Red Hat Enterprise Linux 9.0

Configuring and managing high availability clusters

Using the Red Hat High Availability Add-On to create and maintain Pacemaker clusters

Last Updated: 2024-10-04

Red Hat Enterprise Linux 9.0 Configuring and managing high availability clusters

Using the Red Hat High Availability Add-On to create and maintain Pacemaker clusters

Legal Notice

Copyright © 2024 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at
http://creativecommons.org/licenses/by-sa/3.0/
. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

Abstract

The Red Hat High Availability Add-On configures high availability clusters that use the Pacemaker cluster resource manager. This title provides procedures to familiarize you with Pacemaker cluster configuration as well as example procedures for configuring active/active and active/passive clusters.

Table of Contents

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate your feedback on our documentation. Let us know how we can improve it.

Submitting feedback through Jira (account required)

1. Log in to the Jira website.
2. Click **Create** in the top navigation bar
3. Enter a descriptive title in the **Summary** field.
4. Enter your suggestion for improvement in the **Description** field. Include links to the relevant parts of the documentation.
5. Click **Create** at the bottom of the dialogue.

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**.

Red Hat provides a variety of documentation for planning, configuring, and maintaining a Red Hat high availability cluster. For a listing of articles that provide guided indexes to the various areas of Red Hat cluster documentation, see the Red Hat High Availability Add-On Documentation Guide.

1.1. HIGH AVAILABILITY ADD-ON COMPONENTS

The Red Hat High Availability Add-On consists of several components that provide the high availability service.

The major components of the High Availability Add-On are as follows:

- 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.
- HAProxy — Routing software that provides high availability load balancing and failover in layer 4 (TCP) and layer 7 (HTTP, HTTPS) services.

1.2. HIGH AVAILABILITY ADD-ON CONCEPTS

Some of the key concepts of a Red Hat High Availability Add-On cluster are as follows.

1.2.1. Fencing

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. his 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 .

1.2.2. Quorum

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 about 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.

1.2.3. Cluster resources

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*.

1.3. 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.

1.3.1. 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 shutting 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.

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.

1.3.2. Pacemaker 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.

1.3.3. 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.

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.

1.4. 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.

The cluster configurations you can choose are as follows:

- 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.

1.4.1. 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 can 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.

1.4.2. Configuring LVM volumes in a cluster

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.

NOTE

If an LVM volume group used by a Pacemaker cluster contains one or more physical volumes that reside on remote block storage, such as an iSCSI target, Red Hat recommends that you configure a **systemd resource-agents-deps** target and a **systemd** drop-in unit for the target to ensure that the service starts before Pacemaker starts. For information on configuring a **systemd resource-agents-deps** target, see Configuring startup order for resource dependencies not managed by Pacemaker .

- 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 procedures for configuring shared LVM volumes that use the **lvmlockd** daemon to manage storage devices in active/active configurations, see GFS2 file systems in a cluster and Configuring an active/active Samba server in a Red Hat High Availability cluster .

CHAPTER 2. GETTING STARTED WITH PACEMAKER

To familiarize yourself with the tools and processes you use to create a Pacemaker cluster, you can run the following procedures. 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. For full information about Red Hat's support policies, requirements, and limitations for RHEL High Availability clusters, see Support Policies for RHEL High Availability Clusters .

2.1. LEARNING TO USE PACEMAKER

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.

In this example:

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

Prerequisites

- A single node running RHEL 9
- A floating IP address that resides on the same network as one of the node's statically assigned IP addresses
- The name of the node on which you are running is in your **/etc/hosts** file

Procedure

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

   ```
   # dnf 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.

```
# dnf 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

The following 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.

In this example:

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

Prerequisites

- Two nodes running RHEL 9 that can communicate with each other
- A floating IP address that resides on the same network as one of the node’s statically assigned IP addresses
- The name of the node on which you are running is in your **/etc/hosts** file

Procedure

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.

```
# dnf 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.

```
# dnf 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 node 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 use the **-h** option of **pcs** to display the parameters of a **pcs** command and a description of those parameters.

The following command displays the parameters of the **pcs resource** command.

```
# 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 about 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.

Procedure

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

There are a variety of commands you can use to display the status of a cluster and its components.

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
```

3.6. MODIFYING THE COROSYNC.CONF FILE WITH THE PCS COMMAND

You can use the **pcs** command to modify the parameters in the **corosync.conf** file.

The following command modifies the parameters in the **corosync.conf** file.

```
pcs cluster config update [transport pass:quotes[transport options]] [compression
pass:quotes[compression options]] [crypto pass:quotes[crypto options]] [totem pass:quotes[totem
options]] [--corosync_conf pass:quotes[path]]
```

The following example command udates the **knet_pmtud_interval** transport value and the **token** and **join** totem values.

```
# pcs cluster config update transport knet_pmtud_interval=35 totem token=10000 join=100
```

Additional resources

- For information about adding and removing nodes from an existing cluster, see Managing cluster nodes.
- For information about adding and modifying links in an existing cluster, see Adding and modifying links in an existing cluster.
- For information about modifyng quorum options and managing the quorum device settings in a cluster, see Configuring cluster quorum and Configuring quorum devices.

3.7. DISPLAYING THE COROSYNC.CONF FILE WITH THE PCS COMMAND

The following command displays the contents of the **corosync.conf** cluster configuration file.

```
# pcs cluster corosync
```

You can print the contents of the **corosync.conf** file in a human-readable format with the **pcs cluster config** command, as in the following example.

The output for this command includes the UUID for the cluster if the cluster was created in RHEL 9.1 or later, or if the UUID was added manually as described in Identifying clusters by UUID.

```
[root@r8-node-01 ~]# pcs cluster config
Cluster Name: HACluster
Cluster UUID: ad4ae07dcafe4066b01f1cc9391f54f5
Transport: knet
Nodes:
  r8-node-01:
    Link 0 address: r8-node-01
    Link 1 address: 192.168.122.121
```

```
      nodeid: 1
    r8-node-02:
      Link 0 address: r8-node-02
      Link 1 address: 192.168.122.122
      nodeid: 2
  Links:
    Link 1:
      linknumber: 1
      ping_interval: 1000
      ping_timeout: 2000
      pong_count: 5
  Compression Options:
    level: 9
    model: zlib
    threshold: 150
  Crypto Options:
    cipher: aes256
    hash: sha256
  Totem Options:
    downcheck: 2000
    join: 50
    token: 10000
  Quorum Device: net
    Options:
      sync_timeout: 2000
      timeout: 3000
    Model Options:
      algorithm: lms
      host: r8-node-03
    Heuristics:
      exec_ping: ping -c 1 127.0.0.1
```

You can run the **pcs cluster config show** command with the **--output-format=cmd** option to display the **pcs** configuration commands that can be used to recreate the existing **corosync.conf** file, as in the following example.

```
[root@r8-node-01 ~]# pcs cluster config show --output-format=cmd
pcs cluster setup HACluster \
  r8-node-01 addr=r8-node-01 addr=192.168.122.121 \
  r8-node-02 addr=r8-node-02 addr=192.168.122.122 \
  transport \
  knet \
    link \
      linknumber=1 \
      ping_interval=1000 \
      ping_timeout=2000 \
      pong_count=5 \
    compression \
      level=9 \
      model=zlib \
      threshold=150 \
    crypto \
      cipher=aes256 \
      hash=sha256 \
  totem \
downcheck=2000 \
join=50 \
token=10000
```

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

Create a Red Hat High Availability two-node cluster using the **pcs** command-line interface with the following procedure.

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**.

NOTE

You must ensure that your configuration conforms to Red Hat's support policies. For full information about Red Hat's support policies, requirements, and limitations for RHEL High Availability clusters, see Support Policies for RHEL High Availability Clusters .

4.1. INSTALLING CLUSTER SOFTWARE

Install the cluster software and configure your system for cluster creation with the following procedure.

Procedure

1. On each node in the cluster, enable the repository for high availability that corresponds to your system architecture. For example, to enable the high availability repository for an x86_64 system, you can enter the following **subscription-manager** command:

   ```
   # subscription-manager repos --enable=rhel-9-for-x86_64-highavailability-rpms
   ```

2. 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.

   ```
   # dnf 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.

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

   The following command displays a list 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 more information, see Recommended Practices for Applying Software Updates to a RHEL High Availability or Resilient Storage Cluster.

3. 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 it 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.

4. 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.
```

5. 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
```

4.2. INSTALLING THE PCP-ZEROCONF PACKAGE (RECOMMENDED)

When you set up your cluster, it is recommended that you install the **pcp-zeroconf** package for the Performance Co-Pilot (PCP) tool. PCP is Red Hat's recommended resource-monitoring tool for RHEL systems. Installing the **pcp-zeroconf** package allows you to have PCP running and collecting performance-monitoring data for the benefit of investigations into fencing, resource failures, and other events that disrupt the cluster.

NOTE

Cluster deployments where PCP is enabled will need sufficient space available for PCP's captured data on the file system that contains **/var/log/pcp/**. Typical space usage by PCP varies across deployments, but 10Gb is usually sufficient when using the **pcp-zeroconf** default settings, and some environments may require less. Monitoring usage in this directory over a 14-day period of typical activity can provide a more accurate usage expectation.

Procedure

To install the **pcp-zeroconf** package, run the following command.

```
# dnf install pcp-zeroconf
```

This package enables **pmcd** and sets up data capture at a 10-second interval.

For information about reviewing PCP data, see Why did a RHEL High Availability cluster node reboot - and how can I prevent it from happening again? on the Red Hat Customer Portal.

4.3. CREATING A HIGH AVAILABILITY CLUSTER

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

Procedure

1. 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 a two-node cluster that will consist of **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
```

2. 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
```

3. 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

...
```

4.4. 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 basic command to create a two-node cluster with two links is as follows.

```
pcs cluster setup pass:quotes[cluster_name] pass:quotes[node1_name]
addr=pass:quotes[node1_link0_address] addr=pass:quotes[node1_link1_address]
pass:quotes[node2_name] addr=pass:quotes[node2_link0_address]
addr=pass:quotes[node2_link1_address]
```

For the full syntax of this command, see the **pcs**(8) man page.

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.

- By default, if **link_priority** is not specified for a link, the link's priority is equal to the link number. The link priorities are then 0, 1, 2, 3, and so forth, according to the order specified, with 0 being the highest link priority.
- The default link mode is **passive**, meaning the active link with the lowest-numbered link priority is used.
- With the default values of **link_mode** and **link_priority**, the first link specified will be used as the highest priority link, and if that link fails the next link specified will be used.
- 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.
- It is possible to add, remove, and change links in an existing cluster using the **pcs cluster link add**, the **pcs cluster link remove**, the **pcs cluster link delete**, and the **pcs cluster link update** commands.
- As with single-link clusters, do not mix IPv4 and IPv6 addresses in one link, although 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
```

To set a link priority to a different value than the default value, which is the link number, you can set the link priority with the **link_priority** option of the **pcs cluster setup** command. Each of the following two example commands creates a two-node cluster with two interfaces where the first link, link 0, has a link priority of 1 and the second link, link 1, has a link priority of 0. Link 1 will be used first and link 0 will serve as the failover link. Since link mode is not specified, it defaults to passive.

These two commands are equivalent. If you do not specify a link number following the **link** keyword, the **pcs** interface automatically adds a link number, starting with the lowest unused link number.

```
# 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 transport knet
link link_priority=1 link link_priority=0

# 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 transport knet
link linknumber=1 link_priority=0 link link_priority=1
```

You can set the link mode to a different value than the default value of **passive** with the **link_mode** option of the **pcs cluster setup** command, as in the following example.

```
# 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 transport knet
link_mode=active
```

The following example sets both the link mode and the link priority.

```
# 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 transport knet
link_mode=active link link_priority=1 link link_priority=0
```

For information about adding nodes to an existing cluster with multiple links, see Adding a node to a cluster with multiple links.

For information about changing the links in an existing cluster with multiple links, see Adding and modifying links in an existing cluster.

4.5. 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 about fencing and its importance in a Red Hat High Availability cluster, see Fencing in a Red Hat High Availability Cluster .

NOTE

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

Procedure

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 fencing 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 about 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.

4.6. BACKING UP AND RESTORING A CLUSTER CONFIGURATION

The following commands back up a cluster configuration in a tar archive and restore the cluster configuration files on all nodes from the backup.

Procedure

Use the following command to back up the cluster configuration in a tar archive. 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 cluster nodes from the backup. Specifying the **--local** option restores the cluster configuration files only on the node from which you run this command. If you do not specify a file name, the standard input will be used.

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

4.7. 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.

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

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.

The following table 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 arbitrators or the quorum device host.
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.

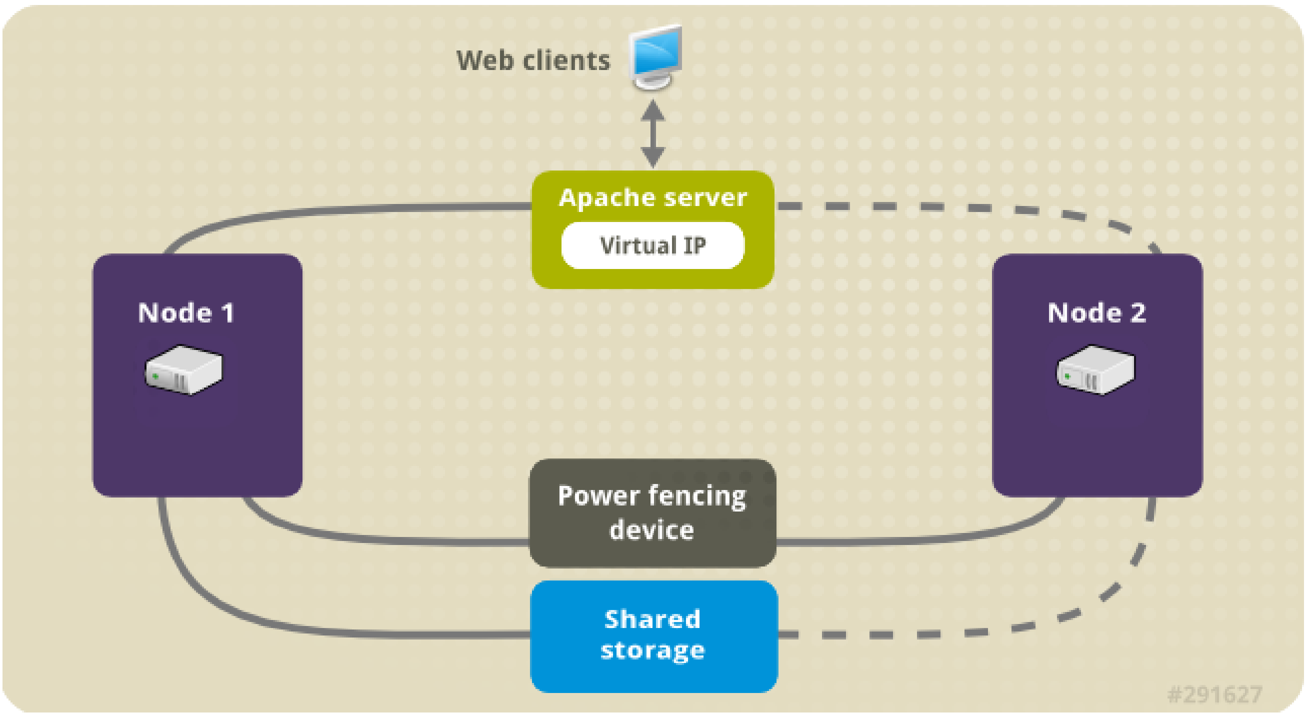
Port	When Required
TCP 5403	Required on the quorum device host when using a quorum device with **corosync-qnetd**. The default value can be changed with the **-p** option of the **corosync-qnetd** command.
UDP 5404-5412	Required on corosync nodes to facilitate communication between nodes. It is crucial 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**).
TCP 9929, UDP 9929	Required to be open on all cluster nodes and Booth arbitrator nodes to connections from any of those same nodes when the Booth ticket manager is used to establish a multi-site cluster.

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

Configure an active/passive Apache HTTP server in a two-node Red Hat Enterprise Linux High Availability Add-On cluster with the following procedure. 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.

The following illustration shows a high-level overview of the cluster in which 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. In this illustration, the web server is running on Node 1 while Node 2 is available to run the server if Node 1 becomes inoperative.

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, Fibre Channel, or other shared network block device.

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

1. Configure an XFS file system on the logical volume **my_lv**.
2. Configure a web server.

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

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

Create an LVM logical volume on storage that is shared between the nodes of the cluster with the following procedure.

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 XFS 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.

Procedure

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** identifier 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 XFS 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.

   NOTE

   If your LVM volume group contains one or more physical volumes that reside on remote block storage, such as an iSCSI target, Red Hat recommends that you ensure that the service starts before Pacemaker starts. For information about configuring startup order for a remote physical volume used by a Pacemaker cluster, see Configuring startup order for resource dependencies not managed by Pacemaker.

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

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

NOTE

If your LVM volume group contains one or more physical volumes that reside on remote block storage, such as an iSCSI target, Red Hat recommends that you ensure that the service starts before Pacemaker starts. For information about configuring startup order for a remote physical volume used by a Pacemaker cluster, see Configuring startup order for resource dependencies not managed by Pacemaker.

b. Create the volume group **my_vg** that consists of the physical volume **/dev/sdb1**. Specify the **--setautoactivation n** flag to ensure that volume groups managed by Pacemaker in a cluster will not be automatically activated on startup. If you are using an existing volume group for the LVM volume you are creating, you can reset this flag with the **vgchange --setautoactivation n** command for the volume group.

```
[root@z1 ~]# vgcreate --setautoactivation n 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.

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

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

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

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

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

```
[root@z1 ~]# mkfs.xfs /dev/my_vg/my_lv
meta-data=/dev/my_vg/my_lv       isize=512    agcount=4, agsize=28928 blks
         =                       sectsz=512   attr=2, projid32bit=1
...
```

3. If the use of a devices file is enabled with the **use_devicesfile = 1** parameter in the **lvm.conf** file, add the shared device to the devices file on the second node in the cluster. This feature is enabled by default.

```
[root@z2 ~]# lvmdevices --adddev /dev/sdb1
```

5.2. CONFIGURING AN APACHE HTTP SERVER

Configure an Apache HTTP Server with the following procedure.

Procedure

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.

   ```
   # dnf 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 and enable the ports you will require for running **httpd**. This example enables the **httpd** ports for public access, but the specific ports to enable for **httpd** may vary for production use.

   ```
   # firewall-cmd --permanent --add-service=http
   # firewall-cmd --permanent --zone=public --add-service=http
   # firewall-cmd --reload
   ```

2. 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
       Require local
   </Location>
   END
   ```

3. Create a web page for Apache to serve up.
   On one node in the cluster, ensure that the logical volume you created in Configuring an LVM volume with an XFS file system is activated, mount the file system that you created on that logical volume, create the file **index.html** on that file system, and then unmount the file system.

   ```
   # lvchange -ay my_vg/my_lv
   # 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

Create the resources for your cluster with the following procedure. 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-activate** resource named **my_lvm** that uses the LVM volume group you created in Configuring an LVM volume with an XFS 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 XFS 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.

Procedure

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

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

Replace the line you removed with the following three lines, specifying **/var/run/httpd-*website*.pid** as the PID file path where *website* is the name of the Apache resource. In this example, the Apache resource name is **Website**.

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

5.4. TESTING THE RESOURCE CONFIGURATION

Test the resource configuration in a cluster with the following procedure.

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.

Procedure

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-activate): Started z2.example.com
     my_fs (ocf::heartbeat:Filesystem): Started z2.example.com
     VirtualIP (ocf::heartbeat:IPaddr2): Started z2.example.com
     Website (ocf::heartbeat:apache): Started z2.example.com
```

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

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

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

NOTE

Removing a node from **standby** mode does not in itself cause the resources to fail back over to that node. This will depend on the **resource-stickiness** value for the resources. For information about the **resource-stickiness** meta attribute, see Configuring a resource to prefer its current node .

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

The Red Hat High Availability Add-On provides support for running a highly available active/passive NFS server on a Red Hat Enterprise Linux High Availability Add-On cluster using shared storage. In the following example, you are configuring a two-node cluster in which clients access the NFS file system through a floating IP address. The NFS server runs on one of the 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, Fibre Channel, or other shared network block device.

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 a file system on an LVM logical volume 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 XFS FILE SYSTEM IN A PACEMAKER CLUSTER

Create an LVM logical volume on storage that is shared between the nodes of the cluster with the following procedure.

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 XFS 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.

Procedure

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** identifier 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

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 XFS 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.

NOTE

If your LVM volume group contains one or more physical volumes that reside on remote block storage, such as an iSCSI target, Red Hat recommends that you ensure that the service starts before Pacemaker starts. For information about configuring startup order for a remote physical volume used by a Pacemaker cluster, see Configuring startup order for resource dependencies not managed by Pacemaker.

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

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

NOTE

If your LVM volume group contains one or more physical volumes that reside on remote block storage, such as an iSCSI target, Red Hat recommends that you ensure that the service starts before Pacemaker starts. For information about configuring startup order for a remote physical volume used by a Pacemaker cluster, see Configuring startup order for resource dependencies not managed by Pacemaker.

b. Create the volume group **my_vg** that consists of the physical volume **/dev/sdb1**. Specify the **--setautoactivation n** flag to ensure that volume groups managed by Pacemaker in a cluster will not be automatically activated on startup. If you are using an existing volume group for the LVM volume you are creating, you can reset this flag with the **vgchange --setautoactivation n** command for the volume group.

```
[root@z1 ~]# vgcreate --setautoactivation n 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.

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

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

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

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

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

```
[root@z1 ~]# mkfs.xfs /dev/my_vg/my_lv
meta-data=/dev/my_vg/my_lv       isize=512    agcount=4, agsize=28928 blks
         =                       sectsz=512   attr=2, projid32bit=1
...
```

3. If the use of a devices file is enabled with the **use_devicesfile = 1** parameter in the **lvm.conf** file, add the shared device to the devices file on the second node in the cluster. This feature is enabled by default.

```
[root@z2 ~]# lvmdevices --adddev /dev/sdb1
```

6.2. CONFIGURING AN NFS SHARE

Configure an NFS share for an NFS service failover with the following procedure.

Procedure

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. Ensure that the logical volume you you created in Configuring an LVM volume with an XFS file system is activated, then mount the file system you created on the logical volume on the /**nfsshare** directory.

```
[root@z1 ~]# lvchange -ay my_vg/my_lv
[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 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

Configure the cluster resources for an NFS server in a cluster with the following procedure.

NOTE

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.

