complete starting before the other can be started. A typical use case is when resource startup puts a high load on the host.

Field	Description
symmetrical option	If true, the reverse of the constraint applies for the opposite action (for example, if B starts after A starts, then B stops before Ordering constraints for which **kind** is **Serialize** cannot be symmetrical. The default value is **true** for **Mandatory** and **Ordering** kinds, **false** for **Serialize**.

Use the following command to remove resources from any ordering constraint.

```
pcs constraint order remove resource1 [resourceN]...
```

11.1. CONFIGURING MANDATORY ORDERING

A mandatory ordering constraint indicates that the second action should not be initiated for the second resource unless and until the first action successfully completes for the first resource. Actions that may be ordered are **stop**, **start**, and additionally for promotable clones, **demote** and **promote**. For example, "A then B" (which is equivalent to "start A then start B") means that B will not be started unless and until A successfully starts. An ordering constraint is mandatory if the **kind** option for the constraint is set to **Mandatory** or left as default.

If the **symmetrical** option is set to **true** or left to default, the opposite actions will be ordered in reverse. The **start** and **stop** actions are opposites, and **demote** and **promote** are opposites. For example, a symmetrical "promote A then start B" ordering implies "stop B then demote A", which means that A cannot be demoted until and unless B successfully stops. A symmetrical ordering means that changes in A's state can cause actions to be scheduled for B. For example, given "A then B", if A restarts due to failure, B will be stopped first, then A will be stopped, then A will be started, then B will be started.

Note that the cluster reacts to each state change. If the first resource is restarted and is in a started state again before the second resource initiated a stop operation, the second resource will not need to be restarted.

11.2. CONFIGURING ADVISORY ORDERING

When the **kind=Optional** option is specified for an ordering constraint, the constraint is considered optional and only applies if both resources are executing the specified actions. Any change in state by the first resource you specify will have no effect on the second resource you specify.

The following command configures an advisory ordering constraint for the resources named **VirtualIP** and **dummy_resource**.

```
# pcs constraint order VirtualIP then dummy_resource kind=Optional
```

11.3. CONFIGURING ORDERED RESOURCE SETS

A common situation is for an administrator to create a chain of ordered resources, where, for example, resource A starts before resource B which starts before resource C. If your configuration requires that you create a set of resources that is colocated and started in order, you can configure a resource group that contains those resources, as described in Configuring resource groups.

There are some situations, however, where configuring the resources that need to start in a specified order as a resource group is not appropriate:

- You may need to configure resources to start in order and the resources are not necessarily colocated.
- You may have a resource C that must start after either resource A or B has started but there is no relationship between A and B.
- You may have resources C and D that must start after both resources A and B have started, but there is no relationship between A and B or between C and D.

In these situations, you can create an ordering constraint on a set or sets of resources with the **pcs constraint order set** command.

You can set the following options for a set of resources with the **pcs constraint order set** command.

- **sequential**, which can be set to **true** or **false** to indicate whether the set of resources must be ordered relative to each other. The default value is **true**.
  Setting **sequential** to **false** allows a set to be ordered relative to other sets in the ordering constraint, without its members being ordered relative to each other. Therefore, this option makes sense only if multiple sets are listed in the constraint; otherwise, the constraint has no effect.
- **require-all**, which can be set to **true** or **false** to indicate whether all of the resources in the set must be active before continuing. Setting **require-all** to **false** means that only one resource in the set needs to be started before continuing on to the next set. Setting **require-all** to **false** has no effect unless used in conjunction with unordered sets, which are sets for which **sequential** is set to **false**. The default value is **true**.
- **action**, which can be set to **start**, **promote**, **demote** or **stop**, as described in Properties of an Order Constraint.
- **role**, which can be set to **Stopped**, **Started**, **Master**, or **Slave**.

You can set the following constraint options for a set of resources following the **setoptions** parameter of the **pcs constraint order set** command.

- **id**, to provide a name for the constraint you are defining.
- **kind**, which indicates how to enforce the constraint, as described in Properties of an Order Constraint.
- **symmetrical**, to set whether the reverse of the constraint applies for the opposite action, as described in Properties of an Order Constraint .

```
pcs constraint order set resource1 resource2 [resourceN]... [options] [set resourceX resourceY ...
[options]] [setoptions [constraint_options]]
```

If you have three resources named **D1**, **D2**, and **D3**, the following command configures them as an ordered resource set.

```
# pcs constraint order set D1 D2 D3
```

If you have six resources named **A**, **B**, **C**, **D**, **E**, and **F**, this example configures an ordering constraint for the set of resources that will start as follows:

- **A** and **B** start independently of each other
- **C** starts once either **A** or **B** has started
- **D** starts once **C** has started
- **E** and **F** start independently of each other once **D** has started

Stopping the resources is not influenced by this constraint since **symmetrical=false** is set.

```
# pcs constraint order set A B sequential=false require-all=false set C D set E F
sequential=false setoptions symmetrical=false
```

CHAPTER 12. COLOCATING CLUSTER RESOURCES

To specify that the location of one resource depends on the location of another resource, you configure a colocation constraint.

There is an important side effect of creating a colocation constraint between two resources: it affects the order in which resources are assigned to a node. This is because you cannot place resource A relative to resource B unless you know where resource B is. So when you are creating colocation constraints, it is important to consider whether you should colocate resource A with resource B or resource B with resource A.

Another thing to keep in mind when creating colocation constraints is that, assuming resource A is colocated with resource B, the cluster will also take into account resource A's preferences when deciding which node to choose for resource B.

The following command creates a colocation constraint.

> pcs constraint colocation add [master|slave] *source_resource* with [master|slave] *target_resource* [*score*] [*options*]

Table 12.1, "Properties of a Colocation Constraint" , summarizes the properties and options for configuring colocation constraints.

Table 12.1. Properties of a Colocation Constraint

Field	Description
source_resource	The colocation source. If the constraint cannot be satisfied, the cluster may decide not to allow the resource to run at all.
target_resource	The colocation target. The cluster will decide where to put this resource first and then decide where to put the source resource.
score	Positive values indicate the resource should run on the same node. Negative values indicate the resources should not run on the same node. A value of +**INFINITY**, the default value, indicates that the *source_resource* must run on the same node as the *target_resource*. A value of -**INFINITY** indicates that the *source_resource* must not run on the same node as the *target_resource*.

12.1. SPECIFYING MANDATORY PLACEMENT OF RESOURCES

Mandatory placement occurs any time the constraint's score is **+INFINITY** or **-INFINITY**. In such cases, if the constraint cannot be satisfied, then the *source_resource* is not permitted to run. For **score=INFINITY**, this includes cases where the *target_resource* is not active.

If you need **myresource1** to always run on the same machine as **myresource2**, you would add the following constraint:

> **# pcs constraint colocation add myresource1 with myresource2 score=INFINITY**

Because **INFINITY** was used, if **myresource2** cannot run on any of the cluster nodes (for whatever reason) then **myresource1** will not be allowed to run.

Alternatively, you may want to configure the opposite, a cluster in which **myresource1** cannot run on the same machine as **myresource2**. In this case use **score=-INFINITY**

```
# pcs constraint colocation add myresource1 with myresource2 score=-INFINITY
```

Again, by specifying **-INFINITY**, the constraint is binding. So if the only place left to run is where **myresource2** already is, then **myresource1** may not run anywhere.

12.2. SPECIFYING ADVISORY PLACEMENT OF RESOURCES

If mandatory placement is about "must" and "must not", then advisory placement is the "I would prefer if" alternative. For constraints with scores greater than **-INFINITY** and less than **INFINITY**, the cluster will try to accommodate your wishes but may ignore them if the alternative is to stop some of the cluster resources. Advisory colocation constraints can combine with other elements of the configuration to behave as if they were mandatory.

12.3. COLOCATING SETS OF RESOURCES

If your configuration requires that you create a set of resources that are colocated and started in order, you can configure a resource group that contains those resources, as described in Configuring resource groups. There are some situations, however, where configuring the resources that need to be colocated as a resource group is not appropriate:

- You may need to colocate a set of resources but the resources do not necessarily need to start in order.
- You may have a resource C that must be colocated with either resource A or B, but there is no relationship between A and B.
- You may have resources C and D that must be colocated with both resources A and B, but there is no relationship between A and B or between C and D.

In these situations, you can create a colocation constraint on a set or sets of resources with the **pcs constraint colocation set** command.

You can set the following options for a set of resources with the **pcs constraint colocation set** command.

- **sequential**, which can be set to **true** or **false** to indicate whether the members of the set must be colocated with each other.
  Setting **sequential** to **false** allows the members of this set to be colocated with another set listed later in the constraint, regardless of which members of this set are active. Therefore, this option makes sense only if another set is listed after this one in the constraint; otherwise, the constraint has no effect.
- **role**, which can be set to **Stopped**, **Started**, **Master**, or **Slave**.

You can set the following constraint option for a set of resources following the **setoptions** parameter of the **pcs constraint colocation set** command.

- **id**, to provide a name for the constraint you are defining.

- **score**, to indicate the degree of preference for this constraint. For information on this option, see Location Constraint Options.

When listing members of a set, each member is colocated with the one before it. For example, "set A B" means "B is colocated with A". However, when listing multiple sets, each set is colocated with the one after it. For example, "set C D sequential=false set A B" means "set C D (where C and D have no relation between each other) is colocated with set A B (where B is colocated with A)".

The following command creates a colocation constraint on a set or sets of resources.

```
pcs constraint colocation set resource1 resource2 [resourceN]... [options] [set resourceX resourceY ... [options]] [setoptions [constraint_options]]
```

12.4. REMOVING COLOCATION CONSTRAINTS

Use the following command to remove colocation constraints with *source_resource*.

```
pcs constraint colocation remove source_resource target_resource
```

CHAPTER 13. DISPLAYING RESOURCE CONSTRAINTS

There are a several commands you can use to display constraints that have been configured.

13.1. DISPLAYING ALL CONFIGURED CONSTRAINTS

The following command lists all current location, order, and colocation constraints.

```
pcs constraint list|show
```

13.2. DISPLAYING LOCATION CONSTRAINTS

The following command lists all current location constraints.

- If **resources** is specified, location constraints are displayed per resource. This is the default behavior.
- If **nodes** is specified, location constraints are displayed per node.
- If specific resources or nodes are specified, then only information about those resources or nodes is displayed.

```
pcs constraint location [show [resources [resource...]] | [nodes [node...]]] [--full]
```

13.3. DISPLAYING ORDERING CONSTRAINTS

The following command lists all current ordering constraints. If the **--full** option is specified, show the internal constraint IDs.

```
pcs constraint order show [--full]
```

13.4. DISPLAYING COLOCATION CONSTRAINTS

The following command lists all current colocation constraints. If the **--full** option is specified, show the internal constraint IDs.

```
pcs constraint colocation show [--full]
```

13.5. DISPLAYING RESOURCE-SPECIFIC CONSTRAINTS

The following command lists the constraints that reference specific resources.

```
pcs constraint ref resource ...
```

CHAPTER 14. DETERMINING RESOURCE LOCATION WITH RULES

For more complicated location constraints, you can use Pacemaker rules to determine a resource's location.

