Oracle Linux 9

Setting Up High Availability Clustering

F60345-05
April 2024

Oracle Linux 9 Setting Up High Availability Clustering,

F60345-05

Contents

Preface

Oracle Linux 9: Setting Up High Availability Clustering describes how to install and configure high availability clustering in Oracle Linux using Corosync and Pacemaker, which are tools that enable you to achieve high availability for applications and services that are running on Oracle Linux.

Documentation License

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at https://www.oracle.com/corporate/accessibility/.

Access to Oracle Support for Accessibility

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit https://www.oracle.com/corporate/accessibility/learning-support.html#support-tab.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

1

About High Availability Clustering

This chapter describes how to set up and configure the Pacemaker and Corosync technologies to create a high availability (HA) cluster that delivers continuous access to services that are running across multiple nodes.

High availability services in Oracle Linux consist of several open source packages, including the Corosync and Pacemaker features. These tools enable you to achieve high availability for applications and services that are running on Oracle Linux. You can download Corosync, Pacemaker, and any dependencies and related packages from the Unbreakable Linux Network (ULN) at https://linux.oracle.com or the Oracle Linux yum server at https://yum.oracle.com.

Corosync is an open source cluster engine that includes an API to implement several high availability features, including an availability manager that can restart a process when it fails, a configuration and statistics database, and a quorum system that can notify applications when quorum is achieved or lost.

Corosync is installed with Pacemaker, which is an open source high availability cluster resource manager that's responsible for managing the life cycle of software that's deployed on a cluster. Pacemaker also provides high availability services, which are achieved by detecting and recovering from node and resource-level failures by using the API that's provided by the cluster engine.

Pacemaker also ships with the Pacemaker Command Shell (`pcs`). You can use the `pcs` command to access and configure the cluster and its resources. The `pcs` daemon runs as a service on each node in the cluster, making it possible to synchronize configuration changes across all of the nodes in the cluster.

Oracle provides support for Corosync and Pacemaker that's used for an active-passive 2-node (1:1) cluster configuration on Oracle Linux 9. Note that support for clustering services doesn't imply support for Oracle products that are clustered by using these services.

Oracle also provides Oracle Clusterware for high availability clustering with Oracle Database. You can find more information at https://www.oracle.com/database/technologies/rac/clusterware.html.

2

Installing and Configuring Pacemaker and Corosync

This chapter describes how to set up and configure the Pacemaker and Corosync features to create a high availability (HA) cluster that delivers continuous access to services running across multiple nodes.

Enabling Access to the Pacemaker and Corosync Packages

The Pacemaker and Corosync packages are available on the Oracle Linux yum server in the `ol9_addons` repository, or on the Unbreakable Linux Network (ULN) in the `ol9_`*`arch`*`_addons` channel.

Some dependency packages may be required from the `ol9_appstream` and `ol9_baseos_latest` yum repositories, or from the `ol9_`*`arch`*`_appstream` and `ol9_`*`arch`*`_baseos_latest` channels on ULN.

Enabling Repositories With ULN

If you are registered to use ULN, use the ULN web interface to subscribe the system to the appropriate channels.

To subscribe to the ULN channels:

1. Log in to https://linux.oracle.com with your ULN user name and password.
2. On the Systems tab, click the link named for the system in the list of registered machines.
3. On the System Details page, click **Manage Subscriptions**.
4. On the System Summary page, select each required channel from the list of available channels and click the right arrow to move the channel to the list of subscribed channels. Subscribe the system to the following channels:
   - `ol9_`*`arch`*`_appstream`
   - `ol9_`*`arch`*`_baseos_latest`
   - `ol9_`*`arch`*`_addons`
5. Click **Save Subscriptions**.

Enabling Repositories With the Oracle Linux Yum Server

If you are using the Oracle Linux yum server for system updates, enable the appropriate Oracle Linux yum repositories.

To enable the yum repositories:

1. Enable the following yum repositories:
   - `ol9_appstream`

- `ol9_baseos_latest`
- `ol9_addons`

Use the `dnf config-manager` tool to enable the yum repositories:

```
sudo dnf config-manager --enable ol9_appstream ol9_baseos_latest
ol9_addons
```

Installing and Enabling the Pacemaker and Corosync Service

On each node in the cluster, install the `pcs` and `pacemaker` software packages, along with all of the available resource and fence agents from the Oracle Linux yum server or from ULN, for example:

```
sudo dnf install pcs pacemaker resource-agents fence-agents-all
```

If you are running `firewalld`, add the `high-availability` service on each of the nodes so that the service components are able to communicate across the network. Per the command that is run in the following example, this step typically enables the following ports: TCP port 2224 (used by the `pcs` daemon), port 3121 (for Pacemaker Remote nodes), port 21064 (for DLM resources), and UDP ports 5405 (for Corosync clustering), and 5404 (for Corosync multicast, if configured):

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

To use the `pcs` command to configure and manage your cluster, you must set a password on each node for the `hacluster` user.

> **Tip:**
>
> It is helpful if you set the same password for this user as the password you set for the user on each node.

Use the `passwd` command on each node to set the password:

```
sudo passwd hacluster
```

Note that to use the `pcs` command, the `pcsd` service must be running on each of the nodes in the cluster. You can set this service to run and to start at boot by running the following command:

```
sudo systemctl enable --now pcsd.service
```

Note:

When running High Availability Clustering in the cloud, please refer to the following documents:

- Create a High Availability Cluster on Oracle Cloud Infrastructure (OCI)

3
Configuring an Initial Cluster and Service

This chapter provides an example, along with step-by-step instructions on configuring an initial cluster across two nodes that are hosted on systems with the resolvable host names `node1` and `node2`. Each system is installed and configured by using the instructions that are provided in Installing and Configuring Pacemaker and Corosync.

The cluster is configured to run a service, `Dummy`, that is included in the `resource-agents` package. You should have installed this package along with the `pacemaker` packages. This tool simply keeps track of whether the service is or is not running. Pacemaker is configured with an interval parameter that determines how long it should wait between checks to determine whether the `Dummy` process has failed.

The `Dummy` process is manually stopped outside of the Pacemaker tool to simulate a failure, which is used to demonstrate how the process is restarted automatically on an alternate node.

Creating the Cluster

To create the cluster:

1. Authenticate the `pcs` cluster configuration tool for the `hacluster` user on each node in your configuration by running the following command on one of the nodes that will form part of the cluster:

   ```
   sudo pcs host auth node1 node2 -u hacluster
   ```

   Replace *node1* and *node2* with the resolvable hostnames of the nodes that will form part of the cluster.

   Alternately, if the node names are not resolvable, specify the IP addresses where the nodes can be accessed, as shown in the following example:

   ```
   sudo pcs host auth node1 addr=192.0.2.1 node2 addr=192.0.2.2 -u hacluster
   ```

   Replace *192.0.2.1* and *192.0.2.2* with the IP addresses of each of the respective hosts in the cluster.

   The tool prompts you to provide a password for the `hacluster` user. Provide the password that you set for this user when you installed and configured the Pacemaker software on each node.