Procedure

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

WARNING

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
```

2. 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-activate):   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
```

3. Configure a **Filesystem** resource for the cluster.
   The following command configures an XFS **Filesystem** resource named **nfsshare** as part of the **nfsgroup** resource group. This file system uses the LVM volume group and XFS file system you created in Configuring an LVM volume with an XFS 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=xfs --group nfsgroup
```

   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.

4. 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-activate):   Started z1.example.com
     nfsshare   (ocf::heartbeat:Filesystem):    Started z1.example.com
...
```

5. Create the **nfsserver** resource named **nfs-daemon** as 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 servers 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 servers are storing their stateful information on a device that will become available to another node if this resource group needs 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
   - **/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
   ...
   ```

6. 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
   ```

7. 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
```

8. 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
```

9. 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-activate):   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 NFS resource configuration in a high availability cluster with the following procedures. You should be able to mount the exported file system with either NFSv3 or NFSv4.

6.4.1. Testing the NFS export

1. If you are running the **firewalld** daemon on your cluster nodes, ensure that the ports that your system requires for NFS access are enabled on all nodes.

2. 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
```

3. 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
```

4. 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
```

6.4.2. Testing for failover

1. On a node outside of the cluster, mount the NFS share and verify access to the **clientdatafile1** file you created in Configuring an NFS share .

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

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

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

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

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

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

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

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

```
# ls nfsshare
clientdatafile1
```

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

6. From a node within the cluster, remove the node that was initially running **nfsgroup** from standby mode.

   NOTE

   Removing a node from **standby** mode does not in itself cause the resources to fail back over to that node. This will depend on the **resource-stickiness** value for the resources. For information about the **resource-stickiness** meta attribute, see Configuring a resource to prefer its current node .

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

CHAPTER 7. GFS2 FILE SYSTEMS IN A CLUSTER

Use the following administrative procedures to configure GFS2 file systems in a Red Hat high availability cluster.

7.1. CONFIGURING A GFS2 FILE SYSTEM IN A CLUSTER

You can set up a Pacemaker cluster that includes GFS2 file systems with the following procedure. In this example, you create three GFS2 file systems on three logical volumes in a two-node cluster.

Prerequisites

- Install and start the cluster software on both cluster nodes and create a basic two-node cluster.
- Configure fencing for the cluster.

For information about creating a Pacemaker cluster and configuring fencing for the cluster, see Creating a Red Hat High-Availability cluster with Pacemaker .

Procedure

1. On both nodes in the cluster, enable the repository for Resilient Storage that corresponds to your system architecture. For example, to enable the Resilient Storage repository for an x86_64 system, you can enter the following **subscription-manager** command:

   ```
   # subscription-manager repos --enable=rhel-9-for-x86_64-resilientstorage-rpms
   ```

   Note that the Resilient Storage repository is a superset of the High Availability repository. If you enable the Resilient Storage repository you do not also need to enable the High Availability repository.

2. On both nodes of the cluster, install the **lvm2-lockd**, **gfs2-utils**, and **dlm** packages. To support these packages, you must be subscribed to the AppStream channel and the Resilient Storage channel.

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

3. On both nodes of the cluster, set the **use_lvmlockd** configuration option in the **/etc/lvm/lvm.conf** file to **use_lvmlockd=1**.

   ```
   ...
   use_lvmlockd = 1
   ...
   ```

4. Set the global Pacemaker parameter **no-quorum-policy** to **freeze**.

NOTE

By default, the value of **no-quorum-policy** is set to **stop**, indicating that once quorum is lost, all the resources on the remaining partition will immediately be stopped. Typically this default is the safest and most optimal option, but unlike most resources, GFS2 requires quorum to function. When quorum is lost both the applications using the GFS2 mounts and the GFS2 mount itself cannot be correctly stopped. Any attempts to stop these resources without quorum will fail which will ultimately result in the entire cluster being fenced every time quorum is lost.

To address this situation, set **no-quorum-policy** to **freeze** when GFS2 is in use. This means that when quorum is lost, the remaining partition will do nothing until quorum is regained.

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

5. 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
```

6. 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
```

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

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

8. 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 ]
```

9. 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.

   NOTE

   If your LVM volume group contains one or more physical volumes that reside on remote block storage, such as an iSCSI target, Red Hat recommends that you ensure that the service starts before Pacemaker starts. For information about configuring startup order for a remote physical volume used by a Pacemaker cluster, see Configuring startup order for resource dependencies not managed by Pacemaker.

   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...
```

10. On the second node in the cluster:

    a. If the use of a devices file is enabled with the **use_devicesfile = 1** parameter in the **lvm.conf** file, add the shared devices to the devices file This feature is enabled by default.

```
[root@z2 ~]# lvmdevices --adddev /dev/vdb
[root@z2 ~]# lvmdevices --adddev /dev/vdc
```

    b. Start the lock manager for each of the shared volume groups.

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

11. On one node in the cluster, create the shared logical volumes and format the volumes with a GFS2 file system. One journal is required for each node that mounts the file system. Ensure that you create enough journals for each of the nodes in your cluster. The format of the lock table name is *ClusterName:FSName* where *ClusterName* is the name of the cluster for which the GFS2 file system is being created and *FSName* is the file system name, which must be unique for all **lock_dlm** file systems over the 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
```

12. 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
    ```

13. Clone the two new resource groups.

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

14. 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)
    ```

15. Configure colocation constraints to ensure that the **vg1** and **vg2** resource groups start on the same node as the **locking** resource group.

```
[root@z1 ~]# pcs constraint colocation add shared_vg1-clone with locking-clone
[root@z1 ~]# pcs constraint colocation add shared_vg2-clone with locking-clone
```

16. 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
```

17. 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 to display the 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
```

Verification

1. 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)
```

2. Check the status of the cluster.

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

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 z1.example.com
     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 ]

...
```

Additional resources

- Configuring GFS2 file systems
- Configuring a Red Hat High Availability cluster on Microsoft Azure
- Configuring a Red Hat High Availability cluster on AWS
- Configuring a Red Hat High Availability Cluster on Google Cloud Platform

7.2. CONFIGURING AN ENCRYPTED GFS2 FILE SYSTEM IN A CLUSTER

You can create a Pacemaker cluster that includes a LUKS encrypted GFS2 file system with the following procedure. In this example, you create one GFS2 file systems on a logical volume and encrypt the file system. Encrypted GFS2 file systems are supported using the **crypt** resource agent, which provides support for LUKS encryption.

There are three parts to this procedure:

- Configuring a shared logical volume in a Pacemaker cluster
- Encrypting the logical volume and creating a **crypt** resource
- Formatting the encrypted logical volume with a GFS2 file system and creating a file system resource for the cluster

7.2.1. Configure a shared logical volume in a Pacemaker cluster

Prerequisites

- Install and start the cluster software on two cluster nodes and create a basic two-node cluster.
- Configure fencing for the cluster.

For information about creating a Pacemaker cluster and configuring fencing for the cluster, see Creating a Red Hat High-Availability cluster with Pacemaker .

Procedure

1. On both nodes in the cluster, enable the repository for Resilient Storage that corresponds to your system architecture. For example, to enable the Resilient Storage repository for an x86_64 system, you can enter the following **subscription-manager** command:

   ```
   # subscription-manager repos --enable=rhel-9-for-x86_64-resilientstorage-rpms
   ```

   Note that the Resilient Storage repository is a superset of the High Availability repository. If you enable the Resilient Storage repository you do not also need to enable the High Availability repository.

2. On both nodes of the cluster, install the **lvm2-lockd**, **gfs2-utils**, and **dlm** packages. To support these packages, you must be subscribed to the AppStream channel and the Resilient Storage channel.

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

3. On both nodes of the cluster, set the **use_lvmlockd** configuration option in the **/etc/lvm/lvm.conf** file to **use_lvmlockd=1**.

   ```
   ...
   use_lvmlockd = 1
   ...
   ```

4. Set the global Pacemaker parameter **no-quorum-policy** to **freeze**.

NOTE

By default, the value of **no-quorum-policy** is set to **stop**, indicating that when quorum is lost, all the resources on the remaining partition will immediately be stopped. Typically this default is the safest and most optimal option, but unlike most resources, GFS2 requires quorum to function. When quorum is lost both the applications using the GFS2 mounts and the GFS2 mount itself cannot be correctly stopped. Any attempts to stop these resources without quorum will fail which will ultimately result in the entire cluster being fenced every time quorum is lost.

To address this situation, set **no-quorum-policy** to **freeze** when GFS2 is in use. This means that when quorum is lost, the remaining partition will do nothing until quorum is regained.

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

5. 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
```

6. 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
```

7. 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
```

8. 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 ]
```

9. On one node of the cluster, create a shared volume group.

   NOTE

   If your LVM volume group contains one or more physical volumes that reside on remote block storage, such as an iSCSI target, Red Hat recommends that you ensure that the service starts before Pacemaker starts. For information about configuring startup order for a remote physical volume used by a Pacemaker cluster, see Configuring startup order for resource dependencies not managed by Pacemaker.

   The following command creates the shared volume group **shared_vg1** on **/dev/sda1**.

   ```
   [root@z1 ~]# vgcreate --shared shared_vg1 /dev/sda1
     Physical volume "/dev/sda1" successfully created.
     Volume group "shared_vg1" successfully created
     VG shared_vg1 starting dlm lockspace
     Starting locking.  Waiting until locks are ready...
   ```

10. On the second node in the cluster:

    a. If the use of a devices file is enabled with the **use_devicesfile = 1** parameter in the **lvm.conf** file, add the shared device to the devices file on the second node in the cluster. This feature is enabled by default.

    ```
    [root@z2 ~]# lvmdevices --adddev /dev/sda1
    ```

    b. Start the lock manager for the shared volume group.

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

11. On one node in the cluster, create the shared logical volume.

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

12. Create an **LVM-activate** resource for the logical volume to automatically activate the logical volume on all nodes.
    The following command creates 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
    ```

13. Clone the new resource group.

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

14. Configure an 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)
    ```

15. Configure a colocation constraints to ensure that the **vg1** and **vg2** resource groups start on the same node as the **locking** resource group.

    ```
    [root@z1 ~]# pcs constraint colocation add shared_vg1-clone with locking-clone
    ```

Verification

On both nodes in the cluster, verify that the logical volume is 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

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

7.2.2. Encrypt the logical volume and create a crypt resource

Prerequisites

- You have configured a shared logical volume in a Pacemaker cluster.

Procedure

1. On one node in the cluster, create a new file that will contain the crypt key and set the permissions on the file so that it is readable only by root.

    ```
    [root@z1 ~]# touch /etc/crypt_keyfile
    [root@z1 ~]# chmod 600 /etc/crypt_keyfile
    ```

2. Create the crypt key.

    ```
    [root@z1 ~]# dd if=/dev/urandom bs=4K count=1 of=/etc/crypt_keyfile
    1+0 records in
    1+0 records out
    4096 bytes (4.1 kB, 4.0 KiB) copied, 0.000306202 s, 13.4 MB/s
    [root@z1 ~]# scp /etc/crypt_keyfile root@z2.example.com:/etc/
    ```

3. Distribute the crypt keyfile to the other nodes in the cluster, using the **-p** parameter to preserve the permissions you set.

    ```
    [root@z1 ~]# scp -p /etc/crypt_keyfile root@z2.example.com:/etc/
    ```

4. Create the encrypted device on the LVM volume where you will configure the encrypted GFS2 file system.

   ```
   [root@z1 ~]# cryptsetup luksFormat /dev/shared_vg1/shared_lv1 --type luks2 --key-file=/etc/crypt_keyfile
   WARNING!
   ========
   This will overwrite data on /dev/shared_vg1/shared_lv1 irrevocably.

   Are you sure? (Type 'yes' in capital letters): YES
   ```

5. Create the crypt resource as part of the **shared_vg1** volume group.

   ```
   [root@z1 ~]# pcs resource create crypt --group shared_vg1 ocf:heartbeat:crypt
   crypt_dev="luks_lv1" crypt_type=luks2 key_file=/etc/crypt_keyfile
   encrypted_dev="/dev/shared_vg1/shared_lv1"
   ```

Verification

Ensure that the crypt resource has created the crypt device, which in this example is **/dev/mapper/luks_lv1**.

```
[root@z1 ~]# ls -l /dev/mapper/
...
lrwxrwxrwx 1 root root 7 Mar 4 09:52 luks_lv1 -> ../dm-3
...
```

7.2.3. Format the encrypted logical volume with a GFS2 file system and create a file system resource for the cluster

Prerequisites

- You have encrypted the logical volume and created a crypt resource.

Procedure

1. On one node in the cluster, format the volume with a GFS2 file system. One journal is required for each node that mounts the file system. Ensure that you create enough journals for each of the nodes in your cluster. The format of the lock table name is *ClusterName:FSName* where *ClusterName* is the name of the cluster for which the GFS2 file system is being created and *FSName* is the file system name, which must be unique for all **lock_dlm** file systems over the cluster.

   ```
   [root@z1 ~]# mkfs.gfs2 -j3 -p lock_dlm -t my_cluster:gfs2-demo1 /dev/mapper/luks_lv1
   /dev/mapper/luks_lv1 is a symbolic link to /dev/dm-3
   This will destroy any data on /dev/dm-3
   Are you sure you want to proceed? [y/n] y
   Discarding device contents (may take a while on large devices): Done
   Adding journals: Done
   Building resource groups: Done
   Creating quota file: Done
   Writing superblock and syncing: Done
   Device:                    /dev/mapper/luks_lv1
   Block size:                4096
   ```

```
Device size:              4.98 GB (1306624 blocks)
Filesystem size:          4.98 GB (1306622 blocks)
Journals:                 3
Journal size:             16MB
Resource groups:          23
Locking protocol:         "lock_dlm"
Lock table:               "my_cluster:gfs2-demo1"
UUID:                     de263f7b-0f12-4d02-bbb2-56642fade293
```

2. Create a file system resource to automatically mount the GFS2 file system on all nodes.
   Do 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 command creates the file system resource. This command adds the 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/mapper/luks_lv1" directory="/mnt/gfs1"
fstype="gfs2" options=noatime op monitor interval=10s on-fail=fence
```

Verification

1. Verify that the GFS2 file system is mounted on both nodes of the cluster.

```
[root@z1 ~]# mount | grep gfs2
/dev/mapper/luks_lv1 on /mnt/gfs1 type gfs2 (rw,noatime,seclabel)

[root@z2 ~]# mount | grep gfs2
/dev/mapper/luks_lv1 on /mnt/gfs1 type gfs2 (rw,noatime,seclabel)
```

2. Check the status of the cluster.

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

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
            crypt       (ocf::heartbeat:crypt) Started z2.example.com
            sharedfs1      (ocf::heartbeat:Filesystem):    Started z2.example.com
```

```
    Resource Group: shared_vg1:1
        sharedlv1      (ocf::heartbeat:LVM-activate):  Started z1.example.com
        crypt      (ocf::heartbeat:crypt)  Started z1.example.com
        sharedfs1      (ocf::heartbeat:Filesystem):    Started z1.example.com
      Started:  [z1.example.com z2.example.com ]
...
```

Additional resources

- Configuring GFS2 file systems

CHAPTER 8. CONFIGURING AN ACTIVE/ACTIVE SAMBA SERVER IN A RED HAT HIGH AVAILABILITY CLUSTER

The Red Hat High Availability Add-On provides support for configuring Samba in an active/active cluster configuration. In the following example, you are configuring an active/active Samba server on a two-node RHEL cluster.

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

To configure Samba in an active/active cluster:

1. Configure a GFS2 file system and its associated cluster resources.
2. Configure Samba on the cluster nodes.
3. Configure the Samba cluster resources.
4. Test the Samba server you have configured.

8.1. CONFIGURING A GFS2 FILE SYSTEM FOR A SAMBA SERVICE IN A HIGH AVAILABILITY CLUSTER

Before configuring an active/active Samba service in a Pacemaker cluster, configure a GFS2 file system for the cluster.

Prerequisites

- A two-node Red Hat High Availability cluster with fencing configured for each node
- Shared storage available for each cluster node
- A subscription to the AppStream channel and the Resilient Storage channel for each cluster node

For information about creating a Pacemaker cluster and configuring fencing for the cluster, see Creating a Red Hat High-Availability cluster with Pacemaker .

Procedure

1. On both nodes in the cluster, perform the following initial setup steps.
   a. Enable the repository for Resilient Storage that corresponds to your system architecture. For example, to enable the Resilient Storage repository for an x86_64 system, enter the following **subscription-manager** command:

      ```
      # subscription-manager repos --enable=rhel-9-for-x86_64-resilientstorage-rpms
      ```

      The Resilient Storage repository is a superset of the High Availability repository. If you enable the Resilient Storage repository, you do not need to also enable the High Availability repository.
   b. Install the **lvm2-lockd**, **gfs2-utils**, and **dlm** packages.

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

c. Set the **use_lvmlockd** configuration option in the **/etc/lvm/lvm.conf** file to **use_lvmlockd=1**.

```
...

use_lvmlockd = 1

...
```

2. On one node in the cluster, set the global Pacemaker parameter **no-quorum-policy** to **freeze**.

> NOTE
>
> By default, the value of **no-quorum-policy** is set to **stop**, indicating that once quorum is lost, all the resources on the remaining partition will immediately be stopped. Typically this default is the safest and most optimal option, but unlike most resources, GFS2 requires quorum to function. When quorum is lost both the applications using the GFS2 mounts and the GFS2 mount itself cannot be correctly stopped. Any attempts to stop these resources without quorum will fail which will ultimately result in the entire cluster being fenced every time quorum is lost.
>
> To address this situation, set **no-quorum-policy** to **freeze** when GFS2 is in use. This means that when quorum is lost, the remaining partition will do nothing until quorum is regained.

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

3. 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**. If you have not previously configured fencing for the cluster, this step fails and the **pcs status** command displays a resource failure message.

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

4. 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
```

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

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

6. Create a physical volume and a shared volume group on the shared device **/dev/vdb**. This example creates the shared volume group **csmb_vg**.

```
[root@z1 ~]# pvcreate /dev/vdb
[root@z1 ~]# vgcreate -Ay --shared csmb_vg /dev/vdb
```

```
Volume group "csmb_vg" successfully created
VG csmb_vg starting dlm lockspace
Starting locking.  Waiting until locks are ready
```

7. On the second node in the cluster:

8. If the use of a devices file is enabled with the **use_devicesfile = 1** parameter in the **lvm.conf** file, add the shared device to the devices file on the second node in the cluster. This feature is enabled by default.

   ```
   [root@z2 ~]# lvmdevices --adddev /dev/vdb
   ```

   a. Start the lock manager for the shared volume group.

      ```
      [root@z2 ~]# vgchange --lockstart csmb_vg
        VG csmb_vg starting dlm lockspace
        Starting locking.  Waiting until locks are ready...
      ```

9. On one node in the cluster, create a logical volume and format the volume with a GFS2 file system that will be used exclusively by CTDB for internal locking. Only one such file system is required in a cluster even if your deployment exports multiple shares.
   When specifying the lock table name with the **-t** option of the **mkfs.gfs2** command, ensure that the first component of the *clustername:filesystemname* you specify matches the name of your cluster. In this example, the cluster name is **my_cluster**.

   ```
   [root@z1 ~]# lvcreate -L1G -n ctdb_lv csmb_vg
   [root@z1 ~]# mkfs.gfs2 -j3 -p lock_dlm -t my_cluster:ctdb /dev/csmb_vg/ctdb_lv
   ```

10. Create a logical volume for each GFS2 file system that will be shared over Samba and format the volume with the GFS2 file system. This example creates a single GFS2 file system and Samba share, but you can create multiple file systems and shares.

    ```
    [root@z1 ~]# lvcreate -L50G -n csmb_lv1 csmb_vg
    [root@z1 ~]# mkfs.gfs2 -j3 -p lock_dlm -t my_cluster:csmb1 /dev/csmb_vg/csmb_lv1
    ```

11. Set up **LVM_Activate** resources to ensure that the required shared volumes are activated. This example creates the **LVM_Activate** resources as part of a resource group **shared_vg**, and then clones that resource group so that it runs on all nodes in the cluster.
    Create the resources as disabled so they do not start automatically before you have configured the necessary order constraints.

    ```
    [root@z1 ~]# pcs resource create --disabled --group shared_vg ctdb_lv
    ocf:heartbeat:LVM-activate lvname=ctdb_lv vgname=csmb_vg
    activation_mode=shared vg_access_mode=lvmlockd
    [root@z1 ~]# pcs resource create --disabled --group shared_vg csmb_lv1
    ocf:heartbeat:LVM-activate lvname=csmb_lv1 vgname=csmb_vg
    activation_mode=shared vg_access_mode=lvmlockd
    [root@z1 ~]# pcs resource clone shared_vg interleave=true
    ```

12. Configure an ordering constraint to start all members of the **locking** resource group before the members of the **shared_vg** resource group.

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

13. Enable the **LVM-activate** resources.