14.1. PACEMAKER RULES

Rules can be used to make your configuration more dynamic. One use of rules might be to assign machines to different processing groups (using a node attribute) based on time and to then use that attribute when creating location constraints.

Each rule can contain a number of expressions, date-expressions and even other rules. The results of the expressions are combined based on the rule's **boolean-op** field to determine if the rule ultimately evaluates to **true** or **false**. What happens next depends on the context in which the rule is being used.

Table 14.1. Properties of a Rule

Field	Description
role	Limits the rule to apply only when the resource is in that role. Allowed values: **Started**, **Slave,** and **Master**. NOTE: A rule with **role="Master"** cannot determine the initial location of a clone instance. It will only affect which of the active instances will be promoted.
score	The score to apply if the rule evaluates to **true**. Limited to use in rules that are part of location constraints.
score-attribute	The node attribute to look up and use as a score if the rule evaluates to **true**. Limited to use in rules that are part of location constraints.
boolean-op	How to combine the result of multiple expression objects. Allowed values: **and** and **or**. The default value is **and**.

14.1.1. Node attribute expressions

Node attribute expressions are used to control a resource based on the attributes defined by a node or nodes.

Table 14.2. Properties of an Expression

Field	Description
attribute	The node attribute to test

Field	Description
type	Determines how the value(s) should be tested. Allowed values: **string**, **integer**, **version**. The default value is **string**.
operation	The comparison to perform. Allowed values: * **lt** - True if the node attribute's value is less than **value** * **gt** - True if the node attribute's value is greater than **value** * **lte** - True if the node attribute's value is less than or equal to **value** * **gte** - True if the node attribute's value is greater than or equal to **value** * **eq** - True if the node attribute's value is equal to **value** * **ne** - True if the node attribute's value is not equal to **value** * **defined** - True if the node has the named attribute * **not_defined** - True if the node does not have the named attribute
value	User supplied value for comparison (required unless **operation** is **defined** or **not_defined**)

In addition to any attributes added by the administrator, the cluster defines special, built-in node attributes for each node that can also be used, as described in Table 14.3, "Built-in Node Attributes" .

Table 14.3. Built-in Node Attributes

Name	Description
#uname	Node name
#id	Node ID

Name	Description
#kind	Node type. Possible values are **cluster**, **remote**, and **container**. The value of **kind** is **remote** for Pacemaker Remote nodes created with the **ocf:pacemaker:remote** resource, and **container** for Pacemaker Remote guest nodes and bundle nodes.
#is_dc	**true** if this node is a Designated Controller (DC), **false** otherwise
#cluster_name	The value of the **cluster-name** cluster property, if set
#site_name	The value of the **site-name** node attribute, if set, otherwise identical to **#cluster-name**
#role	The role the relevant promotable clone has on this node. Valid only within a rule for a location constraint for a promotable clone.

14.1.2. Time/date based expressions

Date expressions are used to control a resource or cluster option based on the current date/time. They can contain an optional date specification.

Table 14.4. Properties of a Date Expression

Field	Description
start	A date/time conforming to the ISO8601 specification.
end	A date/time conforming to the ISO8601 specification.
operation	Compares the current date/time with the start or the end date or both the start and end date, depending on the context. Allowed values: * **gt** - True if the current date/time is after **start** * **lt** - True if the current date/time is before **end** * **in_range** - True if the current date/time is after **start** and before **end** * **date-spec** - performs a cron-like comparison to the current date/time

14.1.3. Date specifications

Date specifications are used to create cron-like expressions relating to time. Each field can contain a single number or a single range. Instead of defaulting to zero, any field not supplied is ignored.

For example, **monthdays="1"** matches the first day of every month and **hours="09-17"** matches the hours between 9 am and 5 pm (inclusive). However, you cannot specify **weekdays="1,2"** or **weekdays="1-2,5-6"** since they contain multiple ranges.

Table 14.5. Properties of a Date Specification

Field	Description
id	A unique name for the date
hours	Allowed values: 0-23
monthdays	Allowed values: 0-31 (depending on month and year)
weekdays	Allowed values: 1-7 (1=Monday, 7=Sunday)
yeardays	Allowed values: 1-366 (depending on the year)
months	Allowed values: 1-12
weeks	Allowed values: 1-53 (depending on **weekyear**)
years	Year according the Gregorian calendar
weekyears	May differ from Gregorian years; for example, **2005-001 Ordinal** is also **2005-01-01 Gregorian** is also **2004-W53-6 Weekly**
moon	Allowed values: 0-7 (0 is new, 4 is full moon).

14.2. CONFIGURING A PACEMAKER LOCATION CONSTRAINT USING RULES

Use the following command to configure a Pacemaker constraint that uses rules. If **score** is omitted, it defaults to INFINITY. If **resource-discovery** is omitted, it defaults to **always**.

For information on the **resource-discovery** option, see Limiting resource discovery to a subset of nodes.

As with basic location constraints, you can use regular expressions for resources with these constraints as well.

When using rules to configure location constraints, the value of **score** can be positive or negative, with a positive value indicating "prefers" and a negative value indicating "avoids".

```
pcs constraint location rsc rule [resource-discovery=option] [role=master|slave] [score=score | score-attribute=attribute] expression
```

The *expression* option can be one of the following where *duration_options* and *date_spec_options* are: hours, monthdays, weekdays, yeardays, months, weeks, years, weekyears, moon as described in Properties of a Date Specification.

- **defined|not_defined *attribute***
- ***attribute* lt|gt|lte|gte|eq|ne [string|integer|version] *value***
- **date gt|lt *date***
- **date in_range *date* to *date***
- **date in_range *date* to duration *duration_options* …**
- **date-spec *date_spec_options***
- ***expression* and|or *expression***
- **(*expression*)**

Note that durations are an alternative way to specify an end for **in_range** operations by means of calculations. For example, you can specify a duration of 19 months.

The following location constraint configures an expression that is true if now is any time in the year 2018.

```
# pcs constraint location Webserver rule score=INFINITY date-spec years=2018
```

The following command configures an expression that is true from 9 am to 5 pm, Monday through Friday. Note that the hours value of 16 matches up to 16:59:59, as the numeric value (hour) still matches.

```
# pcs constraint location Webserver rule score=INFINITY date-spec hours="9-16" weekdays="1-5"
```

The following command configures an expression that is true when there is a full moon on Friday the thirteenth.

```
# pcs constraint location Webserver rule date-spec weekdays=5 monthdays=13 moon=4
```

To remove a rule, use the following command. If the rule that you are removing is the last rule in its constraint, the constraint will be removed.

```
pcs constraint rule remove rule_id
```

CHAPTER 15. MANAGING CLUSTER RESOURCES

This section describes various commands you can use to manage cluster resources.

15.1. DISPLAYING CONFIGURED RESOURCES

To display a list of all configured resources, use the following command.

```
pcs resource status
```

For example, if your system is configured with a resource named **VirtualIP** and a resource named **WebSite**, the **pcs resource show** command yields the following output.

```
# pcs resource status
 VirtualIP (ocf::heartbeat:IPaddr2): Started
 WebSite (ocf::heartbeat:apache): Started
```

To display a list of all configured resources and the parameters configured for those resources, use the **--full** option of the **pcs resource config** command, as in the following example.

```
# pcs resource config
 Resource: VirtualIP (type=IPaddr2 class=ocf provider=heartbeat)
  Attributes: ip=192.168.0.120 cidr_netmask=24
  Operations: monitor interval=30s
 Resource: WebSite (type=apache class=ocf provider=heartbeat)
  Attributes: statusurl=http://localhost/server-status configfile=/etc/httpd/conf/httpd.conf
  Operations: monitor interval=1min
```

To display the configured parameters for a resource, use the following command.

```
pcs resource config resource_id
```

For example, the following command displays the currently configured parameters for resource **VirtualIP**.

```
# pcs resource config VirtualIP
 Resource: VirtualIP (type=IPaddr2 class=ocf provider=heartbeat)
  Attributes: ip=192.168.0.120 cidr_netmask=24
  Operations: monitor interval=30s
```

15.2. MODIFYING RESOURCE PARAMETERS

To modify the parameters of a configured resource, use the following command.

```
pcs resource update resource_id [resource_options]
```

The following sequence of commands show the initial values of the configured parameters for resource **VirtualIP**, the command to change the value of the **ip** parameter, and the values following the update command.

```
# pcs resource config VirtualIP
 Resource: VirtualIP (type=IPaddr2 class=ocf provider=heartbeat)
```

```
  Attributes: ip=192.168.0.120 cidr_netmask=24
  Operations: monitor interval=30s
# pcs resource update VirtualIP ip=192.169.0.120
# pcs resource config VirtualIP
 Resource: VirtualIP (type=IPaddr2 class=ocf provider=heartbeat)
  Attributes: ip=192.169.0.120 cidr_netmask=24
  Operations: monitor interval=30s
```

15.3. CLEARING FAILURE STATUS OF CLUSTER RESOURCES

If a resource has failed, a failure message appears when you display the cluster status. If you resolve that resource, you can clear that failure status with the **pcs resource cleanup** command. This command resets the resource status and **failcount**, telling the cluster to forget the operation history of a resource and re-detect its current state.

The following command cleans up the resource specified by *resource_id*.

```
pcs resource cleanup resource_id
```

If you do not specify a *resource_id*, this command resets the resource status and **failcount** for all resources.

The **pcs resource cleanup** command probes only the resources that display as a failed action. To probe all resources on all nodes you can enter the following command:

```
pcs resource refresh
```

By default, the **pcs resource refresh** command probes only the nodes where a resource's state is known. To probe all resources even if the state is not known, enter the following command:

```
pcs resource refresh --full
```

15.4. MOVING RESOURCES IN A CLUSTER

Pacemaker provides a variety of mechanisms for configuring a resource to move from one node to another and to manually move a resource when needed.

You can manually move resources in a cluster with the **pcs resource move** and **pcs resource relocate** commands, as described in Manually moving cluster resources.

In addition to these commands, you can also control the behavior of cluster resources by enabling, disabling, and banning resources, as described in Enabling, disabling, and banning cluster resources.

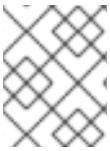
You can configure a resource so that it will move to a new node after a defined number of failures, and you can configure a cluster to move resources when external connectivity is lost.

15.4.1. Moving resources due to failure

When you create a resource, you can configure the resource so that it will move to a new node after a defined number of failures by setting the **migration-threshold** option for that resource. Once the threshold has been reached, this node will no longer be allowed to run the failed resource until:

- The administrator manually resets the resource's **failcount** using the **pcs resource cleanup** command.
- The resource's **failure-timeout** value is reached.