2. Create the cluster by using the `pcs cluster setup` command. You must specify a name for the cluster and the node names and IP addresses for each node in the cluster. For example, run the following command:

   ```
   sudo pcs cluster setup pacemaker1 node1 addr=192.0.2.1 node2
   addr=192.0.2.2
   ```

Replace *pacemaker1* with an appropriate name for the cluster. Replace *node1* and *node2* with the resolvable hostnames of the nodes in the cluster. Replace *192.0.2.1* and *192.0.2.2* with the IP addresses of each of the respective hosts in the cluster.

Note that if you used the `addr` option to specify the IP addresses when authenticated the nodes, you do not need to specify them again when running the `pcs cluster setup` command.

The cluster setup process destroys any existing cluster configuration on the specified nodes and creates a configuration file for the Corosync service that is copied to each of the nodes within the cluster.

You can, optionally, use the `--start` option when running the `pcs cluster setup` command to automatically start the cluster once it is created.

3. If you have not already started the cluster as part of the cluster setup command, start the cluster on all of the nodes. To start the cluster manually, use the `pcs` command:

```
sudo pcs cluster start --all
```

Starting the `pacemaker` service from `systemd` is another way to start the cluster on all nodes, for example:

```
sudo systemctl start pacemaker.service
```

4. Optionally, you can enable these services to start at boot time so that if a node reboots, it automatically rejoins the cluster, for example:

```
sudo pcs cluster enable --all
```

Alternately you can enable the `pacemaker` service from `systemd` on all nodes, for example:

```
sudo systemctl enable pacemaker.service
```

> **Note:**
>
> Some users prefer not to enable these services so that a node failure resulting in a full system reboot can be properly debugged before it rejoins the cluster.

Setting Cluster Parameters

Fencing is an important part of setting up a production-level HA cluster. For simplicity, it is disabled in this example. If you intend to take advantage of `stonith`, see About Fencing Configuration (stonith) for additional information.

To set cluster parameters:

1. Disable the fencing feature by running the following command:

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

   Fencing is an advanced feature that helps protect your data from being corrupted by nodes that might be failing or are unavailable. Pacemaker uses the term `stonith` (shoot the other node in the head) to describe fencing options. This configuration depends on particular hardware and a deeper understanding of the fencing process. For this reason, it is recommended that you disable the fencing feature.

2. Optionally, configure the cluster to ignore the quorum state by running the following command:

   ```
   sudo pcs property set no-quorum-policy=ignore
   ```

   Because this example uses a two-node cluster, disabling the no-quorum policy makes the most sense, as quorum technically requires a minimum of three nodes to be a viable configuration. Quorum is only achieved when more than half of the nodes agree on the status of the cluster.

   In the current release of Corosync, this issue is treated specially for two-node clusters, where the quorum value is artificially set to 1 so that the primary node is always considered in quorum. In the case where a network outage results in both nodes going offline for a period, the nodes race to fence each other and the first to succeed wins quorum. The fencing agent can usually be configured to give one node priority so that it is more likely to win quorum if this is preferred.

3. Configure a migration policy by running the following command:

   ```
   sudo pcs resource defaults update
   ```

   Running this command configures the cluster to move the service to a new node after a single failure.

Creating a Service and Testing Failover

To create a service and test failover:

Services are created and usually configured to run a resource agent that is responsible for starting and stopping processes. Most resource agents are created according to the OCF (Open Cluster Framework) specification, which is defined as an extension for the Linux Standard Base (LSB). Many handy resource agents for commonly used processes are included in the `resource-agents` packages, including various heartbeat agents that track whether commonly used daemons or services are still running.

In the following example, a service is set up that uses a Dummy resource agent created precisely to test Pacemaker. This agent is used because it requires a basic configuration and doesn't make any assumptions about the environment or the types of services that you intend to run with Pacemaker.

1. Add the service as a resource by using the `pcs resource create` command:

   ```
   sudo pcs resource create dummy_service ocf:pacemaker:Dummy op monitor
   interval=120s
   ```

In the previous example, *dummy_service* is the name that is provided for the service for this resource:

To invoke the Dummy resource agent, a notation (`ocf:pacemaker:Dummy`) is used to specify that it conforms to the OCF standard, that it runs in the pacemaker namespace, and that the Dummy script is used. If you were configuring a heartbeat monitor service for an Oracle Database, you might use the `ocf:heartbeat:oracle` resource agent.

The resource is configured to use the monitor operation in the agent and an interval is set to check the health of the service. In this example, the interval is set to 120s to give the service sufficient time to fail while you're demonstrating failover. By default, this interval is typically set to 20 seconds, but it can be modified depending on the type of service and the particular environment.

When you create a service, the cluster starts the resource on a node by using the resource agent's start command.

2. View the resource start and run status, for example:

```
sudo pcs status
```

The following output is displayed:

```
Cluster name: pacemaker1
Stack: corosync
Current DC: node2 (version 2.1.2-4.0.2.el9-f765c3be2f4) - partition
with quorum
Last updated:Mon Jul 18 14:54:28 2022
Last change: Mon Jul 18 14:52:28 2022 by root via cibadmin on node1

2 nodes configured
1 resource configured

Online: [ node1 node2 ]

Full list of resources:

 dummy_service  (ocf::pacemaker:Dummy): Started node1

Daemon Status:
  corosync: active/disabled
  pacemaker: active/disabled
  pcsd: active/enabled
```

3. Run the `crm_resource` command to simulate service failure by force stopping the service directly:

```
sudo crm_resource --resource dummy_service --force-stop
```

Running the `crm_resource` command ensures that the cluster is unaware that the service has been manually stopped.

4. Run the `crm_mon` command in interactive mode so that you can wait until a node fails, to view the `Failed Actions` message, for example:

   ```
   sudo crm_mon
   ```

   The following output is displayed:

   ```
   Stack: corosync
   Current DC: node1 (version 2.1.2-4.0.2.el9-f765c3be2f4) - partition with
   quorum
   Last updated: Mon Jul 18 15:00:28 2022
   Last change: Mon Jul 18 14:58:14 2022 by root via cibadmin on node1

   3 nodes configured
   1 resource configured

   Online: [ node1 node2 ]

   Active resources:

   dummy_service   (ocf::pacemaker:Dummy): Started node2

   Failed Resource Actions:
   * dummy_service_monitor_120000 on node1 'not running' (7): call=7,
   status=complete, exitreason='',
       last-rc-change='Mon Jul 18 15:00:17 2022', queued=0ms, exec=0ms
   ```

   You can see the service restart on the alternate node. Note that the default monitor period is set to 120 seconds, so you might need to wait up to the full period before you see notification that a node has gone offline.

   > **Tip:**
   >
   > You can use the `Ctrl-C` key combination to exit out of `crm_mon` at any point.

5. Reboot the node where the service is running to determine whether failover also occurs in the case of node failure.

   Note that if you didn't enable the `corosync` and `pacemaker` services to start on boot, you might need to manually start the services on the node that you rebooted by running the following command:

   ```
   sudo pcs cluster start node1
   ```

4

Configuring Pacemaker Resources and Resource Groups

Pacemaker enables you to create groups of resources, such as database and file system services, so that they start in the order you specify and stop in the reverse of that order. Adding resources to the same resource group also ensures that they run on the same node.

About Pacemaker Resources and Resource Groups

A Pacemaker resource, at its most basic, is a service managed by Pacemaker. Example of resources include services for the following:

- IP address
- File system
- Website
- Database

The resources in the preceding list are known as **primitive** resources. Primitive resources can be grouped and cloned into more complex resources called **groups** and **clones**.