```
[root@z1 ~]# pcs resource enable ctdb_lv csmb_lv1
```

14. On one node in the cluster, perform the following steps to create the **Filesystem** resources you require.

    a. Create **Filesystem** resources as cloned resources, using the GFS2 file systems you previously configured on your LVM volumes. This configures Pacemaker to mount and manage file systems.

    NOTE

    You should not add the file system to the **/etc/fstab** file because it will be managed as a Pacemaker cluster resource. You can specify mount options as part of the resource configuration with **options=*options***. Run the **pcs resource describe Filesystem** command to display the full configuration options.

    ```
    [root@z1 ~]# pcs resource create ctdb_fs Filesystem
    device="/dev/csmb_vg/ctdb_lv" directory="/mnt/ctdb" fstype="gfs2" op monitor
    interval=10s on-fail=fence clone interleave=true
    [root@z1 ~]# pcs resource create csmb_fs1 Filesystem
    device="/dev/csmb_vg/csmb_lv1" directory="/srv/samba/share1" fstype="gfs2" op
    monitor interval=10s on-fail=fence clone interleave=true
    ```

    b. Configure ordering constraints to ensure that Pacemaker mounts the file systems after the shared volume group **shared_vg** has started.

    ```
    [root@z1 ~]# pcs constraint order start shared_vg-clone then ctdb_fs-clone
    Adding shared_vg-clone ctdb_fs-clone (kind: Mandatory) (Options: first-action=start then-
    action=start)
    [root@z1 ~]# pcs constraint order start shared_vg-clone then csmb_fs1-clone
    Adding shared_vg-clone csmb_fs1-clone (kind: Mandatory) (Options: first-action=start
    then-action=start)
    ```

8.2. CONFIGURING SAMBA IN A HIGH AVAILABILITY CLUSTER

To configure a Samba service in a Pacemaker cluster, configure the service on all nodes in the cluster.

Prerequisites

- A two-node Red Hat High Availability cluster configured with a GFS2 file system, as described in Configuring a GFS2 file system for a Samba service in a high availability cluster .
- A public directory created on your GFS2 file system to use for the Samba share. In this example, the directory is **/srv/samba/share1**.
- Public virtual IP addresses that can be used to access the Samba share exported by this cluster.

Procedure

1. On both nodes in the cluster, configure the Samba service and set up a share definition:

   a. Install the Samba and CTDB packages.

   ```
   # dnf -y install samba ctdb cifs-utils samba-winbind
   ```

   b. Ensure that the **ctdb**, **smb**, **nmb**, and **winbind** services are not running and do not start at boot.

   ```
   # systemctl disable --now ctdb smb nmb winbind
   ```

   c. In the **/etc/samba/smb.conf** file, configure the Samba service and set up the share definition, as in the following example for a standalone server with one share.

   ```
   [global]
       netbios name = linuxserver
       workgroup = WORKGROUP
       security = user
       clustering = yes
   [share1]
       path = /srv/samba/share1
       read only = no
   ```

   d. Verify the **/etc/samba/smb.conf** file.

   ```
   # testparm
   ```

2. On both nodes in the cluster, configure CTDB:

   a. Create the **/etc/ctdb/nodes** file and add the IP addresses of the cluster nodes, as in this example nodes file.

   ```
   192.0.2.11
   192.0.2.12
   ```

   b. Create the **/etc/ctdb/public_addresses** file and add the IP addresses and network device names of the cluster's public interfaces to the file. When assigning IP addresses in the **public_addresses** file, ensure that these addresses are not in use and that those addresses are routable from the intended client. The second field in each entry of the **/etc/ctdb/public_addresses** file is the interface to use on the cluster machines for the corresponding public address. In this example **public_addresses** file, the interface **enp1s0** is used for all the public addresses.

   ```
   192.0.2.201/24 enp1s0
   192.0.2.202/24 enp1s0
   ```

   The public interfaces of the cluster are the ones that clients use to access Samba from their network. For load balancing purposes, add an A record for each public IP address of the cluster to your DNS zone. Each of these records must resolve to the same hostname. Clients use the hostname to access Samba and DNS distributes the clients to the different nodes of the cluster.

c. If you are running the **firewalld** service, enable the ports that are required by the **ctdb** and **samba** services.

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

3. On one node in the cluster, update the SELinux contexts:

a. Update the SELinux contexts on the GFS2 share.

```
[root@z1 ~]# semanage fcontext -at ctdbd_var_run_t -s system_u "/mnt/ctdb(/.)?"
[root@z1 ~]# restorecon -Rv /mnt/ctdb
```

b. Update the SELinux context on the directory shared in Samba.

```
[root@z1 ~]# semanage fcontext -at samba_share_t -s system_u
"/srv/samba/share1(/.)?"
[root@z1 ~]# restorecon -Rv /srv/samba/share1
```

Additional resources

- For further information about configuring Samba as a standalone server, as in this example, see the Using Samba as a server chapter of Configuring and using network file services .
- Setting up a forward zone on a BIND primary server .

8.3. CONFIGURING SAMBA CLUSTER RESOURCES

After configuring a Samba service on both nodes of a two-node high availability cluster, configure the Samba cluster resources for the cluster.

Prerequisites

- A two-node Red Hat High Availability cluster configured with a GFS2 file system, as described in Configuring a GFS2 file system for a Samba service in a high availability cluster .
- Samba service configured on both cluster nodes, as described in Configuring Samba in a high availability cluster.

Procedure

1. On one node in the cluster, configure the Samba cluster resources:

a. Create the CTDB resource, in group **samba-group**. The CTDB resource agent uses the **ctdb_*** options specified with the **pcs** command to create the CTDB configuration file. Create the resource as disabled so it does not start automatically before you have configured the necessary order constraints.

```
[root@z1 ~]# pcs resource create --disabled ctdb --group samba-group
ocf:heartbeat:CTDB ctdb_recovery_lock=/mnt/ctdb/ctdb.lock
ctdb_dbdir=/var/lib/ctdb ctdb_logfile=/var/log/ctdb.log op monitor interval=10
timeout=30 op start timeout=90 op stop timeout=100
```

b. Clone the **samba-group** resource group.

```
[root@z1 ~]# pcs resource clone samba-group
```

c. Create ordering constraints to ensure that all **Filesystem** resources are running before the resources in **samba-group**.

```
[root@z1 ~]# pcs constraint order start ctdb_fs-clone then samba-group-clone
[root@z1 ~]# pcs constraint order start csmb_fs1-clone then samba-group-clone
```

d. Create the **samba** resource in the resource group **samba-group**. This creates an implicit ordering constraint between CTDB and Samba, based on the order they are added.

```
[root@z1 ~]# pcs resource create samba --group samba-group systemd:smb
```

e. Enable the **ctdb** and **samba** resources.

```
[root@z1 ~]# pcs resource enable ctdb samba
```

f. Check that all the services have started successfully.

NOTE

It can take a couple of minutes for CTDB to start Samba, export the shares, and stabilize. If you check the cluster status before this process has completed, you may see that the **samba** services are not yet running.

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

...

Full List of Resources:
  * fence-z1   (stonith:fence_xvm): Started z1.example.com
  * fence-z2   (stonith:fence_xvm): Started z2.example.com
  * Clone Set: locking-clone [locking]:
 * Started: [ z1.example.com z2.example.com ]
  * Clone Set: shared_vg-clone [shared_vg]:
 * Started: [ z1.example.com z2.example.com ]
  * Clone Set: ctdb_fs-clone [ctdb_fs]:
 * Started: [ z1.example.com z2.example.com ]
  * Clone Set: csmb_fs1-clone [csmb_fs1]:
 * Started: [ z1.example.com z2.example.com ]
   * Clone Set: samba-group-clone [samba-group]:
 * Started: [ z1.example.com z2.example.com ]
```

2. On both nodes in the cluster, add a local user for the test share directory.

a. Add the user.

```
# useradd -M -s /sbin/nologin example_user
```

b. Set a password for the user.

```
# passwd example_user
```

c. Set an SMB password for the user.

```
# smbpasswd -a example_user
New SMB password:
Retype new SMB password:
Added user example_user
```

d. Activate the user in the Samba database.

```
# smbpasswd -e example_user
```

e. Update the file ownership and permissions on the GFS2 share for the Samba user.

```
# chown example_user:users /srv/samba/share1/
# chmod 755 /srv/samba/share1/
```

8.4. VERIFYING CLUSTERED SAMBA CONFIGURATION

If your clustered Samba configuration was successful, you are able to mount the Samba share. After mounting the share, you can test for Samba recovery if the cluster node that is exporting the Samba share becomes unavailable.

Procedure

1. On a system that has access to one or more of the public IP addresses configured in the **/etc/ctdb/public_addresses** file on the cluster nodes, mount the Samba share using one of these public IP addresses.

```
[root@testmount ~]# mkdir /mnt/sambashare
[root@testmount ~]# mount -t cifs -o user=example_user //192.0.2.201/share1
/mnt/sambashare
Password for example_user@//192.0.2.201/public: XXXXXXX
```

2. Verify that the file system is mounted.

```
[root@testmount ~]# mount | grep /mnt/sambashare
//192.0.2.201/public on /mnt/sambashare type cifs
(rw,relatime,vers=1.0,cache=strict,username=example_user,domain=LINUXSERVER,uid=0,nof
orceuid,gid=0,noforcegid,addr=192.0.2.201,unix,posixpaths,serverino,mapposix,acl,rsize=10485
76,wsize=65536,echo_interval=60,actimeo=1,user=example_user)
```

3. Verify that you can create a file on the mounted file system.

```
[root@testmount ~]# touch /mnt/sambashare/testfile1
[root@testmount ~]# ls /mnt/sambashare
testfile1
```

4. Determine which cluster node is exporting the Samba share:

   a. On each cluster node, display the IP addresses assigned to the interface specified in the **public_addresses** file. The following commands display the IPv4 addresses assigned to the **enp1s0** interface on each node.

```
[root@z1 ~]# ip -4 addr show enp1s0 | grep inet
     inet 192.0.2.11/24 brd 192.0.2.255 scope global dynamic noprefixroute enp1s0
     inet 192.0.2.201/24 brd 192.0.2.255 scope global secondary enp1s0

[root@z2 ~]# ip -4 addr show enp1s0 | grep inet
     inet 192.0.2.12/24 brd 192.0.2.255 scope global dynamic noprefixroute enp1s0
     inet 192.0.2.202/24 brd 192.0.2.255 scope global secondary enp1s0
```

b. In the output of the **ip** command, find the node with the IP address you specified with the **mount** command when you mounted the share.
   In this example, the IP address specified in the mount command is 192.0.2.201. The output of the **ip** command shows that the IP address 192.0.2.201 is assigned to **z1.example.com**.

5. Put the node exporting the Samba share in **standby** mode, which will cause the node to be unable to host any cluster resources.

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

6. From the system on which you mounted the file system, verify that you can still create a file on the file system.

```
[root@testmount ~]# touch /mnt/sambashare/testfile2
[root@testmount ~]# ls /mnt/sambashare
testfile1  testfile2
```

7. Delete the files you have created to verify that the file system has successfully mounted. If you no longer require the file system to be mounted, unmount it at this point.

```
[root@testmount ~]# rm /mnt/sambashare/testfile1 /mnt/sambashare/testfile2
rm: remove regular empty file '/mnt/sambashare/testfile1'? y
rm: remove regular empty file '/mnt/sambashare/testfile1'? y
[root@testmount ~]# umount /mnt/sambashare
```

8. From one of the cluster nodes, restore cluster services to the node that you previously put into standby mode. This will not necessarily move the service back to that node.

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

CHAPTER 9. GETTING STARTED WITH THE PCSD WEB UI

The **pcsd** Web UI is a graphical user interface to create and configure Pacemaker/Corosync clusters.

9.1. SETTING UP THE PCSD WEB UI

Set up your system to use the **pcsd** Web UI to configure a cluster with the following procedure.

Prerequisites

- The Pacemaker configuration tools are installed.
- Your system is set up for cluster configuration.

For instructions on installing cluster software and setting up your system for cluster configuration, see Installing cluster software.

Procedure

1. On any system, open a browser to the following URL, specifying one of the nodes of the cluster (note that this uses the **https** protocol). This brings up the **pcsd** Web UI login screen.

   ```
   https://nodename:2224
   ```

2. Log in as user **hacluster**. This brings up the **Clusters** page.

9.2. CONFIGURING A HIGH AVAILABILITY PCSD WEB UI

When you use the **pcsd** Web UI, you connect to one of the nodes of the cluster to display the cluster management pages. If the node to which you are connecting goes down or becomes unavailable, you can reconnect to the cluster by opening your browser to a URL that specifies a different node of the cluster. It is possible, however, to configure the **pcsd** Web UI itself for high availability, in which case you can continue to manage the cluster without entering a new URL.

Procedure

To configure the **pcsd** Web UI for high availability, perform the following steps.

1. Ensure that the **pcsd** certificates are synced across the nodes of the cluster by setting **PCSD_SSL_CERT_SYNC_ENABLED** to **true** in the **/etc/sysconfig/pcsd** configuration file. Enabling certificate syncing causes **pcsd** to sync the certificates for the cluster setup and node add commands. **PCSD_SSL_CERT_SYNC_ENABLED** is set to **false** by default.

2. Create an **IPaddr2** cluster resource, which is a floating IP address that you will use to connect to the **pcsd** Web UI. 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.

3. Create custom SSL certificates for use with **pcsd** and ensure that they are valid for the addresses of the nodes used to connect to the **pcsd** Web UI.

   a. To create custom SSL certificates, you can use either wildcard certificates or you can use the Subject Alternative Name certificate extension. For information about the Red Hat Certificate System, see the Red Hat Certificate System Administration Guide .

b. Install the custom certificates for **pcsd** with the **pcs pcsd certkey** command.

c. Sync the **pcsd** certificates to all nodes in the cluster with the **pcs pcsd sync-certificates** command.

4. Connect to the **pcsd** Web UI using the floating IP address you configured as a cluster resource.

NOTE

Even when you configure the **pcsd** Web UI for high availability, you will be asked to log in again when the node to which you are connecting goes down.

CHAPTER 10. 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 about 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.

10.1. DISPLAYING AVAILABLE FENCE AGENTS AND THEIR OPTIONS

The following commands can be used to view available fencing agents and the available options for specific fencing agents.

NOTE

Your system's hardware determines the type of fencing device to use for your cluster. For information about supported platforms and architectures and the different fencing devices, see the Cluster Platforms and Architectures section of the article Support Policies for RHEL High Availability Clusters.

Run the following command to list all available fencing agents. When you specify a filter, this command displays only the fencing agents that match the filter.

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

Run the following command to display the options for the specified fencing 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, with the exception of the **fence_sbd** agent a value of **cycle** is unsupported and should not be specified, as it may cause data corruption. Even for **fence_sbd**, however. you should not specify a method and instead use the default value.

10.2. CREATING A FENCE DEVICE

The format for the command to create a fence device is as follows. For a listing of the available fence device creation options, see the **pcs stonith -h** display.

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

The following command creates a single fencing device for a single node.

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

Some fence devices can fence only a single node, while other devices can fence multiple nodes. The parameters you specify when you create a fencing device depend on what your fencing device supports and requires.

- Some fence devices can automatically determine what nodes they can fence.
- You can use the **pcmk_host_list** parameter when creating a fencing device to specify all of the machines that are controlled by that fencing device.
- Some fence devices require a mapping of host names to the specifications that the fence device understands. You can map host names with the **pcmk_host_map** parameter when creating a fencing device.

For information about the **pcmk_host_list** and **pcmk_host_map** parameters, see General properties of fencing devices.

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

10.3. GENERAL PROPERTIES OF FENCING DEVICES

There are many general properties you can set for fencing devices, as well as various cluster properties that determine fencing behavior.

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.

The following table describes the general properties you can set for fencing devices.

Table 10.1. General Properties of Fencing Devices

Field	Type	Default	Description
pcmk_host_map	string		A mapping of host names to port 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. the **pcmk_host_map** property supports special characters inside **pcmk_host_map** values using a backslash in front of the value. For example, you can specify **pcmk_host_map="node3:plug\ 1"** to include a space in the host alias.
pcmk_host_list	string		A list of machines controlled by this device (Optional unless **pcmk_host_check=static-list**).

Field	Type	Default	Description
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)

The following table summarizes additional properties you can set for fencing devices. Note that these properties are for advanced use only.

Table 10.2. 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.
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.

Field	Type	Default	Description
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.
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.

Field	Type	Default	Description
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.
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.

Field	Type	Default	Description
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	string	0s	Enables a base delay for fencing actions and specifies a base delay value. You can specify different values for different nodes with the **pcmk_delay_base** parameter. For general information about fencing delay parameters and their interactions, see Fencing delays.
pcmk_delay_max	time	0s	Enables a random delay for fencing actions and specifies the maximum delay, which is the maximum value of the combined base delay and random delay. For example, if the base delay is 3 and **pcmk_delay_max** is 10, the random delay will be between 3 and 10. For general information about fencing delay parameters and their interactions, see Fencing delays.
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 (this is the default value). 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.

Field	Type	Default	Description
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.

In addition to the properties you can set for individual fence devices, there are also cluster properties you can set that determine fencing behavior, as described in the following table.

Table 10.3. Cluster Properties that Determine Fencing Behavior

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 fencing device. Allowed values: **reboot**, **off**. The value **poweroff** is also allowed, but is only used for legacy devices.
stonith-timeout	60s	How long to wait for a STONITH action to complete.
stonith-max-attempts	10	How many times fencing can fail for a target before the cluster will no longer immediately re-attempt it.
stonith-watchdog-timeout		The maximum time to wait until a node can be assumed to have been killed by the hardware watchdog. It is recommended that this value be set to twice the value of the hardware watchdog timeout. This option is needed only if watchdog-only SBD configuration is used for fencing.

Option	Default	Description
concurrent-fencing	true	Allow fencing operations to be performed in parallel.
fence-reaction	stop	Determines how a cluster node should react if notified of its own fencing. A cluster node may receive notification of its own fencing if fencing is misconfigured, or if fabric fencing is in use that does not cut cluster communication. Allowed values are **stop** to attempt to immediately stop Pacemaker and stay stopped, or **panic** to attempt to immediately reboot the local node, falling back to stop on failure. Although the default value for this property is **stop**, the safest choice for this value is **panic**, which attempts to immediately reboot the local node. If you prefer the stop behavior, as is most likely to be the case in conjunction with fabric fencing, it is recommended that you set this explicitly.
priority-fencing-delay	0 (disabled)	Sets a fencing delay that allows you to configure a two-node cluster so that in a split-brain situation the node with the fewest or least important resources running is the node that gets fenced. For general information about fencing delay parameters and their interactions, see Fencing delays.

For information about setting cluster properties, see Setting and removing cluster properties.

10.4. FENCING DELAYS

When cluster communication is lost in a two-node cluster, one node may detect this first and fence the other node. If both nodes detect this at the same time, however, each node may be able to initiate fencing of the other, leaving both nodes powered down or reset. By setting a fencing delay, you can decrease the likelihood of both cluster nodes fencing each other. You can set delays in a cluster with more than two nodes, but this is generally not of any benefit because only a partition with quorum will initiate fencing.

You can set different types of fencing delays, depending on your system requirements.

- **static fencing delays**
  A static fencing delay is a fixed, predetermined delay. Setting a static delay on one node makes that node more likely to be fenced because it increases the chances that the other node will initiate fencing first after detecting lost communication. In an active/passive cluster, setting a delay on a passive node makes it more likely that the passive node will be fenced when communication breaks down. You configure a static delay by using the **pcs_delay_base** cluster property. You can set this property when a separate fence device is used for each node or when a single fence device is used for all nodes.

- **dynamic fencing delays**
  A dynamic fencing delay is random. It can vary and is determined at the time fencing is needed. You configure a random delay and specify a maximum value for the combined base delay and random delay with the **pcs_delay_max** cluster property. When the fencing delay for each node is random, which node is fenced is also random. You may find this feature useful if your cluster is configured with a single fence device for all nodes in an active/active design.

- **priority fencing delays**
  A priority fencing delay is based on active resource priorities. If all resources have the same priority, the node with the fewest resources running is the node that gets fenced. In most cases, you use only one delay-related parameter, but it is possible to combine them. Combining delay-related parameters adds the priority values for the resources together to create a total delay. You configure a priority fencing delay with the **priority-fencing-delay** cluster property. You may find this feature useful in an active/active cluster design because it can make the node running the fewest resources more likely to be fenced when communication between the nodes is lost.

The **pcmk_delay_base** cluster property

Setting the **pcmk_delay_base** cluster property enables a base delay for fencing and specifies a base delay value.

When you set the **pcmk_delay_max** cluster property in addition to the **pcmk_delay_base** property, the overall delay is derived from a random delay value added to this static delay so that the sum is kept below the maximum delay. When 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**.

You can specify different values for different nodes with the **pcmk_delay_base** parameter. This allows a single fence device to be used in a two-node cluster, with a different delay for each node. You do not need to configure two separate devices to use separate delays. To specify different values for different nodes, you map the host names to the delay value for that node using a similar syntax to **pcmk_host_map**. For example, **node1:0;node2:10s** would use no delay when fencing **node1** and a 10-second delay when fencing **node2**.

The **pcmk_delay_max** cluster property

Setting the **pcmk_delay_max** cluster property enables a random delay for fencing actions and specifies the maximum delay, which is the maximum value of the combined base delay and random delay. For example, if the base delay is 3 and **pcmk_delay_max** is 10, the random delay will be between 3 and 10.

When you set the **pcmk_delay_base** cluster property in addition to the **pcmk_delay_max** property, the overall delay is derived from a random delay value added to this static delay so that the sum is kept below the maximum delay. When you set **pcmk_delay_max** but do not set **pcmk_delay_base** there is no static component to the delay.

The **priority-fencing-delay** cluster property

Setting the **priority-fencing-delay** cluster property allows you to configure a two-node cluster so that in a split-brain situation the node with the fewest or least important resources running is the node that gets fenced.

The **priority-fencing-delay** property can be set to a time duration. The default value for this property is 0 (disabled). If this property is set to a non-zero value, and the priority meta-attribute is configured for at least one resource, then in a split-brain situation the node with the highest combined priority of all resources running on it will be more likely to remain operational. For example, if you set **pcs resource defaults update priority=1** and **pcs property set priority-fencing-delay=15s** and no other priorities are set, then the node running the most resources will be more likely to remain operational because the other node will wait 15 seconds before initiating fencing. If a particular resource is more important than the rest, you can give it a higher priority.

The node running the promoted role of a promotable clone gets an extra 1 point if a priority has been configured for that clone.

Interaction of fencing delays

Setting more than one type of fencing delay yields the following results:

- Any delay set with the **priority-fencing-delay** property is added to any delay from the **pcmk_delay_base** and **pcmk_delay_max** fence device properties. This behavior allows some delay when both nodes have equal priority, or both nodes need to be fenced for some reason other than node loss, as when **on-fail=fencing** is set for a resource monitor operation. When setting these delays in combination, set the **priority-fencing-delay** property to a value that is significantly greater than the maximum delay from **pcmk_delay_base** and **pcmk_delay_max** to be sure the prioritized node is preferred. Setting this property to twice this value is always safe.

- Only fencing scheduled by Pacemaker itself observes fencing delays. Fencing scheduled by external code such as **dlm_controld** and fencing implemented by the **pcs stonith fence** command do not provide the necessary information to the fence device.

- Some individual fence agents implement a delay parameter, with a name determined by the agent and which is independent of delays configured with a **pcmk_delay_*** property. If both of these delays are configured, they are added together and would generally not be used in conjunction.

10.5. TESTING A FENCE DEVICE

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

Procedure

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

These examples 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 about 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
```

10.6. 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.

Pacemaker processes fencing levels as follows:

- 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
```

10.7. 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
```

10.8. 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 fencing device only. If the **--full** option is specified, all configured fencing options are displayed.

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

10.9. EXPORTING FENCE DEVICES AS PCS COMMANDS

As of Red Hat Enterprise Linux 9.1, you can display the **pcs** commands that can be used to re-create configured fence devices on a different system using the **--output-format=cmd** option of the **pcs stonith config** command.

The following commands create a **fence_apc_snmp** fence device and display the **pcs** command you can use to re-create the device.

