SUSE Linux Enterprise High Availability 15 SP5

Administration Guide

Administration Guide

SUSE Linux Enterprise High Availability 15 SP5

This guide is intended for administrators who need to set up, configure, and maintain clusters with SUSE® Linux Enterprise High Availability. For quick and efficient configuration and administration, the product includes both a graphical user interface and a command line interface (CLI). For performing key tasks, both approaches are covered in this guide. Thus, you can choose the appropriate tool that matches your needs.

Contents

Preface

1 Available documentation

Online documentation

Our documentation is available online at https://documentation.suse.com↗. Browse or download the documentation in various formats.

> **Note: Latest updates**
>
> The latest updates are usually available in the English-language version of this documentation.

SUSE Knowledgebase

If you have run into an issue, also check out the Technical Information Documents (TIDs) that are available online at https://www.suse.com/support/kb/↗. Search the SUSE Knowledgebase for known solutions driven by customer need.

Release notes

For release notes, see https://www.suse.com/releasenotes/↗.

In your system

For offline use, the release notes are also available under `/usr/share/doc/release-notes` on your system. The documentation for individual packages is available at `/usr/share/doc/packages`.

Many commands are also described in their *manual pages*. To view them, run `man`, followed by a specific command name. If the `man` command is not installed on your system, install it with `sudo zypper install man`.

2 Improving the documentation

Your feedback and contributions to this documentation are welcome. The following channels for giving feedback are available:

Service requests and support

For services and support options available for your product, see https://www.suse.com/support/ ↗.

To open a service request, you need a SUSE subscription registered at SUSE Customer Center. Go to https://scc.suse.com/support/requests ↗, log in, and click *Create New*.

Bug reports

Report issues with the documentation at https://bugzilla.suse.com/ ↗.

To simplify this process, click the *Report an issue* icon next to a headline in the HTML version of this document. This preselects the right product and category in Bugzilla and adds a link to the current section. You can start typing your bug report right away.

A Bugzilla account is required.

Contributions

To contribute to this documentation, click the *Edit source document* icon next to a headline in the HTML version of this document. This will take you to the source code on GitHub, where you can open a pull request.

A GitHub account is required.

> ◉ Note: *Edit source document* only available for English
>
> The *Edit source document* icons are only available for the English version of each document. For all other languages, use the *Report an issue* icons instead.

For more information about the documentation environment used for this documentation, see the repository's README.

Mail

You can also report errors and send feedback concerning the documentation to `doc-team@suse.com`. Include the document title, the product version, and the publication date of the document. Additionally, include the relevant section number and title (or provide the URL) and provide a concise description of the problem.

3 Documentation conventions

The following notices and typographic conventions are used in this document:

- `/etc/passwd`: Directory names and file names

- *PLACEHOLDER*: Replace *PLACEHOLDER* with the actual value

- `PATH`: An environment variable

- **ls**, `--help`: Commands, options, and parameters

- `user`: The name of user or group

- `package_name`: The name of a software package

- `Alt`, `Alt`–`F1`: A key to press or a key combination. Keys are shown in uppercase as on a keyboard.

- *File*, *File* › *Save As*: menu items, buttons

- `AMD/Intel` ⟩ This paragraph is only relevant for the AMD64/Intel 64 architectures. The arrows mark the beginning and the end of the text block. ◁
  `IBM Z, POWER` ⟩ This paragraph is only relevant for the architectures `IBM Z` and `POWER`. The arrows mark the beginning and the end of the text block. ◁

- *Chapter 1, "Example chapter"*: A cross-reference to another chapter in this guide.

- Commands that must be run with `root` privileges. You can also prefix these commands with the **sudo** command to run them as a non-privileged user:

  ```
  # command
  > sudo command
  ```

- Commands that can be run by non-privileged users:

  ```
  > command
  ```

- Commands can be split into two or multiple lines by a backslash character (`\`) at the end of a line. The backslash informs the shell that the command invocation will continue after the line's end:

  ```
  > echo a b \
  c d
  ```

- A code block that shows both the command (preceded by a prompt) and the respective output returned by the shell:

```
> command
output
```

- Commands executed in the interactive crm shell.

```
crm(live)#
```

- Notices

> ✋ **Warning: Warning notice**
>
> Vital information you must be aware of before proceeding. Warns you about security issues, potential loss of data, damage to hardware, or physical hazards.

> ❗ **Important: Important notice**
>
> Important information you should be aware of before proceeding.

> 📝 **Note: Note notice**
>
> Additional information, for example about differences in software versions.

> 💡 **Tip: Tip notice**
>
> Helpful information, like a guideline or a piece of practical advice.

- Compact Notices

> 📝 Additional information, for example about differences in software versions.

> 💡 Helpful information, like a guideline or a piece of practical advice.

For an overview of naming conventions for cluster nodes and names, resources, and constraints, see *Appendix B, Naming conventions*.

4 Support

Find the support statement for SUSE Linux Enterprise High Availability and general information about technology previews below. For details about the product lifecycle, see https:// www.suse.com/lifecycle ↗.

If you are entitled to support, find details on how to collect information for a support ticket at https://documentation.suse.com/sles-15/html/SLES-all/cha-adm-support.html ↗.

4.1 Support statement for SUSE Linux Enterprise High Availability

To receive support, you need an appropriate subscription with SUSE. To view the specific support offers available to you, go to https://www.suse.com/support/ ↗ and select your product.

The support levels are defined as follows:

L1

Problem determination, which means technical support designed to provide compatibility information, usage support, ongoing maintenance, information gathering and basic troubleshooting using available documentation.

L2

Problem isolation, which means technical support designed to analyze data, reproduce customer problems, isolate a problem area and provide a resolution for problems not resolved by Level 1 or prepare for Level 3.

L3

Problem resolution, which means technical support designed to resolve problems by engaging engineering to resolve product defects which have been identified by Level 2 Support.

For contracted customers and partners, SUSE Linux Enterprise High Availability is delivered with L3 support for all packages, except for the following:

- Technology previews.

- Sound, graphics, fonts, and artwork.

- Packages that require an additional customer contract.

- Some packages shipped as part of the module *Workstation Extension* are L2-supported only.

- Packages with names ending in `-devel` (containing header files and similar developer resources) will only be supported together with their main packages.

SUSE will only support the usage of original packages. That is, packages that are unchanged and not recompiled.

4.2 Technology previews

Technology previews are packages, stacks, or features delivered by SUSE to provide glimpses into upcoming innovations. Technology previews are included for your convenience to give you a chance to test new technologies within your environment. We would appreciate your feedback. If you test a technology preview, please contact your SUSE representative and let them know about your experience and use cases. Your input is helpful for future development.

Technology previews have the following limitations:

- Technology previews are still in development. Therefore, they may be functionally incomplete, unstable, or otherwise *not* suitable for production use.

- Technology previews are *not* supported.

- Technology previews may only be available for specific hardware architectures.

- Details and functionality of technology previews are subject to change. As a result, upgrading to subsequent releases of a technology preview may be impossible and require a fresh installation.

- SUSE may discover that a preview does not meet customer or market needs, or does not comply with enterprise standards. Technology previews can be removed from a product at any time. SUSE does not commit to providing a supported version of such technologies in the future.

For an overview of technology previews shipped with your product, see the release notes at https://www.suse.com/releasenotes ↗.

I Installation and setup

1 Product overview

SUSE® Linux Enterprise High Availability is an integrated suite of open source clustering technologies. It enables you to implement highly available physical and virtual Linux clusters, and to eliminate single points of failure. It ensures the high availability and manageability of critical network resources including data, applications, and services. Thus, it helps you maintain business continuity, protect data integrity, and reduce unplanned downtime for your mission-critical Linux workloads.

It ships with essential monitoring, messaging, and cluster resource management functionality (supporting failover, failback, and migration (load balancing) of individually managed cluster resources).

This chapter introduces the main product features and benefits of SUSE Linux Enterprise High Availability. Inside you will find several example clusters and learn about the components making up a cluster. The last section provides an overview of the architecture, describing the individual architecture layers and processes within the cluster.

For explanations of some common terms used in the context of High Availability clusters, refer to *Glossary*.

1.1 Availability as a module or extension

High Availability is available as a module or extension for several products. For details, see https://documentation.suse.com/sles/html/SLES-all/article-modules.html#art-modules-high-availability ↗.

1.2 Key features

SUSE® Linux Enterprise High Availability helps you ensure and manage the availability of your network resources. The following sections highlight some key features:

1.2.1 Wide range of clustering scenarios

SUSE Linux Enterprise High Availability supports the following scenarios:

- Active/active configurations

- Active/passive configurations: $N + 1$, $N + M$, N to 1, N to M

- Hybrid physical and virtual clusters, allowing virtual servers to be clustered with physical servers. This improves service availability and resource usage.

- Local clusters

- Metro clusters ("stretched" local clusters)

- Geo clusters (geographically dispersed clusters)

> **❗ Important: No support for mixed architectures**
>
> All nodes belonging to a cluster should have the same processor platform: x86, IBM Z, or POWER. Clusters of mixed architectures are *not* supported.

Your cluster can contain up to 32 Linux servers. Using pacemaker_remote, the cluster can be extended to include additional Linux servers beyond this limit. Any server in the cluster can restart resources (applications, services, IP addresses, and file systems) from a failed server in the cluster.

1.2.2 Flexibility

SUSE Linux Enterprise High Availability ships with Corosync messaging and membership layer and Pacemaker Cluster Resource Manager. Using Pacemaker, administrators can continually monitor the health and status of their resources, and manage dependencies. They can automatically stop and start services based on highly configurable rules and policies. SUSE Linux Enter-

prise High Availability allows you to tailor a cluster to the specific applications and hardware infrastructure that fit your organization. Time-dependent configuration enables services to automatically migrate back to repaired nodes at specified times.

1.2.3 Storage and data replication

With SUSE Linux Enterprise High Availability you can dynamically assign and reassign server storage as needed. It supports Fibre Channel or iSCSI storage area networks (SANs). Shared disk systems are also supported, but they are not a requirement. SUSE Linux Enterprise High Availability also comes with a cluster-aware file system (OCFS2) and the cluster Logical Volume Manager (Cluster LVM). For replication of your data, use DRBD* to mirror the data of a High Availability service from the active node of a cluster to its standby node. Furthermore, SUSE Linux Enterprise High Availability also supports CTDB (Cluster Trivial Database), a technology for Samba clustering.

1.2.4 Support for virtualized environments

SUSE Linux Enterprise High Availability supports the mixed clustering of both physical and virtual Linux servers. SUSE Linux Enterprise Server 15 SP5 ships with Xen, an open source virtualization hypervisor, and with KVM (Kernel-based Virtual Machine). KVM is a virtualization software for Linux which is based on hardware virtualization extensions. The cluster resource manager in SUSE Linux Enterprise High Availability can recognize, monitor, and manage services running within virtual servers and services running in physical servers. Guest systems can be managed as services by the cluster.

> **! Important: Live migration in High Availability clusters**
>
> Use caution when performing live migration of nodes in an active cluster. The cluster stack might not tolerate an operating system freeze caused by the live migration process, which could lead to the node being fenced.

We recommend either of the following actions to help avoid node fencing during live migration:

- Increase the Corosync token timeout and the SBD watchdog timeout, along with any other related settings. The appropriate values depend on your specific setup. For more information, see *Section 13.5, "Calculation of timeouts"*.

- Before performing live migration, stop the cluster services on either the node or the whole cluster. For more information, see *Section 28.2, "Different options for maintenance tasks"*.

You **must** thoroughly test this setup before attempting live migration in a production environment.

1.2.5   Support of local, metro, and Geo clusters

SUSE Linux Enterprise High Availability supports different geographical scenarios, including geographically dispersed clusters (Geo clusters).

Local clusters

A single cluster in one location (for example, all nodes are located in one data center). The cluster uses multicast or unicast for communication between the nodes and manages failover internally. Network latency can be neglected. Storage is typically accessed synchronously by all nodes.

Metro clusters

A single cluster that can stretch over multiple buildings or data centers, with all sites connected by fibre channel. The cluster uses multicast or unicast for communication between the nodes and manages failover internally. Network latency is usually low (<5 ms for distances of approximately 20 miles). Storage is frequently replicated (mirroring or synchronous replication).

Geo clusters (multi-site clusters)

Multiple, geographically dispersed sites with a local cluster each. The sites communicate via IP. Failover across the sites is coordinated by a higher-level entity. Geo clusters need to cope with limited network bandwidth and high latency. Storage is replicated asynchronously.

> **Note: Geo clustering and SAP workloads**
>
> Currently Geo clusters support neither SAP HANA system replication nor SAP S/4HANA and SAP NetWeaver enqueue replication setups.

The greater the geographical distance between individual cluster nodes, the more factors may potentially disturb the high availability of services the cluster provides. Network latency, limited bandwidth and access to storage are the main challenges for long-distance clusters.

1.2.6 Resource agents

SUSE Linux Enterprise High Availability includes many resource agents to manage resources such as Apache, IPv4, IPv6 and many more. It also ships with resource agents for popular third party applications such as IBM WebSphere Application Server. For an overview of Open Cluster Framework (OCF) resource agents included with your product, use the `crm ra` command as described in *Section 5.5.3, "Displaying information about OCF resource agents"*.

1.2.7 User-friendly administration tools

SUSE Linux Enterprise High Availability ships with a set of powerful tools. Use them for basic installation and setup of your cluster and for effective configuration and administration:

YaST

A graphical user interface for general system installation and administration. Use it to install SUSE Linux Enterprise High Availability on top of SUSE Linux Enterprise Server as described in the Installation and Setup Quick Start. YaST also provides the following modules in the High Availability category to help configure your cluster or individual components:

- Cluster: Basic cluster setup. For details, refer to *Chapter 4, Using the YaST cluster module*.

- DRBD: Configuration of a Distributed Replicated Block Device.

- IP Load Balancing: Configuration of load balancing with Linux Virtual Server or HAProxy. For details, refer to *Chapter 17, Load balancing*.

Hawk2

A user-friendly Web-based interface with which you can monitor and administer your High Availability clusters from Linux or non-Linux machines alike. Hawk2 can be accessed from any machine inside or outside of the cluster by using a (graphical) Web browser. Therefore it is the ideal solution even if the system on which you are working only provides a minimal graphical user interface. For details, *Section 5.4, "Introduction to Hawk2"*.

`crm` Shell

A powerful unified command line interface to configure resources and execute all monitoring or administration tasks. For details, refer to *Section 5.5, "Introduction to crmsh"*.

1.3 Benefits

SUSE Linux Enterprise High Availability allows you to configure up to 32 Linux servers into a high-availability cluster (HA cluster). Resources can be dynamically switched or moved to any node in the cluster. Resources can be configured to automatically migrate if a node fails, or they can be moved manually to troubleshoot hardware or balance the workload.

SUSE Linux Enterprise High Availability provides high availability from commodity components. Lower costs are obtained through the consolidation of applications and operations onto a cluster. SUSE Linux Enterprise High Availability also allows you to centrally manage the complete cluster. You can adjust resources to meet changing workload requirements (thus, manually "load balance" the cluster). Allowing clusters of more than two nodes also provides savings by allowing several nodes to share a "hot spare".

An equally important benefit is the potential reduction of unplanned service outages and planned outages for software and hardware maintenance and upgrades.

Reasons that you would want to implement a cluster include:

- Increased availability

- Improved performance

- Low cost of operation

- Scalability

- Disaster recovery

- Data protection

- Server consolidation

- Storage consolidation

Shared disk fault tolerance can be obtained by implementing RAID on the shared disk subsystem.

The following scenario illustrates some benefits SUSE Linux Enterprise High Availability can provide.

Example cluster scenario

Suppose you have configured a three-node cluster, with a Web server installed on each of the three nodes in the cluster. Each of the nodes in the cluster hosts two Web sites. All the data, graphics, and Web page content for each Web site are stored on a shared disk subsystem connected to each of the nodes in the cluster. The following figure depicts how this setup might look.
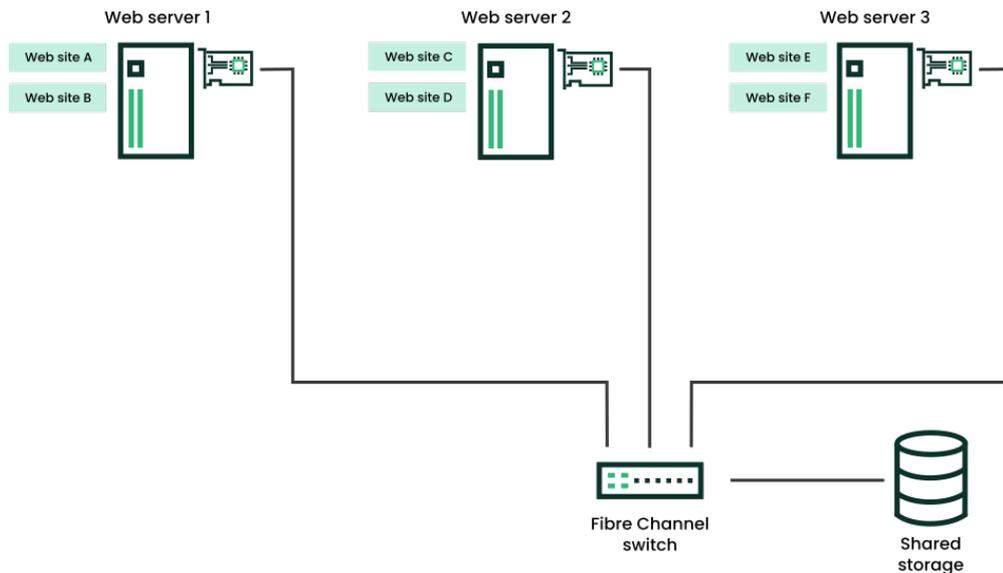
FIGURE 1.1: **THREE-SERVER CLUSTER**

During normal cluster operation, each node is in constant communication with the other nodes in the cluster and performs periodic polling of all registered resources to detect failure.

Suppose Web Server 1 experiences hardware or software problems and the users depending on Web Server 1 for Internet access, e-mail, and information lose their connections. The following figure shows how resources are moved when Web Server 1 fails.
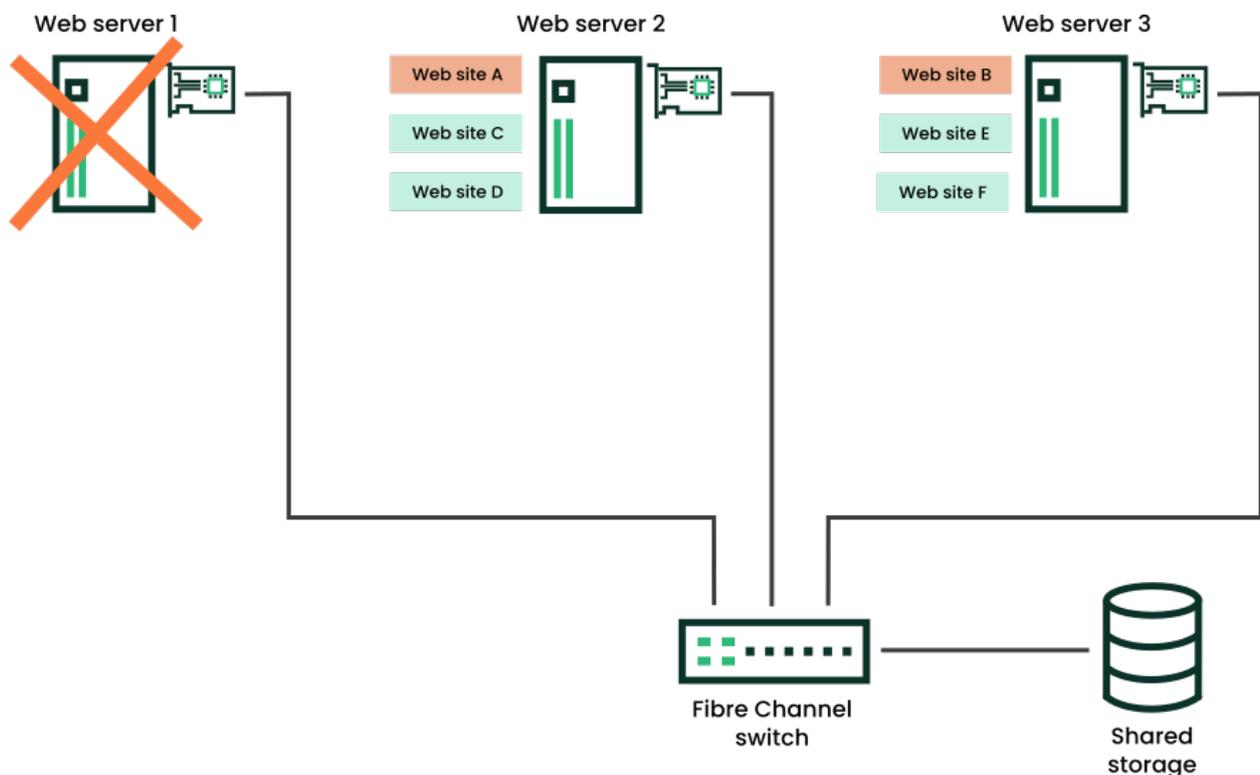
FIGURE 1.2: THREE-SERVER CLUSTER AFTER ONE SERVER FAILS

Web Site A moves to Web Server 2 and Web Site B moves to Web Server 3. IP addresses and certificates also move to Web Server 2 and Web Server 3.

When you configured the cluster, you decided where the Web sites hosted on each Web server would go should a failure occur. In the previous example, you configured Web Site A to move to Web Server 2 and Web Site B to move to Web Server 3. This way, the workload formerly handled by Web Server 1 continues to be available and is evenly distributed between any surviving cluster members.

When Web Server 1 failed, the High Availability software did the following:

- Detected a failure and verified with STONITH that Web Server 1 was really dead. STONITH is an acronym for "Shoot The Other Node In The Head". It is a means of bringing down misbehaving nodes to prevent them from causing trouble in the cluster.

- Remounted the shared data directories that were formerly mounted on Web server 1 on Web Server 2 and Web Server 3.

- Restarted applications that were running on Web Server 1 on Web Server 2 and Web Server 3.

- Transferred IP addresses to Web Server 2 and Web Server 3.

Example cluster scenario   SLE HA 15 SP5

In this example, the failover process happened quickly and users regained access to Web site information within seconds, usually without needing to log in again.

Now suppose the problems with Web Server 1 are resolved, and Web Server 1 is returned to a normal operating state. Web Site A and Web Site B can either automatically fail back (move back) to Web Server 1, or they can stay where they are. This depends on how you configured the resources for them. Migrating the services back to Web Server 1 will incur some downtime. Therefore SUSE Linux Enterprise High Availability also allows you to defer the migration until a period when it will cause little or no service interruption. There are advantages and disadvantages to both alternatives.

SUSE Linux Enterprise High Availability also provides resource migration capabilities. You can move applications, Web sites, etc. to other servers in your cluster as required for system management.

For example, you could have manually moved Web Site A or Web Site B from Web Server 1 to either of the other servers in the cluster. Use cases for this are upgrading or performing scheduled maintenance on Web Server 1, or increasing performance or accessibility of the Web sites.

1.4 Cluster configurations: storage

Cluster configurations with SUSE Linux Enterprise High Availability might or might not include a shared disk subsystem. The shared disk subsystem can be connected via high-speed Fibre Channel cards, cables, and switches, or it can be configured to use iSCSI. If a node fails, another designated node in the cluster automatically mounts the shared disk directories that were previously mounted on the failed node. This gives network users continuous access to the directories on the shared disk subsystem.

> **❗ Important: Shared disk subsystem with LVM**
>
> When using a shared disk subsystem with LVM, that subsystem must be connected to all servers in the cluster from which it needs to be accessed.

Typical resources might include data, applications, and services. The following figures show how a typical Fibre Channel cluster configuration might look. The green lines depict connections to an Ethernet power switch. Such a device can be controlled over a network and can reboot a node when a ping request fails.
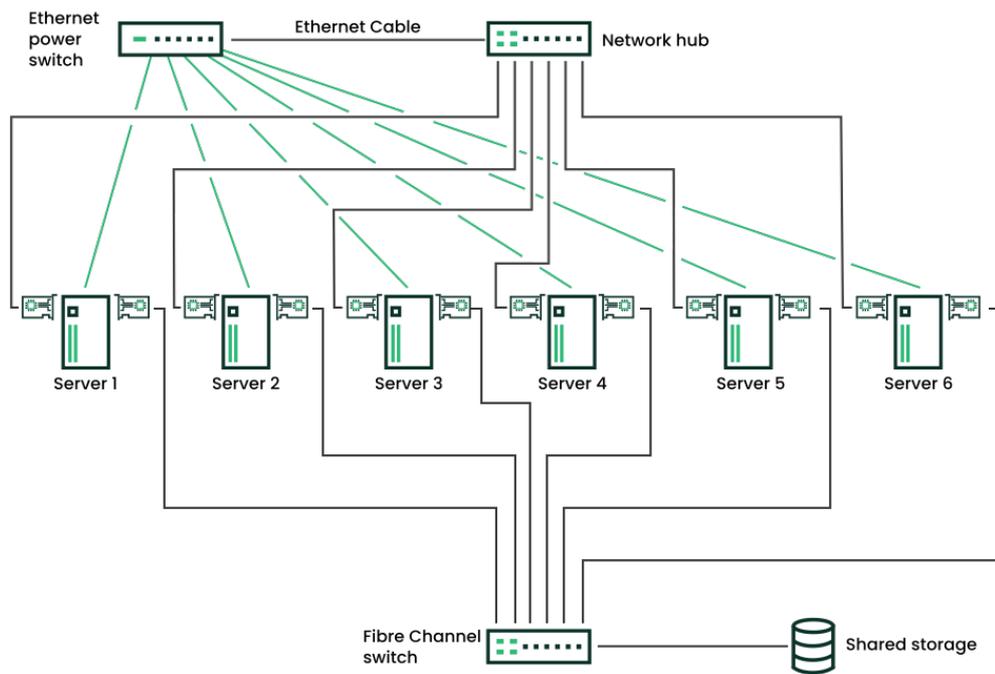
FIGURE 1.3: TYPICAL FIBRE CHANNEL CLUSTER CONFIGURATION

Although Fibre Channel provides the best performance, you can also configure your cluster to use iSCSI. iSCSI is an alternative to Fibre Channel that can be used to create a low-cost Storage Area Network (SAN). The following figure shows how a typical iSCSI cluster configuration might look.
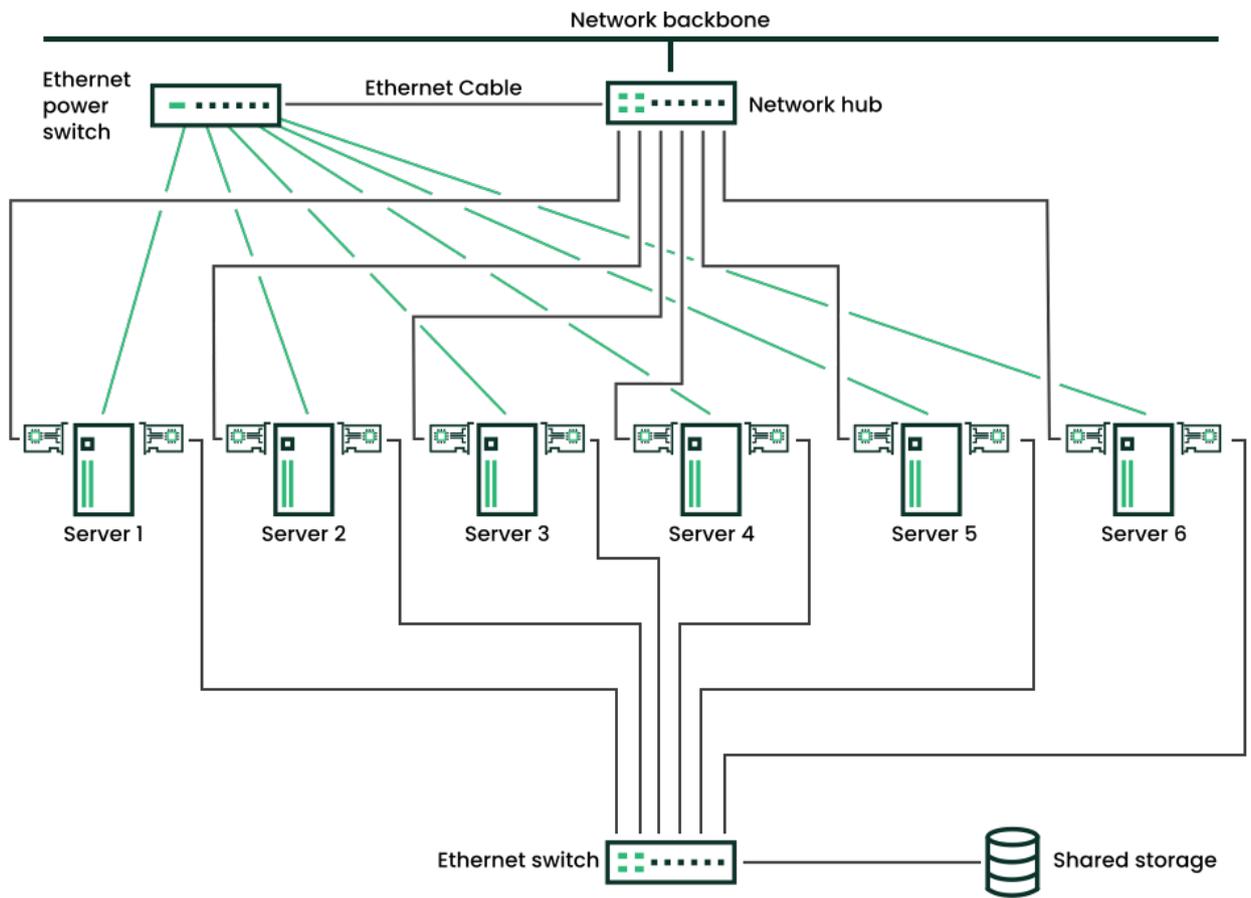
FIGURE 1.4: **TYPICAL ISCSI CLUSTER CONFIGURATION**

Although most clusters include a shared disk subsystem, it is also possible to create a cluster without a shared disk subsystem. The following figure shows how a cluster without a shared disk subsystem might look.
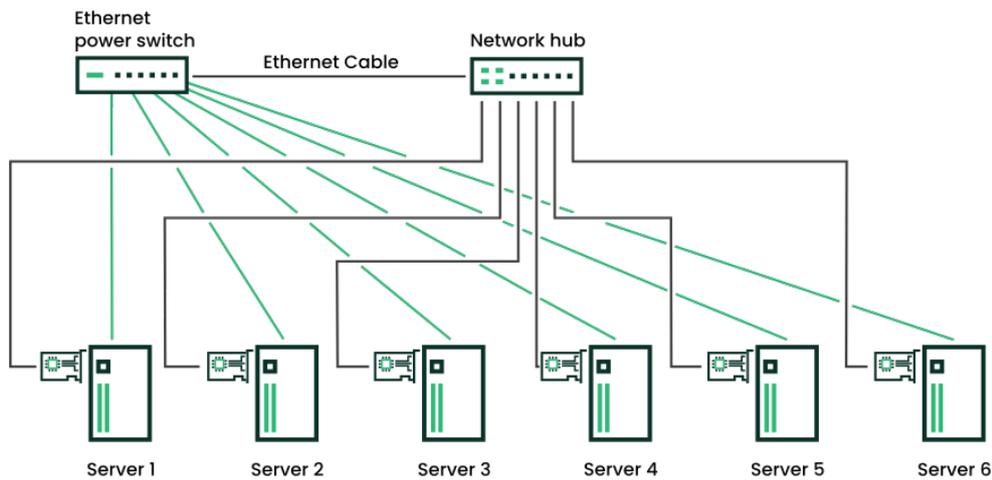
FIGURE 1.5: **TYPICAL CLUSTER CONFIGURATION WITHOUT SHARED STORAGE**

1.5 Architecture

This section provides a brief overview of SUSE Linux Enterprise High Availability architecture. It identifies and provides information on the architectural components, and describes how those components interoperate.

1.5.1 Architecture layers

SUSE Linux Enterprise High Availability has a layered architecture. *Figure 1.6, "Architecture"* illustrates the different layers and their associated components.
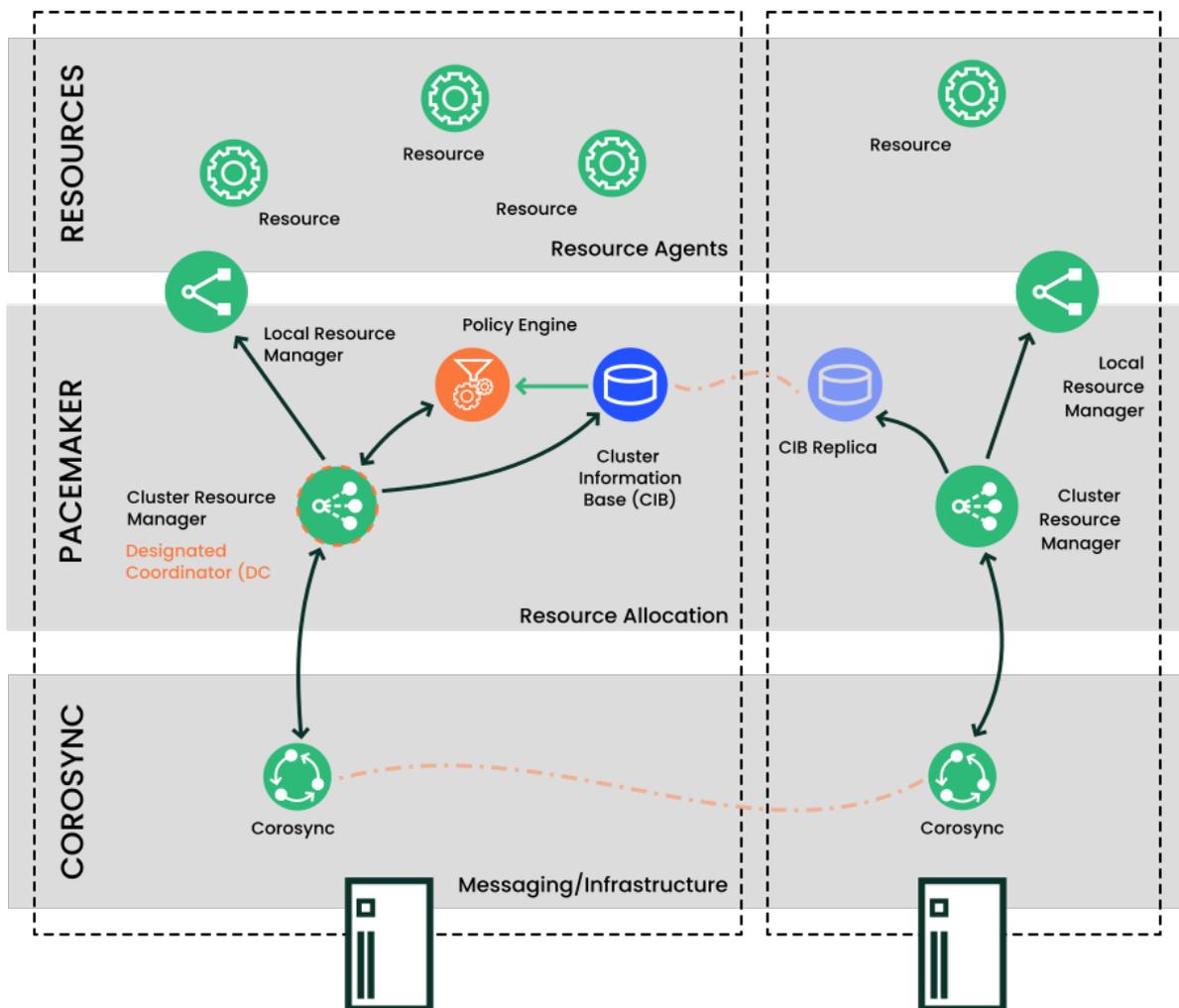
FIGURE 1.6: ARCHITECTURE

1.5.1.1 Membership and messaging layer (Corosync)

This component provides reliable messaging, membership, and quorum information about the cluster. This is handled by the Corosync cluster engine, a group communication system.

1.5.1.2 Cluster resource manager (Pacemaker)

Pacemaker as cluster resource manager is the "brain" which reacts to events occurring in the cluster. It is implemented as `pacemaker-controld`, the cluster controller, which coordinates all actions. Events can be nodes that join or leave the cluster, failure of resources, or scheduled activities such as maintenance, for example.

Local resource manager

The local resource manager is located between the Pacemaker layer and the resources layer on each node. It is implemented as `pacemaker-execd` daemon. Through this daemon, Pacemaker can start, stop, and monitor resources.

Cluster Information Database (CIB)

On every node, Pacemaker maintains the cluster information database (CIB). It is an XML representation of the cluster configuration (including cluster options, nodes, resources, constraints and the relationship to each other). The CIB also reflects the current cluster status. Each cluster node contains a CIB replica, which is synchronized across the whole cluster. The `pacemaker-based` daemon takes care of reading and writing cluster configuration and status.

Designated Coordinator (DC)

The DC is elected from all nodes in the cluster. This happens if there is no DC yet or if the current DC leaves the cluster for any reason. The DC is the only entity in the cluster that can decide that a cluster-wide change needs to be performed, such as fencing a node or moving resources around. All other nodes get their configuration and resource allocation information from the current DC.

Policy Engine

The policy engine runs on every node, but the one on the DC is the active one. The engine is implemented as `pacemaker-schedulerd` daemon. When a cluster transition is needed, based on the current state and configuration, `pacemaker-schedulerd` calculates the expected next state of the cluster. It determines what actions need to be scheduled to achieve the next state.

1.5.1.3 Resources and resource agents

In a High Availability cluster, the services that need to be highly available are called resources. Resource agents (RAs) are scripts that start, stop, and monitor cluster resources.

1.5.2 Process flow

The `pacemakerd` daemon launches and monitors all other related daemons. The daemon that coordinates all actions, `pacemaker-controld`, has an instance on each cluster node. Pacemaker centralizes all cluster decision-making by electing one of those instances as a primary. Should the elected `pacemaker-controld` daemon fail, a new primary is established.

Many actions performed in the cluster will cause a cluster-wide change. These actions can include things like adding or removing a cluster resource or changing resource constraints. It is important to understand what happens in the cluster when you perform such an action.

For example, suppose you want to add a cluster IP address resource. To do this, you can use the crm shell or the Web interface to modify the CIB. It is not required to perform the actions on the DC. You can use either tool on any node in the cluster and they will be relayed to the DC. The DC will then replicate the CIB change to all cluster nodes.

Based on the information in the CIB, the `pacemaker-schedulerd` then computes the ideal state of the cluster and how it should be achieved. It feeds a list of instructions to the DC. The DC sends commands via the messaging/infrastructure layer which are received by the `pacemaker-controld` peers on other nodes. Each of them uses its local resource agent executor (implemented as `pacemaker-execd`) to perform resource modifications. The `pacemaker-execd` is not cluster-aware and interacts directly with resource agents.

All peer nodes report the results of their operations back to the DC. After the DC concludes that all necessary operations are successfully performed in the cluster, the cluster will go back to the idle state and wait for further events. If any operation was not carried out as planned, the `pacemaker-schedulerd` is invoked again with the new information recorded in the CIB.

In some cases, it might be necessary to power off nodes to protect shared data or complete resource recovery. In a Pacemaker cluster, the implementation of node level fencing is STONITH. For this, Pacemaker comes with a fencing subsystem, `pacemaker-fenced`. STONITH devices must be configured as cluster resources (that use specific fencing agents), because this allows monitoring of the fencing devices. When clients detect a failure, they send a request to `pacemaker-fenced`, which then executes the fencing agent to bring down the node.

2 System requirements and recommendations

The following section informs you about system requirements and prerequisites for SUSE® Linux Enterprise High Availability. It also includes recommendations for cluster setup.

2.1 Hardware requirements

The following list specifies hardware requirements for a cluster based on SUSE® Linux Enterprise High Availability. These requirements represent the minimum hardware configuration. Additional hardware might be necessary, depending on how you intend to use your cluster.

Servers

1 to 32 Linux servers with software as specified in *Section 2.2, "Software requirements"*. The servers can be bare metal or virtual machines. They do not require identical hardware (memory, disk space, etc.), but they must have the same architecture. Cross-platform clusters are not supported.

Using `pacemaker_remote`, the cluster can be extended to include additional Linux servers beyond the 32-node limit.

Communication channels

At least two TCP/IP communication media per cluster node. The network equipment must support the communication means you want to use for cluster communication: multicast or unicast. The communication media should support a data rate of 100 Mbit/s or higher. For a supported cluster setup two or more redundant communication paths are required. This can be done via:

- Network Device Bonding (preferred).

- A second communication channel in Corosync.

For details, refer to *Chapter 16, Network device bonding* and *Procedure 4.3, "Defining a redundant communication channel"*, respectively.

Node fencing/STONITH

To avoid a "split-brain" scenario, clusters need a node fencing mechanism. In a split-brain scenario, cluster nodes are divided into two or more groups that do not know about each other (because of a hardware or software failure or because of a cut network connection).

A fencing mechanism isolates the node in question (usually by resetting or powering off the node). This is also called STONITH ("Shoot the other node in the head"). A node fencing mechanism can be either a physical device (a power switch) or a mechanism like SBD (STONITH by disk) in combination with a watchdog. Using SBD requires shared storage. Unless SBD is used, each node in the High Availability cluster must have at least one STONITH device. We strongly recommend multiple STONITH devices per node.

> **❗ Important: No Support Without STONITH**
>
> - You must have a node fencing mechanism for your cluster.
>
> - The global cluster options `stonith-enabled` and `startup-fencing` must be set to `true`. When you change them, you lose support.

2.2 Software requirements

All nodes that will be part of the cluster need at least the following modules and extensions:

- Basesystem Module 15 SP5

- Server Applications Module 15 SP5

- SUSE Linux Enterprise High Availability 15 SP5

Depending on the system role you select during installation, the following software patterns are installed by default:

HA Node system role
High Availability (`sles_ha`)
Enhanced Base System (`enhanced_base`)

HA GEO Node system role
Geo Clustering for High Availability (`ha_geo`)
Enhanced Base System (`enhanced_base`)

> ## 🖊️ Note: Minimal installation
>
> An installation via those system roles results in a minimal installation only. You might need to add more packages manually, if required.
>
> For machines that originally had another system role assigned, you need to manually install the `sles_ha` or `ha_geo` patterns and any further packages that you need.

2.3 Storage requirements

Some services require shared storage. If using an external NFS share, it must be reliably accessible from all cluster nodes via redundant communication paths.

To make data highly available, a shared disk system (Storage Area Network, or SAN) is recommended for your cluster. If a shared disk subsystem is used, ensure the following:

- The shared disk system is properly set up and functional according to the manufacturer's instructions.

- The disks contained in the shared disk system should be configured to use mirroring or RAID to add fault tolerance to the shared disk system.

- If you are using iSCSI for shared disk system access, ensure that you have properly configured iSCSI initiators and targets.

- When using DRBD* to implement a mirroring RAID system that distributes data across two machines, make sure to only access the device provided by DRBD—never the backing device. To leverage the redundancy it is possible to use the same NICs as the rest of the cluster.

When using SBD as STONITH mechanism, additional requirements apply for the shared storage. For details, see *Section 13.3, "Requirements and restrictions"*.

2.4 Other requirements and recommendations

For a supported and useful High Availability setup, consider the following recommendations:

Number of cluster nodes

For clusters with more than two nodes, it is strongly recommended to use an odd number of cluster nodes to have quorum. For more information about quorum, see *Section 5.2, "Quorum determination"*. A regular cluster can contain up to 32 nodes. With the `pacemaker_remote` service, High Availability clusters can be extended to include additional nodes beyond this limit. See Pacemaker Remote Quick Start for more details.

Time synchronization

Cluster nodes must synchronize to an NTP server outside the cluster. Since SUSE Linux Enterprise High Availability 15, chrony is the default implementation of NTP. For more information, see the Administration Guide for SUSE Linux Enterprise Server 15 SP5 (https://documentation.suse.com/sles-15/html/SLES-all/cha-ntp.html) ↗.

The cluster might not work properly if the nodes are not synchronized, or even if they are synchronized but have different timezones configured. In addition, log files and cluster reports are very hard to analyze without synchronization. If you use the bootstrap scripts, you will be warned if NTP is not configured yet.

Network Interface Card (NIC) names

Must be identical on all nodes.

Host name and IP address

- Use static IP addresses.

- Only the primary IP address is supported.

- List all cluster nodes in the `/etc/hosts` file with their fully qualified host name and short host name. It is essential that members of the cluster can find each other by name. If the names are not available, internal cluster communication will fail.
  For details on how Pacemaker gets the node names, see also https://clusterlab-s.org/doc/en-US/Pacemaker/1.1/html/Pacemaker_Explained/s-node-name.html ↗.

SSH

All cluster nodes must be able to access each other via SSH. Tools like `crm report` (for troubleshooting) and Hawk2's *History Explorer* require passwordless SSH access between the nodes, otherwise they can only collect data from the current node.

Note: Regulatory requirements

If passwordless SSH access does not comply with regulatory requirements, you can use the work-around described in *Appendix D, Running cluster reports without* `root` *access* for running `crm report`.

For the *History Explorer* there is currently no alternative for passwordless login.

3 Installing SUSE Linux Enterprise High Availability

If you are setting up a High Availability cluster with SUSE® Linux Enterprise High Availability for the first time, the easiest way is to start with a basic two-node cluster. You can also use the two-node cluster to run some tests. Afterward, you can add more nodes by cloning existing cluster nodes with AutoYaST. The cloned nodes will have the same packages installed and the same system configuration as the original ones.

If you want to upgrade an existing cluster that runs an older version of SUSE Linux Enterprise High Availability, refer to *Chapter 29, Upgrading your cluster and updating software packages*.

3.1 Manual installation

For the manual installation of the packages for High Availability refer to *Article "Installation and Setup Quick Start"*. It leads you through the setup of a basic two-node cluster.

3.2 Mass installation and deployment with AutoYaST

After you have installed and set up a two-node cluster, you can extend the cluster by cloning existing nodes with AutoYaST and adding the clones to the cluster.

AutoYaST uses profiles that contains installation and configuration data. A profile tells AutoYaST what to install and how to configure the installed system to get a ready-to-use system in the end. This profile can then be used for mass deployment in different ways (for example, to clone existing cluster nodes).

For detailed instructions on how to use AutoYaST in various scenarios, see the AutoYaST Guide for SUSE Linux Enterprise Server 15 SP5 (https://documentation.suse.com/sles-15/html/SLES-all/book-autoyast.html) .

❗ Important: Identical hardware

Procedure 3.1, "Cloning a cluster node with AutoYaST" assumes you are rolling out SUSE Linux Enterprise High Availability 15 SP5 to a set of machines with identical hardware configurations.

If you need to deploy cluster nodes on non-identical hardware, refer to the Deployment Guide for SUSE Linux Enterprise Server 15 SP5, chapter *Automated Installation*, section *Rule-Based Autoinstallation*.

PROCEDURE 3.1: CLONING A CLUSTER NODE WITH AUTOYAST

1. Make sure the node you want to clone is correctly installed and configured. For details, see the Installation and Setup Quick Start or *Chapter 4, Using the YaST cluster module*.

2. Follow the description outlined in the SUSE Linux Enterprise 15 SP5 Deployment Guide for simple mass installation. This includes the following basic steps:

   a. Creating an AutoYaST profile. Use the AutoYaST GUI to create and modify a profile based on the existing system configuration. In AutoYaST, choose the *High Availability* module and click the *Clone* button. If needed, adjust the configuration in the other modules and save the resulting control file as XML.
   If you have configured DRBD, you can select and clone this module in the AutoYaST GUI, too.

   b. Determining the source of the AutoYaST profile and the parameter to pass to the installation routines for the other nodes.

   c. Determining the source of the SUSE Linux Enterprise Server and SUSE Linux Enterprise High Availability installation data.

   d. Determining and setting up the boot scenario for autoinstallation.

   e. Passing the command line to the installation routines, either by adding the parameters manually or by creating an `info` file.

   f. Starting and monitoring the autoinstallation process.

After the clone has been successfully installed, execute the following steps to make the cloned node join the cluster:

1. Transfer the key configuration files from the already configured nodes to the cloned node with Csync2 as described in *Section 4.7, "Transferring the configuration to all nodes"*.

2. To bring the node online, start the cluster services on the cloned node as described in *Section 4.8, "Bringing the cluster online"*.

The cloned node now joins the cluster because the `/etc/corosync/corosync.conf` file has been applied to the cloned node via Csync2. The CIB is automatically synchronized among the cluster nodes.

4 Using the YaST cluster module

The YaST cluster module allows you to set up a cluster manually (from scratch) or to modify options for an existing cluster.

However, if you prefer an automated approach for setting up a cluster, refer to *Article "Installation and Setup Quick Start"*. It describes how to install the needed packages and leads you to a basic two-node cluster, which is set up with the bootstrap scripts provided by the crm shell.

You can also use a combination of both setup methods, for example: set up one node with YaST cluster and then use one of the bootstrap scripts to integrate more nodes (or vice versa).

4.1 Definition of terms

Several key terms used in the YaST cluster module and in this chapter are defined below.

Bind network address (`bindnetaddr`)

The network address the Corosync executive should bind to. To simplify sharing configuration files across the cluster, Corosync uses network interface netmask to mask only the address bits that are used for routing the network. For example, if the local interface is `192.168.5.92` with netmask `255.255.255.0`, set `bindnetaddr` to `192.168.5.0`. If the local interface is `192.168.5.92` with netmask `255.255.255.192`, set `bindnetaddr` to `192.168.5.64`.

If `nodelist` with `ringX_addr` is explicitly configured in `/etc/corosync/corosync.conf`, `bindnetaddr` is not strictly required.

> **Note: Network address for all nodes**
>
> As the same Corosync configuration is used on all nodes, make sure to use a network address as `bindnetaddr`, not the address of a specific network interface.

`conntrack` **Tools**

Allow interaction with the in-kernel connection tracking system for enabling *stateful* packet inspection for iptables. Used by SUSE Linux Enterprise High Availability to synchronize the connection status between cluster nodes. For detailed information, refer to https:// conntrack-tools.netfilter.org/ ↗.

Csync2

A synchronization tool that can be used to replicate configuration files across all nodes in the cluster, and even across Geo clusters. Csync2 can handle any number of hosts, sorted into synchronization groups. Each synchronization group has its own list of member hosts and its include/exclude patterns that define which files should be synchronized in the synchronization group. The groups, the host names belonging to each group, and the include/exclude rules for each group are specified in the Csync2 configuration file, `/etc/ csync2/csync2.cfg`.

For authentication, Csync2 uses the IP addresses and pre-shared keys within a synchronization group. You need to generate one key file for each synchronization group and copy it to all group members.

For more information about Csync2, refer to https://oss.linbit.com/csync2/paper.pdf ↗

Existing cluster

The term "existing cluster" is used to refer to any cluster that consists of at least one node. Existing clusters have a basic Corosync configuration that defines the communication channels, but they do not necessarily have resource configuration yet.

Multicast

A technology used for a one-to-many communication within a network that can be used for cluster communication. Corosync supports both multicast and unicast.

> ## Note: Switches and multicast
>
> To use multicast for cluster communication, make sure your switches support multicast.

Multicast address (`mcastaddr`)

IP address to be used for multicasting by the Corosync executive. The IP address can either be IPv4 or IPv6. If IPv6 networking is used, node IDs must be specified. You can use any multicast address in your private network.

Multicast port (`mcastport`)

> The port to use for cluster communication. Corosync uses two ports: the specified `mcastport` for receiving multicast, and `mcastport -1` for sending multicast.

Redundant Ring Protocol (RRP)

> Allows the use of multiple redundant local area networks for resilience against partial or total network faults. This way, cluster communication can still be kept up as long as a single network is operational. Corosync supports the Totem Redundant Ring Protocol. A logical token-passing ring is imposed on all participating nodes to deliver messages in a reliable and sorted manner. A node is allowed to broadcast a message only if it holds the token. When having defined redundant communication channels in Corosync, use RRP to tell the cluster how to use these interfaces. RRP can have three modes (`rrp_mode`):
>
> - If set to `active`, Corosync uses both interfaces actively. However, this mode is deprecated.
>
> - If set to `passive`, Corosync sends messages alternatively over the available networks.
>
> - If set to `none`, RRP is disabled.

Unicast

> A technology for sending messages to a single network destination. Corosync supports both multicast and unicast. In Corosync, unicast is implemented as UDP-unicast (UDPU).

4.2   YaST *Cluster* module

Start YaST and select *High Availability* › *Cluster*. Alternatively, start the module from command line:

```
sudo yast2 cluster
```

The following list shows an overview of the available screens in the YaST cluster module. It also mentions whether the screen contains parameters that are *required* for successful cluster setup or whether its parameters are *optional*.

Communication channels (required)

> Allows you to define one or two communication channels for communication between the cluster nodes. As transport protocol, either use multicast (UDP) or unicast (UDPU). For details, see *Section 4.3, "Defining the communication channels"*.

❗ Important: Redundant communication paths

For a supported cluster setup two or more redundant communication paths are required. The preferred way is to use network device bonding as described in *Chapter 16, Network device bonding*.

If this is impossible, you need to define a second communication channel in Corosync.

Security (optional but recommended)

Allows you to define the authentication settings for the cluster. HMAC/SHA1 authentication requires a shared secret used to protect and authenticate messages. For details, see *Section 4.4, "Defining authentication settings"*.

Configure Csync2 (optional but recommended)

Csync2 helps you to keep track of configuration changes and to keep files synchronized across the cluster nodes. For details, see *Section 4.7, "Transferring the configuration to all nodes"*.

Configure conntrackd (optional)

Allows you to configure the user space `conntrackd`. Use the conntrack tools for *stateful* packet inspection for iptables. For details, see *Section 4.5, "Synchronizing connection status between cluster nodes"*.

Service (required)

Allows you to configure the service for bringing the cluster node online. Define whether to start the cluster services at boot time and whether to open the ports in the firewall that are needed for communication between the nodes. For details, see *Section 4.6, "Configuring services"*.

If you start the cluster module for the first time, it appears as a wizard, guiding you through all the steps necessary for basic setup. Otherwise, click the categories on the left panel to access the configuration options for each step.

🔖 Note: Settings in the YaST *Cluster* module

Some settings in the YaST cluster module apply only to the current node. Other settings may automatically be transferred to all nodes with Csync2. Find detailed information about this in the following sections.

4.3 Defining the communication channels

For successful communication between the cluster nodes, define at least one communication channel. As transport protocol, either use multicast (UDP) or unicast (UDPU) as described in *Procedure 4.1* or *Procedure 4.2*, respectively. To define a second, redundant channel (*Procedure 4.3*), both communication channels must use the *same* protocol.

> **Note: Public clouds: use unicast**
>
> For deploying SUSE Linux Enterprise High Availability in public cloud platforms, use unicast as transport protocol. Multicast is generally not supported by the cloud platforms themselves.

All settings defined in the YaST *Communication Channels* screen are written to `/etc/corosync/corosync.conf`. Find example files for a multicast and a unicast setup in `/usr/share/doc/packages/corosync/`.

If you are using IPv4 addresses, node IDs are optional. If you are using IPv6 addresses, node IDs are required. Instead of specifying IDs manually for each node, the YaST cluster module contains an option to automatically generate a unique ID for every cluster node.