Resource Agents

Each primitive resource is managed by a **resource agent** script that provides Pacemaker with access to the resource through a standardized interface. Some of the standards compatible with Pacemaker are shown in the following list:

- Open Cluster Framework (OCF)
- Systemd
- System Services
- STONITH
- Linux Standard Base (LSB)

> **WARNING:**
>
> LSB scripts do not always comply with the standard. See https://clusterlabs.org/ for more information.

To see the resource agents on your system, you can run the `pcs resource agents` command as shown in the following example (lots of lines have been omitted from the sample output for brevity):

```
pcs resource agents


.
.
.
apache
.
.
.
IPaddr
IPaddr2
.
.
.
oracle
oralsnr
pacemaker
pacemaker
.
.
.
pcsd
pcsd
.
.
.
```

To see the interface standards on your system you can run the `pcs resource standards` command as shown in the following example:

```
pcs resource standards


lsb
ocf
service
systemd
```

Resource Properties

Resource properties determine which resource agent manages the resource, where to find that resource agent and which standards it conforms to. The following list describes the properties of a primitive resource:

id
This is the resource name of your choice.

class
The standard the resource agent conforms to.
For example: `ocf`, `stonith`, or `systemd`.

description
Description of the resource agent.

type
The type of the resource agent you need to use.
For example, `IPaddr2`, `Filesystem`, or `Website`.

provider
With OCF agents you can specify the provider to use for a resource (the OCF spec allows multiple vendors to supply the same resource agent).
Example values: `heartbeat`, `pacemaker`.

Creating Resources

You can create a resource by using the `pcs resource create` command as shown in the following example that creates a resource for a virtual IP address:

```
sudo pcs resource create MyVirtualIP ocf:heartbeat:IPaddr2 ip=192.0.2.3 \
cidr_netmask=24 nic=eth1 \
op monitor interval=1s --group myapachegroup
```

In the preceding example:

- *MyVirtualIP* is the **id**, or name of the resource.
- `ocf` is the **class**, or the standard, the resource agent conforms to.
- `heartbeat` is the **provider** of the resource.
- `IPaddr2` is the type of resource that is to be created.

  The following options are specified for the `IPaddr2` resource:

  - `ip=`*192.0.2.3* specifies IP address to use for the Virtual IP Address.
  - `cidr_netmask=`*24* specifies the netmask for the interface in CIDR format.
  - `nic=`*eth1* The base network interface on which the IP address will be brought online.

  The following operation is specified for the `IPaddr2` resource:

  - `op monitor interval=`*1s* specifies that the system checks every second whether the resource is running.
- `--group`*myapachegroup* specifies the resource group to which the resource is to be added. If the group does not exist, it is created.

Following on from the preceding example, you can create and add a second resource, this time of type `apache`, as shown in the following example:

```
sudo pcs resource create MyApacheWebsite ocf:heartbeat:apache \
configfile="/etc/httpd/conf/httpd.conf" \
statusurl="http://192.0.2.3/server-status" --group myapachegroup
```

You can confirm the status of your resources by running the `pcs status` command. Depending upon your configuration, you r command output will be similar to the following sample code block:

```
sudo pcs status
Cluster name: my_cluster
Cluster Summary:
* Stack: corosync (Pacemaker is running)
* Current DC: node1 (version 2.1.6-9.1.0.1.el8_9-
6fdc9deea29) - partition with quorum
* Last updated: Sat Feb 3 22:35:58 2024 on node1
* Last change: Sat Feb 3 22:33:37 2024 by root via
cibadmin on node1
* 2 nodes configured
* 4 resource instances configured
Node List:
* Online: [ node1 ]
* OFFLINE: [ node2 ]
Full List of Resources:
* Resource Group: myapachegroup:
 * MyVirtualIP (ocf::heartbeat:IPaddr2): Started node1
 * MyApacheWebsite (ocf::heartbeat:apache): Started node1
...
...
Daemon Status:
corosync: active/disabled
pacemaker: active/enabled
pcsd: active/enabled
```

Resource Start and Stop Order in a Resource Group

If the resource group in the preceding examples, *myapachegroup*, is started from a stopped status, the resources will be started in the order they were added to the group:

1. *MyVirtualIP* , the first resource to be added to the resource group, will be started first.
2. *MyApacheWebsite*, the second resource to be added to the resource group, will be started second.

Conversely, when the resource group is stopped, the resources will be stopped in the opposite order of that in which they were added to the group:

1. *MyApacheWebsite*, the last resource added to the resource group in the preceding examples, will be stopped first.
2. *MyVirtualIP* , the first resource added to the resource group in the preceding examples, will be stopped last.

You can also explicitly set the start order by using options `--before` or `--after` when creating a resource with the `pcs resource create` command to specify the position of the resource being created relative to a resource that already exists in the group.

The `pcs resource group add` command also enables you to specify resource sequence. See Creating a Resource Group for more information.

Creating a Resource Group

As documented in the preceding sections, using the `pcs resource create` command will create the group specified with the `--group` option if it does not already exist. A second way of creating a resource group is by using the `pcs resource group add` command, as shown in the following example:

```
sudo pcs resource group add mygroup MyVirtualIP MyApacheWebsite
```

The preceding command creates group *mygroup* If it does not already exist, and adds existing resources *MyVirtualIP* and *MyApacheWebsite* to the group. If the resources specified in the command are in another group, they will be moved to the new group *mygroup*.

You can also use options `--before` or `--after` with the `pcs resource group add` command to specify the position of the resource being created relative to a resource that already exists in the group.

See `pcs(8)` and https://clusterlabs.org/ for more information on configuring resources and resource groups.

5
Configuring Fencing (stonith)

This chapter describes how to configure fencing (`stonith`).

About Fencing Configuration (stonith)

Fencing, or `stonith` (shoot the other node in the head), is used to protect data in the event that nodes become unresponsive. If a node fails to respond, it may still be accessing data. To ensure that your data is safe, you can use fencing to prevent a live node from accessing data until the original node is truly offline. To accomplish this task, you must configure a device that can ensure a node is taken offline. There are a number of available fencing agents that can be configured for this purpose. In general, `stonith` relies on particular hardware and service protocols that can force reboot or shutdown nodes physically to protect the cluster.

The following are different configurations that use some available fencing agents. Note that these examples make certain presumptions about hardware and assume that you already know how to set up, configure, and use the affected hardware. The following examples are provided for basic guidance only. It is recommended that you also refer to upstream documentation to familiarize yourself with some of the concepts that are presented in this documentation.

Before proceeding with any of the following configurations, ensure that `stonith` is enabled for your cluster configuration:

```
sudo pcs property set stonith-enabled=true
```

After configuring `stonith`, run the following commands to check your configuration and ensure that it is set up correctly:

```
sudo pcs stonith config
sudo pcs cluster verify --full
```

To check the status of your `stonith` configuration, run the following command:

```
sudo pcs stonith
```

To check the status of your cluster, run the following command:

```
sudo pcs status
```

Fencing Configuration Examples

The following examples describe that various types of fencing configurations that you can implement.