```
# pcs stonith create myapc fence_apc_snmp ip="zapc.example.com"
pcmk_host_map="z1.example.com:1;z2.example.com:2" username="apc" password="apc"
# pcs stonith config --output-format=cmd
Warning: Only 'text' output format is supported for stonith levels
pcs stonith create --no-default-ops --force -- myapc fence_apc_snmp \
  ip=zapc.example.com password=apc 'pcmk_host_map=z1.example.com:1;z2.example.com:2'
username=apc \
  op \
    monitor interval=60s id=myapc-monitor-interval-60s
```

10.10. MODIFYING AND DELETING FENCE DEVICES

Modify or add options to a currently configured fencing device with the following command.

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

Updating a SCSI fencing device with the **pcs stonith update** command causes a restart of all resources running on the same node where the fencing resource was running. You can use either version of the following command to update SCSI devices without causing a restart of other cluster resources. As of RHEL 9.1, SCSI fencing devices can be configured as multipath devices.

```
pcs stonith update-scsi-devices stonith_id set device-path1 device-path2
pcs stonith update-scsi-devices stonith_id add device-path1 remove device-path2
```

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

```
pcs stonith delete stonith_id
```

10.11. 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 fence 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
```

10.12. DISABLING A FENCE DEVICE

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

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

```
# pcs stonith disable myapc
```

10.13. PREVENTING A NODE FROM USING A FENCING 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
```

10.14. 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 "Disabling ACPI Soft-Off with the Bios" below.

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 "Disabling ACPI Soft-Off in the logind.conf file", below. This is the first alternate method of disabling ACPI Soft-Off.

- Appending **acpi=off** to the kernel boot command line, as described in "Disabling ACPI completely in the GRUB 2 file", below. 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.

10.14.1. 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.

Procedure

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). The **BIOS CMOS Setup Utiliy** example below 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 about 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**.

10.14.2. 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.

Procedure

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

```
HandlePowerKey=ignore
```

2. Restart the **systemd-logind** service:

```
# systemctl restart systemd-logind.service
```

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

10.14.3. 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.

Procedure

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 about testing a fence device, see Testing a fence device.

CHAPTER 11. CONFIGURING CLUSTER RESOURCES

Create and delete cluster resources with the following commands.

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_id]
[clone_options] | promotable [clone_id] [clone_options] [--wait[=n]]
```

Key cluster resource creation options include the following:

- 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.

There is no limit to the number of resources you can create in a cluster.

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

Delete a configured resource with the following command.

```
pcs resource delete resource_id
```

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

```
# pcs resource delete VirtualIP
```

11.1. 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.

The following table describes these properties of a resource agent.

Table 11.1. 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.

The following table summarizes the commands that display the available resource properties.

Table 11.2. 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.

11.2. 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.

...
```

11.3. 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.

The following table describes the resource meta options.

Table 11.3. 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**	Indicates what state the cluster should 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 if appropriate) * **Promoted** - Allow the resource to be started and, if appropriate, promoted * **Unpromoted** - Allow the resource to be started, but only in unpromoted mode if the resource is promotable
is-managed	**true**	Indicates whether the cluster is allowed to start and stop the resource. Allowed values: **true**, **false**
resource-stickiness	1	Value to indicate how much the resource prefers to stay where it is.

Field	Default	Description
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)	Ignore previously failed resource actions after this much time has passed without new failures. This potentially allows the resource to move back to the node on which it failed, if it previously reached its migration threshold there. A value of 0 indicates that failures do not expire. *WARNING*: If this value is low, and pending cluster activity prevents the cluster from responding to a failure within that time, the failure is ignored completely and does not cause recovery of the resource, even if a recurring action continues to report failure. The value of this option should be at least greater than the longest action timeout for all resources in the cluster. A value in hours or days is reasonable.
multiple-active	**stop_start**	Indicates what the cluster should 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 * **stop_unexpected** - (RHEL 9.1 and later) stop only unexpected instances of the resource, without requiring a full restart. It is the user’s responsibility to verify that the service and its resource agent can function with extra active instances without requiring a full restart.

Field	Default	Description
critical	**true**	Sets the default value for the **influence** option for all colocation constraints involving the resource as a dependent resource (*target_resource*), including implicit colocation constraints created when the resource is part of a resource group. The **influence** colocation constraint option determines whether the cluster will move both the primary and dependent resources to another node when the dependent resource reaches its migration threshold for failure, or whether the cluster will leave the dependent resource offline without causing a service switch. The **critical** resource meta option can have a value of **true** or **false**, with a default value of **true**.
allow-unhealthy-nodes	**false**	(RHEL 9.1 and later) When set to **true**, the resource is not forced off a node due to degraded node health. When health resources have this attribute set, the cluster can automatically detect if the node's health recovers and move resources back to it. A node's health is determined by a combination of the health attributes set by health resource agents based on local conditions, and the strategy-related options that determine how the cluster reacts to those conditions.

11.3.1. Changing the default value of a resource option

You can change the default value of a resource option for all resources with the **pcs resource defaults update** command. The following command resets the default value of **resource-stickiness** to 100.

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

The original **pcs resource defaults *name=value*** command, which set defaults for all resources in previous releases, remains supported unless there is more than one set of defaults configured. However, **pcs resource defaults update** is now the preferred version of the command.

11.3.2. Changing the default value of a resource option for sets of resources

You can create multiple sets of resource defaults with the **pcs resource defaults set create** command, which allows you to specify a rule that contains **resource** expressions. Only **resource** and **date** expressions, including **and**, **or** and parentheses, are allowed in rules that you specify with this command.

With the **pcs resource defaults set create** command, you can configure a default resource value for all resources of a particular type. If, for example, you are running databases which take a long time to stop, you can increase the **resource-stickiness** default value for all resources of the database type to prevent those resources from moving to other nodes more often than you desire.

The following command sets the default value of **resource-stickiness** to 100 for all resources of type **pqsql**.

- The **id** option, which names the set of resource defaults, is not mandatory. If you do not set this option **pcs** will generate an ID automatically. Setting this value allows you to provide a more descriptive name.
- In this example, **::pgsql** means a resource of any class, any provider, of type **pgsql**.
    - Specifying **ocf:heartbeat:pgsql** would indicate class **ocf**, provider **heartbeat**, type **pgsql**,
    - Specifying **ocf:pacemaker:** would indicate all resources of class **ocf**, provider **pacemaker**, of any type.

```
# pcs resource defaults set create id=pgsql-stickiness meta resource-stickiness=100 rule
resource ::pgsql
```

To change the default values in an existing set, use the **pcs resource defaults set update** command.

11.3.3. Displaying currently configured resource defaults

The **pcs resource defaults** command displays a list of currently configured default values for resource options, including any rules you specified.

The following example shows the output of this command after you have reset the default value of **resource-stickiness** to 100.

```
# pcs resource defaults
Meta Attrs: rsc_defaults-meta_attributes
  resource-stickiness=100
```

The following example shows the output of this command after you have reset the default value of **resource-stickiness** to 100 for all resources of type **pqsql** and set the **id** option to **id=pgsql-stickiness**.

```
# pcs resource defaults
Meta Attrs: pgsql-stickiness
  resource-stickiness=100
  Rule: boolean-op=and score=INFINITY
    Expression: resource ::pgsql
```

11.3.4. 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, or cloned 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
  ...
```

11.4. 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.

11.4.1. 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.

11.4.2. 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...
```

11.4.3. Displaying resource groups

The following command lists all currently configured resource groups.

```
pcs resource group list
```

11.4.4. 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 about resource meta options, see Configuring resource meta options.

11.4.5. 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.

11.5. 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 about 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 about 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 about 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.

CHAPTER 12. 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.

In addition to location constraints, the node on which a resource runs is influenced by the **resource-stickiness** value for that resource, which determines to what degree a resource prefers to remain on the node where it is currently running. For information about setting the **resource-stickiness** value, see Configuring a resource to prefer its current node .

12.1. 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]] ...
```

The following table summarizes the meanings of the basic options for configuring location constraints.

Table 12.1. Location Constraint Options

Field	Description
rsc	A resource name
node	A node’s name

Field	Description
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 **pcs constraint location *rsc* prefers** command 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** for **score** in a **pcs constraint location *rsc* avoids** command indicates that the resource will never run on that node, even if no other node is available. This is the equivalent of setting a **pcs constraint location add** command with a score of **-INFINITY**. 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 constraints 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
```

12.2. 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]
```

The following table summarizes the meanings of the basic parameters for configuring constraints for resource discovery.

Table 12.2. 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** for **score** 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).

12.3. 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.

12.3.1. 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
```

12.3.2. 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 "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.

12.4. CONFIGURING A RESOURCE TO PREFER ITS CURRENT NODE

Resources have a **resource-stickiness** value that you can set as a meta attribute when you create the resource, as described in Configuring resource meta options. The **resource-stickiness** value determines how much a resource wants to remain on the node where it is currently running. Pacemaker considers the **resource-stickiness** value in conjunction with other settings (for example, the score values of location constraints) to determine whether to move a resource to another node or to leave it in place.

With a **resource-stickiness** value of 0, a cluster may move resources as needed to balance resources across nodes. This may result in resources moving when unrelated resources start or stop. With a positive stickiness, resources have a preference to stay where they are, and move only if other circumstances outweigh the stickiness. This may result in newly-added nodes not getting any resources assigned to them without administrator intervention.

Newly-created clusters in RHEL 9 set the default value for **resource-stickiness** to 1. This small value can easily be overridden by other constraints that you create, but it is enough to prevent Pacemaker from needlessly moving healthy resources around the cluster. If you prefer cluster behavior that results from a **resource-stickiness** value of 0, you can change the **resource-stickiness** default value to 0 with the following command:

```
# pcs resource defaults update resource-stickiness=0
```

If you upgrade an existing cluster to RHEL 9 and you have not explicitly set a default value for **resource-stickiness**, the **resource-stickiness** value remains 0 and the **pcs resource defaults** command will not show anything for stickiness.

With a positive **resource-stickiness** value, no resources will move to a newly-added node. If resource balancing is desired at that point, you can temporarily set the **resource-stickiness** value to 0.

Note that if a location constraint score is higher than the **resource-stickiness** value, the cluster may still move a healthy resource to the node where the location constraint points.

For further information about how Pacemaker determines where to place a resource, see Configuring a node placement strategy.

12.5. EXPORTING RESOURCE CONSTRAINTS AS PCS COMMANDS

As of Red Hat Enterprise Linux 9.3, you can display the **pcs** commands that can be used to re-create configured resource constraints on a different system using the **--output-format=cmd** option of the **pcs constraint** command.

The following commands create an **IPaddr2** resource and an **apache** resource.

```
# pcs resource create VirtualIP IPaddr2 ip=198.51.100.3 cidr_netmask=24
Assumed agent name 'ocf:heartbeat:IPaddr2' (deduced from 'IPaddr2')
# pcs resource create Website apache configfile="/etc/httpd/conf/httpd.conf"
statusurl="http://127.0.0.1/server-status"
Assumed agent name 'ocf:heartbeat:apache' (deduced from 'apache')
```

The following commands configure a location constraint, a colocation constraint, and an order constraint for the two resources.

```
# pcs constraint location Website avoids node1
# pcs constraint colocation add Website with VirtualIP
# pcs constraint order VirtualIP then Website
Adding VirtualIP Website (kind: Mandatory) (Options: first-action=start then-action=start)
```

After you create the resources and the constraints, the following command displays the **pcs** commands you can use to re-create the constraints on a different system.

```
# pcs constraint --output-format=cmd
pcs -- constraint location add location-Website-node1--INFINITY resource%Website node1 -
INFINITY;
pcs -- constraint colocation add Website with VirtualIP INFINITY \
  id=colocation-Website-VirtualIP-INFINITY;
pcs -- constraint order start VirtualIP then start Website \
  id=order-VirtualIP-Website-mandatory
```

CHAPTER 13. 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]
```

The following table summarizes the properties and options for configuring ordering constraints.

Table 13.1. Properties of an Order Constraint

Field	Description
resource_id	The name of a resource on which an action is performed.
action	The action to be ordered on the resource. Possible values of the *action* property are as follows: * **start** - Order start actions of the resource. * **stop** - Order stop actions of the resource. * **promote** - Promote the resource from an unpromoted resource to a promoted resource. * **demote** - Demote the resource from a promoted resource to an unpromoted 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 about optional ordering, see Configuring advisory ordering. * **Mandatory** - Always enforce the constraint (default value). If the first resource you specified is stopping or cannot be started, the second resource you specified must be stopped. For information about 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 A stops). Ordering constraints for which **kind** is **Serialize** cannot be symmetrical. The default value is**true** for **Mandatory** and **Optional** kinds, **false** for **Serialize**.

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

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

13.1. 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.

13.2. 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
```

13.3. 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.

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 the "Properties of an Order Constraint" table in Determining the order in which cluster resources are run .
- **role**, which can be set to **Stopped**, **Started**, **Promoted**, or **Unpromoted**.

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 the "Properties of an Order Constraint" table in Determining the order in which cluster resources are run .
- **symmetrical**, to set whether the reverse of the constraint applies for the opposite action, as described in in the "Properties of an Order Constraint" table in Determining the order in which cluster resources are run.

```
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
```

13.4. CONFIGURING STARTUP ORDER FOR RESOURCE DEPENDENCIES NOT MANAGED BY PACEMAKER

It is possible for a cluster to include resources with dependencies that are not themselves managed by the cluster. In this case, you must ensure that those dependencies are started before Pacemaker is started and stopped after Pacemaker is stopped.

You can configure your startup order to account for this situation by means of the **systemd resource-agents-deps** target. You can create a **systemd** drop-in unit for this target and Pacemaker will order itself appropriately relative to this target.

For example, if a cluster includes a resource that depends on the external service **foo** that is not managed by the cluster, perform the following procedure.

1. Create the drop-in unit **/etc/systemd/system/resource-agents-deps.target.d/foo.conf** that contains the following:

   ```
   [Unit]
   Requires=foo.service
   After=foo.service
   ```

2. Run the **systemctl daemon-reload** command.

A cluster dependency specified in this way can be something other than a service. For example, you may have a dependency on mounting a file system at **/srv**, in which case you would perform the following procedure:

1. Ensure that **/srv** is listed in the **/etc/fstab** file. This will be converted automatically to the **systemd** file **srv.mount** at boot when the configuration of the system manager is reloaded. For more information, see the **systemd.mount**(5) and the **systemd-fstab-generator**(8) man pages.

2. To make sure that Pacemaker starts after the disk is mounted, create the drop-in unit **/etc/systemd/system/resource-agents-deps.target.d/srv.conf** that contains the following.

   ```
   [Unit]
   Requires=srv.mount
   After=srv.mount
   ```

3. Run the **systemctl daemon-reload** command.

If an LVM volume group used by a Pacemaker cluster contains one or more physical volumes that reside on remote block storage, such as an iSCSI target, you can configure a **systemd resource-agents-deps** target and a **systemd** drop-in unit for the target to ensure that the service starts before Pacemaker starts.

The following procedure configures **blk-availability.service** as a dependency. The **blk-availability.service** service is a wrapper that includes **iscsi.service**, among other services. If your deployment requires it, you could configure **iscsi.service** (for iSCSI only) or **remote-fs.target** as the dependency instead of **blk-availability**.

1. Create the drop-in unit **/etc/systemd/system/resource-agents-deps.target.d/blk-availability.conf** that contains the following:

   ```
   [Unit]
   Requires=blk-availability.service
   After=blk-availability.service
   ```

2. Run the **systemctl daemon-reload** command.

CHAPTER 14. 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 [promoted|unpromoted] source_resource with [promoted|unpromoted]
target_resource [score] [options]
```

The following table summarizes the properties and options for configuring colocation constraints.

Table 14.1. Parameters of a Colocation Constraint

Parameter	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*.

Parameter	Description
influence option	Determines whether the cluster will move both the primary resource (*source_resource*) and dependent resources (*target_resource)* to another node when the dependent resource reaches its migration threshold for failure, or whether the cluster will leave the dependent resource offline without causing a service switch. The **influence** colocation constraint option can have a value of **true** or **false**. The default value for this option is determined by the value of the dependent resource's **critical** resource meta option, which has a default value of **true**. When this option has a value of **true**, Pacemaker will attempt to keep both the primary and dependent resource active. If the dependent resource reaches its migration threshold for failures, both resources will move to another node if possible. When this option has a value of **false**, Pacemaker will avoid moving the primary resource as a result of the status of the dependent resource. In this case, if the dependent resource reaches its migration threshold for failures, it will stop if the primary resource is active and can remain on its current node.

14.1. 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.

14.2. SPECIFYING ADVISORY PLACEMENT OF RESOURCES

Advisory placement of resources indicates the placement of resources is a preference, but is not mandatory. 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.

14.3. 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. 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**, **Promoted**, or **Unpromoted**.

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 about this option, see the "Location Constraint Options" table in Configuring Location Constraints

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]]
```

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

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

CHAPTER 15. DISPLAYING RESOURCE CONSTRAINTS AND RESOURCE DEPENDENCIES

There are a several commands you can use to display constraints that have been configured. You can display all configured resource constraints, or you can limit the display of esource constraints to specific types of resource constraints. Additionally, you can display configured resource dependencies.

Displaying all configured constraints

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

```
pcs constraint [list|show] [--full]
```

By default, listing resource constraints does not display expired constraints. To include expired constaints in the listing, use the **--all** option of the **pcs constraint** command. This will list expired constraints, noting the constraints and their associated rules as **(expired)** in the display.

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]
```

Displaying ordering constraints

The following command lists all current ordering constraints.

```
pcs constraint order [show]
```

Displaying colocation constraints

The following command lists all current colocation constraints.

```
pcs constraint colocation [show]
```

Displaying resource-specific constraints

The following command lists the constraints that reference specific resources.

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

Displaying resource dependencies

The following command displays the relations between cluster resources in a tree structure.

```
pcs resource relations resource [--full]
```

-

If the **--full** option is used, the command displays additional information, including the constraint IDs and the resource types.

In the following example, there are 3 configured resources: C, D, and E.

```
# pcs constraint order start C then start D
Adding C D (kind: Mandatory) (Options: first-action=start then-action=start)
# pcs constraint order start D then start E
Adding D E (kind: Mandatory) (Options: first-action=start then-action=start)

# pcs resource relations C
C
`- order
   |  start C then start D
   `- D
      `- order
         |  start D then start E
         `- E
# pcs resource relations D
D
|- order
|  |  start C then start D
|  `- C
`- order
   |  start D then start E
   `- E
# pcs resource relations E
E
`- order
   |  start D then start E
   `- D
      `- order
         |  start C then start D
         `- C
```

In the following example, there are 2 configured resources: A and B. Resources A and B are part of resource group G.

```
# pcs resource relations A
A
`- outer resource
   `- G
      `- inner resource(s)
         |  members: A B
         `- B
# pcs resource relations B
B
`- outer resource
   `- G
      `- inner resource(s)
         |  members: A B
         `- A
# pcs resource relations G
G
`- inner resource(s)
```

```
|  members: A B
|- A
`- B
```

CHAPTER 16. DETERMINING RESOURCE LOCATION WITH RULES

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

16.1. PACEMAKER RULES

Pacemaker 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 16.1. Properties of a Rule

Field	Description
role	Limits the rule to apply only when the resource is in that role. Allowed values: **Started**, **Unpromoted,** and **Promoted**. NOTE: A rule with **role="Promoted"** 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**.

16.1.1. Node attribute expressions

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

Table 16.2. 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**, **number**, **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 the following table.

Table 16.3. 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.

16.1.2. 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 16.4. 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

16.1.3. 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 16.5. 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).

16.2. 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 about 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=promoted|unpromoted]
[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, and moon as described in the "Properties of a Date Specification" table in Date specifications.

- **defined|not_defined *attribute***
- ***attribute* lt|gt|lte|gte|eq|ne [string|integer|number|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 17. MANAGING CLUSTER RESOURCES

There are a variety of commands you can use to display, modify, and administer cluster resources.

17.1. 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 status** command yields the following output.

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

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
```

To display the status of an individual resource, use the following command.

```
pcs resource status resource_id
```

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

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

To display the status of the resources running on a specific node, use the following command. You can use this command to display the status of resources on both cluster and remote nodes.

```
pcs resource status node=node_id
```

For example, if **node-01** is running resources named **VirtualIP** and **WebSite** the **pcs resource status node=node-01** command might yield the following output.

```
# pcs resource status node=node-01
 VirtualIP      (ocf::heartbeat:IPaddr2):       Started
 WebSite        (ocf::heartbeat:apache):        Started
```

17.2. EXPORTING CLUSTER RESOURCES AS PCS COMMANDS

As of Red Hat Enterprise Linux 9.1, you can display the **pcs** commands that can be used to re-create configured cluster resources on a different system using the **--output-format=cmd** option of the **pcs resource config** command.

The following commands create four resources created for an active/passive Apache HTTP server in a Red Hat high availability cluster: an **LVM-activate** resource, a **Filesystem** resource, an **IPaddr2** resource, and an **Apache** resource.

```
# pcs resource create my_lvm ocf:heartbeat:LVM-activate vgname=my_vg
vg_access_mode=system_id --group apachegroup
# pcs resource create my_fs Filesystem device="/dev/my_vg/my_lv" directory="/var/www"
fstype="xfs" --group apachegroup
# pcs resource create VirtualIP IPaddr2 ip=198.51.100.3 cidr_netmask=24 --group apachegroup
# pcs resource create Website apache configfile="/etc/httpd/conf/httpd.conf"
statusurl="http://127.0.0.1/server-status" --group apachegroup
```

After you create the resources, the following command displays the **pcs** commands you can use to re-create those resources on a different system.

```
# pcs resource config --output-format=cmd
pcs resource create --no-default-ops --force -- my_lvm ocf:heartbeat:LVM-activate \
  vg_access_mode=system_id vgname=my_vg \
  op \
    monitor interval=30s id=my_lvm-monitor-interval-30s timeout=90s \
    start interval=0s id=my_lvm-start-interval-0s timeout=90s \
    stop interval=0s id=my_lvm-stop-interval-0s timeout=90s;
pcs resource create --no-default-ops --force -- my_fs ocf:heartbeat:Filesystem \
  device=/dev/my_vg/my_lv directory=/var/www fstype=xfs \
  op \
    monitor interval=20s id=my_fs-monitor-interval-20s timeout=40s \
    start interval=0s id=my_fs-start-interval-0s timeout=60s \
    stop interval=0s id=my_fs-stop-interval-0s timeout=60s;
pcs resource create --no-default-ops --force -- VirtualIP ocf:heartbeat:IPaddr2 \
  cidr_netmask=24 ip=198.51.100.3 \
  op \
    monitor interval=10s id=VirtualIP-monitor-interval-10s timeout=20s \
    start interval=0s id=VirtualIP-start-interval-0s timeout=20s \
    stop interval=0s id=VirtualIP-stop-interval-0s timeout=20s;
pcs resource create --no-default-ops --force -- Website ocf:heartbeat:apache \
  configfile=/etc/httpd/conf/httpd.conf statusurl=http://127.0.0.1/server-status \
  op \
    monitor interval=10s id=Website-monitor-interval-10s timeout=20s \
    start interval=0s id=Website-start-interval-0s timeout=40s \
    stop interval=0s id=Website-stop-interval-0s timeout=60s;
pcs resource group add apachegroup \
  my_lvm my_fs VirtualIP Website
```

To display the **pcs** command or commands you can use to re-create only one configured resource, specify the resource ID for that resource.