The value of **migration-threshold** is set to **INFINITY** by default. **INFINITY** is defined internally as a very large but finite number. A value of 0 disables the **migration-threshold** feature.



NOTE

Setting a **migration-threshold** for a resource is not the same as configuring a resource for migration, in which the resource moves to another location without loss of state.

The following example adds a migration threshold of 10 to the resource named **dummy_resource**, which indicates that the resource will move to a new node after 10 failures.

```
# pcs resource meta dummy_resource migration-threshold=10
```

You can add a migration threshold to the defaults for the whole cluster with the following command.

```
# pcs resource defaults migration-threshold=10
```

To determine the resource's current failure status and limits, use the **pcs resource failcount show** command.

There are two exceptions to the migration threshold concept; they occur when a resource either fails to start or fails to stop. If the cluster property **start-failure-is-fatal** is set to **true** (which is the default), start failures cause the **failcount** to be set to **INFINITY** and thus always cause the resource to move immediately.

Stop failures are slightly different and crucial. If a resource fails to stop and STONITH is enabled, then the cluster will fence the node in order to be able to start the resource elsewhere. If STONITH is not enabled, then the cluster has no way to continue and will not try to start the resource elsewhere, but will try to stop it again after the failure timeout.

15.4.2. Moving resources due to connectivity changes

Setting up the cluster to move resources when external connectivity is lost is a two step process.

1. Add a **ping** resource to the cluster. The **ping** resource uses the system utility of the same name to test if a list of machines (specified by DNS host name or IPv4/IPv6 address) are reachable and uses the results to maintain a node attribute called **pingd**.
2. Configure a location constraint for the resource that will move the resource to a different node when connectivity is lost.

Table 9.1, "Resource Agent Identifiers" describes the properties you can set for a **ping** resource.

Table 15.1. Properties of a ping resources

Field	Description

Field	Description
dampen	The time to wait (dampening) for further changes to occur. This prevents a resource from bouncing around the cluster when cluster nodes notice the loss of connectivity at slightly different times.
multiplier	The number of connected ping nodes gets multiplied by this value to get a score. Useful when there are multiple ping nodes configured.
host_list	The machines to contact in order to determine the current connectivity status. Allowed values include resolvable DNS host names, IPv4 and IPv6 addresses. The entries in the host list are space separated.

The following example command creates a **ping** resource that verifies connectivity to **gateway.example.com**. In practice, you would verify connectivity to your network gateway/router. You configure the **ping** resource as a clone so that the resource will run on all cluster nodes.

```
# pcs resource create ping ocf:pacemaker:ping dampen=5s multiplier=1000
host_list=gateway.example.com clone
```

The following example configures a location constraint rule for the existing resource named **Webserver**. This will cause the **Webserver** resource to move to a host that is able to ping **gateway.example.com** if the host that it is currently running on cannot ping **www.example.com**

```
# pcs constraint location Webserver rule score=-INFINITY pingd lt 1 or not_defined pingd
```

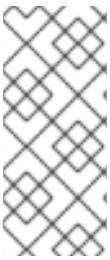
```
Module included in the following assemblies:
//
// <List assemblies here, each on a new line>
// rhel-8-docs/enterprise/assemblies/assembly_managing-cluster-resources.adoc
```

15.5. DISABLING A MONITOR OPERATION

The easiest way to stop a recurring monitor is to delete it. However, there can be times when you only want to disable it temporarily. In such cases, add **enabled="false"** to the operation's definition. When you want to reinstate the monitoring operation, set **enabled="true"** to the operation's definition.

CHAPTER 16. CREATING CLUSTER RESOURCES THAT ARE ACTIVE ON MULTIPLE NODES (CLONED RESOURCES)

You can clone a cluster resource so that the resource can be active on multiple nodes. For example, you can use cloned resources to configure multiple instances of an IP resource to distribute throughout a cluster for node balancing. You can clone any resource provided the resource agent supports it. A clone consists of one resource or one resource group.



NOTE

Only resources that can be active on multiple nodes at the same time are suitable for cloning. For example, a **Filesystem** resource mounting a non-clustered file system such as **ext4** from a shared memory device should not be cloned. Since the **ext4** partition is not cluster aware, this file system is not suitable for read/write operations occurring from multiple nodes at the same time.

16.1. CREATING AND REMOVING A CLONED RESOURCE

You can create a resource and a clone of that resource at the same time with the following command.

```
pcs resource create resource_id [standard:[provider:]]type [resource options] [meta resource meta options] clone [clone options]
```

The name of the clone will be ***resource_id*-clone**.

You cannot create a resource group and a clone of that resource group in a single command.

Alternately, you can create a clone of a previously-created resource or resource group with the following command.

```
pcs resource clone resource_id | group_name [clone options]...
```

The name of the clone will be ***resource_id*-clone** or ***group_name*-clone**.

NOTE

You need to configure resource configuration changes on one node only.

NOTE

When configuring constraints, always use the name of the group or clone.

When you create a clone of a resource, the clone takes on the name of the resource with **-clone** appended to the name. The following commands creates a resource of type **apache** named **webfarm** and a clone of that resource named **webfarm-clone**.

```
# pcs resource create webfarm apache clone
```

NOTE

When you create a resource or resource group clone that will be ordered after another clone, you should almost always set the **interleave=true** option. This ensures that copies of the dependent clone can stop or start when the clone it depends on has stopped or started on the same node. If you do not set this option, if a cloned resource B depends on a cloned resource A and a node leaves the cluster, when the node returns to the cluster and resource A starts on that node, then all of the copies of resource B on all of the nodes will restart. This is because when a dependent cloned resource does not have the **interleave** option set, all instances of that resource depend on any running instance of the resource it depends on.

Use the following command to remove a clone of a resource or a resource group. This does not remove the resource or resource group itself.

```
pcs resource unclone resource_id | group_name
```

Table 16.1, "Resource Clone Options" describes the options you can specify for a cloned resource.

Table 16.1. Resource Clone Options

Field	Description
priority, target-role, is-managed	Options inherited from resource that is being cloned, as described in Table 9.3, "Resource Meta Options".
clone-max	How many copies of the resource to start. Defaults to the number of nodes in the cluster.
clone-node-max	How many copies of the resource can be started on a single node; the default value is **1**.
notify	When stopping or starting a copy of the clone, tell all the other copies beforehand and when the action was successful. Allowed values: **false**, **true**. The default value is **false**.
globally-unique	Does each copy of the clone perform a different function? Allowed values: **false**, **true** If the value of this option is **false**, these resources behave identically everywhere they are running and thus there can be only one copy of the clone active per machine. If the value of this option is **true**, a copy of the clone running on one machine is not equivalent to another instance, whether that instance is running on another node or on the same node. The default value is **true** if the value of **clone-node-max** is greater than one; otherwise the default value is **false**.

Field	Description
ordered	Should the copies be started in series (instead of in parallel). Allowed values: **false**, **true**. The default value is **false**.
interleave	Changes the behavior of ordering constraints (between clones) so that copies of the first clone can start or stop as soon as the copy on the same node of the second clone has started or stopped (rather than waiting until every instance of the second clone has started or stopped). Allowed values: **false**, **true**. The default value is **false**.
clone-min	If a value is specified, any clones which are ordered after this clone will not be able to start until the specified number of instances of the original clone are running, even if the **interleave** option is set to **true**.

To achieve a stable allocation pattern, clones are slightly sticky by default, which indicates that they have a slight preference for staying on the node where they are running. If no value for **resource-stickiness** is provided, the clone will use a value of 1. Being a small value, it causes minimal disturbance to the score calculations of other resources but is enough to prevent Pacemaker from needlessly moving copies around the cluster. For information on setting the **resource-stickiness** resource meta-option, see Configuring resource meta options.

16.2. CONFIGURING CLONE RESOURCE CONSTRAINTS

In most cases, a clone will have a single copy on each active cluster node. You can, however, set **clone-max** for the resource clone to a value that is less than the total number of nodes in the cluster. If this is the case, you can indicate which nodes the cluster should preferentially assign copies to with resource location constraints. These constraints are written no differently to those for regular resources except that the clone's id must be used.

The following command creates a location constraint for the cluster to preferentially assign resource clone **webfarm-clone** to **node1**.

```
# pcs constraint location webfarm-clone prefers node1
```

Ordering constraints behave slightly differently for clones. In the example below, **webfarm-stats** will wait until all copies of **webfarm-clone** that need to be started have done so before being started itself. Only if no copies of **webfarm-clone** can be started then **webfarm-stats** will be prevented from being active. Additionally, **webfarm-clone** will wait for **webfarm-stats** to be stopped before stopping itself.

```
# pcs constraint order start webfarm-clone then webfarm-stats
```

Colocation of a regular (or group) resource with a clone means that the resource can run on any machine with an active copy of the clone. The cluster will choose a copy based on where the clone is running and the resource's own location preferences.

Colocation between clones is also possible. In such cases, the set of allowed locations for the clone is limited to nodes on which the clone is (or will be) active. Allocation is then performed as normally.

The following command creates a colocation constraint to ensure that the resource **webfarm-stats** runs on the same node as an active copy of **webfarm-clone**.

```
# pcs constraint colocation add webfarm-stats with webfarm-clone
```

16.3. CREATING PROMOTABLE CLONE RESOURCES

Promotable clone resources are clone resources with the **promotable** meta attribute set to **true**. They allow the instances to be in one of two operating modes; these are called **Master** and **Slave**. The names of the modes do not have specific meanings, except for the limitation that when an instance is started, it must come up in the **Slave** state.

16.3.1. Creating a promotable resource

You can create a resource as a promotable clone with the following single command.

```
pcs resource create resource_id [standard:[provider:]]type [resource options] promotable [clone options]
```

The name of the promotable clone will be ***resource_id*-clone**.

Alternately, you can create a promotable resource from a previously-created resource or resource group with the following command. The name of the promotable clone will be ***resource_id*-clone** or ***group_name*-clone**.

```
pcs resource promotable resource_id [clone options]
```

Table 16.2, "Extra Clone Options Available for Promotable Clones" describes the extra clone options you can specify for a promotable resource.

Table 16.2. Extra Clone Options Available for Promotable Clones

Field	Description
promoted-max	How many copies of the resource can be promoted; default 1.
promoted-node-max	How many copies of the resource can be promoted on a single node; default 1.

16.3.2. Configuring promotable resource constraints

In most cases, a promotable resources will have a single copy on each active cluster node. If this is not the case, you can indicate which nodes the cluster should preferentially assign copies to with resource location constraints. These constraints are written no differently than those for regular resources.

You can create a colocation constraint which specifies whether the resources are operating in a master or slave role. The following command creates a resource colocation constraint.

```
pcs constraint colocation add [master|slave] source_resource with [master|slave] target_resource
[score] [options]
```

For information on colocation constraints, see Colocating cluster resources.