IPMI LAN Fencing

Intelligent Platform Management Interface (IPMI) is an interface to a subsystem that provides management features of the host system's hardware and firmware and includes facilities to power cycle a system over a dedicated network without any requirement to access the system's operating system. You can configure the `fence_ipmilan` fencing agent for the cluster so that `stonith` can be achieved across the IPMI LAN.

If your systems are configured for IPMI, you can run the following commands on one of the nodes in the cluster to enable the `ipmilan` fencing agent and configure `stonith` for both nodes, for example:

```
sudo pcs stonith create ipmilan_n1_fencing fence_ipmilan
pcmk_host_list=node1 delay=5 \
ipaddr=203.0.113.1 login=root passwd=password lanplus=1 op monitor
interval=60s
sudo pcs stonith create ipmilan_n2_fencing fence_ipmilan
pcmk_host_list=node2 \
ipaddr=203.0.113.2 login=root passwd=password lanplus=1 op monitor
interval=60s
```

In the example, *node1* is a host that has an IPMI LAN interface configured on the IP address *203.0.113.1*. The host named *node2* has an IPMI LAN interface that is configured on the IP *203.0.113.2*. The `root` user password for the IPMI login on both systems is specified in this example as *password*. In each instance. You should replace these configuration variables with the appropriate values for your particular environment.

Note that the delay option should only be set to one node. This setting ensures that in the rare case of a fence race condition that only one node is killed and the other continues to run. Without this option set, it is possible that both nodes make the assumption that they are the only surviving node and then simultaneously reset each other.

> **NOT_SUPPORTED:**
>
> The IPMI LAN agent exposes the login credentials of the IPMI subsystem in plain text. Your security policy should ensure that it is acceptable for users with access to the Pacemaker configuration and tools to also have access to these credentials and the underlying subsystems that are involved.

SCSI Fencing

The SCSI Fencing agent is used to provide storage-level fencing. This configuration protects storage resources from being written to by two nodes simultaneously by using SCSI-3 PR (Persistent Reservation). Used in conjunction with a watchdog service, a node can be reset automatically by using `stonith` when it attempts to access the SCSI resource without a reservation.

To configure an environment in this way:

1. Install the watchdog service on *both* nodes and then copy the provided `fence_scsi_check` script to the watchdog configuration before enabling the service, as shown in the following example:

```
sudo dnf install watchdog
sudo cp /usr/share/cluster/fence_scsi_check /etc/watchdog.d/
sudo systemctl enable --now watchdog
```

2. Enable the `iscsid` service that is provided in the `iscsi-initiator-utils` package on both nodes:

```
sudo dnf install -y iscsi-initiator-utils
sudo systemctl enable --now iscsid
```

3. After both nodes are configured with the watchdog service and the `iscsid` service, you can configure the `fence_scsi` fencing agent on one of the cluster nodes to monitor a shared storage device, such as an iSCSI target, for example:

```
sudo pcs stonith create scsi_fencing fence_scsi pcmk_host_list="node1
node2" \
 devices="/dev/sdb" meta provides="unfencing"
```

In the example, *node1* and *node2* represent the hostnames of the nodes in the cluster and */dev/sdb* is the shared storage device. Replace these variables with the appropriate values for your particular environment.

SBD Fencing

The Storage Based Death (SBD) daemon can run on a system and monitor shared storage. The SBD daemon can use a messaging system to track cluster health. SBD can also trigger a reset if the appropriate fencing agent determines that `stonith` should be implemented.

> **Note:**
>
> SBD Fencing is the method used with Oracle Linux HA clusters running on Oracle Cloud Infrastructure, as documented in Create a High Availability Cluster on Oracle Cloud Infrastructure (OCI) .

To set up and configure SBD fencing:

1. Stop the cluster by running the following command on one of the nodes:

```
sudo pcs cluster stop --all
```

2. On each node, install and configure the SBD daemon:

```
sudo dnf install sbd
```

3. Enable the `sbd` systemd service:

   ```
   sudo systemctl enable sbd
   ```

   Note that the `sbd` systemd service is automatically started and stopped as a dependency of the `pacemaker` service, you do not need to run this service independently. Attempting to start or stop the `sbd` systemd service fails and returns an error indicating that it is controlled as a dependency service.

4. Edit the `/etc/sysconfig/sbd` file and set the `SBD_DEVICE` parameter to identify the shared storage device. For example, if your shared storage device is available on */dev/sdc*, make sure the file contains the following line:

   ```
   SBD_DEVICE="/dev/sdc"
   ```

5. On one of the nodes, create the SBD messaging layout on the shared storage device and confirm that it is in place. For example, to set up and verify messaging on the shared storage device at */dev/sdc*, run the following commands:

   ```
   sudo sbd -d /dev/sdc create
   sudo sbd -d /dev/sdc list
   ```

6. Finally, start the cluster and configure the `fence_sbd` fencing agent for the shared storage device. For example, to configure the shared storage device, */dev/sdc*, run the following commands on one of the nodes:

   ```
   sudo pcs cluster start --all
   sudo pcs stonith create sbd_fencing fence_sbd devices=/dev/sdc
   ```

IF-MIB Fencing

IF-MIB fencing takes advantage of SNMP to access the IF-MIB on an Ethernet network switch and to also shutdown the port on the switch, which effectively takes a host offline. This configuration leaves the host running, while disconnecting it from the network. Bear in mind that any FibreChannel or InfiniBand connections could remain intact, even after the Ethernet connection has been stopped, which means that any data made available on these connections could still be at risk. Thus, consider configuring this fencing method as a fallback fencing mechanism. See Configuring Fencing Levels for more information about how to use multiple fencing agents in combination to maximize `stonith` success.

To configure IF-MIB fencing:

1. Configure the switch for SNMP v2c, at minimum, and ensure that SNMP SET messages are enabled. For example, on an Oracle Switch, by using the ILOM CLI, you could run the following commands:

   ```
   sudo set /SP/services/snmp/ sets=enabled
   sudo set /SP/services/snmp/ v2c=enabled
   ```

2. On one of the nodes in the cluster, configure the `fence_ifmib` fencing agent for each node in the environment, as shown in the following example:

```
sudo pcs stonith create ifmib_n1_fencing fence_ifmib pcmk_host_list=node1
\
ipaddr=203.0.113.10 community=private port=1 delay=5 op monitor
interval=60s
sudo pcs stonith create ifmib_n2_fencing fence_ifmib pcmk_host_list=node2
\
ipaddr=203.0.113.10 community=private port=2 op monitor interval=60s
```

In the example, the SNMP IF-MIB switch is accessible at the IP address *203.0.113.10*; the *node1* host is connected to port *1* on the switch, and the *node2* host is connected to port *2* on the switch. Replace these variables with the appropriate values for the particular environment.

Configuring Fencing Levels

If you have configured multiple fencing agents, you may want to set different fencing levels. Fencing levels enable you to prioritize different approaches to fencing and can provide a valuable mechanism for fallback options should your default fencing mechanism fail.

Each fencing level is attempted in ascending order, starting from level 1. If the fencing agent that is configured for a particular level fails, the fencing agent from the next level is then attempted, and so on.