```
# pcs resource config VirtualIP --output-format=cmd
pcs resource create --no-default-ops --force -- VirtualIP ocf:heartbeat:IPaddr2 \
  cidr_netmask=24 ip=198.51.100.3 \
op \
  monitor interval=10s id=VirtualIP-monitor-interval-10s timeout=20s \
  start interval=0s id=VirtualIP-start-interval-0s timeout=20s \
  stop interval=0s id=VirtualIP-stop-interval-0s timeout=20s
```

17.3. 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
```

NOTE

When you update a resource's operation with the **pcs resource update** command, any options you do not specifically call out are reset to their default values.

17.4. CLEARING FAILURE STATUS OF CLUSTER RESOURCES

If a resource has failed, a failure message appears when you display the cluster status with the **pcs status** command. After attempting to resolve the cause of the failure, you can check the updated status of the resource by running the **pcs status** command again, and you can check the failure count for the cluster resources with the **pcs resource failcount show --full** command.

You can clear that failure status of a resource with the **pcs resource cleanup** command. The **pcs resource cleanup** command resets the resource status and **failcount** value for the resource. This command also removes the operation history for the resource and re-detects its current state.

The following command resets the resource status and **failcount** value for the resource specified by *resource_id*.

```
pcs resource cleanup resource_id
```

If you do not specify *resource_id*, the **pcs resource cleanup** command resets the resource status and **failcount** value for all resources with a failure count.

In addition to the **pcs resource cleanup *resource_id*** command, you can also reset the resource status and clear the operation history of a resource with the **pcs resource refresh *resource_id*** command. As with the **pcs resource cleanup** command, you can run the **pcs resource refresh** command with no options specified to reset the resource status and **failcount** value for all resources.

Both the **pcs resource cleanup** and the **pcs resource refresh** commands clear the operation history for a resource and re-detect the current state of the resource. The **pcs resource cleanup** command operates only on resources with failed actions as shown in the cluster status, while the **pcs resource refresh** command operates on resources regardless of their current state.

17.5. 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 Disabling, enabling, 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.

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

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 update 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 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 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.

17.5.2. 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.

The following table describes the properties you can set for a **ping** resource.

Table 17.1. Properties of a ping resources

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 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 **gateway.example.com**.

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

17.6. 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.

When you update a resource's operation with the **pcs resource update** command, any options you do not specifically call out are reset to their default values. For example, if you have configured a monitoring operation with a custom timeout value of 600, running the following commands will reset the timeout value to the default value of 20 (or whatever you have set the default value to with the **pcs resource op defaults** command).

```
# pcs resource update resourceXZY op monitor enabled=false
# pcs resource update resourceXZY op monitor enabled=true
```

In order to maintain the original value of 600 for this option, when you reinstate the monitoring operation you must specify that value, as in the following example.

```
# pcs resource update resourceXZY op monitor timeout=600 enabled=true
```

17.7. CONFIGURING AND MANAGING CLUSTER RESOURCE TAGS

You can use the **pcs** command to tag cluster resources. This allows you to enable, disable, manage, or unmanage a specified set of resources with a single command.

17.7.1. Tagging cluster resources for administration by category

The following procedure tags two resources with a resource tag and disables the tagged resources. In this example, the existing resources to be tagged are named **d-01** and **d-02**.

Procedure

1. Create a tag named **special-resources** for resources **d-01** and **d-02**.

```
[root@node-01]# pcs tag create special-resources d-01 d-02
```

2. Display the resource tag configuration.

```
[root@node-01]# pcs tag config
special-resources
  d-01
  d-02
```

3. Disable all resources that are tagged with the **special-resources** tag.

```
[root@node-01]# pcs resource disable special-resources
```

4. Display the status of the resources to confirm that resources **d-01** and **d-02** are disabled.

```
[root@node-01]# pcs resource
  * d-01        (ocf::pacemaker:Dummy): Stopped (disabled)
  * d-02        (ocf::pacemaker:Dummy): Stopped (disabled)
```

In addition to the **pcs resource disable** command, the **pcs resource enable**, **pcs resource manage**, and **pcs resource unmanage** commands support the administration of tagged resources.

After you have created a resource tag:

- You can delete a resource tag with the **pcs tag delete** command.
- You can modify resource tag configuration for an existing resource tag with the **pcs tag update** command.

17.7.2. Deleting a tagged cluster resource

You cannot delete a tagged cluster resource with the **pcs** command. To delete a tagged resource, use the following procedure.

Procedure

1. Remove the resource tag.

   a. The following command removes the resource tag **special-resources** from all resources with that tag,

   ```
   [root@node-01]# pcs tag remove special-resources
   [root@node-01]# pcs tag
    No tags defined
   ```

   b. The following command removes the resource tag **special-resources** from the resource **d-01** only.

   ```
   [root@node-01]# pcs tag update special-resources remove d-01
   ```

2. Delete the resource.

   ```
   [root@node-01]# pcs resource delete d-01
   Attempting to stop: d-01... Stopped
   ```

CHAPTER 18. 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.

18.1. CREATING AND REMOVING A CLONED RESOURCE

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

To create a resource and clone of the resource with the following single command.

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

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

By default, the name of the clone will be ***resource_id*-clone**. You can set a custom name for the clone by specifying a value for the *clone_id* option.

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_id [clone_id][clone options]...
```

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

By default, the name of the clone will be ***resource_id*-clone** or ***group_name*-clone**. You can set a custom name for the clone by specifying a value for the *clone_id* option.

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, by default the clone takes on the name of the resource with **-clone** appended to the name. The following command 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 | clone_id | group_name
```

The following table describes the options you can specify for a cloned resource.

Table 18.1. Resource Clone Options

Field	Description
priority, target-role, is-managed	Options inherited from resource that is being cloned, as described in the "Resource Meta Options" table in Configuring 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**.

Field	Description
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**.
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 about setting the **resource-stickiness** resource meta-option, see Configuring resource meta options.

18.2. 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, because the **interleave** clone option is left to default as **false**, no instance of **webfarm-stats** will start until all instances of **webfarm-clone** that need to be started have done so. 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
```

18.3. 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 **promoted** and **unpromoted**. 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 **Unpromoted** state. Note: The Promoted and Unpromoted role names are the functional equivalent of the Master and Slave Pacemaker roles in previous RHEL releases.

18.3.1. Creating a promotable clone 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_id]
[clone options]
```

By default, the name of the promotable clone will be ***resource_id*-clone**.

You can set a custom name for the clone by specifying a value for the *clone_id* option.

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

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

By default, the name of the promotable clone will be ***resource_id*-clone** or ***group_name*-clone**.

You can set a custom name for the clone by specifying a value for the *clone_id* option.

The following table describes the extra clone options you can specify for a promotable resource.

Table 18.2. 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.

18.3.2. Configuring promotable resource constraints

In most cases, a promotable resource 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 promoted or unpromoted role. The following command creates a resource colocation constraint.

```
pcs constraint colocation add [promoted|unpromoted] source_resource with [promoted|unpromoted]
target_resource [score] [options]
```

For information about 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 unpromoted role to promoted role. Additionally, you can specify an action of **demote**, indicated that the resource be demoted from promoted role to unpromoted role.

The command for configuring an order constraint is as follows.

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

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

18.4. DEMOTING A PROMOTED RESOURCE ON FAILURE

You can configure a promotable resource so that when a **promote** or **monitor** action fails for that resource, or the partition in which the resource is running loses quorum, the resource will be demoted but will not be fully stopped. This can prevent the need for manual intervention in situations where fully stopping the resource would require it.

- To configure a promotable resource to be demoted when a **promote** action fails, set the **on-fail** operation meta option to **demote**, as in the following example.

  ```
  # pcs resource op add my-rsc promote on-fail="demote"
  ```

- To configure a promotable resource to be demoted when a **monitor** action fails, set **interval** to a nonzero value, set the **on-fail** operation meta option to **demote**, and set **role** to **Promoted**, as in the following example.

  ```
  # pcs resource op add my-rsc monitor interval="10s" on-fail="demote"
  role="Promoted"
  ```

- To configure a cluster so that when a cluster partition loses quorum any promoted resources will be demoted but left running and all other resources will be stopped, set the **no-quorum-policy** cluster property to **demote**

Setting the **on-fail** meta-attribute to **demote** for an operation does not affect how promotion of a resource is determined. If the affected node still has the highest promotion score, it will be selected to be promoted again.

CHAPTER 19. MANAGING CLUSTER NODES

There are a variety of **pcs** commands you can use to manage cluster nodes, including commands to start and stop cluster services and to add and remove cluster nodes.

19.1. 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
```

19.2. ENABLING AND DISABLING CLUSTER SERVICES

Enable the cluster services with the following command. This 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] [...]
```

19.3. ADDING CLUSTER NODES

Add a new node to an existing cluster with the following procedure.

This procedure adds standard clusters nodes running **corosync**. For information about integrating non-corosync nodes into a cluster, see Integrating non-corosync nodes into a cluster: the pacemaker_remote service.

NOTE

It is 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.

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**.

Procedure

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 ~]# dnf 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.

19.4. 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
```

19.5. ADDING A NODE TO A CLUSTER WITH MULTIPLE LINKS

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 for the first link and 192.168.123.203 as the second link.

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

19.6. ADDING AND MODIFYING LINKS IN AN EXISTING CLUSTER

In most cases, you can add or modify the links in an existing cluster without restarting the cluster.

19.6.1. Adding and removing links in an existing cluster

To add a new link to a running cluster, use the **pcs cluster link add** command.

- When adding a link, you must specify an address for each node.
- Adding and removing a link is only possible when you are using the **knet** transport protocol.
- At least one link in the cluster must be defined at any time.
- The maximum number of links in a cluster is 8, numbered 0-7. It does not matter which links are defined, so, for example, you can define only links 3, 6 and 7.
- When you add a link without specifying its link number, **pcs** uses the lowest link available.
- The link numbers of currently configured links are contained in the **corosync.conf** file. To display the **corosync.conf** file, run the **pcs cluster corosync** command or the **pcs cluster config show** command.

The following command adds link number 5 to a three node cluster.

```
[root@node1 ~] # pcs cluster link add node1=10.0.5.11 node2=10.0.5.12 node3=10.0.5.31
options linknumber=5
```

To remove an existing link, use the **pcs cluster link delete** or **pcs cluster link remove** command. Either of the following commands will remove link number 5 from the cluster.

```
[root@node1 ~] # pcs cluster link delete 5

[root@node1 ~] # pcs cluster link remove 5
```

19.6.2. Modifying a link in a cluster with multiple links

If there are multiple links in the cluster and you want to change one of them, perform the following procedure.

Procedure

1. Remove the link you want to change.

   ```
   [root@node1 ~] # pcs cluster link remove 2
   ```

2. Add the link back to the cluster with the updated addresses and options.

   ```
   [root@node1 ~] # pcs cluster link add node1=10.0.5.11 node2=10.0.5.12
   node3=10.0.5.31 options linknumber=2
   ```

19.6.3. Modifying the link addresses in a cluster with a single link

If your cluster uses only one link and you want to modify that link to use different addresses, perform the following procedure. In this example, the original link is link 1.

1. Add a new link with the new addresses and options.

   ```
   [root@node1 ~] # pcs cluster link add node1=10.0.5.11 node2=10.0.5.12
   node3=10.0.5.31 options linknumber=2
   ```

2. Remove the original link.

   ```
   [root@node1 ~] # pcs cluster link remove 1
   ```

Note that you cannot specify addresses that are currently in use when adding links to a cluster. This means, for example, that if you have a two-node cluster with one link and you want to change the address for one node only, you cannot use the above procedure to add a new link that specifies one new address and one existing address. Instead, you can add a temporary link before removing the existing link and adding it back with the updated address, as in the following example.

In this example:

- The link for the existing cluster is link 1, which uses the address 10.0.5.11 for node 1 and the address 10.0.5.12 for node 2.

- You would like to change the address for node 2 to 10.0.5.31.

Procedure

To update only one of the addresses for a two-node cluster with a single link, use the following procedure.

1. Add a new temporary link to the existing cluster, using addresses that are not currently in use.

   ```
   [root@node1 ~] # pcs cluster link add node1=10.0.5.13 node2=10.0.5.14 options
   linknumber=2
   ```

2. Remove the original link.

   ```
   [root@node1 ~] # pcs cluster link remove 1
   ```

3. Add the new, modified link.

   ```
   [root@node1 ~] # pcs cluster link add node1=10.0.5.11 node2=10.0.5.31 options
   linknumber=1
   ```

4. Remove the temporary link you created

   ```
   [root@node1 ~] # pcs cluster link remove 2
   ```

19.6.4. Modifying the link options for a link in a cluster with a single link

If your cluster uses only one link and you want to modify the options for that link but you do not want to change the address to use, you can add a temporary link before removing and updating the link to modify.

In this example:

- The link for the existing cluster is link 1, which uses the address 10.0.5.11 for node 1 and the address 10.0.5.12 for node 2.
- You would like to change the link option **link_priority** to 11.

Procedure

Modify the link option in a cluster with a single link with the following procedure.

1. Add a new temporary link to the existing cluster, using addresses that are not currently in use.

   ```
   [root@node1 ~] # pcs cluster link add node1=10.0.5.13 node2=10.0.5.14 options
   linknumber=2
   ```

2. Remove the original link.

   ```
   [root@node1 ~] # pcs cluster link remove 1
   ```

3. Add back the original link with the updated options.

```
[root@node1 ~] # pcs cluster link add node1=10.0.5.11 node2=10.0.5.12 options
linknumber=1 link_priority=11
```

4. Remove the temporary link.

```
[root@node1 ~] # pcs cluster link remove 2
```

19.6.5. Modifying a link when adding a new link is not possible

If for some reason adding a new link is not possible in your configuration and your only option is to modify a single existing link, you can use the following procedure, which requires that you shut your cluster down.

Procedure

The following example procedure updates link number 1 in the cluster and sets the **link_priority** option for the link to 11.

1. Stop the cluster services for the cluster.

```
[root@node1 ~] # pcs cluster stop --all
```

2. Update the link addresses and options.
   The **pcs cluster link update** command does not require that you specify all of the node addresses and options. Instead, you can specify only the addresses to change. This example modifies the addresses for **node1** and **node3** and the **link_priority** option only.

```
[root@node1 ~] # pcs cluster link update 1 node1=10.0.5.11 node3=10.0.5.31 options
link_priority=11
```

   To remove an option, you can set the option to a null value with the ***option*=** format.

3. Restart the cluster

```
[root@node1 ~] # pcs cluster start --all
```

19.7. CONFIGURING A NODE HEALTH STRATEGY

A node might be functioning well enough to maintain its cluster membership and yet be unhealthy in some respect that makes it an undesirable location for resources. For example, a disk drive might be reporting SMART errors, or the CPU might be highly loaded. As of RHEL 9.1, You can use a node health strategy in Pacemaker to automatically move resources off unhealthy nodes.

You can monitor a node’s health with the the following health node resource agents, which set node attributes based on CPU and disk status:

- **ocf:pacemaker:HealthCPU**, which monitors CPU idling
- **ocf:pacemaker:HealthIOWait**, which monitors the CPU I/O wait
- **ocf:pacemaker:HealthSMART**, which monitors SMART status of a disk drive
- **ocf:pacemaker:SysInfo**, which sets a variety of node attributes with local system information and also functions as a health agent monitoring disk space usage

Additionally, any resource agent might provide node attributes that can be used to define a health node strategy.

Procedure

The following procedure configures a health node strategy for a cluster that will move resources off of any node whose CPU I/O wait goes above 15%.

1. Set the **health-node-strategy** cluster property to define how Pacemaker responds to changes in node health.

   ```
   # pcs property set node-health-strategy=migrate-on-red
   ```

2. Create a cloned cluster resource that uses a health node resource agent, setting the **allow-unhealthy-nodes** resource meta option to define whether the cluster will detect if the node's health recovers and move resources back to the node. Configure this resource with a recurring monitor action, to continually check the health of all nodes.
   This example creates a **HealthIOWait** resource agent to monitor the CPU I/O wait, setting a red limit for moving resources off a node to 15%. This command sets the **allow-unhealthy-nodes** resource meta option to **true** and configures a recurring monitor interval of 10 seconds.

   ```
   # pcs resource create io-monitor ocf:pacemaker:HealthIOWait red_limit=15 op monitor
   interval=10s meta allow-unhealthy-nodes=true clone
   ```

19.8. CONFIGURING A LARGE CLUSTER WITH MANY RESOURCES

If the cluster you are deploying consists of a large number of nodes and many resources, you may need to modify the default values of the following parameters for your cluster.

The cluster-ipc-limit cluster property

The **cluster-ipc-limit** cluster property is the maximum IPC message backlog before one cluster daemon will disconnect another. When a large number of resources are cleaned up or otherwise modified simultaneously in a large cluster, a large number of CIB updates arrive at once. This could cause slower clients to be evicted if the Pacemaker service does not have time to process all of the configuration updates before the CIB event queue threshold is reached.
The recommended value of **cluster-ipc-limit** for use in large clusters is the number of resources in the cluster multiplied by the number of nodes. This value can be raised if you see "Evicting client" messages for cluster daemon PIDs in the logs.

You can increase the value of **cluster-ipc-limit** from its default value of 500 with the **pcs property set** command. For example, for a ten-node cluster with 200 resources you can set the value of **cluster-ipc-limit** to 2000 with the following command.

```
# pcs property set cluster-ipc-limit=2000
```

The PCMK_ipc_buffer Pacemaker parameter

On very large deployments, internal Pacemaker messages may exceed the size of the message buffer. When this occurs, you will see a message in the system logs of the following format:

```
Compressed message exceeds X% of configured IPC limit (X bytes); consider setting
PCMK_ipc_buffer to X or higher
```

When you see this message, you can increase the value of **PCMK_ipc_buffer** in the

/etc/sysconfig/pacemaker configuration file on each node. For example, to increase the value of **PCMK_ipc_buffer** from its default value to 13396332 bytes, change the uncommented **PCMK_ipc_buffer** field in the **/etc/sysconfig/pacemaker** file on each node in the cluster as follows.

```
PCMK_ipc_buffer=13396332
```

To apply this change, run the following comand.

```
# systemctl restart pacemaker
```

CHAPTER 20. SETTING USER PERMISSIONS FOR A PACEMAKER CLUSTER

You can grant permission for specific users other than user **hacluster** to manage a Pacemaker cluster. There are two sets of permissions that you can grant to individual users:

- Permissions that allow individual users to manage the cluster through the Web UI and to run **pcs** commands that connect to nodes over a network. Commands that connect to nodes over a network include commands to set up a cluster, or to add or remove nodes from a cluster.
- Permissions for local users to allow read-only or read-write access to the cluster configuration. Commands that do not require connecting over a network include commands that edit the cluster configuration, such as those that create resources and configure constraints.

In situations where both sets of permissions have been assigned, the permissions for commands that connect over a network are applied first, and then permissions for editing the cluster configuration on the local node are applied. Most **pcs** commands do not require network access and in those cases the network permissions will not apply.

20.1. SETTING PERMISSIONS FOR NODE ACCESS OVER A NETWORK

To grant permission for specific users to manage the cluster through the Web UI and to run **pcs** commands that connect to nodes over a network, add those users to the group **haclient**. This must be done on every node in the cluster.

20.2. SETTING LOCAL PERMISSIONS USING ACLS

You can use the **pcs acl** command to set permissions for local users to allow read-only or read-write access to the cluster configuration by using access control lists (ACLs).

By default, ACLs are not enabled. When ACLs are not enabled, any user who is a member of the group **haclient** on all nodes has full local read/write access to the cluster configuration while users who are not members of **haclient** have no access. When ACLs are enabled, however, even users who are members of the **haclient** group have access only to what has been granted to that user by the ACLs. The root and **hacluster** user accounts always have full access to the cluster configuration, even when ACLs are enabled.

Setting permissions for local users is a two step process:

1. Execute the **pcs acl role create...** command to create a *role* which defines the permissions for that role.
2. Assign the role you created to a user with the **pcs acl user create** command. If you assign multiple roles to the same user, any **deny** permission takes precedence, then **write**, then **read**.

Procedure

The following example procedure provides read-only access for a cluster configuration to a local user named **rouser**. Note that it is also possible to restrict access to certain portions of the configuration only.

WARNING

It is important to perform this procedure as root or to save all of the configuration updates to a working file which you can then push to the active CIB when you are finished. Otherwise, you can lock yourself out of making any further changes. For information on saving configuration updates to a working file, see Saving a configuration change to a working file.

1. This procedure requires that the user **rouser** exists on the local system and that the user **rouser** is a member of the group **haclient**.

```
# adduser rouser
# usermod -a -G haclient rouser
```

2. Enable Pacemaker ACLs with the **pcs acl enable** command.

```
# pcs acl enable
```

3. Create a role named **read-only** with read-only permissions for the cib.

```
# pcs acl role create read-only description="Read access to cluster" read xpath /cib
```

4. Create the user **rouser** in the pcs ACL system and assign that user the **read-only** role.

```
# pcs acl user create rouser read-only
```

5. View the current ACLs.

```
# pcs acl
User: rouser
  Roles: read-only
Role: read-only
  Description: Read access to cluster
  Permission: read xpath /cib (read-only-read)
```

6. On each node where **rouser** will run **pcs** commands, log in as **rouser** and authenticate to the local **pcsd** service. This is required in order to run certain **pcs** commands, such as **pcs status**, as the ACL user.