When configuring an ordering constraint that includes promotable resources, one of the actions that you can specify for the resources is **promote**, indicating that the resource be promoted from slave role to master role. Additionally, you can specify an action of **demote**, indicated that the resource be demoted from master role to slave role.

The command for configuring an order constraint is as follows.

```
pcs constraint order [action] resource_id then [action] resource_id [options]
```

For information on resource order constraints, see ifdef:: Determining the order in which cluster resources are run.

CHAPTER 17. MANAGING CLUSTER NODES

The following sections describe the commands you use to manage cluster nodes, including commands to start and stop cluster services and to add and remove cluster nodes.

17.1. STOPPING CLUSTER SERVICES

The following command stops cluster services on the specified node or nodes. As with the **pcs cluster start**, the **--all** option stops cluster services on all nodes and if you do not specify any nodes, cluster services are stopped on the local node only.

```
pcs cluster stop [--all | node] [...]
```

You can force a stop of cluster services on the local node with the following command, which performs a **kill -9** command.

```
pcs cluster kill
```

17.2. ENABLING AND DISABLING CLUSTER SERVICES

Use the following command to enables the cluster services, which configures the cluster services to run on startup on the specified node or nodes. Enabling allows nodes to automatically rejoin the cluster after they have been fenced, minimizing the time the cluster is at less than full strength. If the cluster services are not enabled, an administrator can manually investigate what went wrong before starting the cluster services manually, so that, for example, a node with hardware issues in not allowed back into the cluster when it is likely to fail again.

- If you specify the **--all** option, the command enables cluster services on all nodes.
- If you do not specify any nodes, cluster services are enabled on the local node only.

```
pcs cluster enable [--all | node] [...]
```

Use the following command to configure the cluster services not to run on startup on the specified node or nodes.

- If you specify the **--all** option, the command disables cluster services on all nodes.
- If you do not specify any nodes, cluster services are disabled on the local node only.

```
pcs cluster disable [--all | node] [...]
```

17.3. ADDING CLUSTER NODES

NOTE

It is highly recommended that you add nodes to existing clusters only during a production maintenance window. This allows you to perform appropriate resource and deployment testing for the new node and its fencing configuration.

Use the following procedure to add a new node to an existing cluster. This procedure adds standard clusters nodes running **corosync**. For information on integrating non-corosync nodes into a cluster, see Integrating non-corosync nodes into a cluster: the pacemaker_remote service .

In this example, the existing cluster nodes are **clusternode-01.example.com**, **clusternode-02.example.com**, and **clusternode-03.example.com**. The new node is **newnode.example.com**.

On the new node to add to the cluster, perform the following tasks.

1. Install the cluster packages. If the cluster uses SBD, the Booth ticket manager, or a quorum device, you must manually install the respective packages (**sbd**, **booth-site**, **corosync-qdevice**) on the new node as well.

```
[root@newnode ~]# yum install -y pcs fence-agents-all
```

   In addition to the cluster packages, you will also need to install and configure all of the services that you are running in the cluster, which you have installed on the existing cluster nodes. For example, if you are running an Apache HTTP server in a Red Hat high availability cluster, you will need to install the server on the node you are adding, as well as the **wget** tool that checks the status of the server.

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.

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

3. Set a password for the user ID **hacluster**. It is recommended that you use the same password for each node in the cluster.

```
[root@newnode ~]# passwd hacluster
Changing password for user hacluster.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
```

4. Execute the following commands to start the **pcsd** service and to enable **pcsd** at system start.

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

On a node in the existing cluster, perform the following tasks.

1. Authenticate user **hacluster** on the new cluster node.

```
[root@clusternode-01 ~]# pcs host auth newnode.example.com
Username: hacluster
Password:
newnode.example.com: Authorized
```

2. Add the new node to the existing cluster. This command also syncs the cluster configuration file **corosync.conf** to all nodes in the cluster, including the new node you are adding.

```
[root@clusternode-01 ~]# pcs cluster node add newnode.example.com
```

On the new node to add to the cluster, perform the following tasks.

1. Start and enable cluster services on the new node.

```
[root@newnode ~]# pcs cluster start
Starting Cluster...
[root@newnode ~]# pcs cluster enable
```

2. Ensure that you configure and test a fencing device for the new cluster node.

17.4. REMOVING CLUSTER NODES

The following command shuts down the specified node and removes it from the cluster configuration file, **corosync.conf**, on all of the other nodes in the cluster.

```
pcs cluster node remove node
```

CHAPTER 18. PACEMAKER CLUSTER PROPERTIES

Cluster properties control how the cluster behaves when confronted with situations that may occur during cluster operation.

18.1. SUMMARY OF CLUSTER PROPERTIES AND OPTIONS

Table 18.1, "Cluster Properties" summaries the Pacemaker cluster properties, showing the default values of the properties and the possible values you can set for those properties.

NOTE

In addition to the properties described in this table, there are additional cluster properties that are exposed by the cluster software. For these properties, it is recommended that you not change their values from their defaults.

Table 18.1. Cluster Properties

Option	Default	Description
batch-limit	0	The number of resource actions that the cluster is allowed to execute in parallel. The "correct" value will depend on the speed and load of your network and cluster nodes. The default value of 0 means that the cluster will dynamically impose a limit when any node has a high CPU load.
migration-limit	-1 (unlimited)	The number of migration jobs that the cluster is allowed to execute in parallel on a node.
no-quorum-policy	stop	What to do when the cluster does not have quorum. Allowed values: * ignore - continue all resource management * freeze - continue resource management, but do not recover resources from nodes not in the affected partition * stop - stop all resources in the affected cluster partition * suicide - fence all nodes in the affected cluster partition
symmetric-cluster	true	Indicates whether resources can run on any node by default.

Option	Default	Description
stonith-enabled	true	Indicates that failed nodes and nodes with resources that cannot be stopped should be fenced. Protecting your data requires that you set this **true**. If **true**, or unset, the cluster will refuse to start resources unless one or more STONITH resources have been configured also. Red Hat only supports clusters with this value set to **true**.
stonith-action	reboot	Action to send to STONITH device. Allowed values: **reboot**, **off**. The value **poweroff** is also allowed, but is only used for legacy devices.
cluster-delay	60s	Round trip delay over the network (excluding action execution). The "correct" value will depend on the speed and load of your network and cluster nodes.
stop-orphan-resources	true	Indicates whether deleted resources should be stopped.
stop-orphan-actions	true	Indicates whether deleted actions should be canceled.
start-failure-is-fatal	true	Indicates whether a failure to start a resource on a particular node prevents further start attempts on that node. When set to **false**, the cluster will decide whether to try starting on the same node again based on the resource's current failure count and migration threshold. For information on setting the **migration-threshold** option for a resource, see Configuring resource meta options. Setting **start-failure-is-fatal** to **false** incurs the risk that this will allow one faulty node that is unable to start a resource to hold up all dependent actions. This is why **start-failure-is-fatal** defaults to true. The risk of setting **start-failure-is-fatal=false** can be mitigated by setting a low migration threshold so that other actions can proceed after that many failures.

Option	Default	Description
pe-error-series-max	-1 (all)	The number of scheduler inputs resulting in ERRORs to save. Used when reporting problems.
pe-warn-series-max	-1 (all)	The number of scheduler inputs resulting in WARNINGs to save. Used when reporting problems.
pe-input-series-max	-1 (all)	The number of "normal" scheduler inputs to save. Used when reporting problems.
cluster-infrastructure		The messaging stack on which Pacemaker is currently running. Used for informational and diagnostic purposes; not user-configurable.
dc-version		Version of Pacemaker on the cluster's Designated Controller (DC). Used for diagnostic purposes; not user-configurable.
cluster-recheck-interval	15 minutes	Polling interval for time-based changes to options, resource parameters and constraints. Allowed values: Zero disables polling, positive values are an interval in seconds (unless other SI units are specified, such as 5min).
maintenance-mode	false	Maintenance Mode tells the cluster to go to a "hands off" mode, and not start or stop any services until told otherwise. When maintenance mode is completed, the cluster does a sanity check of the current state of any services, and then stops or starts any that need it.
shutdown-escalation	20min	The time after which to give up trying to shut down gracefully and just exit. Advanced use only.
stonith-timeout	60s	How long to wait for a STONITH action to complete.
stop-all-resources	false	Should the cluster stop all resources.
enable-acl	false	Indicates whether the cluster can use access control lists, as set with the **pcs acl** command.

Option	Default	Description
placement-strategy	**default**	Indicates whether and how the cluster will take utilization attributes into account when determining resource placement on cluster nodes.

18.2. SETTING AND REMOVING CLUSTER PROPERTIES

To set the value of a cluster property, use the following **pcs** command.

```
pcs property set property=value
```

For example, to set the value of **symmetric-cluster** to **false**, use the following command.

```
# pcs property set symmetric-cluster=false
```

You can remove a cluster property from the configuration with the following command.

```
pcs property unset property
```

Alternately, you can remove a cluster property from a configuration by leaving the value field of the **pcs property set** command blank. This restores that property to its default value. For example, if you have previously set the **symmetric-cluster** property to **false**, the following command removes the value you have set from the configuration and restores the value of **symmetric-cluster** to **true**, which is its default value.

```
# pcs property set symmetic-cluster=
```

18.3. QUERYING CLUSTER PROPERTY SETTINGS

In most cases, when you use the **pcs** command to display values of the various cluster components, you can use **pcs list** or **pcs show** interchangeably. In the following examples, **pcs list** is the format used to display an entire list of all settings for more than one property, while **pcs show** is the format used to display the values of a specific property.

To display the values of the property settings that have been set for the cluster, use the following **pcs** command.

```
pcs property list
```

To display all of the values of the property settings for the cluster, including the default values of the property settings that have not been explicitly set, use the following command.

```
pcs property list --all
```

To display the current value of a specific cluster property, use the following command.

```
pcs property show property
```

For example, to display the current value of the **cluster-infrastructure** property, execute the following command:

```
# pcs property show cluster-infrastructure
Cluster Properties:
 cluster-infrastructure: cman
```

For informational purposes, you can display a list of all of the default values for the properties, whether they have been set to a value other than the default or not, by using the following command.

```
pcs property [list|show] --defaults
```

CHAPTER 19. CONFIGURING A VIRTUAL DOMAIN AS A RESOURCE

You can configure a virtual domain that is managed by the **libvirt** virtualization framework as a cluster resource with the **pcs resource create** command, specifying **VirtualDomain** as the resource type.

When configuring a virtual domain as a resource, take the following considerations into account:

- A virtual domain should be stopped before you configure it as a cluster resource.
- Once a virtual domain is a cluster resource, it should not be started, stopped, or migrated except through the cluster tools.
- Do not configure a virtual domain that you have configured as a cluster resource to start when its host boots.
- All nodes allowed to run a virtual domain must have access to the necessary configuration files and storage devices for that virtual domain.