For example, you may wish to configure IPMI-LAN fencing at level 1, but fallback to IF-MIB fencing as a level 2 option. Using the example configurations from Fencing Configuration Examples, you would run the following commands on one of the nodes to set the fencing levels for each configured agent:

```
sudo pcs stonith level add 1 node1 ipmilan_n1_fencing
sudo pcs stonith level add 1 node2ipmilan_n2_fencing
sudo pcs stonith level add 2 node1ifmib_n1_fencing
sudo pcs stonith level add 2 node2ifmib_n2_fencing
```

6
Working With Quorum Devices

A quorum device acts as a third-party arbitrator in the event where standard quorum rules might not adequately cater for node failure. A quorum device is typically used where there may be an even number of nodes in a cluster. For example, in a cluster that contains two nodes failure of the nodes to communicate can result in a split-brain issue where both nodes function as primary at the same time, which results in possible data corruption. By using a quorum device, quorum arbitration can be achieved and a selected node survives.

A quorum device is a service that ideally runs on a separate physical network to the cluster itself. It should run on a system that's not a node in the cluster. Although the quorum device can service multiple clusters at the same time, it should be the only quorum device for each cluster that it serves. Each node in the cluster is configured for the quorum device. The quorum device is installed and run as a network bound service on a system outside of the cluster network.

Installing and Enabling a Quorum Device

Installation of the quorum device requires that you install the `pcs` and `corosync-qnetd` packages on the system where you intend to run the quorum device service and then install the `corosync-qdevice` package on each of the nodes in the existing cluster.

1. On the system assigned to run the quorum device service, run:

   ```
   sudo dnf install -y pcs corosync-qnetd
   ```

2. Enable and start the systemd `pcsd` service by running:

   ```
   sudo systemctl enable --now pcsd
   ```

3. If you're running a firewall on the quorum device service host, you must open the firewall ports to allow the host to communicate with the cluster. For example, run:

   ```
   sudo firewall-cmd --permanent --add-service=high-availability
   sudo systemctl restart firewalld
   ```

4. On the quorum device service host, enable and start the quorum device service by setting the Pacemaker configuration for the node to use the `net` model. Run:

   ```
   sudo pcs qdev setup model net --enable --start
   ```

   This command creates a configuration for the host and names the node `qdev`. It sets the model to `net` and enables and starts the node. The command triggers the `corosync-qnetd` daemon to load and run at boot.

5. On each of the nodes within the existing cluster, install the `corosync-qdevice` package by running:

```
sudo dnf install -y corosync-qdevice
```

Configuring the Cluster for a Quorum Device

The node running the quorum device service must be authenticated to the rest of the cluster and must then be added to the cluster. When you add the quorum device service node, you can set configuration options such as which algorithm to use to determine quorum. After the quorum device is added to the cluster you can verify the quorum device status to check that the device is functioning correctly.

1. Authenticate the quorum device service node to the cluster. On a node within the existing cluster, to authenticate the node named *qdev*, run:

```
sudo pcs host auth qdev
```

You're prompted for the cluster username and password.

2. Check that no quorum device is already configured for the cluster. A cluster must never have more than one quorum device configured. On a node within the existing cluster, run:

```
sudo pcs quorum status
```

Note that the output includes membership information:

```
...
Membership information
----------------------
    Nodeid      Votes    Qdevice Name
         1          1         NR node1 (local)
         2          1         NR node2
```

Under the Qdevice column, the value NR is displayed. The NR value indicates that no quorum devices are registered with any of the nodes within the cluster. If any other value is displayed, don't proceed with adding another quorum device to the cluster without removing the existing device first.

3. Add the quorum device to the cluster. On one of the nodes within the existing cluster, run:

```
sudo pcs quorum device add model net host=qdev algorithm=ffsplit
```

Note that you specify the host to match the host where you're running the quorum device service, in this case named *qdev*; and the algorithm that you want to use to determine quorum, in this case *ffsplit*.
Algorithm options are:

- `ffsplit`: is a fifty-fifty split algorithm that favors the partition with the highest number of active nodes in the cluster.

- `lms`: is a last-man-standing algorithm that returns a vote for the nodes that are still able to connect to the quorum device service node. If a single node is still active and it can connect to the quorum device service, the cluster remains quorate. If none of the nodes can connect to the quorum device service and any one node loses connection with the rest of the cluster, the cluster becomes inquorate.

See the `corosync-qdevice(8)` manual page for more information.

4. Verify that the quorum device is configured within the cluster. On any node in the existing cluster, run:

```
sudo pcs quorum config
```

The output displays that a quorum device is configured and indicates the algorithm that is in use:

```
Options:
Device:
  Model: net
    algorithm: ffsplit
    host: qdev
```

You can also query the quorum status for the cluster by running:

```
sudo pcs quorum status
```

The output displays the quorum status.

```
Quorum information
------------------
Date:             Fri Jul 15 14:19:07 2022
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
```

Note that the membership information now displays values A,V,NMW for the Qdevice field. Values for this field can be equal to any of the following:

- `A/NA`: indicates that the quorum device is alive or not alive to each node in the cluster.
- `V/NV`: indicates whether the quorum device has provided a vote to a node. In the case where the cluster is split, one node would be set to V and the other to NV.
- `MW/NMW`: indicates whether the quorum device `master_wins` flag is set. Any node with an active quorum device that also has the `master_wins` flag set becomes quorate regardless of the node votes of the cluster. By default the option is unset.

Managing Quorum Devices

The quorum device service must be managed from the host system where the quorum device service is running.

Quorum configuration for the cluster and the configuration of the quorum device on the cluster nodes is performed by running operations on any of the nodes within the cluster itself.

Controlling the Quorum Device Service

You can perform various operations to directly control the quorum device service. Commands that control the quorum device service must be run on the host where the quorum device service is running.

- To view the full status for the service, run:

```
sudo pcs qdevice status net --full
```

Output similar to the following is displayed:

```
QNetd address:                  *:5403
TLS:                            Supported (client certificate
required)
Connected clients:              2
Connected clusters:             1
Maximum send/receive size:      32768/32768 bytes
Cluster "test":
    Algorithm:          ffsplit
    Tie-breaker:        Node with lowest node ID
    Node ID 2:
        Client address:         ::ffff:192.168.2.25:33526
        HB interval:            8000ms
        Configured node list:   1, 2
        Ring ID:                1.16
        Membership node list:   1, 2
        TLS active:             Yes (client certificate verified)
        Vote:                   ACK (ACK)
    Node ID 1:
        Client address:         ::ffff:192.168.2.26:48786
```

```
HB interval:            8000ms
Configured node list:   1, 2
Ring ID:                1.16
Membership node list:   1, 2
TLS active:             Yes (client certificate verified)
Vote:                   ACK (ACK)
```

- To start the service, run:

  ```
  sudo pcs qdevice start net
  ```

- To stop the service, run:

  ```
  sudo pcs qdevice stop net
  ```

- To enable the service so that it runs at boot time, run:

  ```
  sudo pcs qdevice enable net
  ```

- To disable the service to prevent it from restarting at boot, run:

  ```
  sudo pcs qdevice disable net
  ```

- To force the service to stop if the normal stop process is not working, run:

  ```
  sudo pcs qdevice kill net
  ```

Updating Quorum Device Settings

The quorum device can be updated in the cluster configuration at any time. Modifications to the quorum device configuration must be performed on a node within the cluster. Typically modifications to the quorum device involve changing the algorithm, however you can modify other options that are available for a quorum device in the same way.