PROCEDURE 4.1: DEFINING THE FIRST COMMUNICATION CHANNEL (MULTICAST)

When using multicast, the same `bindnetaddr`, `mcastaddr`, and `mcastport` is used for all cluster nodes. All nodes in the cluster know each other by using the same multicast address. For different clusters, use different multicast addresses.

1. Start the YaST cluster module and switch to the *Communication Channels* category.

2. Set the *Transport* protocol to `Multicast`.

3. Define the *Bind Network Address*. Set the value to the subnet you will use for cluster multicast.

4. Define the *Multicast Address*.

5. Define the *Port*.

6. To automatically generate a unique ID for every cluster node keep *Auto Generate Node ID* enabled.

7. Define a *Cluster Name*.

8. Enter the number of *Expected Votes*. This is important for Corosync to calculate *quorum* in case of a partitioned cluster. By default, each node has 1 vote. The number of *Expected Votes* must match the number of nodes in your cluster.

9. Confirm your changes.

10. If needed, define a redundant communication channel in Corosync as described in *Procedure 4.3, "Defining a redundant communication channel"*.


```
Cluster - Communication Channels
Transport:
Multicast
┌Channel──────────────────────────────────┐ ┌[ ] Redundant Channel──────────────────────┐
│Bind Network Address:                     │ │Bind Network Address:                      │
│192.168.1.0                              ↓│ │                                          ↓│
│Multicast Address:                        │ │Multicast Address:                         │
│239.255.1.1                               │ │                                           │
│Port:                                     │ │Port:                                      │
│5405                                      │ │                                           │
└──────────────────────────────────────────┘ └────────────────────────────────────────────┘
Member Address:
┌──────────────────────────────────────────────────────────────────────────────────────────┐
│IP|Redundant IP|Node ID                                                                     │
│                                                                                            │
│                                                                                            │
│                                                                                            │
│                                                                                            │
│                                                                                            │
│                                                                                            │
│                                                                                            │
│                                                                                            │
│                                                                                            │
│                                                                                            │
│                                                                                            │
│                                                                                            │
│                                                                                 [Add][Del][Edit]│
Cluster Name:                Expected Votes:              rrp mode:
amsterdam                    3                            none                                  ↓
[x] Auto Generate Node ID
```
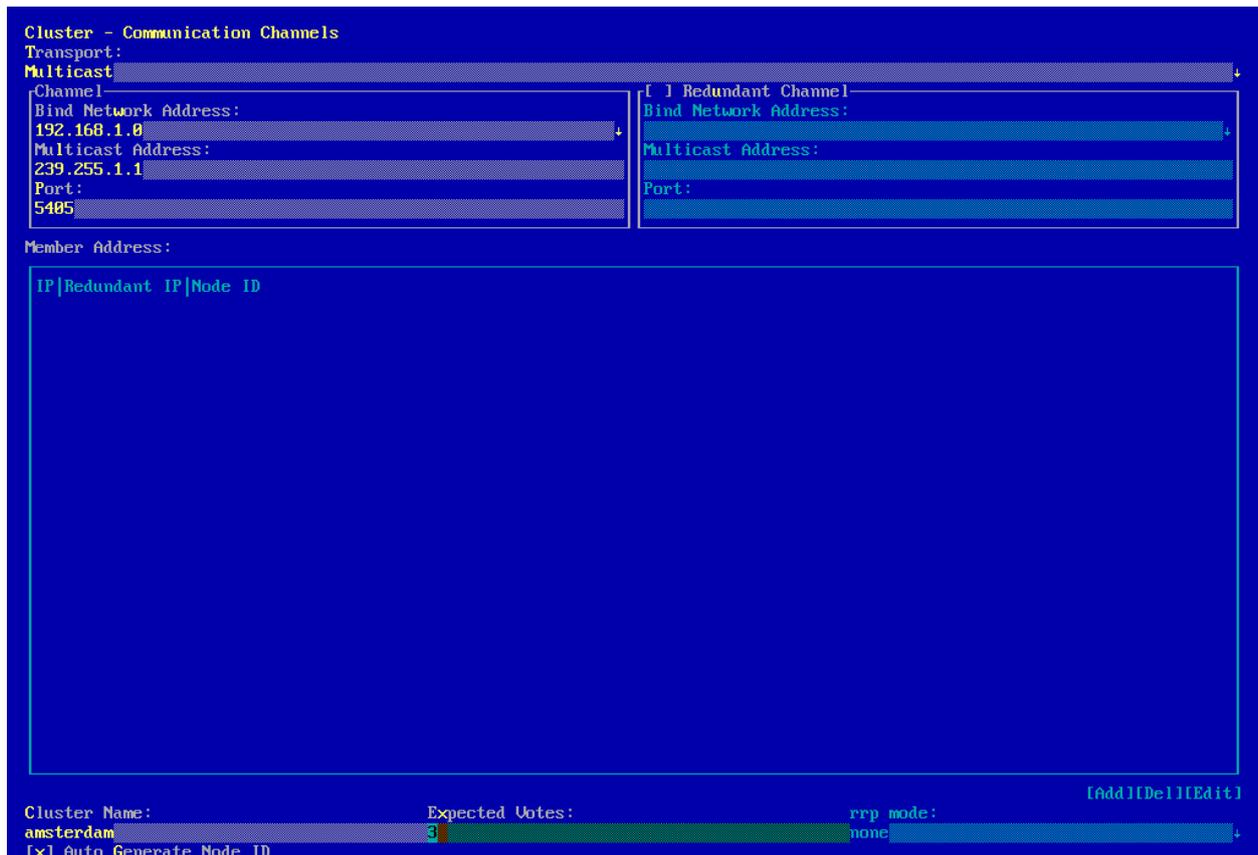
FIGURE 4.1: YAST *CLUSTER*—MULTICAST CONFIGURATION

To use unicast instead of multicast for cluster communication, proceed as follows.

PROCEDURE 4.2: DEFINING THE FIRST COMMUNICATION CHANNEL (UNICAST)

1. Start the YaST cluster module and switch to the *Communication Channels* category.

2. Set the *Transport* protocol to Unicast.

3. Define the *Port*.

4. For unicast communication, Corosync needs to know the IP addresses of all nodes in the cluster. For each node that will be part of the cluster, click *Add* and enter the following details:

   - *IP Address*

   - *Redundant IP Address* (only required if you use a second communication channel in Corosync)

   - *Node ID* (only required if the option *Auto Generate Node ID* is disabled)

   To modify or remove any addresses of cluster members, use the *Edit* or *Del* buttons.

5. To automatically generate a unique ID for every cluster node keep *Auto Generate Node ID* enabled.

6. Define a *Cluster Name*.

7. Enter the number of *Expected Votes*. This is important for Corosync to calculate *quorum* in case of a partitioned cluster. By default, each node has 1 vote. The number of *Expected Votes* must match the number of nodes in your cluster.

8. Confirm your changes.

9. If needed, define a redundant communication channel in Corosync as described in *Procedure 4.3, "Defining a redundant communication channel"*.
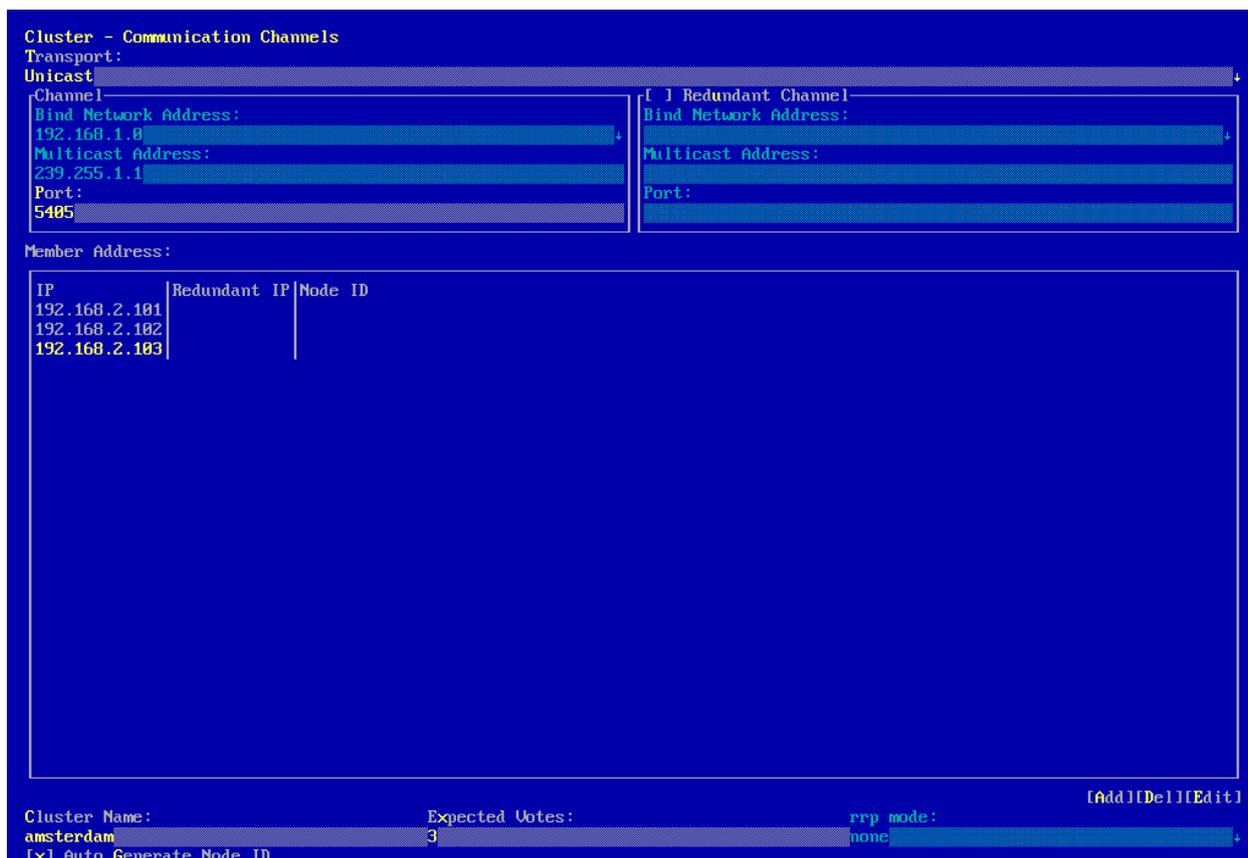
FIGURE 4.2: YAST *CLUSTER*—UNICAST CONFIGURATION

If network device bonding cannot be used for any reason, the second best choice is to define a redundant communication channel (a second ring) in Corosync. That way, two physically separate networks can be used for communication. If one network fails, the cluster nodes can still communicate via the other network.

The additional communication channel in Corosync forms a second token-passing ring. In `/etc/corosync/corosync.conf`, the first channel you configured is the primary ring and gets the ring number `0`. The second ring (redundant channel) gets the ring number `1`.

When having defined redundant communication channels in Corosync, use RRP to tell the cluster how to use these interfaces. With RRP, two physically separate networks are used for communication. If one network fails, the cluster nodes can still communicate via the other network.

RRP can have three modes:

- If set to `active`, Corosync uses both interfaces actively. However, this mode is deprecated.

- If set to `passive`, Corosync sends messages alternatively over the available networks.

- If set to `none`, RRP is disabled.

> ❗ **Important: Redundant rings and `/etc/hosts`**
>
> If multiple rings are configured in Corosync, each node can have multiple IP addresses. This needs to be reflected in the `/etc/hosts` file of all nodes.

1. Start the YaST cluster module and switch to the *Communication Channels* category.

2. Activate *Redundant Channel*. The redundant channel must use the same protocol as the first communication channel you defined.

3. If you use multicast, enter the following parameters: the *Bind Network Address* to use, the *Multicast Address* and the *Port* for the redundant channel.

   If you use unicast, define the following parameters: the *Bind Network Address* to use, and the *Port*. Enter the IP addresses of all nodes that will be part of the cluster.

4. To tell Corosync how and when to use the different channels, select the *rrp_mode* to use:

   - If only one communication channel is defined, *rrp_mode* is automatically disabled (value `none`).

   - If set to `active`, Corosync uses both interfaces actively. However, this mode is deprecated.

   - If set to `passive`, Corosync sends messages alternatively over the available networks.

   When RRP is used, SUSE Linux Enterprise High Availability monitors the status of the current rings and automatically re-enables redundant rings after faults.
   Alternatively, check the ring status manually with `corosync-cfgtool`. View the available options with `-h`.

5. Confirm your changes.

4.4 Defining authentication settings

To define the authentication settings for the cluster, you can use HMAC/SHA1 authentication. This requires a shared secret used to protect and authenticate messages. The authentication key (password) you specify is used on all nodes in the cluster.

PROCEDURE 4.4: ENABLING SECURE AUTHENTICATION

1. Start the YaST cluster module and switch to the *Security* category.

2. Activate *Enable Security Auth*.

3. For a newly created cluster, click *Generate Auth Key File.* An authentication key is created and written to `/etc/corosync/authkey`.
   If you want the current machine to join an existing cluster, do not generate a new key file. Instead, copy the `/etc/corosync/authkey` from one of the nodes to the current machine (either manually or with Csync2).

4. Confirm your changes. YaST writes the configuration to `/etc/corosync/coro-sync.conf`.


```
Cluster - Security

┌[x] Enable Security Auth────────────────────────────────────────────
          Crypto Hash:                    Crypto Cipher:
          sha1▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒↓          aes256▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒↓
     For a newly created cluster, push the button below to generate /etc/corosync/authkey.
     To join an existing cluster, please copy /etc/corosync/authkey from other nodes manually.
                           [Generate Auth Key File]
```
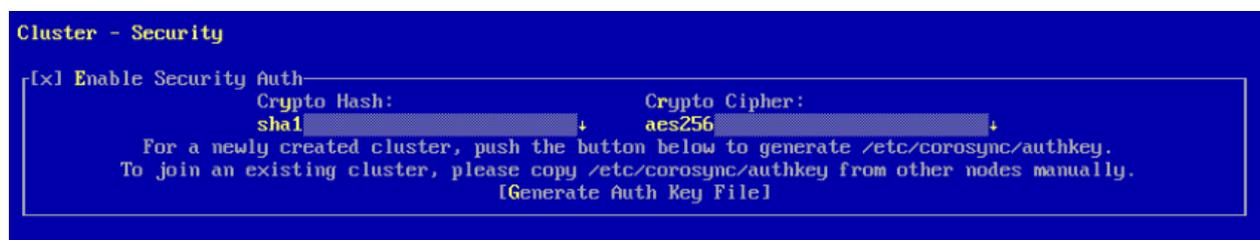
FIGURE 4.3: YAST *CLUSTER*—SECURITY

4.5 Synchronizing connection status between cluster nodes

To enable *stateful* packet inspection for iptables, configure and use the conntrack tools. This requires the following basic steps:

PROCEDURE 4.5: CONFIGURING THE `conntrackd` WITH YAST

Use the YaST cluster module to configure the user space `conntrackd` (see *Figure 4.4, "YaST Cluster—`conntrackd`"*). It needs a dedicated network interface that is not used for other communication channels. The daemon can be started via a resource agent afterward.

1. Start the YaST cluster module and switch to the *Configure conntrackd* category.

2. Define the *Multicast Address* to be used for synchronizing the connection status.

3. In *Group Number*, define a numeric ID for the group to synchronize the connection status to.

4. Click *Generate /etc/conntrackd/conntrackd.conf* to create the configuration file for `conntrackd`.

5. If you modified any options for an existing cluster, confirm your changes and close the cluster module.

6. For further cluster configuration, click *Next* and proceed with *Section 4.6, "Configuring services"*.

7. Select a *Dedicated Interface* for synchronizing the connection status. The IPv4 address of the selected interface is automatically detected and shown in YaST. It must already be configured and it must support multicast.

```
Cluster - Configure conntrackd






















Conntrackd is a daemon which helps to duplicate firewall status between cluster nodes.
YaST can help to configure some basic aspects of conntrackd.
You need to start it with the ocf:heartbeat:conntrackd.
Dedicated Interface:                                        IP: 10.161.10.176
eth0                                                      ↓
Multicast Address:
239.180.93.156
Group Number:
1
[                              Generate /etc/conntrackd/conntrackd.conf                         ]
```
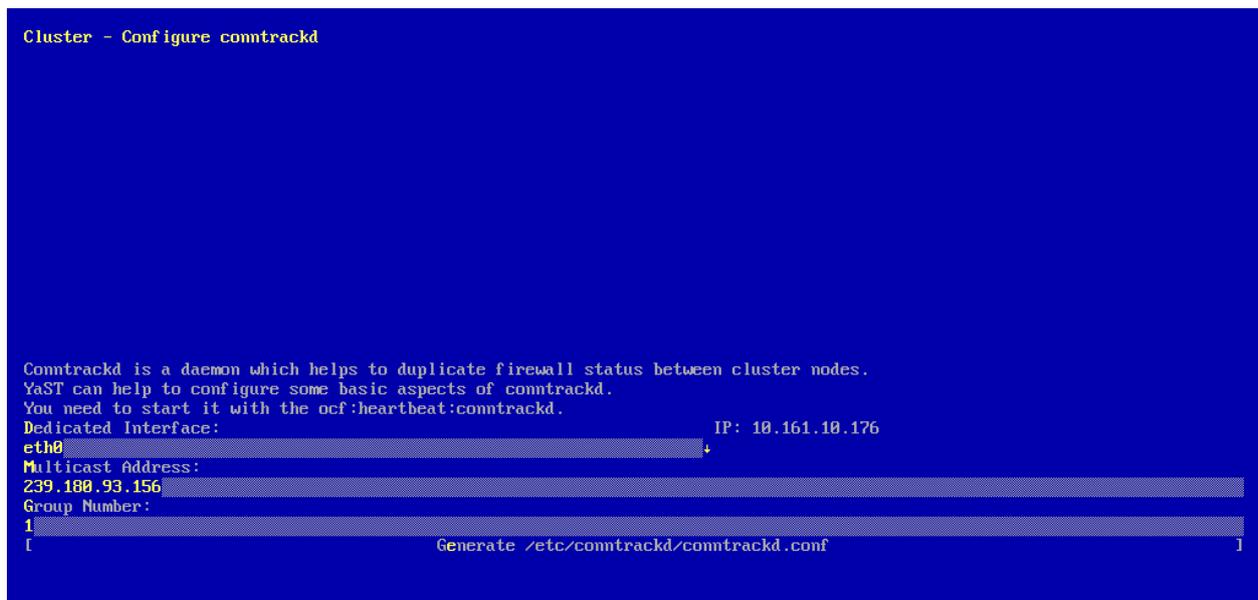
FIGURE 4.4: YAST *CLUSTER*—`conntrackd`

After having configured the conntrack tools, you can use them for Linux Virtual Server (see *Load balancing*).

4.6 Configuring services

In the YaST cluster module define whether to start certain services on a node at boot time. You can also use the module to start and stop the services manually. To bring the cluster nodes online and start the cluster resource manager, Pacemaker must be running as a service.

PROCEDURE 4.6: ENABLING THE CLUSTER SERVICES

1. In the YaST cluster module, switch to the *Service* category.

2. To start the cluster services each time this cluster node is booted, select the respective option in the *Booting* group. If you select *Off* in the *Booting* group, you must start the cluster services manually each time this node is booted. To start the cluster services manually, use the command:

```
# crm cluster start
```

3. To start or stop the cluster services immediately, click the respective button.

4. To open the ports in the firewall that are needed for cluster communication on the current machine, activate *Open Port in Firewall*.

5. Confirm your changes. Note that the configuration only applies to the current machine, not to all cluster nodes.
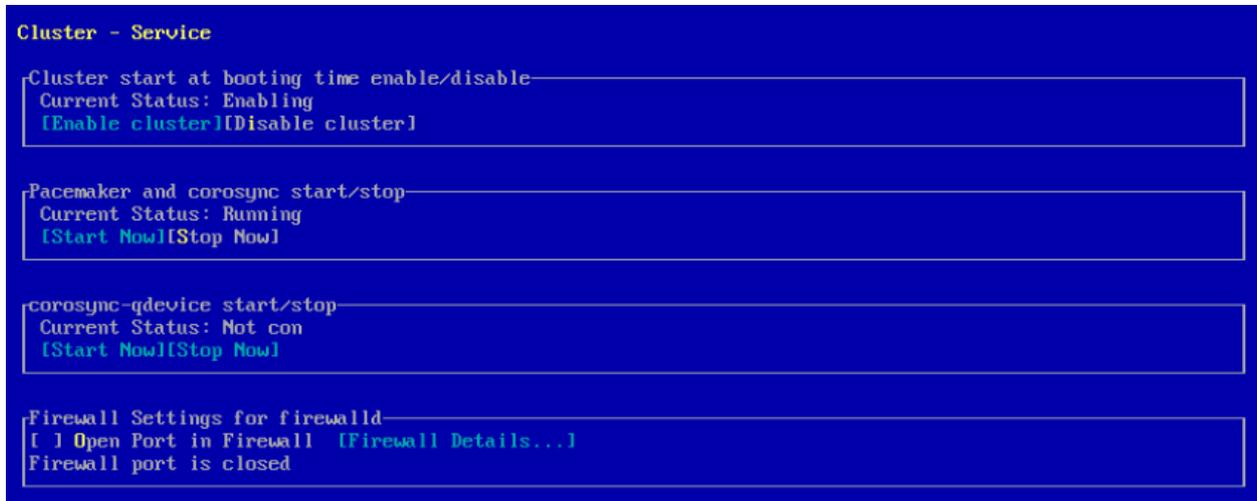
FIGURE 4.5: YAST *CLUSTER*—SERVICES

4.7 Transferring the configuration to all nodes

Instead of copying the resulting configuration files to all nodes manually, use the `csync2` tool for replication across all nodes in the cluster.

This requires the following basic steps:

1. *Configuring Csync2 with YaST*.

2. *Synchronizing the configuration files with Csync2*.

Csync2 helps you to keep track of configuration changes and to keep files synchronized across the cluster nodes:

- You can define a list of files that are important for operation.

- You can show changes to these files (against the other cluster nodes).

- You can synchronize the configured files with a single command.

- With a simple shell script in `~/.bash_logout`, you can be reminded about unsynchronized changes before logging out of the system.

Find detailed information about Csync2 at https://oss.linbit.com/csync2/ ↗ and https://oss.linbit.com/csync2/paper.pdf ↗.

4.7.1 Configuring Csync2 with YaST

PROCEDURE 4.7: CONFIGURING CSYNC2 WITH YAST

1. Start the YaST cluster module and switch to the *Csync2* category.

2. To specify the synchronization group, click *Add* in the *Sync Host* group and enter the local host names of all nodes in your cluster. For each node, you must use exactly the strings that are returned by the **hostname** command.

   > 💡 Tip: Host name resolution
   >
   > If host name resolution does not work properly in your network, you can also specify a combination of host name and IP address for each cluster node. To do so, use the string *HOSTNAME@IP* such as `alice@192.168.2.100`, for example. Csync2 will then use the IP addresses when connecting.

3. Click *Generate Pre-Shared-Keys* to create a key file for the synchronization group. The key file is written to `/etc/csync2/key_hagroup`. After it has been created, it must be copied manually to all members of the cluster.

4. To populate the *Sync File* list with the files that usually need to be synchronized among all nodes, click *Add Suggested Files*.

5. To *Edit*, *Add* or *Remove* files from the list of files to be synchronized use the respective buttons. You must enter the absolute path for each file.

6. Activate Csync2 by clicking *Turn Csync2 ON*. This executes the following command to start Csync2 automatically at boot time:

```
# systemctl enable csync2.socket
```

7. Click *Finish*. YaST writes the Csync2 configuration to `/etc/csync2/csync2.cfg`.

```
Cluster - Configure Csync2
┌Sync Host───────────────────────────────┐  ┌Sync File─────────────────────────────────┐
│alice                                    │  │/etc/corosync/corosync.conf                │
│bob                                      │  │/etc/corosync/authkey                      │
│charlie                                  │  │/etc/sysconfig/pacemaker                   │
│                                         │  │/etc/drbd.d                                │
│                                         │  │/etc/drbd.conf                             │
│                                         │  │/etc/lvm/lvm.conf                          │
│                                         │  │/etc/multipath.conf                        │
│                                         │  │/etc/ha.d/ldirectord.cf                    │
│                                         │  │/etc/ctdb/nodes                            │
│                                         │  │/etc/samba/smb.conf                        │
│                                         │  │/etc/booth                                 │
│                                         │  │/etc/sysconfig/sbd                         │
│                                         │  │/etc/csync2/csync2.cfg                     │
│                                         │  │/etc/csync2/key_hagroup                    │
│                                         │  │                                           │
│                                         │  │                                           │
│             [Add][Del][Edit]            │  │     [Add][Del][Edit][Add Suggested Files] │
└─────────────────────────────────────────┘  └───────────────────────────────────────────┘
[            Generate Pre-Shared-Keys         ][          Turn csync2 ON                  ]
[Help]                    [Back]                      [Abort]                      [Next]
```

FIGURE 4.6: YAST *CLUSTER*—CSYNC2

4.7.2   Synchronizing changes with Csync2

Before running Csync2 for the first time, you need to make the following preparations:

PROCEDURE 4.8: **PREPARING FOR INITIAL SYNCHRONIZATION WITH CSYNC2**

1. Copy the file `/etc/csync2/csync2.cfg` manually to all nodes after you have configured it as described in *Section 4.7.1, "Configuring Csync2 with YaST"*.

2. Copy the file `/etc/csync2/key_hagroup` that you have generated on one node in *Step 3* of *Section 4.7.1* to *all* nodes in the cluster. It is needed for authentication by Csync2. However, do *not* regenerate the file on the other nodes—it needs to be the same file on all nodes.

3. Execute the following command on all nodes to start the service now:

```
# systemctl start csync2.socket
```

PROCEDURE 4.9: SYNCHRONIZING THE CONFIGURATION FILES WITH CSYNC2

1. To initially synchronize all files once, execute the following command on the machine that you want to copy the configuration *from*:

```
# csync2 -xv
```

This synchronizes all the files once by pushing them to the other nodes. If all files are synchronized successfully, Csync2 finishes with no errors.

If one or several files that are to be synchronized have been modified on other nodes (not only on the current one), Csync2 reports a conflict with an output similar to the one below:

```
While syncing file /etc/corosync/corosync.conf:
ERROR from peer hex-14: File is also marked dirty here!
Finished with 1 errors.
```

2. If you are sure that the file version on the current node is the "best" one, you can resolve the conflict by forcing this file and resynchronizing:

```
# csync2 -f /etc/corosync/corosync.conf
# csync2 -x
```

For more information on the Csync2 options, run

```
# csync2 -help
```

Note: Pushing synchronization after any changes

Csync2 only pushes changes. It does *not* continuously synchronize files between the machines.

Each time you update files that need to be synchronized, you need to push the changes to the other machines by running `csync2 -xv` on the machine where you did the changes. If you run the command on any of the other machines with unchanged files, nothing happens.

4.8 Bringing the cluster online

Before starting the cluster, make sure passwordless SSH is configured between the nodes. If you did not already configure passwordless SSH before setting up the cluster, you can do so now by using the `ssh` stage of the bootstrap scripts:

1. On the first node, run **`crm cluster init ssh`**.

2. On the rest of the nodes, run **`crm cluster join ssh -c` *NODE1***.

After the initial cluster configuration is done, start the cluster services on all cluster nodes to bring the stack online:

PROCEDURE 4.10: STARTING CLUSTER SERVICES AND CHECKING THE STATUS

1. Log in to an existing node.

2. Start the cluster services on all cluster nodes:

   ```
   # crm cluster start --all
   ```

3. Check the cluster status with the **`crm status`** command. If all nodes are online, the output should be similar to the following:

   ```
   # crm status
   Cluster Summary:
     * Stack: corosync
     * Current DC: alice (version ...) - partition with quorum
     * Last updated: ...
     * Last change:  ... by hacluster via crmd on bob
     * 2 nodes configured
     * 1 resource instance configured

   Node List:
     * Online: [ alice bob ]
   ...
   ```

   This output indicates that the cluster resource manager is started and is ready to manage resources.

After the basic configuration is done and the nodes are online, you can start to configure cluster resources. Use one of the cluster management tools like the crm shell (crmsh) or Hawk2. For more information, see *Section 5.5, "Introduction to crmsh"* or *Section 5.4, "Introduction to Hawk2"*.

II Configuration and administration

5 Configuration and administration basics

The main purpose of an HA cluster is to manage user services. Typical examples of user services are an Apache Web server or a database. From the user's point of view, the services do something specific when ordered to do so. To the cluster, however, they are only resources which may be started or stopped—the nature of the service is irrelevant to the cluster.

This chapter introduces some basic concepts you need to know when administering your cluster. The following chapters show you how to execute the main configuration and administration tasks with each of the management tools SUSE Linux Enterprise High Availability provides.

5.1 Use case scenarios

In general, clusters fall into one of two categories:

- Two-node clusters

- Clusters with more than two nodes. This usually means an odd number of nodes.

Adding also different topologies, different use cases can be derived. The following use cases are the most common:

Two-node cluster in one location

　　Configuration: FC SAN or similar shared storage, layer 2 network.

　　Usage scenario: Embedded clusters that focus on service high availability and not data redundancy for data replication. Such a setup is used for radio stations or assembly line controllers, for example.

Two-node clusters in two locations (most widely used)

　　Configuration: Symmetrical stretched cluster, FC SAN, and layer 2 network all across two locations.

　　Usage scenario: Classic stretched clusters, focus on high availability of services and local data redundancy. For databases and enterprise resource planning. One of the most popular setups.

Odd number of nodes in three locations

Configuration: $2 \times N + 1$ nodes, FC SAN across two main locations. Auxiliary third site with no FC SAN, but acts as a majority maker. Layer 2 network at least across two main locations.

Usage scenario: Classic stretched cluster, focus on high availability of services and data redundancy. For example, databases, enterprise resource planning.

5.2 Quorum determination

Whenever communication fails between one or more nodes and the rest of the cluster, a cluster partition occurs. The nodes can only communicate with other nodes in the same partition and are unaware of the separated nodes. A cluster partition is defined as having quorum (being "quorate") if it has the majority of nodes (or votes). How this is achieved is done by *quorum calculation*. Quorum is a requirement for fencing.

Quorum is not calculated or determined by Pacemaker. Corosync can handle quorum for two-node clusters directly without changing the Pacemaker configuration.

How quorum is calculated is influenced by the following factors:

Number of cluster nodes

To keep services running, a cluster with more than two nodes relies on quorum (majority vote) to resolve cluster partitions. Based on the following formula, you can calculate the minimum number of operational nodes required for the cluster to function:

```
N ≥ C/2 + 1

N = minimum number of operational nodes
C = number of cluster nodes
```

For example, a five-node cluster needs a minimum of three operational nodes (or two nodes which can fail).

We strongly recommend to use either a two-node cluster or an odd number of cluster nodes. Two-node clusters make sense for stretched setups across two sites. Clusters with an odd number of nodes can either be built on one single site or might be spread across three sites.

Corosync configuration

Corosync is a messaging and membership layer, see *Section 5.2.1, "Corosync configuration for two-node clusters"* and *Section 5.2.2, "Corosync configuration for n-node clusters"*.

5.2.1 Corosync configuration for two-node clusters

When using the bootstrap scripts, the Corosync configuration contains a `quorum` section with the following options:

EXAMPLE 5.1: EXCERPT OF COROSYNC CONFIGURATION FOR A TWO-NODE CLUSTER

```
quorum {
    # Enable and configure quorum subsystem (default: off)
    # see also corosync.conf.5 and votequorum.5
    provider: corosync_votequorum
    expected_votes: 2
    two_node: 1
}
```

By default, when `two_node: 1` is set, the `wait_for_all` option is automatically enabled. If `wait_for_all` is not enabled, the cluster should be started on both nodes in parallel. Otherwise, the first node will perform a startup-fencing on the missing second node.

5.2.2 Corosync configuration for n-node clusters

When not using a two-node cluster, we strongly recommend an odd number of nodes for your N-node cluster. For quorum configuration, you have the following options:

- Adding additional nodes with the **crm cluster join** command, or

- Adapting the Corosync configuration manually.

If you adjust `/etc/corosync/corosync.conf` manually, use the following settings:

EXAMPLE 5.2: EXCERPT OF COROSYNC CONFIGURATION FOR AN N-NODE CLUSTER

```
quorum {
    provider: corosync_votequorum  ❶
    expected_votes: N  ❷
    wait_for_all: 1  ❸
}
```

❶   Use the quorum service from Corosync

❷   The number of votes to expect. This parameter can either be provided inside the `quorum` section, or is automatically calculated when the `nodelist` section is available.

❸ Enables the wait for all (WFA) feature. When WFA is enabled, the cluster will be quorate for the first time only after all nodes have become visible. To avoid some startup race conditions, setting `wait_for_all` to `1` may help. For example, in a five-node cluster every node has one vote and thus, `expected_votes` is set to `5`. When three or more nodes are visible to each other, the cluster partition becomes quorate and can start operating.

5.3   Global cluster options

Global cluster options control how the cluster behaves when confronted with certain situations. They are grouped into sets and can be viewed and modified with the cluster management tools like Hawk2 and the `crm` shell.

The predefined values can usually be kept. However, to make key functions of your cluster work as expected, you might need to adjust the following parameters after basic cluster setup:

- *Global option* `no-quorum-policy`

- *Global option* `stonith-enabled`

5.3.1   Global option `no-quorum-policy`

This global option defines what to do when a cluster partition does not have quorum (no majority of nodes is part of the partition).

The following values are available:

`ignore`

> Setting `no-quorum-policy` to `ignore` makes a cluster partition behave like it has quorum, even if it does not. The cluster partition is allowed to issue fencing and continue resource management.
>
> On SLES 11 this was the recommended setting for a two-node cluster. Starting with SLES 12, the value `ignore` is obsolete and must not be used. Based on configuration and conditions, Corosync gives cluster nodes or a single node "quorum"—or not.
>
> For two-node clusters the only meaningful behavior is to always react in case of node loss. The first step should always be to try to fence the lost node.

`freeze`

> If quorum is lost, the cluster partition freezes. Resource management is continued: running resources are not stopped (but possibly restarted in response to monitor events), but no further resources are started within the affected partition.
>
> This setting is recommended for clusters where certain resources depend on communication with other nodes (for example, OCFS2 mounts). In this case, the default setting `no-quorum-policy=stop` is not useful, as it would lead to the following scenario: stopping those resources would not be possible while the peer nodes are unreachable. Instead, an attempt to stop them would eventually time out and cause a `stop failure`, triggering escalated recovery and fencing.

`stop` **(default value)**

> If quorum is lost, all resources in the affected cluster partition are stopped in an orderly fashion.

`suicide`

> If quorum is lost, all nodes in the affected cluster partition are fenced. This option works only in combination with SBD, see *Chapter 13, Storage protection and SBD*.

5.3.2   Global option `stonith-enabled`

This global option defines whether to apply fencing, allowing STONITH devices to shoot failed nodes and nodes with resources that cannot be stopped. By default, this global option is set to `true`, because for normal cluster operation it is necessary to use STONITH devices. According to the default value, the cluster will refuse to start any resources if no STONITH resources have been defined.

If you need to disable fencing for any reasons, set `stonith-enabled` to `false`, but be aware that this has impact on the support status for your product. Furthermore, with `stonith-enabled="false"`, resources like the Distributed Lock Manager (DLM) and all services depending on DLM (such as lvmlockd, GFS2, and OCFS2) will fail to start.

> ❗ Important: No support without STONITH
>
> A cluster without STONITH is not supported.

5.4 Introduction to Hawk2

To configure and manage cluster resources, either use Hawk2, or the crm shell (crmsh) command line utility.

Hawk2's user-friendly Web interface allows you to monitor and administer your High Availability clusters from Linux or non-Linux machines alike. Hawk2 can be accessed from any machine inside or outside of the cluster by using a (graphical) Web browser.

5.4.1 Hawk2 requirements

Before users can log in to Hawk2, the following requirements need to be fulfilled:

`hawk2` **Package**

> The `hawk2` package must be installed on all cluster nodes you want to connect to with Hawk2.

Web browser

> On the machine from which to access a cluster node using Hawk2, you need a (graphical) Web browser (with JavaScript and cookies enabled) to establish the connection.

Hawk2 service

> To use Hawk2, the respective Web service must be started on the node that you want to connect to via the Web interface. See *Procedure 5.1, "Starting Hawk2 services"*.
>
> If you have set up your cluster with the bootstrap scripts provided by the crm shell, the Hawk2 service is already enabled.

Username, group, and password on each cluster node

> Hawk2 users must be members of the `haclient` group. The installation creates a Linux user named `hacluster`, who is added to the `haclient` group.
>
> When using the **crm cluster init** script for setup, a default password is set for the `hacluster` user. Before starting Hawk2, change it to a secure password. If you did not use the **crm cluster init** script, either set a password for the `hacluster` first or create a new user which is a member of the `haclient` group. Do this on every node you will connect to with Hawk2.

Wild card certificate handling

> A wild card certificate is a public key certificate that is valid for multiple sub-domains. For example, a wild card certificate for `*.example.com` secures the domains `www.example.com`, `login.example.com`, and more.

Hawk2 supports wild card certificates as well as conventional certificates. A self-signed default private key and certificate is generated by `/srv/www/hawk/bin/generate-ssl-cert`.

To use your own certificate (conventional or wild card), replace the generated certificate at `/etc/ssl/certs/hawk.pem` with your own.

PROCEDURE 5.1: STARTING HAWK2 SERVICES

1. On the node you want to connect to, open a shell and log in as `root`.

2. Check the status of the service by entering

   ```
   # systemctl status hawk
   ```

3. If the service is not running, start it with

   ```
   # systemctl start hawk
   ```

   If you want Hawk2 to start automatically at boot time, execute the following command:

   ```
   # systemctl enable hawk
   ```

5.4.2 Logging in

The Hawk2 Web interface uses the HTTPS protocol and port `7630`.

Instead of logging in to an individual cluster node with Hawk2, you can configure a floating, virtual IP address (`IPaddr` or `IPaddr2`) as a cluster resource. It does not need any special configuration. It allows clients to connect to the Hawk service no matter which physical node the service is running on.

When setting up the cluster with the bootstrap scripts provided by the crm shell, you will be asked whether to configure a virtual IP for cluster administration.

PROCEDURE 5.2: LOGGING IN TO THE HAWK2 WEB INTERFACE

1. On any machine, start a Web browser and enter the following URL:

   ```
   https://HAWKSERVER:7630/
   ```

   Replace `HAWKSERVER` with the IP address or host name of any cluster node running the Hawk Web service. If a virtual IP address has been configured for cluster administration with Hawk2, replace `HAWKSERVER` with the virtual IP address.

> ## ✎ Note: Certificate warning
>
> If a certificate warning appears when you try to access the URL for the first time, a self-signed certificate is in use. Self-signed certificates are not considered trustworthy by default.
>
> To verify the certificate, ask your cluster operator for the certificate details.
>
> To proceed anyway, you can add an exception in the browser to bypass the warning.
>
> For information on how to replace the self-signed certificate with a certificate signed by an official Certificate Authority, refer to *Replacing the self-signed certificate*.

2. On the Hawk2 login screen, enter the *Username* and *Password* of the `hacluster` user (or of any other user that is a member of the `haclient` group).

3. Click *Log In*.

5.4.3   Hawk2 overview: main elements

After logging in to Hawk2, you will see a navigation bar on the left-hand side and a top-level row with several links on the right-hand side.

> ## ✎ Note: Available functions in Hawk2
>
> By default, users logged in as `root` or `hacluster` have full read-write access to all cluster configuration tasks. However, *Access control lists* (ACLs) can be used to define fine-grained access permissions.
>
> If ACLs are enabled in the CRM, the available functions in Hawk2 depend on the user role and their assigned access permissions. The *History Explorer* in Hawk2 can only be executed by the user `hacluster`.

5.4.3.1   Left navigation bar

Monitoring

- *Status*: Displays the current cluster status at a glance (similar to `crm status` on the crmsh). For details, see *Section 11.1.1, "Monitoring a single cluster"*. If your cluster includes `guest nodes` (nodes that run the `pacemaker_remote` daemon), they are displayed, too. The screen refreshes in near real-time: any status changes for nodes or resources are visible almost immediately.

- *Dashboard*: Allows you to monitor multiple clusters (also located on different sites, in case you have a Geo cluster setup). For details, see *Section 11.1.2, "Monitoring multiple clusters"*. If your cluster includes `guest nodes` (nodes that run the `pacemaker_remote` daemon), they are displayed, too. The screen refreshes in near real-time: any status changes for nodes or resources are visible almost immediately.

Troubleshooting

- *History*: Opens the *History Explorer* from which you can generate cluster reports. For details, see *Section 11.3, "Viewing the cluster history"*.

- *Command Log*: Lists the crmsh commands recently executed by Hawk2.

Configuration

- *Add Resource*: Opens the resource configuration screen. For details, see *Chapter 6, Configuring cluster resources*.

- *Add Constraint*: Opens the constraint configuration screen. For details, see *Chapter 7, Configuring resource constraints*.

- *Wizards*: Allows you to select from several wizards that guide you through the creation of resources for a certain workload, for example, a DRBD block device. For details, see *Section 5.4.6, "Adding resources with the wizard"*.

- *Edit Configuration*: Allows you to edit resources, constraints, node names and attributes, tags, alerts (https://crmsh.github.io/man/#cmdhelp_configure_alert) ↗, and fencing topologies (https://crmsh.github.io/man/#cmdhelp_configure_fencing_topology) ↗.

- *Cluster Configuration*: Allows you to modify global cluster options and resource and operation defaults. For details, see *Section 5.4.4, "Configuring global cluster options"*.

- *Access Control* › *Roles*: Opens a screen where you can create roles for access control lists (sets of rules describing access rights to the CIB). For details, see *Procedure 15.2, "Adding a monitor role with Hawk2"*.

- *Access Control* › *Targets*: Opens a screen where you can create targets (system users) for access control lists and assign roles to them. For details, see *Procedure 15.3, "Assigning a role to a target with Hawk2"*.

5.4.3.2 Top-level row

Hawk2's top-level row shows the following entries:

- *Batch*: Click to switch to batch mode. This allows you to simulate and stage changes and to apply them as a single transaction. For details, see *Section 5.4.7, "Using batch mode"*.

- `USERNAME`: Allows you to set preferences for Hawk2 (for example, the language for the Web interface, or whether to display a warning if STONITH is disabled).

- *Help*: Access the SUSE Linux Enterprise High Availability documentation, read the release notes or report a bug.

- *Logout*: Click to log out.

5.4.4 Configuring global cluster options

Global cluster options control how the cluster behaves when confronted with certain situations. They are grouped into sets and can be viewed and modified with cluster management tools like Hawk2 and crmsh. The predefined values can usually be kept. However, to ensure the key functions of your cluster work correctly, you need to adjust the following parameters after basic cluster setup:

- *Global option* `no-quorum-policy`

- *Global option* `stonith-enabled`

PROCEDURE 5.3: MODIFYING GLOBAL CLUSTER OPTIONS

1. Log in to Hawk2:

   ```
   https://HAWKSERVER:7630/
   ```

2. From the left navigation bar, select *Configuration* › *Cluster Configuration*.

The *Cluster Configuration* screen opens. It displays the global cluster options and their current values.

To display a short description of the parameter on the right-hand side of the screen, hover your mouse over a parameter.
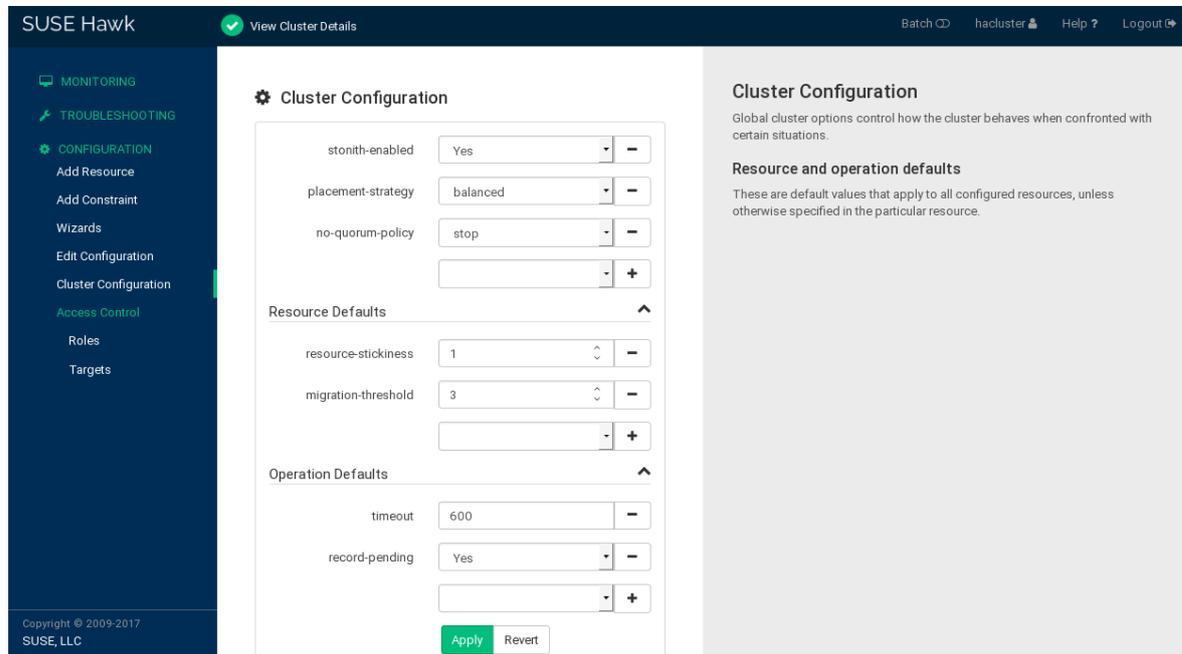
3. Check the values for *no-quorum-policy* and *stonith-enabled* and adjust them, if necessary:

   a. Set *no-quorum-policy* to the appropriate value. See *Section 5.3.1, "Global option* `no-quorum-policy`*"* for more details.

   b. If you need to disable fencing for any reason, set *stonith-enabled* to `no`. By default, it is set to `true`, because using STONITH devices is necessary for normal cluster operation. According to the default value, the cluster will refuse to start any resources if no STONITH resources have been configured.

      ## Important: No Support Without STONITH

      - You must have a node fencing mechanism for your cluster.

      - The global cluster options `stonith-enabled` and `startup-fencing` must be set to `true`. When you change them, you lose support.

c. To remove a parameter from the cluster configuration, click the *Minus* icon next to the parameter. If a parameter is deleted, the cluster will behave as if that parameter had the default value.

d. To add a new parameter to the cluster configuration, choose one from the drop-down box.

4. If you need to change *Resource Defaults* or *Operation Defaults*, proceed as follows:

   a. To adjust a value, either select a different value from the drop-down box or edit the value directly.

   b. To add a new resource default or operation default, choose one from the empty drop-down box and enter a value. If there are default values, Hawk2 proposes them automatically.

   c. To remove a parameter, click the *Minus* icon next to it. If no values are specified for *Resource Defaults* and *Operation Defaults*, the cluster uses the default values that are documented in *Section 6.12, "Resource options (meta attributes)"* and *Section 6.14, "Resource operations"*.

5. Confirm your changes.

5.4.5   Showing the current cluster configuration (CIB)

Sometimes a cluster administrator needs to know the cluster configuration. Hawk2 can show the current configuration in crm shell syntax, as XML and as a graph. To view the cluster configuration in crm shell syntax, from the left navigation bar select *Configuration* › *Edit Configuration* and click *Show*. To show the configuration in raw XML instead, click *XML*. Click *Graph* for a graphical representation of the nodes and resources configured in the CIB. It also shows the relationships between resources.

5.4.6   Adding resources with the wizard

The Hawk2 wizard is a convenient way of setting up simple resources like a virtual IP address or an SBD STONITH resource, for example. It is also useful for complex configurations that include multiple resources, like the resource configuration for a DRBD block device or an Apache Web server. The wizard guides you through the configuration steps and provides information about the parameters you need to enter.

PROCEDURE 5.4: USING THE RESOURCE WIZARD

1. Log in to Hawk2:

   ```
   https://HAWKSERVER:7630/
   ```

2. From the left navigation bar, select *Configuration* › *Wizards*.

3. Expand the individual categories by clicking the arrow down icon next to them and select the desired wizard.

4. Follow the instructions on the screen. After the last configuration step, *Verify* the values you have entered.
   Hawk2 shows which actions it is going to perform and what the configuration looks like. Depending on the configuration, you might be prompted for the `root` password before you can *Apply* the configuration.
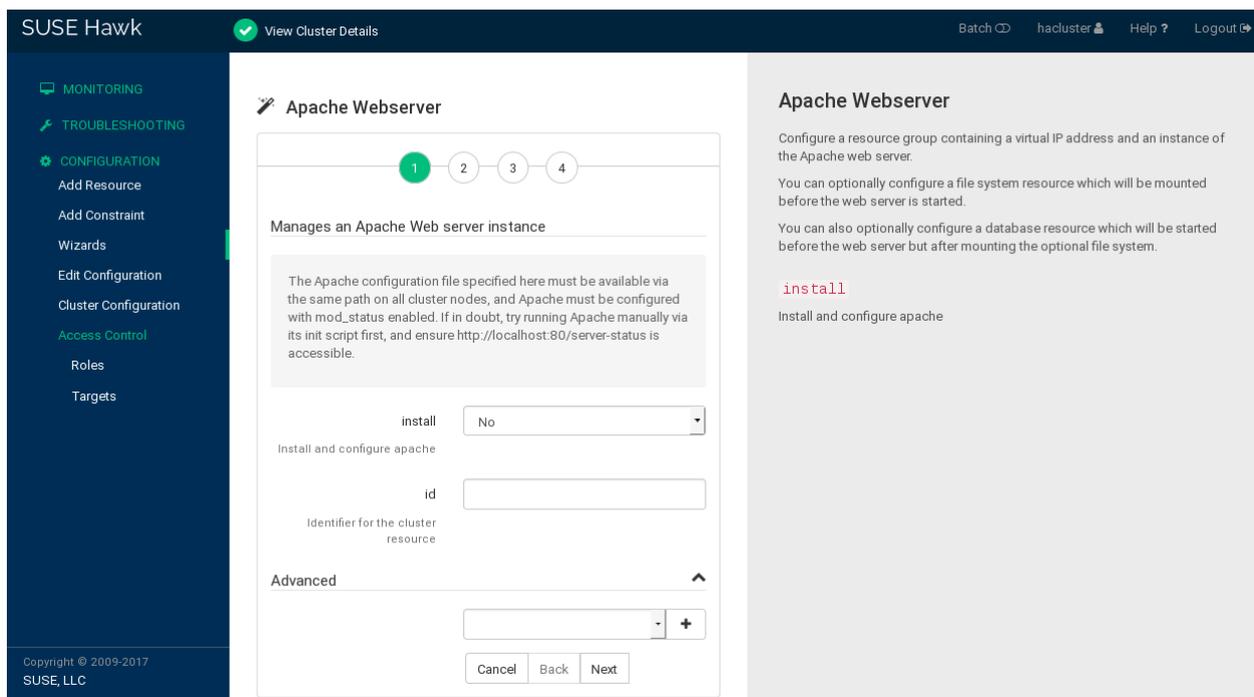
FIGURE 5.2: HAWK2—WIZARD FOR APACHE WEB SERVER

For more information, see *Chapter 6, Configuring cluster resources*.

5.4.7   Using batch mode

Hawk2 provides a *Batch Mode*, including a *cluster simulator*. It can be used for the following:

- Staging changes to the cluster and applying them as a single transaction, instead of having each change take effect immediately.

- Simulating changes and cluster events, for example, to explore potential failure scenarios.

For example, batch mode can be used when creating groups of resources that depend on each other. Using batch mode, you can avoid applying intermediate or incomplete configurations to the cluster.

While batch mode is enabled, you can add or edit resources and constraints or change the cluster configuration. It is also possible to simulate events in the cluster, including nodes going online or offline, resource operations and tickets being granted or revoked. See *Procedure 5.6, "Injecting node, resource or ticket events"* for details.

The *cluster simulator* runs automatically after every change and shows the expected outcome in the user interface. For example, this also means: If you stop a resource while in batch mode, the user interface shows the resource as stopped—while actually, the resource is still running.

> ❗ **Important: Wizards and changes to the live system**
>
> Some wizards include actions beyond mere cluster configuration. When using those wizards in batch mode, any changes that go beyond cluster configuration would be applied to the live system immediately.
>
> Therefore, wizards that require `root` permission cannot be executed in batch mode.

PROCEDURE 5.5: WORKING WITH BATCH MODE

1. Log in to Hawk2:

   ```
   https://HAWKSERVER:7630/
   ```

2. To activate batch mode, select *Batch* from the top-level row.
   An additional bar appears below the top-level row. It indicates that batch mode is active and contains links to actions that you can execute in batch mode.

FIGURE 5.3: HAWK2 BATCH MODE ACTIVATED

3. While batch mode is active, perform any changes to your cluster, like adding or editing resources and constraints, or editing the cluster configuration.
   The changes will be simulated and shown in all screens.

4. To view details of the changes you have made, select *Show* from the batch mode bar. The *Batch Mode* window opens.

   For any configuration changes it shows the difference between the live state and the simulated changes in crmsh syntax: Lines starting with a `-` character represent the current state whereas lines starting with `+` show the proposed state.

5. To inject events or view even more details, see *Procedure 5.6*. Otherwise *Close* the window.

6. Choose to either *Discard* or *Apply* the simulated changes and confirm your choice. This also deactivates batch mode and takes you back to normal mode.

When running in batch mode, Hawk2 also allows you to inject *Node Events* and *Resource Events*.

Node events

Let you change the state of a node. Available states are *online, offline* and *unclean*.

Resource events

Let you change some properties of a resource. For example, you can set an operation (like `start`, `stop`, `monitor`), the node it applies to, and the expected result to be simulated.

Ticket events

Let you test the impact of granting and revoking tickets (used for Geo clusters).

PROCEDURE 5.6: INJECTING NODE, RESOURCE OR TICKET EVENTS

1. Log in to Hawk2:

   ```
   https://HAWKSERVER:7630/
   ```

2. If batch mode is not active yet, click *Batch* at the top-level row to switch to batch mode.

3. In the batch mode bar, click *Show* to open the *Batch Mode* window.

4. To simulate a status change of a node:

   a. Click *Inject* › *Node Event.*

   b. Select the *Node* you want to manipulate and select its target *State*.

   c. Confirm your changes. Your event is added to the queue of events listed in the *Batch Mode* dialog.

5. To simulate a resource operation:

   a. Click *Inject* › *Resource Event.*

b. Select the *Resource* you want to manipulate and select the *Operation* to simulate.

c. If necessary, define an *Interval.*

d. Select the *Node* on which to run the operation and the targeted *Result.* Your event is added to the queue of events listed in the *Batch Mode* dialog.

e. Confirm your changes.

6. To simulate a ticket action:

a. Click *Inject* › *Ticket Event.*

b. Select the *Ticket* you want to manipulate and select the *Action* to simulate.

c. Confirm your changes. Your event is added to the queue of events listed in the *Batch Mode* dialog.