If you want the cluster to manage services within the virtual domain itself, you can configure the virtual domain as a guest node.

19.1. VIRTUAL DOMAIN RESOURCE OPTIONS

Table 19.1, "Resource Options for Virtual Domain Resources" describes the resource options you can configure for a **VirtualDomain** resource.

Table 19.1. Resource Options for Virtual Domain Resources

Field	Default	Description
config		(required) Absolute path to the **libvirt** configuration file for this virtual domain.
hypervisor	System dependent	Hypervisor URI to connect to. You can determine the system's default URI by running the **virsh --quiet uri** command.
force_stop	**0**	Always forcefully shut down ("destroy") the domain on stop. The default behavior is to resort to a forceful shutdown only after a graceful shutdown attempt has failed. You should set this to **true** only if your virtual domain (or your virtualization back end) does not support graceful shutdown.

Field	Default	Description
migration_transport	System dependent	Transport used to connect to the remote hypervisor while migrating. If this parameter is omitted, the resource will use **libvirt**'s default transport to connect to the remote hypervisor.
migration_network_suffix		Use a dedicated migration network. The migration URI is composed by adding this parameter's value to the end of the node name. If the node name is a fully qualified domain name (FQDN), insert the suffix immediately prior to the first period (.) in the FQDN. Ensure that this composed host name is locally resolvable and the associated IP address is reachable through the favored network.
monitor_scripts		To additionally monitor services within the virtual domain, add this parameter with a list of scripts to monitor. *Note*: When monitor scripts are used, the **start** and **migrate_from** operations will complete only when all monitor scripts have completed successfully. Be sure to set the timeout of these operations to accommodate this delay
autoset_utilization_cpu	**true**	If set to **true**, the agent will detect the number of **domainU**'s **vCPU**s from **virsh**, and put it into the CPU utilization of the resource when the monitor is executed.
autoset_utilization_hv_memory	**true**	If set it true, the agent will detect the number of **Max memory** from **virsh**, and put it into the **hv_memory** utilization of the source when the monitor is executed.
migrateport	random highport	This port will be used in the **qemu** migrate URI. If unset, the port will be a random highport.

Field	Default	Description
snapshot		Path to the snapshot directory where the virtual machine image will be stored. When this parameter is set, the virtual machine's RAM state will be saved to a file in the snapshot directory when stopped. If on start a state file is present for the domain, the domain will be restored to the same state it was in right before it stopped last. This option is incompatible with the **force_stop** option.

In addition to the **VirtualDomain** resource options, you can configure the **allow-migrate** metadata option to allow live migration of the resource to another node. When this option is set to **true**, the resource can be migrated without loss of state. When this option is set to **false**, which is the default state, the virtual domain will be shut down on the first node and then restarted on the second node when it is moved from one node to the other.

19.2. CREATING THE VIRTUAL DOMAIN RESOURCE

Use the following procedure to create a **VirtualDomain** resource:

1. To create the **VirtualDomain** resource agent for the management of the virtual machine, Pacemaker requires the virtual machine's **xml** configuration file to be dumped to a file on disk. For example, if you created a virtual machine named **guest1**, dump the **xml** file to a file somewhere on the host. You can use a file name of your choosing; this example uses **/etc/pacemaker/guest1.xml**.

   ```
   # virsh dumpxml guest1 > /etc/pacemaker/guest1.xml
   ```

2. If it is running, shut down the guest node. Pacemaker will start the node when it is configured in the cluster.

3. Configure the **VirtualDoman** resource with the **pcs resource create** command. For example, The following command configures a **VirtualDomain** resource named **VM**. Since the **allow-migrate** option is set to **true** a **pcs move VM *nodeX*** command would be done as a live migration.

   ```
   # pcs resource create VM VirtualDomain config=.../vm.xml \
         migration_transport=ssh meta allow-migrate=true
   ```

CHAPTER 20. CLUSTER QUORUM

A Red Hat Enterprise Linux High Availability Add-On cluster uses the **votequorum** service, in conjunction with fencing, to avoid split brain situations. A number of votes is assigned to each system in the cluster, and cluster operations are allowed to proceed only when a majority of votes is present. The service must be loaded into all nodes or none; if it is loaded into a subset of cluster nodes, the results will be unpredictable. For information on the configuration and operation of the **votequorum** service, see the **votequorum**(5) man page.

20.1. CONFIGURING QUORUM OPTIONS

There are some special features of quorum configuration that you can set when you create a cluster with the **pcs cluster setup** command. Table 20.1, "Quorum Options" summarizes these options.

Table 20.1. Quorum Options

Option	Description
auto_tie_breaker	When enabled, the cluster can suffer up to 50% of the nodes failing at the same time, in a deterministic fashion. The cluster partition, or the set of nodes that are still in contact with the **nodeid** configured in **auto_tie_breaker_node** (or lowest **nodeid** if not set), will remain quorate. The other nodes will be inquorate. The **auto_tie_breaker** option is principally used for clusters with an even number of nodes, as it allows the cluster to continue operation with an even split. For more complex failures, such as multiple, uneven splits, it is recommended that you use a quorum device, as described in Quorum devices. The **auto_tie_breaker** option is incompatible with quorum devices.
wait_for_all	When enabled, the cluster will be quorate for the first time only after all nodes have been visible at least once at the same time. The **wait_for_all** option is primarily used for two-node clusters and for even-node clusters using the quorum device **lms** (last man standing) algorithm. The **wait_for_all** option is automatically enabled when a cluster has two nodes, does not use a quorum device, and **auto_tie_breaker** is disabled. You can override this by explicitly setting **wait_for_all** to 0.
last_man_standing	When enabled, the cluster can dynamically recalculate **expected_votes** and quorum under specific circumstances. You must enable **wait_for_all** when you enable this option. The **last_man_standing** option is incompatible with quorum devices.

Option	Description
last_man_standing_window	The time, in milliseconds, to wait before recalculating **expected_votes** and quorum after a cluster loses nodes.

For further information about configuring and using these options, see the **votequorum**(5) man page.

20.2. MODIFYING QUORUM OPTIONS

You can modify general quorum options for your cluster with the **pcs quorum update** command. Executing this command requires that the cluster be stopped. For information on the quorum options, see the **votequorum**(5) man page.

The format of the **pcs quorum update** command is as follows.

```
pcs quorum update [auto_tie_breaker=[0|1]] [last_man_standing=[0|1]] [last_man_standing_window=[time-in-ms] [wait_for_all=[0|1]]
```

The following series of commands modifies the **wait_for_all** quorum option and displays the updated status of the option. Note that the system does not allow you to execute this command while the cluster is running.

```
[root@node1:~]# pcs quorum update wait_for_all=1
Checking corosync is not running on nodes...
Error: node1: corosync is running
Error: node2: corosync is running

[root@node1:~]# pcs cluster stop --all
node2: Stopping Cluster (pacemaker)...
node1: Stopping Cluster (pacemaker)...
node1: Stopping Cluster (corosync)...
node2: Stopping Cluster (corosync)...

[root@node1:~]# pcs quorum update wait_for_all=1
Checking corosync is not running on nodes...
node2: corosync is not running
node1: corosync is not running
Sending updated corosync.conf to nodes...
node1: Succeeded
node2: Succeeded

[root@node1:~]# pcs quorum config
Options:
  wait_for_all: 1
```

20.3. DISPLAYING QUORUM CONFIGURATION AND STATUS

Once a cluster is running, you can enter the following cluster quorum commands.

The following command shows the quorum configuration.

```
pcs quorum [config]
```

The following command shows the quorum runtime status.

```
pcs quorum status
```

20.4. RUNNING INQUORATE CLUSTERS

If you take nodes out of a cluster for a long period of time and the loss of those nodes would cause quorum loss, you can change the value of the **expected_votes** parameter for the live cluster with the **pcs quorum expected-votes** command. This allows the cluster to continue operation when it does not have quorum.

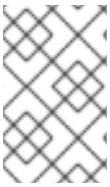
WARNING

Changing the expected votes in a live cluster should be done with extreme caution. If less than 50% of the cluster is running because you have manually changed the expected votes, then the other nodes in the cluster could be started separately and run cluster services, causing data corruption and other unexpected results. If you change this value, you should ensure that the **wait_for_all** parameter is enabled.

The following command sets the expected votes in the live cluster to the specified value. This affects the live cluster only and does not change the configuration file; the value of **expected_votes** is reset to the value in the configuration file in the event of a reload.

```
pcs quorum expected-votes votes
```

In a situation in which you know that the cluster is inquorate but you want the cluster to proceed with resource management, you can use the **pcs quorum unblock** command to prevent the cluster from waiting for all nodes when establishing quorum.



NOTE

This command should be used with extreme caution. Before issuing this command, it is imperative that you ensure that nodes that are not currently in the cluster are switched off and have no access to shared resources.

```
# pcs quorum unblock
```

20.5. QUORUM DEVICES

You can allow a cluster to sustain more node failures than standard quorum rules allows by configuring a separate quorum device which acts as a third-party arbitration device for the cluster. A quorum device is recommended for clusters with an even number of nodes and highly recommended for two-node clusters.

You must take the following into account when configuring a quorum device.

- It is recommended that a quorum device be run on a different physical network at the same site as the cluster that uses the quorum device. Ideally, the quorum device host should be in a separate rack than the main cluster, or at least on a separate PSU and not on the same network segment as the corosync ring or rings.
- You cannot use more than one quorum device in a cluster at the same time.
- Although you cannot use more than one quorum device in a cluster at the same time, a single quorum device may be used by several clusters at the same time. Each cluster using that quorum device can use different algorithms and quorum options, as those are stored on the cluster nodes themselves. For example, a single quorum device can be used by one cluster with an **ffsplit** (fifty/fifty split) algorithm and by a second cluster with an **lms** (last man standing) algorithm.
- A quorum device should not be run on an existing cluster node.

20.5.1. Installing quorum device packages

Configuring a quorum device for a cluster requires that you install the following packages:

- Install **corosync-qdevice** on the nodes of an existing cluster.

```
[root@node1:~]# yum install corosync-qdevice
[root@node2:~]# yum install corosync-qdevice
```

- Install **pcs** and **corosync-qnetd** on the quorum device host.

```
[root@qdevice:~]# yum install pcs corosync-qnetd
```

- Start the **pcsd** service and enable **pcsd** at system start on the quorum device host.

```
[root@qdevice:~]# systemctl start pcsd.service
[root@qdevice:~]# systemctl enable pcsd.service
```

20.5.2. Configuring a quorum device

The following procedure configures a quorum device and adds it to the cluster. In this example:

- The node used for a quorum device is **qdevice**.
- The quorum device model is **net**, which is currently the only supported model. The **net** model supports the following algorithms:
  - **ffsplit**: fifty-fifty split. This provides exactly one vote to the partition with the highest number of active nodes.
  - **lms**: last-man-standing. If the node is the only one left in the cluster that can see the **qnetd** server, then it returns a vote.