```
[rouser ~]$ pcs client local-auth
```

CHAPTER 21. RESOURCE MONITORING OPERATIONS

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, by default the **pcs** command will create a monitoring operation, with an interval that is determined by the resource agent. If the resource agent does not provide a default monitoring interval, the pcs command will create a monitoring operation with an interval of 60 seconds.

The following table summarizes the properties of a resource monitoring operation.

Table 21.1. Properties of an Operation

Field	Description
id	Unique name for the action. The system assigns this when you configure an operation.
name	The action to perform. Common values: **monitor**, **start**, **stop**
interval	If set to a nonzero value, a recurring operation is created that repeats at this frequency, in seconds. A nonzero value makes sense only when the action **name** is set to **monitor**. A recurring monitor action will be executed immediately after a resource start completes, and subsequent monitor actions are scheduled starting at the time the previous monitor action completed. For example, if a monitor action with **interval=20s** is executed at 01:00:00, the next monitor action does not occur at 01:00:20, but at 20 seconds after the first monitor action completes. If set to zero, which is the default value, this parameter allows you to provide values to be used for operations created by the cluster. For example, if the **interval** is set to zero, the **name** of the operation is set to**start**, and the **timeout** value is set to 40, then Pacemaker will use a timeout of 40 seconds when starting this resource. A **monitor** operation with a zero interval allows you to set the **timeout**/**on-fail**/**enabled** values for the probes that Pacemaker does at startup to get the current status of all resources when the defaults are not desirable.
timeout	If the operation does not complete in the amount of time set by this parameter, abort the operation and consider it failed. The default value is the value of **timeout** if set with the**pcs resource op defaults** command, or 20 seconds if it is not set. If you find that your system includes a resource that requires more time than the system allows to perform an operation (such as **start**, **stop**, or **monitor**), investigate the cause and if the lengthy execution time is expected you can increase this value. The **timeout** value is not a delay of any kind, nor does the cluster wait the entire timeout period if the operation returns before the timeout period has completed.

Field	Description
on-fail	The action to take if this action ever fails. Allowed values: * **ignore** - Pretend the resource did not fail * **block** - Do not perform any further operations on the resource * **stop** - Stop the resource and do not start it elsewhere * **restart** - Stop the resource and start it again (possibly on a different node) * **fence** - STONITH the node on which the resource failed * **standby** - Move *all* resources away from the node on which the resource failed * **demote** - When a **promote** action fails for the resource, the resource will be demoted but will not be fully stopped. When a **monitor** action fails for a resource, if **interval** is set to a nonzero value and **role** is set to **Promoted** the resource will be demoted but will not be fully stopped. The default for the **stop** operation is **fence** when STONITH is enabled and **block** otherwise. All other operations default to **restart**.
enabled	If **false**, the operation is treated as if it does not exist. Allowed values **true**, **false**

21.1. CONFIGURING RESOURCE MONITORING OPERATIONS

You can configure monitoring operations when you create a resource with the following command.

```
pcs resource create resource_id standard:provider:type|type [resource_options] [op operation_action
operation_options [operation_type operation_options]...]
```

For example, the following command creates an **IPaddr2** resource with a monitoring operation. The new resource is called **VirtualIP** with an IP address of 192.168.0.99 and a netmask of 24 on **eth2**. A monitoring operation will be performed every 30 seconds.

```
# pcs resource create VirtualIP ocf:heartbeat:IPaddr2 ip=192.168.0.99 cidr_netmask=24
nic=eth2 op monitor interval=30s
```

Alternately, you can add a monitoring operation to an existing resource with the following command.

```
pcs resource op add resource_id operation_action [operation_properties]
```

Use the following command to delete a configured resource operation.

```
pcs resource op remove resource_id operation_name operation_properties
```

NOTE

You must specify the exact operation properties to properly remove an existing operation.

To change the values of a monitoring option, you can update the resource. For example, you can create a **VirtualIP** with the following command.

```
# pcs resource create VirtualIP ocf:heartbeat:IPaddr2 ip=192.168.0.99 cidr_netmask=24
nic=eth2
```

By default, this command creates these operations.

```
Operations: start interval=0s timeout=20s (VirtualIP-start-timeout-20s)
            stop interval=0s timeout=20s (VirtualIP-stop-timeout-20s)
            monitor interval=10s timeout=20s (VirtualIP-monitor-interval-10s)
```

To change the stop timeout operation, execute the following command.

```
# pcs resource update VirtualIP op stop interval=0s timeout=40s

# pcs resource config VirtualIP
 Resource: VirtualIP (class=ocf provider=heartbeat type=IPaddr2)
  Attributes: ip=192.168.0.99 cidr_netmask=24 nic=eth2
  Operations: start interval=0s timeout=20s (VirtualIP-start-timeout-20s)
              monitor interval=10s timeout=20s (VirtualIP-monitor-interval-10s)
              stop interval=0s timeout=40s (VirtualIP-name-stop-interval-0s-timeout-40s)
```

21.2. CONFIGURING GLOBAL RESOURCE OPERATION DEFAULTS

You can change the default value of a resource operation for all resources with the **pcs resource op defaults update** command.

The following command sets a global default of a **timeout** value of 240 seconds for all monitoring operations.

```
# pcs resource op defaults update timeout=240s
```

The original **pcs resource op defaults *name*=*value*** command, which set resource operation defaults for all resources in previous releases, remains supported unless there is more than one set of defaults configured. However, **pcs resource op defaults update** is now the preferred version of the command.

21.2.1. Overriding resource-specific operation values

Note that a cluster resource will use the global default only when the option is not specified in the cluster resource definition. By default, resource agents define the **timeout** option for all operations. For the global operation timeout value to be honored, you must create the cluster resource without the **timeout** option explicitly or you must remove the **timeout** option by updating the cluster resource, as in the following command.

```
# pcs resource update VirtualIP op monitor interval=10s
```

For example, after setting a global default of a **timeout** value of 240 seconds for all monitoring operations and updating the cluster resource **VirtualIP** to remove the timeout value for the **monitor** operation, the resource **VirtualIP** will then have timeout values for **start**, **stop**, and **monitor** operations of 20s, 40s and 240s, respectively. The global default value for timeout operations is applied here only on the **monitor** operation, where the default **timeout** option was removed by the previous command.

```
# pcs resource config VirtualIP
 Resource: VirtualIP (class=ocf provider=heartbeat type=IPaddr2)
  Attributes: ip=192.168.0.99 cidr_netmask=24 nic=eth2
  Operations: start interval=0s timeout=20s (VirtualIP-start-timeout-20s)
              monitor interval=10s (VirtualIP-monitor-interval-10s)
              stop interval=0s timeout=40s (VirtualIP-name-stop-interval-0s-timeout-40s)
```

21.2.2. Changing the default value of a resource operation for sets of resources

You can create multiple sets of resource operation defaults with the **pcs resource op defaults set create** command, which allows you to specify a rule that contains **resource** and operation expressions. All of the rule expressions supported by Pacemaker are allowed.

With this comand, you can configure a default resource operation value for all resources of a particular type. For example, it is now possible to configure implicit **podman** resources created by Pacemaker when bundles are in use.

The following command sets a default timeout value of 90s for all operations for all **podman** resources. In this example, **::podman** means a resource of any class, any provider, of type **podman**.

The **id** option, which names the set of resource operation defaults, is not mandatory. If you do not set this option, **pcs** will generate an ID automatically. Setting this value allows you to provide a more descriptive name.

```
# pcs resource op defaults set create id=podman-timeout meta timeout=90s rule resource
::podman
```

The following command sets a default timeout value of 120s for the **stop** operation for all resources.

```
# pcs resource op defaults set create id=stop-timeout meta timeout=120s rule op stop
```

It is possible to set the default timeout value for a specific operation for all resources of a particular type. The following example sets a default timeout value of 120s for the **stop** operation for all **podman** resources.

```
# pcs resource op defaults set create id=podman-stop-timeout meta timeout=120s rule
resource ::podman and op stop
```

21.2.3. Displaying currently configured resource operation default values

The **pcs resource op defaults** command displays a list of currently configured default values for resource operations, including any rules you specified.

The following command displays the default operation values for a cluster which has been configured with a default timeout value of 90s for all operations for all **podman** resources, and for which an ID for the set of resource operation defaults has been set as **podman-timeout**.

```
# pcs resource op defaults
Meta Attrs: podman-timeout
  timeout=90s
  Rule: boolean-op=and score=INFINITY
    Expression: resource ::podman
```

The following command displays the default operation values for a cluster which has been configured with a default timeout value of 120s for the **stop** operation for all **podman** resources, and for which an ID for the set of resource operation defaults has been set as **podman-stop-timeout**.

```
# pcs resource op defaults]
Meta Attrs: podman-stop-timeout
  timeout=120s
  Rule: boolean-op=and score=INFINITY
    Expression: resource ::podman
    Expression: op stop
```

21.3. CONFIGURING MULTIPLE MONITORING OPERATIONS

You can configure a single resource with as many monitor operations as a resource agent supports. In this way you can do a superficial health check every minute and progressively more intense ones at higher intervals.

NOTE

When configuring multiple monitor operations, you must ensure that no two operations are performed at the same interval.

To configure additional monitoring operations for a resource that supports more in-depth checks at different levels, you add an **OCF_CHECK_LEVEL=*n*** option.

For example, if you configure the following **IPaddr2** resource, by default this creates a monitoring operation with an interval of 10 seconds and a timeout value of 20 seconds.

```
# pcs resource create VirtualIP ocf:heartbeat:IPaddr2 ip=192.168.0.99 cidr_netmask=24
nic=eth2
```

If the Virtual IP supports a different check with a depth of 10, the following command causes Pacemaker to perform the more advanced monitoring check every 60 seconds in addition to the normal Virtual IP check every 10 seconds. (As noted, you should not configure the additional monitoring operation with a 10-second interval as well.)

```
# pcs resource op add VirtualIP monitor interval=60s OCF_CHECK_LEVEL=10
```

CHAPTER 22. PACEMAKER CLUSTER PROPERTIES

Cluster properties control how the cluster behaves when confronted with situations that might occur during cluster operation.

22.1. SUMMARY OF CLUSTER PROPERTIES AND OPTIONS

The following table summaries the Pacemaker cluster properties, showing the default values of the properties and the possible values you can set for those properties.

There are additional cluster properties that determine fencing behavior. For information about these properties, see the table of cluster properties that determine fencing behavior in General properties of fencing devices.

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 22.1. 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.

Option	Default	Description
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 * demote - if a cluster partition loses quorum, demote any promoted resources and stop all other resources
symmetric-cluster	true	Indicates whether resources can run on any node by default.
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.
dc-deadtime	20s	How long to wait for a response from other nodes during startup. The "correct" value will depend on the speed and load of your network and the type of switches used.
stop-orphan-resources	true	Indicates whether deleted resources should be stopped.
stop-orphan-actions	true	Indicates whether deleted actions should be canceled.

Option	Default	Description
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 about 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.
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.

Option	Default	Description
cluster-recheck-interval	15 minutes	Pacemaker is primarily event-driven, and looks ahead to know when to recheck the cluster for failure timeouts and most time-based rules. Pacemaker will also recheck the cluster after the duration of inactivity specified by this property. This cluster recheck has two purposes: rules with **date-spec** are guaranteed to be checked this often, and it serves as a fail-safe for some kinds of scheduler bugs. A value of 0 disables this polling; positive values indicate a time interval.
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.
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.
placement-strategy	default	Indicates whether and how the cluster will take utilization attributes into account when determining resource placement on cluster nodes.

Option	Default	Description
node-health-strategy	none	When used in conjunction with a health resource agent, controls how Pacemaker responds to changes in node health. Allowed values: * **none** - Do not track node health. * **migrate-on-red** - Resources are moved off any node where a health agent has determined that the node's status is **red**, based on the local conditions that the agent monitors. * **only-green** - Resources are moved off any node where a health agent has determined that the node's status is **yellow** or **red**, based on the local conditions that the agent monitors. * **progressive**, **custom** - Advanced node health strategies that offer finer-grained control over the cluster's response to health conditions according to the internal numeric values of health attributes.

22.2. 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=
```

22.3. 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
```

22.4. EXPORTING CLUSTER PROPERTIES AS PCS COMMANDS

As of Red Hat Enterprise Linux 9.3, you can display the **pcs** commands that can be used to re-create configured cluster properties on a different system using the **--output-format=cmd** option of the **pcs property config** command.

The following command sets the **migration-limit** cluster property to 10.

```
# pcs property set migration-limit=10
```

After you set the cluster property, the following command displays the **pcs** command you can use to set the cluster property on a different system.

```
# pcs property config --output-format=cmd
pcs property set --force -- \
 migration-limit=10 \
 placement-strategy=minimal
```

CHAPTER 23. CONFIGURING RESOURCES TO REMAIN STOPPED ON CLEAN NODE SHUTDOWN

When a cluster node shuts down, Pacemaker's default response is to stop all resources running on that node and recover them elsewhere, even if the shutdown is a clean shutdown. You can configure Pacemaker so that when a node shuts down cleanly, the resources attached to the node will be locked to the node and unable to start elsewhere until they start again when the node that has shut down rejoins the cluster. This allows you to power down nodes during maintenance windows when service outages are acceptable without causing that node's resources to fail over to other nodes in the cluster.

23.1. CLUSTER PROPERTIES TO CONFIGURE RESOURCES TO REMAIN STOPPED ON CLEAN NODE SHUTDOWN

The ability to prevent resources from failing over on a clean node shutdown is implemented by means of the following cluster properties.

shutdown-lock

When this cluster property is set to the default value of **false**, the cluster will recover resources that are active on nodes being cleanly shut down. When this property is set to **true**, resources that are active on the nodes being cleanly shut down are unable to start elsewhere until they start on the node again after it rejoins the cluster.
The **shutdown-lock** property will work for either cluster nodes or remote nodes, but not guest nodes.

If **shutdown-lock** is set to **true**, you can remove the lock on one cluster resource when a node is down so that the resource can start elsewhere by performing a manual refresh on the node with the following command.

pcs resource refresh *resource* node=*nodename*

Note that once the resources are unlocked, the cluster is free to move the resources elsewhere. You can control the likelihood of this occurring by using stickiness values or location preferences for the resource.

NOTE

A manual refresh will work with remote nodes only if you first run the following commands:

1. Run the **systemctl stop pacemaker_remote** command on the remote node to stop the node.

2. Run the **pcs resource disable *remote-connection-resource*** command.

You can then perform a manual refresh on the remote node.

shutdown-lock-limit

When this cluster property is set to a time other than the default value of 0, resources will be available for recovery on other nodes if the node does not rejoin within the specified time since the shutdown was initiated.

NOTE

The **shutdown-lock-limit** property will work with remote nodes only if you first run the following commands:

1. Run the **systemctl stop pacemaker_remote** command on the remote node to stop the node.
2. Run the **pcs resource disable *remote-connection-resource*** command.

After you run these commands, the resources that had been running on the remote node will be available for recovery on other nodes when the amount of time specified as the **shutdown-lock-limit** has passed.

23.2. SETTING THE SHUTDOWN-LOCK CLUSTER PROPERTY

The following example sets the **shutdown-lock** cluster property to **true** in an example cluster and shows the effect this has when the node is shut down and started again. This example cluster consists of three nodes: **z1.example.com**, **z2.example.com**, and **z3.example.com**.

Procedure

1. Set the **shutdown-lock** property to to **true** and verify its value. In this example the **shutdown-lock-limit** property maintains its default value of 0.

```
[root@z3 ~]# pcs property set shutdown-lock=true
[root@z3 ~]# pcs property list --all | grep shutdown-lock
 shutdown-lock: true
 shutdown-lock-limit: 0
```

2. Check the status of the cluster. In this example, resources **third** and **fifth** are running on **z1.example.com**.

```
[root@z3 ~]# pcs status
...
Full List of Resources:
...
 * first (ocf::pacemaker:Dummy): Started z3.example.com
 * second (ocf::pacemaker:Dummy): Started z2.example.com
 * third (ocf::pacemaker:Dummy): Started z1.example.com
 * fourth (ocf::pacemaker:Dummy): Started z2.example.com
 * fifth (ocf::pacemaker:Dummy): Started z1.example.com
...
```

3. Shut down **z1.example.com**, which will stop the resources that are running on that node.

```
[root@z3 ~] # pcs cluster stop z1.example.com
Stopping Cluster (pacemaker)...
Stopping Cluster (corosync)...
```

4. Running the **pcs status** command shows that node **z1.example.com** is offline and that the resources that had been running on **z1.example.com** are **LOCKED** while the node is down.

```
[root@z3 ~]# pcs status
...

Node List:
 * Online: [ z2.example.com z3.example.com ]
 * OFFLINE: [ z1.example.com ]

Full List of Resources:
...
 * first (ocf::pacemaker:Dummy): Started z3.example.com
 * second (ocf::pacemaker:Dummy): Started z2.example.com
 * third (ocf::pacemaker:Dummy): Stopped z1.example.com (LOCKED)
 * fourth (ocf::pacemaker:Dummy): Started z3.example.com
 * fifth (ocf::pacemaker:Dummy): Stopped z1.example.com (LOCKED)

...
```

5. Start cluster services again on **z1.example.com** so that it rejoins the cluster. Locked resources should get started on that node, although once they start they will not not necessarily remain on the same node.

```
[root@z3 ~]# pcs cluster start z1.example.com
Starting Cluster...
```

6. In this example, resouces **third** and **fifth** are recovered on node **z1.example.com**.

```
[root@z3 ~]# pcs status
...

Node List:
 * Online: [ z1.example.com z2.example.com z3.example.com ]

Full List of Resources:
..
 * first (ocf::pacemaker:Dummy): Started z3.example.com
 * second (ocf::pacemaker:Dummy): Started z2.example.com
 * third (ocf::pacemaker:Dummy): Started z1.example.com
 * fourth (ocf::pacemaker:Dummy): Started z3.example.com
 * fifth (ocf::pacemaker:Dummy): Started z1.example.com

...
```

CHAPTER 24. CONFIGURING A NODE PLACEMENT STRATEGY

Pacemaker decides where to place a resource according to the resource allocation scores on every node. The resource will be allocated to the node where the resource has the highest score. This allocation score is derived from a combination of factors, including resource constraints, **resource-stickiness** settings, prior failure history of a resource on each node, and utilization of each node.

If the resource allocation scores on all the nodes are equal, by the default placement strategy Pacemaker will choose a node with the least number of allocated resources for balancing the load. If the number of resources on each node is equal, the first eligible node listed in the CIB will be chosen to run the resource.

Often, however, different resources use significantly different proportions of a node's capacities (such as memory or I/O). You cannot always balance the load ideally by taking into account only the number of resources allocated to a node. In addition, if resources are placed such that their combined requirements exceed the provided capacity, they may fail to start completely or they may run with degraded performance. To take these factors into account, Pacemaker allows you to configure the following components:

- the capacity a particular node provides
- the capacity a particular resource requires
- an overall strategy for placement of resources

24.1. UTILIZATION ATTRIBUTES AND PLACEMENT STRATEGY

To configure the capacity that a node provides or a resource requires, you can use *utilization attributes* for nodes and resources. You do this by setting a utilization variable for a resource and assigning a value to that variable to indicate what the resource requires, and then setting that same utilization variable for a node and assigning a value to that variable to indicate what that node provides.

You can name utilization attributes according to your preferences and define as many name and value pairs as your configuration needs. The values of utilization attributes must be integers.

24.1.1. Configuring node and resource capacity

The following example configures a utilization attribute of CPU capacity for two nodes, setting this attribute as the variable **cpu**. It also configures a utilization attribute of RAM capacity, setting this attribute as the variable **memory**. In this example:

- Node 1 is defined as providing a CPU capacity of two and a RAM capacity of 2048
- Node 2 is defined as providing a CPU capacity of four and a RAM capacity of 2048

```
# pcs node utilization node1 cpu=2 memory=2048
# pcs node utilization node2 cpu=4 memory=2048
```

The following example specifies the same utilization attributes that three different resources require. In this example:

- resource **dummy-small** requires a CPU capacity of 1 and a RAM capacity of 1024
- resource **dummy-medium** requires a CPU capacity of 2 and a RAM capacity of 2048

- resource **dummy-large** requires a CPU capacity of 1 and a RAM capacity of 3072

```
# pcs resource utilization dummy-small cpu=1 memory=1024
# pcs resource utilization dummy-medium cpu=2 memory=2048
# pcs resource utilization dummy-large cpu=3 memory=3072
```

A node is considered eligible for a resource if it has sufficient free capacity to satisfy the resource’s requirements, as defined by the utilization attributes.

24.1.2. Configuring placement strategy

After you have configured the capacities your nodes provide and the capacities your resources require, you need to set the **placement-strategy** cluster property, otherwise the capacity configurations have no effect.

Four values are available for the **placement-strategy** cluster property:

- **default** — Utilization values are not taken into account at all. Resources are allocated according to allocation scores. If scores are equal, resources are evenly distributed across nodes.
- **utilization** — Utilization values are taken into account only when deciding whether a node is considered eligible (that is, whether it has sufficient free capacity to satisfy the resource’s requirements). Load-balancing is still done based on the number of resources allocated to a node.
- **balanced** — Utilization values are taken into account when deciding whether a node is eligible to serve a resource and when load-balancing, so an attempt is made to spread the resources in a way that optimizes resource performance.
- **minimal** — Utilization values are taken into account only when deciding whether a node is eligible to serve a resource. For load-balancing, an attempt is made to concentrate the resources on as few nodes as possible, thereby enabling possible power savings on the remaining nodes.

The following example command sets the value of **placement-strategy** to **balanced**. After running this command, Pacemaker will ensure the load from your resources will be distributed evenly throughout the cluster, without the need for complicated sets of colocation constraints.