7. The *Batch Mode* dialog (*Figure 5.4*) shows a new line per injected event. Any event listed here is simulated immediately and is reflected on the *Status* screen.
If you have made any configuration changes, too, the difference between the live state and the simulated changes is shown below the injected events.
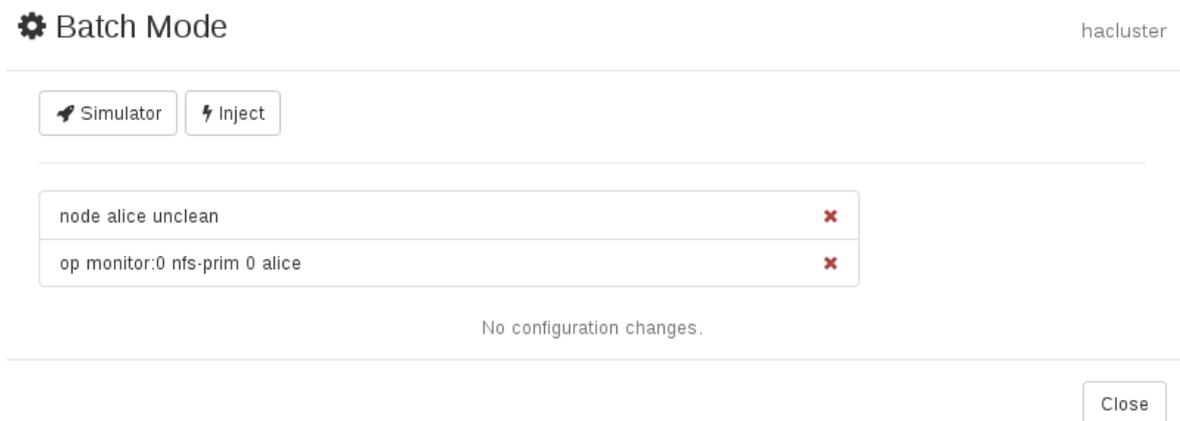
FIGURE 5.4: HAWK2 BATCH MODE—INJECTED INVENTS AND CONFIGURATION CHANGES

8. To remove an injected event, click the *Remove* icon next to it. Hawk2 updates the *Status* screen accordingly.

9. To view more details about the simulation run, click *Simulator* and choose one of the following:

Summary
    Shows a detailed summary.

CIB (in)/CIB (out)
    CIB (in) shows the initial CIB state. *CIB (out)* shows what the CIB would look like after the transition.

Transition graph
    Shows a graphical representation of the transition.

Transition
    Shows an XML representation of the transition.

10. If you have reviewed the simulated changes, close the *Batch Mode* window.

11. To leave batch mode, either *Apply* or *Discard* the simulated changes.

5.5 Introduction to crmsh

To configure and manage cluster resources, either use the crm shell (crmsh) command line utility or Hawk2, a Web-based user interface.

This section introduces the command line tool `crm`. The `crm` command has several subcommands which manage resources, CIBs, nodes, resource agents, and others. It offers a thorough help system with embedded examples. All examples follow a naming convention described in *Appendix B*.

Events are logged to `/var/log/crmsh/crmsh.log`.

Note: User privileges

Sufficient privileges are necessary to manage a cluster. The `crm` command and its sub-commands need to be run as one of the following users:

- The `root` user or a user with **sudo** privileges. These users have full privileges for `crm cluster init`, `crm cluster join`, `crm report`, the cluster CIB, and other operations.

- The CRM owner user (typically the user `hacluster`). This user can make changes to the cluster CIB, but has limited privileges for `crm report`.

However, the `user` option allows you to run `crm` and its subcommands as a regular (unprivileged) user and to change its ID using **sudo** when necessary. For example, with the following command `crm` uses `hacluster` as the privileged user ID:

```
# crm options user hacluster
```

You must set up `/etc/sudoers` so that **sudo** does not ask for a password.

Tip: Interactive crm prompt

By using crm without arguments (or with only one sublevel as argument), the crm shell enters the interactive mode. This mode is indicated by the following prompt:

```
crm(live/HOSTNAME)
```

For readability reasons, we omit the host name in the interactive crm prompts in our documentation. We only include the host name if you need to run the interactive shell on a specific node, like alice for example:

```
crm(live/alice)
```

5.5.1   Getting help

Help can be accessed in several ways:

- To output the usage of **crm** and its command line options:

  ```
  # crm --help
  ```

- To give a list of all available commands:

  ```
  # crm help
  ```

- To access other help sections, not only the command reference:

  ```
  # crm help topics
  ```

- To view the extensive help text of the **configure** subcommand:

  ```
  # crm configure help
  ```

- To print the syntax, its usage, and examples of the **group** subcommand of **configure**:

  ```
  # crm configure help group
  ```

  This is the same:

  ```
  # crm help configure group
  ```

Almost all output of the **help** subcommand (do not mix it up with the `--help` option) opens a text viewer. This text viewer allows you to scroll up or down and read the help text more comfortably. To leave the text viewer, press the `Q` key.

> 💡 **Tip: Use tab completion in Bash and interactive shell**
>
> The crmsh supports full tab completion in Bash directly, not only for the interactive shell. For example, typing `crm help config` `→|` completes the word like in the interactive shell.

5.5.2 Executing crmsh's subcommands

The `crm` command itself can be used in the following ways:

- **Directly:** Concatenate all subcommands to `crm`, press `Enter` and you see the output immediately. For example, enter `crm help ra` to get information about the `ra` subcommand (resource agents).

  It is possible to abbreviate subcommands if they are unique. For example, you can shorten `status` as `st` and crmsh knows what you mean.

  Another feature is to shorten parameters. Usually, you add parameters through the `params` keyword. You can leave out the `params` section if it is the first and only section. For example, this line:

  ```
  # crm primitive ipaddr IPaddr2 params ip=192.168.0.55
  ```

  is equivalent to this line:

  ```
  # crm primitive ipaddr IPaddr2 ip=192.168.0.55
  ```

- **As crm shell script:** Crm shell scripts contain subcommands of `crm`. For more information, see *Section 5.5.4, "Using crmsh's shell scripts"*.

- **As crmsh cluster scripts:** These are a collection of metadata, references to RPM packages, configuration files, and crmsh subcommands bundled under a single, yet descriptive name. They are managed through the `crm script` command.

  Do not confuse them with crmsh shell scripts: although both share some common objectives, the crm shell scripts only contain subcommands whereas cluster scripts incorporate much more than a simple enumeration of commands. For more information, see *Section 5.5.5, "Using crmsh's cluster scripts"*.

- **Interactive as internal shell:** Type `crm` to enter the internal shell. The prompt changes to `crm(live)`. With `help` you can get an overview of the available subcommands. As the internal shell has different levels of subcommands, you can "enter" one by typing this subcommand and press `Enter`.

  For example, if you type `resource` you enter the resource management level. Your prompt changes to `crm(live)resource#`. To leave the internal shell, use the command `quit`. If you need to go one level back, use `back`, `up`, `end`, or `cd`.

  You can enter the level directly by typing `crm` and the respective subcommand without any options and press `Enter`.

The internal shell supports also tab completion for subcommands and resources. Type the beginning of a command, press `→|` and `crm` completes the respective object.

crmsh also supports synchronous command execution. Use the `-w` option to activate it. If you have started `crm` without `-w`, you can enable it later with the user preference's `wait` set to `yes` (`options wait yes`). If this option is enabled, `crm` waits until the transition is finished. Whenever a transaction is started, dots are printed to indicate progress. Synchronous command execution is only applicable for commands like `resource start`.

> **Note: Differentiate between management and configuration subcommands**
>
> The `crm` tool has management capability (the subcommands `resource` and `node`) and can be used for configuration (`cib`, `configure`).

The following subsections give you an overview of important aspects of the `crm` tool.

5.5.3   Displaying information about OCF resource agents

As you need to deal with resource agents in your cluster configuration all the time, the `crm` tool contains the `ra` command. Use it to show information about resource agents and to manage them (for additional information, see also *Section 6.2, "Supported resource agent classes"*):

```
# crm ra
crm(live)ra#
```

The command `classes` lists all classes and providers:

```
crm(live)ra# classes
 lsb
 ocf / heartbeat linbit lvm2 ocfs2 pacemaker
 service
 stonith
 systemd
```

To get an overview of all available resource agents for a class (and provider) use the `list` command:

```
crm(live)ra# list ocf
```

```
AoEtarget          AudibleAlarm      CTDB            ClusterMon
Delay              Dummy             EvmsSCC         Evmsd
Filesystem         HealthCPU         HealthSMART     ICP
IPaddr             IPaddr2           IPsrcaddr       IPv6addr
LVM                LinuxSCSI         MailTo          ManageRAID
ManageVE           Pure-FTPd         Raid1           Route
SAPDatabase        SAPInstance       SendArp         ServeRAID
...
```

An overview of a resource agent can be viewed with `info`:

```
crm(live)ra# info ocf:linbit:drbd
This resource agent manages a DRBD* resource
as a master/slave resource. DRBD is a shared-nothing replicated storage
device. (ocf:linbit:drbd)

Master/Slave OCF Resource Agent for DRBD

Parameters (* denotes required, [] the default):

drbd_resource* (string): drbd resource name
    The name of the drbd resource from the drbd.conf file.

drbdconf (string, [/etc/drbd.conf]): Path to drbd.conf
    Full path to the drbd.conf file.

Operations' defaults (advisory minimum):

    start          timeout=240
    promote        timeout=90
    demote         timeout=90
    notify         timeout=90
    stop           timeout=100
    monitor_Slave_0 interval=20 timeout=20 start-delay=1m
    monitor_Master_0 interval=10 timeout=20 start-delay=1m
```

Leave the viewer by pressing `Q`.

💡 Tip: Use `crm` directly

In the former example we used the internal shell of the `crm` command. However, you do not necessarily need to use it. You get the same results if you add the respective subcommands to `crm`. For example, you can list all the OCF resource agents by entering `crm ra list ocf` in your shell.

5.5.4 Using crmsh's shell scripts

The crmsh shell scripts provide a convenient way to enumerate crmsh subcommands into a file. This makes it easy to comment specific lines or to replay them later. Keep in mind that a crmsh shell script can contain *only crmsh subcommands*. Any other commands are not allowed.

Before you can use a crmsh shell script, create a file with specific commands. For example, the following file prints the status of the cluster and gives a list of all nodes:

EXAMPLE 5.3: A SIMPLE CRMSH SHELL SCRIPT

```
# A small example file with some crm subcommands
status
node list
```

Any line starting with the hash symbol (`#`) is a comment and is ignored. If a line is too long, insert a backslash (`\`) at the end and continue in the next line. We recommend that you indent lines belonging to a certain subcommand to improve readability.

To use this script, use one of the following methods:

```
# crm -f example.cli
# crm < example.cli
```

5.5.5 Using crmsh's cluster scripts

Collecting information from all cluster nodes and deploying any changes is a key cluster administration task. Instead of performing the same procedures manually on different nodes (which is error-prone), you can use the crmsh cluster scripts.

Do not confuse them with the *crmsh shell scripts*, which are explained in *Section 5.5.4, "Using crmsh's shell scripts"*.

In contrast to crmsh shell scripts, cluster scripts performs additional tasks like:

- Installing software that is required for a specific task.

- Creating or modifying any configuration files.

- Collecting information and reporting potential problems with the cluster.

- Deploying the changes to all nodes.

crmsh cluster scripts do not replace other tools for managing clusters—they provide an integrated way to perform the above tasks across the cluster. Find detailed information at https://crmsh.github.io/scripts/ ↗.

5.5.5.1 Usage

To get a list of all available cluster scripts, run:

```
# crm script list
```

To view the components of a script, use the `show` command and the name of the cluster script, for example:

```
# crm script show mailto
mailto (Basic)
MailTo

 This is a resource agent for MailTo. It sends email to a sysadmin
whenever  a takeover occurs.

1. Notifies recipients by email in the event of resource takeover

  id (required)  (unique)
      Identifier for the cluster resource
  email (required)
      Email address
  subject
      Subject
```

The output of `show` contains a title, a short description, and a procedure. Each procedure is divided into a series of steps, performed in the given order.

Each step contains a list of required and optional parameters, along with a short description and its default value.

Each cluster script understands a set of common parameters. These parameters can be passed to any script:

TABLE 5.1: COMMON PARAMETERS

Parameter	Argument	Description
action	INDEX	If set, only execute a single action (index, as returned by verify)

Parameter	Argument	Description
dry_run	*BOOL*	If set, simulate execution only (default: no)
nodes	*LIST*	List of nodes to execute the script for
port	*NUMBER*	Port to connect to
statefile	*FILE*	When single-stepping, the state is saved in the given file
sudo	*BOOL*	If set, crm will prompt for a sudo password and use sudo where appropriate (default: no)
timeout	*NUMBER*	Execution timeout in seconds (default: 600)
user	*USER*	Run script as the given user

5.5.5.2 Verifying and running a cluster script

Before running a cluster script, review the actions that it will perform and verify its parameters to avoid problems. A cluster script can potentially perform a series of actions and may fail for various reasons. Thus, verifying your parameters before running it helps to avoid problems.

For example, the `mailto` resource agent requires a unique identifier and an e-mail address. To verify these parameters, run:

```
# crm script verify mailto id=sysadmin email=tux@example.org
1. Ensure mail package is installed

        mailx


2. Configure cluster resources

        primitive sysadmin MailTo
                email="tux@example.org"
                op start timeout="10"
                op stop timeout="10"
```

```
            op monitor interval="10" timeout="10"

        clone c-sysadmin sysadmin
```

The **verify** command prints the steps and replaces any placeholders with your given parameters. If **verify** finds any problems, it reports it. If everything is OK, replace the **verify** command with **run**:

```
# crm script run mailto id=sysadmin email=tux@example.org
INFO: MailTo
INFO: Nodes: alice, bob
OK: Ensure mail package is installed
OK: Configure cluster resources
```

Check whether your resource is integrated into your cluster with **crm status**:

```
# crm status
[...]
 Clone Set: c-sysadmin [sysadmin]
     Started: [ alice bob ]
```

5.5.6   Using configuration templates

Note: Deprecation notice

The use of configuration templates is deprecated and will be removed in the future. Configuration templates will be replaced by cluster scripts. See *Section 5.5.5, "Using crmsh's cluster scripts"*.

Configuration templates are ready-made cluster configurations for crmsh. Do not confuse them with the *resource templates* (as described in *Section 6.8.2, "Creating resource templates with crmsh"*). Those are templates for the *cluster* and not for the crm shell.

Configuration templates require minimum effort to be tailored to the particular user's needs. Whenever a template creates a configuration, warning messages give hints which can be edited later for further customization.

The following procedure shows how to create a simple yet functional Apache configuration:

1. Log in as **root** and start the **crm** interactive shell:

   ```
   # crm configure
   ```

2. Create a new configuration from a configuration template:

   a. Switch to the **template** subcommand:

   ```
   crm(live)configure# template
   ```

   b. List the available configuration templates:

   ```
   crm(live)configure template# list templates
   gfs2-base    filesystem  virtual-ip  apache   clvm     ocfs2    gfs2
   ```

   c. Decide which configuration template you need. As we need an Apache configuration, we select the apache template and name it g-intranet:

   ```
   crm(live)configure template# new g-intranet apache
   INFO: pulling in template apache
   INFO: pulling in template virtual-ip
   ```

3. Define your parameters:

   a. List the configuration you have created:

   ```
   crm(live)configure template# list
   g-intranet
   ```

   b. Display the minimum required changes that need to be filled out by you:

   ```
   crm(live)configure template# show
   ERROR: 23: required parameter ip not set
   ERROR: 61: required parameter id not set
   ERROR: 65: required parameter configfile not set
   ```

   c. Invoke your preferred text editor and fill out all lines that have been displayed as errors in *Step 3.b*:

   ```
   crm(live)configure template# edit
   ```

4. Show the configuration and check whether it is valid (bold text depends on the configuration you have entered in *Step 3.c*):

```
crm(live)configure template# show
primitive virtual-ip ocf:heartbeat:IPaddr \
    params ip="192.168.1.101"
primitive apache apache \
    params configfile="/etc/apache2/httpd.conf"
```

Using configuration templates   SLE HA 15 SP5

```
      monitor apache 120s:60s
group g-intranet \
      apache virtual-ip
```

5. Apply the configuration:

```
crm(live)configure template# apply
crm(live)configure# cd ..
crm(live)configure# show
```

6. Submit your changes to the CIB:

```
crm(live)configure# commit
```

It is possible to simplify the commands even more, if you know the details. The above procedure can be summarized with the following command on the shell:

```
# crm configure template \
   new g-intranet apache params \
   configfile="/etc/apache2/httpd.conf" ip="192.168.1.101"
```

If you are inside your internal `crm` shell, use the following command:

```
crm(live)configure template# new intranet apache params \
   configfile="/etc/apache2/httpd.conf" ip="192.168.1.101"
```

However, the previous command only creates its configuration from the configuration template. It does not apply nor commit it to the CIB.

5.5.7   Testing with shadow configuration

A shadow configuration is used to test different configuration scenarios. If you have created several shadow configurations, you can test them one by one to see the effects of your changes. The usual process looks like this:

1. Log in as `root` and start the `crm` interactive shell:

```
# crm configure
```

2. Create a new shadow configuration:

```
crm(live)configure# cib new myNewConfig
INFO: myNewConfig shadow CIB created
```

If you omit the name of the shadow CIB, a temporary name `@tmp@` is created.

3. To copy the current live configuration into your shadow configuration, use the following command, otherwise skip this step:

```
crm(myNewConfig)# cib reset myNewConfig
```

The previous command makes it easier to modify any existing resources later.

4. Make your changes as usual. After you have created the shadow configuration, all changes go there. To save all your changes, use the following command:

```
crm(myNewConfig)# commit
```

5. If you need the live cluster configuration again, switch back with the following command:

```
crm(myNewConfig)configure# cib use live
crm(live)#
```

5.5.8   Debugging your configuration changes

Before loading your configuration changes back into the cluster, it is recommended to review your changes with `ptest`. The `ptest` command can show a diagram of actions that will be induced by committing the changes. You need the `graphviz` package to display the diagrams. The following example is a transcript, adding a monitor operation:

```
# crm configure
crm(live)configure# show fence-bob
primitive fence-bob stonith:apcsmart \
        params hostlist="bob"
crm(live)configure# monitor fence-bob 120m:60s
crm(live)configure# show changed
primitive fence-bob stonith:apcsmart \
        params hostlist="bob" \
        op monitor interval="120m" timeout="60s"
crm(live)configure# ptest
crm(live)configure# commit
```

5.5.9   Cluster diagram

To output a cluster diagram, use the command `crm configure graph`. It displays the current configuration on its current window, therefore requiring X11.

If you prefer Scalable Vector Graphics (SVG), use the following command:

```
# crm configure graph dot config.svg svg
```

5.5.10   Managing Corosync configuration

Corosync is the underlying messaging layer for most HA clusters. The `corosync` subcommand provides commands for editing and managing the Corosync configuration.

For example, to list the status of the cluster, use `status`:

```
# crm corosync status
Printing ring status.
Local node ID 175704363
RING ID 0
        id      = 10.121.9.43
        status  = ring 0 active with no faults
Quorum information
------------------
Date:             Thu May  8 16:41:56 2014
Quorum provider:  corosync_votequorum
Nodes:            2
Node ID:          175704363
Ring ID:          4032
Quorate:          Yes

Votequorum information
----------------------
Expected votes:   2
Highest expected: 2
Total votes:      2
Quorum:           2
Flags:            Quorate

Membership information
----------------------
    Nodeid      Votes Name
 175704363          1 alice.example.com (local)
 175704619          1 bob.example.com
```

The `diff` command compares the Corosync configuration on all nodes (if not stated otherwise) and prints the difference between them:

```
# crm corosync diff
```

```
--- bob
+++ alice
@@ -46,2 +46,2 @@
-        expected_votes: 2
-        two_node: 1
+        expected_votes: 1
+        two_node: 0
```

For more details, see https://crmsh.nongnu.org/crm.8.html#cmdhelp_corosync ↗.

5.5.11   Setting passwords independent of `cib.xml`

If your cluster configuration contains sensitive information, such as passwords, it should be stored in local files. That way, these parameters will never be logged or leaked in support reports.

Before using **secret**, run the **show** command to get an overview of all your resources:

```
# crm configure show
primitive mydb mysql \
   params replication_user=admin ...
```

To set a password for the above mydb resource, use the following commands:

```
# crm resource secret mydb set passwd linux
INFO: syncing /var/lib/heartbeat/lrm/secrets/mydb/passwd to [your node list]
```

You can get the saved password back with:

```
# crm resource secret mydb show passwd
linux
```

The parameters need to be synchronized between nodes; the **crm resource secret** command takes care of that. We highly recommend that you only use this command to manage secret parameters.

5.6   For more information

https://crmsh.github.io/ ↗

> Home page of the crm shell (crmsh), the advanced command line interface for High Availability cluster management.

https://crmsh.github.io/documentation ↗

Holds several documents about the crm shell, including a *Getting Started* tutorial for basic cluster setup with crmsh and the comprehensive *Manual* for the crm shell. The latter is available at https://crmsh.github.io/man-2.0/ ↗ . Find the tutorial at https://crmsh.github.io/start-guide/ ↗ .

https://clusterlabs.org/ ↗

Home page of Pacemaker, the cluster resource manager shipped with SUSE Linux Enterprise High Availability.

https://www.clusterlabs.org/pacemaker/doc/ ↗

Holds several comprehensive manuals and some shorter documents explaining general concepts. For example:

- *Pacemaker Explained*: Contains comprehensive and detailed information for reference.

- *Colocation Explained*

- *Ordering Explained*

6 Configuring cluster resources

As a cluster administrator, you need to create cluster resources for every resource or application you run on servers in your cluster. Cluster resources can include Web sites, e-mail servers, databases, file systems, virtual machines and any other server-based applications or services you want to make available to users at all times.

6.1 Types of resources

The following types of resources can be created:

Primitives

A primitive resource, the most basic type of resource.

Groups

Groups contain a set of resources that need to be located together, started sequentially and stopped in the reverse order.

Clones

Clones are resources that can be active on multiple hosts. Any resource can be cloned, provided the respective resource agent supports it.

Promotable clones (also known as multi-state resources) are a special type of clone resource that can be promoted.

6.2 Supported resource agent classes

For each cluster resource you add, you need to define the standard that the resource agent conforms to. Resource agents abstract the services they provide and present an accurate status to the cluster, which allows the cluster to be non-committal about the resources it manages. The cluster relies on the resource agent to react appropriately when given a start, stop or monitor command.

Typically, resource agents come in the form of shell scripts. SUSE Linux Enterprise High Availability supports the following classes of resource agents:

Open Cluster Framework (OCF) resource agents

OCF RA agents are best suited for use with High Availability, especially when you need promotable clone resources or special monitoring abilities. The agents are generally located in `/usr/lib/ocf/resource.d/`*`provider`*`/`. Their functionality is similar to that of LSB scripts. However, the configuration is always done with environmental variables that allow them to accept and process parameters easily. OCF specifications have strict definitions of which exit codes must be returned by actions. See *Section 10.3, "OCF return codes and failure recovery"*. The cluster follows these specifications exactly.

All OCF Resource Agents are required to have at least the actions `start`, `stop`, `status`, `monitor` and `meta-data`. The `meta-data` action retrieves information about how to configure the agent. For example, to know more about the `IPaddr` agent by the provider `heartbeat`, use the following command:

```
OCF_ROOT=/usr/lib/ocf /usr/lib/ocf/resource.d/heartbeat/IPaddr meta-data
```

The output is information in XML format, including several sections (general description, available parameters, available actions for the agent).

Alternatively, use the crmsh to view information on OCF resource agents. For details, see *Section 5.5.3, "Displaying information about OCF resource agents"*.

Linux Standards Base (LSB) scripts

LSB resource agents are generally provided by the operating system/distribution and are found in `/etc/init.d`. To be used with the cluster, they must conform to the LSB init script specification. For example, they must have several actions implemented, which are, at minimum, `start`, `stop`, `restart`, `reload`, `force-reload` and `status`. For more information, see https://refspecs.linuxbase.org/LSB_4.1.0/LSB-Core-generic/LSB-Core-generic/iniscrptact.html .

The configuration of those services is not standardized. If you intend to use an LSB script with High Availability, make sure that you understand how the relevant script is configured. You can often find information about this in the documentation of the relevant package in `/usr/share/doc/packages/`*`PACKAGENAME`*.

systemd

Pacemaker can manage systemd services if they are present. Instead of init scripts, systemd has unit files. Generally, the services (or unit files) are provided by the operating system. In case you want to convert existing init scripts, find more information at https://0pointer.de/blog/projects/systemd-for-admins-3.html ↗.

Service

There are currently many types of system services that exist in parallel: `LSB` (belonging to System V init), `systemd` and (in some distributions) `upstart`. Therefore, Pacemaker supports a special alias that figures out which one applies to a given cluster node. This is particularly useful when the cluster contains a mix of systemd, upstart and LSB services. Pacemaker tries to find the named service in the following order: as an LSB (SYS-V) init script, a systemd unit file or an Upstart job.

Nagios

Monitoring plug-ins (formerly called Nagios plug-ins) allow to monitor services on remote hosts. Pacemaker can do remote monitoring with the monitoring plug-ins if they are present. For detailed information, see *Section 9.1, "Monitoring services on remote hosts with monitoring plug-ins"*.

STONITH (fencing) resource agents

This class is used exclusively for fencing related resources. For more information, see *Chapter 12, Fencing and STONITH*.

The agents supplied with SUSE Linux Enterprise High Availability are written to OCF specifications.

6.3   Timeout values

Timeouts values for resources can be influenced by the following parameters:

- `op_defaults` (global timeout for operations),

- a specific timeout value defined in a resource template,

- a specific timeout value defined for a resource.

Note: Priority of values

If a *specific* value is defined for a resource, it takes precedence over the global default. A specific value for a resource also takes precedence over a value that is defined in a resource template.

Getting timeout values right is very important. Setting them too low results in a lot of (unnecessary) fencing operations for the following reasons:

1. If a resource runs into a timeout, it fails and the cluster tries to stop it.

2. If stopping the resource also fails (for example, because the timeout for stopping is set too low), the cluster fences the node. It considers the node where this happens to be out of control.

You can adjust the global default for operations and set any specific timeout values with both crmsh and Hawk2. The best practice for determining and setting timeout values is as follows:

PROCEDURE 6.1: DETERMINING TIMEOUT VALUES

1. Check how long it takes your resources to start and stop (under load).

2. If needed, add the `op_defaults` parameter and set the (default) timeout value accordingly:

   a. For example, set `op_defaults` to `60` seconds:

   ```
   crm(live)configure# op_defaults timeout=60
   ```

   b. For resources that need longer periods of time, define individual timeout values.

3. When configuring operations for a resource, add separate `start` and `stop` operations. When configuring operations with Hawk2, it provides useful timeout proposals for those operations.

6.4 Creating primitive resources

Before you can use a resource in the cluster, it must be set up. For example, to use an Apache server as a cluster resource, set up the Apache server first and complete the Apache configuration before starting the respective resource in your cluster.

If a resource has specific environment requirements, make sure they are present and identical on all cluster nodes. This kind of configuration is not managed by SUSE Linux Enterprise High Availability. You must do this yourself.

You can create primitive resources using either Hawk2 or crmsh.

Note: Do not touch services managed by the cluster

When managing a resource with SUSE Linux Enterprise High Availability, the same resource must not be started or stopped otherwise (outside of the cluster, for example manually or on boot or reboot). The High Availability software is responsible for all service start or stop actions.

If you need to execute testing or maintenance tasks after the services are already running under cluster control, make sure to put the resources, nodes, or the whole cluster into maintenance mode before you touch any of them manually. For details, see *Section 28.2, "Different options for maintenance tasks"*.

Important: Resource IDs and node names

Cluster resources and cluster nodes should be named differently. Otherwise, Hawk2 will fail.

6.4.1 Creating primitive resources with Hawk2

To create the most basic type of resource, proceed as follows:

PROCEDURE 6.2: ADDING A PRIMITIVE RESOURCE WITH HAWK2

1. Log in to Hawk2:

   ```
   https://HAWKSERVER:7630/
   ```

2. From the left navigation bar, select *Configuration* › *Add Resource* › *Primitive*.

3. Enter a unique *Resource ID*.

4. If a resource template exists on which you want to base the resource configuration, select the respective *Template*.

5. Select the resource agent *Class* you want to use: `lsb`, `ocf`, `service`, `stonith` or `sys-temd`. For more information, see *Section 6.2, "Supported resource agent classes"*.

6. If you selected `ocf` as class, specify the *Provider* of your OCF resource agent. The OCF specification allows multiple vendors to supply the same resource agent.

7. From the *Type* list, select the resource agent you want to use (for example, *IPaddr* or *Filesystem*). A short description for this resource agent is displayed.

> ⊘ Note
>
> The selection you get in the *Type* list depends on the *Class* (and for OCF resources also on the *Provider*) you have chosen.
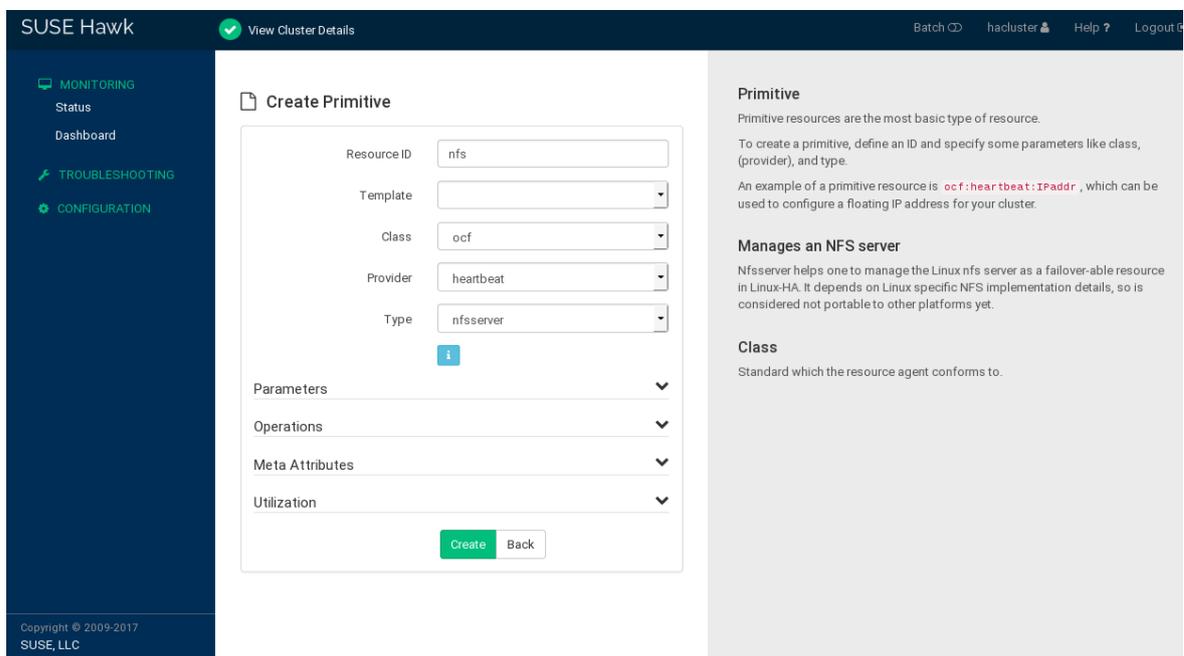
FIGURE 6.1: HAWK2—PRIMITIVE RESOURCE

8. After you have specified the resource basics, Hawk2 shows the following categories. Either keep these categories as suggested by Hawk2, or edit them as required.

Parameters (instance attributes)

Determines which instance of a service the resource controls. When creating a resource, Hawk2 automatically shows any required parameters. Edit them to get a valid resource configuration.

For more information, refer to *Section 6.13, "Instance attributes (parameters)"*.

Operations

> Needed for resource monitoring. When creating a resource, Hawk2 displays the most important resource operations (`monitor`, `start` and `stop`).
>
> For more information, refer to *Section 6.14, "Resource operations"*.

Meta attributes

> Tells the CRM how to treat a specific resource. When creating a resource, Hawk2 automatically lists the important meta attributes for that resource (for example, the `target-role` attribute that defines the initial state of a resource. By default, it is set to `Stopped`, so the resource will not start immediately).
>
> For more information, refer to *Section 6.12, "Resource options (meta attributes)"*.

Utilization

> Tells the CRM what capacity a certain resource requires from a node.
>
> For more information, refer to *Section 7.10.1, "Placing resources based on their load impact with Hawk2"*.

9. Click *Create* to finish the configuration. A message at the top of the screen shows if the action has been successful.

6.4.2   Creating primitive resources with crmsh

PROCEDURE 6.3: ADDING A PRIMITIVE RESOURCE WITH CRMSH

1. Log in as `root` and start the **crm** tool:

```
# crm configure
```

2. Configure a primitive IP address:

```
crm(live)configure# primitive myIP IPaddr \
      params ip=127.0.0.99 op monitor interval=60s
```

The previous command configures a "primitive" with the name `myIP`. You need to choose a class (here `ocf`), provider (`heartbeat`), and type (`IPaddr`). Furthermore, this primitive expects other parameters like the IP address. Change the address to your setup.

3. Display and review the changes you have made:

```
crm(live)configure# show
```

4. Commit your changes to take effect:

```
crm(live)configure# commit
```

6.5 Creating resource groups

Some cluster resources depend on other components or resources. They require that each component or resource starts in a specific order and runs together on the same server as the resources it depends on. To simplify this configuration, you can use cluster resource groups.

You can create resource groups using either Hawk2 or crmsh.

EXAMPLE 6.1: RESOURCE GROUP FOR A WEB SERVER

An example of a resource group would be a Web server that requires an IP address and a file system. In this case, each component is a separate resource that is combined into a cluster resource group. The resource group would run on one or more servers. In case of a software or hardware malfunction, the group would fail over to another server in the cluster, similar to an individual cluster resource.
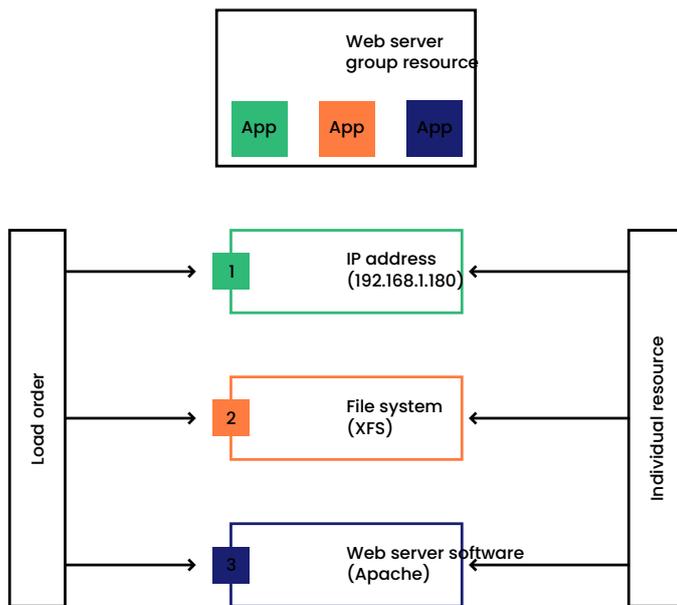
FIGURE 6.2: GROUP RESOURCE

Groups have the following properties:

Starting and stopping

Resources are started in the order they appear in and stopped in reverse order.

Dependency

If a resource in the group cannot run anywhere, then none of the resources located after that resource in the group is allowed to run.

Contents

Groups may only contain a collection of primitive cluster resources. Groups must contain at least one resource, otherwise the configuration is not valid. To refer to the child of a group resource, use the child's ID instead of the group's ID.

Constraints

Although it is possible to reference the group's children in constraints, it is usually preferable to use the group's name instead.

Stickiness

Stickiness is additive in groups. Every *active* member of the group contributes 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), the group as a whole will prefer its current location with a score of `500`.

Resource monitoring

To enable resource monitoring for a group, you must configure monitoring separately for each resource in the group that you want monitored.

6.5.1   Creating resource groups with Hawk2

📝 Note: Empty groups

Groups must contain at least one resource, otherwise the configuration is not valid. While creating a group, Hawk2 allows you to create more primitives and add them to the group.

PROCEDURE 6.4: ADDING A RESOURCE GROUP WITH HAWK2

1. Log in to Hawk2:

   ```
   https://HAWKSERVER:7630/
   ```

2. From the left navigation bar, select *Configuration* › *Add Resource* › *Group*.

3. Enter a unique *Group ID*.

4. To define the group members, select one or multiple entries in the list of *Children*. Re-sort group members by dragging and dropping them into the order you want by using the "handle" icon on the right.

5. If needed, modify or add *Meta Attributes*.

6. Click *Create* to finish the configuration. A message at the top of the screen shows if the action has been successful.
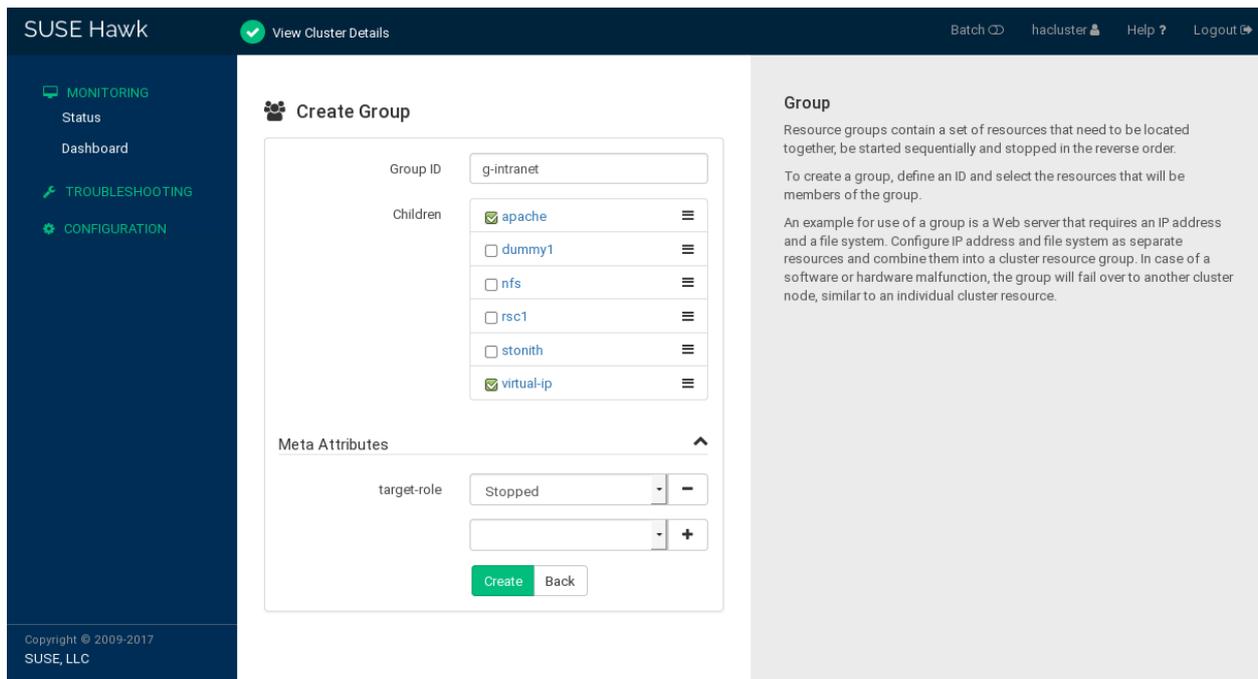
FIGURE 6.3: HAWK2—RESOURCE GROUP

6.5.2 Creating a resource group with crmsh

The following example creates two primitives (an IP address and an e-mail resource).

PROCEDURE 6.5: ADDING A RESOURCE GROUP WITH CRMSH

1. Run the `crm` command as system administrator. The prompt changes to `crm(live)`.

2. Configure the primitives:

```
crm(live)# configure
crm(live)configure# primitive Public-IP ocf:heartbeat:IPaddr2 \
    params ip=1.2.3.4 \
```

Creating a resource group with crmsh   SLE HA 15 SP5

```
        op monitor interval=10s
crm(live)configure# primitive Email systemd:postfix \
        op monitor interval=10s
```

3. Group the primitives with their relevant identifiers in the correct order:

```
crm(live)configure# group g-mailsvc Public-IP Email
```

6.6   Creating clone resources

You may want certain resources to run simultaneously on multiple nodes in your cluster. To do this, you must configure a resource as a clone. Examples of resources that might be configured as clones include cluster file systems like OCFS2. You can clone any resource provided. This is supported by the resource's Resource Agent. Clone resources may even be configured differently depending on which nodes they are hosted.

There are three types of resource clones:

Anonymous clones

These are the simplest type of clones. They behave identically anywhere they are running. Because of this, there can only be one instance of an anonymous clone active per machine.

Globally unique clones

These resources are distinct entities. An instance of the clone running on one node is not equivalent to another instance on another node; nor would any two instances on the same node be equivalent.

Promotable clones (multi-state resources)

Active instances of these resources are divided into two states: active and passive. These are also sometimes called primary and secondary. Promotable clones can be either anonymous or globally unique. For more information, see *Section 6.7, "Creating promotable clones (multi-state resources)"*.

Clones must contain exactly one group or one regular resource.

When configuring resource monitoring or constraints, clones have different requirements than simple resources. For details, see *Pacemaker Explained*, available from https://www.clusterlab-s.org/pacemaker/doc/ .

You can create clone resources using either Hawk2 or crmsh.

6.6.1 Creating clone resources with Hawk2

🖊️ **Note: Child resources for clones**

Clones can either contain a primitive or a group as child resources. In Hawk2, child resources cannot be created or modified while creating a clone. Before adding a clone, create child resources and configure them as desired.

PROCEDURE 6.6: ADDING A CLONE RESOURCE WITH HAWK2

1. Log in to Hawk2:

   ```
   https://HAWKSERVER:7630/
   ```

2. From the left navigation bar, select *Configuration* › *Add Resource* › *Clone*.

3. Enter a unique *Clone ID*.

4. From the *Child Resource* list, select the primitive or group to use as a subresource for the clone.

5. If needed, modify or add *Meta Attributes*.

6. Click *Create* to finish the configuration. A message at the top of the screen shows if the action has been successful.
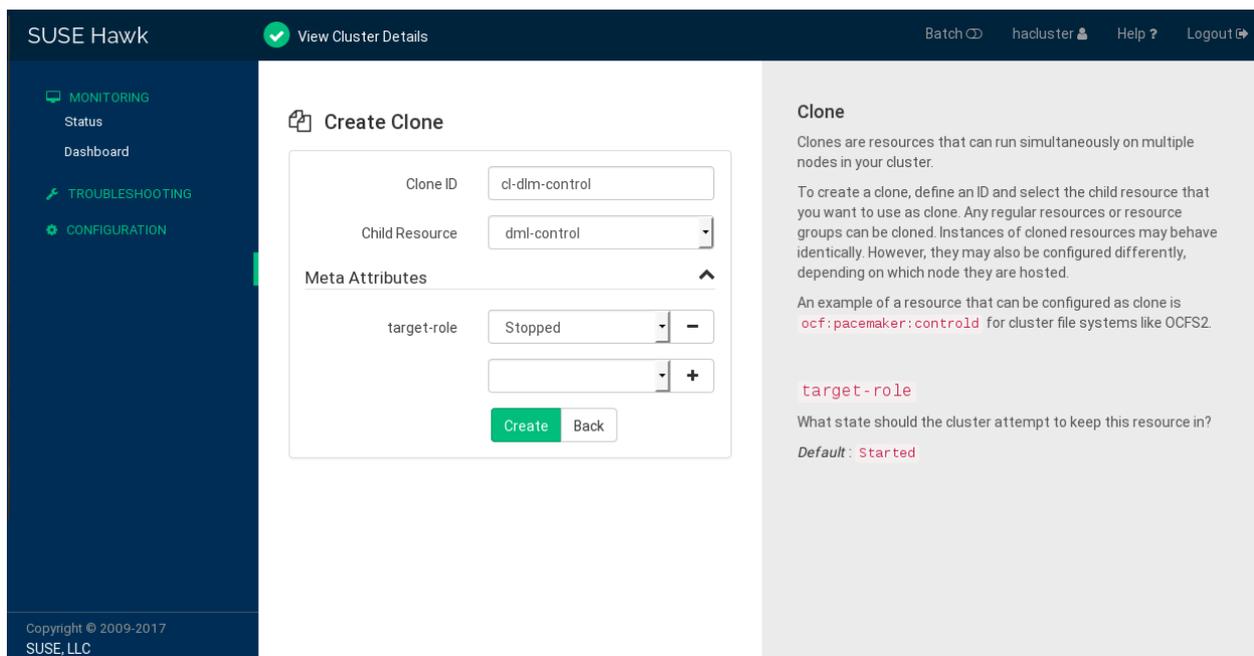
FIGURE 6.4: HAWK2—CLONE RESOURCE

6.6.2   Creating clone resources with crmsh

To create an anonymous clone resource, first create a primitive resource and then refer to it with the `clone` command.

PROCEDURE 6.7: ADDING A CLONE RESOURCE WITH CRMSH

1. Log in as `root` and start the `crm` interactive shell:

   ```
   # crm configure
   ```

2. Configure the primitive, for example:

   ```
   crm(live)configure# primitive Apache apache
   ```

3. Clone the primitive:

   ```
   crm(live)configure# clone cl-apache Apache
   ```

6.7 Creating promotable clones (multi-state resources)

Promotable clones (formerly known as multi-state resources) are a specialization of clones. They allow the instances to be in one of two operating modes (primary or secondary). Promotable clones must contain exactly one group or one regular resource.

When configuring resource monitoring or constraints, promotable clones have different requirements than simple resources. For details, see *Pacemaker Explained,* available from https://www.clusterlabs.org/pacemaker/doc/ ⬈.

You can create promotable clones using either Hawk2 or crmsh.

6.7.1 Creating promotable clones with Hawk2

📝 Note: Child resources for promotable clones

Promotable clones can either contain a primitive or a group as child resources. In Hawk2, child resources cannot be created or modified while creating a promotable clone. Before adding a promotable clone, create child resources and configure them as desired. See *Section 6.4.1, "Creating primitive resources with Hawk2"* or *Section 6.5.1, "Creating resource groups with Hawk2".*

PROCEDURE 6.8: ADDING A PROMOTABLE CLONE WITH HAWK2

1. Log in to Hawk2:

   ```
   https://HAWKSERVER:7630/
   ```

2. From the left navigation bar, select *Configuration* › *Add Resource* › *Multi-state*.

3. Enter a unique *Multi-state ID*.

4. From the *Child Resource* list, select the primitive or group to use as a subresource for the multi-state resource.

5. If needed, modify or add *Meta Attributes*.

6. Click *Create* to finish the configuration. A message at the top of the screen shows if the action has been successful.

6.7.2   Creating promotable clones with crmsh

To create a promotable clone resource, first create a primitive resource and then the promotable clone resource. The promotable clone resource must support at least promote and demote operations.

PROCEDURE 6.9: ADDING A PROMOTABLE CLONE WITH CRMSH

1. Log in as `root` and start the `crm` interactive shell:

   ```
   # crm configure
   ```

2. Configure the primitive. Change the intervals if needed:

   ```
   crm(live)configure# primitive my-rsc ocf:myCorp:myAppl \
       op monitor interval=60 \
       op monitor interval=61 role=Promoted
   ```

3. Create the promotable clone resource:

   ```
   crm(live)configure# clone clone-rsc my-rsc meta promotable=true
   ```

6.8   Creating resource templates

If you want to create lots of resources with similar configurations, defining a resource template is the easiest way. After being defined, it can be referenced in primitives, or in certain types of constraints as described in *Section 7.3, "Resource templates and constraints"*.

If a template is referenced in a primitive, the primitive inherits all operations, instance attributes (parameters), meta attributes and utilization attributes defined in the template. Additionally, you can define specific operations or attributes for your primitive. If any of these are defined in both the template and the primitive, the values defined in the primitive take precedence over the ones defined in the template.

You can create resource templates using either Hawk2 or crmsh.

6.8.1   Creating resource templates with Hawk2

Resource templates are configured like primitive resources.

1. Log in to Hawk2:

   ```
   https://HAWKSERVER:7630/
   ```

2. From the left navigation bar, select *Configuration* › *Add Resource* › *Template*.

3. Enter a unique *Resource ID*.

4. Follow the instructions in *Procedure 6.2, "Adding a primitive resource with Hawk2"*, starting from *Step 5*.

6.8.2 Creating resource templates with crmsh

Use the `rsc_template` command to get familiar with the syntax:

```
# crm configure rsc_template
 usage: rsc_template <name> [<class>:[<provider>:]]<type>
        [params <param>=<value> [<param>=<value>...]]
        [meta <attribute>=<value> [<attribute>=<value>...]]
        [utilization <attribute>=<value> [<attribute>=<value>...]]
        [operations id_spec
            [op op_type [<attribute>=<value>...] ...]]
```

For example, the following command creates a new resource template with the name `BigVM` derived from the `ocf:heartbeat:Xen` resource and some default values and operations:

```
crm(live)configure# rsc_template BigVM ocf:heartbeat:Xen \
   params allow_mem_management="true" \
   op monitor timeout=60s interval=15s \
   op stop timeout=10m \
   op start timeout=10m
```

Once you define the new resource template, you can use it in primitives or reference it in order, colocation or rsc_ticket constraints. To reference the resource template, use the `@` sign:

```
crm(live)configure# primitive MyVM1 @BigVM \
   params xmfile="/etc/xen/shared-vm/MyVM1" name="MyVM1"
```

The new primitive MyVM1 is going to inherit everything from the BigVM resource templates. For example, the equivalent of the above two would be:

```
crm(live)configure# primitive MyVM1 Xen \
   params xmfile="/etc/xen/shared-vm/MyVM1" name="MyVM1" \
   params allow_mem_management="true" \
```

```
   op monitor timeout=60s interval=15s \
   op stop timeout=10m \
   op start timeout=10m
```

If you want to overwrite some options or operations, add them to your (primitive) definition. For example, the following new primitive MyVM2 doubles the timeout for monitor operations but leaves others untouched:

```
crm(live)configure# primitive MyVM2 @BigVM \
   params xmfile="/etc/xen/shared-vm/MyVM2" name="MyVM2" \
   op monitor timeout=120s interval=30s
```

A resource template may be referenced in constraints to stand for all primitives which are derived from that template. This helps to produce a more concise and clear cluster configuration. Resource template references are allowed in all constraints except location constraints. Colocation constraints may not contain more than one template reference.

6.9 Creating STONITH resources

Important: No Support Without STONITH

- You must have a node fencing mechanism for your cluster.

- The global cluster options `stonith-enabled` and `startup-fencing` must be set to `true`. When you change them, you lose support.

By default, the global cluster option `stonith-enabled` is set to `true`. If no STONITH resources have been defined, the cluster will refuse to start any resources. Configure one or more STONITH resources to complete the STONITH setup. While STONITH resources are configured similarly to other resources, their behavior is different in some respects. For details refer to *Section 12.3, "STONITH resources and configuration"*.

You can create STONITH resources using either Hawk2 or crmsh.

6.9.1 Creating STONITH resources with Hawk2

To add a STONITH resource for SBD, for libvirt (KVM/Xen), or for vCenter/ESX Server, the easiest way is to use the Hawk2 wizard.

PROCEDURE 6.11: ADDING A STONITH RESOURCE WITH HAWK2

1. Log in to Hawk2:

   ```
   https://HAWKSERVER:7630/
   ```

2. From the left navigation bar, select *Configuration* › *Add Resource* › *Primitive*.

3. Enter a unique *Resource ID*.

4. From the *Class* list, select the resource agent class *stonith*.

5. From the *Type* list, select the STONITH plug-in to control your STONITH device. A short description for this plug-in is displayed.

6. Hawk2 automatically shows the required *Parameters* for the resource. Enter values for each parameter.

7. Hawk2 displays the most important resource *Operations* and proposes default values. If you do not modify any settings here, Hawk2 adds the proposed operations and their default values when you confirm.

8. If there is no reason to change them, keep the default *Meta Attributes* settings.
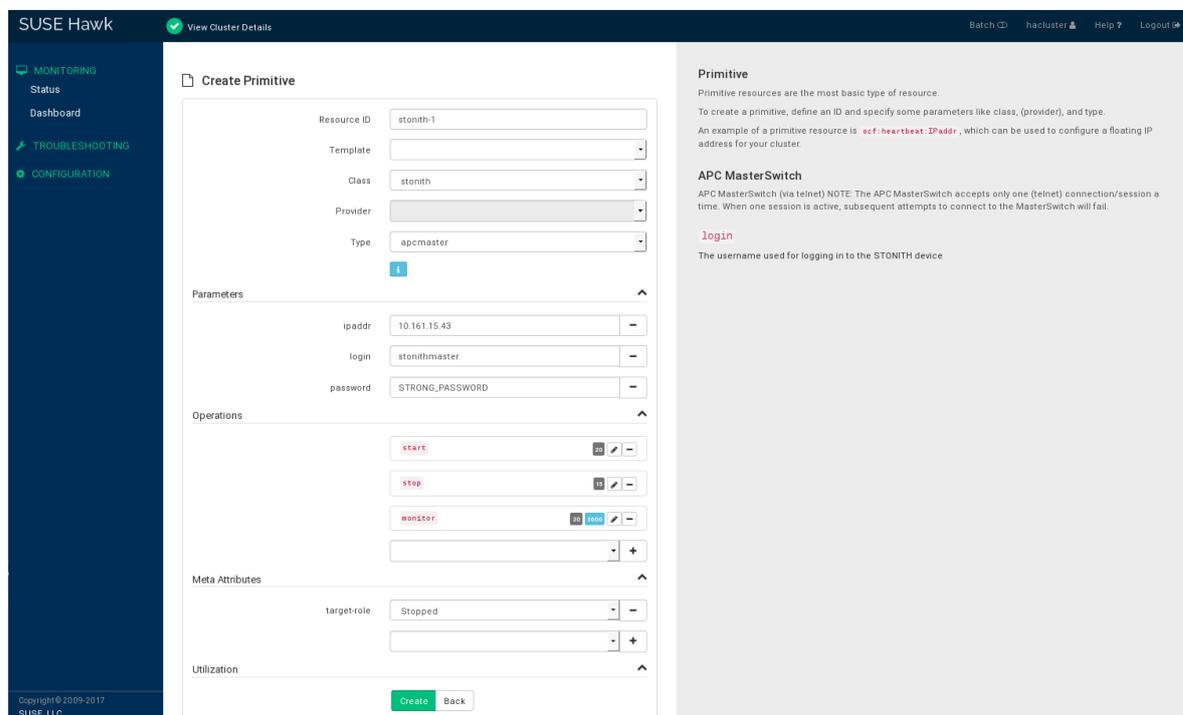
FIGURE 6.5: HAWK2—STONITH RESOURCE

Creating STONITH resources with Hawk2   SLE HA 15 SP5

9. Confirm your changes to create the STONITH resource.

   A message at the top of the screen shows if the action has been successful.

To complete your fencing configuration, add constraints. For more details, refer to *Chapter 12, Fencing and STONITH*.