WARNING

The LMS algorithm allows the cluster to remain quorate even with only one remaining node, but it also means that the voting power of the quorum device is great since it is the same as number_of_nodes - 1. Losing connection with the quorum device means losing number_of_nodes - 1 votes, which means that only a cluster with all nodes active can remain quorate (by overvoting the quorum device); any other cluster becomes inquorate.

For more detailed information on the implementation of these algorithms, see the **corosync-qdevice**(8) man page.

- The cluster nodes are **node1** and **node2**.

The following procedure configures a quorum device and adds that quorum device to a cluster.

1. On the node that you will use to host your quorum device, configure the quorum device with the following command. This command configures and starts the quorum device model **net** and configures the device to start on boot.

```
[root@qdevice:~]# pcs qdevice setup model net --enable --start
Quorum device 'net' initialized
quorum device enabled
Starting quorum device...
quorum device started
```

After configuring the quorum device, you can check its status. This should show that the **corosync-qnetd** daemon is running and, at this point, there are no clients connected to it. The **--full** command option provides detailed output.

```
[root@qdevice:~]# pcs qdevice status net --full
QNetd address:                  *:5403
TLS:                            Supported (client certificate required)
Connected clients:              0
Connected clusters:             0
Maximum send/receive size:      32768/32768 bytes
```

2. Enable the ports on the firewall needed by the **pcsd** daemon and the **net** quorum device by enabling the **high-availability** service on **firewalld** with following commands.

```
[root@qdevice:~]# firewall-cmd --permanent --add-service=high-availability
[root@qdevice:~]# firewall-cmd --add-service=high-availability
```

3. From one of the nodes in the existing cluster, authenticate user **hacluster** on the node that is hosting the quorum device. This allows **pcs** on the cluster to connect to **pcs** on the **qdevice** host, but does not allow **pcs** on the **qdevice** host to connect to **pcs** on the cluster.

```
[root@node1:~] # pcs host auth qdevice
Username: hacluster
```

```
Password:
qdevice: Authorized
```

4. Add the quorum device to the cluster.
   Before adding the quorum device, you can check the current configuration and status for the quorum device for later comparison. The output for these commands indicates that the cluster is not yet using a quorum device.

```
[root@node1:~]# pcs quorum config
Options:
```

```
[root@node1:~]# pcs quorum status
Quorum information
------------------
Date:             Wed Jun 29 13:15:36 2016
Quorum provider:  corosync_votequorum
Nodes:            2
Node ID:          1
Ring ID:          1/8272
Quorate:          Yes

Votequorum information
----------------------
Expected votes:   2
Highest expected: 2
Total votes:      2
Quorum:           1
Flags:            2Node Quorate

Membership information
----------------------
    Nodeid      Votes    Qdevice Name
         1          1         NR node1 (local)
         2          1         NR node2
```

   The following command adds the quorum device that you have previously created to the cluster. You cannot use more than one quorum device in a cluster at the same time. However, one quorum device can be used by several clusters at the same time. This example command configures the quorum device to use the **ffsplit** algorithm. For information on the configuration options for the quorum device, see the **corosync-qdevice**(8) man page.

```
[root@node1:~]# pcs quorum device add model net host=qdevice algorithm=ffsplit
Setting up qdevice certificates on nodes...
node2: Succeeded
node1: Succeeded
Enabling corosync-qdevice...
node1: corosync-qdevice enabled
node2: corosync-qdevice enabled
Sending updated corosync.conf to nodes...
node1: Succeeded
node2: Succeeded
Corosync configuration reloaded
Starting corosync-qdevice...
node1: corosync-qdevice started
node2: corosync-qdevice started
```

5. Check the configuration status of the quorum device.
   From the cluster side, you can execute the following commands to see how the configuration has changed.

   The **pcs quorum config** shows the quorum device that has been configured.

```
[root@node1:~]# pcs quorum config
Options:
Device:
  Model: net
    algorithm: ffsplit
    host: qdevice
```

   The **pcs quorum status** command shows the quorum runtime status, indicating that the quorum device is in use.

```
[root@node1:~]# pcs quorum status
Quorum information
------------------
Date:             Wed Jun 29 13:17:02 2016
Quorum provider:  corosync_votequorum
Nodes:            2
Node ID:          1
Ring ID:          1/8272
Quorate:          Yes

Votequorum information
----------------------
Expected votes:   3
Highest expected: 3
Total votes:      3
Quorum:           2
Flags:            Quorate Qdevice

Membership information
----------------------
    Nodeid      Votes    Qdevice Name
         1          1    A,V,NMW node1 (local)
         2          1    A,V,NMW node2
         0          1            Qdevice
```

   The **pcs quorum device status** shows the quorum device runtime status.

```
[root@node1:~]# pcs quorum device status
Qdevice information
-------------------
Model:                  Net
Node ID:                1
Configured node list:
    0   Node ID = 1
    1   Node ID = 2
Membership node list:   1, 2

Qdevice-net information
```

```
----------------------
Cluster name:          mycluster
QNetd host:            qdevice:5403
Algorithm:             ffsplit
Tie-breaker:           Node with lowest node ID
State:                 Connected
```

From the quorum device side, you can execute the following status command, which shows the status of the **corosync-qnetd** daemon.

```
[root@qdevice:~]# pcs qdevice status net --full
QNetd address:                  *:5403
TLS:                            Supported (client certificate required)
Connected clients:              2
Connected clusters:             1
Maximum send/receive size:      32768/32768 bytes
Cluster "mycluster":
    Algorithm:          ffsplit
    Tie-breaker:        Node with lowest node ID
    Node ID 2:
        Client address:         ::ffff:192.168.122.122:50028
        HB interval:            8000ms
        Configured node list:   1, 2
        Ring ID:                1.2050
        Membership node list:   1, 2
        TLS active:             Yes (client certificate verified)
        Vote:                   ACK (ACK)
    Node ID 1:
        Client address:         ::ffff:192.168.122.121:48786
        HB interval:            8000ms
        Configured node list:   1, 2
        Ring ID:                1.2050
        Membership node list:   1, 2
        TLS active:             Yes (client certificate verified)
        Vote:                   ACK (ACK)
```

20.5.3. Managing the Quorum Device Service

PCS provides the ability to manage the quorum device service on the local host (**corosync-qnetd**), as shown in the following example commands. Note that these commands affect only the **corosync-qnetd** service.

```
[root@qdevice:~]# pcs qdevice start net
[root@qdevice:~]# pcs qdevice stop net
[root@qdevice:~]# pcs qdevice enable net
[root@qdevice:~]# pcs qdevice disable net
[root@qdevice:~]# pcs qdevice kill net
```

20.5.4. Managing the quorum device settings in a cluster

The following sections describe the PCS commands that you can use to manage the quorum device settings in a cluster.

20.5.4.1. Changing quorum device settings

You can change the setting of a quorum device with the **pcs quorum device update** command.

WARNING

To change the **host** option of quorum device model **net**, use the **pcs quorum device remove** and the **pcs quorum device add** commands to set up the configuration properly, unless the old and the new host are the same machine.

The following command changes the quorum device algorithm to **lms**.

```
[root@node1:~]# pcs quorum device update model algorithm=lms
Sending updated corosync.conf to nodes...
node1: Succeeded
node2: Succeeded
Corosync configuration reloaded
Reloading qdevice configuration on nodes...
node1: corosync-qdevice stopped
node2: corosync-qdevice stopped
node1: corosync-qdevice started
node2: corosync-qdevice started
```

20.5.4.2. Removing a quorum device

Use the following command to remove a quorum device configured on a cluster node.

```
[root@node1:~]# pcs quorum device remove
Sending updated corosync.conf to nodes...
node1: Succeeded
node2: Succeeded
Corosync configuration reloaded
Disabling corosync-qdevice...
node1: corosync-qdevice disabled
node2: corosync-qdevice disabled
Stopping corosync-qdevice...
node1: corosync-qdevice stopped
node2: corosync-qdevice stopped
Removing qdevice certificates from nodes...
node1: Succeeded
node2: Succeeded
```

After you have removed a quorum device, you should see the following error message when displaying the quorum device status.

```
[root@node1:~]# pcs quorum device status
Error: Unable to get quorum status: corosync-qdevice-tool: Can't connect to QDevice socket (is
QDevice running?): No such file or directory
```

20.5.4.3. Destroying a quorum device

To disable and stop a quorum device on the quorum device host and delete all of its configuration files, use the following command.

```
[root@qdevice:~]# pcs qdevice destroy net
Stopping quorum device...
quorum device stopped
quorum device disabled
Quorum device 'net' configuration files removed
```

CHAPTER 21. INTEGRATING NON-COROSYNC NODES INTO A CLUSTER: THE PACEMAKER_REMOTE SERVICE

The **pacemaker_remote** service allows nodes not running **corosync** to integrate into the cluster and have the cluster manage their resources just as if they were real cluster nodes.

Among the capabilities that the **pacemaker_remote** service provides are the following:

- The **pacemaker_remote** service allows you to scale beyond the **corosync** 16-node limit.
- The **pacemaker_remote** service allows you to manage a virtual environment as a cluster resource and also to manage individual services within the virtual environment as cluster resources.

The following terms are used to describe the **pacemaker_remote** service.

- *cluster node* — A node running the High Availability services (**pacemaker** and **corosync**).
- *remote node* — A node running **pacemaker_remote** to remotely integrate into the cluster without requiring **corosync** cluster membership. A remote node is configured as a cluster resource that uses the **ocf:pacemaker:remote** resource agent.
- *guest node* — A virtual guest node running the **pacemaker_remote** service. The virtual guest resource is managed by the cluster; it is both started by the cluster and integrated into the cluster as a remote node.
- *pacemaker_remote* — A service daemon capable of performing remote application management within remote nodes and KVM guest nodes in a Pacemaker cluster environment. This service is an enhanced version of Pacemaker's local executor daemon (**pacemaker-execd**) that is capable of managing resources remotely on a node not running corosync.

A Pacemaker cluster running the **pacemaker_remote** service has the following characteristics.

- Remote nodes and guest nodes run the **pacemaker_remote** service (with very little configuration required on the virtual machine side).
- The cluster stack (**pacemaker** and **corosync**), running on the cluster nodes, connects to the **pacemaker_remote** service on the remote nodes, allowing them to integrate into the cluster.
- The cluster stack (**pacemaker** and **corosync**), running on the cluster nodes, launches the guest nodes and immediately connects to the **pacemaker_remote** service on the guest nodes, allowing them to integrate into the cluster.

The key difference between the cluster nodes and the remote and guest nodes that the cluster nodes manage is that the remote and guest nodes are not running the cluster stack. This means the remote and guest nodes have the following limitations:

- they do not take place in quorum
- they do not execute fencing device actions
- they are not eligible to be the cluster's Designated Controller (DC)
- they do not themselves run the full range of **pcs** commands

On the other hand, remote nodes and guest nodes are not bound to the scalability limits associated with the cluster stack.