```
# pcs property set placement-strategy=balanced
```

24.2. PACEMAKER RESOURCE ALLOCATION

Pacemaker allocates resources according to node preference, node capacity, and resource allocation preference.

24.2.1. Node preference

Pacemaker determines which node is preferred when allocating resources according to the following strategy.

- The node with the highest node weight gets consumed first. Node weight is a score maintained by the cluster to represent node health.
- If multiple nodes have the same node weight:

- If the **placement-strategy** cluster property is **default** or **utilization**:
  - The node that has the least number of allocated resources gets consumed first.
  - If the numbers of allocated resources are equal, the first eligible node listed in the CIB gets consumed first.
- If the **placement-strategy** cluster property is **balanced**:
  - The node that has the most free capacity gets consumed first.
  - If the free capacities of the nodes are equal, the node that has the least number of allocated resources gets consumed first.
  - If the free capacities of the nodes are equal and the number of allocated resources is equal, the first eligible node listed in the CIB gets consumed first.
- If the **placement-strategy** cluster property is **minimal**, the first eligible node listed in the CIB gets consumed first.

24.2.2. Node capacity

Pacemaker determines which node has the most free capacity according to the following strategy.

- If only one type of utilization attribute has been defined, free capacity is a simple numeric comparison.
- If multiple types of utilization attributes have been defined, then the node that is numerically highest in the most attribute types has the most free capacity. For example:
  - If NodeA has more free CPUs, and NodeB has more free memory, then their free capacities are equal.
  - If NodeA has more free CPUs, while NodeB has more free memory and storage, then NodeB has more free capacity.

24.2.3. Resource allocation preference

Pacemaker determines which resource is allocated first according to the following strategy.

- The resource that has the highest priority gets allocated first. You can set a resource’s priority when you create the resource.
- If the priorities of the resources are equal, the resource that has the highest score on the node where it is running gets allocated first, to prevent resource shuffling.
- If the resource scores on the nodes where the resources are running are equal or the resources are not running, the resource that has the highest score on the preferred node gets allocated first. If the resource scores on the preferred node are equal in this case, the first runnable resource listed in the CIB gets allocated first.

24.3. RESOURCE PLACEMENT STRATEGY GUIDELINES

To ensure that Pacemaker’s placement strategy for resources works most effectively, you should take the following considerations into account when configuring your system.

- Make sure that you have sufficient physical capacity.
  If the physical capacity of your nodes is being used to near maximum under normal conditions, then problems could occur during failover. Even without the utilization feature, you may start to experience timeouts and secondary failures.

- Build some buffer into the capabilities you configure for the nodes.
  Advertise slightly more node resources than you physically have, on the assumption the that a Pacemaker resource will not use 100% of the configured amount of CPU, memory, and so forth all the time. This practice is sometimes called overcommit.

- Specify resource priorities.
  If the cluster is going to sacrifice services, it should be the ones you care about least. Ensure that resource priorities are properly set so that your most important resources are scheduled first.

24.4. THE NODEUTILIZATION RESOURCE AGENT

The **NodeUtilization** resoure agent can detect the system parameters of available CPU, host memory availability, and hypervisor memory availability and add these parameters into the CIB. You can run the agent as a clone resource to have it automatically populate these parameters on each node.

For information about the **NodeUtilization** resource agent and the resource options for this agent, run the **pcs resource describe NodeUtilization** command.

CHAPTER 25. 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.

25.1. VIRTUAL DOMAIN RESOURCE OPTIONS

The following table describes the resource options you can configure for a **VirtualDomain** resource.

Table 25.1. 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.

25.2. CREATING THE VIRTUAL DOMAIN RESOURCE

The following procedure creates a **VirtualDomain** resource in a cluster for a virtual machine you have previously created.

Procedure

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 one of the cluster nodes that will be allowed to run the guest. You can use a file name of your choosing; this example uses **/etc/pacemaker/guest1.xml**.

   ```
   # virsh dumpxml guest1 > /etc/pacemaker/guest1.xml
   ```

2. Copy the virtual machine's **xml** configuration file to all of the other cluster nodes that will be allowed to run the guest, in the same location on each node.

3. Ensure that all of the nodes allowed to run the virtual domain have access to the necessary storage devices for that virtual domain.

4. Separately test that the virtual domain can start and stop on each node that will run the virtual domain.

5. If it is running, shut down the guest node. Pacemaker will start the node when it is configured in the cluster. The virtual machine should not be configured to start automatically when the host boots.

6. Configure the **VirtualDomain** 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 resource move VM *nodeX*** command would be done as a live migration.

In this example **migration_transport** is set to **ssh**. Note that for SSH migration to work properly, keyless logging must work between nodes.

```
# pcs resource create VM VirtualDomain config=/etc/pacemaker/guest1.xml
migration_transport=ssh meta allow-migrate=true
```

CHAPTER 26. CONFIGURING 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 about the configuration and operation of the **votequorum** service, see the **votequorum**(5) man page.

26.1. 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. The following table summarizes these options.

Table 26.1. 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. 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.

26.2. 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
```

26.3. DISPLAYING QUORUM CONFIGURATION AND STATUS

Once a cluster is running, you can enter the following cluster quorum commands to display the quorum configuration and status.

The following command shows the quorum configuration.

```
pcs quorum [config]
```

The following command shows the quorum runtime status.

```
pcs quorum status
```

26.4. 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
```

CHAPTER 27. CONFIGURING 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. With two-node clusters, the use of a quorum device can better determine which node survives in a split-brain situation.

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.

27.1. 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:~]# dnf install corosync-qdevice
[root@node2:~]# dnf install corosync-qdevice
```

- Install **pcs** and **corosync-qnetd** on the quorum device host.

```
[root@qdevice:~]# dnf 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
```

27.2. CONFIGURING A QUORUM DEVICE

Configure a quorum device and add it to the cluster with the following procedure.

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 about the implementation of these algorithms, see the **corosync-qdevice**(8) man page.

- The cluster nodes are **node1** and **node2**.

Procedure

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, and the **Qdevice** membership status for each node is **NR** (Not Registered).

```
[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 about 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. The meanings of of the **Qdevice** membership information status values for each cluster node are as follows:

   - **A/NA** — The quorum device is alive or not alive, indicating whether there is a heartbeat between **qdevice** and **corosync**. This should always indicate that the quorum device is alive.
   - **V/NV** — **V** is set when the quorum device has given a vote to a node. In this example, both nodes are set to **V** since they can communicate with each other. If the cluster were to split into two single-node clusters, one of the nodes would be set to **V** and the other node would be set to **NV**.
   - **MW/NMW** — The internal quorum device flag is set (**MW**) or not set (**NMW**). By default the flag is not set and the value is **NMW**.

     ```
     [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)
```

27.3. 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
```

27.4. MANAGING A QUORUM DEVICE IN A CLUSTER

There are a variety of **pcs** commands that you can use to change the quorum device settings in a cluster, disable a quorum device, and remove a quorum device.

27.4.1. 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
```

27.4.2. Removing a quorum device

The following command removes 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
```

27.4.3. Destroying a quorum device

The following command disables and stops a quorum device on the quorum device host and deletes all of its configuration files.

```
[root@qdevice:~]# pcs qdevice destroy net
Stopping quorum device...
quorum device stopped
quorum device disabled
Quorum device 'net' configuration files removed
```

CHAPTER 28. TRIGGERING SCRIPTS FOR CLUSTER EVENTS

A Pacemaker cluster is an event-driven system, where an event might be a resource or node failure, a configuration change, or a resource starting or stopping. You can configure Pacemaker cluster alerts to take some external action when a cluster event occurs by means of alert agents, which are external programs that the cluster calls in the same manner as the cluster calls resource agents to handle resource configuration and operation.

The cluster passes information about the event to the agent by means of environment variables. Agents can do anything with this information, such as send an email message or log to a file or update a monitoring system.

- Pacemaker provides several sample alert agents, which are installed in **/usr/share/pacemaker/alerts** by default. These sample scripts may be copied and used as is, or they may be used as templates to be edited to suit your purposes. Refer to the source code of the sample agents for the full set of attributes they support.

- If the sample alert agents do not meet your needs, you can write your own alert agents for a Pacemaker alert to call.

28.1. INSTALLING AND CONFIGURING SAMPLE ALERT AGENTS

When you use one of the sample alert agents, you should review the script to ensure that it suits your needs. These sample agents are provided as a starting point for custom scripts for specific cluster environments. Note that while Red Hat supports the interfaces that the alert agents scripts use to communicate with Pacemaker, Red Hat does not provide support for the custom agents themselves.

To use one of the sample alert agents, you must install the agent on each node in the cluster. For example, the following command installs the **alert_file.sh.sample** script as **alert_file.sh**.

```
# install --mode=0755 /usr/share/pacemaker/alerts/alert_file.sh.sample
/var/lib/pacemaker/alert_file.sh
```

After you have installed the script, you can create an alert that uses the script.

The following example configures an alert that uses the installed **alert_file.sh** alert agent to log events to a file. Alert agents run as the user **hacluster**, which has a minimal set of permissions.

This example creates the log file **pcmk_alert_file.log** that will be used to record the events. It then creates the alert agent and adds the path to the log file as its recipient.

```
# touch /var/log/pcmk_alert_file.log
# chown hacluster:haclient /var/log/pcmk_alert_file.log
# chmod 600 /var/log/pcmk_alert_file.log
# pcs alert create id=alert_file description="Log events to a file."
path=/var/lib/pacemaker/alert_file.sh
# pcs alert recipient add alert_file id=my-alert_logfile value=/var/log/pcmk_alert_file.log
```

The following example installs the **alert_snmp.sh.sample** script as **alert_snmp.sh** and configures an alert that uses the installed **alert_snmp.sh** alert agent to send cluster events as SNMP traps. By default, the script will send all events except successful monitor calls to the SNMP server. This example configures the timestamp format as a meta option. After configuring the alert, this example configures a recipient for the alert and displays the alert configuration.

```
# install --mode=0755 /usr/share/pacemaker/alerts/alert_snmp.sh.sample
/var/lib/pacemaker/alert_snmp.sh
# pcs alert create id=snmp_alert path=/var/lib/pacemaker/alert_snmp.sh meta timestamp-
format="%Y-%m-%d,%H:%M:%S.%01N"
# pcs alert recipient add snmp_alert value=192.168.1.2
# pcs alert
Alerts:
 Alert: snmp_alert (path=/var/lib/pacemaker/alert_snmp.sh)
  Meta options: timestamp-format=%Y-%m-%d,%H:%M:%S.%01N.
  Recipients:
   Recipient: snmp_alert-recipient (value=192.168.1.2)
```

The following example installs the **alert_smtp.sh** agent and then configures an alert that uses the installed alert agent to send cluster events as email messages. After configuring the alert, this example configures a recipient and displays the alert configuration.

```
# install --mode=0755 /usr/share/pacemaker/alerts/alert_smtp.sh.sample
/var/lib/pacemaker/alert_smtp.sh
# pcs alert create id=smtp_alert path=/var/lib/pacemaker/alert_smtp.sh options
email_sender=donotreply@example.com
# pcs alert recipient add smtp_alert value=admin@example.com
# pcs alert
Alerts:
 Alert: smtp_alert (path=/var/lib/pacemaker/alert_smtp.sh)
  Options: email_sender=donotreply@example.com
  Recipients:
   Recipient: smtp_alert-recipient (value=admin@example.com)
```

28.2. CREATING A CLUSTER ALERT

The following command creates a cluster alert. The options that you configure are agent-specific configuration values that are passed to the alert agent script at the path you specify as additional environment variables. If you do not specify a value for **id**, one will be generated.

```
pcs alert create path=path [id=alert-id] [description=description] [options [option=value]...] [meta
[meta-option=value]...]
```

Multiple alert agents may be configured; the cluster will call all of them for each event. Alert agents will be called only on cluster nodes. They will be called for events involving Pacemaker Remote nodes, but they will never be called on those nodes.

The following example creates a simple alert that will call **myscript.sh** for each event.

```
# pcs alert create id=my_alert path=/path/to/myscript.sh
```

28.3. DISPLAYING, MODIFYING, AND REMOVING CLUSTER ALERTS

There are a variety of **pcs** commands you can use to display, modify, and remove cluster alerts.

The following command shows all configured alerts along with the values of the configured options.

```
pcs alert [config|show]
```

The following command updates an existing alert with the specified *alert-id* value.

```
pcs alert update alert-id [path=path] [description=description] [options [option=value]...] [meta [meta-option=value]...]
```

The following command removes an alert with the specified *alert-id* value.

```
pcs alert remove alert-id
```

Alternately, you can run the **pcs alert delete** command, which is identical to the **pcs alert remove** command. Both the **pcs alert delete** and the **pcs alert remove** commands allow you to specify more than one alert to be deleted.

28.4. CONFIGURING CLUSTER ALERT RECIPIENTS

Usually alerts are directed towards a recipient. Thus each alert may be additionally configured with one or more recipients. The cluster will call the agent separately for each recipient.

The recipient may be anything the alert agent can recognize: an IP address, an email address, a file name, or whatever the particular agent supports.

The following command adds a new recipient to the specified alert.

```
pcs alert recipient add alert-id value=recipient-value [id=recipient-id] [description=description] [options [option=value]...] [meta [meta-option=value]...]
```

The following command updates an existing alert recipient.

```
pcs alert recipient update recipient-id [value=recipient-value] [description=description] [options [option=value]...] [meta [meta-option=value]...]
```

The following command removes the specified alert recipient.

```
pcs alert recipient remove recipient-id
```

Alternately, you can run the **pcs alert recipient delete** command, which is identical to the **pcs alert recipient remove** command. Both the **pcs alert recipient remove** and the **pcs alert recipient delete** commands allow you to remove more than one alert recipient.

The following example command adds the alert recipient **my-alert-recipient** with a recipient ID of **my-recipient-id** to the alert **my-alert**. This will configure the cluster to call the alert script that has been configured for **my-alert** for each event, passing the recipient **some-address** as an environment variable.

```
# pcs alert recipient add my-alert value=my-alert-recipient id=my-recipient-id options value=some-address
```

28.5. ALERT META OPTIONS

As with resource agents, meta options can be configured for alert agents to affect how Pacemaker calls them. The following table describes the alert meta options. Meta options can be configured per alert agent as well as per recipient.

Table 28.1. Alert Meta Options

Meta-Attribute	Default	Description
enabled	**true**	(RHEL 9.3 and later) If set to **false** for an alert, the alert will not be used. If set to **true** for an alert and **false** for a particular recipient of that alert, that recipient will not be used.
timestamp-format	%H:%M:%S.%06N	Format the cluster will use when sending the event's timestamp to the agent. This is a string as used with the **date**(1) command.
timeout	30s	If the alert agent does not complete within this amount of time, it will be terminated.

The following example configures an alert that calls the script **myscript.sh** and then adds two recipients to the alert. The first recipient has an ID of **my-alert-recipient1** and the second recipient has an ID of **my-alert-recipient2**. The script will get called twice for each event, with each call using a 15-second timeout. One call will be passed to the recipient **someuser@example.com** with a timestamp in the format %D %H:%M, while the other call will be passed to the recipient **otheruser@example.com** with a timestamp in the format %c.

```
# pcs alert create id=my-alert path=/path/to/myscript.sh meta timeout=15s
# pcs alert recipient add my-alert value=someuser@example.com id=my-alert-recipient1 meta
timestamp-format="%D %H:%M"
# pcs alert recipient add my-alert value=otheruser@example.com id=my-alert-recipient2 meta
timestamp-format="%c"
```

28.6. CLUSTER ALERT CONFIGURATION COMMAND EXAMPLES

The following sequential examples show some basic alert configuration commands to show the format to use to create alerts, add recipients, and display the configured alerts.

Note that while you must install the alert agents themselves on each node in a cluster, you need to run the **pcs** commands only once.

The following commands create a simple alert, add two recipients to the alert, and display the configured values.

- Since no alert ID value is specified, the system creates an alert ID value of **alert**.
- The first recipient creation command specifies a recipient of **rec_value**. Since this command does not specify a recipient ID, the value of **alert-recipient** is used as the recipient ID.
- The second recipient creation command specifies a recipient of **rec_value2**. This command specifies a recipient ID of **my-recipient** for the recipient.

```
# pcs alert create path=/my/path
# pcs alert recipient add alert value=rec_value
# pcs alert recipient add alert value=rec_value2 id=my-recipient
# pcs alert config
Alerts:
 Alert: alert (path=/my/path)
  Recipients:
   Recipient: alert-recipient (value=rec_value)
   Recipient: my-recipient (value=rec_value2)
```

This following commands add a second alert and a recipient for that alert. The alert ID for the second alert is **my-alert** and the recipient value is **my-other-recipient**. Since no recipient ID is specified, the system provides a recipient id of **my-alert-recipient**.

```
# pcs alert create id=my-alert path=/path/to/script description=alert_description options
option1=value1 opt=val meta timeout=50s timestamp-format="%H%B%S"
# pcs alert recipient add my-alert value=my-other-recipient
# pcs alert
Alerts:
 Alert: alert (path=/my/path)
  Recipients:
   Recipient: alert-recipient (value=rec_value)
   Recipient: my-recipient (value=rec_value2)
 Alert: my-alert (path=/path/to/script)
  Description: alert_description
  Options: opt=val option1=value1
  Meta options: timestamp-format=%H%B%S timeout=50s
  Recipients:
   Recipient: my-alert-recipient (value=my-other-recipient)
```

The following commands modify the alert values for the alert **my-alert** and for the recipient **my-alert-recipient**.

```
# pcs alert update my-alert options option1=newvalue1 meta timestamp-format="%H%M%S"
# pcs alert recipient update my-alert-recipient options option1=new meta timeout=60s
# pcs alert
Alerts:
 Alert: alert (path=/my/path)
  Recipients:
   Recipient: alert-recipient (value=rec_value)
   Recipient: my-recipient (value=rec_value2)
 Alert: my-alert (path=/path/to/script)
  Description: alert_description
  Options: opt=val option1=newvalue1
  Meta options: timestamp-format=%H%M%S timeout=50s
  Recipients:
   Recipient: my-alert-recipient (value=my-other-recipient)
    Options: option1=new
    Meta options: timeout=60s
```

The following command removes the recipient **my-alert-recipient** from **alert**.

```
# pcs alert recipient remove my-recipient
# pcs alert
Alerts:
 Alert: alert (path=/my/path)
```

```
  Recipients:
   Recipient: alert-recipient (value=rec_value)
 Alert: my-alert (path=/path/to/script)
  Description: alert_description
  Options: opt=val option1=newvalue1
  Meta options: timestamp-format="%M%B%S" timeout=50s
  Recipients:
   Recipient: my-alert-recipient (value=my-other-recipient)
    Options: option1=new
    Meta options: timeout=60s
```

The following command removes **myalert** from the configuration.

```
# pcs alert remove myalert
# pcs alert
Alerts:
 Alert: alert (path=/my/path)
  Recipients:
   Recipient: alert-recipient (value=rec_value)
```

28.7. WRITING A CLUSTER ALERT AGENT

There are three types of Pacemaker cluster alerts: node alerts, fencing alerts, and resource alerts. The environment variables that are passed to the alert agents can differ, depending on the type of alert. The following table describes the environment variables that are passed to alert agents and specifies when the environment variable is associated with a specific alert type.

Table 28.2. Environment Variables Passed to Alert Agents

Environment Variable	Description
CRM_alert_kind	The type of alert (node, fencing, or resource)
CRM_alert_version	The version of Pacemaker sending the alert
CRM_alert_recipient	The configured recipient
CRM_alert_node_sequence	A sequence number increased whenever an alert is being issued on the local node, which can be used to reference the order in which alerts have been issued by Pacemaker. An alert for an event that happened later in time reliably has a higher sequence number than alerts for earlier events. Be aware that this number has no cluster-wide meaning.
CRM_alert_timestamp	A timestamp created prior to executing the agent, in the format specified by the **timestamp-format** meta option. This allows the agent to have a reliable, high-precision time of when the event occurred, regardless of when the agent itself was invoked (which could potentially be delayed due to system load or other circumstances).
CRM_alert_node	Name of affected node

Environment Variable	Description
CRM_alert_desc	Detail about event. For node alerts, this is the node's current state (member or lost). For fencing alerts, this is a summary of the requested fencing operation, including origin, target, and fencing operation error code, if any. For resource alerts, this is a readable string equivalent of **CRM_alert_status**.
CRM_alert_nodeid	ID of node whose status changed (provided with node alerts only)
CRM_alert_task	The requested fencing or resource operation (provided with fencing and resource alerts only)
CRM_alert_rc	The numerical return code of the fencing or resource operation (provided with fencing and resource alerts only)
CRM_alert_rsc	The name of the affected resource (resource alerts only)
CRM_alert_interval	The interval of the resource operation (resource alerts only)
CRM_alert_target_rc	The expected numerical return code of the operation (resource alerts only)
CRM_alert_status	A numerical code used by Pacemaker to represent the operation result (resource alerts only)

When writing an alert agent, you must take the following concerns into account.

- Alert agents may be called with no recipient (if none is configured), so the agent must be able to handle this situation, even if it only exits in that case. Users may modify the configuration in stages, and add a recipient later.
- If more than one recipient is configured for an alert, the alert agent will be called once per recipient. If an agent is not able to run concurrently, it should be configured with only a single recipient. The agent is free, however, to interpret the recipient as a list.
- When a cluster event occurs, all alerts are fired off at the same time as separate processes. Depending on how many alerts and recipients are configured and on what is done within the alert agents, a significant load burst may occur. The agent could be written to take this into consideration, for example by queueing resource-intensive actions into some other instance, instead of directly executing them.
- Alert agents are run as the **hacluster** user, which has a minimal set of permissions. If an agent requires additional privileges, it is recommended to configure **sudo** to allow the agent to run the necessary commands as another user with the appropriate privileges.
- Take care to validate and sanitize user-configured parameters, such as **CRM_alert_timestamp** (whose content is specified by the user-configured **timestamp-format**), **CRM_alert_recipient**, and all alert options. This is necessary to protect against configuration errors. In addition, if some user can modify the CIB without having **hacluster**-level access to the cluster nodes, this is a potential security concern as well, and you should avoid the possibility of code injection.

- If a cluster contains resources with operations for which the **on-fail** parameter is set to **fence**, there will be multiple fence notifications on failure, one for each resource for which this parameter is set plus one additional notification. Both the **pacemaker-fenced** and **pacemaker-controld** will send notifications. Pacemaker performs only one actual fence operation in this case, however, no matter how many notifications are sent.

NOTE

The alerts interface is designed to be backward compatible with the external scripts interface used by the **ocf:pacemaker:ClusterMon** resource. To preserve this compatibility, the environment variables passed to alert agents are available prepended with **CRM_notify_** as well as **CRM_alert_**. One break in compatibility is that the **ClusterMon** resource ran external scripts as the root user, while alert agents are run as the **hacluster** user.

CHAPTER 29. MULTI-SITE PACEMAKER CLUSTERS

When a cluster spans more than one site, issues with network connectivity between the sites can lead to split-brain situations. When connectivity drops, there is no way for a node on one site to determine whether a node on another site has failed or is still functioning with a failed site interlink. In addition, it can be problematic to provide high availability services across two sites which are too far apart to keep synchronous. To address these issues, Pacemaker provides full support for the ability to configure high availability clusters that span multiple sites through the use of a Booth cluster ticket manager.

29.1. OVERVIEW OF BOOTH CLUSTER TICKET MANAGER

The Booth *ticket manager* is a distributed service that is meant to be run on a different physical network than the networks that connect the cluster nodes at particular sites. It yields another, loose cluster, a *Booth formation*, that sits on top of the regular clusters at the sites. This aggregated communication layer facilitates consensus-based decision processes for individual Booth tickets.

A Booth *ticket* is a singleton in the Booth formation and represents a time-sensitive, movable unit of authorization. Resources can be configured to require a certain ticket to run. This can ensure that resources are run at only one site at a time, for which a ticket or tickets have been granted.

You can think of a Booth formation as an overlay cluster consisting of clusters running at different sites, where all the original clusters are independent of each other. It is the Booth service which communicates to the clusters whether they have been granted a ticket, and it is Pacemaker that determines whether to run resources in a cluster based on a Pacemaker ticket constraint. This means that when using the ticket manager, each of the clusters can run its own resources as well as shared resources. For example there can be resources A, B and C running only in one cluster, resources D, E, and F running only in the other cluster, and resources G and H running in either of the two clusters as determined by a ticket. It is also possible to have an additional resource J that could run in either of the two clusters as determined by a separate ticket.

29.2. CONFIGURING MULTI-SITE CLUSTERS WITH PACEMAKER

You can configure a multi-site configuration that uses the Booth ticket manager with the following procedure.

These example commands use the following arrangement:

- Cluster 1 consists of the nodes **cluster1-node1** and **cluster1-node2**
- Cluster 1 has a floating IP address assigned to it of 192.168.11.100
- Cluster 2 consists of **cluster2-node1** and **cluster2-node2**
- Cluster 2 has a floating IP address assigned to it of 192.168.22.100
- The arbitrator node is **arbitrator-node** with an ip address of 192.168.99.100
- The name of the Booth ticket that this configuration uses is **apacheticket**

These example commands assume that the cluster resources for an Apache service have been configured as part of the resource group **apachegroup** for each cluster. It is not required that the resources and resource groups be the same on each cluster to configure a ticket constraint for those resources, since the Pacemaker instance for each cluster is independent, but that is a common failover scenario.

Note that at any time in the configuration procedure you can enter the **pcs booth config** command to display the booth configuration for the current node or cluster or the **pcs booth status** command to display the current status of booth on the local node.

Procedure

1. Install the **booth-site** Booth ticket manager package on each node of both clusters.

   ```
   [root@cluster1-node1 ~]# dnf install -y booth-site
   [root@cluster1-node2 ~]# dnf install -y booth-site
   [root@cluster2-node1 ~]# dnf install -y booth-site
   [root@cluster2-node2 ~]# dnf install -y booth-site
   ```

2. Install the **pcs**, **booth-core**, and **booth-arbitrator** packages on the arbitrator node.

   ```
   [root@arbitrator-node ~]# dnf install -y pcs booth-core booth-arbitrator
   ```

3. If you are running the **firewalld** daemon, execute the following commands on all nodes in both clusters as well as on the arbitrator node 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
   ```

   You may need to modify which ports are open to suit local conditions. For more information about the ports that are required by the Red Hat High-Availability Add-On, see Enabling ports for the High Availability Add-On.