6.9.2 Creating STONITH resources with crmsh

PROCEDURE 6.12: ADDING A STONITH RESOURCE WITH CRMSH

1. Log in as `root` and start the `crm` interactive shell:

   ```
   # crm
   ```

2. Get a list of all STONITH types with the following command:

   ```
   crm(live)# ra list stonith
   apcmaster               apcmastersnmp           apcsmart
   baytech                 bladehpi                cyclades
   drac3                   external/drac5          external/dracmc-telnet
   external/hetzner        external/hmchttp        external/ibmrsa
   external/ibmrsa-telnet  external/ipmi           external/ippower9258
   external/kdumpcheck     external/libvirt        external/nut
   external/rackpdu        external/riloe          external/sbd
   external/vcenter        external/vmware         external/xen0
   external/xen0-ha        fence_legacy            ibmhmc
   ipmilan                 meatware                nw_rpc100s
   rcd_serial              rps10                   suicide
   wti_mpc                 wti_nps
   ```

3. Choose a STONITH type from the above list and view the list of possible options. Use the following command:

   ```
   crm(live)# ra info stonith:external/ipmi
   IPMI STONITH external device (stonith:external/ipmi)

   ipmitool based power management. Apparently, the power off
   method of ipmitool is intercepted by ACPI which then makes
   a regular shutdown. In case of a split brain on a two-node,
   it may happen that no node survives. For two-node clusters,
   use only the reset method.

   Parameters (* denotes required, [] the default):
   ```

```
hostname (string): Hostname
    The name of the host to be managed by this STONITH device.
...
```

4. Create the STONITH resource with the `stonith` class, the type you have chosen in *Step 3*, and the respective parameters if needed, for example:

```
crm(live)# configure
crm(live)configure# primitive my-stonith stonith:external/ipmi \
    params hostname="alice" \
    ipaddr="192.168.1.221" \
    userid="admin" passwd="secret" \
    op monitor interval=60m timeout=120s
```

6.10 Configuring resource monitoring

If you want to ensure that a resource is running, you must configure resource monitoring for it. You can configure resource monitoring using either Hawk2 or crmsh.

If the resource monitor detects a failure, the following takes place:

- Log file messages are generated, according to the configuration specified in the `logging` section of `/etc/corosync/corosync.conf`.

- The failure is reflected in the cluster management tools (Hawk2, `crm status`), and in the CIB status section.

- The cluster initiates noticeable recovery actions, which may include stopping the resource to repair the failed state and restarting the resource locally or on another node. The resource also may not be restarted, depending on the configuration and state of the cluster.

If you do not configure resource monitoring, resource failures after a successful start will not be communicated, and the cluster will always show the resource as healthy.

Usually, resources are only monitored by the cluster while they are running. However, to detect concurrency violations, also configure monitoring for resources which are stopped. For resource monitoring, specify a timeout and/or start delay value, and an interval. The interval tells the CRM how often it should check the resource status. You can also set particular parameters such as `timeout` for `start` or `stop` operations.

For more information about monitor operation parameters, see *Section 6.14, "Resource operations"*.

6.10.1 Configuring resource monitoring with Hawk2

PROCEDURE 6.13: ADDING AND MODIFYING AN OPERATION

1. Log in to Hawk2:

   ```
   https://HAWKSERVER:7630/
   ```

2. Add a resource as described in *Procedure 6.2, "Adding a primitive resource with Hawk2"* or select an existing primitive to edit.
   Hawk2 automatically shows the most important *Operations* (`start`, `stop`, `monitor`) and proposes default values.
   To see the attributes belonging to each proposed value, hover the mouse pointer over the respective value.
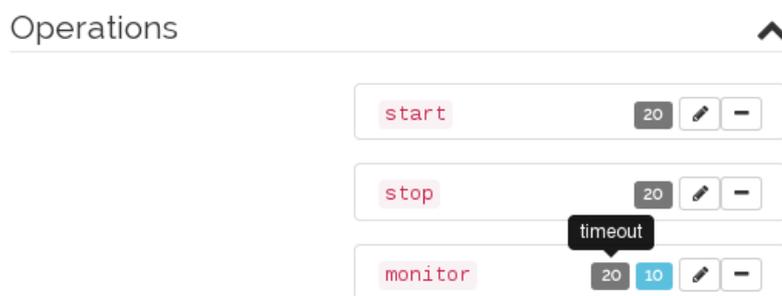
   FIGURE 6.6: OPERATION VALUES

3. To change the suggested `timeout` values for the `start` or `stop` operation:

   a. Click the pen icon next to the operation.

   b. In the dialog that opens, enter a different value for the `timeout` parameter, for example `10`, and confirm your change.

4. To change the suggested *interval* value for the `monitor` operation:

   a. Click the pen icon next to the operation.

   b. In the dialog that opens, enter a different value for the monitoring `interval`.

   c. To configure resource monitoring in the case that the resource is stopped:

      i. Select the `role` entry from the empty drop-down box below.

      ii. From the `role` drop-down box, select `Stopped`.

        iii. Click *Apply* to confirm your changes and to close the dialog for the operation.

5. Confirm your changes in the resource configuration screen. A message at the top of the screen shows if the action has been successful.

To view resource failures, switch to the *Status* screen in Hawk2 and select the resource you are interested in. In the *Operations* column click the arrow down icon and select *Recent Events*. The dialog that opens lists recent actions performed for the resource. Failures are displayed in red. To view the resource details, click the magnifier icon in the *Operations* column.

FIGURE 6.7: HAWK2—RESOURCE DETAILS

6.10.2   Configuring resource monitoring with crmsh

To monitor a resource, there are two possibilities: either define a monitor operation with the `op` keyword or use the `monitor` command. The following example configures an Apache resource and monitors it every 60 seconds with the `op` keyword:

```
crm(live)configure# primitive apache apache \
```

```
    params ... \
  op monitor interval=60s timeout=30s
```

The same can be done with the following commands:

```
crm(live)configure# primitive apache apache \
    params ...
 crm(live)configure# monitor apache 60s:30s
```

Monitoring stopped resources

Usually, resources are only monitored by the cluster while they are running. However, to detect concurrency violations, also configure monitoring for resources which are stopped. For example:

```
crm(live)configure# primitive dummy1 Dummy \
      op monitor interval="300s" role="Stopped" timeout="10s" \
      op monitor interval="30s" timeout="10s"
```

This configuration triggers a monitoring operation every `300` seconds for the resource `dummy1` when it is in `role="Stopped"`. When running, it will be monitored every `30` seconds.

Probing

The CRM executes an initial monitoring for each resource on every node, the so-called `probe`. A probe is also executed after the cleanup of a resource. If multiple monitoring operations are defined for a resource, the CRM selects the one with the smallest interval and uses its timeout value as default timeout for probing. If no monitor operation is configured, the cluster-wide default applies. The default is `20` seconds (if not specified otherwise by configuring the `op_defaults` parameter). If you do not want to rely on the automatic calculation or the `op_defaults` value, define a specific monitoring operation for the *probing* of this resource. Do so by adding a monitoring operation with the `interval` set to `0`, for example:

```
crm(live)configure# primitive rsc1 ocf:pacemaker:Dummy \
      op monitor interval="0" timeout="60"
```

The probe of `rsc1` times out in `60s`, independent of the global timeout defined in `op_de-faults`, or any other operation timeouts configured. If you did not set `interval="0"` for specifying the probing of the respective resource, the CRM automatically checks for any other monitoring operations defined for that resource and calculates the timeout value for probing as described above.

6.11 Loading resources from a file

Parts or all of the configuration can be loaded from a local file or a network URL. Three different methods can be defined:

`replace`

> This option replaces the current configuration with the new source configuration.

`update`

> This option tries to import the source configuration. It adds new items or updates existing items to the current configuration.

`push`

> This option imports the content from the source into the current configuration (same as `update`). However, it removes objects that are not available in the new configuration.

To load the new configuration from the file `mycluster-config.txt` use the following syntax:

```
# crm configure load push mycluster-config.txt
```

6.12 Resource options (meta attributes)

For each resource you add, you can define options. Options are used by the cluster to decide how your resource should behave; they tell the CRM how to treat a specific resource. Resource options can be set with the **crm_resource --meta** command or with Hawk2.

The following list shows some common options:

`priority`

> If not all resources can be active, the cluster stops lower-priority resources to keep higher-priority resources active.
> The default value is `0`.

`target-role`

> In what state should the cluster attempt to keep this resource? Allowed values: `Stopped`, `Started`, `Unpromoted`, `Promoted`.
> The default value is `Started`.

`is-managed`

Is the cluster allowed to start and stop the resource? Allowed values: `true`, `false`. If the value is set to `false`, the status of the resource is still monitored and any failures are reported. This is different from setting a resource to `maintenance="true"`.

The default value is `true`.

`maintenance`

Can the resources be touched manually? Allowed values: `true`, `false`. If set to `true`, all resources become unmanaged: the cluster stops monitoring them and does not know their status. You can stop or restart cluster resources without the cluster attempting to restart them.

The default value is `false`.

`resource-stickiness`

How much does the resource prefer to stay where it is?

The default value is `1` for individual clone instances, and `0` for all other resources.

`migration-threshold`

How many failures should occur for this resource on a node before making the node ineligible to host this resource?

The default value is `INFINITY`.

`multiple-active`

What should the cluster do if it ever finds the resource active on more than one node? Allowed values: `block` (mark the resource as unmanaged), `stop_only`, `stop_start`.

The default value is `stop_start`.

`failure-timeout`

How many seconds to wait before acting as if the failure did not occur (and potentially allowing the resource back to the node on which it failed)?

The default value is `0` (disabled).

`allow-migrate`

Whether to allow live migration for resources that support `migrate_to` and `migrate_from` actions. If the value is set to `true`, the resource can be migrated without loss of state. If the value is set to `false`, the resource will be shut down on the first node and restarted on the second node.

The default value is `true` for `ocf:pacemaker:remote` resources, and `false` for all other resources.

Resource options (meta attributes)   SLE HA 15 SP5

`allow-unhealthy-nodes`

> Allows the resource to run on a node even if the node's health score would otherwise prevent it.
>
> The default value is `false`.

`remote-node`

> The name of the remote node this resource defines. This both enables the resource as a remote node and defines the unique name used to identify the remote node. If no other parameters are set, this value is also assumed as the host name to connect to at the `remote-port` port.
>
> This option is disabled by default.
>
> ⚠️ Warning: Use unique IDs
>
> > This value must not overlap with any existing resource or node IDs.

`remote-port`

> Custom port for the guest connection to pacemaker_remote.
>
> The default value is `3121`.

`remote-addr`

> The IP address or host name to connect to if the remote node's name is not the host name of the guest.
>
> The default value is the value set by `remote-node`.

`remote-connect-timeout`

> How long before a pending guest connection times out?
>
> The default value is `60s`.

6.13 Instance attributes (parameters)

The scripts of all resource classes can be given parameters which determine how they behave and which instance of a service they control. If your resource agent supports parameters, you can add them with the **crm_resource** command or with Hawk2. In the **crm** command line

utility and in Hawk2, instance attributes are called `params` or `Parameter`, respectively. The list of instance attributes supported by an OCF script can be found by executing the following command as `root`:

```
# crm ra info [class:[provider:]]resource_agent
```

or (without the optional parts):

```
# crm ra info resource_agent
```

The output lists all the supported attributes, their purpose and default values.

Note: Instance attributes for groups, clones or promotable clones

Note that groups, clones and promotable clone resources do not have instance attributes. However, any instance attributes set will be inherited by the group's, clone's or promotable clone's children.

6.14 Resource operations

By default, the cluster will not ensure that your resources are still healthy. To instruct the cluster to do this, you need to add a monitor operation to the resource's definition. Monitor operations can be added for all classes or resource agents.

Monitor operations can have the following properties:

`id`

Your name for the action. Must be unique. (The ID is not shown.)

`name`

The action to perform. Common values: `monitor`, `start`, `stop`.

`interval`

How frequently to perform the operation, in seconds.

`timeout`

How long to wait before declaring the action has failed.

requires

What conditions need to be satisfied before this action occurs. Allowed values: `nothing`, `quorum`, `fencing`. The default depends on whether fencing is enabled and if the resource's class is `stonith`. For STONITH resources, the default is `nothing`.

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`: Bring down the node on which the resource failed (STONITH).

- `standby`: Move *all* resources away from the node on which the resource failed.

enabled

If `false`, the operation is treated as if it does not exist. Allowed values: `true`, `false`.

role

Run the operation only if the resource has this role.

record-pending

Can be set either globally or for individual resources. Makes the CIB reflect the state of "in-flight" operations on resources.

description

Description of the operation.

7 Configuring resource constraints

Having all the resources configured is only part of the job. Even if the cluster knows all needed resources, it still might not be able to handle them correctly. Resource constraints let you specify which cluster nodes resources can run on, what order resources will load in, and what other resources a specific resource is dependent on.

7.1 Types of constraints

There are three different kinds of constraints available:

Resource location

Location constraints define on which nodes a resource may be run, may not be run or is preferred to be run.

Resource colocation

Colocation constraints tell the cluster which resources may or may not run together on a node.

Resource order

Order constraints define the sequence of actions.

> **!** Important: Restrictions for constraints and certain types of resources
>
> - Do not create colocation constraints for *members* of a resource group. Create a colocation constraint pointing to the resource group as a whole instead. All other types of constraints are safe to use for members of a resource group.
>
> - Do not use any constraints on a resource that has a clone resource or a promotable clone resource applied to it. The constraints must apply to the clone or promotable clone resource, not to the child resource.

7.2 Scores and infinity

When defining constraints, you also need to deal with scores. Scores of all kinds are integral to how the cluster works. Practically everything from migrating a resource to deciding which resource to stop in a degraded cluster is achieved by manipulating scores. Scores are calculated on a per-resource basis and any node with a negative score for a resource cannot run that resource. After calculating the scores for a resource, the cluster then chooses the node with the highest score.

`INFINITY` is currently defined as `1,000,000`. Additions or subtractions with it stick to the following three basic rules:

- Any value + INFINITY = INFINITY

- Any value - INFINITY = -INFINITY

- INFINITY - INFINITY = -INFINITY

When defining resource constraints, you specify a score for each constraint. The score indicates the value you are assigning to this resource constraint. Constraints with higher scores are applied before those with lower scores. By creating additional location constraints with different scores for a given resource, you can specify an order for the nodes that a resource will fail over to.

7.3 Resource templates and constraints

If you have defined a resource template (see *Section 6.8, "Creating resource templates"*), it can be referenced in the following types of constraints:

- order constraints

- colocation constraints

- rsc_ticket constraints (for Geo clusters)

However, colocation constraints must not contain more than one reference to a template. Resource sets must not contain a reference to a template.

Resource templates referenced in constraints stand for all primitives which are derived from that template. This means, the constraint applies to all primitive resources referencing the resource template. Referencing resource templates in constraints is an alternative to resource sets and can simplify the cluster configuration considerably. For details about resource sets, refer to *Section 7.7, "Using resource sets to define constraints"*.

7.4 Adding location constraints

A location constraint determines on which node a resource may be run, is preferably run, or may not be run. An example of a location constraint is to place all resources related to a certain database on the same node. This type of constraint may be added multiple times for each resource. All location constraints are evaluated for a given resource.

You can add location constraints using either Hawk2 or crmsh.

7.4.1 Adding location constraints with Hawk2

PROCEDURE 7.1: ADDING A LOCATION CONSTRAINT

1. Log in to Hawk2:

   ```
   https://HAWKSERVER:7630/
   ```

2. From the left navigation bar, select *Configuration* › *Add Constraint* › *Location.*

3. Enter a unique *Constraint ID*.

4. From the list of *Resources* select the resource or resources for which to define the constraint.

5. Enter a *Score*. The score indicates the value you are assigning to this resource constraint. Positive values indicate the resource can run on the *Node* you specify in the next step. Negative values mean it should not run on that node. Constraints with higher scores are applied before those with lower scores.
   Some often-used values can also be set via the drop-down box:

   - To force the resources to run on the node, click the arrow icon and select `Always`. This sets the score to `INFINITY`.

   - If you never want the resources to run on the node, click the arrow icon and select `Never`. This sets the score to `-INFINITY`, meaning that the resources must not run on the node.

   - To set the score to `0`, click the arrow icon and select `Advisory`. This disables the constraint. This is useful when you want to set resource discovery but do not want to constrain the resources.

6. Select a *Node.*

7. Click *Create* to finish the configuration. A message at the top of the screen shows if the action has been successful.
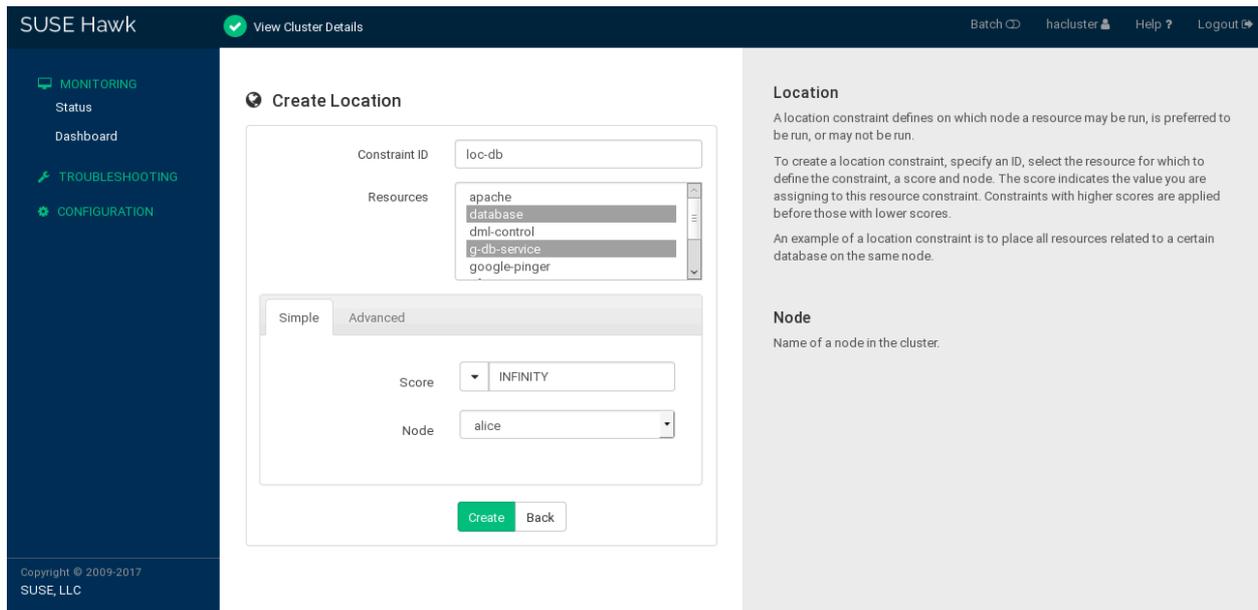
FIGURE 7.1: HAWK2—LOCATION CONSTRAINT

7.4.2   Adding location constraints with crmsh

The `location` command defines on which nodes a resource may be run, may not be run or is preferred to be run.

A simple example that expresses a preference to run the resource `fs1` on the node with the name `alice` to 100 would be the following:

```
crm(live)configure# location loc-fs1 fs1 100: alice
```

Another example is a location with ping:

```
crm(live)configure# primitive ping ping \
    params name=ping dampen=5s multiplier=100 host_list="r1 r2"
crm(live)configure# clone cl-ping ping meta interleave=true
crm(live)configure# location loc-node_pref internal_www \
    rule 50: #uname eq alice \
    rule ping: defined ping
```

The parameter `host_list` is a space-separated list of hosts to ping and count. Another use case for location constraints are grouping primitives as a *resource set*. This can be useful if several resources depend on, for example, a ping attribute for network connectivity. In former times, the `-inf/ping` rules needed to be duplicated several times in the configuration, making it unnecessarily complex.

The following example creates a resource set `loc-alice`, referencing the virtual IP addresses `vip1` and `vip2`:

```
crm(live)configure# primitive vip1 IPaddr2 params ip=192.168.1.5
crm(live)configure# primitive vip2 IPaddr2 params ip=192.168.1.6
crm(live)configure# location loc-alice { vip1 vip2 } inf: alice
```

In some cases it is much more efficient and convenient to use resource patterns for your `location` command. A resource pattern is a regular expression between two slashes. For example, the above virtual IP addresses can be all matched with the following:

```
crm(live)configure# location loc-alice /vip.*/ inf: alice
```

7.5 Adding colocation constraints

A colocation constraint tells the cluster which resources may or may not run together on a node. As a colocation constraint defines a dependency between resources, you need at least two resources to create a colocation constraint.

You can add colocation constraints using either Hawk2 or crmsh.

7.5.1 Adding colocation constraints with Hawk2

PROCEDURE 7.2: ADDING COLOCATION CONSTRAINTS

1. Log in to Hawk2:

   ```
   https://HAWKSERVER:7630/
   ```

2. From the left navigation bar, select *Configuration* › *Add Constraint* › *Colocation*.

3. Enter a unique *Constraint ID*.

4. Enter a *Score*. The score determines the location relationship between the resources. Positive values indicate that the resources should run on the same node. Negative values indicate that the resources should not run on the same node. The score will be combined with other factors to decide where to put the resource.

   Some often-used values can also be set via the drop-down box:

   - To force the resources to run on the same node, click the arrow icon and select `Always`. This sets the score to `INFINITY`.

   - If you never want the resources to run on the same node, click the arrow icon and select `Never`. This sets the score to `-INFINITY`, meaning that the resources must not run on the same node.

5. To define the resources for the constraint:

   a. From the drop-down box in the *Resources* category, select a resource (or a template). The resource is added and a new empty drop-down box appears beneath.

   b. Repeat this step to add more resources.
      As the topmost resource depends on the next resource and so on, the cluster first decides where to put the last resource, then place the depending ones based on that decision. If the constraint cannot be satisfied, the cluster may not allow the dependent resource to run.

   c. To swap the order of resources within the colocation constraint, click the arrow up icon next to a resource to swap it with the entry above.

6. If needed, specify further parameters for each resource (such as `Started`, `Stopped`, `Promote`, `Demote`). Click the empty drop-down box next to the resource and select the desired entry.

7. Click *Create* to finish the configuration. A message at the top of the screen shows if the action has been successful.
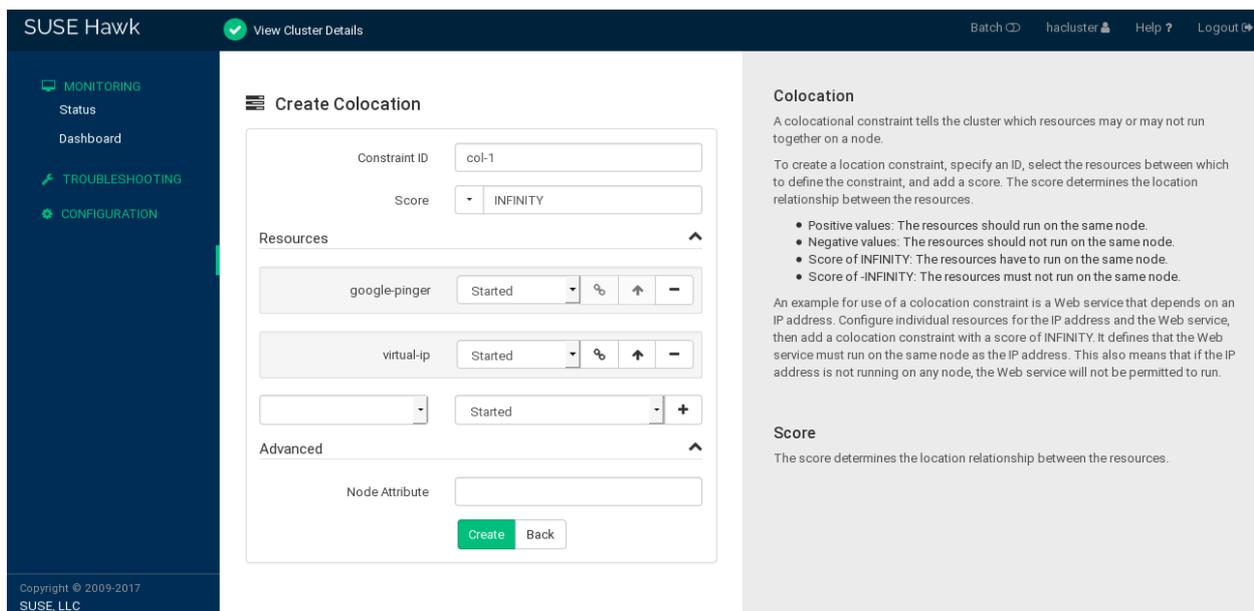
FIGURE 7.2: HAWK2—COLOCATION CONSTRAINT

7.5.2 Adding colocation constraints with crmsh

The `colocation` command is used to define what resources should run on the same or on different hosts.

You can set a score of either +inf or -inf, defining resources that must always or must never run on the same node. You can also use non-infinite scores. In that case the colocation is called *advisory* and the cluster may decide not to follow them in favor of not stopping other resources if there is a conflict.

For example, to always run the resources `resource1` and `resource2` on the same host, use the following constraint:

```
crm(live)configure# colocation coloc-2resource inf: resource1 resource2
```

For a primary/secondary configuration, it is necessary to know if the current node is a primary in addition to running the resource locally.

7.6 Adding order constraints

Order constraints can be used to start or stop a service right before or after a different resource meets a special condition, such as being started, stopped, or promoted to primary. For example, you cannot mount a file system before the device is available to a system. As an order constraint defines a dependency between resources, you need at least two resources to create an order constraint.

You can add order constraints using either Hawk2 or crmsh.

7.6.1 Adding order constraints with Hawk2

PROCEDURE 7.3: ADDING AN ORDER CONSTRAINT

1. Log in to Hawk2:

   ```
   https://HAWKSERVER:7630/
   ```

2. From the left navigation bar, select *Configuration* › *Add Constraint* › *Order*.

3. Enter a unique *Constraint ID*.

4. Enter a *Score*. If the score is greater than zero, the order constraint is mandatory, otherwise it is optional.
   Some often-used values can also be set via the drop-down box:

   - To make the order constraint mandatory, click the arrow icon and select `Mandatory`.

   - To make the order constraint a suggestion only, click the arrow icon and select `Optional`.

   - `Serialize`: To ensure that no two stop/start actions occur concurrently for the resources, click the arrow icon and select `Serialize`. This makes sure that one resource must complete starting before the other can be started. A typical use case are resources that put a high load on the host during start-up.

5. For order constraints, you can usually keep the option *Symmetrical* enabled. This specifies that resources are stopped in reverse order.

6. To define the resources for the constraint:

   a. From the drop-down box in the *Resources* category, select a resource (or a template).

The resource is added and a new empty drop-down box appears beneath.

b. Repeat this step to add more resources.

The topmost resource starts first, then the second, etc. Usually the resources are stopped in reverse order.

c. To swap the order of resources within the order constraint, click the arrow up icon next to a resource to swap it with the entry above.

7. If needed, specify further parameters for each resource (like `Started`, `Stopped`, `Promote`, `Demote`). Click the empty drop-down box next to the resource and select the desired entry.

8. Confirm your changes to finish the configuration. A message at the top of the screen shows if the action has been successful.
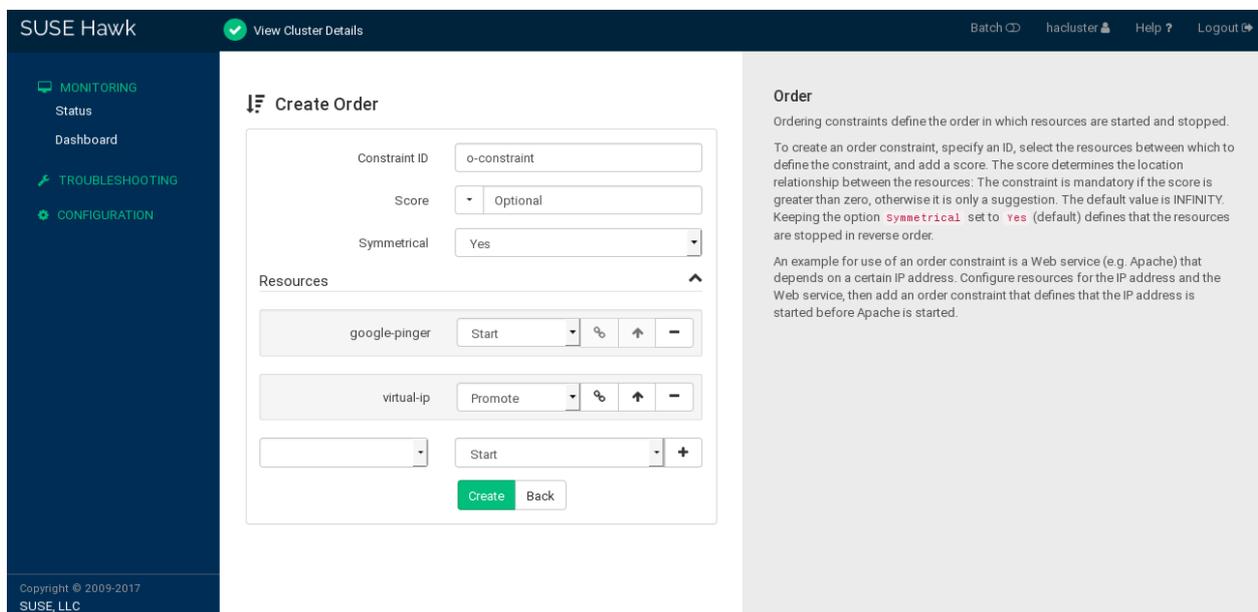
FIGURE 7.3: HAWK2—ORDER CONSTRAINT

7.6.2 Adding order constraints using crmsh

The `order` command defines a sequence of action.

For example, to always start `resource1` before `resource2`, use the following constraint:

```
crm(live)configure# order res1_before_res2 Mandatory: resource1 resource2
```

7.7 Using resource sets to define constraints

As an alternative format for defining location, colocation or order constraints, you can use *resource sets*, where primitives are grouped together in one set. Previously this was possible either by defining a resource group (which could not always accurately express the design), or by defining each relationship as an individual constraint. The latter caused a constraint explosion as the number of resources and combinations grew. The configuration via resource sets is not necessarily less verbose, but is easier to understand and maintain.

You can configure resource sets using either Hawk2 or crmsh.

7.7.1 Using resource sets to define constraints with Hawk2

PROCEDURE 7.4: USING A RESOURCE SET FOR CONSTRAINTS

1. To use a resource set within a location constraint:

   a. Proceed as outlined in *Procedure 7.1, "Adding a location constraint"*, apart from *Step 4*. Instead of selecting a single resource, select multiple resources by pressing `Ctrl` or `Shift` and mouse click. This creates a resource set within the location constraint.

   b. To remove a resource from the location constraint, press `Ctrl` and click the resource again to deselect it.

2. To use a resource set within a colocation or order constraint:

   a. Proceed as described in *Procedure 7.2, "Adding colocation constraints"* or *Procedure 7.3, "Adding an order constraint"*, apart from the step where you define the resources for the constraint (*Step 5.a* or *Step 6.a*):

   b. Add multiple resources.

   c. To create a resource set, click the chain icon next to a resource to link it to the resource above. A resource set is visualized by a frame around the resources belonging to a set.

   d. You can combine multiple resources in a resource set or create multiple resource sets.
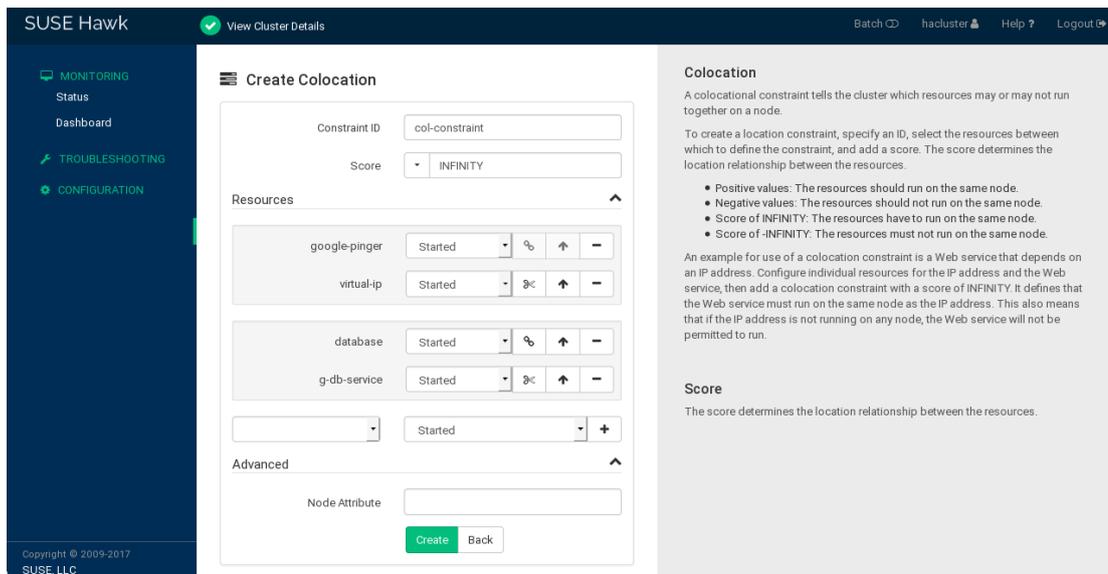
FIGURE 7.4: HAWK2—TWO RESOURCE SETS IN A COLOCATION CONSTRAINT

    e. To unlink a resource from the resource above, click the scissors icon next to the resource.

3. Confirm your changes to finish the constraint configuration.

7.7.2 Using resource sets to define constraints with crmsh

EXAMPLE 7.1: A RESOURCE SET FOR LOCATION CONSTRAINTS

For example, you can use the following configuration of a resource set (`loc-alice`) in the crmsh to place two virtual IPs (`vip1` and `vip2`) on the same node, `alice`:

```
crm(live)configure# primitive vip1 IPaddr2 params ip=192.168.1.5
crm(live)configure# primitive vip2 IPaddr2 params ip=192.168.1.6
crm(live)configure# location loc-alice { vip1 vip2 } inf: alice
```

If you want to use resource sets to replace a configuration of colocation constraints, consider the following two examples:

EXAMPLE 7.2: A CHAIN OF COLOCATED RESOURCES

```
<constraints>
    <rsc_colocation id="coloc-1" rsc="B" with-rsc="A" score="INFINITY"/>
    <rsc_colocation id="coloc-2" rsc="C" with-rsc="B" score="INFINITY"/>
```

```
    <rsc_colocation id="coloc-3" rsc="D" with-rsc="C" score="INFINITY"/>
</constraints>
```

The same configuration expressed by a resource set:

```
<constraints>
   <rsc_colocation id="coloc-1" score="INFINITY" >
    <resource_set id="colocated-set-example" sequential="true">
     <resource_ref id="A"/>
     <resource_ref id="B"/>
     <resource_ref id="C"/>
     <resource_ref id="D"/>
    </resource_set>
   </rsc_colocation>
</constraints>
```

If you want to use resource sets to replace a configuration of order constraints, consider the following two examples:

EXAMPLE 7.3: A CHAIN OF ORDERED RESOURCES

```
<constraints>
    <rsc_order id="order-1" first="A" then="B" />
    <rsc_order id="order-2" first="B" then="C" />
    <rsc_order id="order-3" first="C" then="D" />
</constraints>
```

The same purpose can be achieved by using a resource set with ordered resources:

EXAMPLE 7.4: A CHAIN OF ORDERED RESOURCES EXPRESSED AS RESOURCE SET

```
<constraints>
    <rsc_order id="order-1">
    <resource_set id="ordered-set-example" sequential="true">
    <resource_ref id="A"/>
    <resource_ref id="B"/>
    <resource_ref id="C"/>
    <resource_ref id="D"/>
    </resource_set>
    </rsc_order>
</constraints>
```

Sets can be either ordered (`sequential=true`) or unordered (`sequential=false`). Furthermore, the `require-all` attribute can be used to switch between `AND` and `OR` logic.

7.7.3 Collocating sets for resources without dependency

It is sometimes useful to place a group of resources on the same node (defining a colocation constraint), but without having hard dependencies between the resources. For example, you want two resources to be placed on the same node, but you do *not* want the cluster to restart the other one if one of them fails.

This can be achieved on the crm shell by using the `weak-bond` command:

```
# crm configure assist weak-bond resource1 resource2
```

The `weak-bond` command creates a dummy resource and a colocation constraint with the given resources automatically.

7.8 Specifying resource failover nodes

A resource is automatically restarted if it fails. If that cannot be achieved on the current node, or it fails `N` times on the current node, it tries to fail over to another node. Each time the resource fails, its fail count is raised. You can define several failures for resources (a `migration-threshold`), after which they will migrate to a new node. If you have more than two nodes in your cluster, the node a particular resource fails over to is chosen by the High Availability software.

However, you can specify the node a resource will fail over to by configuring one or several location constraints and a `migration-threshold` for that resource.

You can specify resource failover nodes using either Hawk2 or crmsh.

EXAMPLE 7.5: MIGRATION THRESHOLD—PROCESS FLOW

> For example, let us assume you have configured a location constraint for resource `rsc1` to preferably run on `alice`. If it fails there, `migration-threshold` is checked and compared to the fail count. If `failcount` $>=$ migration-threshold, then the resource is migrated to the node with the next best preference.
>
> After the threshold has been reached, the node will no longer be allowed to run the failed resource until the resource's fail count is reset. This can be done manually by the cluster administrator or by setting a `failure-timeout` option for the resource.
>
> For example, a setting of `migration-threshold=2` and `failure-timeout=60s` would cause the resource to migrate to a new node after two failures. It would be allowed to move back (depending on the stickiness and constraint scores) after one minute.

There are two exceptions to the migration threshold concept, occurring when a resource either fails to start or fails to stop:

- Start failures set the fail count to `INFINITY` and thus always cause an immediate migration.

- Stop failures cause fencing (when `stonith-enabled` is set to `true`, which is the default). If there is no STONITH resource defined (or `stonith-enabled` is set to `false`), the resource will not migrate.

7.8.1 Specifying resource failover nodes with Hawk2

PROCEDURE 7.5: SPECIFYING A FAILOVER NODE

1. Log in to Hawk2:

   ```
   https://HAWKSERVER:7630/
   ```

2. Configure a location constraint for the resource as described in *Procedure 7.1, "Adding a location constraint"*.

3. Add the `migration-threshold` meta attribute to the resource as described in *Procedure 8.1: Modifying a resource or group*, *Step 5* and enter a value for the migration-threshold. The value should be positive and less than INFINITY.

4. If you want to automatically expire the fail count for a resource, add the `failure-timeout` meta attribute to the resource as described in *Procedure 6.2: Adding a primitive resource with Hawk2*, *Step 5* and enter a *Value* for the `failure-timeout`.
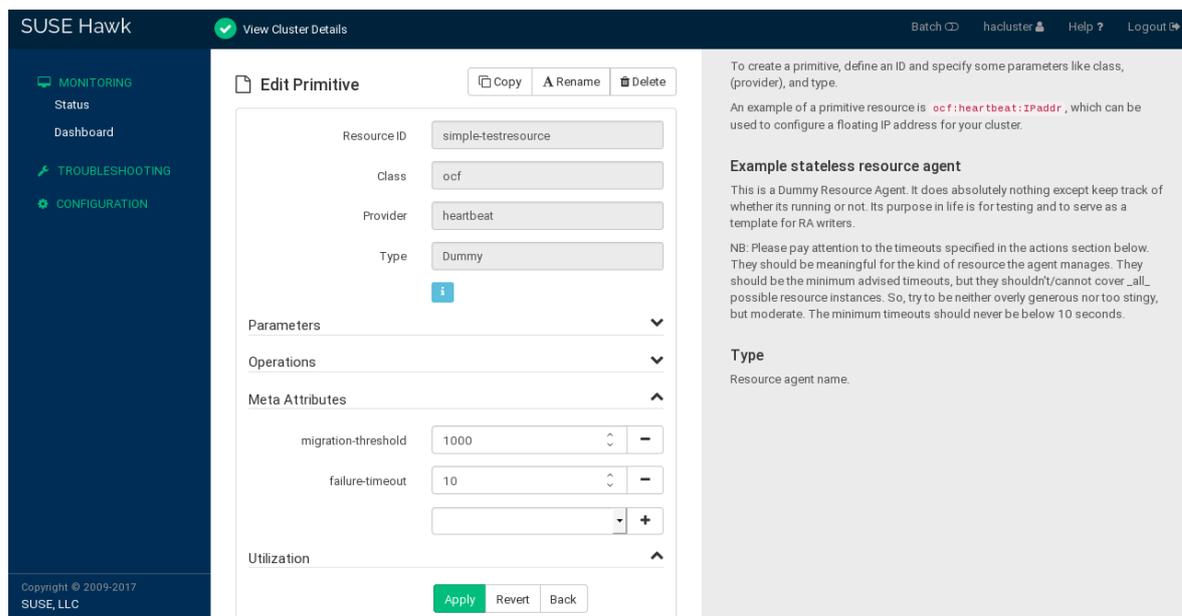
FIGURE 7.5: HAWK2: META ATTRIBUTES FOR RESOURCE FAILOVER

5. To specify additional failover nodes with preferences for a resource, create additional location constraints.

Instead of letting the fail count for a resource expire automatically, you can also clean up fail counts for a resource manually at any time. Refer to *Section 8.5.1, "Cleaning up cluster resources with Hawk2"* for details.

7.8.2 Specifying resource failover nodes with crmsh

To determine a resource failover, use the meta attribute `migration-threshold`. If the fail count exceeds `migration-threshold` on all nodes, the resource remains stopped. For example:

```
crm(live)configure# location rsc1-alice rsc1 100: alice
```

Normally, `rsc1` prefers to run on `alice`. If it fails there, `migration-threshold` is checked and compared to the fail count. If `failcount` `>=` `migration-threshold`, then the resource is migrated to the node with the next best preference.

Start failures set the fail count to inf depend on the `start-failure-is-fatal` option. Stop failures cause fencing. If there is no STONITH defined, the resource will not migrate.

7.9 Specifying resource failback nodes (resource stickiness)

A resource might fail back to its original node when that node is back online and in the cluster. To prevent a resource from failing back to the node that it was running on, or to specify a different node for the resource to fail back to, change its resource stickiness value. You can either specify resource stickiness when you are creating a resource or afterward.

Consider the following implications when specifying resource stickiness values:

Value is `0`:

> The resource is placed optimally in the system. This may mean that it is moved when a "better" or less loaded node becomes available. The option is almost equivalent to automatic failback, except that the resource may be moved to a node that is not the one it was previously active on. This is the Pacemaker default.

Value is greater than `0`:

> The resource prefers to remain in its current location, but may be moved if a more suitable node is available. Higher values indicate a stronger preference for a resource to stay where it is.

Value is less than `0`:

> The resource prefers to move away from its current location. Higher absolute values indicate a stronger preference for a resource to be moved.

Value is `INFINITY`:

> The resource always remains in its current location unless forced off because the node is no longer eligible to run the resource (node shutdown, node standby, reaching the `migration-threshold`, or configuration change). This option is almost equivalent to disabling automatic failback.

Value is `-INFINITY`:

> The resource always moves away from its current location.

7.9.1 Specifying resource failback nodes with Hawk2

PROCEDURE 7.6: SPECIFYING RESOURCE STICKINESS

1. Log in to Hawk2:

   ```
   https://HAWKSERVER:7630/
   ```

2. Add the `resource-stickiness` meta attribute to the resource as described in *Procedure 8.1: Modifying a resource or group*, *Step 5*.

3. Specify a value between `-INFINITY` and `INFINITY` for `resource-stickiness`.
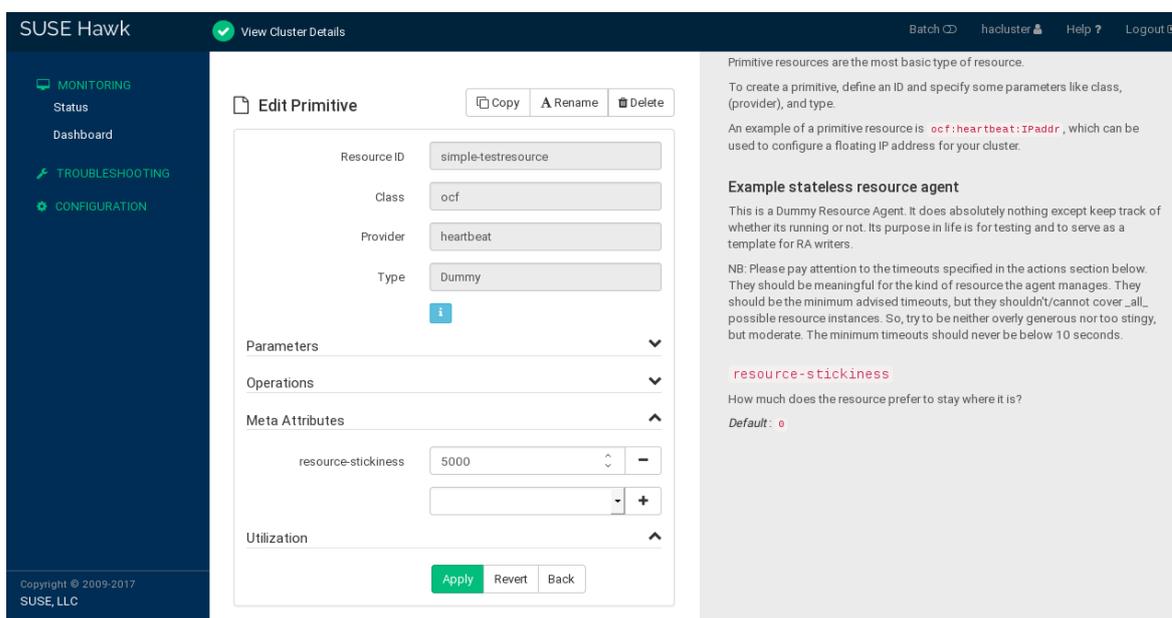
FIGURE 7.6: HAWK2: META ATTRIBUTES FOR RESOURCE FAILBACK

7.10 Placing resources based on their load impact

Not all resources are equal. Some, such as Xen guests, require that the node hosting them meets their capacity requirements. If resources are placed such that their combined need exceed the provided capacity, the resources diminish in performance (or even fail).

To take this into account, SUSE Linux Enterprise High Availability allows you to specify the following parameters:

1. The capacity a certain node *provides*.

2. The capacity a certain resource *requires*.

3. An overall strategy for placement of resources.

You can configure these setting using either Hawk2 or crmsh:

- Hawk2: *Section 7.10.1, "Placing resources based on their load impact with Hawk2"*

- crmsh: *Section 7.10.2, "Placing resources based on their load impact with crmsh"*

A node is considered eligible for a resource if it has sufficient free capacity to satisfy the resource's requirements. To manually configure the resource's requirements and the capacity a node provides, use utilization attributes. You can name the utilization attributes according to your preferences and define as many name/value pairs as your configuration needs. However, the attribute's values must be integers.

If multiple resources with utilization attributes are grouped or have colocation constraints, SUSE Linux Enterprise High Availability takes that into account. If possible, the resources is placed on a node that can fulfill *all* capacity requirements.

Note: Utilization attributes for groups

It is impossible to set utilization attributes directly for a resource group. However, to simplify the configuration for a group, you can add a utilization attribute with the total capacity needed to any of the resources in the group.

SUSE Linux Enterprise High Availability also provides the means to detect and configure both node capacity and resource requirements automatically:

The `NodeUtilization` resource agent checks the capacity of a node (regarding CPU and RAM). To configure automatic detection, create a clone resource of the following class, provider and type: `ocf:pacemaker:NodeUtilization`. One instance of the clone should be running on each node. After the instance starts, a utilization section is added to the node's configuration in CIB. For more information, see **`crm ra info NodeUtilization`**.

For automatic detection of a resource's minimal requirements (regarding RAM and CPU) the `Xen` resource agent has been improved. Upon start of a `Xen` resource, it reflects the consumption of RAM and CPU. Utilization attributes are automatically added to the resource configuration.

> ### Note: Different resource agents for Xen and libvirt
>
> The `ocf:heartbeat:Xen` resource agent should not be used with `libvirt`, as `libvirt` expects to be able to modify the machine description file.
>
> For `libvirt`, use the `ocf:heartbeat:VirtualDomain` resource agent.

Apart from detecting the minimal requirements, you can also monitor the current utilization via the `VirtualDomain` resource agent. It detects CPU and RAM use of the virtual machine. To use this feature, configure a resource of the following class, provider and type: `ocf:heartbeat:VirtualDomain`. The following instance attributes are available:

- `autoset_utilization_cpu`

- `autoset_utilization_hv_memory` (for Xen) or `autoset_utilization_host_memory` (for KVM)

These attributes default to `true`. This updates the utilization values in the CIB during each monitoring cycle. For more information, see **`crm ra info VirtualDomain`**.

> ### Note: `hv_memory` and `host_memory`
>
> In the `NodeUtilization` and `VirtualDomain` resource agents, `hv_memory` and `host_memory` both default to `true`. However, Xen only requires `hv_memory`, and KVM only requires `host_memory`. To avoid confusion, we recommend disabling the unnecessary attributes. For example:

EXAMPLE 7.6: CREATING RESOURCE AGENTS FOR KVM WITH `hv_memory` DISABLED

```
# crm configure primitive p_nu NodeUtilization \
      params utilization_hv_memory=false \
      op monitor timeout=20s interval=60
# crm configure primitive p_vm VirtualDomain \
      params autoset_utilization_hv_memory=false \
      op monitor timeout=30s interval=10s
```

```
# crm configure primitive p_nu NodeUtilization \
      params utilization_host_memory=false \
      op monitor timeout=20s interval=60
# crm configure primitive p_vm VirtualDomain \
      params autoset_utilization_host_memory=false \
      op monitor timeout=30s interval=10s
```

For either virtualization type, make sure to clone the `NodeUtilization` resource so it can run on all nodes:

```
# crm configure cl_nu p_nu
```

You can set the `placement-strategy` property to `balanced` with the following command:

```
# crm configure property placement-strategy=balanced
```

Independent of manually or automatically configuring capacity and requirements, the placement strategy must be specified with the `placement-strategy` property (in the global cluster options). The following values are available:

`default` (default value)

Utilization values are not considered. Resources are allocated according to location scoring. If scores are equal, resources are evenly distributed across nodes.

`utilization`

Utilization values are considered when deciding if a node has enough free capacity to satisfy a resource's requirements. However, load-balancing is still done based on the number of resources allocated to a node.

`minimal`

Utilization values are considered when deciding if a node has enough free capacity to satisfy a resource's requirements. An attempt is made to concentrate the resources on as few nodes as possible (to achieve power savings on the remaining nodes).

`balanced`

Utilization values are considered when deciding if a node has enough free capacity to satisfy a resource's requirements. An attempt is made to distribute the resources evenly, thus optimizing resource performance.

> ◈ **Note: Configuring resource priorities**
>
> The available placement strategies are best-effort—they do not yet use complex heuristic solvers to always reach optimum allocation results. Ensure that resource priorities are properly set so that your most important resources are scheduled first.

7.10.1 Placing resources based on their load impact with Hawk2

Utilization attributes are used to configure both the resource's requirements and the capacity a node provides. You first need to configure a node's capacity before you can configure the capacity a resource requires.

PROCEDURE 7.7: CONFIGURING THE CAPACITY A NODE PROVIDES

1. Log in to Hawk2:

   ```
   https://HAWKSERVER:7630/
   ```

2. From the left navigation bar, select *Monitoring* › *Status*.

3. On the *Nodes* tab, select the node whose capacity you want to configure.

4. In the *Operations* column, click the arrow down icon and select *Edit*.
   The *Edit Node* screen opens.

5. Below *Utilization*, enter a name for a utilization attribute into the empty drop-down box. The name can be arbitrary (for example, `RAM_in_GB`).

6. Click the *Add* icon to add the attribute.

7. In the empty text box next to the attribute, enter an attribute value. The value must be an integer.
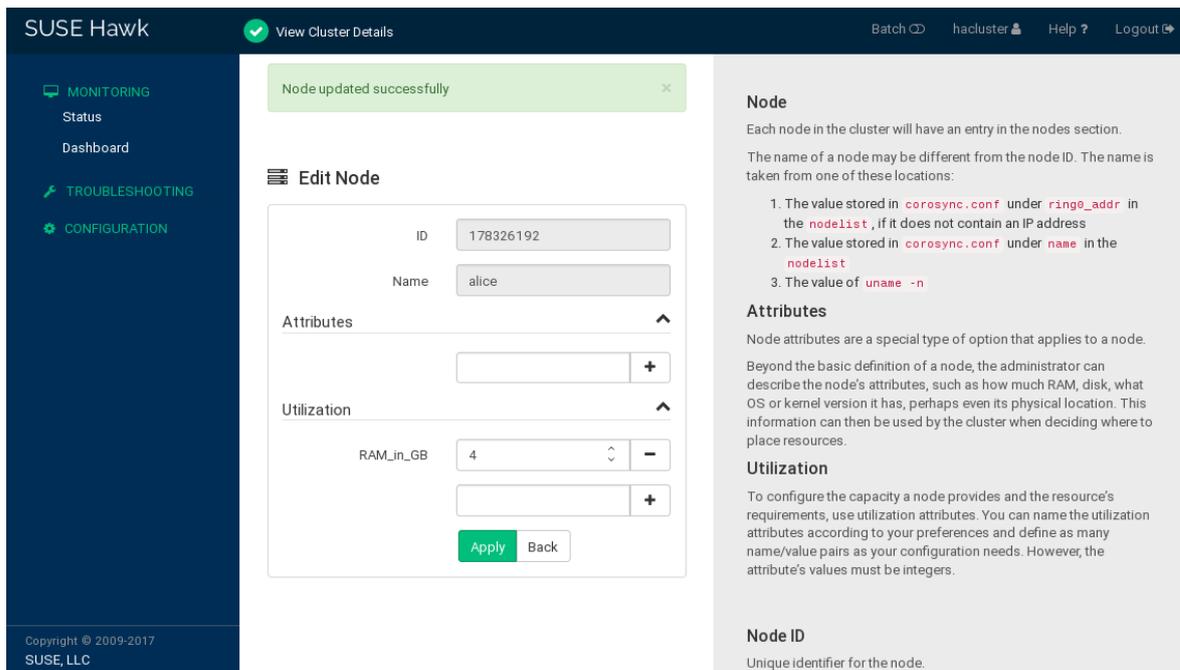
FIGURE 7.7: HAWK2: EDIT NODE UTILIZATION ATTRIBUTES

8. Add as many utilization attributes as you need and add values for all of them.

9. Confirm your changes. A message at the top of the screen shows if the action has been successful.

PROCEDURE 7.8: CONFIGURING THE CAPACITY A RESOURCE REQUIRES

Configure the capacity a certain resource requires from a node either when creating a primitive resource or when editing an existing primitive resource.

Before you can add utilization attributes to a resource, you need to have set utilization attributes for your cluster nodes as described in *Procedure 7.7*.

1. Log in to Hawk2:

```
https://HAWKSERVER:7630/
```

2. To add a utilization attribute to an existing resource, go to *Manage* › *Status* and open the resource configuration dialog as described in *Section 8.2.1, "Editing resources and groups with Hawk2"*.
If you create a new resource, go to *Configuration* › *Add Resource* and proceed as described in *Section 6.4.1, "Creating primitive resources with Hawk2"*.

3. In the resource configuration dialog, go to the *Utilization* category.

4. From the empty drop-down box, select one of the utilization attributes that you have configured for the nodes in *Procedure 7.7*.

5. In the empty text box next to the attribute, enter an attribute value. The value must be an integer.

6. Add as many utilization attributes as you need and add values for all of them.

7. Confirm your changes. A message at the top of the screen shows if the action has been successful.

After you have configured the capacities your nodes provide and the capacities your resources require, set the placement strategy in the global cluster options. Otherwise, the capacity configurations have no effect. Several strategies are available to schedule the load: for example, you can concentrate it on as few nodes as possible, or balance it evenly over all available nodes.