Other than these noted limitations, the remote and guest nodes behave just like cluster nodes in respect to resource management, and the remote and guest nodes can themselves be fenced. The cluster is fully capable of managing and monitoring resources on each remote and guest node: You can build constraints against them, put them in standby, or perform any other action you perform on cluster nodes with the **pcs** commands. Remote and guest nodes appear in cluster status output just as cluster nodes do.

21.1. HOST AND GUEST AUTHENTICATION OF PACEMAKER_REMOTE NODES

The connection between cluster nodes and pacemaker_remote is secured using Transport Layer Security (TLS) with pre-shared key (PSK) encryption and authentication over TCP (using port 3121 by default). This means both the cluster node and the node running **pacemaker_remote** must share the same private key. By default this key must be placed at **/etc/pacemaker/authkey** on both cluster nodes and remote nodes.

The **pcs cluster node add-guest** command sets up the **authkey** for guest nodes and the **pcs cluster node add-remote** command sets up the **authkey** for remote nodes.

21.2. CONFIGURING KVM GUEST NODES

A Pacemaker guest node is a virtual guest node running the **pacemaker_remote** service. The virtual guest node is managed by the cluster.

21.2.1. Guest node resource options

When configuring a virtual machine to act as a guest node, you create a **VirtualDomain** resource, which manages the virtual machine. For descriptions of the options you can set for a **VirtualDomain** resource, see Table 19.1, “Resource Options for Virtual Domain Resources”.

In addition to the **VirtualDomain** resource options, metadata options define the resource as a guest node and define the connection parameters. You set these resource options with the **pcs cluster node add-guest** command. Table 21.1, “Metadata Options for Configuring KVM Resources as Remote Nodes” describes these metadata options.

Table 21.1. Metadata Options for Configuring KVM Resources as Remote Nodes

Field	Default	Description
remote-node	<none>	The name of the guest node this resource defines. This both enables the resource as a guest node and defines the unique name used to identify the guest node. *WARNING*: This value cannot overlap with any resource or node IDs.

Field	Default	Description
remote-port	3121	Configures a custom port to use for the guest connection to **pacemaker_remote**
remote-addr	The address provided in the **pcs host auth** command	The IP address or host name to connect to
remote-connect-timeout	60s	Amount of time before a pending guest connection will time out

21.2.2. Integrating a virtual machine as a guest node

The following procedure is a high-level summary overview of the steps to perform to have Pacemaker launch a virtual machine and to integrate that machine as a guest node, using **libvirt** and KVM virtual guests.

1. Configure the **VirtualDomain** resources.
2. Enter the following commands on every virtual machine to install **pacemaker_remote** packages, start the **pcsd** service and enable it to run on startup, and allow TCP port 3121 through the firewall.

```
# yum install pacemaker-remote resource-agents pcs
# systemctl start pcsd.service
# systemctl enable pcsd.service
# firewall-cmd --add-port 3121/tcp --permanent
# firewall-cmd --add-port 2224/tcp --permanent
# firewall-cmd --reload
```

3. Give each virtual machine a static network address and unique host name, which should be known to all nodes. For information on setting a static IP address for the guest virtual machine, see the *Virtualization Deployment and Administration Guide*.
4. If you have not already done so, authenticate **pcs** to the node you will be integrating as a quest node.

```
# pcs host auth nodename
```

5. Use the following command to convert an existing **VirtualDomain** resource into a guest node. This command must be run on a cluster node and not on the guest node which is being added. In addition to converting the resource, this command copies the **/etc/pacemaker/authkey** to the guest node and starts and enables the **pacemaker_remote** daemon on the guest node. The node name for the guest node, which you can define arbitrarily, can differ from the host name for the node.

```
# pcs cluster node add-guest nodename resource_id [options]
```

6. After creating the **VirtualDomain** resource, you can treat the guest node just as you would treat any other node in the cluster. For example, you can create a resource and place a resource constraint on the resource to run on the guest node as in the following commands, which are run

from a cluster node. You can include guest nodes in groups, which allows you to group a storage device, file system, and VM.

```
# pcs resource create webserver apache configfile=/etc/httpd/conf/httpd.conf op
monitor interval=30s
# pcs constraint location webserver prefers nodename
```

21.3. CONFIGURING PACEMAKER REMOTE NODES

A remote node is defined as a cluster resource with **ocf:pacemaker:remote** as the resource agent. You create this resource with the **pcs cluster node add-remote** command.

21.3.1. Remote node resource options

Table 21.2, "Resource Options for Remote Nodes" describes the resource options you can configure for a **remote** resource.

Table 21.2. Resource Options for Remote Nodes

Field	Default	Description
reconnect_interval	0	Time in seconds to wait before attempting to reconnect to a remote node after an active connection to the remote node has been severed. This wait is recurring. If reconnect fails after the wait period, a new reconnect attempt will be made after observing the wait time. When this option is in use, Pacemaker will keep attempting to reach out and connect to the remote node indefinitely after each wait interval.
server	Address specified with **pcs host auth** command	Server to connect to. This can be an IP address or host name.
port		TCP port to connect to.

21.3.2. Remote node configuration overview

This section provides a high-level summary overview of the steps to perform to configure a Pacemaker Remote node and to integrate that node into an existing Pacemaker cluster environment.

1. On the node that you will be configuring as a remote node, allow cluster-related services through the local firewall.

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

NOTE

If you are using **iptables** directly, or some other firewall solution besides **firewalld**, simply open the following ports: TCP ports 2224 and 3121.

2. Install the **pacemaker_remote** daemon on the remote node.

```
# yum install -y pacemaker-remote resource-agents pcs
```

3. Start and enable **pcsd** on the remote node.

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

4. If you have not already done so, authenticate **pcs** to the node you will be adding as a remote node.

```
# pcs host auth remote1
```

5. Add the remote node resource to the cluster with the following command. This command also syncs all relevant configuration files to the new node, starts the node, and configures it to start **pacemaker_remote** on boot. This command must be run on a cluster node and not on the remote node which is being added.

```
# pcs cluster node add-remote remote1
```

6. After adding the **remote** resource to the cluster, you can treat the remote node just as you would treat any other node in the cluster. For example, you can create a resource and place a resource constraint on the resource to run on the remote node as in the following commands, which are run from a cluster node.

```
# pcs resource create webserver apache configfile=/etc/httpd/conf/httpd.conf op
monitor interval=30s
# pcs constraint location webserver prefers remote1
```

WARNING

Never involve a remote node connection resource in a resource group, colocation constraint, or order constraint.

7. Configure fencing resources for the remote node. Remote nodes are fenced the same way as cluster nodes. Configure fencing resources for use with remote nodes the same as you would with cluster nodes. Note, however, that remote nodes can never initiate a fencing action. Only cluster nodes are capable of actually executing a fencing operation against another node.

21.4. CHANGING THE DEFAULT PORT LOCATION

If you need to change the default port location for either Pacemaker or **pacemaker_remote**, you can set the **PCMK_remote_port** environment variable that affects both of these daemons. This environment variable can be enabled by placing it in the **/etc/sysconfig/pacemaker** file as follows.

```
====# Pacemaker Remote
...
#
# Specify a custom port for Pacemaker Remote connections
PCMK_remote_port=3121
```

When changing the default port used by a particular guest node or remote node, the **PCMK_remote_port** variable must be set in that node's **/etc/sysconfig/pacemaker** file, and the cluster resource creating the guest node or remote node connection must also be configured with the same port number (using the **remote-port** metadata option for guest nodes, or the **port** option for remote nodes).

21.5. UPGRADING SYSTEMS WITH PACEMAKER_REMOTE NODES

If the **pacemaker_remote** service is stopped on an active Pacemaker Remote node, the cluster will gracefully migrate resources off the node before stopping the node. This allows you to perform software upgrades and other routine maintenance procedures without removing the node from the cluster. Once **pacemaker_remote** is shut down, however, the cluster will immediately try to reconnect. If **pacemaker_remote** is not restarted within the resource's monitor timeout, the cluster will consider the monitor operation as failed.

If you wish to avoid monitor failures when the **pacemaker_remote** service is stopped on an active Pacemaker Remote node, you can use the following procedure to take the node out of the cluster before performing any system administration that might stop **pacemaker_remote**

1. Stop the node's connection resource with the **pcs resource disable *resourcename***, which will move all services off the node. For guest nodes, this will also stop the VM, so the VM must be started outside the cluster (for example, using **virsh**) to perform any maintenance.

2. Perform the required maintenance.

3. When ready to return the node to the cluster, re-enable the resource with the **pcs resource enable**.

CHAPTER 22. PERFORMING CLUSTER MAINTENANCE

In order to perform maintenance on the nodes of your cluster, you may need to stop or move the resources and services running on that cluster. Or you may need to stop the cluster software while leaving the services untouched. Pacemaker provides a variety of methods for performing system maintenance.

- If you need to stop a node in a cluster while continuing to provide the services running on that cluster on another node, you can put the cluster node in standby mode. A node that is in standby mode is no longer able to host resources. Any resource currently active on the node will be moved to another node, or stopped if no other node is eligible to run the resource. For information on standby mode, see Putting a node into standby mode.

- If you need to move an individual resource off the node on which it is currently running without stopping that resource, you can use the **pcs resource move** command to move the resource to a different node. For information on the **pcs resource move** command, see Manually moving cluster resources.
  When you execute the **pcs resource move** command, this adds a constraint to the resource to prevent it from running on the node on which it is currently running. When you are ready to move the resource back, you can execute the **pcs resource clear** or the **pcs constraint delete** command to remove the constraint. This does not necessarily move the resources back to the original node, however, since where the resources can run at that point depends on how you have configured your resources initially. You can relocate a resource to its preferred node with the **pcs resource relocate run** command, as described in Moving a resource to its preferred node.

- If you need to stop a running resource entirely and prevent the cluster from starting it again, you can use the **pcs resource disable** command. For information on the **pcs resource disable** command, see Enabling, disabling, and banning cluster resources.

- If you want to prevent Pacemaker from taking any action for a resource (for example, if you want to disable recovery actions while performing maintenance on the resource, or if you need to reload the **/etc/sysconfig/pacemaker** settings), use the **pcs resource unmanage** command, as described in Setting a resource to unmanaged mode. Pacemaker Remote connection resources should never be unmanaged.

- If you need to put the cluster in a state where no services will be started or stopped, you can set the **maintenance-mode** cluster property. Putting the cluster into maintenance mode automatically unmanages all resources. For information on putting the cluster in maintenance mode, see Putting a cluster in maintenance mode.

- If you need to update the packages that make up the RHEL High Availability and Resilient Storage Add-Ons, you can update the packages on one node at a time or on the entire cluster as a whole, as summarized in Updating a Red Hat Enterprise Linux high availability cluster.