To update the algorithm used for the quorum device, run:

```
sudo pcs quorum device update model algorithm=lms
```

The example changes the algorithm to use the `lms` or last-man-standing algorithm.

> **Note:**
>
> You can't update the host for a quorum device. You must remove the device and add it back into the cluster if you need to change the host.

Removing the Quorum Device From the Cluster

To remove the quorum device from the cluster, run the following command on a node within the cluster:

```
sudo pcs quorum device remove
```

Removing the quorum device updates the cluster configuration to remove any configuration entries for the quorum device, reloads the cluster configuration into the cluster and then disables and stops the quorum device on each node.

Because you might use the same quorum device service across multiple clusters, removing the quorum device from the cluster doesn't affect the quorum device service in any way. The service continues to run on the service host, but no longer serves the cluster where it has been removed.

Destroying the Quorum Device Service

You can destroy the quorum device service on the host where the service is running. This action stops the service and removes any configuration for the service from the host.

```
sudo pcs qdevice destroy net
```

> **Note:**
>
> Remove the quorum device from any clusters that it services before destroying the quorum device service.

7

Using the Pacemaker/Corosync Web User Interface

This chapter describes how to create and manage clusters by using the web UI tool instead of the `pcs` command line.

About the Pacemaker Web User Interface

The Pacemaker service provides a web user interface tool (web UI) that enables you to configure and manage clusters in graphical mode. Use this tool as an alternative to typing `pcs` commands to perform those tasks.

This chapter assumes that you have completed the tasks that are described in Installing and Enabling the Pacemaker and Corosync Service and that the nodes have been authenticated for the `hacluster` user.

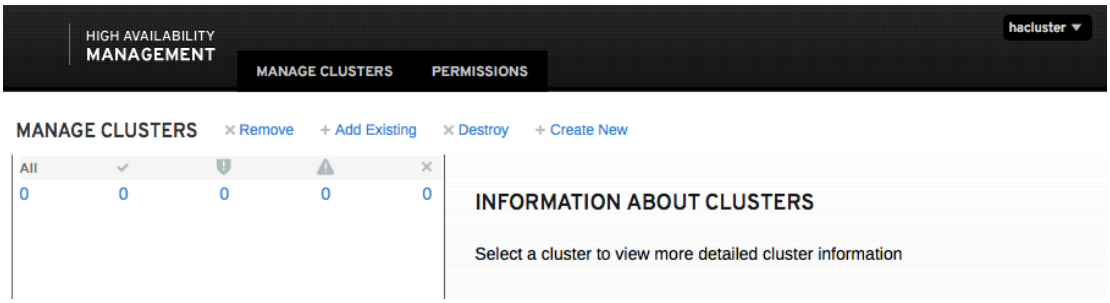
For information about authentication and configuring `hacluster` credentials, see Step 1 of Creating the Cluster.

To access the web UI, log in as user `hacluster` at `https://node:2224`, where *node* refers to a node authenticated for `hacluster`. Specify the node either by its node name or IP address.

> **Note:**
>
> The rest of this chapter assumes that you have configured resolvable names for all the nodes.

After you log in, the home page's Manage Clusters page is displayed. This page lists clusters that are under the web UI's management.

Figure 7-1 Manage Clusters



Initial Cluster Configuration Tasks

The Manage Clusters page contains the following options:

- `Create New`: Create a cluster consisting of nodes. While creating the cluster, you can optionally configure additional cluster properties.
- `Add Existing`: Add more nodes to an existing cluster.
- `Remove`: Remove a cluster from being managed by the web UI.
- `Destroy`: Delete a cluster.

Creating a New Cluster

To create new clusters, do the following:

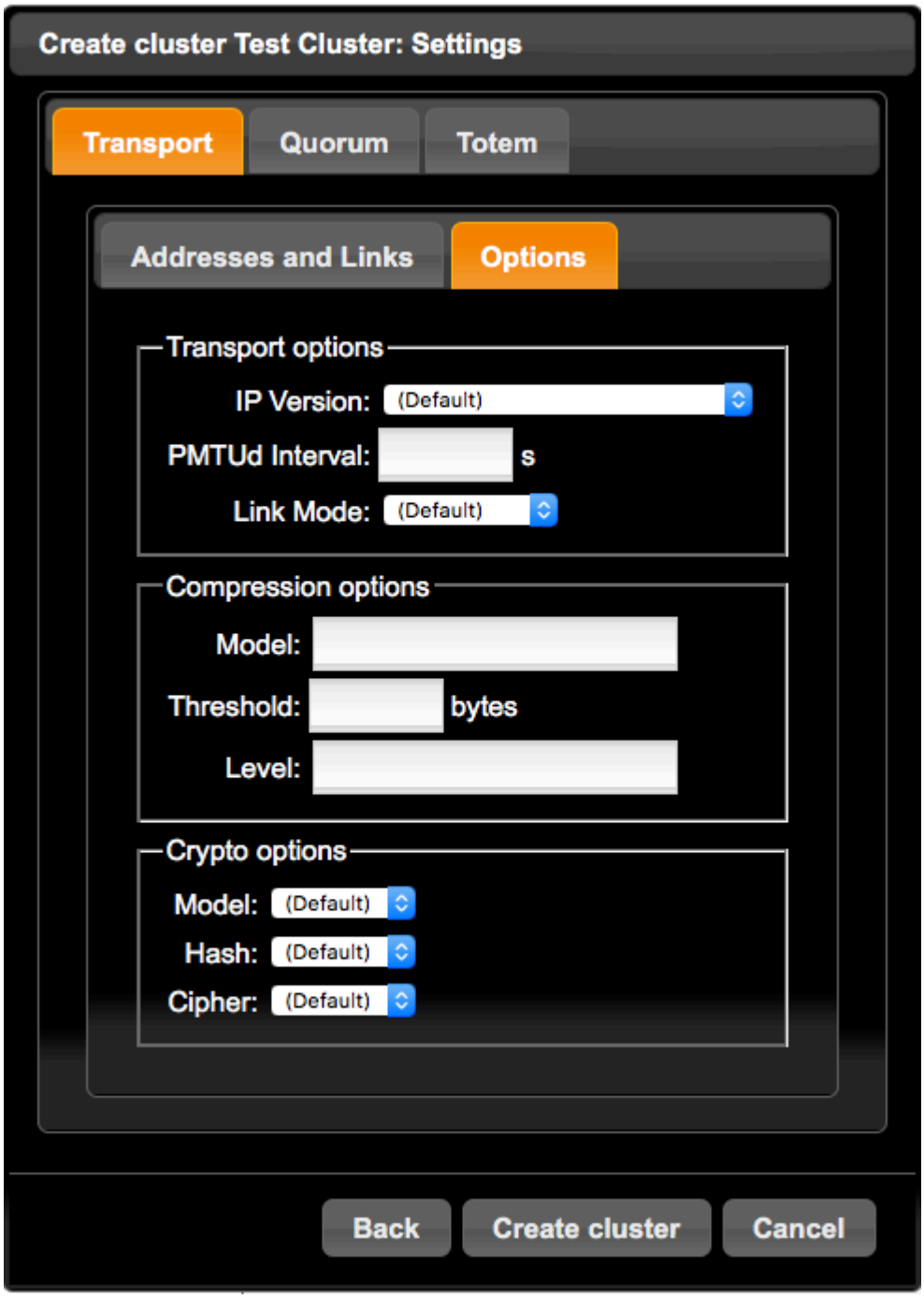
1. Click **+ Create New**.