PROCEDURE 7.9: SETTING THE PLACEMENT STRATEGY

1. Log in to Hawk2:

   ```
   https://HAWKSERVER:7630/
   ```

2. From the left navigation bar, select *Configuration* › *Cluster Configuration* to open the respective screen. It shows global cluster options and resource and operation defaults.

3. From the empty drop-down box in the upper part of the screen, select `placement-strategy`.
   By default, its value is set to *Default*, which means that utilization attributes and values are not considered.

4. Depending on your requirements, set *Placement Strategy* to the appropriate value.

5. Confirm your changes.

7.10.2   Placing resources based on their load impact with crmsh

To configure the resource's requirements and the capacity a node provides, use utilization attributes. You can name the utilization attributes according to your preferences and define as many name/value pairs as your configuration needs. In certain cases, some agents update the utilization themselves, for example the `VirtualDomain`.

In the following example, we assume that you already have a basic configuration of cluster nodes and resources. You now additionally want to configure the capacities a certain node provides and the capacity a certain resource requires.

PROCEDURE 7.10: ADDING OR MODIFYING UTILIZATION ATTRIBUTES WITH `crm`

1. Log in as `root` and start the `crm` interactive shell:

   ```
   # crm configure
   ```

2. To specify the capacity a node *provides,* use the following command and replace the place-holder `NODE_1` with the name of your node:

   ```
   crm(live)configure# node NODE_1 utilization hv_memory=16384 cpu=8
   ```

   With these values, `NODE_1` would be assumed to provide 16 GB of memory and 8 CPU cores to resources.

3. To specify the capacity a resource *requires,* use:

   ```
   crm(live)configure# primitive xen1 Xen ... \
           utilization hv_memory=4096 cpu=4
   ```

   This would make the resource consume 4096 of those memory units from `NODE_1`, and 4 of the CPU units.

4. Configure the placement strategy with the `property` command:

   ```
   crm(live)configure# property ...
   ```

   The following values are available:

   `default` (default value)
   > Utilization values are not considered. Resources are allocated according to location scoring. If scores are equal, resources are evenly distributed across nodes.

   `utilization`
   > Utilization values are considered when deciding if a node has enough free capacity to satisfy a resource's requirements. However, load-balancing is still done based on the number of resources allocated to a node.

Utilization values are considered when deciding if a node has enough free capacity to satisfy a resource's requirements. An attempt is made to concentrate the resources on as few nodes as possible (to achieve power savings on the remaining nodes).

balanced

Utilization values are considered when deciding if a node has enough free capacity to satisfy a resource's requirements. An attempt is made to distribute the resources evenly, thus optimizing resource performance.

> ### Note: Configuring resource priorities
>
> The available placement strategies are best-effort—they do not yet use complex heuristic solvers to always reach optimum allocation results. Ensure that resource priorities are properly set so that your most important resources are scheduled first.

5. Commit your changes before leaving crmsh:

```
crm(live)configure# commit
```

The following example demonstrates a three node cluster of equal nodes, with four virtual machines:

```
crm(live)configure# node alice utilization hv_memory="4000"
crm(live)configure# node bob utilization hv_memory="4000"
crm(live)configure# node charlie utilization hv_memory="4000"
crm(live)configure# primitive xenA Xen \
    utilization hv_memory="3500" meta priority="10" \
    params xmfile="/etc/xen/shared-vm/vm1"
crm(live)configure# primitive xenB Xen \
    utilization hv_memory="2000" meta priority="1" \
    params xmfile="/etc/xen/shared-vm/vm2"
crm(live)configure# primitive xenC Xen \
    utilization hv_memory="2000" meta priority="1" \
    params xmfile="/etc/xen/shared-vm/vm3"
crm(live)configure# primitive xenD Xen \
    utilization hv_memory="1000" meta priority="5" \
    params xmfile="/etc/xen/shared-vm/vm4"
crm(live)configure# property placement-strategy="minimal"
```

With all three nodes up, xenA is placed onto a node first, followed by xenD. xenB and xenC would either be allocated together or one of them with xenD.

If one node failed, too little total memory would be available to host them all. xenA would be ensured to be allocated, as would xenD. However, only one of xenB or xenC could still be placed, and since their priority is equal, the result is not defined yet. To resolve this ambiguity as well, you would need to set a higher priority for either one.

7.11 For more information

For more information on configuring constraints and detailed background information about the basic concepts of ordering and colocation, refer to the following documents at https://www.clusterlabs.org/pacemaker/doc/ ↗:

- *Pacemaker Explained,* chapter *Resource Constraints*

- *Colocation Explained*

- *Ordering Explained*

8 Managing cluster resources

After configuring the resources in the cluster, use the cluster management tools to start, stop, clean up, remove or migrate the resources. This chapter describes how to use Hawk2 or crmsh for resource management tasks.

8.1 Showing cluster resources

8.1.1 Showing cluster resources with crmsh

When administering a cluster the command `crm configure show` lists the current CIB objects like cluster configuration, global options, primitives, and others:

```
# crm configure show
node 178326192: alice
node 178326448: bob
primitive admin_addr IPaddr2 \
        params ip=192.168.2.1 \
        op monitor interval=10 timeout=20
primitive stonith-sbd stonith:external/sbd \
        params pcmk_delay_max=30
property cib-bootstrap-options: \
        have-watchdog=true \
        dc-version=1.1.15-17.1-e174ec8 \
        cluster-infrastructure=corosync \
        cluster-name=hacluster \
        stonith-enabled=true \
        placement-strategy=balanced \
        standby-mode=true
rsc_defaults rsc-options: \
        resource-stickiness=1 \
        migration-threshold=3
op_defaults op-options: \
        timeout=600 \
        record-pending=true
```

If you have lots of resources, the output of `show` is too verbose. To restrict the output, use the name of the resource. For example, to list the properties of the primitive `admin_addr` only, append the resource name to `show`:

```
# crm configure show admin_addr
```

```
primitive admin_addr IPaddr2 \
        params ip=192.168.2.1 \
        op monitor interval=10 timeout=20
```

However, in some cases, you want to limit the output of specific resources even more. This can be achieved with *filters*. Filters limit the output to specific components. For example, to list the nodes only, use `type:node`:

```
# crm configure show type:node
node 178326192: alice
node 178326448: bob
```

If you are also interested in primitives, use the `or` operator:

```
# crm configure show type:node or type:primitive
node 178326192: alice
node 178326448: bob
primitive admin_addr IPaddr2 \
        params ip=192.168.2.1 \
        op monitor interval=10 timeout=20
primitive stonith-sbd stonith:external/sbd \
        params pcmk_delay_max=30
```

Furthermore, to search for an object that starts with a certain string, use this notation:

```
# crm configure show type:primitive and 'admin*'
primitive admin_addr IPaddr2 \
        params ip=192.168.2.1 \
        op monitor interval=10 timeout=20
```

To list all available types, enter **`crm configure show type:`** and press the `⇥|` key. The Bash completion gives you a list of all types.

8.2 Editing resources and groups

You can edit resources or groups using either Hawk2 or crmsh.

8.2.1 Editing resources and groups with Hawk2

If you have created a resource, you can edit its configuration at any time by adjusting parameters, operations or meta attributes as needed.

1. Log in to Hawk2:

```
https://HAWKSERVER:7630/
```

2. On the Hawk2 *Status* screen, go to the *Resources* list.

3. In the *Operations* column, click the arrow down icon next to the resource or group you want to modify and select *Edit*.
   The resource configuration screen opens.
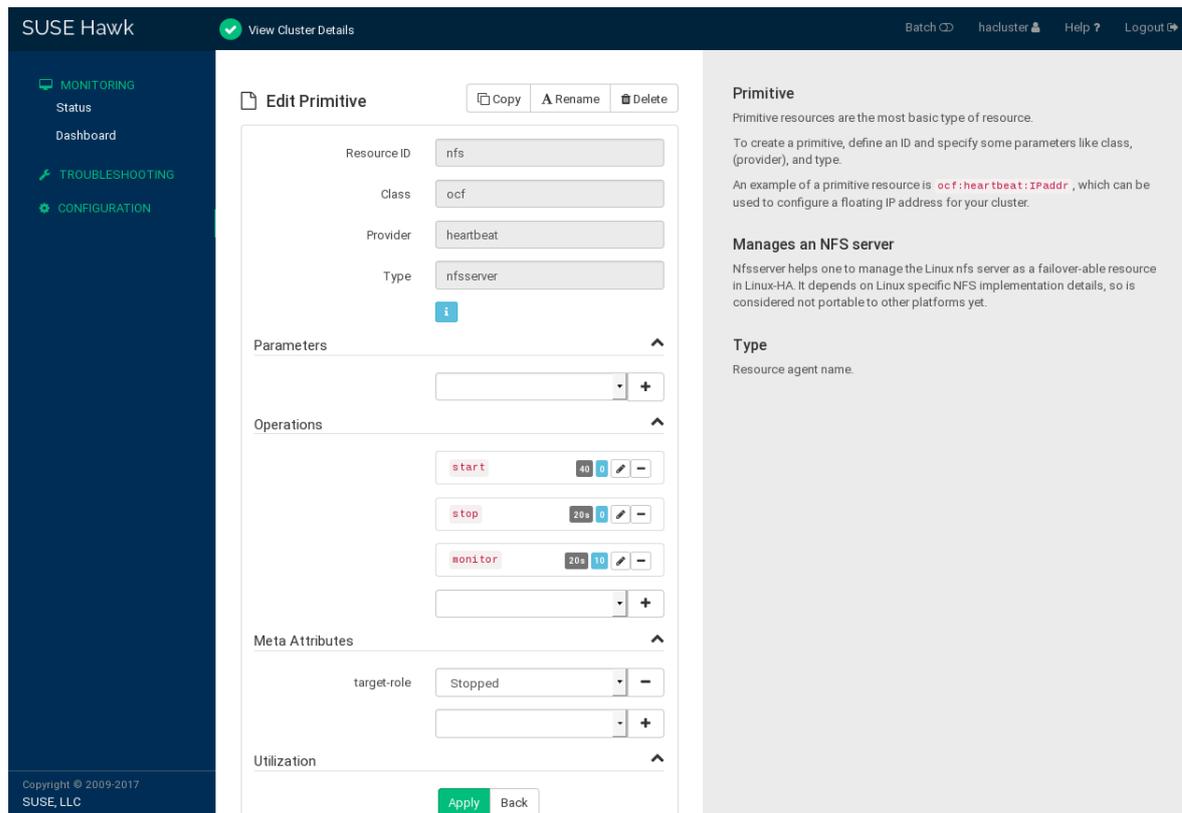
FIGURE 8.1: HAWK2—EDITING A PRIMITIVE RESOURCE

4. At the top of the configuration screen, you can select operations to perform.
   If you edit a primitive resource, the following operations are available:

   - Copying the resource

   - Renaming the resource (changing its ID)

   - Deleting the resource

If you edit a group, the following operations are available:

- Creating a new primitive to be added to this group

- Renaming the group (changing its ID)

- Dragging and dropping group members into a new order

5. To add a new parameter, operation or meta attribute, select an entry from the empty drop-down box.

6. To edit any values in the *Operations* category, click the *Edit* icon of the respective entry, enter a different value for the operation, and click *Apply*.

7. When you are finished, click the *Apply* button in the resource configuration screen to confirm your changes to the parameters, operations or meta attributes.
A message at the top of the screen shows if the action has been successful.

8.2.2 Editing groups with crmsh

To change the order of a group member, use the `modgroup` command from the `configure` subcommand. For example, use the following command to move the primitive `Email` before `Public-IP`:

```
crm(live)configure# modgroup g-mailsvc add Email before Public-IP
```

To remove a resource from a group (for example, `Email`), use this command:

```
crm(live)configure# modgroup g-mailsvc remove Email
```

8.3 Starting cluster resources

Before you start a cluster resource, make sure it is set up correctly. For example, if you use an Apache server as a cluster resource, set up the Apache server first. Complete the Apache configuration before starting the respective resource in your cluster.

Note: Do not touch services managed by the cluster

When managing a resource via the High Availability software, the resource must not be started or stopped otherwise (outside the cluster, for example manually or on boot or reboot). The High Availability software is responsible for all service start or stop actions.

However, if you want to check if the service is configured properly, start it manually, but make sure that it is stopped again before the High Availability software takes over.

For interventions in resources that are currently managed by the cluster, set the resource to `maintenance mode` first. For details, see *Procedure 28.5, "Putting a resource into maintenance mode with Hawk2"*.

You can start a cluster resource using either Hawk2 or crmsh.

8.3.1 Starting cluster resources with Hawk2

When creating a resource with Hawk2, you can set its initial state with the `target-role` meta attribute. If you set its value to `stopped`, the resource does not start automatically after being created.

PROCEDURE 8.2: STARTING A NEW RESOURCE

1. Log in to Hawk2:

   ```
   https://HAWKSERVER:7630/
   ```

2. From the left navigation bar, select *Monitoring* › *Status*. The list of *Resources* also shows the *Status*.

3. Select the resource to start. In its *Operations* column, click the *Start* icon. To continue, confirm the message that appears.

When the resource has started, Hawk2 changes the resource's *Status* to green and shows on which node it is running.

8.3.2 Starting cluster resources with crmsh

To start a new cluster resource, you need the respective identifier.

1. Log in as `root` and start the `crm` interactive shell:

   ```
   # crm
   ```

2. Switch to the resource level:

   ```
   crm(live)# resource
   ```

3. Start the resource with **start** and press the ␤ key to show all known resources:

   ```
   crm(live)resource# start ID
   ```

8.4 Stopping cluster resources

8.4.1 Stopping cluster resources with crmsh

To stop one or more existing cluster resources, you need the respective identifiers.

PROCEDURE 8.4: STOPPING CLUSTER RESOURCES WITH CRMSH

1. Log in as `root` and start the `crm` interactive shell:

   ```
   # crm
   ```

2. Switch to the resource level:

   ```
   crm(live)# resource
   ```

3. Stop the resource with **stop** and press the ␤ key to show all known resources:

   ```
   crm(live)resource# stop ID
   ```

   You can stop multiple resources at once:

   ```
   crm(live)resource# stop ID1 ID2 ...
   ```

8.5 Cleaning up cluster resources

A resource is automatically restarted if it fails, but each failure increases the resource's fail count.

If a `migration-threshold` has been set for the resource, the node will no longer run the resource when the number of failures reaches the migration threshold.

By default, fail counts are not automatically reset. You can configure a fail count to be reset automatically by setting a `failure-timeout` option for the resource, or you can manually reset the fail count using either Hawk2 or crmsh.

8.5.1 Cleaning up cluster resources with Hawk2

PROCEDURE 8.5: CLEANING UP A RESOURCE

1. Log in to Hawk2:

   ```
   https://HAWKSERVER:7630/
   ```

2. From the left navigation bar, select *Status*. The list of *Resources* also shows the *Status*.

3. Go to the resource to clean up. In the *Operations* column, click the arrow down button and select *Cleanup*. To continue, confirm the message that appears.
   This executes the command `crm resource cleanup` and cleans up the resource on all nodes.

8.5.2 Cleaning up cluster resources with crmsh

PROCEDURE 8.6: CLEANING UP A RESOURCE WITH CRMSH

1. Open a shell and log in as user `root`.

2. Get a list of all your resources:

   ```
   # crm resource status
   Full List of Resources
       * admin-ip      (ocf:heartbeat:IPaddr2):    Started
       * stonith-sbd   (stonith:external/sbd):     Started
       * Resource Group: dlm-clvm:
         * dlm:        (ocf:pacemaker:controld)    Started
         * clvm:       (ocf:heartbeat:lvmlockd)    Started
   ```

3. Show the fail count of a resource:

   ```
   # crm resource failcount RESOURCE show NODE
   ```

For example, to show the fail count of the resource `dlm` on node `alice`:

```
# crm resource failcount dlm show alice
scope=status name=fail-count-dlm value=2
```

4. Clean up the resource:

```
# crm resource cleanup RESOURCE
```

This command cleans up the resource on all nodes. If the resource is part of a group, crmsh also cleans up the other resources in the group.

8.6 Removing cluster resources

To remove a resource from the cluster, follow either the Hawk2 or crmsh procedure below to avoid configuration errors.

8.6.1 Removing cluster resources with Hawk2

PROCEDURE 8.7: REMOVING A CLUSTER RESOURCE

1. Log in to Hawk2:

```
https://HAWKSERVER:7630/
```

2. Clean up the resource on all nodes as described in *Procedure 8.5, "Cleaning up a resource"*.

3. Stop the resource:

   a. From the left navigation bar, select *Monitoring* › *Status*. The list of *Resources* also shows the *Status*.

   b. In the *Operations* column, click the *Stop* button next to the resource.

   c. To continue, confirm the message that appears.
      The *Status* column will reflect the change when the resource is stopped.

4. Delete the resource:

   a. From the left navigation bar, select *Configuration* › *Edit Configuration.*

b. In the list of *Resources,* go to the respective resource. From the *Operations* column, click the *Delete* icon next to the resource.

c. To continue, confirm the message that appears.

8.6.2 Removing cluster resources with crmsh

PROCEDURE 8.8: REMOVING A CLUSTER RESOURCE WITH CRMSH

1. Log in as `root` and start the **crm** interactive shell:

```
# crm
```

2. Get a list of your resources:

```
crm(live)# resource status
Full List of Resources:
  * admin-ip      (ocf:heartbeat:IPaddr2):    Started
  * stonith-sbd   (stonith:external/sbd):     Started
  * nfsserver     (ocf:heartbeat:nfsserver):  Started
```

3. Stop the resource you want to remove:

```
crm(live)# resource stop RESOURCE
```

4. Delete the resource:

```
crm(live)# configure delete RESOURCE
```

8.7 Migrating cluster resources

The cluster will fail over (migrate) resources automatically in case of software or hardware failures, according to certain parameters you can define (for example, migration threshold or resource stickiness). You can also manually migrate a resource to another node in the cluster, or move the resource away from the current node and let the cluster decide where to put it.

You can migrate a cluster resource using either Hawk2 or crmsh.

8.7.1   Migrating cluster resources with Hawk2

PROCEDURE 8.9: MANUALLY MIGRATING A RESOURCE

1. Log in to Hawk2:

   ```
   https://HAWKSERVER:7630/
   ```

2. From the left navigation bar, select *Monitoring* › *Status*. The list of *Resources* also shows the *Status*.

3. In the list of *Resources*, select the respective resource.

4. In the *Operations* column, click the arrow down button and select *Migrate*.

5. In the window that opens, you have the following choices:

   - *Away from current node*: This creates a location constraint with a `-INFINITY` score for the current node.

   - Alternatively, you can move the resource to another node. This creates a location constraint with an `INFINITY` score for the destination node.

6. Confirm your choice.

To allow a resource to move back again, proceed as follows:

PROCEDURE 8.10: UNMIGRATING A RESOURCE

1. Log in to Hawk2:

   ```
   https://HAWKSERVER:7630/
   ```

2. From the left navigation bar, select *Monitoring* › *Status*. The list of *Resources* also shows the *Status*.

3. In the list of *Resources,* go to the respective resource.

4. In the *Operations* column, click the arrow down button and select *Clear*. To continue, confirm the message that appears.
   Hawk2 uses the `crm_resource` `--clear` command. The resource can move back to its original location or it may stay where it is (depending on resource stickiness).

For more information, see *Pacemaker Explained*, available from https://www.clusterlabs.org/pacemaker/doc/ ↗. Refer to section *Resource Migration*.

8.7.2   Migrating cluster resources with crmsh

Use the **move** command for this task. For example, to migrate the resource `ipaddress1` to a cluster node named `bob`, use these commands:

```
# crm resource
 crm(live)resource# move ipaddress1 bob
```

8.8   Grouping resources by using tags

Tags are a way to refer to multiple resources at once, without creating any colocation or ordering relationship between them. This can be useful for grouping conceptually related resources. For example, if you have several resources related to a database, create a tag called `databases` and add all resources related to the database to this tag. This allows you to stop or start them all with a single command.

Tags can also be used in constraints. For example, the following location constraint `loc-db-prefer` applies to the set of resources tagged with `databases`:

```
location loc-db-prefer databases 100: alice
```

You can create tags using either Hawk2 or crmsh.

8.8.1   Grouping resources by using tags with Hawk2

PROCEDURE 8.11: ADDING A TAG

1. Log in to Hawk2:

   ```
   https://HAWKSERVER:7630/
   ```

2. From the left navigation bar, select *Configuration* › *Add Resource* › *Tag*.

3. Enter a unique *Tag ID*.

4. From the *Objects* list, select the resources you want to refer to with the tag.

5. Click *Create* to finish the configuration. A message at the top of the screen shows if the action has been successful.
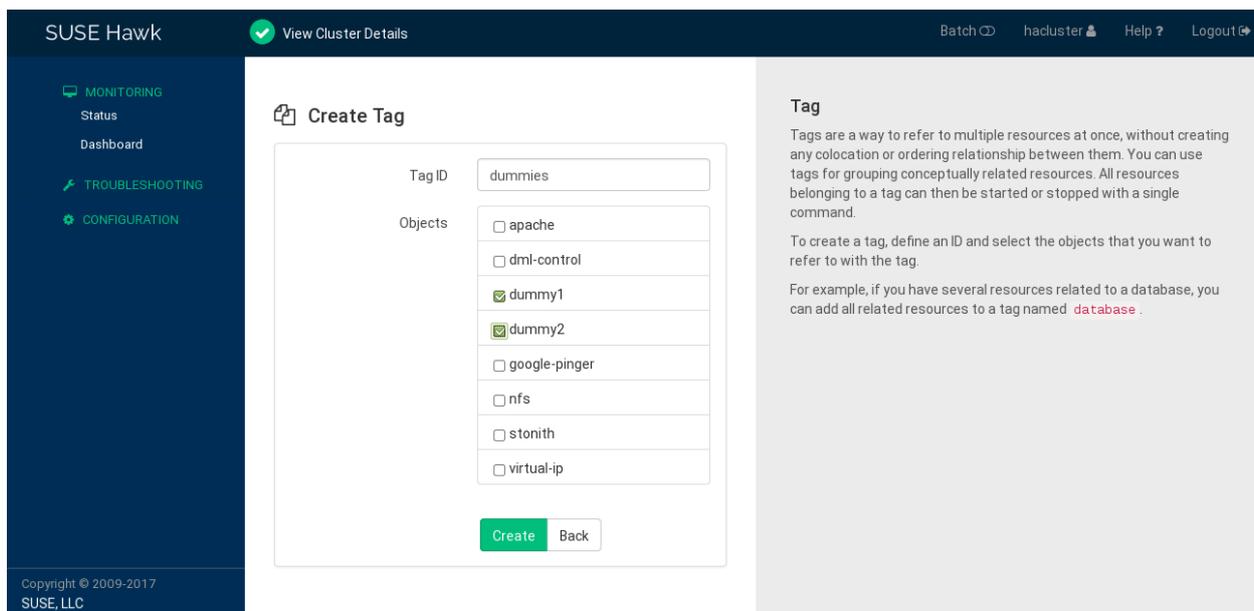
FIGURE 8.2: HAWK2—TAG

8.8.2 Grouping resources by using tags with crmsh

For example, if you have several resources related to a database, create a tag called `databases` and add all resources related to the database to this tag:

```
# crm configure tag databases: db1 db2 db3
```

This allows you to start them all with a single command:

```
# crm resource start databases
```

Similarly, you can stop them all too:

```
# crm resource stop databases
```

9 Managing services on remote hosts

The possibilities for monitoring and managing services on remote hosts has become increasingly important during the last few years. SUSE Linux Enterprise High Availability 11 SP3 offered fine-grained monitoring of services on remote hosts via monitoring plug-ins. The recent addition of the `pacemaker_remote` service now allows SUSE Linux Enterprise High Availability 15 SP5 to fully manage and monitor resources on remote hosts just as if they were a real cluster node—without the need to install the cluster stack on the remote machines.

9.1 Monitoring services on remote hosts with monitoring plug-ins

Monitoring of virtual machines can be done with the VM agent (which only checks if the guest shows up in the hypervisor), or by external scripts called from the VirtualDomain or Xen agent. Up to now, more fine-grained monitoring was only possible with a full setup of the High Availability stack within the virtual machines.

By providing support for monitoring plug-ins (formerly named Nagios plug-ins), SUSE Linux Enterprise High Availability now also allows you to monitor services on remote hosts. You can collect external statuses on the guests without modifying the guest image. For example, VM guests might run Web services or simple network resources that need to be accessible. With the Nagios resource agents, you can now monitor the Web service or the network resource on the guest. If these services are not reachable anymore, SUSE Linux Enterprise High Availability triggers a restart or migration of the respective guest.

If your guests depend on a service (for example, an NFS server to be used by the guest), the service can either be an ordinary resource, managed by the cluster, or an external service that is monitored with Nagios resources instead.

To configure the Nagios resources, the following packages must be installed on the host:

- `monitoring-plugins`

- `monitoring-plugins-metadata`

YaST or Zypper will resolve any dependencies on further packages, if required.

A typical use case is to configure the monitoring plug-ins as resources belonging to a resource container, which usually is a VM. The container is restarted if any of its resources fail. Refer to *Example 9.1, "Configuring resources for monitoring plug-ins"* for a configuration example. Alternatively, Nagios resource agents can also be configured as ordinary resources to be used for monitoring hosts or services via the network.

EXAMPLE 9.1: CONFIGURING RESOURCES FOR MONITORING PLUG-INS

```
primitive vm1 VirtualDomain \
    params hypervisor="qemu:///system" config="/etc/libvirt/qemu/vm1.xml" \
    op start interval="0" timeout="90" \
    op stop interval="0" timeout="90" \
    op monitor interval="10" timeout="30"
primitive vm1-sshd nagios:check_tcp \
    params hostname="vm1" port="22" \ ❶
    op start interval="0" timeout="120" \ ❷
    op monitor interval="10"
group g-vm1-and-services vm1 vm1-sshd \
    meta container="vm1" ❸
```

❶ The supported parameters are the same as the long options of a monitoring plug-in. Monitoring plug-ins connect to services with the parameter `hostname`. Therefore, the attribute's value must be a resolvable host name or an IP address.

❷ As it takes time to get the guest operating system up and its services running, the start timeout of the monitoring resource must be long enough.

❸ A cluster resource container of type `ocf:heartbeat:Xen`, `ocf:heartbeat:VirtualDomain` or `ocf:heartbeat:lxc`. It can either be a VM or a Linux Container.

The example above contains only one resource for the `check_tcp` plug-in, but multiple resources for different plug-in types can be configured (for example, `check_http` or `check_udp`).

If the host names of the services are the same, the `hostname` parameter can also be specified for the group, instead of adding it to the individual primitives. For example:

```
group g-vm1-and-services vm1 vm1-sshd vm1-httpd \
    meta container="vm1" \
    params hostname="vm1"
```

If any of the services monitored by the monitoring plug-ins fail within the VM, the cluster detects that and restarts the container resource (the VM). Which action to take in this case can be configured by specifying the `on-fail` attribute for the service's monitoring operation. It defaults to `restart-container`.

> Failure counts of services are taken into account when considering the VM's migration-threshold.

9.2 Managing services on remote nodes with `pacemaker_remote`

With the `pacemaker_remote` service, High Availability clusters can be extended to virtual nodes or remote bare-metal machines. They do not need to run the cluster stack to become members of the cluster.

SUSE Linux Enterprise High Availability can now launch virtual environments (KVM and LXC), plus the resources that live within those virtual environments without requiring the virtual environments to run Pacemaker or Corosync.

For the use case of managing both virtual machines as cluster resources plus the resources that live within the VMs, you can now use the following setup:

- The "normal" (bare-metal) cluster nodes run SUSE Linux Enterprise High Availability.

- The virtual machines run the `pacemaker_remote` service (almost no configuration required on the VM's side).

- The cluster stack on the "normal" cluster nodes launches the VMs and connects to the `pacemaker_remote` service running on the VMs to integrate them as remote nodes into the cluster.

As the remote nodes do not have the cluster stack installed, this has the following implications:

- Remote nodes do not take part in quorum.

- Remote nodes cannot become the DC.

- Remote nodes are not bound by the scalability limits (Corosync has a member limit of 32 nodes).

Find more information about the `remote_pacemaker` service, including multiple use cases with detailed setup instructions in *Article "Pacemaker Remote Quick Start"*.

10 Adding or modifying resource agents

All tasks that need to be managed by a cluster must be available as a resource.
There are two major groups here to consider: resource agents and STONITH agents.
For both categories, you can add your own agents, extending the abilities of the
cluster to your own needs.

10.1 STONITH agents

A cluster sometimes detects that one of the nodes is behaving strangely and needs to remove it.
This is called *fencing* and is commonly done with a STONITH resource.

✋ Warning: External SSH/STONITH are not supported

It is impossible to know how SSH might react to other system problems. For this reason,
external SSH/STONITH agents (like `stonith:external/ssh`) are not supported for pro-
duction environments. If you still want to use such agents for testing, install the `libglue-devel` package.

To get a list of all currently available STONITH devices (from the software side), use the com-
mand `crm ra list stonith`. If you do not find your favorite agent, install the `-devel` pack-
age. For more information on STONITH devices and resource agents, see *Chapter 12, Fencing and STONITH*.

There is currently no documentation about writing STONITH agents. If you want to write new
STONITH agents, consult the examples available in the source of the `cluster-glue` package.

10.2 Writing OCF resource agents

All OCF resource agents (RAs) are available in `/usr/lib/ocf/resource.d/`, see *Section 6.2,
"Supported resource agent classes"* for more information. Each resource agent must supported the
following operations to control it:

`start`

start or enable the resource

stop

> stop or disable the resource

status

> returns the status of the resource

monitor

> similar to `status`, but checks also for unexpected states

validate

> validate the resource's configuration

meta-data

> returns information about the resource agent in XML

The general procedure of how to create an OCF RA is like the following:

1. Load the file `/usr/lib/ocf/resource.d/pacemaker/Dummy` as a template.

2. Create a new subdirectory for each new resource agents to avoid naming contradictions. For example, if you have a resource group `kitchen` with the resource `coffee_machine`, add this resource to the directory `/usr/lib/ocf/resource.d/kitchen/`. To access this RA, execute the command `crm`:

   ```
   # crm configure primitive coffee_1 ocf:coffee_machine:kitchen ...
   ```

3. Implement the different shell functions and save your file under a different name.

More details about writing OCF resource agents can be found at https://www.clusterlabs.org/pacemaker/doc/ ↗ in the guide *Pacemaker Administration*. Find special information about several concepts at *Chapter 1, Product overview*.

10.3 OCF return codes and failure recovery

According to the OCF specification, there are strict definitions of the exit codes an action must return. The cluster always checks 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 a recovery action is initiated. There are three types of failure recovery:

TABLE 10.1: FAILURE RECOVERY TYPES

Recovery 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 occurred. The error may be specific to the current node.	Move the resource elsewhere and prevent it from being retried on the current node.
fatal	A non-transient error occurred that will be common to all cluster nodes. This means a bad configuration was specified.	Stop the resource and prevent it from being started on any cluster node.

Assuming an action is considered to have failed, the following table outlines the different OCF return codes. It also shows the type of recovery the cluster will initiate when the respective error code is received.

TABLE 10.2: OCF RETURN CODES

OCF Return Code	OCF Alias	Description	Recovery Type
0	OCF_SUCCESS	Success. The command completed successfully. This is the expected result for all start, stop, promote and demote commands.	soft
1	OCF_ERR_GENERIC	Generic "there was a problem" error code.	soft

OCF Return Code	OCF Alias	Description	Recovery Type
2	OCF_ERR_ARGS	The resource's configuration is not valid on this machine (for example, it refers to a location/tool not found on the node).	hard
3	OCF_ERR_UN-IMPLEMENTED	The requested action is not implemented.	hard
4	OCF_ERR_PERM	The resource agent does not have sufficient privileges to complete the task.	hard
5	OCF_ERR_-INSTALLED	The tools required by the resource are not installed on this machine.	hard
6	OCF_ERR_-CONFIGURED	The resource's configuration is invalid (for example, required parameters are missing).	fatal
7	OCF_NOT_-RUNNING	The resource is not running. The cluster will not attempt to stop a resource that returns this for any action. This OCF return code may or may not require resource recovery—it depends on what is the expected resource status. If unexpected, then `soft` recovery.	N/A
8	OCF_RUNNING_-PROMOTED	The resource is running in Promoted mode.	soft
9	OCF_FAILED_-PROMOTED	The resource is in Promoted mode but has failed. The resource will be demoted, stopped and then started (and possibly promoted) again.	soft
other	N/A	Custom error code.	soft

11 Monitoring clusters

This chapter describes how to monitor a cluster's health and view its history.

11.1 Monitoring cluster status

Hawk2 has different screens for monitoring single clusters and multiple clusters: the *Status* and the *Dashboard* screen.

11.1.1 Monitoring a single cluster

To monitor a single cluster, use the *Status* screen. After you have logged in to Hawk2, the *Status* screen is displayed by default. An icon in the upper-right corner shows the cluster status at a glance. For further details, take a look at the following categories:

Errors

  If errors have occurred, they are shown at the top of the page.

Resources

  Shows the configured resources including their *Status*, *Name* (ID), *Location* (node on which they are running), and resource agent *Type*. From the *Operations* column, you can start or stop a resource, trigger several actions, or view details. Actions that can be triggered include setting the resource to maintenance mode (or removing maintenance mode), migrating it to a different node, cleaning up the resource, showing any recent events, or editing the resource.

Nodes

  Shows the nodes belonging to the cluster site you are logged in to, including the nodes' *Status* and *Name*. In the *Maintenance* and *Standby* columns, you can set or remove the `maintenance` or `standby` flag for a node. The *Operations* column allows you to view recent events for the node or further details: for example, if a `utilization`, `standby` or `maintenance` attribute is set for the respective node.

Tickets

  Only shown if tickets have been configured (for use with Geo clustering).

FIGURE 11.1: HAWK2—CLUSTER STATUS

11.1.2 Monitoring multiple clusters

To monitor multiple clusters, use the Hawk2 *Dashboard*. The cluster information displayed in the *Dashboard* screen is stored on the server side. It is synchronized between the cluster nodes (if passwordless SSH access between the cluster nodes has been configured). For details, see *Section D.2, "Configuring a passwordless SSH account"*. However, the machine running Hawk2 does not even need to be part of any cluster for that purpose—it can be a separate, unrelated system.

In addition to the general *Hawk2 requirements*, the following prerequisites need to be fulfilled to monitor multiple clusters with Hawk2:

PREREQUISITES

- All clusters to be monitored from Hawk2's *Dashboard* must be running SUSE Linux Enterprise High Availability 15 SP5.

- If you did not replace the self-signed certificate for Hawk2 on every cluster node with your own certificate (or a certificate signed by an official Certificate Authority) yet, do the following: log in to Hawk2 on *every* node in *every* cluster at least once. Verify the certificate (or add an exception in the browser to bypass the warning). Otherwise, Hawk2 cannot connect to the cluster.

Monitoring multiple clusters SLE HA 15 SP5

1. Log in to Hawk2:

   ```
   https://HAWKSERVER:7630/
   ```

2. From the left navigation bar, select *Monitoring › Dashboard.*
   Hawk2 shows an overview of the resources and nodes on the current cluster site. In addition, it shows any *Tickets* that have been configured for use with a Geo cluster. If you need information about the icons used in this view, click *Legend*. To search for a resource ID, enter the name (ID) into the *Search* text box. To only show specific nodes, click the filter icon and select a filtering option.
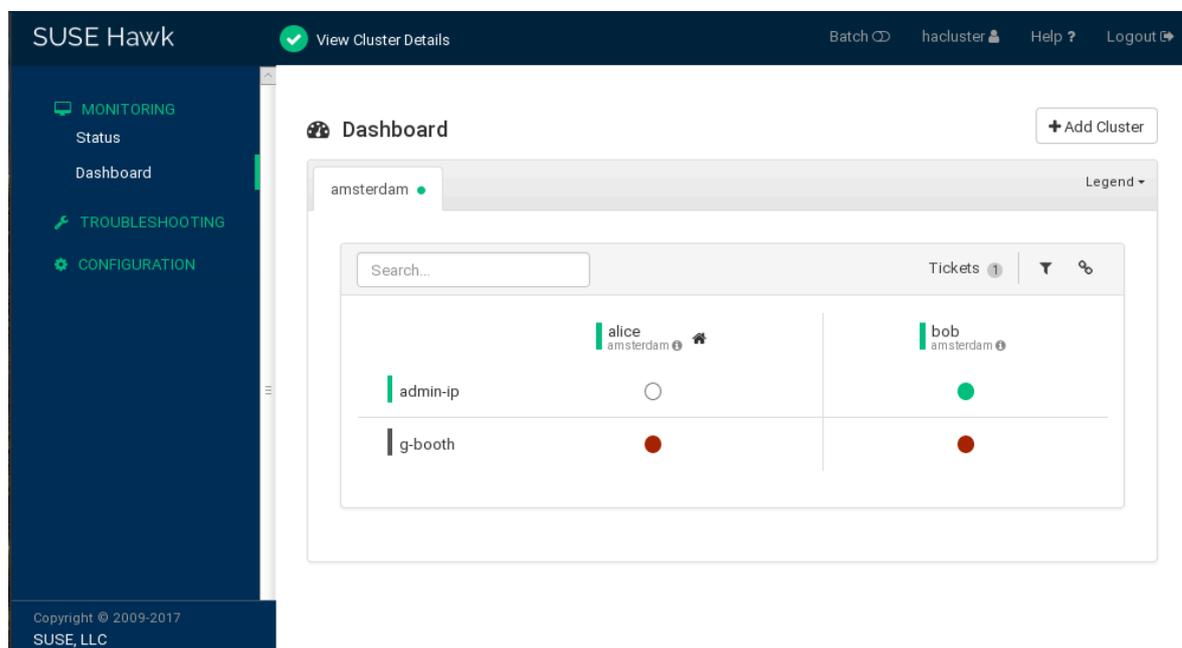
   FIGURE 11.2: HAWK2 DASHBOARD WITH ONE CLUSTER SITE (`amsterdam`)

3. To add dashboards for multiple clusters:

   a. Click *Add Cluster*.

   b. Enter the *Cluster name* with which to identify the cluster in the *Dashboard*. For example, `berlin`.

   c. Enter the fully qualified host name of one of the nodes in the second cluster. For example, `charlie`.
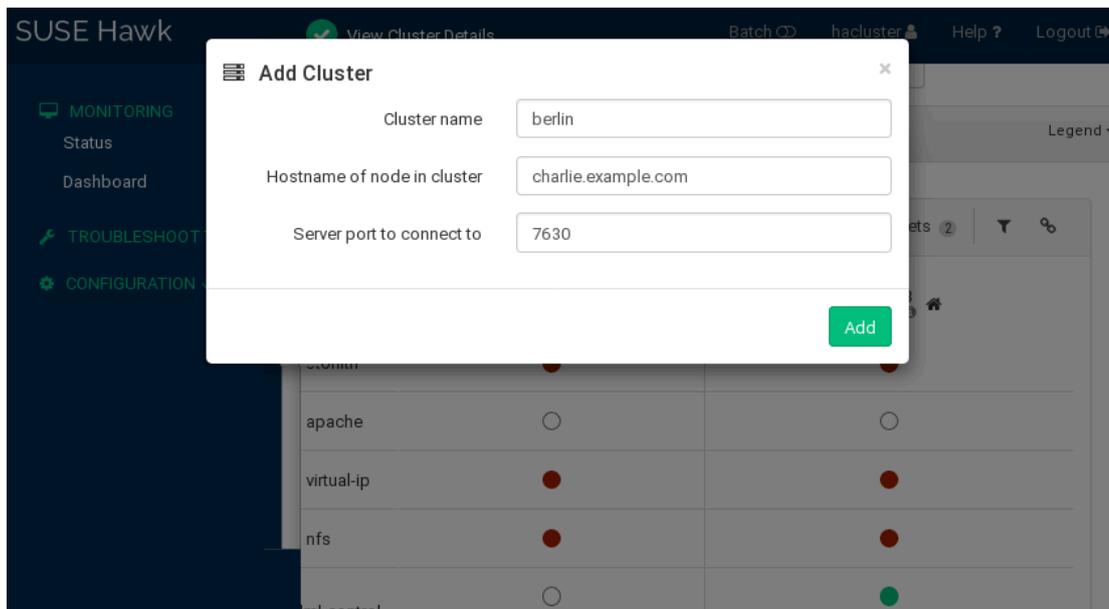
FIGURE 11.3: HAWK2 ADD CLUSTER DETAILS

d. Click *Add*. Hawk2 displays a second tab for the newly added cluster site with an overview of its nodes and resources.

> ### Note: Connection error
>
> If instead you are prompted to log in to this node by entering a password, you might not be connected to this node yet and have not replaced the self-signed certificate. In that case, even after you enter the password, the connection fails with the following message: `Error connecting to server. Retrying every 5 seconds... '`.
>
> To proceed, see *Replacing the self-signed certificate*.

4. To view more details for a cluster site or to manage it, switch to the site's tab and click the chain icon.

   Hawk2 opens the *Status* view for this site in a new browser window or tab. From there, you can administer this part of the Geo cluster.

5. To remove a cluster from the dashboard, click the `x` icon on the right-hand side of the cluster's details.

11.2 Verifying cluster health

You can check the health of a cluster using either Hawk2 or crmsh.

11.2.1 Verifying cluster health with Hawk2

Hawk2 provides a wizard that checks and detects issues with your cluster. After the analysis is complete, Hawk2 creates a cluster report with further details. To verify cluster health and generate the report, Hawk2 requires passwordless SSH access between the nodes. Otherwise, it can only collect data from the current node. If you have set up your cluster with the bootstrap scripts, which are provided by the crm shell, passwordless SSH access is already configured. If you need to configure it manually, see *Section D.2, "Configuring a passwordless SSH account"*.

1. Log in to Hawk2:

   ```
   https://HAWKSERVER:7630/
   ```

2. From the left navigation bar, select *Configuration* › *Wizards*.

3. Expand the *Basic* category.

4. Select the *Verify health and configuration* wizard.

5. Confirm with *Verify*.

6. Enter the root password for your cluster and click *Apply*. Hawk2 generates the report.

11.2.2 Getting health status with crmsh

The "health" status of a cluster or node can be displayed with so called *scripts*. A script can perform different tasks—they are not targeted to health. However, for this subsection, we focus on how to get the health status.

To get all the details about the **health** command, use **describe**:

```
# crm script describe health
```

It shows a description and a list of all parameters and their default values. To execute a script, use **run**:

```
# crm script run health
```

If you prefer to run only one step from the suite, the `describe` command lists all available steps in the *Steps* category.

For example, the following command executes the first step of the `health` command. The output is stored in the `health.json` file for further investigation:

```
# crm script run health statefile='health.json'
```

It is also possible to run the above commands with `crm cluster health`.

For additional information regarding scripts, see https://crmsh.github.io/scripts/ ↗.

11.3 Viewing the cluster history

Hawk2 provides the following possibilities to view past events on the cluster (on different levels and in varying detail):

- *Section 11.3.1, "Viewing recent events of nodes or resources"*

- *Section 11.3.2, "Using the history explorer for cluster reports"*

- *Section 11.3.3, "Viewing transition details in the history explorer"*

You can also view cluster history information using crmsh:

- *Section 11.3.4, "Retrieving history information with crmsh"*

11.3.1 Viewing recent events of nodes or resources

1. Log in to Hawk2:

   ```
   https://HAWKSERVER:7630/
   ```

2. From the left navigation bar, select *Monitoring* › *Status*. It lists *Resources* and *Nodes*.

3. To view recent events of a resource:

   a. Click *Resources* and select the respective resource.

   b. In the *Operations* column for the resource, click the arrow down button and select *Recent events*.

Hawk2 opens a new window and displays a table view of the latest events.

4. To view recent events of a node:

   a. Click *Nodes* and select the respective node.

   b. In the *Operations* column for the node, select *Recent events*.
      Hawk2 opens a new window and displays a table view of the latest events.
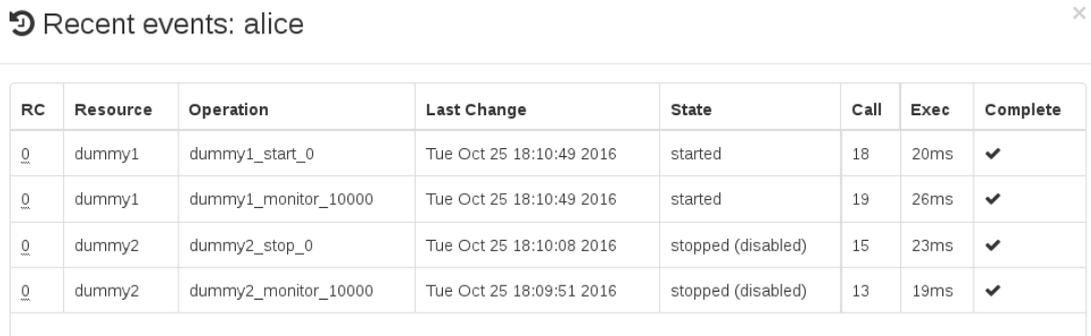
↺ Recent events: alice   ✕

RC	Resource	Operation	Last Change	State	Call	Exec	Complete
0	dummy1	dummy1_start_0	Tue Oct 25 18:10:49 2016	started	18	20ms	✔
0	dummy1	dummy1_monitor_10000	Tue Oct 25 18:10:49 2016	started	19	26ms	✔
0	dummy2	dummy2_stop_0	Tue Oct 25 18:10:08 2016	stopped (disabled)	15	23ms	✔
0	dummy2	dummy2_monitor_10000	Tue Oct 25 18:09:51 2016	stopped (disabled)	13	19ms	✔

FIGURE 11.4: RECENT EVENTS TABLE

11.3.2   Using the history explorer for cluster reports

From the left navigation bar, select *Troubleshooting* › *History* to access the *History Explorer*. The *History Explorer* allows you to create detailed cluster reports and view transition information. It provides the following options:

Generate

Create a cluster report for a certain time. Hawk2 calls the `crm report` command to generate the report.

Upload

Allows you to upload `crm report` archives that have either been created with the crm shell directly or even on a different cluster.

After reports have been generated or uploaded, they are shown below *Reports*. From the list of reports, you can show a report's details, or download or delete the report.
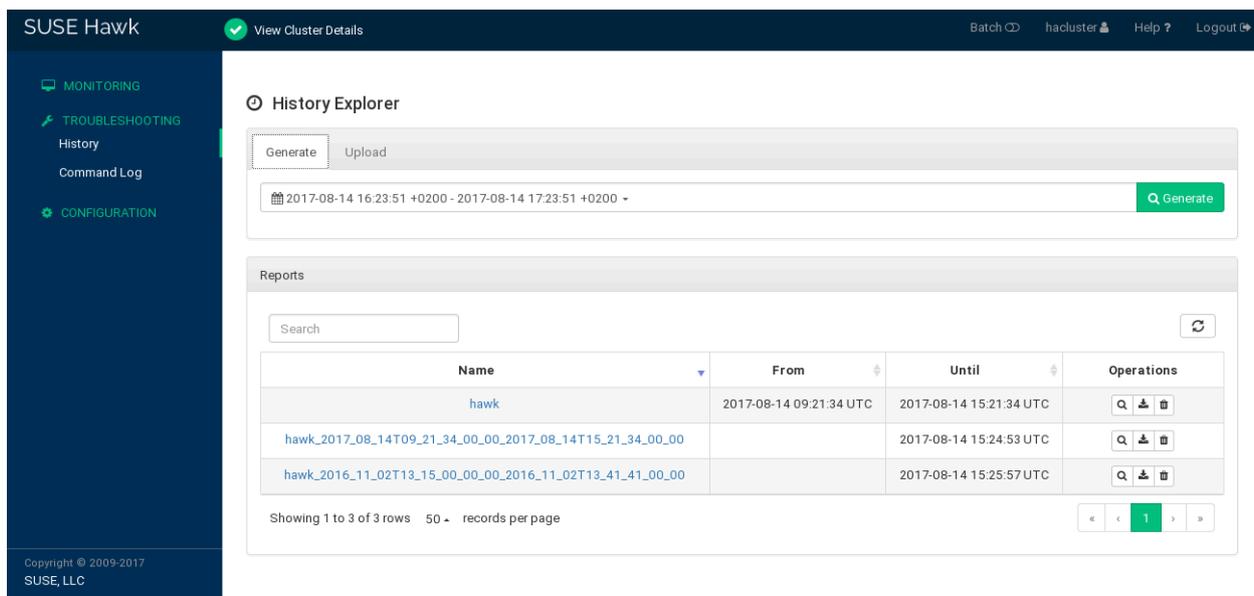
FIGURE 11.5: HAWK2—HISTORY EXPLORER MAIN VIEW

PROCEDURE 11.2: GENERATING OR UPLOADING A CLUSTER REPORT

1. Log in to Hawk2:

   ```
   https://HAWKSERVER:7630/
   ```

2. From the left navigation bar, select *Troubleshooting* › *History*.

   The *History Explorer* screen opens in the *Generate* view. By default, the suggested time frame for a report is the last hour.

3. To create a cluster report:

   a. To immediately start a report, click *Generate*.

   b. To modify the time frame for the report, click anywhere on the suggested time frame and select another option from the drop-down box. You can also enter a *Custom* start date, end date and hour, respectively. To start the report, click *Generate*.

      After the report has finished, it is shown below *Reports*.

4. To upload a cluster report, the `crm report` archive must be located on a file system that you can access with Hawk2. Proceed as follows:

   a. Switch to the *Upload* tab.

   b. *Browse* for the cluster report archive and click *Upload*.

      After the report is uploaded, it is shown below *Reports*.

5. To download or delete a report, click the respective icon next to the report in the *Operations* column.

6. To view *Report details in history explorer*, click the report's name or select *Show* from the *Operations* column.
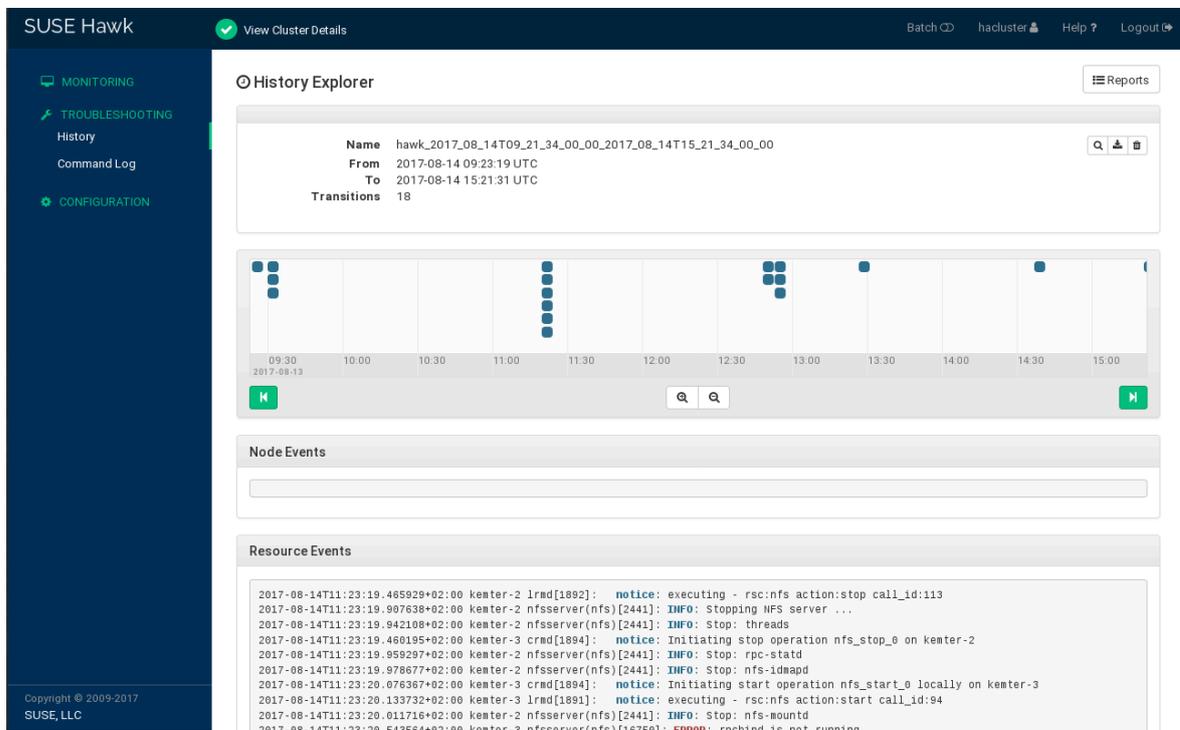
FIGURE 11.6: HAWK2 REPORT DETAILS

7. Return to the list of reports by clicking the *Reports* button.

REPORT DETAILS IN HISTORY EXPLORER

- Name of the report.

- Start time of the report.

- End time of the report.

- Number of transitions plus time line of all transitions in the cluster that are covered by the report. To learn how to view more details for a transition, see *Section 11.3.3*.

- Node events.

- Resource events.

11.3.3 Viewing transition details in the history explorer

For each transition, the cluster saves a copy of the state which it provides as input to `pacemaker-schedulerd`. The path to this archive is logged. All `pe-*` files are generated on the Designated Coordinator (DC). As the DC can change in a cluster, there may be `pe-*` files from several nodes. Any `pe-*` files are saved snapshots of the CIB, used as input of calculations by `pacemaker-schedulerd`.

In Hawk2, you can display the name of each `pe-*` file plus the time and node on which it was created. The *History Explorer* can also visualize the following details, based on the respective `pe-*` file:

TRANSITION DETAILS IN THE HISTORY EXPLORER

Details

    Shows snippets of logging data that belongs to the transition. Displays the output of the following command (including the resource agents' log messages):

```
crm history transition peinput
```

Configuration

    Shows the cluster configuration at the time that the `pe-*` file was created.

Diff

    Shows the differences of configuration and status between the selected `pe-*` file and the following one.

Log

    Shows snippets of logging data that belongs to the transition. Displays the output of the following command:

```
crm history transition log peinput
```

    This includes details from the following daemons: `pacemaker-schedulerd`, `pacemaker-controld` and `pacemaker-execd`.

Graph

    Shows a graphical representation of the transition. If you click *Graph*, the calculation is simulated (exactly as done by `pacemaker-schedulerd`) and a graphical visualization is generated.

1. Log in to Hawk2:

   ```
   https://HAWKSERVER:7630/
   ```

2. From the left navigation bar, select *Troubleshooting* › *History*.
   If reports have already been generated or uploaded, they are shown in the list of *Reports*. Otherwise generate or upload a report as described in *Procedure 11.2*.

3. Click the report's name or select *Show* from the *Operations* column to open the *Report details in history explorer*.

4. To access the transition details, you need to select a transition point in the transition time line that is shown below. Use the *Previous* and *Next* icons and the *Zoom In* and *Zoom Out* icons to find the transition that you are interested in.

5. To display the name of a `pe-input*` file plus the time and node on which it was created, hover the mouse pointer over a transition point in the time line.

6. To view the *Transition details in the history explorer*, click the transition point for which you want to know more.

7. To show *Details*, *Configuration*, *Diff*, *Logs* or *Graph*, click the respective buttons to show the content described in *Transition details in the history explorer*.