4. Create a Booth configuration on one node of one cluster. The addresses you specify for each cluster and for the arbitrator must be IP addresses. For each cluster, you specify a floating IP address.

   ```
   [cluster1-node1 ~] # pcs booth setup sites 192.168.11.100 192.168.22.100 arbitrators
   192.168.99.100
   ```

   This command creates the configuration files **/etc/booth/booth.conf** and **/etc/booth/booth.key** on the node from which it is run.

5. Create a ticket for the Booth configuration. This is the ticket that you will use to define the resource constraint that will allow resources to run only when this ticket has been granted to the cluster.
   This basic failover configuration procedure uses only one ticket, but you can create additional tickets for more complicated scenarios where each ticket is associated with a different resource or resources.

   ```
   [cluster1-node1 ~] # pcs booth ticket add apacheticket
   ```

6. Synchronize the Booth configuration to all nodes in the current cluster.

   ```
   [cluster1-node1 ~] # pcs booth sync
   ```

7. From the arbitrator node, pull the Booth configuration to the arbitrator. If you have not previously done so, you must first authenticate **pcs** to the node from which you are pulling the configuration.

```
[arbitrator-node ~] # pcs host auth cluster1-node1
[arbitrator-node ~] # pcs booth pull cluster1-node1
```

8. Pull the Booth configuration to the other cluster and synchronize to all the nodes of that cluster. As with the arbitrator node, if you have not previously done so, you must first authenticate **pcs** to the node from which you are pulling the configuration.

```
[cluster2-node1 ~] # pcs host auth cluster1-node1
[cluster2-node1 ~] # pcs booth pull cluster1-node1
[cluster2-node1 ~] # pcs booth sync
```

9. Start and enable Booth on the arbitrator.

   NOTE

   You must not manually start or enable Booth on any of the nodes of the clusters since Booth runs as a Pacemaker resource in those clusters.

```
[arbitrator-node ~] # pcs booth start
[arbitrator-node ~] # pcs booth enable
```

10. Configure Booth to run as a cluster resource on both cluster sites, using the floating IP addresses assigned to each cluster. This creates a resource group with **booth-ip** and **booth-service** as members of that group.

```
[cluster1-node1 ~] # pcs booth create ip 192.168.11.100
[cluster2-node1 ~] # pcs booth create ip 192.168.22.100
```

11. Add a ticket constraint to the resource group you have defined for each cluster.

```
[cluster1-node1 ~] # pcs constraint ticket add apacheticket apachegroup
[cluster2-node1 ~] # pcs constraint ticket add apacheticket apachegroup
```

    You can enter the following command to display the currently configured ticket constraints.

```
pcs constraint ticket [show]
```

12. Grant the ticket you created for this setup to the first cluster.
    Note that it is not necessary to have defined ticket constraints before granting a ticket. Once you have initially granted a ticket to a cluster, then Booth takes over ticket management unless you override this manually with the **pcs booth ticket revoke** command. For information about the **pcs booth** administration commands, see the PCS help screen for the **pcs booth** command.

```
[cluster1-node1 ~] # pcs booth ticket grant apacheticket
```

It is possible to add or remove tickets at any time, even after completing this procedure. After adding or removing a ticket, however, you must synchronize the configuration files to the other nodes and clusters as well as to the arbitrator and grant the ticket as is shown in this procedure.

For information about additional Booth administration commands that you can use for cleaning up and removing Booth configuration files, tickets, and resources, see the PCS help screen for the **pcs booth** command.

CHAPTER 30. 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 Red Hat support limit of 32 nodes.
- 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.

30.1. 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.

30.2. 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.

30.2.1. 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 the "Resource Options for Virtual Domain Resources" table in Virtual domain resource options.

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. The following table describes these metadata options.

Table 30.1. 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.
remote-port	3121	Configures a custom port to use for the guest connection to **pacemaker_remote**

Field	Default	Description
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

30.2.2. 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.

Procedure

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.

```
# dnf 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.
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
```

30.3. 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.

30.3.1. Remote node resource options

The following table describes the resource options you can configure for a **remote** resource.

Table 30.2. 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.

30.3.2. Remote node configuration overview

The following procedure 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.

Procedure

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.

```
# dnf 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.

30.4. 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).

30.5. 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**.

Procedure

1. Stop the node's connection resource with the **pcs resource disable *resourcename*** command, which will move all services off the node. The connection resource would be the **ocf:pacemaker:remote** resource for a remote node or, commonly, the **ocf:heartbeat:VirtualDomain** resource for a guest node. For guest nodes, this command will also stop the VM, so the VM must be started outside the cluster (for example, using **virsh**) to perform any maintenance.

   ```
   pcs resource disable resourcename
   ```

2. Perform the required maintenance.

3. When ready to return the node to the cluster, re-enable the resource with the **pcs resource enable** command.

   ```
   pcs resource enable resourcename
   ```

CHAPTER 31. 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 about 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.
  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.

- 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 Disabling, enabling, 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 about 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 RHEL 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.

- If you need to migrate a VM in a RHEL cluster, you will first need to stop the cluster services on the VM to remove the node from the cluster and then start the cluster back up after performing the migration. as described in Migrating VMs in a RHEL cluster .

31.1. 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.

31.2. 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.

31.2.1. 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] [--promoted] [--strict] [--wait[=n]]
```

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. By default, the location constraint that the command creates is automatically removed once the resource has been moved. If removing the constraint would cause the resource to move back to the original node, as might happen if the

resource-stickiness value for the resource is 0, the **pcs resource move** command fails. If you would like to move a resource and leave the resulting constraint in place, use the **pcs resource move-with-constraint** command.

If you specify the **--promoted** parameter of the **pcs resource move** command, the constraint applies only to promoted instances of the resource.

If you specify the **--strict** parameter of the **pcs resource move** command, the command will fail if other resources than the one specified in the command would be affected.

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, it defaults to a value of 60 minutes.

31.2.2. 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.

31.3. DISABLING, ENABLING, 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.

Disabling a cluster resource

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 specify that a resource be disabled only if disabling the resource would not have an effect on other resources. Ensuring that this would be the case can be impossible to do by hand when complex resource relations are set up.

- The **pcs resource disable --simulate** command shows the effects of disabling a resource while not changing the cluster configuration.

- The **pcs resource disable --safe** command disables a resource only if no other resources would be affected in any way, such as being migrated from one node to another. The **pcs resource safe-disable** command is an alias for the **pcs resource disable --safe** command.
- The **pcs resource disable --safe --no-strict** command disables a resource only if no other resources would be stopped or demoted

You can specify the **--brief** option for the **pcs resource disable --safe** command to print errors only. The error report that the **pcs resource disable --safe** command generates if the safe disable operation fails contains the affected resource IDs. If you need to know only the resource IDs of resources that would be affected by disabling a resource, use the **--brief** option, which does not provide the full simulation result.

Enabling a cluster resource

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]]
```

Preventing a resource from running on a particular node

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] [--promoted] [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 **--promoted** parameter of the **pcs resource ban** command, the scope of the constraint is limited to the promoted role and you must specify *promotable_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.

Forcing a resource to start on the current node

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
```

31.4. 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.

31.5. 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
```

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 symmetric-cluster=
```

31.6. 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.

31.7. 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**.

Procedure

1. Stop the node's connection resource with the **pcs resource disable *resourcename*** command, which will move all services off the node. The connection resource would be the **ocf:pacemaker:remote** resource for a remote node or, commonly, the **ocf:heartbeat:VirtualDomain** resource for a guest node. For guest nodes, this command will also stop the VM, so the VM must be started outside the cluster (for example, using **virsh**) to perform any maintenance.

   ```
   pcs resource disable resourcename
   ```

2. Perform the required maintenance.

3. When ready to return the node to the cluster, re-enable the resource with the **pcs resource enable** command.

```
pcs resource enable resourcename
```

31.8. MIGRATING VMS IN A RHEL CLUSTER

Red Hat does not support live migration of active cluster nodes across hypervisors or hosts, as noted in Support Policies for RHEL High Availability Clusters - General Conditions with Virtualized Cluster Members. If you need to perform a live migration, you will first need to stop the cluster services on the VM to remove the node from the cluster, and then start the cluster back up after performing the migration. The following steps outline the procedure for removing a VM from a cluster, migrating the VM, and restoring the VM to the cluster.

The following steps outline the procedure for removing a VM from a cluster, migrating the VM, and restoring the VM to the cluster.

This procedure applies to VMs that are used as full cluster nodes, not to VMs managed as cluster resources (including VMs used as guest nodes) which can be live-migrated without special precautions. For general information about the fuller procedure required for updating packages that make up the RHEL High Availability and Resilient Storage Add-Ons, either individually or as a whole, see Recommended Practices for Applying Software Updates to a RHEL High Availability or Resilient Storage Cluster.

NOTE

Before performing this procedure, consider the effect on cluster quorum of removing a cluster node. For example, if you have a three-node cluster and you remove one node, your cluster can not withstand any node failure. This is because if one node of a three-node cluster is already down, removing a second node will lose quorum.

Procedure

1. If any preparations need to be made before stopping or moving the resources or software running on the VM to migrate, perform those steps.

2. Run the following command on the VM to stop the cluster software on the VM.

```
# pcs cluster stop
```

3. Perform the live migration of the VM.

4. Start cluster services on the VM.

```
# pcs cluster start
```

31.9. IDENTIFYING CLUSTERS BY UUID

As of Red Hat Enterprise Linux 9.1, when you create a cluster it has an associated UUID. Since a cluster name is not a unique cluster identifier, a third-party tool such as a configuration management database that manages multiple clusters with the same name can uniquely identify a cluster by means of its UUID. You can display the current cluster UUID with the **pcs cluster config [show]** command, which includes the cluster UUID in its output.

To add a UUID to an existing cluster, run the following command.

```
# pcs cluster config uuid generate
```

To regenerate a UUID for a cluster with an existing UUID, run the following command.

```
# pcs cluster config uuid generate --force
```

CHAPTER 32. CONFIGURING DISASTER RECOVERY CLUSTERS

One method of providing disaster recovery for a high availability cluster is to configure two clusters. You can then configure one cluster as your primary site cluster, and the second cluster as your disaster recovery cluster.

In normal circumstances, the primary cluster is running resources in production mode. The disaster recovery cluster has all the resources configured as well and is either running them in demoted mode or not at all. For example, there may be a database running in the primary cluster in promoted mode and running in the disaster recovery cluster in demoted mode. The database in this setup would be configured so that data is synchronized from the primary to disaster recovery site. This is done through the database configuration itself rather than through the **pcs** command interface.

When the primary cluster goes down, users can use the **pcs** command interface to manually fail the resources over to the disaster recovery site. They can then log in to the disaster site and promote and start the resources there. Once the primary cluster has recovered, users can use the **pcs** command interface to manually move resources back to the primary site.

You can use the **pcs** command to display the status of both the primary and the disaster recovery site cluster from a single node on either site.

32.1. CONSIDERATIONS FOR DISASTER RECOVERY CLUSTERS

When planning and configuring a disaster recovery site that you will manage and monitor with the **pcs** command interface, note the following considerations.

- The disaster recovery site must be a cluster. This makes it possible to configure it with same tools and similar procedures as the primary site.
- The primary and disaster recovery clusters are created by independent **pcs cluster setup** commands.
- The clusters and their resources must be configured so that that the data is synchronized and failover is possible.
- The cluster nodes in the recovery site can not have the same names as the nodes in the primary site.
- The pcs user **hacluster** must be authenticated for each node in both clusters on the node from which you will be running **pcs** commands.

32.2. DISPLAYING STATUS OF RECOVERY CLUSTERS

To configure a primary and a disaster recovery cluster so that you can display the status of both clusters, perform the following procedure.

NOTE

Setting up a disaster recovery cluster does not automatically configure resources or replicate data. Those items must be configured manually by the user.

In this example:

- The primary cluster will be named **PrimarySite** and will consist of the nodes **z1.example.com**. and **z2.example.com**.
- The disaster recovery site cluster will be named **DRsite** and will consist of the nodes **z3.example.com** and **z4.example.com**.

This example sets up a basic cluster with no resources or fencing configured.

Procedure

1. Authenticate all of the nodes that will be used for both clusters.

```
[root@z1 ~]# pcs host auth z1.example.com z2.example.com z3.example.com
z4.example.com -u hacluster -p password
z1.example.com: Authorized
z2.example.com: Authorized
z3.example.com: Authorized
z4.example.com: Authorized
```

2. Create the cluster that will be used as the primary cluster and start cluster services for the cluster.

```
[root@z1 ~]# pcs cluster setup PrimarySite z1.example.com z2.example.com --start
{...}
Cluster has been successfully set up.
Starting cluster on hosts: 'z1.example.com', 'z2.example.com'...
```

3. Create the cluster that will be used as the disaster recovery cluster and start cluster services for the cluster.

```
[root@z1 ~]# pcs cluster setup DRSite z3.example.com z4.example.com --start
{...}
Cluster has been successfully set up.
Starting cluster on hosts: 'z3.example.com', 'z4.example.com'...
```

4. From a node in the primary cluster, set up the second cluster as the recovery site. The recovery site is defined by a name of one of its nodes.

```
[root@z1 ~]# pcs dr set-recovery-site z3.example.com
Sending 'disaster-recovery config' to 'z3.example.com', 'z4.example.com'
z3.example.com: successful distribution of the file 'disaster-recovery config'
z4.example.com: successful distribution of the file 'disaster-recovery config'
Sending 'disaster-recovery config' to 'z1.example.com', 'z2.example.com'
z1.example.com: successful distribution of the file 'disaster-recovery config'
z2.example.com: successful distribution of the file 'disaster-recovery config'
```

5. Check the disaster recovery configuration.

```
[root@z1 ~]# pcs dr config
Local site:
  Role: Primary
Remote site:
  Role: Recovery
```

```
Nodes:
  z3.example.com
  z4.example.com
```

6. Check the status of the primary cluster and the disaster recovery cluster from a node in the primary cluster.

```
[root@z1 ~]# pcs dr status
--- Local cluster - Primary site ---
Cluster name: PrimarySite

WARNINGS:
No stonith devices and stonith-enabled is not false

Cluster Summary:
  * Stack: corosync
  * Current DC: z2.example.com (version 2.0.3-2.el8-2c9cea563e) - partition with quorum
  * Last updated: Mon Dec  9 04:10:31 2019
  * Last change:  Mon Dec  9 04:06:10 2019 by hacluster via crmd on z2.example.com
  * 2 nodes configured
  * 0 resource instances configured

Node List:
  * Online: [ z1.example.com z2.example.com ]

Full List of Resources:
  * No resources

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


--- Remote cluster - Recovery site ---
Cluster name: DRSite

WARNINGS:
No stonith devices and stonith-enabled is not false

Cluster Summary:
  * Stack: corosync
  * Current DC: z4.example.com (version 2.0.3-2.el8-2c9cea563e) - partition with quorum
  * Last updated: Mon Dec  9 04:10:34 2019
  * Last change:  Mon Dec  9 04:09:55 2019 by hacluster via crmd on z4.example.com
  * 2 nodes configured
  * 0 resource instances configured

Node List:
  * Online: [ z3.example.com z4.example.com ]

Full List of Resources:
  * No resources

Daemon Status:
```

```
corosync: active/disabled
pacemaker: active/disabled
pcsd: active/enabled
```

For additional display options for a disaster recovery configuration, see the help screen for the **pcs dr** command.

CHAPTER 33. INTERPRETING RESOURCE AGENT OCF RETURN CODES

Pacemaker resource agents conform to the Open Cluster Framework (OCF) Resource Agent API. This following tables describe the OCF return codes and how they are interpreted by Pacemaker.

The first thing the cluster does when an agent returns a code is to check the return code against the expected result. If the result does not match the expected value, then the operation is considered to have failed, and recovery action is initiated.

For any invocation, resource agents must exit with a defined return code that informs the caller of the outcome of the invoked action.

There are three types of failure recovery, as described in the following table.

Table 33.1. Types of Recovery Performed by the Cluster

Type	Description	Action Taken by the Cluster
soft	A transient error occurred.	Restart the resource or move it to a new location .
hard	A non-transient error that may be specific to the current node occurred.	Move the resource elsewhere and prevent it from being retried on the current node.
fatal	A non-transient error that will be common to all cluster nodes occurred (for example, a bad configuration was specified).	Stop the resource and prevent it from being started on any cluster node.

The following table provides The OCF return codes and the type of recovery the cluster will initiate when a failure code is received.Note that even actions that return 0 (OCF alias **OCF_SUCCESS**) can be considered to have failed, if 0 was not the expected return value.

Table 33.2. OCF Return Codes

Return Code	OCF Label	Description
0	**OCF_SUCCESS**	* The action completed successfully. This is the expected return code for any successful start, stop, promote, and demote command. * Type if unexpected: soft
1	**OCF_ERR_GENERIC**	* The action returned a generic error. * Type: soft * The resource manager will attempt to recover the resource or move it to a new location.

Return Code	OCF Label	Description
2	**OCF_ERR_ARGS**	* The resource’s configuration is not valid on this machine. For example, it refers to a location not found on the node. * Type: hard * The resource manager will move the resource elsewhere and prevent it from being retried on the current node
3	**OCF_ERR_UNIMPLEMENTE D**	* The requested action is not implemented. * Type: hard
4	**OCF_ERR_PERM**	* The resource agent does not have sufficient privileges to complete the task. This may be due, for example, to the agent not being able to open a certain file, to listen on a specific socket, or to write to a directory. * Type: hard * Unless specifically configured otherwise, the resource manager will attempt to recover a resource which failed with this error by restarting the resource on a different node (where the permission problem may not exist).
5	**OCF_ERR_INSTALLED**	* A required component is missing on the node where the action was executed. This may be due to a required binary not being executable, or a vital configuration file being unreadable. * Type: hard * Unless specifically configured otherwise, the resource manager will attempt to recover a resource which failed with this error by restarting the resource on a different node (where the required files or binaries may be present).
6	**OCF_ERR_CONFIGURED**	* The resource’s configuration on the local node is invalid. * Type: fatal * When this code is returned, Pacemaker will prevent the resource from running on any node in the cluster, even if the service configuraiton is valid on some other node.
7	**OCF_NOT_RUNNING**	* The resource is safely stopped. This implies that the resource has either gracefully shut down, or has never been started. * Type if unexpected: soft * The cluster will not attempt to stop a resource that returns this for any action.

Return Code	OCF Label	Description
8	**OCF_RUNNING_PROMOTED**	* The resource is running in promoted role. * Type if unexpected: soft
9	**OCF_FAILED_PROMOTED**	* The resource is (or might be) in promoted role but has failed. * Type: soft * The resource will be demoted, stopped and then started (and possibly promoted) again.
190		* The service is found to be properly active, but in such a condition that future failures are more likely.
191		* The resource agent supports roles and the service is found to be properly active in the promoted role, but in such a condition that future failures are more likely.
other	N/A	Custom error code.

CHAPTER 34. CONFIGURING A RED HAT HIGH AVAILABILITY CLUSTER WITH IBM Z/VM INSTANCES AS CLUSTER MEMBERS

Red Hat provides several articles that may be useful when designing, configuring, and administering a Red Hat High Availability cluster running on z/VM virtual machines.

- Design Guidance for RHEL High Availability Clusters - IBM z/VM Instances as Cluster Members
- Administrative Procedures for RHEL High Availability Clusters - Configuring z/VM SMAPI Fencing with fence_zvmip for RHEL 7 or 8 IBM z Systems Cluster Members
- RHEL High Availability cluster nodes on IBM z Systems experience STONITH-device timeouts around midnight on a nightly basis
- Administrative Procedures for RHEL High Availability Clusters - Preparing a dasd Storage Device for Use by a Cluster of IBM z Systems Members

You may also find the following articles useful when designing a Red Hat High Availability cluster in general.

- Support Policies for RHEL High Availability Clusters
- Exploring Concepts of RHEL High Availability Clusters - Fencing/STONITH