   The Create cluster window opens.
2. Specify the cluster name.
3. Specify the nodes to include in the cluster.
4. Optional: Configure other properties of the cluster.
   a. Click **Go to advanced settings**.
   b. Click the tab for the setting you want to customize.

      You can configure the cluster's Transport, Quorum, or Totem settings.
   c. For each setting, specify parameter values on their corresponding fields.

      For example, for the Transport mechanism, you can define transport, compression, and crypto options, as shown in the following image:

Figure 7-2 Transport Options



5. Click **Create Cluster**.
6. Click **Finish**.

   If you don't want to start the cluster, clear **Start the Cluster** first before clicking **Finish**.

The new cluster is listed on the page. Selecting it displays cluster information such as its nodes and any configured resources and fences.

Adding an Existing Cluster

This option enables you to add nodes to an existing cluster.

1. From the list, select the cluster to which you want to add an existing node.

   If the cluster is unlisted, then you would need to create the cluster first.
2. Click **+ Add Existing**.
3. Specify the node that you want to add.

   You can add only one node at a time.
4. Click **Add Existing**.

> **Note:**
>
> The web UI provides another method of adding nodes to a cluster. See Configuring Nodes.

Removing or Destroying Clusters

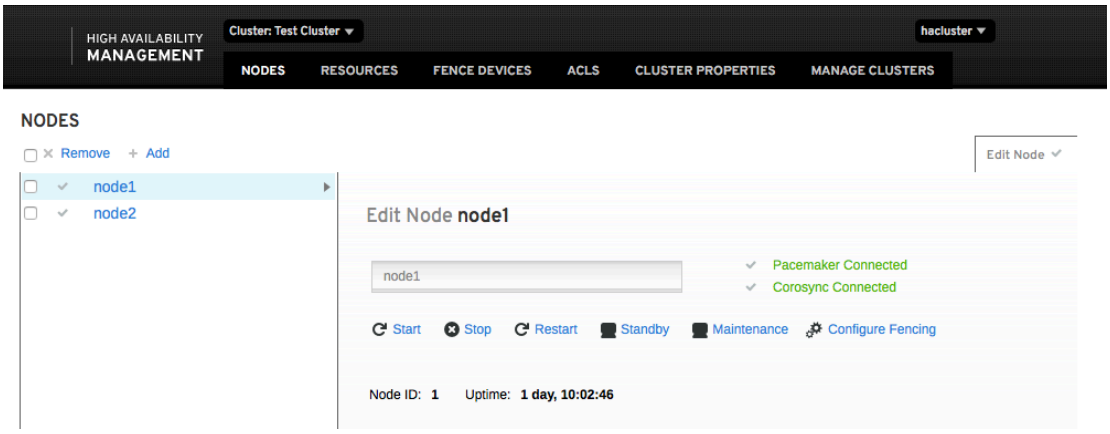
The remaining options on the Manage Cluster page are removing clusters or destroying them.

- Removing a cluster means disabling it from being managed by the web UI. The cluster continues to run. However, it can only be administered through the command line.
- Destroying a cluster means deleting the cluster and all its associated configuration files. Its constituent nodes thus become independent units. Destroying a cluster is irreversible.

Managing Clusters With the Web UI

To further customize existing clusters, from the list of clusters that is displayed, click the cluster name. A new window opens where additional menu options enable you to configure the cluster's nodes, resources, fence devices, access control lists, and cluster properties. By default, the Nodes page is displayed.

Figure 7-3 Nodes



Configuring Nodes

The Nodes page contains options to add nodes to the cluster or remove nodes.

- To add nodes:
  1. Click **+ Add**.
  2. Specify the nodes to add.
  3. Click **Add Node**.
- To remove nodes:
  1. Select one or more nodes from the list.
  2. Click **x Remove**.
  3. Click **Remove Node(s)** to confirm.

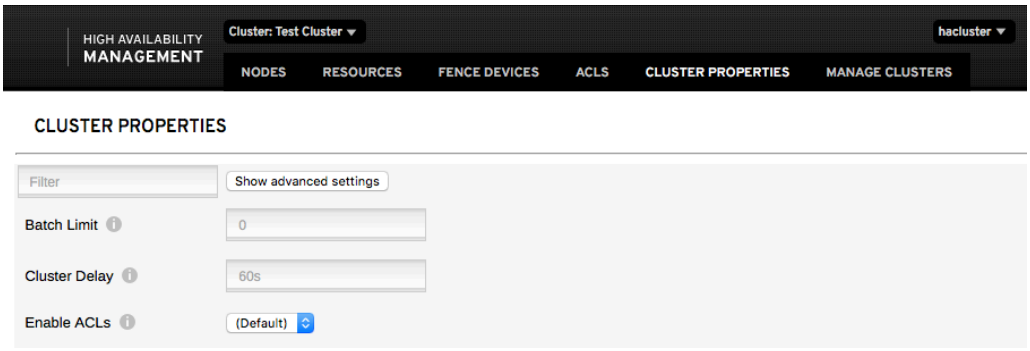
For every node that you select from the list, information about that node is displayed, including the status of the cluster daemons running on the node, resource information, node attributes, and so on. You can manipulate the node further by clicking the options that correspond to the following actions:

- Stop, start, or restart the node.
- Put the node on standby mode.
- Put the node on maintenance mode.

Configuring Additional Cluster Properties

To configure more properties of a selected cluster, open the Cluster Properties page. The page displays a list of basic properties for which you specify values on their corresponding fields. To configure properties other than the basic ones, click **Show advanced settings**.

Figure 7-4 Cluster Properties



To obtain information about a specific property, hover the mouse pointer over the information icon (***i***). The icon displays a short description of the property and its default value.

For example, the **Batch Limit** property is described as follows:

```
The number of jobs that the TE is allowed to
execute in parallel.

The "correct" value will depend on the speed
and load of your network and cluster nodes.

Default value: 0
```

The properties you customize depend on circumstances and needs. Suppose that you have a two-node cluster. For this cluster, you want to disable the fencing feature. Because the cluster consists only of two nodes, you do not need any quorum policy. Finally you want to set the migration threshold such that the cluster moves services to a new node after a single failure on a current node. In this case, you would do the following:

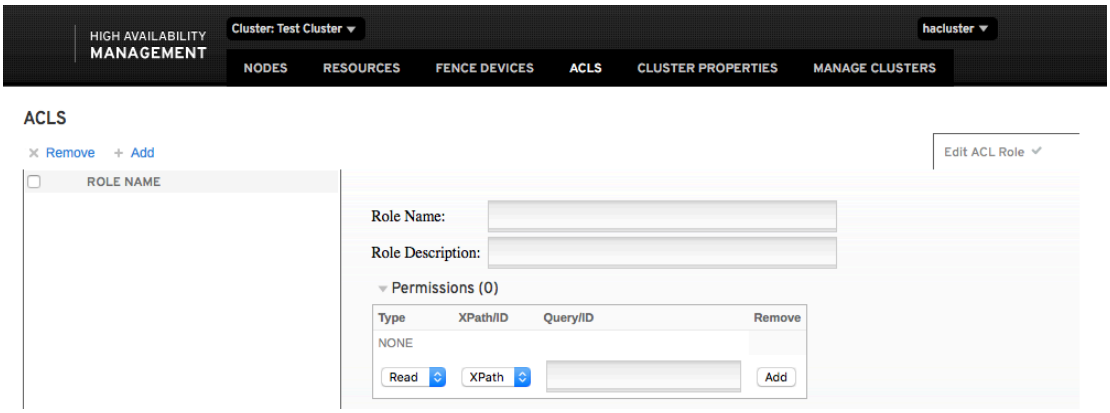
1. From the drop down list of **Stonith Enabled**, select **false** to disable the fencing feature.
2. From the drop down list of **No Quorum Policy**, select **ignore** to disregard the quorum policy
3. Click **Show advanced settings** to display migration parameters.
4. On the **migration limit** field, type `1` to set the threshold to a single failure event before services are moved.
5. Click **Apply Changes** to accept the revisions.
6. Click **Refresh** so that the page reflects the changed parameters with their new values.

Typically, you can configure multiple properties in any random order. However, as a final step, you must click **Apply Changes** to effect the new configuration.

Setting Local Permissions Through ACLs

Access control lists (ACLs) on the ACLS page are a way of regulating access to a specific cluster so that local users are granted only the permissions they need in the cluster to perform their tasks.

Figure 7-5 ACLs



Creating ACLs for the cluster assumes that you have already created users and optionally have added them to defined groups on all the cluster nodes, for example:

```
sudo adduser user
sudo usermod -a -G groupuser
```

To define local permissions, do the following:

1. Click **+ Add**.

   The Add ACL Role window opens.
2. Provide a role name and description.
3. Click **Add Role**.
4. On the ACLS page, select the permission to be assigned to the role.

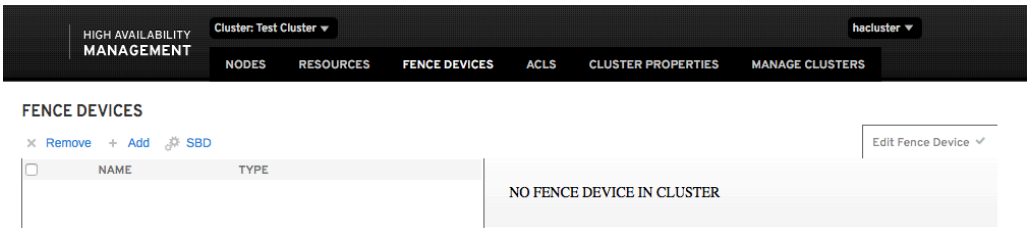
   You can grant three types of permissions: **read-only**, **write-only**, or **deny**. **Deny** takes precedence over the other permissions.
5. Specify users who can assume the role and click **Add**.
6. Optionally, specify a group whose members can assume the role and click **Add**.
7. Click the **Cluster Properties** menu item.
8. From the pull-down list of the **Enable ACLs** property, select **true**.

   If you omit this step, the new role remains configured but deactivated.

Configuring Fencing

To configure fencing for the cluster, click the appropriate menu item to open the Fence Devices page.

Figure 7-6 Fence Devices



For a brief description of fencing and its purpose, see About Fencing Configuration (stonith).

Creating Fence Devices

The web UI enables you to configure different kinds of fencing configuration. The configuration options that become available depend on the fencing type you create.

Because of the multiplicity of fencing types, the following steps show you how to configure specifically an IPMI LAN fencing and is based on the example in IPMI LAN Fencing. These steps aim to be a guide for creating other types of fencing:

1. If necessary, check that the **Stonith Enabled** property on the Cluster Properties page is set to **(Default)** or **true**.
2. On the Fence Devices page, click **+ Add**.

   The Add Fence Device window opens.
3. From the **Type** pull-down list, select `fence_ipmilan`.

   You can view a brief description of your selected fencing type through the information icon (***i***).
4. Provide a name for the fence configuration, for example, `ipmilan_n1_fencing`.

   You can configure additional arguments later after creating the fence instance.
5. Click **Create Fence Instance**.
6. Repeat the procedure to create the fence device for the next node, such as `node2`.

The fence device is listed on the page and information about it is displayed.

Configuring Fence Device Arguments

The fence device information includes expandable lists for Optional and Advanced Arguments. For the sample IPMI LAN fence devices you just created, you can define new values for the following arguments:

- **pcmk_host_list**, for example, `node1`
- **delay**, for example, `5` seconds
- **ip** for `node1`'s IP address
- **password** for the administrator password

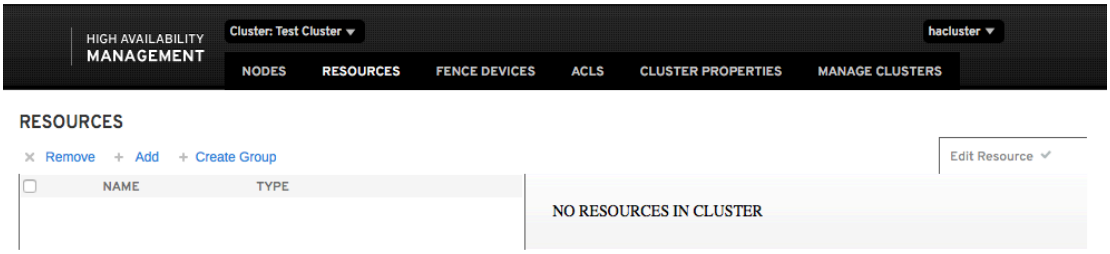
- **lanplus**, for example, `1` as the level of priority over other fencing types for actions that take effect in case of failure.
- **pcmk_monitor_timeout**, for example, `60` seconds

As with all other properties, information about each argument can be obtained through the information icon.

Adding Resources to the Cluster

You add resources and services to for the cluster's operations through the Resources page.

Figure 7-7 Resources



Add resources to the cluster as follows:

1. Click **+ Add**.

   The Add Resource window opens.
2. Select a class or provider from the pull-down list.

   The selection determines which types of resources are available to add.
3. Select the resource type from the pull-down list.

   The resource type you select automatically creates a description of the selection. For example, the **apache** service is described as *Manage an Apache web server*. More information about the type is viewable through the information icon.
4. Required: Specify a valid resource ID.
5. Optionally, configure the other listed parameters.

   You can configure Optional and Advanced Arguments later.
6. Click **Create Resource**.

The created resource is added to the list.

If you have multiple resources on the list, you can assign resources to belong to a group as follows:

1. Select resources by clicking their associated boxes.
2. Click **+ Create Group**.
3. Provide a name for the group.
4. Click **+ Create Group**.

On the resource's information detail, you can manage the resource further through the following options:

- Enforce actions on the resource such as enabling, disabling, refreshing, or removing the resource; performing resource cleanups; and putting the resource in manage or unmanage mode.
- Create a clone or a promotable clone.
- Update the resource's group information, such as assigning it to another group.
- Configure optional and advanced arguments.

8
More Information

For more information and documentation on Pacemaker and Corosync, see https://clusterlabs.org/pacemaker/doc/.