8. To return to the list of reports, click the *Reports* button.

11.3.4   Retrieving history information with crmsh

Investigating the cluster history is a complex task. To simplify this task, crmsh contains the `history` command with its subcommands. It is assumed SSH is configured correctly.

Each cluster moves states, migrates resources, or starts important processes. All these actions can be retrieved by subcommands of `history`.

By default, all `history` commands look at the events of the last hour. To change this time frame, use the `limit` subcommand. The syntax is:

```
# crm history
crm(live)history# limit FROM_TIME [TO_TIME]
```

Some valid examples include:

`limit 4:00pm`,

`limit 16:00`

>   Both commands mean the same, today at 4pm.

`limit 2012/01/12 6pm`

>   January 12th 2012 at 6pm

`limit "Sun 5 20:46"`

>   In the current year of the current month at Sunday the 5th at 8:46pm

Find more examples and how to create time frames at https://labix.org/python-dateutil ↗.

The `info` subcommand shows all the parameters which are covered by the `crm report`:

```
crm(live)history# info
Source: live
Period: 2012-01-12 14:10:56 - end
Nodes: alice
Groups:
Resources:
```

To limit `crm report` to certain parameters view the available options with the subcommand `help`.

To narrow down the level of detail, use the subcommand `detail` with a level:

```
crm(live)history# detail 1
```

The higher the number, the more detailed your report will be. Default is `0` (zero).

After you have set above parameters, use `log` to show the log messages.

To display the last transition, use the following command:

```
crm(live)history# transition -1
INFO: fetching new logs, please wait ...
```

This command fetches the logs and runs `dotty` (from the `graphviz` package) to show the transition graph. The shell opens the log file which you can browse with the ↓ and ↑ cursor keys.

If you do not want to open the transition graph, use the `nograph` option:

```
crm(live)history# transition -1 nograph
```

11.4 Monitoring system health with the `SysInfo` resource agent

To prevent a node from running out of disk space and thus being unable to manage any resources that have been assigned to it, SUSE Linux Enterprise High Availability provides a resource agent, `ocf:pacemaker:SysInfo`. Use it to monitor a node's health with regard to disk partitions. The SysInfo RA creates a node attribute named `#health_disk` which will be set to `red` if any of the monitored disks' free space is below a specified limit.

To define how the CRM should react in case a node's health reaches a critical state, use the global cluster option `node-health-strategy`.

PROCEDURE 11.4: CONFIGURING SYSTEM HEALTH MONITORING

To automatically move resources away from a node in case the node runs out of disk space, proceed as follows:

1. Configure an `ocf:pacemaker:SysInfo` resource:

   ```
   primitive sysinfo ocf:pacemaker:SysInfo \
       params disks="/tmp /var" ❶ min_disk_free="100M" ❷ disk_unit="M" ❸ \
       op monitor interval="15s"
   ```

   ❶ Which disk partitions to monitor. For example, `/tmp`, `/usr`, `/var`, and `/dev`. To specify multiple partitions as attribute values, separate them with a blank.

   > ### Note: / file system always monitored
   > You do not need to specify the root partition (`/`) in `disks`. It is always monitored by default.

   ❷ The minimum free disk space required for those partitions. Optionally, you can specify the unit to use for measurement (in the example above, `M` for megabytes is used). If not specified, `min_disk_free` defaults to the unit defined in the `disk_unit` parameter.

   ❸ The unit in which to report the disk space.

2. To complete the resource configuration, create a clone of `ocf:pacemaker:SysInfo` and start it on each cluster node.

3. Set the `node-health-strategy` to `migrate-on-red`:

```
property node-health-strategy="migrate-on-red"
```

In case of a `#health_disk` attribute set to `red`, the `pacemaker-schedulerd` adds `-INF` to the resources' score for that node. This causes any resources to move away from this node. The STONITH resource is the last one to be stopped, but even if the STONITH resource is not running anymore, the node can still be fenced. Fencing has direct access to the CIB and will continue to work.

After a node's health status has turned to `red`, solve the issue that led to the problem. Then clear the `red` status to make the node eligible again for running resources. Log in to the cluster node and use one of the following methods:

- Execute the following command:

```
# crm node status-attr NODE delete #health_disk
```

- Restart the cluster services on that node.

- Reboot the node.

The node is returned to service and can run resources again.

12 Fencing and STONITH

Fencing is an important concept in computer clusters for HA (High Availability). A cluster sometimes detects that one of the nodes is behaving strangely and needs to remove it. This is called *fencing* and is commonly done with a STONITH resource. Fencing may be defined as a method to bring an HA cluster to a known state.

Every resource in a cluster has a state attached. For example: "resource r1 is started on alice". In an HA cluster, such a state implies that "resource r1 is stopped on all nodes except alice", because the cluster must make sure that every resource may be started on only one node. Every node must report every change that happens to a resource. The cluster state is thus a collection of resource states and node states.

When the state of a node or resource cannot be established with certainty, fencing comes in. Even when the cluster is not aware of what is happening on a given node, fencing can ensure that the node does not run any important resources.

12.1 Classes of fencing

There are two classes of fencing: resource level and node level fencing. The latter is the primary subject of this chapter.

Resource level fencing

Resource level fencing ensures exclusive access to a given resource. Common examples of this are changing the zoning of the node from a SAN fiber channel switch (thus locking the node out of access to its disks) or methods like SCSI reserve. For examples, refer to *Section 13.10, "Additional mechanisms for storage protection"*.

Node level fencing

Node level fencing prevents a failed node from accessing shared resources entirely. This is usually done in a simple and abrupt way: reset or power off the node.

TABLE 12.1: CLASSES OF FENCING

Classes	Methods	Options	Examples
Node fencing[a] (reboot or shutdown)	Remote management	In-node	ILO, DRAC, IPMI

Classes	Methods	Options	Examples
I/O fencing (locking or reservation)		External	HMC, vCenter, EC2
	SBD and watchdog	Disk-based	hpwdt, iTCO_wdt, ipmi_wdt, softdog
		Diskless	
	Pure locking	In-cluster	SFEX
		External	SCSI2 reservation, SCSI3 reservation
	Built-in locking	Cluster-based	Cluster MD, LVM with lvmlockd and DLM
		Cluster-handled	MD-RAID

a Mandatory in High Availability clusters

12.2   Node level fencing

In a Pacemaker cluster, the implementation of node level fencing is STONITH (Shoot The Other Node in the Head). SUSE Linux Enterprise High Availability includes the `stonith` command line tool, an extensible interface for remotely powering down a node in the cluster. For an overview of the available options, run `stonith --help` or refer to the man page of `stonith` for more information.

12.2.1   STONITH devices

To use node level fencing, you first need to have a fencing device. To get a list of STONITH devices which are supported by SUSE Linux Enterprise High Availability, run one of the following commands on any of the nodes:

```
# stonith -L
```

or

```
# crm ra list stonith
```

STONITH devices may be classified into the following categories:

Power Distribution Units (PDU)

Power Distribution Units are an essential element in managing power capacity and functionality for critical network, server, and data center equipment. They can provide remote load monitoring of connected equipment and individual outlet power control for remote power recycling.

Uninterruptible Power Supplies (UPS)

A stable power supply provides emergency power to connected equipment by supplying power from a separate source if a utility power failure occurs.

Blade power control devices

If you are running a cluster on a set of blades, then the power control device in the blade enclosure is the only candidate for fencing. This device must be capable of managing single blade computers.

Lights-out devices

Lights-out devices (IBM RSA, HP iLO, Dell DRAC) are becoming increasingly popular and may even become standard in off-the-shelf computers. However, they are inferior to UPS devices, because they share a power supply with their host (a cluster node). If a node stays without power, the device supposed to control it would be useless. In that case, the CRM would continue its attempts to fence the node indefinitely while all other resource operations would wait for the fencing/STONITH operation to complete.

Testing devices

Testing devices are used exclusively for testing purposes. They are usually more gentle on the hardware. Before the cluster goes into production, they must be replaced with real fencing devices.

The choice of the STONITH device depends mainly on your budget and the kind of hardware you use.

12.2.2   STONITH implementation

The STONITH implementation of SUSE® Linux Enterprise High Availability consists of two components:

pacemaker-fenced

> `pacemaker-fenced` is a daemon which can be accessed by local processes or over the network. It accepts the commands which correspond to fencing operations: reset, power-off, and power-on. It can also check the status of the fencing device.
>
> The `pacemaker-fenced` daemon runs on every node in the High Availability cluster. The `pacemaker-fenced` instance running on the DC node receives a fencing request from the `pacemaker-controld`. It is up to this and other `pacemaker-fenced` programs to carry out the desired fencing operation.

STONITH plug-ins

> For every supported fencing device there is a STONITH plug-in which is capable of controlling said device. A STONITH plug-in is the interface to the fencing device. The STONITH plug-ins contained in the `cluster-glue` package reside in `/usr/lib64/stonith/plugins` on each node. (If you installed the `fence-agents` package, too, the plug-ins contained there are installed in `/usr/sbin/fence_*`.) All STONITH plug-ins look the same to `pacemaker-fenced`, but are different on the other side, reflecting the nature of the fencing device.
>
> Some plug-ins support more than one device. A typical example is `ipmilan` (or `external/ipmi`) which implements the IPMI protocol and can control any device which supports this protocol.

12.3   STONITH resources and configuration

To set up fencing, you need to configure one or more STONITH resources—the `pacemaker-fenced` daemon requires no configuration. All configuration is stored in the CIB. A STONITH resource is a resource of class `stonith` (see *Section 6.2, "Supported resource agent classes"*). STONITH resources are a representation of STONITH plug-ins in the CIB. Apart from the fencing operations, the STONITH resources can be started, stopped and monitored, like any other resource. Starting or stopping STONITH resources means loading and unloading the STONITH device driver on a node. Starting and stopping are thus only administrative operations and do not translate to any operation on the fencing device itself. However, monitoring does translate

to logging in to the device (to verify that the device will work in case it is needed). When a STONITH resource fails over to another node it enables the current node to talk to the STONITH device by loading the respective driver.

STONITH resources can be configured like any other resource. For details how to do so with your preferred cluster management tool:

- Hawk2: *Section 6.9.1, "Creating STONITH resources with Hawk2"*

- crmsh: *Section 6.9.2, "Creating STONITH resources with crmsh"*

The list of parameters (attributes) depends on the respective STONITH type. To view a list of parameters for a specific device, use the `stonith` command:

```
# stonith -t stonith-device-type -n
```

For example, to view the parameters for the `ibmhmc` device type, enter the following:

```
# stonith -t ibmhmc -n
```

To get a short help text for the device, use the `-h` option:

```
# stonith -t stonith-device-type -h
```

12.3.1   Example STONITH resource configurations

The following are example configurations written in the syntax of the `crm` command line tool. To apply them, put the sample in a text file (for example, `sample.txt`) and run:

```
# crm < sample.txt
```

For more information about configuring resources with the `crm` command line tool, refer to *Section 5.5, "Introduction to crmsh"*.

EXAMPLE 12.1: CONFIGURATION OF AN IBM RSA LIGHTS-OUT DEVICE

An IBM RSA lights-out device might be configured like this:

```
# crm configure
crm(live)configure# primitive st-ibmrsa-1 stonith:external/ibmrsa-telnet \
params nodename=alice ip_address=192.168.0.101 \
username=USERNAME password=PASSW0RD
crm(live)configure# primitive st-ibmrsa-2 stonith:external/ibmrsa-telnet \
params nodename=bob ip_address=192.168.0.102 \
```

```
username=USERNAME password=PASSW0RD
crm(live)configure# location l-st-alice st-ibmrsa-1 -inf: alice
crm(live)configure# location l-st-bob st-ibmrsa-2 -inf: bob
crm(live)configure# commit
```

In this example, location constraints are used for the following reason: there is always a certain probability that the STONITH operation is going to fail. Therefore, a STONITH operation on the node which is the executioner as well is not reliable. If the node is reset, it cannot send the notification about the fencing operation outcome. The only way to do that is to assume that the operation is going to succeed and send the notification beforehand. But if the operation fails, problems could arise. Therefore, by convention, pacemaker-fenced refuses to terminate its host.

EXAMPLE 12.2: CONFIGURATION OF A UPS FENCING DEVICE

The configuration of a UPS type fencing device is similar to the examples above. The details are not covered here. All UPS devices employ the same mechanics for fencing. How the device is accessed varies. Old UPS devices only had a serial port, usually connected at 1200baud using a special serial cable. Many new ones still have a serial port, but often they also use a USB or Ethernet interface. The kind of connection you can use depends on what the plug-in supports.

For example, compare the apcmaster with the apcsmart device by using the **stonith -t** *stonith-device-type* **-n** command:

```
# stonith -t apcmaster -h
```

returns the following information:

```
STONITH Device: apcmaster - APC MasterSwitch (via telnet)
NOTE: The APC MasterSwitch accepts only one (telnet)
connection/session a time. When one session is active,
subsequent attempts to connect to the MasterSwitch will fail.
For more information see http://www.apc.com/
List of valid parameter names for apcmaster STONITH device:
        ipaddr
        login
        password
For Config info [-p] syntax, give each of the above parameters in order as
the -p value.
Arguments are separated by white space.
Config file [-F] syntax is the same as -p, except # at the start of a line
denotes a comment
```

With

```
# stonith -t apcsmart -h
```

you get the following output:

```
STONITH Device: apcsmart - APC Smart UPS
(via serial port - NOT USB!).
Works with higher-end APC UPSes, like
Back-UPS Pro, Smart-UPS, Matrix-UPS, etc.
(Smart-UPS may have to be >= Smart-UPS 700?).
See http://www.networkupstools.org/protocols/apcsmart.html
for protocol compatibility details.
For more information see http://www.apc.com/
List of valid parameter names for apcsmart STONITH device:
ttydev
hostlist
```

The first plug-in supports APC UPS with a network port and telnet protocol. The second plug-in uses the APC SMART protocol over the serial line, which is supported by many APC UPS product lines.

EXAMPLE 12.3: CONFIGURATION OF A KDUMP DEVICE

Kdump belongs to the *Special fencing devices* and is in fact the opposite of a fencing device. The plug-in checks if a Kernel dump is in progress on a node. If so, it returns true and acts *as if* the node has been fenced, because the node will reboot after the Kdump is complete. If not, it returns a failure and the next fencing device is triggered.

The Kdump plug-in must be used together with another, real STONITH device, for example, `external/ipmi`. It does *not* work with SBD as the STONITH device. For the fencing mechanism to work properly, you must specify the order of the fencing devices so that Kdump is checked before a real STONITH device is triggered, as shown in the following procedure.

1. Use the `stonith:fence_kdump` fence agent. A configuration example is shown below. For more information, see **crm ra info stonith:fence_kdump**.

```
# crm configure
crm(live)configure# primitive st-kdump stonith:fence_kdump \
    params nodename="alice "\ ❶
    pcmk_host_list="alice" \
    pcmk_host_check="static-list" \
    pcmk_reboot_action="off" \
    pcmk_monitor_action="metadata" \
    pcmk_reboot_retries="1" \
    timeout="60" ❷
```

```
crm(live)configure# commit
```

❶ Name of the node to listen for a message from `fence_kdump_send`. Configure more STONITH resources for other nodes if needed.

❷ Defines how long to wait for a message from `fence_kdump_send`. If a message is received, then a Kdump is in progress and the fencing mechanism considers the node to be fenced. If no message is received, `fence_kdump` times out, which indicates that the fence operation failed. The next STONITH device in the `fencing_topology` eventually fences the node.

2. On each node, configure `fence_kdump_send` to send a message to all nodes when the Kdump process is finished. In `/etc/sysconfig/kdump`, edit the `KDUMP_POSTSCRIPT` line. For example:

```
KDUMP_POSTSCRIPT="/usr/lib/fence_kdump_send -i 10 -p 7410 -c 1 NODELIST"
```

Replace *NODELIST* with the host names of all the cluster nodes.

3. Run either **systemctl restart kdump.service** or **mkdumprd**. Either of these commands will detect that `/etc/sysconfig/kdump` was modified, and will regenerate the `initrd` to include the library `fence_kdump_send` with network enabled.

4. Open a port in the firewall for the `fence_kdump` resource. The default port is `7410`.

5. To have Kdump checked before triggering a real fencing mechanism (like `external/ipmi`), use a configuration similar to the following:

```
crm(live)configure# fencing_topology \
  alice: kdump-node1 ipmi-node1 \
  bob: kdump-node2 ipmi-node2
crm(live)configure# commit
```

For more details on `fencing_topology`:

```
crm(live)configure# help fencing_topology
```

12.4 Monitoring fencing devices

Like any other resource, the STONITH class agents also support the monitoring operation for checking status.

Fencing devices are an indispensable part of an HA cluster, but the less you need to use them, the better. Power management equipment is often affected by too much broadcast traffic. Some devices cannot handle more than ten or so connections per minute. Some get confused if two clients try to connect at the same time. Most cannot handle more than one session at a time.

The probability that a fencing operation needs to be performed and the fencing device fails is low. For most devices, a monitoring interval of at least 1800 seconds (30 minutes) should suffice. The exact value depends on the device and infrastructure. STONITH SBD resources do not need a monitor at all. See *Section 12.5, "Special fencing devices"* and *Chapter 13, Storage protection and SBD*.

For detailed information on how to configure monitor operations, refer to *Section 6.10.2, "Configuring resource monitoring with crmsh"* for the command line approach.

12.5 Special fencing devices

In addition to plug-ins which handle real STONITH devices, there are special purpose STONITH plug-ins.

🖐 Warning: For testing only

Some STONITH plug-ins mentioned below are for demonstration and testing purposes only. Do not use any of the following devices in real-life scenarios because this may lead to data corruption and unpredictable results:

- `external/ssh`

- `ssh`

`fence_kdump`

This plug-in checks if a Kernel dump is in progress on a node. If so, it returns `true`, and acts as if the node has been fenced. The node cannot run any resources during the dump anyway. This avoids fencing a node that is already down but doing a dump, which takes some time. The plug-in must be used in concert with another, real STONITH device.

For configuration details, see *Example 12.3, "Configuration of a Kdump device"*.

`external/sbd`

This is a self-fencing device. It reacts to a so-called "poison pill" which can be inserted into a shared disk. On shared-storage connection loss, it stops the node from operating. Learn how to use this STONITH agent to implement storage-based fencing in *Chapter 13*, *Procedure 13.7, "Configuring the cluster to use SBD"*.

> **❗ Important: `external/sbd` and DRBD**
>
> The `external/sbd` fencing mechanism requires that the SBD partition is readable directly from each node. Thus, a DRBD* device must not be used for an SBD partition.
>
> However, you can use the fencing mechanism for a DRBD cluster, provided the SBD partition is located on a shared disk that is not mirrored or replicated.

`external/ssh`

Another software-based "fencing" mechanism. The nodes must be able to log in to each other as `root` without passwords. It takes a single parameter, `hostlist`, specifying the nodes that it will target. As it is not able to reset a truly failed node, it must not be used for real-life clusters—for testing and demonstration purposes only. Using it for shared storage would result in data corruption.

`meatware`

`meatware` requires help from the user to operate. Whenever invoked, `meatware` logs a CRIT severity message which shows up on the node's console. The operator then confirms that the node is down and issues a **`meatclient(8)`** command. This tells `meatware` to inform the cluster that the node should be considered dead. See `/usr/share/doc/packages/cluster-glue/README.meatware` for more information.

`suicide`

This is a software-only device, which can reboot a node it is running on, using the **`reboot`** command. This requires action by the node's operating system and can fail under certain circumstances. Therefore avoid using this device whenever possible. However, it is safe to use on one-node clusters.

Diskless SBD

This configuration is useful if you want a fencing mechanism without shared storage. In this diskless mode, SBD fences nodes by using the hardware watchdog without relying on any shared device. However, diskless SBD cannot handle a split-brain scenario for a two-node cluster. Use this option only for clusters with *more than two* nodes.

`suicide` is the only exception to the "I do not shoot my host" rule.

12.6   Basic recommendations

Check the following list of recommendations to avoid common mistakes:

- Do not configure several power switches in parallel.

- To test your STONITH devices and their configuration, pull the plug once from each node and verify that fencing the node does takes place.

- Test your resources under load and verify the timeout values are appropriate. Setting timeout values too low can trigger (unnecessary) fencing operations. For details, refer to *Section 6.3, "Timeout values"*.

- Use appropriate fencing devices for your setup. For details, also refer to *Section 12.5, "Special fencing devices"*.

- Configure one or more STONITH resources. By default, the global cluster option `stonith-enabled` is set to `true`. If no STONITH resources have been defined, the cluster refuses to start any resources.

- Do not set the global cluster option `stonith-enabled` to `false` for the following reasons:

    - Clusters without STONITH enabled are not supported.

    - DLM/OCFS2 will block forever waiting for a fencing operation that will never happen.

- Do not set the global cluster option `startup-fencing` to `false`. By default, it is set to `true` for the following reason: if a node is in an unknown state during cluster start-up, the node will be fenced once to clarify its status.

12.7 For more information

`/usr/share/doc/packages/cluster-glue`

In your installed system, this directory contains README files for many STONITH plug-ins and devices.

https://www.clusterlabs.org/pacemaker/doc/ ↗

Pacemaker Explained: Explains the concepts used to configure Pacemaker. Contains comprehensive and detailed information for reference.

https://techthoughts.typepad.com/managing_computers/2007/10/split-brain-quo.html ↗

Article explaining the concepts of split brain, quorum, and fencing in HA clusters.

13 Storage protection and SBD

SBD (STONITH Block Device) provides a node fencing mechanism for Pacemaker-based clusters through the exchange of messages via shared block storage (SAN, iSCSI, FCoE, etc.). This isolates the fencing mechanism from changes in firmware version or dependencies on specific firmware controllers. SBD needs a watchdog on each node to ensure that misbehaving nodes are really stopped. Under certain conditions, it is also possible to use SBD without shared storage, by running it in diskless mode.

The cluster bootstrap scripts provide an automated way to set up a cluster with the option of using SBD as fencing mechanism. For details, see the *Article "Installation and Setup Quick Start"*. However, manually setting up SBD provides you with more options regarding the individual settings.

This chapter explains the concepts behind SBD. It guides you through configuring the components needed by SBD to protect your cluster from potential data corruption in case of a split-brain scenario.

In addition to node level fencing, you can use additional mechanisms for storage protection, such as LVM exclusive activation or OCFS2 file locking support (resource level fencing). They protect your system against administrative or application faults.

13.1 Conceptual overview

SBD expands to *Storage-Based Death* or *STONITH Block Device*.

The highest priority of the High Availability cluster stack is to protect the integrity of data. This is achieved by preventing uncoordinated concurrent access to data storage. The cluster stack takes care of this using several control mechanisms.

However, network partitioning or software malfunction could potentially cause scenarios where several DCs are elected in a cluster. This split-brain scenario can cause data corruption.

Node fencing via STONITH is the primary mechanism to prevent split brain. Using SBD as a node fencing mechanism is one way of shutting down nodes without using an external power off device in case of a split-brain scenario.

SBD partition

In an environment where all nodes have access to shared storage, a small partition of the device is formatted for use with SBD. The size of the partition depends on the block size of the used disk (for example, 1 MB for standard SCSI disks with 512 byte block size or 4 MB for DASD disks with 4 kB block size). The initialization process creates a message layout on the device with slots for up to 255 nodes.

SBD daemon

After the respective SBD daemon is configured, it is brought online on each node before the rest of the cluster stack is started. It is terminated after all other cluster components have been shut down, thus ensuring that cluster resources are never activated without SBD supervision.

Messages

The daemon automatically allocates one of the message slots on the partition to itself, and constantly monitors it for messages addressed to itself. Upon receipt of a message, the daemon immediately complies with the request, such as initiating a power-off or reboot cycle for fencing.

Also, the daemon constantly monitors connectivity to the storage device, and terminates itself if the partition becomes unreachable. This guarantees that it is not disconnected from fencing messages. If the cluster data resides on the same logical unit in a different partition, this is not an additional point of failure; the workload terminates anyway if the storage connectivity is lost.

Watchdog

Whenever SBD is used, a correctly working watchdog is crucial. Modern systems support a *hardware watchdog* that needs to be "tickled" or "fed" by a software component. The software component (in this case, the SBD daemon) "feeds" the watchdog by regularly writing a service pulse to the watchdog. If the daemon stops feeding the watchdog, the hardware enforces a system restart. This protects against failures of the SBD process itself, such as dying, or becoming stuck on an I/O error.

If Pacemaker integration is activated, loss of device majority alone does not trigger self-fencing. For example, your cluster contains three nodes: A, B, and C. Because of a network split, A can only see itself while B and C can still communicate. In this case, there are two cluster partitions:

one with quorum because of being the majority (B, C), and one without (A). If this happens while the majority of fencing devices are unreachable, node A would self-fence, but nodes B and C would continue to run.

13.2 Overview of manually setting up SBD

The following steps are necessary to manually set up storage-based protection. They must be executed as `root`. Before you start, check *Section 13.3, "Requirements and restrictions"*.

1. *Setting up the watchdog*

2. Depending on your scenario, either use SBD with one to three devices or in diskless mode. For an outline, see *Section 13.4, "Number of SBD devices"*. The detailed setup is described in:

    - *Setting up SBD with devices*

    - *Setting up diskless SBD*

3. *Testing SBD and fencing*

13.3 Requirements and restrictions

- You can use up to three SBD devices for storage-based fencing. When using one to three devices, the shared storage must be accessible from all nodes.

- The path to the shared storage device must be persistent and consistent across all nodes in the cluster. Use stable device names such as `/dev/disk/by-id/dm-uuid-part1-mpath-abcedf12345`.

- The shared storage can be connected via Fibre Channel (FC), Fibre Channel over Ethernet (FCoE), or even iSCSI.

- The shared storage segment *must not* use host-based RAID, LVM, or DRBD*. DRBD can be split, which is problematic for SBD, as there cannot be two states in SBD. Cluster multi-device (Cluster MD) cannot be used for SBD.

- However, using storage-based RAID and multipathing is recommended for increased reliability.

- An SBD device can be shared between different clusters, as long as no more than 255 nodes share the device.

- Fencing does not work with an asymmetric SBD setup. When using more than one SBD device, all nodes must have a slot in all SBD devices.

- When using more than one SBD device, all devices must have the same configuration, for example, the same timeout values.

- For clusters with more than two nodes, you can also use SBD in *diskless* mode.

13.4 Number of SBD devices

SBD supports the use of up to three devices:

One device

The most simple implementation. It is appropriate for clusters where all of your data is on the same shared storage.

Two devices

This configuration is primarily useful for environments that use host-based mirroring, but where no third storage device is available. SBD will not terminate itself if it loses access to one mirror leg, allowing the cluster to continue. However, since SBD does not have enough knowledge to detect an asymmetric split of the storage, it will not fence the other side while only one mirror leg is available. Thus, it cannot automatically tolerate a second failure while one of the storage arrays is down.

Three devices

The most reliable configuration. It is resilient against outages of one device, be it because of failures or maintenance. SBD will terminate itself only if more than one device is lost and if required, depending on the status of the cluster partition or node. If at least two devices are still accessible, fencing messages can be successfully transmitted.

This configuration is suitable for more complex scenarios where storage is not restricted to a single array. Host-based mirroring solutions can have one SBD per mirror leg (not mirrored itself), and an additional tie-breaker on iSCSI.

Diskless

This configuration is useful if you want a fencing mechanism without shared storage. In this diskless mode, SBD fences nodes by using the hardware watchdog without relying on any shared device. However, diskless SBD cannot handle a split-brain scenario for a two-node cluster. Use this option only for clusters with *more than two* nodes.

13.5 Calculation of timeouts

When using SBD as a fencing mechanism, it is vital to consider the timeouts of all components, because they depend on each other. When using more than one SBD device, all devices must have the same timeout values.

`watchdog` **timeout**

This timeout is set during initialization of the SBD device. It depends mostly on your storage latency. The majority of devices must be successfully read within this time. Otherwise, the node might self-fence.

> ### Note: Multipath or iSCSI setup
>
> If your SBD devices reside on a multipath setup or iSCSI, the timeout should be set to the time required to detect a path failure and switch to the next path.
>
> This also means that in `/etc/multipath.conf` the value of `max_polling_interval` must be less than the `watchdog` timeout.

`msgwait` **timeout**

This timeout is set during initialization of the SBD device. It defines the time after which a message written to a node's slot on the SBD device is considered delivered. The timeout should be long enough for the node to detect that it needs to self-fence.

However, if the `msgwait` timeout is long, a fenced cluster node might rejoin before the fencing action returns. This can be mitigated by setting the `SBD_DELAY_START` parameter in the SBD configuration, as described in *Procedure 13.4* in *Step 3*.

`stonith-timeout` **in the CIB**

This timeout is set in the CIB as a global cluster property. It defines how long to wait for the STONITH action (reboot, on, off) to complete.

`stonith-watchdog-timeout` **in the CIB**

> This timeout is set in the CIB as a global cluster property. If not set explicitly, it defaults to `0`, which is appropriate for using SBD with one to three devices. For SBD in diskless mode, this timeout must *not* be `0`. For details, see *Procedure 13.8, "Configuring diskless SBD"*.

If you change the watchdog timeout, you need to adjust the other two timeouts as well. The following "formula" expresses the relationship between these three values:

EXAMPLE 13.1: FORMULA FOR TIMEOUT CALCULATION

```
Timeout (msgwait) >= (Timeout (watchdog) * 2)
stonith-timeout >= Timeout (msgwait) + 20%
```

For example, if you set the watchdog timeout to `120`, set the `msgwait` timeout to at least `240` and the `stonith-timeout` to at least `288`.

If you use the bootstrap scripts provided by the crm shell to set up a cluster and to initialize the SBD device, the relationship between these timeouts is automatically considered.

13.6 Setting up the watchdog

SUSE Linux Enterprise High Availability ships with several kernel modules that provide hardware-specific watchdog drivers. For clusters in production environments, we recommend using a hardware-specific watchdog driver. However, if no watchdog matches your hardware, `softdog` can be used as kernel watchdog module.

SUSE Linux Enterprise High Availability uses the SBD daemon as the software component that "feeds" the watchdog.

13.6.1 Using a hardware watchdog

Finding the right watchdog kernel module for a given system is not trivial. Automatic probing fails often. As a result, many modules are already loaded before the right one gets a chance.

The following table lists some commonly used watchdog drivers. However, this is not a complete list of supported drivers. If your hardware is not listed here, you can also find a list of choices in the directories `/lib/modules/KERNEL_VERSION/kernel/drivers/watchdog` and `/lib/modules/KERNEL_VERSION/kernel/drivers/ipmi`. Alternatively, ask your hardware or system vendor for details on system-specific watchdog configuration.

Hardware	Driver
HP	`hpwdt`
Dell, Lenovo (Intel TCO)	`iTCO_wdt`
Fujitsu	`ipmi_watchdog`
LPAR on IBM Power	`pseries-wdt`
VM on IBM z/VM	`vmwatchdog`
Xen VM (DomU)	`xen_xdt`
VM on VMware vSphere	`wdat_wdt`
Generic	`softdog`

Important: Accessing the watchdog timer

Some hardware vendors ship systems management software that uses the watchdog for system resets (for example, HP ASR daemon). If the watchdog is used by SBD, disable such software. No other software must access the watchdog timer.

PROCEDURE 13.1: LOADING THE CORRECT KERNEL MODULE

To make sure the correct watchdog module is loaded, proceed as follows:

1. List the drivers that have been installed with your kernel version:

   ```
   # rpm -ql kernel-VERSION | grep watchdog
   ```

2. List any watchdog modules that are currently loaded in the kernel:

   ```
   # lsmod | egrep "(wd|dog)"
   ```

3. If you get a result, unload the wrong module:

   ```
   # rmmod WRONG_MODULE
   ```

4. Enable the watchdog module that matches your hardware:

   ```
   # echo WATCHDOG_MODULE > /etc/modules-load.d/watchdog.conf
   ```

```
# systemctl restart systemd-modules-load
```

5. Test whether the watchdog module is loaded correctly:

```
# lsmod | grep dog
```

6. Verify if the watchdog device is available and works:

```
# ls -l /dev/watchdog*
# sbd query-watchdog
```

If your watchdog device is not available, stop here and check the module name and options. Maybe use another driver.

7. Verify if the watchdog device works:

```
# sbd -w WATCHDOG_DEVICE test-watchdog
```

8. Reboot your machine to make sure there are no conflicting kernel modules. For example, if you find the message `cannot register ...` in your log, this would indicate such conflicting modules. To ignore such modules, refer to https://documentation.suse.com/sles/html/SLES-all/cha-mod.html#sec-mod-modprobe-blacklist ↗.

13.6.2   Using the software watchdog (softdog)

For clusters in production environments we recommend to use a hardware-specific watchdog driver. However, if no watchdog matches your hardware, `softdog` can be used as kernel watchdog module.

> ❗ **Important: Softdog limitations**
>
> The softdog driver assumes that at least one CPU is still running. If all CPUs are stuck, the code in the softdog driver that should reboot the system will never be executed. In contrast, hardware watchdogs keep working even if all CPUs are stuck.

PROCEDURE 13.2: LOADING THE SOFTDOG KERNEL MODULE

1. Enable the softdog watchdog:

```
# echo softdog > /etc/modules-load.d/watchdog.conf
# systemctl restart systemd-modules-load
```

2. Test whether the softdog watchdog module is loaded correctly:

```
# lsmod | grep softdog
```

13.7 Setting up SBD with devices

The following steps are necessary for setup:

1. *Initializing the SBD devices*

2. *Editing the SBD configuration file*

3. *Enabling and starting the SBD service*

4. *Testing the SBD devices*

5. *Configuring the cluster to use SBD*

Before you start, make sure the block device or devices you want to use for SBD meet the requirements specified in *Section 13.3*.

When setting up the SBD devices, you need to take several timeout values into account. For details, see *Section 13.5, "Calculation of timeouts"*.

The node terminates itself if the SBD daemon running on it has not updated the watchdog timer fast enough. After having set the timeouts, test them in your specific environment.

PROCEDURE 13.3: INITIALIZING THE SBD DEVICES

To use SBD with shared storage, you must first create the messaging layout on one to three block devices. The `sbd create` command writes a metadata header to the specified device or devices. It also initializes the messaging slots for up to 255 nodes. If executed without any further options, the command uses the default timeout settings.

> ✋ Warning: Overwriting existing data
>
> Make sure the device or devices you want to use for SBD do not hold any important data. When you execute the `sbd create` command, roughly the first megabyte of the specified block devices is overwritten without further requests or backup.

1. Decide which block device or block devices to use for SBD.

2. Initialize the SBD device with the following command:

```
# sbd -d /dev/disk/by-id/DEVICE_ID create
```

To use more than one device for SBD, specify the `-d` option multiple times, for example:

```
# sbd -d /dev/disk/by-id/DEVICE_ID1 -d /dev/disk/by-id/DEVICE_ID2 -d /dev/disk/by-id/DEVICE_ID3 create
```

3. If your SBD device resides on a multipath group, use the `-1` and `-4` options to adjust the timeouts to use for SBD. If you initialized more than one device, you must set the same timeout values for all devices. For details, see *Section 13.5, "Calculation of timeouts"*. All timeouts are given in seconds:

```
# sbd -d /dev/disk/by-id/DEVICE_ID -1 90❶ -4 180❷ create
```

❶ The `-1` option is used to specify the `watchdog` timeout. In the example above, it is set to `90` seconds. The minimum allowed value for the emulated watchdog is `15` seconds.

❷ The `-4` option is used to specify the `msgwait` timeout. In the example above, it is set to `180` seconds.

4. Check what has been written to the device:

```
# sbd -d /dev/disk/by-id/DEVICE_ID dump
Header version     : 2.1
UUID               : 619127f4-0e06-434c-84a0-ea82036e144c
Number of slots    : 255
Sector size        : 512
Timeout (watchdog) : 5
Timeout (allocate) : 2
Timeout (loop)     : 1
Timeout (msgwait)  : 10
==Header on disk /dev/disk/by-id/DEVICE_ID is dumped
```

As you can see, the timeouts are also stored in the header, to ensure that all participating nodes agree on them.

After you have initialized the SBD devices, edit the SBD configuration file, then enable and start the respective services for the changes to take effect.

PROCEDURE 13.4: EDITING THE SBD CONFIGURATION FILE

1. Open the file `/etc/sysconfig/sbd`.

2. Search for the following parameter: `SBD_DEVICE`.

   It specifies the devices to monitor and to use for exchanging SBD messages.

   Edit this line by replacing /dev/disk/by-id/`DEVICE_ID` with your SBD device:

   ```
   SBD_DEVICE="/dev/disk/by-id/DEVICE_ID"
   ```

   If you need to specify multiple devices in the first line, separate them with semicolons (the order of the devices does not matter):

   ```
   SBD_DEVICE="/dev/disk/by-id/DEVICE_ID1;/dev/disk/by-id/DEVICE_ID2;/dev/disk/by-id/DEVICE_ID3"
   ```

   If the SBD device is not accessible, the daemon fails to start and inhibits cluster start-up.

3. Search for the following parameter: `SBD_DELAY_START`.

   Enables or disables a delay. Set `SBD_DELAY_START` to `yes` if `msgwait` is long, but your cluster nodes boot quickly. Setting this parameter to `yes` delays the start of SBD on boot. This is sometimes necessary with virtual machines.

   The default delay length is the same as the `msgwait` timeout value. Alternatively, you can specify an integer, in seconds, instead of `yes`.

   If you enable `SBD_DELAY_START`, you must also check the SBD service file to ensure that the value of `TimeoutStartSec` is greater than the value of `SBD_DELAY_START`. For more information, see https://www.suse.com/support/kb/doc/?id=000019356 ↗.

4. Copy the configuration file to all nodes by using **csync2**:

   ```
   # csync2 -xv
   ```

   For more information, see *Section 4.7, "Transferring the configuration to all nodes"*.

After you have added your SBD devices to the SBD configuration file, enable the SBD daemon. The SBD daemon is a critical piece of the cluster stack. It needs to be running when the cluster stack is running. Thus, the `sbd` service is started as a dependency whenever the cluster services are started.

PROCEDURE 13.5: ENABLING AND STARTING THE SBD SERVICE

1. On each node, enable the SBD service:

   ```
   # systemctl enable sbd
   ```

   SBD starts together with the Corosync service whenever the cluster services are started.

2. Restart the cluster services on all nodes at once by using the `--all` option:

```
# crm cluster restart --all
```

This automatically triggers the start of the SBD daemon.

> ❗ **Important: Restart cluster services for SBD changes**
>
> If any SBD metadata changes, you must restart the cluster services again. To keep critical cluster resources running during the restart, consider putting the cluster into maintenance mode first. For more information, see *Chapter 28, Executing maintenance tasks*.

As a next step, test the SBD devices as described in *Procedure 13.6*.

PROCEDURE 13.6: TESTING THE SBD DEVICES

1. The following command dumps the node slots and their current messages from the SBD device:

```
# sbd -d /dev/disk/by-id/DEVICE_ID list
```

Now you should see all cluster nodes that have ever been started with SBD listed here. For example, if you have a two-node cluster, the message slot should show `clear` for both nodes:

```
0       alice       clear
1       bob         clear
```

2. Try sending a test message to one of the nodes:

```
# sbd -d /dev/disk/by-id/DEVICE_ID message alice test
```

3. The node acknowledges the receipt of the message in the system log files:

```
May 03 16:08:31 alice sbd[66139]: /dev/disk/by-id/DEVICE_ID: notice: servant:
Received command test from bob on disk /dev/disk/by-id/DEVICE_ID
```

This confirms that SBD is indeed up and running on the node and that it is ready to receive messages.

As a final step, you need to adjust the cluster configuration as described in *Procedure 13.7*.

1. Start a shell and log in as `root` or equivalent.

2. Run **`crm configure`**.

3. Enter the following:

   ```
   crm(live)configure# property stonith-enabled="true" ❶
   crm(live)configure# property stonith-watchdog-timeout=0 ❷
   crm(live)configure# property stonith-timeout="40s" ❸
   ```

   ❶ This is the default configuration, because clusters without STONITH are not support-
       ed. But in case STONITH has been deactivated for testing purposes, make sure this
       parameter is set to `true` again.

   ❷ If not explicitly set, this value defaults to `0`, which is appropriate for use of SBD with
       one to three devices.

   ❸ To calculate the `stonith-timeout`, refer to *Section 13.5, "Calculation of timeouts"*. A
       `stonith-timeout` value of `40` would be appropriate if the `msgwait` timeout value
       for SBD was set to `30` seconds.

4. Configure the SBD STONITH resource. You do not need to clone this resource.

   For a two-node cluster, in case of split brain, fencing is issued from each node to the other
   as expected. To prevent both nodes from being reset at practically the same time, it is
   recommended to apply the following fencing delays to help one of the nodes, or even the
   preferred node, win the fencing match. For clusters with more than two nodes, you do
   not need to apply these delays.

   Priority fencing delay

   The `priority-fencing-delay` cluster property is disabled by default. By configur-
   ing a delay value, if the other node is lost and it has the higher total resource priority,
   the fencing targeting it is delayed for the specified amount of time. This means that
   in case of split-brain, the more important node wins the fencing match.

   Resources that matter can be configured with priority meta attribute. On calculation,
   the priority values of the resources or instances that are running on each node are
   summed up to be accounted. A promoted resource instance takes the configured base
   priority plus one so that it receives a higher value than any unpromoted instance.

   ```
   # crm configure property priority-fencing-delay=30
   ```

Even if `priority-fencing-delay` is used, we still recommend also using `pcmk_delay_base` or `pcmk_delay_max` as described below to address any situations where the nodes happen to have equal priority. The value of `priority-fencing-delay` should be significantly greater than the maximum of `pcmk_delay_base` / `pcmk_delay_max`, and preferably twice the maximum.

Predictable static delay

This parameter adds a static delay before executing STONITH actions. To prevent the nodes from being reset at the same time under split-brain of a two-node cluster, configure separate fencing resources with different delay values. The preferred node can be marked with the parameter to be targeted with a longer fencing delay so that it wins any fencing match. To make this succeed, each node must have two primitive STONITH devices. In the following configuration, alice will win and survive in case of a split-brain scenario:

```
crm(live)configure# primitive st-sbd-alice stonith:external/sbd params \
pcmk_host_list=alice pcmk_delay_base=20
crm(live)configure# primitive st-sbd-bob stonith:external/sbd params \
pcmk_host_list=bob pcmk_delay_base=0
```

Dynamic random delay

This parameter adds a random delay for STONITH actions on the fencing device. Rather than a static delay targeting a specific node, the parameter `pcmk_delay_max` adds a random delay for any fencing with the fencing resource to prevent double reset. Unlike `pcmk_delay_base`, this parameter can be specified for a unified fencing resource targeting multiple nodes.

```
crm(live)configure# primitive stonith_sbd stonith:external/sbd \
params pcmk_delay_max=30
```

> ✋ Warning: `pcmk_delay_max` might not prevent double reset in a split-brain scenario
>
> The lower the value of `pcmk_delay_max`, the higher the chance that a double reset might still occur.
>
> If your aim is to have a predictable survivor, use a priority fencing delay or predictable static delay.

5. Review your changes with **show**.

6. Submit your changes with **commit** and leave the crm live configuration with **quit**.

After the resource starts, your cluster is successfully configured to use SBD if a node needs to be fenced.

13.8   Setting up diskless SBD

SBD can be operated in a diskless mode. In this mode, a watchdog device is used to reset the node in the following cases: if it loses quorum, if any monitored daemon is lost and not recovered, or if Pacemaker decides that the node requires fencing. Diskless SBD is based on "self-fencing" of a node, depending on the status of the cluster, the quorum and some reasonable assumptions. No STONITH SBD resource primitive is needed in the CIB.

> **❗ Important: Do not block Corosync traffic in the local firewall**
>
> Diskless SBD relies on reformed membership and loss of quorum to achieve fencing. Corosync traffic must be able to pass through all network interfaces, including the loopback interface, and must not be blocked by a local firewall. Otherwise, Corosync cannot reform a new membership, which can cause a split-brain scenario that cannot be handled by diskless SBD fencing.

> **❗ Important: Number of cluster nodes**
>
> Do *not* use diskless SBD as a fencing mechanism for two-node clusters. Use diskless SBD only for clusters with three or more nodes. SBD in diskless mode cannot handle split-brain scenarios for two-node clusters. If you want to use diskless SBD for two-node clusters, use QDevice as described in *Chapter 14, QDevice and QNetd*.

PROCEDURE 13.8: **CONFIGURING DISKLESS SBD**

1. Open the file `/etc/sysconfig/sbd` and use the following entries:

```
SBD_PACEMAKER=yes
SBD_STARTMODE=always
SBD_DELAY_START=no
```

```
SBD_WATCHDOG_DEV=/dev/watchdog
SBD_WATCHDOG_TIMEOUT=5
```

The `SBD_DEVICE` entry is not needed as no shared disk is used. When this parameter is missing, the `sbd` service does not start any watcher process for SBD devices.

If you need to delay the start of SBD on boot, change `SBD_DELAY_START` to `yes`. The default delay length is double the value of `SBD_WATCHDOG_TIMEOUT`. Alternatively, you can specify an integer, in seconds, instead of `yes`.

> **!** **Important: `SBD_WATCHDOG_TIMEOUT` for diskless SBD and QDevice**
>
> If you use QDevice with diskless SBD, the `SBD_WATCHDOG_TIMEOUT` value must be greater than QDevice's `sync_timeout` value, or SBD will time out and fail to start.
>
> The default value for `sync_timeout` is 30 seconds. Therefore, set `SBD_WATCH-DOG_TIMEOUT` to a greater value, such as `35`.

2. On each node, enable the SBD service:

```
# systemctl enable sbd
```

SBD starts together with the Corosync service whenever the cluster services are started.

3. Restart the cluster services on all nodes at once by using the `--all` option:

```
# crm cluster restart --all
```

This automatically triggers the start of the SBD daemon.

> **!** **Important: Restart cluster services for SBD changes**
>
> If any SBD metadata changes, you must restart the cluster services again. To keep critical cluster resources running during the restart, consider putting the cluster into maintenance mode first. For more information, see *Chapter 28, Executing maintenance tasks*.

4. Check if the parameter `have-watchdog=true` has been automatically set:

```
# crm configure show | grep have-watchdog
        have-watchdog=true
```

5. Run **crm configure** and set the following cluster properties on the crm shell:

```
crm(live)configure# property stonith-enabled="true" ❶
crm(live)configure# property stonith-watchdog-timeout=10 ❷
crm(live)configure# property stonith-timeout=15 ❸
```

❶ This is the default configuration, because clusters without STONITH are not supported. But in case STONITH has been deactivated for testing purposes, make sure this parameter is set to `true` again.

❷ For diskless SBD, this parameter must not equal zero. It defines after how long it is assumed that the fencing target has already self-fenced. Use the following formula to calculate this timeout:

```
stonith-watchdog-timeout >= (SBD_WATCHDOG_TIMEOUT * 2)
```

If you set `stonith-watchdog-timeout` to a negative value, Pacemaker automatically calculates this timeout and sets it to twice the value of `SBD_WATCHDOG_TIMEOUT`.

❸ This parameter must allow sufficient time for fencing to complete. For diskless SBD, use the following formula to calculate this timeout:

```
stonith-timeout >= stonith-watchdog-timeout + 20%
```

> **❗ Important: Diskless SBD timeouts**
>
> With diskless SBD, if the `stonith-timeout` value is smaller than the `stonith-watchdog-timeout` value, failed nodes can become stuck in an `UNCLEAN` state and block failover of active resources.

6. Review your changes with **show**.

7. Submit your changes with **commit** and leave the crm live configuration with **quit**.

13.9 Testing SBD and fencing

To test whether SBD works as expected for node fencing purposes, use one or all of the following methods:

Manually triggering fencing of a node

To trigger a fencing action for node _NODENAME_ :

```
# crm node fence NODENAME
```

Check if the node is fenced and if the other nodes consider the node as fenced after the `stonith-watchdog-timeout`.

Simulating an SBD failure

1. Identify the process ID of the SBD inquisitor:

```
# systemctl status sbd
● sbd.service - Shared-storage based fencing daemon

   Loaded: loaded (/usr/lib/systemd/system/sbd.service; enabled; vendor
 preset: disabled)
   Active: active (running) since Tue 2018-04-17 15:24:51 CEST; 6 days ago
     Docs: man:sbd(8)
  Process: 1844 ExecStart=/usr/sbin/sbd $SBD_OPTS -p /var/run/sbd.pid watch
 (code=exited, status=0/SUCCESS)
 Main PID: 1859 (sbd)
    Tasks: 4 (limit: 4915)
    CGroup: /system.slice/sbd.service
            ├─1859 sbd: inquisitor
[...]
```

2. Simulate an SBD failure by terminating the SBD inquisitor process. In our example, the process ID of the SBD inquisitor is `1859`):

```
# kill -9 1859
```

The node proactively self-fences. The other nodes notice the loss of the node and consider it has self-fenced after the `stonith-watchdog-timeout`.

Triggering fencing through a monitor operation failure

With a normal configuration, a failure of a resource *stop operation* triggers fencing. To trigger fencing manually, you can produce a failure of a resource stop operation. Alternatively, you can temporarily change the configuration of a resource *monitor operation* and produce a monitor failure as described below:

1. Configure an `on-fail=fence` property for a resource monitor operation:

   ```
   op monitor interval=10 on-fail=fence
   ```

2. Let the monitoring operation fail (for example, by terminating the respective daemon, if the resource relates to a service).
   This failure triggers a fencing action.

13.10 Additional mechanisms for storage protection

Apart from node fencing via STONITH there are other methods to achieve storage protection at a resource level. For example, SCSI-3 and SCSI-4 use persistent reservations whereas `sfex` provides a locking mechanism. Both methods are explained in the following subsections.

13.10.1 Configuring an sg_persist resource

The SCSI specifications 3 and 4 define *persistent reservations*. These are SCSI protocol features and can be used for I/O fencing and failover. This feature is implemented in the **sg_persist** Linux command.

> ### Note: SCSI disk compatibility
>
> Any backing disks for `sg_persist` must be SCSI disk compatible. `sg_persist` only works for devices like SCSI disks or iSCSI LUNs. Do *not* use it for IDE, SATA, or any block devices which do not support the SCSI protocol.

Before you proceed, check if your disk supports persistent reservations. Use the following command (replace `DEVICE_ID` with your device name):

```
# sg_persist -n --in --read-reservation -d /dev/disk/by-id/DEVICE_ID
```

The result shows whether your disk supports persistent reservations:

- Supported disk:

```
PR generation=0x0, there is NO reservation held
```

- Unsupported disk:

```
PR in (Read reservation): command not supported
Illegal request, Invalid opcode
```

If you get an error message (like the one above), replace the old disk with an SCSI compatible disk. Otherwise proceed as follows:

1. Create the primitive resource `sg_persist`, using a stable device name for the disk:

```
# crm configure
crm(live)configure# primitive sg sg_persist \
    params devs="/dev/disk/by-id/DEVICE_ID" reservation_type=3 \
    op monitor interval=60 timeout=60
```

2. Create a promotable clone of the `sg_persist` primitive:

```
crm(live)configure# clone clone-sg sg \
    meta promotable=true promoted-max=1 notify=true
```

3. Test the setup. When the resource is promoted, you can mount and write to the disk partitions on the cluster node where the primary instance is running, but you cannot write on the cluster node where the secondary instance is running.

4. Add a file system primitive for Ext4, using a stable device name for the disk partition:

```
crm(live)configure# primitive ext4 Filesystem \
    params device="/dev/disk/by-id/DEVICE_ID" directory="/mnt/ext4" fstype=ext4
```

5. Add the following order relationship plus a collocation between the `sg_persist` clone and the file system resource:

```
crm(live)configure# order o-clone-sg-before-ext4 Mandatory: clone-sg:promote
 ext4:start
crm(live)configure# colocation col-ext4-with-sg-persist inf: ext4 clone-sg:Promoted
```

6. Check all your changes with the **show changed** command.

7. Commit your changes.

For more information, refer to the `sg_persist` man page.

13.10.2   Ensuring exclusive storage activation with `sfex`

This section introduces `sfex`, an additional low-level mechanism to lock access to shared storage exclusively to one node. Note that sfex does not replace STONITH. As sfex requires shared storage, it is recommended that the SBD node fencing mechanism described above is used on another partition of the storage.

By design, sfex cannot be used with workloads that require concurrency (such as OCFS2). It serves as a layer of protection for classic failover style workloads. This is similar to an SCSI-2 reservation in effect, but more general.

13.10.2.1   Overview

In a shared storage environment, a small partition of the storage is set aside for storing one or more locks.

Before acquiring protected resources, the node must first acquire the protecting lock. The ordering is enforced by Pacemaker. The sfex component ensures that even if Pacemaker were subject to a split-brain situation, the lock will never be granted more than once.

These locks must also be refreshed periodically, so that a node's death does not permanently block the lock and other nodes can proceed.

13.10.2.2   Setup

In the following, learn how to create a shared partition for use with sfex and how to configure a resource for the sfex lock in the CIB. A single sfex partition can hold any number of locks, and needs 1 KB of storage space allocated per lock. By default, `sfex_init` creates one lock on the partition.

> ⓘ **Important: Requirements**
>
> - The shared partition for sfex should be on the same logical unit as the data you want to protect.
>
> - The shared sfex partition must not use host-based RAID or DRBD.
>
> - Using an LVM logical volume is possible.

PROCEDURE 13.9: CREATING AN SFEX PARTITION

1. Create a shared partition for use with sfex. Note the name of this partition and use it as a substitute for `/dev/sfex` below.

2. Create the sfex metadata with the following command:

```
# sfex_init -n 1 /dev/sfex
```

3. Verify that the metadata has been created correctly:

```
# sfex_stat -i 1 /dev/sfex ; echo $?
```

   This should return `2`, since the lock is not currently held.

PROCEDURE 13.10: CONFIGURING A RESOURCE FOR THE SFEX LOCK

1. The sfex lock is represented via a resource in the CIB, configured as follows:

```
crm(live)configure# primitive sfex_1 ocf:heartbeat:sfex \
      params device="/dev/sfex" index="1" collision_timeout="1" \
      lock_timeout="70" monitor_interval="10" \
      op monitor interval="10s" timeout="30s" on-fail="fence"
```

2. To protect resources via an sfex lock, create mandatory order and placement constraints between the resources to protect the sfex resource. If the resource to be protected has the ID `filesystem1`:

```
crm(live)configure# order order-sfex-1 Mandatory: sfex_1 filesystem1
crm(live)configure# colocation col-sfex-1 inf: filesystem1 sfex_1
```

3. If using group syntax, add the sfex resource as the first resource to the group:

```
crm(live)configure# group LAMP sfex_1 filesystem1 apache ipaddr
```

13.11   Changing SBD configuration

You might need to change the cluster's SBD configuration for various reasons. For example:

- Changing disk-based SBD to diskless SBD

- Changing diskless SBD to disk-based SBD

- Replacing an SBD device with a new device

- Changing timeout values and other settings

You can use crmsh to change the SBD configuration. This method uses crmsh's default settings, including timeout values.

- *Procedure 13.11, "Redeploying disk-based SBD with crmsh"*

- *Procedure 13.13, "Redeploying diskless SBD with crmsh"*.

If you need to change any settings, or if you use custom settings and need to retain them when replacing a device, you must manually edit the SBD configuration.