- If you need to perform maintenance on a Pacemaker remote node, you can remove that node from the cluster by disabling the remote node resource, as described in Upgrading remote nodes and guest nodes.

22.1. PUTTING A NODE INTO STANDBY MODE

When a cluster node is in standby mode, the node is no longer able to host resources. Any resources currently active on the node will be moved to another node.

The following command puts the specified node into standby mode. If you specify the **--all**, this command puts all nodes into standby mode.

You can use this command when updating a resource's packages. You can also use this command when testing a configuration, to simulate recovery without actually shutting down a node.

```
pcs node standby node | --all
```

The following command removes the specified node from standby mode. After running this command, the specified node is then able to host resources. If you specify the **--all**, this command removes all nodes from standby mode.

```
pcs node unstandby node | --all
```

Note that when you execute the **pcs node standby** command, this prevents resources from running on the indicated node. When you execute the **pcs node unstandby** command, this allows resources to run on the indicated node. This does not necessarily move the resources back to the indicated node; where the resources can run at that point depends on how you have configured your resources initially.

22.2. MANUALLY MOVING CLUSTER RESOURCES

You can override the cluster and force resources to move from their current location. There are two occasions when you would want to do this:

- When a node is under maintenance, and you need to move all resources running on that node to a different node
- When individually specified resources needs to be moved

To move all resources running on a node to a different node, you put the node in standby mode.

You can move individually specified resources in either of the following ways.

- You can use the **pcs resource move** command to move a resource off a node on which it is currently running.
- You can use the **pcs resource relocate run** command to move a resource to its preferred node, as determined by current cluster status, constraints, location of resources and other settings.

22.2.1. Moving a resource from its current node

To move a resource off the node on which it is currently running, use the following command, specifying the *resource_id* of the resource as defined. Specify the **destination_node** if you want to indicate on which node to run the resource that you are moving.

```
pcs resource move resource_id [destination_node] [--master] [lifetime=lifetime]
```

NOTE

When you execute the **pcs resource move** command, this adds a constraint to the resource to prevent it from running on the node on which it is currently running. You can execute the **pcs resource clear** or the **pcs constraint delete** command to remove the constraint. This does not necessarily move the resources back to the original node; where the resources can run at that point depends on how you have configured your resources initially.

If you specify the **--master** parameter of the **pcs resource move** command, the scope of the constraint is limited to the master role and you must specify *master_id* rather than *resource_id*.

You can optionally configure a **lifetime** parameter for the **pcs resource move** command to indicate a period of time the constraint should remain. You specify the units of a **lifetime** parameter according to the format defined in ISO 8601, which requires that you specify the unit as a capital letter such as Y (for years), M (for months), W (for weeks), D (for days), H (for hours), M (for minutes), and S (for seconds).

To distinguish a unit of minutes(M) from a unit of months(M), you must specify PT before indicating the value in minutes. For example, a **lifetime** parameter of 5M indicates an interval of five months, while a **lifetime** parameter of PT5M indicates an interval of five minutes.

The **lifetime** parameter is checked at intervals defined by the **cluster-recheck-interval** cluster property. By default this value is 15 minutes. If your configuration requires that you check this parameter more frequently, you can reset this value with the following command.

```
pcs property set cluster-recheck-interval=value
```

You can optionally configure a **--wait[=*n*]** parameter for the **pcs resource move** command to indicate the number of seconds to wait for the resource to start on the destination node before returning 0 if the resource is started or 1 if the resource has not yet started. If you do not specify n, the default resource timeout will be used.

The following command moves the resource **resource1** to node **example-node2** and prevents it from moving back to the node on which it was originally running for one hour and thirty minutes.

```
pcs resource move resource1 example-node2 lifetime=PT1H30M
```

The following command moves the resource **resource1** to node **example-node2** and prevents it from moving back to the node on which it was originally running for thirty minutes.

```
pcs resource move resource1 example-node2 lifetime=PT30M
```

22.2.2. Moving a resource to its preferred node

After a resource has moved, either due to a failover or to an administrator manually moving the node, it will not necessarily move back to its original node even after the circumstances that caused the failover have been corrected. To relocate resources to their preferred node, use the following command. A preferred node is determined by the current cluster status, constraints, resource location, and other settings and may change over time.

```
pcs resource relocate run [resource1] [resource2] ...
```

If you do not specify any resources, all resource are relocated to their preferred nodes.

This command calculates the preferred node for each resource while ignoring resource stickiness. After calculating the preferred node, it creates location constraints which will cause the resources to move to their preferred nodes. Once the resources have been moved, the constraints are deleted automatically. To remove all constraints created by the **pcs resource relocate run** command, you can enter the **pcs resource relocate clear** command. To display the current status of resources and their optimal node ignoring resource stickiness, enter the **pcs resource relocate show** command.

22.3. ENABLING, DISABLING, AND BANNING CLUSTER RESOURCES

In addition to the **pcs resource move** and **pcs resource relocate** commands, there are a variety of other commands you can use to control the behavior of cluster resources.

You can manually stop a running resource and prevent the cluster from starting it again with the following command. Depending on the rest of the configuration (constraints, options, failures, and so on), the resource may remain started. If you specify the **--wait** option, **pcs** will wait up to 'n' seconds for the resource to stop and then return 0 if the resource is stopped or 1 if the resource has not stopped. If 'n' is not specified it defaults to 60 minutes.

```
pcs resource disable resource_id [--wait[=n]]
```

You can use the following command to allow the cluster to start a resource. Depending on the rest of the configuration, the resource may remain stopped. If you specify the **--wait** option, **pcs** will wait up to 'n' seconds for the resource to start and then return 0 if the resource is started or 1 if the resource has not started. If 'n' is not specified it defaults to 60 minutes.

```
pcs resource enable resource_id [--wait[=n]]
```

Use the following command to prevent a resource from running on a specified node, or on the current node if no node is specified.

```
pcs resource ban resource_id [node] [--master] [lifetime=lifetime] [--wait[=n]]
```

Note that when you execute the **pcs resource ban** command, this adds a -INFINITY location constraint to the resource to prevent it from running on the indicated node. You can execute the **pcs resource clear** or the **pcs constraint delete** command to remove the constraint. This does not necessarily move the resources back to the indicated node; where the resources can run at that point depends on how you have configured your resources initially.

If you specify the **--master** parameter of the **pcs resource ban** command, the scope of the constraint is limited to the master role and you must specify *master_id* rather than *resource_id*.

You can optionally configure a **lifetime** parameter for the **pcs resource ban** command to indicate a period of time the constraint should remain.

You can optionally configure a **--wait[=*n*]** parameter for the **pcs resource ban** command to indicate the number of seconds to wait for the resource to start on the destination node before returning 0 if the resource is started or 1 if the resource has not yet started. If you do not specify n, the default resource timeout will be used.

You can use the **debug-start** parameter of the **pcs resource** command to force a specified resource to start on the current node, ignoring the cluster recommendations and printing the output from starting the resource. This is mainly used for debugging resources; starting resources on a cluster is (almost) always done by Pacemaker and not directly with a **pcs** command. If your resource is not starting, it is usually due to either a misconfiguration of the resource (which you debug in the system log), constraints that prevent the resource from starting, or the resource being disabled. You can use this command to test resource configuration, but it should not normally be used to start resources in a cluster.

The format of the **debug-start** command is as follows.

```
pcs resource debug-start resource_id
```

22.4. SETTING A RESOURCE TO UNMANAGED MODE

When a resource is in **unmanaged** mode, the resource is still in the configuration but Pacemaker does not manage the resource.

The following command sets the indicated resources to **unmanaged** mode.

```
pcs resource unmanage resource1 [resource2] ...
```

The following command sets resources to **managed** mode, which is the default state.

```
pcs resource manage resource1 [resource2] ...
```

You can specify the name of a resource group with the **pcs resource manage** or **pcs resource unmanage** command. The command will act on all of the resources in the group, so that you can set all of the resources in a group to **managed** or **unmanaged** mode with a single command and then manage the contained resources individually.

22.5. PUTTING A CLUSTER IN MAINTENANCE MODE

When a cluster is in maintenance mode, the cluster does not start or stop any services until told otherwise. When maintenance mode is completed, the cluster does a sanity check of the current state of any services, and then stops or starts any that need it.

To put a cluster in maintenance mode, use the following command to set the **maintenance-mode** cluster property to **true**.

```
# pcs property set maintenance-mode=true
```

To remove a cluster from maintenance mode, use the following command to set the **maintenance-mode** cluster property to **false**.

```
# pcs property set maintenance-mode=false
```

u can remove a cluster property from the configuration with the following command.

```
pcs property unset property
```

Alternately, you can remove a cluster property from a configuration by leaving the value field of the **pcs property set** command blank. This restores that property to its default value. For example, if you have previously set the **symmetric-cluster** property to **false**, the following command removes the value you have set from the configuration and restores the value of **symmetric-cluster** to **true**, which is its default value.

```
# pcs property set symmetric-cluster=
```

22.6. UPDATING A RHEL HIGH AVAILABILITY CLUSTER

Updating packages that make up the RHEL High Availability and Resilient Storage Add-Ons, either individually or as a whole, can be done in one of two general ways:

- *Rolling Updates*: Remove one node at a time from service, update its software, then integrate it back into the cluster. This allows the cluster to continue providing service and managing resources while each node is updated.

- *Entire Cluster Update*: Stop the entire cluster, apply updates to all nodes, then start the cluster back up.

WARNING

It is critical that when performing software update procedures for Red Hat Enterprise LInux High Availability and Resilient Storage clusters, you ensure that any node that will undergo updates is not an active member of the cluster before those updates are initiated.

For a full description of each of these methods and the procedures to follow for the updates, see Recommended Practices for Applying Software Updates to a RHEL High Availability or Resilient Storage Cluster.

22.7. UPGRADING REMOTE NODES AND GUEST NODES

If the **pacemaker_remote** service is stopped on an active remote node or guest node, the cluster will gracefully migrate resources off the node before stopping the node. This allows you to perform software upgrades and other routine maintenance procedures without removing the node from the cluster. Once **pacemaker_remote** is shut down, however, the cluster will immediately try to reconnect. If **pacemaker_remote** is not restarted within the resource's monitor timeout, the cluster will consider the monitor operation as failed.

If you wish to avoid monitor failures when the **pacemaker_remote** service is stopped on an active Pacemaker Remote node, you can use the following procedure to take the node out of the cluster before performing any system administration that might stop **pacemaker_remote**

1. Stop the node's connection resource with the **pcs resource disable *resourcename***, which will move all services off the node. For guest nodes, this will also stop the VM, so the VM must be started outside the cluster (for example, using **virsh**) to perform any maintenance.

2. Perform the required maintenance.

3. When ready to return the node to the cluster, re-enable the resource with the **pcs resource enable**.