- *Procedure 13.12, "Redeploying disk-based SBD manually"*

- *Procedure 13.14, "Redeploying diskless SBD manually"*

PROCEDURE 13.11: REDEPLOYING DISK-BASED SBD WITH CRMSH

Use this procedure to change diskless SBD to disk-based SBD, or to replace an existing SBD device with a new device.

1. Put the cluster into maintenance mode:

   ```
   # crm maintenance on
   ```

   In this state, the cluster stops monitoring all resources, so the services managed by the resources can continue to run even if the cluster services stop.

2. Configure the new device:

   ```
   # crm -F cluster init sbd -s /dev/disk/by-id/DEVICE_ID
   ```

   The `-F` option allows crmsh to reconfigure SBD even if the SBD service is already running.

3. Check the status of the cluster:

```
# crm status
```

Initially, the nodes have a status of `UNCLEAN (offline)`, but after a short time they change to `Online`.

4. Check the SBD configuration. First, check the device metadata:

```
# sbd -d /dev/disk/by-id/DEVICE_ID dump
```

Then check that all nodes in the cluster are assigned to a slot in the device:

```
# sbd -d /dev/disk/by-id/DEVICE_ID list
```

5. Check the status of the SBD service:

```
# systemctl status sbd
```

If you changed diskless SBD to disk-based SBD, check that the following section includes a device ID:

```
CGroup: /system/.slice/sbd.service
        |—23314 "sbd: inquisitor"
        |—23315 "sbd: watcher: /dev/disk/by-id/DEVICE_ID - slot: 0 -
 uuid: DEVICE_UUID"
        |—23316 "sbd: watcher: Pacemaker"
        |—23317 "sbd: watcher: Cluster"
```

6. When the nodes are back online, move the cluster out of maintenance mode and back into normal operation:

```
# crm maintenance off
```

PROCEDURE 13.12: REDEPLOYING DISK-BASED SBD MANUALLY

Use this procedure to change diskless SBD to disk-based SBD, to replace an existing SBD device with a new device, or to change the timeout settings for disk-based SBD.

1. Put the cluster into maintenance mode:

```
# crm maintenance on
```

In this state, the cluster stops monitoring all resources, so the services managed by the resources can continue to run even when you stop the cluster services.

Changing SBD configuration   SLE HA 15 SP5

2. Stop the cluster services, including SBD, on all nodes:

```
# crm cluster stop --all
```

3. Reinitialize the device metadata, specifying the new device ID and timeouts as required:

```
# sbd -d /dev/disk/by-id/DEVICE_ID -1 VALUE -4 VALUE create
```

The `-1` option specifies the `watchdog` timeout.
The `-4` option specifies the `msgwait` timeout. This must be at least double the `watchdog` timeout.
For more information, see *Procedure 13.3, "Initializing the SBD devices"*.

4. Open the file `/etc/sysconfig/sbd`.

5. If you are changing diskless SBD to disk-based SBD, add the following line and specify the device ID. If you are replacing an SBD device, change the value of this line to the new device ID:

```
SBD_DEVICE="/dev/disk/by-id/DEVICE_ID"
```

6. Adjust the other settings as required. For more information, see *Procedure 13.4, "Editing the SBD configuration file"*.

7. Copy the configuration file to all nodes:

```
# csync2 -xv
```

8. Start the cluster services on all nodes:

```
# crm cluster start --all
```

9. Check the status of the cluster:

```
# crm status
```

Initially, the nodes have a status of `UNCLEAN (offline)`, but after a short time they change to `Online`.

10. Check the SBD configuration. First, check the device metadata:

```
# sbd -d /dev/disk/by-id/DEVICE_ID dump
```

Then check that all nodes in the cluster are assigned to a slot in the device:

```
# sbd -d /dev/disk/by-id/DEVICE_ID list
```

11. Check the status of the SBD service:

```
# systemctl status sbd
```

If you changed diskless SBD to disk-based SBD, check that the following section includes a device ID:

```
CGroup: /system/.slice/sbd.service
        |-23314 "sbd: inquisitor"
        |-23315 "sbd: watcher: /dev/disk/by-id/DEVICE_ID - slot: 0 -
 uuid: DEVICE_UUID"
        |-23316 "sbd: watcher: Pacemaker"
        |-23317 "sbd: watcher: Cluster"
```

12. If you changed any timeouts, or if you changed diskless SBD to disk-based SBD, you might also need to change the CIB properties `stonith-timeout` and `stonith-watchdog-timeout`. For disk-based SBD, `stonith-watchdog-timeout` should be `0` or defaulted. For more information, see *Section 13.5, "Calculation of timeouts"*.
To check the current values, run the following command:

```
# crm configure show
```

If you need to change the values, use the following commands:

```
# crm configure property stonith-watchdog-timeout=0
# crm configure property stonith-timeout=VALUE
```

13. If you changed diskless SBD to disk-based SBD, you must configure a STONITH resource for SBD. For example:

```
# crm configure primitive stonith-sbd stonith:external/sbd
```

For more information, see *Step 4* in *Procedure 13.7, "Configuring the cluster to use SBD"*.

14. When the nodes are back online, move the cluster out of maintenance mode and back into normal operation:

```
# crm maintenance off
```

PROCEDURE 13.13: REDEPLOYING DISKLESS SBD WITH CRMSH

Use this procedure to change disk-based SBD to diskless SBD.

1. Put the cluster into maintenance mode:

   ```
   # crm maintenance on
   ```

   In this state, the cluster stops monitoring all resources, so the services managed by the resources can continue to run even if the cluster services stop.

2. Configure diskless SBD:

   ```
   # crm -F cluster init sbd -S
   ```

   The `-F` option allows crmsh to reconfigure SBD even if the SBD service is already running.

3. Check the status of the cluster:

   ```
   # crm status
   ```

   Initially, the nodes have a status of `UNCLEAN (offline)`, but after a short time they change to `Online`.

4. Check the status of the SBD service:

   ```
   # systemctl status sbd
   ```

   The following section should *not* include a device ID:

   ```
   CGroup: /system/.slice/sbd.service
           |—23314 "sbd: inquisitor"
           |—23315 "sbd: watcher: Pacemaker"
           |—23316 "sbd: watcher: Cluster"
   ```

5. When the nodes are back online, move the cluster out of maintenance mode and back into normal operation:

   ```
   # crm maintenance off
   ```

PROCEDURE 13.14: REDEPLOYING DISKLESS SBD MANUALLY

Use this procedure to change disk-based SBD to diskless SBD, or to change the timeout values for diskless SBD.

1. Put the cluster into maintenance mode:

   ```
   # crm maintenance on
   ```

   In this state, the cluster stops monitoring all resources, so the services managed by the resources can continue to run even when you stop the cluster services.

2. Stop the cluster services, including SBD, on all nodes:

   ```
   # crm cluster stop --all
   ```

3. Open the file `/etc/sysconfig/sbd`.

4. If you are changing disk-based SBD to diskless SBD, remove or comment out the `SBD_DE-VICE` entry.

5. Adjust the other settings as required. For more information, see *Section 13.8, "Setting up diskless SBD"*.

6. Copy the configuration file to all nodes:

   ```
   # csync2 -xv
   ```

7. Start the cluster services on all nodes:

   ```
   # crm cluster start --all
   ```

8. Check the status of the cluster:

   ```
   # crm status
   ```

   Initially, the nodes have a status of `UNCLEAN (offline)`, but after a short time they change to `Online`.

9. Check the status of the SBD service:

   ```
   # systemctl status sbd
   ```

   If you changed disk-based SBD to diskless SBD, check that the following section does *not* include a device ID:

   ```
   CGroup: /system/.slice/sbd.service
           |—23314 "sbd: inquisitor"
           |—23315 "sbd: watcher: Pacemaker"
           |—23316 "sbd: watcher: Cluster"
   ```

Changing SBD configuration   SLE HA 15 SP5

10. If you changed any timeouts, or if you changed disk-based SBD to diskless SBD, you might also need to change the CIB properties `stonith-timeout` and `stonith-watchdog-timeout`. For more information, see *Step 5* of *Procedure 13.8, "Configuring diskless SBD"*.

To check the current values, run the following command:

```
# crm configure show
```

If you need to change the values, use the following commands:

```
# crm configure property stonith-watchdog-timeout=VALUE
# crm configure property stonith-timeout=VALUE
```

11. When the nodes are back online, move the cluster out of maintenance mode and back into normal operation:

```
# crm maintenance off
```

13.12 For more information

For more details, see `man sbd`.

14 QDevice and QNetd

QDevice and QNetd participate in quorum decisions. With assistance from the arbitrator `corosync-qnetd`, `corosync-qdevice` provides a configurable number of votes, allowing a cluster to sustain more node failures than the standard quorum rules allow. We recommend deploying `corosync-qnetd` and `corosync-qdevice` for clusters with an even number of nodes, and especially for two-node clusters.

14.1 Conceptual overview

In comparison to calculating quora among cluster nodes, the QDevice-and-QNetd approach has the following benefits:

- It provides better sustainability in case of node failures.

- You can write your own heuristics scripts to affect votes. This is especially useful for complex setups.

- It enables you to configure a QNetd server to provide votes for multiple clusters.

- It allows using diskless SBD for two-node clusters.

- It helps with quorum decisions for clusters with an even number of nodes under split-brain situations, especially for two-node clusters.

A setup with QDevice/QNetd consists of the following components and mechanisms:

QDEVICE/QNETD COMPONENTS AND MECHANISMS

QNetd(`corosync-qnetd`)

> A systemd service (a daemon, the "QNetd server") which is not part of the cluster. The systemd service provides a vote to the `corosync-qdevice` daemon.
>
> To improve security, `corosync-qnetd` can work with TLS for client certificate checking.

QDevice(`corosync-qdevice`)

> A systemd service (a daemon) on each cluster node running together with Corosync. This is the client of `corosync-qnetd`. Its primary use is to allow a cluster to sustain more node failures than standard quorum rules allow.
>
> QDevice is designed to work with different arbitrators. However, currently, only QNetd is supported.

Algorithms

QDevice supports different algorithms, which determine the behavior of how votes are assigned. Currently, the following exist:

- FFSplit ("fifty-fifty split") is the default. It is used for clusters with an even number of nodes. If the cluster splits into two similar partitions, this algorithm provides one vote to one of the partitions, based on the results of heuristics checks and other factors.

- LMS ("last man standing") allows the only remaining node that can see the QNetd server to get the votes. So this algorithm is useful when a cluster with only one active node should remain quorate.

Heuristics

QDevice supports a set of commands ("heuristics"). The commands are executed locally on start-up of cluster services, cluster membership change, successful connection to `corosync-qnetd`, or, optionally, at regular times. The heuristics can be set with the `quorum.device.heuristics` key (in the `corosync.conf` file) or with the `--qdevice-heuristics-mode` option. Both know the values `off` (default), `sync`, and `on`. The difference between `sync` and `on` is that you can additionally execute the above commands regularly.

Only if all commands are executed successfully are the heuristics considered to have passed; otherwise, they failed. The heuristics' result is sent to `corosync-qnetd` where it is used in calculations to determine which partition should be quorate.

Tiebreaker

This is used as a fallback if the cluster partitions are completely equal even with the same heuristics results. It can be configured to be the lowest, the highest, or a specific node ID.

14.2 Requirements and prerequisites

Before setting up QDevice and QNetd, you need to prepare the environment as follows:

- In addition to the cluster nodes, you have a separate machine that will become the QNetd server. See *Section 14.3, "Setting up the QNetd server"*.

- A different physical network than the one that Corosync uses. It is recommended for QDevice to reach the QNetd server. Ideally, the QNetd server should be in a separate rack than the main cluster, or at least on a separate PSU and not in the same network segment as the Corosync ring or rings.

14.3 Setting up the QNetd server

The QNetd server runs outside the cluster and can support multiple clusters (as long as each cluster has a unique name). As such, you cannot move resources to this server.

The QNetd server is almost "state free". Usually, you do not need to change anything in the configuration file `/etc/sysconfig/corosync-qnetd`. By default, the `corosync-qnetd` service runs the daemon as user `coroqnetd` in the group `coroqnetd`. This avoids running the daemon as `root`.

To create a QNetd server, proceed as follows:

1. On the machine that will become the QNetd server, install SUSE Linux Enterprise Server 15 SP5.

2. Enable the SUSE Linux Enterprise High Availability using the command listed in **SUSE-Connect --list-extensions**.

3. Install the `corosync-qnetd` package:

   ```
   # zypper install corosync-qnetd
   ```

   You do not need to manually start the `corosync-qnetd` service. It starts automatically when you configure QDevice on the cluster.

Your QNetd server is ready to accept connections from a QDevice client (`corosync-qdevice`). Further configuration is handled by crmsh when you connect QDevice clients.

14.4 Connecting QDevice clients to the QNetd server

After you set up your QNetd server, you can set up and run the clients. You can connect the clients to the QNetd server during the installation of your cluster, or you can add them later. This procedure documents how to add them later.

> ❗ To complete the QNetd configuration, the clients must have `root` SSH access to the QNetd server.

1. On all nodes, install the `corosync-qdevice` package:

   ```
   # zypper install corosync-qdevice
   ```

2. On one of the nodes, run the following command to configure QDevice:

```
# crm cluster init qdevice
Do you want to configure QDevice (y/n)? y
HOST or IP of the QNetd server to be used []QNETD_SERVER
TCP PORT of QNetd server [5403]
QNetd decision ALGORITHM (ffsplit/lms) [ffsplit]
QNetd TIE_BREAKER (lowest/highest/valid node id) [lowest]
Whether using TLS on QDevice/QNetd (on/off/required) [on]
Heuristics COMMAND to run with absolute path; For multiple commands, use ";" to
  separate []
```

Confirm with `y` that you want to configure QDevice, then enter the host name or IP address of the QNetd server. For the remaining fields, you can accept the default values or change them if required.

The script configures QDevice on the nodes and completes the QNetd configuration on the QNetd server, including generating CA and server certificates, starting the QNetd daemon, and updating the cluster's quorum configuration.

> **❗ Important: `SBD_WATCHDOG_TIMEOUT` for diskless SBD and QDevice**
>
> If you use QDevice with diskless SBD, the `SBD_WATCHDOG_TIMEOUT` value must be greater than QDevice's `sync_timeout` value, or SBD will time out and fail to start.
>
> The default value for `sync_timeout` is 30 seconds. Therefore, in the file `/etc/sysconfig/sbd`, make sure that `SBD_WATCHDOG_TIMEOUT` is set to a greater value, such as `35`.

14.5 Setting up a QDevice with heuristics

If you need additional control over how votes are determined, use heuristics. Heuristics are a set of commands that are executed in parallel.

For this purpose, the command `crm cluster init qdevice` provides the option `--qdevice-heuristics`. You can pass one or more commands (separated by semicolons) with absolute paths.

For example, if your own command for heuristic checks is located at `/usr/sbin/my-script.sh` you can run it on one of your cluster nodes as follows:

```
# crm cluster init qdevice --qnetd-hostname=charlie \
```

```
    --qdevice-heuristics=/usr/sbin/my-script.sh \
    --qdevice-heuristics-mode=on
```

The command or commands can be written in any language such as Shell, Python, or Ruby. If they succeed, they return `0` (zero), otherwise they return an error code.

You can also pass a set of commands. Only when all commands finish successfully (return code is zero), have the heuristics passed.

The `--qdevice-heuristics-mode=on` option lets the heuristics commands run regularly.

14.6 Checking and showing quorum status

You can query the quorum status on one of your cluster nodes as shown in *Example 14.1, "Status of QDevice"*. It shows the status of your QDevice nodes.

EXAMPLE 14.1: STATUS OF QDEVICE

```
# corosync-quorumtool ❶
 Quorum information
 ------------------
Date:           ...
Quorum provider:  corosync_votequorum
Nodes:          2 ❷
Node ID:        3232235777 ❸
Ring ID:        3232235777/8
Quorate:        Yes ❹

Votequorum information
----------------------
Expected votes:  3
Highest expected: 3
Total votes:     3
Quorum:          2
Flags:           Quorate Qdevice

Membership information
----------------------
    Nodeid     Votes    Qdevice Name
 3232235777        1    A,V,NMW 192.168.1.1 (local) ❺
 3232235778        1    A,V,NMW 192.168.1.2 ❺
        0          1            Qdevice
```

❶ As an alternative with an identical result, you can also use the `crm corosync status quorum` command.

② The number of nodes we are expecting. In this example, it is a two-node cluster.

③ As the node ID is not explicitly specified in `corosync.conf` this ID is a 32-bit integer representation of the IP address. In this example, the value `3232235777` stands for the IP address `192.168.1.1`.

④ The quorum status. In this case, the cluster has quorum.

⑤ The status for each cluster node means:

`A` (alive) or `NA` (not alive)

Shows the connectivity status between QDevice and Corosync. If there is a heartbeat between QDevice and Corosync, it is shown as alive (A).

`V` (vote) or `NV` (non vote)

Shows if the quorum device has given a vote (letter `V`) to the node. A letter `V` means that both nodes can communicate with each other. In a split-brain situation, one node would be set to `V` and the other node would be set to `NV`.

`MW` (master wins) or `NMW` (not master wins)

Shows if the quorum device `master_wins` flag is set. By default, the flag is not set, so you see `NMW` (not master wins). See the man page votequorum_qdevice_master_wins(3) for more information.

`NR` (not registered)

Shows that the cluster is not using a quorum device.

If you query the status of the QNetd server, you get a similar output to that shown in *Example 14.2, "Status of QNetd server"*:

EXAMPLE 14.2: STATUS OF QNETD SERVER

```
# corosync-qnetd-tool -lv ❶
Cluster "hacluster": ❷
    Algorithm:          Fifty-Fifty split ❸
    Tie-breaker:        Node with lowest node ID
    Node ID 3232235777: ❹
        Client address:         ::ffff:192.168.1.1:54732
        HB interval:        8000ms
        Configured node list:   3232235777, 3232235778
        Ring ID:            aa10ab0.8
        Membership node list:   3232235777, 3232235778
        Heuristics:         Undefined (membership: Undefined, regular: Undefined)
        TLS active:         Yes (client certificate verified)
        Vote:               ACK (ACK)
```

```
    Node ID 3232235778:
        Client address:          ::ffff:192.168.1.2:43016
        HB interval:             8000ms
        Configured node list:    3232235777, 3232235778
        Ring ID:                 aa10ab0.8
        Membership node list:    3232235777, 3232235778
        Heuristics:              Undefined (membership: Undefined, regular: Undefined)
        TLS active:              Yes (client certificate verified)
        Vote:                    No change (ACK)
```

❶ As an alternative with an identical result, you can also use the `crm corosync status qnetd` command on one of the cluster nodes.

❷ The name of your cluster as set in the configuration file `/etc/corosync/corosync.conf` in the `totem.cluster_name` section.

❸ The algorithm currently used. In this example, it is `FFSplit`.

❹ This is the entry for the node with the IP address `192.168.1.1`. TLS is active.

14.7 For more information

For additional information about QDevice and QNetd, see the man pages of corosync-qdevice(8) and corosync-qnetd(8).

15 Access control lists

The cluster administration tools like crm shell (crmsh) or Hawk2 can be used by `root` or any user in the group `haclient`. By default, these users have full read/write access. To limit access or assign more fine-grained access rights, you can use *Access control lists* (ACLs).

Access control lists consist of an ordered set of access rules. Each rule allows read or write access or denies access to a part of the cluster configuration. Rules are typically combined to produce a specific role, then users may be assigned to a role that matches their tasks.

> **Note: CIB syntax validation version and ACL differences**
>
> This ACL documentation only applies if your CIB is validated with the CIB syntax version `pacemaker-2.0` or higher. For details on how to check this and upgrade the CIB version, see *Note: Upgrading the CIB syntax version*.

15.1 Requirements and prerequisites

Before you start using ACLs on your cluster, make sure the following conditions are fulfilled:

- Ensure you have the same users on all nodes in your cluster, either by using NIS, Active Directory, or by manually adding the same users to all nodes.

- All users for whom you want to modify access rights with ACLs must belong to the `haclient` group.

- All users need to run crmsh by its absolute path `/usr/sbin/crm`.

- If non-privileged users want to run crmsh, their `PATH` variable needs to be extended with `/usr/sbin`.

> ⓘ **Important: Default access rights**
>
> - ACLs are an optional feature. By default, use of ACLs is disabled.
>
> - If ACLs are not enabled, `root` and all users belonging to the `haclient` group have full read/write access to the cluster configuration.
>
> - Even if ACLs are enabled and configured, both `root` and the default CRM owner `hacluster` *always* have full access to the cluster configuration.

15.2   Conceptual overview

Access control lists consist of an ordered set of access rules. Each rule allows read or write access or denies access to a part of the cluster configuration. Rules are typically combined to produce a specific role, then users may be assigned to a role that matches their tasks. An ACL role is a set of rules which describe access rights to CIB. A rule consists of the following:

- an access right like `read`, `write`, or `deny`

- a specification where to apply the rule. This specification can be a type, an ID reference, or an XPath expression. XPath is a language for selecting nodes in an XML document. Refer to https://en.wikipedia.org/wiki/XPath ↗.

Usually, it is convenient to bundle ACLs into roles and assign a specific role to system users (ACL targets). There are these methods to create ACL roles:

- *Section 15.7, "Setting ACL rules via XPath expressions"*. You need to know the structure of the underlying XML to create ACL rules.

- *Section 15.8, "Setting ACL rules via abbreviations"*. Create a shorthand syntax and ACL rules to apply to the matched objects.

15.3   Enabling use of ACLs in your cluster

Before you can start configuring ACLs, you need to *enable* use of ACLs. To do so, use the following command in the crmsh:

```
# crm configure property enable-acl=true
```

Alternatively, use Hawk2 as described in *Procedure 15.1, "Enabling use of ACLs with Hawk2"*.

PROCEDURE 15.1: ENABLING USE OF ACLS WITH HAWK2

1. Log in to Hawk2:

   ```
   https://HAWKSERVER:7630/
   ```

2. In the left navigation bar, select *Cluster Configuration* to display the global cluster options and their current values.

3. Below *Cluster Configuration* click the empty drop-down box and select *enable-acl* to add the parameter. It is added with its default value `No`.

4. Set its value to `Yes` and apply your changes.

15.4 Creating a read-only monitor role

The following subsections describe how to configure read-only access by defining a `monitor` role either in Hawk2 or crm shell.

15.4.1 Creating a read-only monitor role with Hawk2

The following procedures show how to configure read-only access to the cluster configuration by defining a `monitor` role and assigning it to a user. Alternatively, you can use crmsh to do so, as described in *Procedure 15.4, "Adding a monitor role and assigning a user with crmsh"*.

PROCEDURE 15.2: ADDING A MONITOR ROLE WITH HAWK2

1. Log in to Hawk2:

   ```
   https://HAWKSERVER:7630/
   ```

2. In the left navigation bar, select *Roles*.

3. Click *Create*.

4. Enter a unique *Role ID*, for example, `monitor`.

5. As access *Right*, select `Read`.

6. As *Xpath*, enter the XPath expression `/cib`.
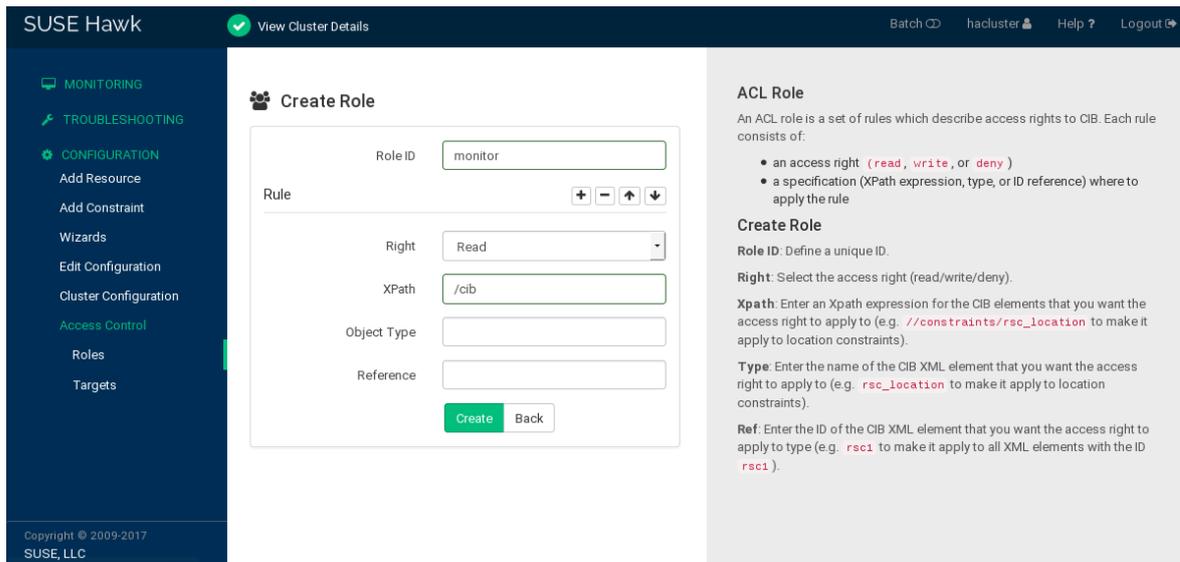
FIGURE 15.1: HAWK2 CREATE ROLE

7. Click *Create*.

   This creates a new role with the name `monitor`, sets the `read` rights and applies this to all elements in the CIB by using the XPath expression `/cib`.

8. If necessary, add more rules by clicking the plus icon and specifying the respective parameters.

9. Sort the individual rules by using the arrow up or down buttons.

PROCEDURE 15.3: ASSIGNING A ROLE TO A TARGET WITH HAWK2

To assign the role we created in *Procedure 15.2* to a system user (target), proceed as follows:

1. Log in to Hawk2:

```
https://HAWKSERVER:7630/
```

2. In the left navigation bar, select *Targets*.

3. To create a system user (ACL Target), click *Create* and enter a unique *Target ID*, for example, `tux`. Make sure this user belongs to the `haclient` group.

4. To assign a role to the target, select one or multiple *Roles*.
   In our example, select the `monitor` role you created in *Procedure 15.2*.
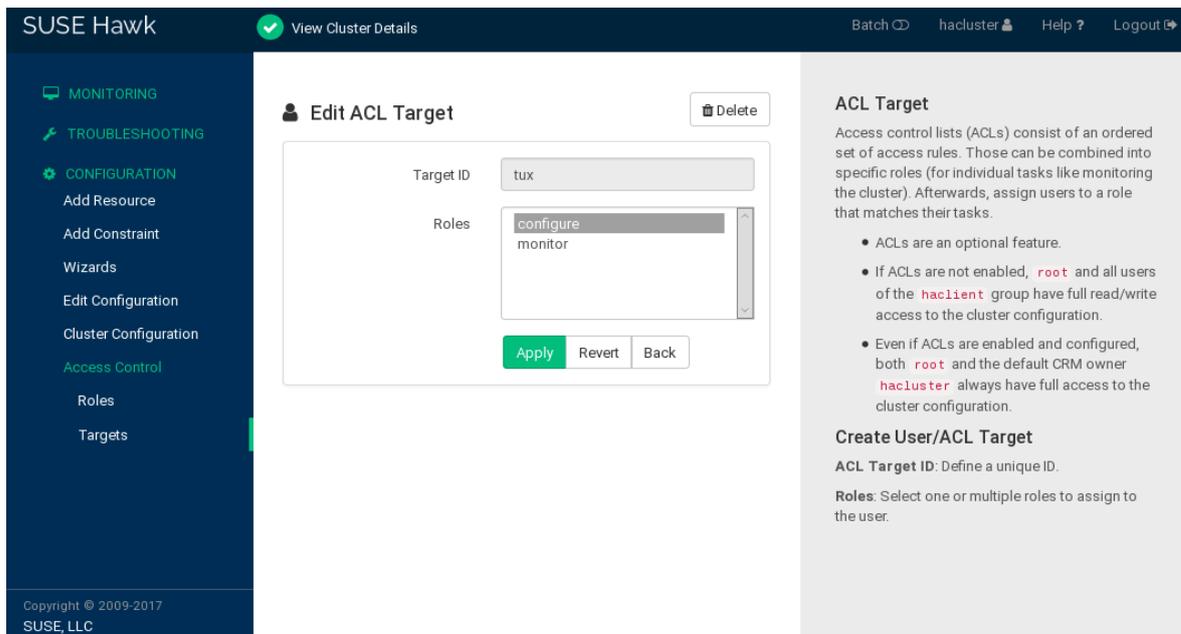
5. Confirm your choice.

To configure access rights for resources or constraints, you can also use the abbreviated syntax as explained in *Section 15.8, "Setting ACL rules via abbreviations"*.

15.4.2 Creating a read-only monitor role with crmsh

The following procedure shows how to configure a read-only access to the cluster configuration by defining a `monitor` role and assigning it to a user.

PROCEDURE 15.4: ADDING A MONITOR ROLE AND ASSIGNING A USER WITH CRMSH

1. Log in as `root`.

2. Start the interactive mode of crmsh:

```
# crm configure
crm(live)configure#
```

3. Define your ACL roles:

a. Use the `role` command to define a new role:

```
crm(live)configure# role monitor read xpath:"/cib"
```

The previous command creates a new role with the name `monitor`, sets the `read` rights and applies it to all elements in the CIB by using the XPath expression `/cib`. If necessary, you can add more access rights and XPath arguments.

b. Add additional roles as needed.

4. Assign your roles to one or multiple ACL targets, which are the corresponding system users. Make sure they belong to the `haclient` group.

```
crm(live)configure# acl_target tux monitor
```

5. Check your changes:

```
crm(live)configure# show
```

6. Commit your changes:

```
crm(live)configure# commit
```

To configure access rights for resources or constraints, you can also use the abbreviated syntax as explained in *Section 15.8, "Setting ACL rules via abbreviations"*.

15.5 Removing a user

The following subsections describe how to remove an existing user from ACL either in Hawk2 or crmsh.

15.5.1 Removing a user with Hawk2

To remove a user from ACL, proceed as follows:

1. Log in to Hawk2:

```
https://HAWKSERVER:7630/
```

2. In the left navigation bar, select *Targets*.

3. To remove a system user (ACL target), click the garbage bin icon under the *Operations* column.

4. Confirm the dialog box.

15.5.2   Removing a user with crmsh

To remove a user from ACL, replace the placeholder _USER_ with the name of the user:

```
# crm configure delete USERNAME
```

As an alternative, you can use the **edit** subcommand:

```
# crm configure edit USERNAME
```

15.6   Removing an existing role

The following subsections describe how to remove an existing role in either Hawk2 or crmsh.

> **Note: Deleting roles with referenced users**
>
> Keep in mind, no user should belong to this role. If there is still a reference to a user in the role, the role cannot be deleted. Delete the references to users first, before you delete the role.

15.6.1   Removing an existing role with Hawk2

To remove a role, proceed as follows:

1. Log in to Hawk2:

   ```
   https://HAWKSERVER:7630/
   ```

2. In the left navigation bar, select _Roles_.

3. To remove a role, click the garbage bin icon under the _Operations_ column.

4. Confirm the dialog box. If an error message appears, make sure your role is "empty" and does not reference users.

15.6.2   Removing an existing role with crmsh

To remove an existing role, replace the placeholder _ROLE_ with the name of the role:

```
# crm configure delete ROLE
```

15.7 Setting ACL rules via XPath expressions

To manage ACL rules via XPath, you need to know the structure of the underlying XML. Retrieve the structure with the following command that shows your cluster configuration in XML (see *Example 15.1*):

```
# crm configure show xml
```

EXAMPLE 15.1: EXCERPT OF A CLUSTER CONFIGURATION IN XML

```
<cib>
  <!-- ... -->
  <configuration>
    <crm_config>
       <cluster_property_set id="cib-bootstrap-options">
        <nvpair name="stonith-enabled" value="true" id="cib-bootstrap-options-stonith-
enabled"/>
       [...]
     </cluster_property_set>
    </crm_config>
    <nodes>
      <node id="175704363" uname="alice"/>
      <node id="175704619" uname="bob"/>
    </nodes>
    <resources> [...]  </resources>
    <constraints/>
    <rsc_defaults> [...] </rsc_defaults>
    <op_defaults> [...] </op_defaults>
  <configuration>
</cib>
```

With the XPath language you can locate nodes in this XML document. For example, to select the root node (`cib`) use the XPath expression `/cib`. To locate the global cluster configurations, use the XPath expression `/cib/configuration/crm_config`.

As an example, the following list shows the XPath expressions to create an "operator" role. Users with this role can only execute the tasks listed here—they cannot reconfigure any resources (for example, change parameters or operations), nor change the configuration of colocation or order constraints.

`//crm_config//nvpair[@name='maintenance-mode']`

    Turn cluster maintenance mode on or off.

`//op_defaults//nvpair[@name='record-pending']`

    Choose whether pending operations are recorded.

`//nodes/node//nvpair[@name='standby']`

Set the node in online or standby mode.

`//resources//nvpair[@name='target-role']`

Start, stop, promote, or demote any resource.

`//resources//nvpair[@name='maintenance']`

Select whether a resource should be put in maintenance mode or not.

`//constraints/rsc_location`

Migrate/move resources from one node to another.

`/cib`

View the status of the cluster.

15.8 Setting ACL rules via abbreviations

For users who do not want to deal with the XML structure, there is an easier method.

For example, consider the following XPath:

```
//*[@id="rsc1"]
```

which locates all the XML nodes with the ID `rsc1`.

The abbreviated syntax is written like this:

```
ref:"rsc1"
```

This also works for constraints. Here is the verbose XPath:

```
//constraints/rsc_location
```

The abbreviated syntax is written like this:

```
type:"rsc_location"
```

The abbreviated syntax can be used in crmsh and Hawk2. The CIB daemon knows how to apply the ACL rules to the matching objects.

16 Network device bonding

For many systems, it is desirable to implement network connections that comply to more than the standard data security or availability requirements of a typical Ethernet device. In these cases, several Ethernet devices can be aggregated to a single bonding device.

The configuration of the bonding device is done by bonding module options. The behavior is determined through the mode of the bonding device. By default, this is `mode=active-backup`, which means that a different device becomes active if the primary device fails.

When using Corosync, the bonding device is not managed by the cluster software. Therefore, the bonding device must be configured on every cluster node that might need to access the bonding device.

16.1 Configuring bonding devices with YaST

To configure a bonding device, you need to have multiple Ethernet devices that can be aggregated to a single bonding device. Proceed as follows:

1. Start YaST as `root` and select *System* › *Network Settings*.

2. In the *Network Settings*, switch to the *Overview* tab, which shows the available devices.

3. Check if the Ethernet devices to be aggregate to a bonding device have an IP address assigned. If yes, change it:

   a. Select the device to change and click *Edit*.

   b. In the *Address* tab of the *Network Card Setup* dialog that opens, select the option *No Link and IP Setup (Bond Ports)*.

   c. Click *Next* to return to the *Overview* tab in the *Network Settings* dialog.

4. To add a new bonding device:

   a. Click *Add* and set the *Device Type* to *Bond*. Proceed with *Next*.

b. Select how to assign the IP address to the bonding device. Three methods are at your disposal:

- No Link and IP Setup (Bond Ports)

- Dynamic Address (with DHCP or Zeroconf)

- Statically assigned IP Address

Use the method that is appropriate for your environment. If Corosync manages virtual IP addresses, select *Statically assigned IP Address* and assign an IP address to the interface.

c. Switch to the *Bond Ports* tab.

d. To select the Ethernet devices that you want to include into the bond, activate the check box in front of the respective devices.

e. Edit the *Bond Driver Options*. The following modes are available:

`balance-rr`
> Provides load balancing and fault tolerance, at the cost of out-of-order packet transmission. This may cause delays, for example, for TCP reassembly.

`active-backup`
> Provides fault tolerance.

`balance-xor`
> Provides load balancing and fault tolerance.

`broadcast`
> Provides fault tolerance.

`802.3ad`
> Provides dynamic link aggregation if supported by the connected switch.

`balance-tlb`
> Provides load balancing for outgoing traffic.

```
balance-alb
```
> Provides load balancing for incoming and outgoing traffic, if the network devices used allow the modifying of the network device's hardware address while in use.

    f. Make sure to add the parameter `miimon=100` to *Bond Driver Options*. Without this parameter, the link is not checked regularly, so the bonding driver might continue to lose packets on a faulty link.

5. Click *Next* and leave YaST with *OK* to finish the configuration of the bonding device. YaST writes the configuration to `/etc/sysconfig/network/ifcfg-bond`*DEVICENUMBER*.

16.2 Hotplugging devices into a bond

Sometimes it is necessary to replace an interface in a bond with another one, for example, if the respective network device constantly fails. The solution is to set up hotplugging. It is also necessary to change the `udev` rules to match the device by bus ID instead of by MAC address. This enables you to replace defective hardware (a network card in the same slot but with a different MAC address), if the hardware allows for that.

PROCEDURE 16.1: HOTPLUGGING DEVICES INTO A BOND WITH YAST

> If you prefer manual configuration instead, refer to the SUSE Linux Enterprise Server Administration Guide, chapter *Basic Networking*, section *Hotplugging of bond ports*.

1. Start YaST as `root` and select *System* › *Network Settings*.

2. In the *Network Settings*, switch to the *Overview* tab, which shows the already configured devices. If devices are already configured in a bond, the *Note* column shows it.

3. For each of the Ethernet devices that have been aggregated to a bonding device, execute the following steps:

    a. Select the device to change and click *Edit*. The *Network Card Setup* dialog opens.

    b. Switch to the *General* tab and make sure that *Activate device* is set to `On Hotplug`.

    c. Switch to the *Hardware* tab.

    d. For the *Udev rules*, click *Change* and select the *BusID* option.

e. Click *OK* and *Next* to return to the *Overview* tab in the *Network Settings* dialog. If you click the Ethernet device entry now, the bottom pane shows the device's details, including the bus ID.

4. Click *OK* to confirm your changes and leave the network settings.

At boot time, the network setup does not wait for the hotplug devices, but for the bond to become ready, which needs at least one available device. When one of the interfaces is removed from the system (unbind from NIC driver, `rmmod` of the NIC driver or true PCI hotplug removal), the Kernel removes it from the bond automatically. When a new card is added to the system (replacement of the hardware in the slot), `udev` renames it by applying the bus-based persistent name rule and calls `ifup` for it. The `ifup` call automatically joins it into the bond.

16.3 For more information

All modes and many options are explained in detail in the *Linux Ethernet Bonding Driver HOWTO* at https://www.kernel.org/doc/Documentation/networking/bonding.txt ↗ .

For High Availability setups, the following options are especially important: `miimon` and `use_carrier`.

17 Load balancing

Load Balancing makes a cluster of servers appear as one large, fast server to outside clients. This apparent single server is called a *virtual server*. It consists of one or more load balancers dispatching incoming requests and several real servers running the actual services. With a load balancing setup of SUSE Linux Enterprise High Availability, you can build highly scalable and highly available network services, such as Web, cache, mail, FTP, media and VoIP services.

17.1 Conceptual overview

SUSE Linux Enterprise High Availability supports two technologies for load balancing: Linux Virtual Server (LVS) and HAProxy. The key difference is Linux Virtual Server operates at OSI layer 4 (Transport), configuring the network layer of kernel, while HAProxy operates at layer 7 (Application), running in user space. Thus Linux Virtual Server needs fewer resources and can handle higher loads, while HAProxy can inspect the traffic, do SSL termination and make dispatching decisions based on the content of the traffic.

On the other hand, Linux Virtual Server includes two different software: IPVS (IP Virtual Server) and KTCPVS (Kernel TCP Virtual Server). IPVS provides layer 4 load balancing whereas KTCPVS provides layer 7 load balancing.

This section gives you a conceptual overview of load balancing in combination with high availability, then briefly introduces you to Linux Virtual Server and HAProxy. Finally, it points you to further reading.

The real servers and the load balancers may be interconnected by either high-speed LAN or by geographically dispersed WAN. The load balancers dispatch requests to the different servers. They make parallel services of the cluster appear as one virtual service on a single IP address (the virtual IP address or VIP). Request dispatching can use IP load balancing technologies or application-level load balancing technologies. Scalability of the system is achieved by transparently adding or removing nodes in the cluster.

High availability is provided by detecting node or service failures and reconfiguring the whole virtual server system appropriately, as usual.

There are several load balancing strategies. Here are some Layer 4 strategies, suitable for Linux Virtual Server:

- **Round robin.** The simplest strategy is to direct each connection to a different address, taking turns. For example, a DNS server can have several entries for a given host name. With DNS round robin, the DNS server will return all of them in a rotating order. Thus different clients will see different addresses.

- **Selecting the "best" server.** Although this has several drawbacks, balancing could be implemented with a "first server who responds" or "least loaded server" approach.

- **Balancing the number of connections per server.** A load balancer between users and servers can divide the number of users across multiple servers.

- **Geographical location.** It is possible to direct clients to a server nearby.

Here are some Layer 7 strategies, suitable for HAProxy:

- **URI.** Inspect the HTTP content and dispatch to a server most suitable for this specific URI.

- **URL parameter, RDP cookie.** Inspect the HTTP content for a session parameter, possibly in post parameters, or the RDP (remote desktop protocol) session cookie, and dispatch to the server serving this session.

Although there is some overlap, HAProxy can be used in scenarios where LVS/`ipvsadm` is not adequate and vice versa:

- **SSL termination.** The front-end load balancers can handle the SSL layer. Thus the cloud nodes do not need to have access to the SSL keys, or could take advantage of SSL accelerators in the load balancers.

- **Application level.** HAProxy operates at the application level, allowing the load balancing decisions to be influenced by the content stream. This allows for persistence based on cookies and other such filters.

On the other hand, LVS/`ipvsadm` cannot be fully replaced by HAProxy:

- LVS supports "direct routing", where the load balancer is only in the inbound stream, whereas the outbound traffic is routed to the clients directly. This allows for potentially much higher throughput in asymmetric environments.

- LVS supports stateful connection table replication (via `conntrackd`). This allows for load balancer failover that is transparent to the client and server.

17.2 Configuring load balancing with Linux Virtual Server

The following sections give an overview of the main LVS components and concepts. Then we explain how to set up Linux Virtual Server on SUSE Linux Enterprise High Availability.

17.2.1 Director

The main component of LVS is the ip_vs (or IPVS) Kernel code. It is part of the default Kernel and implements transport-layer load balancing inside the Linux Kernel (layer-4 switching). The node that runs a Linux Kernel including the IPVS code is called *director*. The IPVS code running on the director is the essential feature of LVS.

When clients connect to the director, the incoming requests are load-balanced across all cluster nodes. The director forwards packets to the real servers, using a modified set of routing rules that make the LVS work. For example, connections do not originate or terminate on the director, it does not send acknowledgments. The director acts as a specialized router that forwards packets from end users to real servers (the hosts that run the applications that process the requests).

17.2.2 User space controller and daemons

The `ldirectord` daemon is a user space daemon for managing Linux Virtual Server and monitoring the real servers in an LVS cluster of load balanced virtual servers. A configuration file (see below) specifies the virtual services and their associated real servers and tells `ldirectord` how to configure the server as an LVS redirector. When the daemon is initialized, it creates the virtual services for the cluster.

By periodically requesting a known URL and checking the responses, the `ldirectord` daemon monitors the health of the real servers. If a real server fails, it is removed from the list of available servers at the load balancer. When the service monitor detects that the dead server has recovered and is working again, it adds the server back to the list of available servers. In case that all real servers should be down, a fall-back server can be specified to which to redirect a Web service. Typically the fall-back server is localhost, presenting an emergency page about the Web service being temporarily unavailable.

The `ldirectord` uses the `ipvsadm` tool (package `ipvsadm`) to manipulate the virtual server table in the Linux Kernel.

17.2.3   Packet forwarding

There are three different methods of how the director can send packets from the client to the real servers:

Network address translation (NAT)

Incoming requests arrive at the virtual IP. They are forwarded to the real servers by changing the destination IP address and port to that of the chosen real server. The real server sends the response to the load balancer which in turn changes the destination IP address and forwards the response back to the client. Thus, the end user receives the replies from the expected source. As all traffic goes through the load balancer, it usually becomes a bottleneck for the cluster.

IP tunneling (IP-IP encapsulation)

IP tunneling enables packets addressed to an IP address to be redirected to another address, possibly on a different network. The LVS sends requests to real servers through an IP tunnel (redirecting to a different IP address) and the real servers reply directly to the client using their own routing tables. Cluster members can be in different subnets.

Direct routing

Packets from end users are forwarded directly to the real server. The IP packet is not modified, so the real servers must be configured to accept traffic for the virtual server's IP address. The response from the real server is sent directly to the client. The real servers and load balancers need to be in the same physical network segment.

17.2.4   Scheduling algorithms

Deciding which real server to use for a new connection requested by a client is implemented using different algorithms. They are available as modules and can be adapted to specific needs. For an overview of available modules, refer to the `ipvsadm(8)` man page. Upon receiving a connect request from a client, the director assigns a real server to the client based on a *schedule*. The scheduler is the part of the IPVS Kernel code which decides which real server gets the next new connection.

More detailed description about Linux Virtual Server scheduling algorithms can be found at http://kb.linuxvirtualserver.org/wiki/IPVS↗. Furthermore, search for `--scheduler` in the `ipvsadm` man page.

Related load balancing strategies for HAProxy can be found at https://www.haproxy.org/download/1.6/doc/configuration.txt ↗.

17.2.5 Setting up IP load balancing with YaST

You can configure Kernel-based IP load balancing with the YaST IP Load Balancing module. It is a front-end for `ldirectord`.

To access the IP Load Balancing dialog, start YaST as `root` and select *High Availability › IP Load Balancing*. Alternatively, start the YaST cluster module as `root` on a command line with `yast2 iplb`.

The default installation does not include the configuration file `/etc/ha.d/ldirectord.cf`. This file is created by the YaST module. The tabs available in the YaST module correspond to the structure of the `/etc/ha.d/ldirectord.cf` configuration file, defining global options and defining the options for the virtual services.

For an example configuration and the resulting processes between load balancers and real servers, refer to *Example 17.1, "Simple ldirectord configuration"*.

> ### Note: Global parameters and virtual server parameters
>
> If a certain parameter is specified in both the virtual server section and in the global section, the value defined in the virtual server section overrides the value defined in the global section.

PROCEDURE 17.1: CONFIGURING GLOBAL PARAMETERS

> The following procedure describes how to configure the most important global parameters. For more details about the individual parameters (and the parameters not covered here), click *Help* or refer to the `ldirectord` man page.
>
> 1. With *Check Interval*, define the interval in which `ldirectord` connects to each of the real servers to check if they are still online.
>
> 2. With *Check Timeout*, set the time in which the real server should have responded after the last check.
>
> 3. With *Failure Count* you can define how many times `ldirectord` attempts to request the real servers until the check is considered failed.

4. With *Negotiate Timeout* define a timeout in seconds for negotiate checks.

5. In *Fallback*, enter the host name or IP address of the Web server onto which to redirect a Web service in case all real servers are down.

6. If you want the system to send alerts in case the connection status to any real server changes, enter a valid e-mail address in *Email Alert*.

7. With *Email Alert Frequency*, define after how many seconds the e-mail alert should be repeated if any of the real servers remains inaccessible.

8. In *Email Alert Status*, specify the server states for which e-mail alerts should be sent. To define more than one state, use a comma-separated list.

9. With *Auto Reload* define, if `ldirectord` should continuously monitor the configuration file for modification. If set to `yes`, the configuration is automatically reloaded upon changes.

10. With the *Quiescent* switch, define whether to remove failed real servers from the Kernel's LVS table or not. If set to *Yes*, failed servers are not removed. Instead their weight is set to `0`, which means that no new connections will be accepted. Already established connections will persist until they time out.

11. To use an alternative path for logging, specify a path for the log files in *Log File*. By default, `ldirectord` writes its log files to `/var/log/ldirectord.log`.

FIGURE 17.1: YAST IP LOAD BALANCING—GLOBAL PARAMETERS

Setting up IP load balancing with YaST   SLE HA 15 SP5
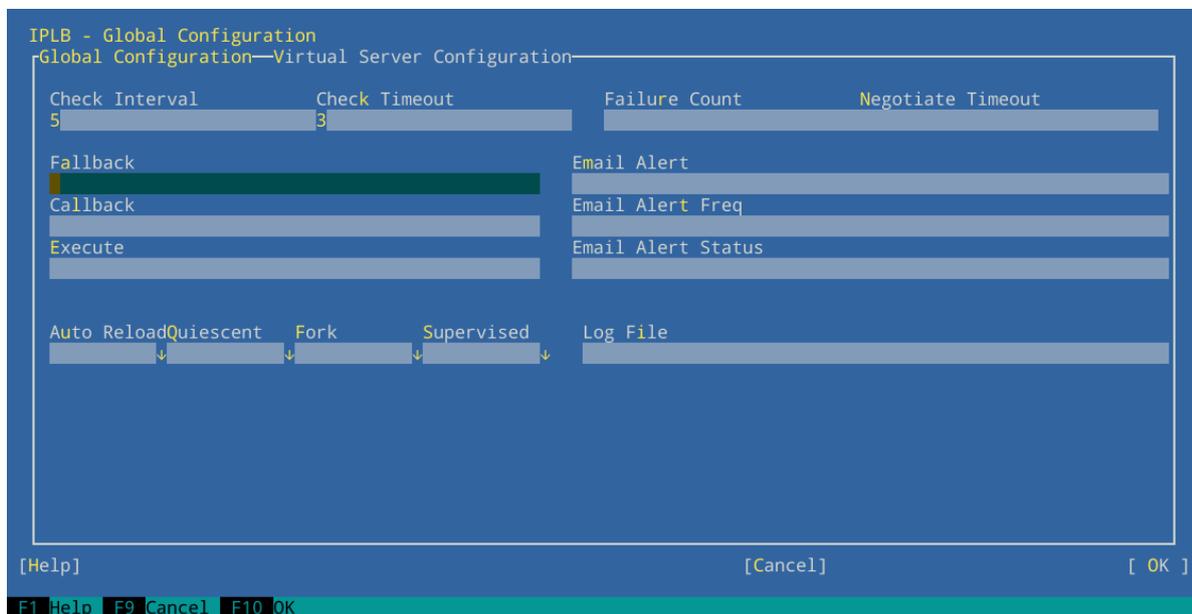
You can configure one or more virtual services by defining a couple of parameters for each. The following procedure describes how to configure the most important parameters for a virtual service. For more details about the individual parameters (and the parameters not covered here), click *Help* or refer to the `ldirectord` man page.

1. In the YaST IP Load Balancing module, switch to the *Virtual Server Configuration* tab.

2. *Add* a new virtual server or *Edit* an existing virtual server. A new dialog shows the available options.

3. In *Virtual Server* enter the shared virtual IP address (IPv4 or IPv6) and port under which the load balancers and the real servers are accessible as LVS. Instead of IP address and port number you can also specify a host name and a service. Alternatively, you can also use a firewall mark. A firewall mark is a way of aggregating an arbitrary collection of `VIP:port` services into one virtual service.

4. To specify the *Real Servers*, you need to enter the IP addresses (IPv4, IPv6, or host names) of the servers, the ports (or service names) and the forwarding method. The forwarding method must either be `gate`, `ipip` or `masq`, see *Section 17.2.3, "Packet forwarding"*. Click the *Add* button and enter the required arguments for each real server.

5. As *Check Type*, select the type of check that should be performed to test if the real servers are still alive. For example, to send a request and check if the response contains an expected string, select `Negotiate`.

6. If you have set the *Check Type* to `Negotiate`, you also need to define the type of service to monitor. Select it from the *Service* drop-down box.

7. In *Request*, enter the URI to the object that is requested on each real server during the check intervals.

8. If you want to check if the response from the real servers contains a certain string ("I'm alive" message), define a regular expression that needs to be matched. Enter the regular expression into *Receive*. If the response from a real server contains this expression, the real server is considered to be alive.

9. Depending on the type of *Service* you have selected in *Step 6*, you also need to specify further parameters for authentication. Switch to the *Auth type* tab and enter the details like *Login, Password, Database,* or *Secret*. For more information, refer to the YaST help text or to the `ldirectord` man page.

10. Switch to the *Others* tab.

11. Select the *Scheduler* to be used for load balancing. For information on the available schedulers, refer to the `ipvsadm(8)` man page.

12. Select the *Protocol* to be used. If the virtual service is specified as an IP address and port, it must be either `tcp` or `udp`. If the virtual service is specified as a firewall mark, the protocol must be `fwm`.

13. Define further parameters, if needed. Confirm your configuration with *OK*. YaST writes the configuration to `/etc/ha.d/ldirectord.cf`.

```
IPLB - Virtual Servers Configuration
Virtual Server
192.168.0.200:80
┌Real Servers────────────────────────────────────────────────────────┐
│192.168.0.110:80 gate                                                 │
│192.168.0.120:80 gate                                                 │
│                                                                      │
│                                                                      │
│                                                           [Add]      │
│                                                           [Delete]   │
│                                                           [Edit]     │
┌Check type──Auth type──Others────────────────────────────────────────┐
│                                                                      │
│Check Type    Check PorService      Check Command                     │
│negotiate     ↓                 ↓                                     │
│Http Method        Request                      Receive               │
│            ↓      test.html                     still alive           │
│Virtual Host                                    Fallback              │
│                                                127.0.0.1:80          │
│                                                                      │
│                                                                      │
│                                                                      │
[Help]                                [Cancel]                 [ OK ]
```
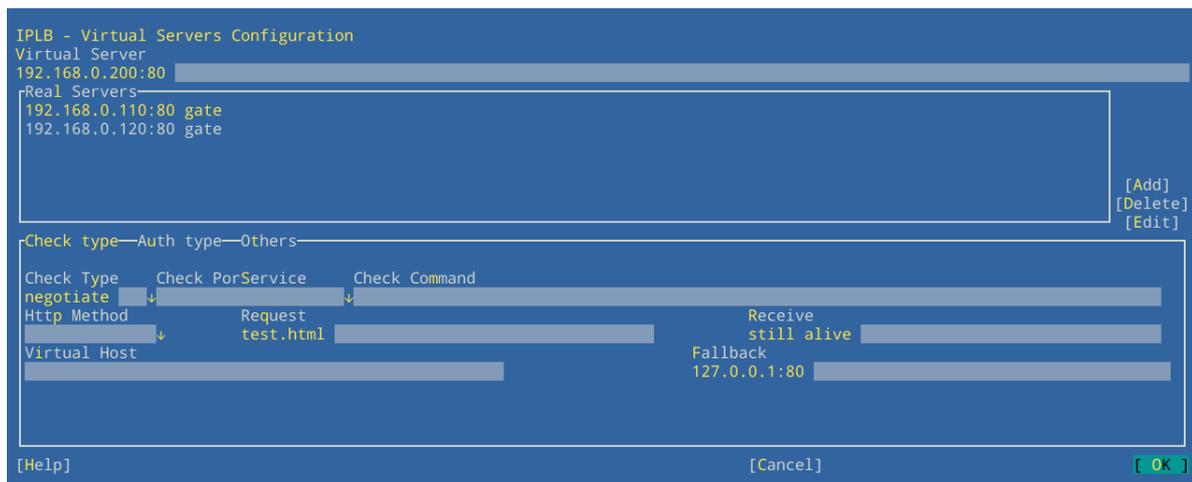
FIGURE 17.2: **YAST IP LOAD BALANCING—VIRTUAL SERVICES**

EXAMPLE 17.1: **SIMPLE LDIRECTORD CONFIGURATION**

The values shown in *Figure 17.1, "YaST IP load balancing—global parameters"* and *Figure 17.2, "YaST IP load balancing—virtual services"*, would lead to the following configuration, defined in `/etc/ha.d/ldirectord.cf`:

```
autoreload = yes ❶
    checkinterval = 5 ❷
    checktimeout = 3 ❸
    quiescent = yes ❹
    virtual = 192.168.0.200:80 ❺
    checktype = negotiate ❻
    fallback = 127.0.0.1:80 ❼
    protocol = tcp ❽
    real = 192.168.0.110:80 gate ❾
    real = 192.168.0.120:80 gate ❾
    receive = "still alive" ❿
```

```
        request = "test.html"  ⑪
        scheduler = wlc  ⑫
        service = http  ⑬
```

① Defines that `ldirectord` should continuously check the configuration file for modification.

② Interval in which `ldirectord` connects to each of the real servers to check if they are still online.

③ Time in which the real server should have responded after the last check.

④ Defines not to remove failed real servers from the Kernel's LVS table, but to set their weight to `0` instead.

⑤ Virtual IP address (VIP) of the LVS. The LVS is available at port `80`.

⑥ Type of check that should be performed to test if the real servers are still alive.

⑦ Server onto which to redirect a Web service all real servers for this service are down.

⑧ Protocol to be used.

⑨ Two real servers defined, both available at port `80`. The packet forwarding method is `gate`, meaning that direct routing is used.

⑩ Regular expression that needs to be matched in the response string from the real server.

⑪ URI to the object that is requested on each real server during the check intervals.

⑫ Selected scheduler to be used for load balancing.

⑬ Type of service to monitor.

This configuration leads to the following process flow: the `ldirectord` connects to each real server once every 5 seconds (②) and requests `192.168.0.110:80/test.html` or `192.168.0.120:80/test.html` as specified in ⑨ and ⑪. If it does not receive the expected `still alive` string (⑩) from a real server within 3 seconds (③) of the last check, it removes the real server from the available pool. However, because of the `quiescent=yes` setting (④), the real server is not removed from the LVS table. Instead its weight is set to `0` so that no new connections to this real server are accepted. Already established connections persist until they time out.

17.2.6 Further setup

Apart from the configuration of `ldirectord` with YaST, you need to make sure the following conditions are fulfilled to complete the LVS setup:

- The real servers are set up correctly to provide the needed services.

- The load balancing server (or servers) must be able to route traffic to the real servers using IP forwarding. The network configuration of the real servers depends on which packet forwarding method you have chosen.

- To prevent the load balancing server (or servers) from becoming a single point of failure for the whole system, you need to set up one or several backups of the load balancer. In the cluster configuration, configure a primitive resource for `ldirectord`, so that `ldirectord` can fail over to other servers in case of hardware failure.

- As the backup of the load balancer also needs the `ldirectord` configuration file to fulfill its task, make sure the `/etc/ha.d/ldirectord.cf` is available on all servers that you want to use as backup for the load balancer. You can synchronize the configuration file with Csync2 as described in *Section 4.7, "Transferring the configuration to all nodes"*.

17.3 Configuring load balancing with HAProxy

The following section gives an overview of the HAProxy and how to set up on High Availability. The load balancer distributes all requests to its back-end servers. It is configured as active/passive, meaning if one server fails, the passive server becomes active. In such a scenario, the user will not notice any interruption.

In this section, we will use the following setup:

- A load balancer, with the IP address `192.168.1.99`.

- A virtual, floating IP address `192.168.1.99`.

- Our servers (usually for Web content) `www.example1.com` (IP: `192.168.1.200`) and `www.example2.com` (IP: `192.168.1.201`)

To configure HAProxy, use the following procedure:

1. Install the `haproxy` package.

2. Create the file `/etc/haproxy/haproxy.cfg` with the following contents:

```
global ❶
  maxconn 256
  daemon

defaults ❷
  log      global
  mode     http
  option   httplog
  option   dontlognull
  retries 3
  option redispatch
  maxconn 2000
  timeout connect    5000   ❸
  timeout client     50s    ❹
  timeout server     50000  ❺

frontend LB
  bind 192.168.1.99:80 ❻
  reqadd X-Forwarded-Proto:\ http
  default_backend LB

backend LB
  mode http
  stats enable
  stats hide-version
  stats uri /stats
  stats realm Haproxy\ Statistics
  stats auth haproxy:password ❼
  balance roundrobin ❽
  option  httpclose
  option forwardfor
  cookie LB insert
  option httpchk GET /robots.txt HTTP/1.0
  server web1-srv 192.168.1.200:80 cookie web1-srv check
  server web2-srv 192.168.1.201:80 cookie web2-srv check
```

❶ Section which contains process-wide and OS-specific options.

> `maxconn`
>> Maximum per-process number of concurrent connections.

> `daemon`
>> Recommended mode, HAProxy runs in the background.

❷ Section which sets default parameters for all other sections following its declaration. Some important lines:

`redispatch`

> Enables or disables session redistribution in case of connection failure.

`log`

> Enables logging of events and traffic.

`mode http`

> Operates in HTTP mode (recommended mode for HAProxy). In this mode, a request will be analyzed before a connection to any server is performed. Request that are not RFC-compliant will be rejected.

`option forwardfor`

> Adds the HTTP `X-Forwarded-For` header into the request. You need this option if you want to preserve the client's IP address.

❸ The maximum time to wait for a connection attempt to a server to succeed.

❹ The maximum time of inactivity on the client side.

❺ The maximum time of inactivity on the server side.

❻ Section which combines front-end and back-end sections in one.

`balance leastconn`

> Defines the load balancing algorithm, see https://cbonte.github.io/haproxy-dconv/configuration-1.5.html#4-balance ↗.

`stats enable` ,
`stats auth`

> Enables statistics reporting (by `stats enable`). The `auth` option logs statistics with authentication to a specific account.

❼ Credentials for HAProxy Statistic report page.

❽ Load balancing will work in a round-robin process.

3. Test your configuration file:

```
# haproxy -f /etc/haproxy/haproxy.cfg -c
```

4. Add the following line to Csync2's configuration file `/etc/csync2/csync2.cfg` to make sure the HAProxy configuration file is included:

```
include /etc/haproxy/haproxy.cfg
```

5. Synchronize it:

```
# csync2 -f /etc/haproxy/haproxy.cfg
# csync2 -xv
```

> **Note**
>
> The Csync2 configuration part assumes that the HA nodes were configured using the bootstrap scripts provided by the crm shell. For details, see the Installation and Setup Quick Start.

6. Make sure HAProxy is disabled on both load balancers (`alice` and `bob`) as it is started by Pacemaker:

```
# systemctl disable haproxy
```

7. Configure a new CIB:

```
# crm
crm(live)# cib new haproxy-config
crm(haproxy-config)# primitive haproxy systemd:haproxy \
    op start timeout=120 interval=0 \
    op stop timeout=120 interval=0 \
    op monitor timeout=100 interval=5s \
    meta target-role=Started
crm(haproxy-config)# primitive vip IPaddr2 \
    params ip=192.168.1.99 nic=eth0 cidr_netmask=23 broadcast=192.168.1.255 \
    op monitor interval=5s timeout=120 on-fail=restart
crm(haproxy-config)# group g-haproxy vip haproxy
```

8. Verify the new CIB and correct any errors:

```
crm(haproxy-config)# verify
```

9. Commit the new CIB:

```
crm(haproxy-config)# cib use live
crm(live)# cib commit haproxy-config
```

17.4 For more information

- https://www.haproxy.org ↗

- Project home page at http://www.linuxvirtualserver.org/ ↗.

- For more information about `ldirectord`, refer to its comprehensive man page.

- LVS Knowledge Base: http://kb.linuxvirtualserver.org/wiki/Main_Page ↗

18 High Availability for virtualization

This chapter explains how to configure virtual machines as highly available cluster resources.

18.1 Overview

Virtual machines can take different roles in a High Availability cluster:

- A virtual machine can be managed by the cluster as a resource, without the cluster managing the services that run on the virtual machine. In this case, the VM is opaque to the cluster. This is the scenario described in this document.

- A virtual machine can be a cluster resource and run `pacemaker_remote`, which allows the cluster to manage services running on the virtual machine. In this case, the VM is a guest node and is transparent to the cluster. For this scenario, see *Article "Pacemaker Remote Quick Start", Section 4 "Use case 2: setting up a cluster with guest nodes"*.

- A virtual machine can run a full cluster stack. In this case, the VM is a regular cluster node and is not managed by the cluster as a resource. For this scenario, see *Article "Installation and Setup Quick Start"*.

The following procedures describe how to set up highly available virtual machines on block storage, with another block device used as an OCFS2 volume to store the VM lock files and XML configuration files. The virtual machines and the OCFS2 volume are configured as resources managed by the cluster, with resource constraints to ensure that the lock file directory is always available before a virtual machine starts on any node. This prevents the virtual machines from starting on multiple nodes.

18.2 Requirements

- A running High Availability cluster with at least two nodes and a fencing device such as SBD.

- Passwordless `root` SSH login between the cluster nodes.

- A network bridge on each cluster node, to be used for installing and running the VMs. This must be separate from the network used for cluster communication and management.

- Two or more shared storage devices (or partitions on a single shared device), so that all cluster nodes can access the files and storage required by the VMs:

  - A device to use as an OCFS2 volume, which will store the VM lock files and XML configuration files. Creating and mounting the OCFS2 volume is explained in the following procedure.

  - A device containing the VM installation source (such as an ISO file or disk image).

  - Depending on the installation source, you might also need another device for the VM storage disks.

  To avoid I/O starvation, these devices must be separate from the shared device used for SBD.

- Stable device names for all storage paths, for example, `/dev/disk/by-id/DEVICE_ID`. A shared storage device might have mismatched `/dev/sdX` names on different nodes, which will cause VM migration to fail.

18.3 Configuring cluster resources to manage the lock files

Use this procedure to configure the cluster to manage the virtual machine lock files. The lock file directory must be available on all nodes so that the cluster is aware of the lock files no matter which node the VMs are running on.

You only need to run the following commands on one of the cluster nodes.

PROCEDURE 18.1: CONFIGURING CLUSTER RESOURCES TO MANAGE THE LOCK FILES

1. Create an OCFS2 volume on one of the shared storage devices:

   ```
   # mkfs.ocfs2 /dev/disk/by-id/DEVICE_ID
   ```

2. Run `crm configure` to start the `crm` interactive shell.

3. Create a primitive resource for DLM:

   ```
   crm(live)configure# primitive dlm ocf:pacemaker:controld \
     op monitor interval=60 timeout=60
   ```

4. Create a primitive resource for the OCFS2 volume:

```
crm(live)configure# primitive ocfs2 Filesystem \
  params device="/dev/disk/by-id/DEVICE_ID" directory="/mnt/shared" fstype=ocfs2 \
  op monitor interval=20 timeout=40
```

5. Create a group for the DLM and OCFS2 resources:

```
crm(live)configure# group g-virt-lock dlm ocfs2
```

6. Clone the group so that it runs on all nodes:

```
crm(live)configure# clone cl-virt-lock g-virt-lock \
  meta interleave=true
```

7. Review your changes with `show`.

8. If everything is correct, submit your changes with `commit` and leave the crm live configuration with `quit`.

9. Check the status of the group clone. It should be running on all nodes:

```
# crm status
[...]
Full List of Resources:
[...]
  * Clone Set: cl-virt-lock [g-virt-lock]:
    * Started: [ alice bob ]
```

18.4 Preparing the cluster nodes to host virtual machines

Use this procedure to install and start the required virtualization services, and to configure the nodes to store the VM lock files on the shared OCFS2 volume.

This procedure uses `crm cluster run` to run commands on all nodes at once. If you prefer to manage each node individually, you can omit the `crm cluster run` portion of the commands.

PROCEDURE 18.2: PREPARING THE CLUSTER NODES TO HOST VIRTUAL MACHINES

1. Install the virtualization packages on all nodes in the cluster:

```
# crm cluster run "zypper install -y -t pattern kvm_server kvm_tools"
```

2. On one node, find and enable the `lock_manager` setting in the file `/etc/libvirt/qe-mu.conf`:

```
lock_manager = "lockd"
```

3. On the same node, find and enable the `file_lockspace_dir` setting in the file `/etc/libvirt/qemu-lockd.conf`, and change the value to point to a directory on the OCFS2 volume:

```
file_lockspace_dir = "/mnt/shared/lockd"
```

4. Copy these files to the other nodes in the cluster:

```
# crm cluster copy /etc/libvirt/qemu.conf
# crm cluster copy /etc/libvirt/qemu-lockd.conf
```

5. Enable and start the `libvirtd` service on all nodes in the cluster:

```
# crm cluster run "systemctl enable --now libvirtd"
```

This also starts the `virtlockd` service.

18.5   Adding virtual machines as cluster resources

Use this procedure to add virtual machines to the cluster as cluster resources, with resource constraints to ensure the VMs can always access the lock files. The lock files are managed by the resources in the group `g-virt-lock`, which is available on all nodes via the clone `cl-virt-lock`.

PROCEDURE 18.3: ADDING VIRTUAL MACHINES AS CLUSTER RESOURCES

1. Install your virtual machines on one of the cluster nodes, with the following restrictions:

   - The installation source and storage must be on shared devices.

   - Do not configure the VMs to start on host boot.

   For more information, see *Virtualization Guide* for SUSE Linux Enterprise Server (https://documentation.suse.com/sles/html/SLES-all/book-virtualization.html) ↗.

2. If the virtual machines are running, shut them down. The cluster will start the VMs after you add them as resources.

3. Dump the XML configuration to the OCFS2 volume. Repeat this step for each VM:

```
# virsh dumpxml VM1 > /mnt/shared/VM1.xml
```

> ❗ Make sure the XML files do not contain any references to unshared local paths.

4. Run **crm configure** to start the **crm** interactive shell.

5. Create primitive resources to manage the virtual machines. Repeat this step for each VM:

```
crm(live)configure# primitive VM1 VirtualDomain \
  params config="/mnt/shared/VM1.xml" remoteuri="qemu+ssh://%n/system" \
  meta allow-migrate=true \
  op monitor timeout=30s interval=10s
```

The option `allow-migrate=true` enables live migration. If the value is set to `false`, the cluster migrates the VM by shutting it down on one node and restarting it on another node. If you need to set utilization attributes to help place VMs based on their load impact, see *Section 7.10, "Placing resources based on their load impact"*.

6. Create a colocation constraint so that the virtual machines can only start on nodes where `cl-virt-lock` is running:

```
crm(live)configure# colocation col-fs-virt inf: ( VM1 VM2 VMX ) cl-virt-lock
```

7. Create an ordering constraint so that `cl-virt-lock` always starts before the virtual machines:

```
crm(live)configure# order o-fs-virt Mandatory: cl-virt-lock ( VM1 VM2 VMX )
```

8. Review your changes with **show**.

9. If everything is correct, submit your changes with **commit** and leave the crm live configuration with **quit**.

10. Check the status of the virtual machines:

```
# crm status
[...]
Full List of Resources:
[...]
  * Clone Set: cl-virt-lock [g-virt-lock]:
    * Started: [ alice bob ]
```

```
    * VM1 (ocf::heartbeat:VirtualDomain): Started alice
    * VM2 (ocf::heartbeat:VirtualDomain): Started alice
    * VMX (ocf::heartbeat:VirtualDomain): Started alice
```

The virtual machines are now managed by the High Availability cluster, and can migrate between the cluster nodes.

> ❗ **Important: Do not manually start or stop cluster-managed VMs**
>
> After adding virtual machines as cluster resources, do not manage them manually. Only use the cluster tools as described in *Chapter 8, Managing cluster resources*.
>
> To perform maintenance tasks on cluster-managed VMs, see *Section 28.2, "Different options for maintenance tasks"*.

18.6 Testing the setup

Use the following tests to confirm that the virtual machine High Availability setup works as expected.

> ❗ Perform these tests in a test environment, not a production environment.

PROCEDURE 18.4: VERIFYING THAT THE VM RESOURCE IS PROTECTED ACROSS CLUSTER NODES

1. The virtual machine `VM1` is running on node `alice`.

2. On node `bob`, try to start the VM manually with `virsh start VM1`.

3. **Expected result:** The `virsh` command fails. `VM1` cannot be started manually on `bob` when it is running on `alice`.

PROCEDURE 18.5: VERIFYING THAT THE VM RESOURCE CAN LIVE MIGRATE BETWEEN CLUSTER NODES

1. The virtual machine `VM1` is running on node `alice`.

2. Open two terminals.

3. In the first terminal, connect to `VM1` via SSH.

4. In the second terminal, try to migrate `VM1` to node `bob` with `crm resource move VM1 bob`.

5. Run `crm_mon -r` to monitor the cluster status until it stabilizes. This might take a short time.

6. In the first terminal, check whether the SSH connection to `VM1` is still active.

7. **Expected result:** The cluster status shows that `VM1` has started on `bob`. The SSH connection to `VM1` remains active during the whole migration.

PROCEDURE 18.6: VERIFYING THAT THE VM RESOURCE CAN MIGRATE TO ANOTHER NODE WHEN THE CURRENT NODE REBOOTS

1. The virtual machine `VM1` is running on node `bob`.

2. Reboot `bob`.

3. On node `alice`, run `crm_mon -r` to monitor the cluster status until it stabilizes. This might take a short time.

4. **Expected result:** The cluster status shows that `VM1` has started on `alice`.

PROCEDURE 18.7: VERIFYING THAT THE VM RESOURCE CAN FAIL OVER TO ANOTHER NODE WHEN THE CURRENT NODE CRASHES

1. The virtual machine `VM1` is running on node `alice`.

2. Simulate a crash on `alice` by forcing the machine off or unplugging the power cable.

3. On node `bob`, run `crm_mon -r` to monitor the cluster status until it stabilizes. VM failover after a node crashes usually takes longer than VM migration after a node reboots.

4. **Expected result:** After a short time, the cluster status shows that `VM1` has started on `bob`.

19 Geo clusters (multi-site clusters)

Apart from local clusters and metro area clusters, SUSE® Linux Enterprise High
Availability 15 SP5 also supports geographically dispersed clusters (Geo clusters,
sometimes also called multi-site clusters). That means you can have multiple, geo-
graphically dispersed sites with a local cluster each. Failover between these clusters
is coordinated by a higher level entity, the so-called `booth`. For details on how to
use and set up Geo clusters, refer to *Article "Geo Clustering Quick Start"* and *Book "Geo
Clustering Guide"*.

III   Storage and data replication

20 Distributed Lock Manager (DLM)

The Distributed Lock Manager (DLM) in the kernel is the base component used by OCFS2, GFS2, Cluster MD, and Cluster LVM (lvmlockd) to provide active-active storage at each respective layer.

20.1 Protocols for DLM communication

To avoid single points of failure, redundant communication paths are important for High Availability clusters. This is also true for DLM communication. If network bonding (Link Aggregation Control Protocol, LACP) cannot be used for any reason, we highly recommend defining a redundant communication channel (a second ring) in Corosync. For details, see *Procedure 4.3, "Defining a redundant communication channel"*.

DLM communicates through port 21064 using either the TCP or SCTP protocol, depending on the configuration in `/etc/corosync/corosync.conf`:

- If *rrp_mode* is set to `none` (which means redundant ring configuration is disabled), DLM automatically uses TCP. However, without a redundant communication channel, DLM communication fails if the TCP link is down.

- If *rrp_mode* is set to `passive` (which is the typical setting), and a second communication ring in `/etc/corosync/corosync.conf` is configured correctly, DLM automatically uses SCTP. In this case, DLM messaging has the redundancy capability provided by SCTP.

20.2 Configuring DLM cluster resources

DLM uses the cluster membership services from Pacemaker which run in user space. Therefore, DLM needs to be configured as a clone resource that is present on each node in the cluster.

Note: DLM resource for several solutions

As OCFS2, GFS2, Cluster MD, and Cluster LVM (lvmlockd) all use DLM, it is enough to configure one resource for DLM. As the DLM resource runs on all nodes in the cluster it is configured as a clone resource.

If you have a setup that includes both OCFS2 and Cluster LVM, configuring *one* DLM resource for both OCFS2 and Cluster LVM is enough. In this case, configure DLM using *Procedure 20.1, "Configuring a base group for DLM"*.

However, if you need to keep the resources that use DLM independent from one another (such as multiple OCFS2 mount points), use separate colocation and order constraints instead of a group. In this case, configure DLM using *Procedure 20.2, "Configuring an independent DLM resource"*.

PROCEDURE 20.1: CONFIGURING A BASE GROUP FOR DLM

This configuration consists of a base group that includes several primitives and a base clone. Both base group and base clone can be used in various scenarios afterward (for both OCFS2 and Cluster LVM, for example). You only need to extend the base group with the respective primitives as needed. As the base group has internal colocation and ordering, this simplifies the overall setup as you do not need to specify several individual groups, clones and their dependencies.

1. Log in to a node as `root` or equivalent.

2. Run **crm configure**.

3. Create the primitive resource for DLM:

```
crm(live)configure# primitive dlm ocf:pacemaker:controld \
  op monitor interval="60" timeout="60"
```

4. Create a base group for the `dlm` resource and further storage-related resources:

```
crm(live)configure# group g-storage dlm
```

5. Clone the `g-storage` group so that it runs on all nodes:

```
crm(live)configure#  clone cl-storage g-storage \
  meta interleave=true target-role=Started
```

6. Review your changes with **show**.

7. If everything is correct, submit your changes with **commit** and leave the crm live configuration with **quit**.

Note: Failure when disabling STONITH

Clusters without STONITH are not supported. If you set the global cluster option `stonith-enabled` to `false` for testing or troubleshooting purposes, the DLM resource and all services depending on it (such as Cluster LVM, GFS2, and OCFS2) will fail to start.

PROCEDURE 20.2: CONFIGURING AN INDEPENDENT DLM RESOURCE

This configuration consists of a primitive and a clone, but no group. By adding colocation and order constraints, you can avoid introducing dependencies between multiple resources that use DLM (such as multiple OCFS2 mount points).

1. Log in to a node as `root` or equivalent.

2. Run **crm configure**.

3. Create the primitive resource for DLM:

```
crm(live)configure# primitive dlm ocf:pacemaker:controld \
  op start timeout=90 interval=0 \
  op stop timeout=100 interval=0 \
  op monitor interval=60 timeout=60
```

4. Clone the `dlm` resource so that it runs on all nodes:

```
crm(live)configure#  clone cl-dlm dlm meta interleave=true
```

5. Review your changes with **show**.

6. If everything is correct, submit your changes with **commit** and leave the crm live configuration with **quit**.

21 OCFS2

Oracle Cluster File System 2 (OCFS2) is a general-purpose journaling file system that has been fully integrated since the Linux 2.6 Kernel. OCFS2 allows you to store application binary files, data files, and databases on devices on shared storage. All nodes in a cluster have concurrent read and write access to the file system. A user space control daemon, managed via a clone resource, provides the integration with the HA stack, in particular with Corosync and the Distributed Lock Manager (DLM).

21.1 Features and benefits

OCFS2 can be used for the following storage solutions for example:

- General applications and workloads.

- Xen image store in a cluster. Xen virtual machines and virtual servers can be stored on OCFS2 volumes that are mounted by cluster servers. This provides quick and easy portability of Xen virtual machines between servers.

- LAMP (Linux, Apache, MySQL, and PHP | Perl | Python) stacks.

As a high-performance, symmetric, and parallel cluster file system, OCFS2 supports the following functions:

- An application's files are available to all nodes in the cluster. Users simply install it once on an OCFS2 volume in the cluster.

- All nodes can concurrently read and write directly to storage via the standard file system interface, enabling easy management of applications that run across the cluster.

- File access is coordinated through DLM. DLM control is good for most cases, but an application's design might limit scalability if it contends with the DLM to coordinate file access.

- Storage backup functionality is available on all back-end storage. An image of the shared application files can be easily created, which can help provide effective disaster recovery.

OCFS2 also provides the following capabilities:

- Metadata caching.

- Metadata journaling.

- Cross-node file data consistency.

- Support for multiple-block sizes up to 4 KB, cluster sizes up to 1 MB, for a maximum volume size of 4 PB (Petabyte).

- Support for up to 32 cluster nodes.

- Asynchronous and direct I/O support for database files for improved database performance.

> ## Note: Support for OCFS2
>
> OCFS2 is only supported by SUSE when used with the pcmk (Pacemaker) stack, as provided by SUSE Linux Enterprise High Availability. SUSE does not provide support for OCFS2 in combination with the o2cb stack.

21.2 OCFS2 packages and management utilities

The OCFS2 Kernel module (`ocfs2`) is installed automatically in SUSE Linux Enterprise High Availability 15 SP5. To use OCFS2, make sure the following packages are installed on each node in the cluster: `ocfs2-tools` and the matching `ocfs2-kmp-*` packages for your Kernel.

The `ocfs2-tools` package provides the following utilities for management of OCFS2 volumes. For syntax information, see their man pages.

debugfs.ocfs2

    Examines the state of the OCFS2 file system for debugging.

defragfs.ocfs2

    Reduces fragmentation of the OCFS2 file system.

fsck.ocfs2

    Checks the file system for errors and optionally repairs errors.

mkfs.ocfs2

    Creates an OCFS2 file system on a device, usually a partition on a shared physical or logical disk.

mounted.ocfs2

    Detects and lists all OCFS2 volumes on a clustered system. Detects and lists all nodes on the system that have mounted an OCFS2 device or lists all OCFS2 devices.

tunefs.ocfs2

Changes OCFS2 file system parameters, including the volume label, number of node slots, journal size for all node slots, and volume size.

21.3   Configuring OCFS2 services and a STONITH resource

Before you can create OCFS2 volumes, you must configure the following resources as services in the cluster: DLM and a STONITH resource.

The following procedure uses the `crm` shell to configure the cluster resources. Alternatively, you can also use Hawk2 to configure the resources as described in *Section 21.6, "Configuring OCFS2 resources with Hawk2"*.

PROCEDURE 21.1: CONFIGURING A STONITH RESOURCE

> 📝 **Note: STONITH device needed**
>
> You need to configure a fencing device. Without a STONITH mechanism (like `external/sbd`) in place the configuration will fail.

1. Start a shell and log in as `root` or equivalent.

2. Create an SBD partition as described in *Procedure 13.3, "Initializing the SBD devices"*.

3. Run `crm configure`.

4. Configure `external/sbd` as the fencing device:

   ```
   crm(live)configure# primitive sbd_stonith stonith:external/sbd \
     params pcmk_delay_max=30 meta target-role="Started"
   ```

5. Review your changes with `show`.

6. If everything is correct, submit your changes with `commit` and leave the crm live configuration with `quit`.

For details on configuring the resource for DLM, see *Section 20.2, "Configuring DLM cluster resources"*.

21.4   Creating OCFS2 volumes

After you have configured a DLM cluster resource as described in *Section 21.3, "Configuring OCFS2 services and a STONITH resource"*, configure your system to use OCFS2 and create OCFs2 volumes.

> ### 🏅 Note: OCFS2 volumes for application and data files
>
> We recommend that you generally store application files and data files on different OCFS2 volumes. If your application volumes and data volumes have different requirements for mounting, it is mandatory to store them on different volumes.

Before you begin, prepare the block devices you plan to use for your OCFS2 volumes. Leave the devices as free space.

Then create and format the OCFS2 volume with the `mkfs.ocfs2` as described in *Procedure 21.2, "Creating and formatting an OCFS2 volume"*. The most important parameters for the command are listed below. For more information and the command syntax, refer to the `mkfs.ocfs2` man page.

Volume Label (`-L`)

A descriptive name for the volume to make it uniquely identifiable when it is mounted on different nodes. Use the `tunefs.ocfs2` utility to modify the label as needed.

Cluster Size (`-C`)

The smallest unit of space allocated to a file to hold the data. For the available options and recommendations, refer to the `mkfs.ocfs2` man page.

Number of Node Slots (`-N`)

The maximum number of nodes that can concurrently mount a volume. For each of the nodes, OCFS2 creates separate system files, such as the journals. Nodes that access the volume can be a combination of little-endian architectures (such as AMD64/Intel 64) and big-endian architectures (such as S/390x).

Node-specific files are called local files. A node slot number is appended to the local file. For example: `journal:0000` belongs to whatever node is assigned to slot number `0`.

Set each volume's maximum number of node slots when you create it, according to how many nodes that you expect to concurrently mount the volume. Use the `tunefs.ocfs2` utility to increase the number of node slots as needed. The value cannot be decreased, and one node slot consumes about 100 MiB of disk space.

If the `-N` parameter is not specified, the number of node slots is decided based on the size of the file system. For the default value, refer to the **mkfs.ocfs2** man page.

Block Size (`-b`)

The smallest unit of space addressable by the file system. Specify the block size when you create the volume. For the available options and recommendations, refer to the **mkfs.ocfs2** man page.

Specific Features On/Off (`--fs-features`)

A comma-separated list of feature flags can be provided, and `mkfs.ocfs2` will try to create the file system with those features set according to the list. To turn a feature on, include it in the list. To turn a feature off, prepend `no` to the name.

For an overview of all available flags, refer to the **mkfs.ocfs2** man page.

Predefined Features (`--fs-feature-level`)

Allows you to choose from a set of predetermined file system features. For the available options, refer to the **mkfs.ocfs2** man page.

If you do not specify any features when creating and formatting the volume with **mkfs.ocfs2**, the following features are enabled by default: `backup-super`, `sparse`, `inline-data`, `unwritten`, `metaecc`, `indexed-dirs`, and `xattr`.

PROCEDURE 21.2: CREATING AND FORMATTING AN OCFS2 VOLUME

Execute the following steps only on *one* of the cluster nodes.

1. Open a terminal window and log in as `root`.

2. Check if the cluster is online with the command **crm status**.

3. Create and format the volume using the **mkfs.ocfs2** utility. For information about the syntax for this command, refer to the **mkfs.ocfs2** man page.

   For example, to create a new OCFS2 file system that supports up to 32 cluster nodes, enter the following command:

   ```
   # mkfs.ocfs2 -N 32 /dev/disk/by-id/DEVICE_ID
   ```

   Always use a stable device name (for example: `/dev/disk/by-id/scsi-ST2000D-M001-0123456_Wabcdefg`).

21.5 Mounting OCFS2 volumes

You can either mount an OCFS2 volume manually or with the cluster manager, as described in *Procedure 21.4, "Mounting an OCFS2 volume with the cluster resource manager"*.

To mount multiple OCFS2 volumes, see *Procedure 21.5, "Mounting multiple OCFS2 volumes with the cluster resource manager"*.

PROCEDURE 21.3: MANUALLY MOUNTING AN OCFS2 VOLUME

1. Open a terminal window and log in as `root`.

2. Check if the cluster is online with the command **`crm status`**.

3. Mount the volume from the command line, using the **`mount`** command.

Tip: Mounting an existing OCFS2 volume on a single node

You can mount an OCFS2 volume on a single node without a fully functional cluster stack: for example, to quickly access the data from a backup. To do so, use the **`mount`** command with the `-o nocluster` option.

> This mount method lacks cluster-wide protection. To avoid damaging the file system, you must ensure that it is only mounted on one node.

PROCEDURE 21.4: MOUNTING AN OCFS2 VOLUME WITH THE CLUSTER RESOURCE MANAGER

To mount an OCFS2 volume with the High Availability software, configure an OCFS2 file system resource in the cluster. The following procedure uses the **`crm`** shell to configure the cluster resources. Alternatively, you can also use Hawk2 to configure the resources as described in *Section 21.6, "Configuring OCFS2 resources with Hawk2"*.

1. Log in to a node as `root` or equivalent.

2. Run **`crm configure`**.

3. Configure Pacemaker to mount the OCFS2 file system on every node in the cluster:

```
crm(live)configure# primitive ocfs2-1 ocf:heartbeat:Filesystem \
  params device="/dev/disk/by-id/DEVICE_ID" directory="/mnt/shared" fstype="ocfs2" \
  op monitor interval="20" timeout="40" \
```

```
    op start timeout="60" op stop timeout="60" \
    meta target-role="Started"
```

4. Add the `ocfs2-1` primitive to the `g-storage` group you created in *Procedure 20.1, "Configuring a base group for DLM"*.

```
crm(live)configure# modgroup g-storage add ocfs2-1
```

Because of the base group's internal colocation and ordering, the `ocfs2-1` resource will only start on nodes that also have a `dlm` resource already running.

> **❗ Important: Do not use a group for multiple OCFS2 resources**
>
> Adding multiple OCFS2 resources to a group creates a dependency between the OCFS2 volumes. For example, if you created a group with **crm configure group g-storage dlm ocfs2-1 ocfs2-2**, then stopping `ocfs2-1` will also stop `ocfs2-2`, and starting `ocfs2-2` will also start `ocfs2-1`.
>
> To use multiple OCFS2 resources in the cluster, use colocation and order constraints as described in *Procedure 21.5, "Mounting multiple OCFS2 volumes with the cluster resource manager"*.

5. Review your changes with **show**.

6. If everything is correct, submit your changes with **commit** and leave the crm live configuration with **quit**.

PROCEDURE 21.5: MOUNTING MULTIPLE OCFS2 VOLUMES WITH THE CLUSTER RESOURCE MANAGER

To mount multiple OCFS2 volumes in the cluster, configure an OCFS2 file system resource for each volume, and colocate them with the `dlm` resource you created in *Procedure 20.2, "Configuring an independent DLM resource"*.

> **❗** Do *not* add multiple OCFS2 resources to a group with DLM. This creates a dependency between the OCFS2 volumes. For example, if `ocfs2-1` and `ocfs2-2` are in the same group, then stopping `ocfs2-1` will also stop `ocfs2-2`.

1. Log in to a node as `root` or equivalent.

2. Run **crm configure**.

3. Create the primitive for the first OCFS2 volume:

```
crm(live)configure# primitive ocfs2-1 Filesystem \
  params directory="/srv/ocfs2-1" fstype=ocfs2 device="/dev/disk/by-id/DEVICE_ID1" \
  op monitor interval=20 timeout=40 \
  op start timeout=60 interval=0 \
  op stop timeout=60 interval=0
```

4. Create the primitive for the second OCFS2 volume:

```
crm(live)configure# primitive ocfs2-2 Filesystem \
  params directory="/srv/ocfs2-2" fstype=ocfs2 device="/dev/disk/by-id/DEVICE_ID2" \
  op monitor interval=20 timeout=40 \
  op start timeout=60 interval=0 \
  op stop timeout=60 interval=0
```

5. Clone the OCFS2 resources so that they can run on all nodes:

```
crm(live)configure# clone cl-ocfs2-1 ocfs2-1 meta interleave=true
crm(live)configure# clone cl-ocfs2-2 ocfs2-2 meta interleave=true
```

6. Add a colocation constraint for both OCFS2 resources so that they can only run on nodes where DLM is also running:

```
crm(live)configure# colocation co-ocfs2-with-dlm inf: ( cl-ocfs2-1 cl-ocfs2-2 ) cl-dlm
```

7. Add an order constraint for both OCFS2 resources so that they can only start after DLM is already running:

```
crm(live)configure# order o-dlm-before-ocfs2 Mandatory: cl-dlm ( cl-ocfs2-1 cl-ocfs2-2 )
```

8. Review your changes with `show`.

9. If everything is correct, submit your changes with `commit` and leave the crm live configuration with `quit`.

21.6   Configuring OCFS2 resources with Hawk2

Instead of configuring the DLM and the file system resource for OCFS2 manually with the crm shell, you can also use the OCFS2 template in Hawk2's *Setup Wizard*.

! Important: Differences between manual configuration and Hawk2

The OCFS2 template in the *Setup Wizard* does *not* include the configuration of a STONITH resource. If you use the wizard, you still need to create an SBD partition on the shared storage and configure a STONITH resource as described in *Procedure 21.1, "Configuring a STONITH resource"*.

Using the OCFS2 template in the Hawk2 *Setup Wizard* also leads to a slightly different resource configuration than the manual configuration described in *Procedure 20.1, "Configuring a base group for DLM"* and *Procedure 21.4, "Mounting an OCFS2 volume with the cluster resource manager"*.

PROCEDURE 21.6: CONFIGURING OCFS2 RESOURCES WITH HAWK2'S *WIZARD*

1. Log in to Hawk2:

   ```
   https://HAWKSERVER:7630/
   ```

2. In the left navigation bar, select *Wizard*.

3. Expand the *File System* category and select `OCFS2 File System`.

4. Follow the instructions on the screen. If you need information about an option, click it to display a short help text in Hawk2. After the last configuration step, *Verify* the values you have entered.

   The wizard displays the configuration snippet that will be applied to the CIB and any additional changes, if required.
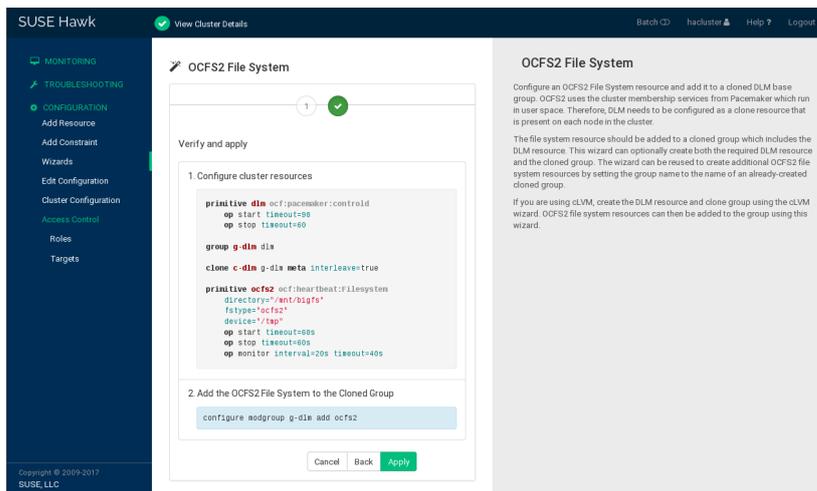
FIGURE 21.1: HAWK2 SUMMARY SCREEN OF OCFS2 CIB CHANGES

5. Check the proposed changes. If everything is according to your wishes, apply the changes. A message on the screen shows if the action has been successful.

21.7 Using quotas on OCFS2 file systems

To use quotas on an OCFS2 file system, create and mount the files system with the appropriate quota features or mount options, respectively: `ursquota` (quota for individual users) or `grpquota` (quota for groups). These features can also be enabled later on an unmounted file system using **`tunefs.ocfs2`**.

When a file system has the appropriate quota feature enabled, it tracks in its metadata how much space and files each user (or group) uses. Since OCFS2 treats quota information as file system-internal metadata, you do not need to run the `quotacheck` (8) program. All functionality is built into fsck.ocfs2 and the file system driver itself.

To enable enforcement of limits imposed on each user or group, run `quotaon` (8) like you would do for any other file system.

For performance reasons each cluster node performs quota accounting locally and synchronizes this information with a common central storage once per 10 seconds. This interval is tuneable with **`tunefs.ocfs2`**, options `usrquota-sync-interval` and `grpquota-sync-interval`. Therefore quota information may not be exact at all times and as a consequence users or groups can slightly exceed their quota limit when operating on several cluster nodes in parallel.

21.8 For more information

For more information about OCFS2, see the following links:

https://ocfs2.wiki.kernel.org/ ↗
>The OCFS2 project home page.

https://oss.oracle.com/projects/ocfs2/ ↗
>The former OCFS2 project home page at Oracle.

https://oss.oracle.com/projects/ocfs2/documentation ↗
>The project's former documentation home page.

22 GFS2

Global File System 2 or GFS2 is a shared disk file system for Linux computer clusters. GFS2 allows all nodes to have direct concurrent access to the same shared block storage. GFS2 has no disconnected operating-mode, and no client or server roles. All nodes in a GFS2 cluster function as peers. GFS2 supports up to 32 cluster nodes. Using GFS2 in a cluster requires hardware to allow access to the shared storage, and a lock manager to control access to the storage.

SUSE recommends OCFS2 over GFS2 for your cluster environments if performance is one of your major requirements. Our tests have revealed that OCFS2 performs better as compared to GFS2 in such settings.

! **Important: GFS2 support**

   SUSE only supports GFS2 in read-only mode. Write operations are not supported.

22.1 GFS2 packages and management utilities

To use GFS2, make sure `gfs2-utils` and a matching `gfs2-kmp-*` package for your Kernel is installed on each node of the cluster.

The `gfs2-utils` package provides the following utilities for management of GFS2 volumes. For syntax information, see their man pages.

fsck.gfs2

   Checks the file system for errors and optionally repairs errors.

gfs2_jadd

   Adds additional journals to a GFS2 file system.

gfs2_grow

   Grow a GFS2 file system.

mkfs.gfs2

   Create a GFS2 file system on a device, usually a shared device or partition.

tunegfs2

Allows viewing and manipulating the GFS2 file system parameters such as `UUID`, `label`, `lockproto` and `locktable`.

22.2 Configuring GFS2 services and a STONITH resource

Before you can create GFS2 volumes, you must configure DLM and a STONITH resource.

PROCEDURE 22.1: CONFIGURING A STONITH RESOURCE

> ### Note: STONITH device needed
>
> You need to configure a fencing device. Without a STONITH mechanism (like `external/sbd`) in place the configuration will fail.

1. Start a shell and log in as `root` or equivalent.

2. Create an SBD partition as described in *Procedure 13.3, "Initializing the SBD devices"*.

3. Run **`crm configure`**.

4. Configure `external/sbd` as the fencing device:

   ```
   crm(live)configure# primitive sbd_stonith stonith:external/sbd \
       params pcmk_delay_max=30 meta target-role="Started"
   ```

5. Review your changes with **`show`**.

6. If everything is correct, submit your changes with **`commit`** and leave the crm live configuration with **`quit`**.

For details on configuring the resource for DLM, see *Section 20.2, "Configuring DLM cluster resources"*.

22.3 Creating GFS2 volumes

After you have configured DLM as cluster resources as described in *Section 22.2, "Configuring GFS2 services and a STONITH resource"*, configure your system to use GFS2 and create GFS2 volumes.

✎ Note: GFS2 volumes for application and data files

We recommend that you generally store application files and data files on different GFS2 volumes. If your application volumes and data volumes have different requirements for mounting, it is mandatory to store them on different volumes.

Before you begin, prepare the block devices you plan to use for your GFS2 volumes. Leave the devices as free space.

Then create and format the GFS2 volume with the `mkfs.gfs2` as described in *Procedure 22.2, "Creating and formatting a GFS2 volume"*. The most important parameters for the command are listed below. For more information and the command syntax, refer to the `mkfs.gfs2` man page.

Lock Protocol Name (`-p`)

The name of the locking protocol to use. Acceptable locking protocols are lock_dlm (for shared storage) or, if you are using GFS2 as a local file system (1 node only), you can specify the lock_nolock protocol. If this option is not specified, lock_dlm protocol is assumed.

Lock Table Name (`-t`)

The lock table field appropriate to the lock module you are using. It is `clustername:fs-name`. The `clustername` value must match that in the cluster configuration file, `/etc/corosync/corosync.conf`. Only members of this cluster are permitted to use this file system. The `fsname` value is a unique file system name used to distinguish this GFS2 file system from others created (1 to 16 characters).

Number of Journals (`-j`)

The number of journals for gfs2_mkfs to create. You need at least one journal per machine that will mount the file system. If this option is not specified, one journal is created.

PROCEDURE 22.2: CREATING AND FORMATTING A GFS2 VOLUME

Execute the following steps only on *one* of the cluster nodes.

1. Open a terminal window and log in as `root`.

2. Check if the cluster is online with the command `crm status`.

3. Create and format the volume using the `mkfs.gfs2` utility. For information about the syntax for this command, refer to the `mkfs.gfs2` man page.

For example, to create a new GFS2 file system that supports up to 32 cluster nodes, use the following command:

```
# mkfs.gfs2 -t hacluster:mygfs2 -p lock_dlm -j 32 /dev/disk/by-id/DEVICE_ID
```

The `hacluster` name relates to the entry `cluster_name` in the file `/etc/corosync/corosync.conf` (this is the default).

Always use a stable device name (for example: `/dev/disk/by-id/scsi-ST2000D-M001-0123456_Wabcdefg`).

22.4 Mounting GFS2 volumes

You can either mount a GFS2 volume manually or with the cluster manager, as described in *Procedure 22.4, "Mounting a GFS2 volume with the cluster manager"*.

PROCEDURE 22.3: MANUALLY MOUNTING A GFS2 VOLUME

1. Open a terminal window and log in as `root`.

2. Check if the cluster is online with the command **`crm status`**.

3. Mount the volume from the command line, using the **`mount`** command.

> ✋ Warning: Manually mounted GFS2 devices
>
> If you mount the GFS2 file system manually for testing purposes, make sure to unmount it again before starting to use it via cluster resources.

PROCEDURE 22.4: MOUNTING A GFS2 VOLUME WITH THE CLUSTER MANAGER

To mount a GFS2 volume with the High Availability software, configure an OCF file system resource in the cluster. The following procedure uses the **`crm`** shell to configure the cluster resources. Alternatively, you can also use Hawk2 to configure the resources.

1. Start a shell and log in as `root` or equivalent.

2. Run **`crm configure`**.

3. Configure Pacemaker to mount the GFS2 file system on every node in the cluster:

```
crm(live)configure# primitive gfs2-1 ocf:heartbeat:Filesystem \
```

```
params device="/dev/disk/by-id/DEVICE_ID" directory="/mnt/shared" fstype="gfs2" \
op monitor interval="20" timeout="40" \
op start timeout="60" op stop timeout="60" \
meta target-role="Stopped"
```

4. Add the `gfs2-1` primitive to the `g-storage` group you created in *Procedure 20.1, "Configuring a base group for DLM"*.

```
crm(live)configure# modgroup g-storage add gfs2-1
```

Because of the base group's internal colocation and ordering, the `gfs2-1` resource will only start on nodes that also have a `dlm` resource already running.

5. Review your changes with **show**.

6. If everything is correct, submit your changes with **commit** and leave the crm live configuration with **quit**.

23 DRBD

The *distributed replicated block device* (DRBD*) allows you to create a mirror of two block devices that are located at two different sites across an IP network. When used with Corosync, DRBD supports distributed high-availability Linux clusters. This chapter shows you how to install and set up DRBD.

23.1 Conceptual overview

DRBD replicates data on the primary device to the secondary device in a way that ensures that both copies of the data remain identical. Think of it as a networked RAID 1. It mirrors data in real-time, so its replication occurs continuously. Applications do not need to know that in fact their data is stored on different disks.

DRBD is a Linux Kernel module and sits between the I/O scheduler at the lower end and the file system at the upper end, see *Figure 23.1, "Position of DRBD within Linux"*. To communicate with DRBD, users use the high-level command `drbdadm`. For maximum flexibility DRBD comes with the low-level tool `drbdsetup`.
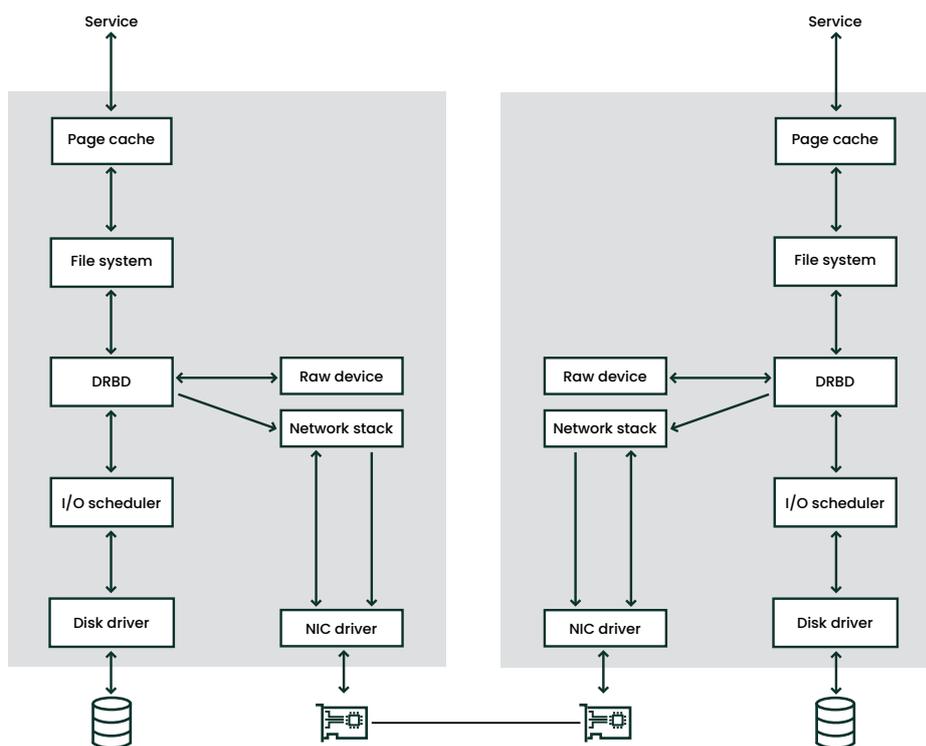
FIGURE 23.1: **POSITION OF DRBD WITHIN LINUX**

> **❗ Important: Unencrypted data**
>
> The data traffic between mirrors is not encrypted. For secure data exchange, you should deploy a Virtual Private Network (VPN) solution for the connection.

DRBD allows you to use any block device supported by Linux, usually:

- partition or complete hard disk

- software RAID

- Logical Volume Manager (LVM)

- Enterprise Volume Management System (EVMS)

By default, DRBD uses the TCP ports 7788 and higher for communication between DRBD nodes. Make sure that your firewall does not prevent communication on the used ports.

You must set up the DRBD devices before creating file systems on them. Everything pertaining to user data should be done solely via the `/dev/drbdN` device and not on the raw device, as DRBD uses the last part of the raw device for metadata. Using the raw device will cause inconsistent data.

With udev integration, you will also get symbolic links in the form `/dev/drbd/by-res/RESOURCES` which are easier to use and provide safety against remembering the wrong minor number of the device.

For example, if the raw device is 1024 MB in size, the DRBD device has only 1023 MB available for data, with about 70 KB hidden and reserved for the metadata. Any attempt to access the remaining kilobytes via raw disks fails because it is not available for user data.

23.2   Installing DRBD services

Install the High Availability pattern on both SUSE Linux Enterprise Server machines in your networked cluster as described in *Part I, "Installation and setup"*. Installing the pattern also installs the DRBD program files.

If you do not need the complete cluster stack but only want to use DRBD, install the packages `drbd`, `drbd-kmp-FLAVOR`, `drbd-utils`, and `yast2-drbd`.

23.3   Setting up the DRBD service

Note: Adjustments needed

The following procedures use the server names alice and bob, and the DRBD resource name `r0`. It sets up alice as the primary node and `/dev/disk/by-id/example-disk1` for storage. Make sure to modify the instructions to use your own nodes and file names.

The following procedures assume that the cluster nodes use the TCP port `7788`. Make sure this port is open in your firewall.

To set up DRBD, perform the following procedures:

1. Section 23.3.1, "Preparing your system to use DRBD"

2. Configure DRBD using one of the following methods:

   - Section 23.3.2, "Configuring DRBD manually"

   - Section 23.3.3, "Configuring DRBD with YaST"

3. Section 23.3.4, "Initializing and formatting DRBD resources"

4. Section 23.3.5, "Creating cluster resources for DRBD"

23.3.1   Preparing your system to use DRBD

Before you start configuring DRBD, you might need to perform some or all of the following steps:

PROCEDURE 23.1: PREPARING YOUR SYSTEM TO USE DRBD

1. Make sure the block devices in your Linux nodes are ready and partitioned (if needed).

2. If your disk already contains a file system that you do not need anymore, destroy the file system structure with the following command:

   ```
   # dd if=/dev/zero of=YOUR_DEVICE count=16 bs=1M
   ```

   If you have more file systems to destroy, repeat this step on all devices you want to include into your DRBD setup.

3. If the cluster is already using DRBD, put your cluster in maintenance mode:

   ```
   # crm maintenance on
   ```

   If you skip this step when your cluster already uses DRBD, a syntax error in the live configuration leads to a service shutdown. Alternatively, you can also use **drbdadm -c** *FILE* to test a configuration file.

23.3.2 Configuring DRBD manually

> 📝 **Note: Restricted support for "auto promote" feature**
>
> The DRBD9 "auto-promote" feature can automatically promote a resource to the primary role when one of its devices is mounted or opened for writing.
>
> The auto promote feature has currently restricted support. With DRBD 9, SUSE supports the same use cases that were also supported with DRBD 8. Use cases beyond that, such as setups with more than two nodes, are not supported.

To set up DRBD manually, proceed as follows:

PROCEDURE 23.2: MANUALLY CONFIGURING DRBD

Beginning with DRBD version 8.3, the former configuration file is split into separate files, located under the directory `/etc/drbd.d/`.

1. Open the file `/etc/drbd.d/global_common.conf`. It already contains global, pre-defined values. Go to the `startup` section and insert these lines:

   ```
   startup {
       # wfc-timeout degr-wfc-timeout outdated-wfc-timeout
       # wait-after-sb;
       wfc-timeout 100;
       degr-wfc-timeout 120;
   }
   ```

   These options are used to reduce the timeouts when booting, see https://docs.lin-bit.com/docs/users-guide-9.0/#ch-configure ↗ for more details.

2. Create the file `/etc/drbd.d/r0.res`. Change the lines according to your situation and save it:

   ```
   resource r0 {  ❶
     device /dev/drbd0;  ❷
     disk /dev/disk/by-id/example-disk1;  ❸
     meta-disk internal;  ❹
     on alice {  ❺
       address  192.168.1.10:7788;  ❻
       node-id 0;  ❼
     }
     on bob {  ❺
       address 192.168.1.11:7788;  ❻
   ```

```
      node-id 1;  ➐
    }
    disk {
      resync-rate 10M;  ➑
    }
    connection-mesh {  ➒
      hosts alice bob;
    }
    net {
      protocol C;  ➓
      fencing resource-and-stonith;  ⑪
     }
    handlers {  ⑫
      fence-peer "/usr/lib/drbd/crm-fence-peer.9.sh";
      after-resync-target "/usr/lib/drbd/crm-unfence-peer.9.sh";
     }
}
```

➊ DRBD resource name that allows some association to the service that needs them.
  For example, `nfs`, `http`, `mysql_0`, `postgres_wal`, etc. Here a more general name
  `r0` is used.

➋ The device name for DRBD and its minor number.
  In the example above, the minor number 0 is used for DRBD. The udev integration
  scripts will give you a symbolic link `/dev/drbd/by-res/nfs/0`. Alternatively, omit
  the device node name in the configuration and use the following line instead:
  `drbd0 minor 0` (`/dev/` is optional) or `/dev/drbd0`

➌ The raw device that is replicated between nodes. Note, in this example the devices
  are the *same* on both nodes. If you need different devices, move the `disk` parameter
  into the `on` host.

➍ The meta-disk parameter usually contains the value `internal`, but it is possible to
  specify an explicit device to hold the metadata. See https://docs.linbit.com/docs/users-
  guide-9.0/#s-metadata ↗ for more information.

➎ The `on` section states which host this configuration statement applies to.

➏ The IP address and port number of the respective node. Each resource needs an in-
  dividual port, usually starting with `7788`. Both ports must be the same for a DRBD
  resource.

➐ The node ID is required when configuring more than two nodes. It is a unique, non-
  negative integer to distinguish the different nodes.

⑧ The synchronization rate. Set it to one third of the lower of the disk- and network bandwidth. It only limits the resynchronization, not the replication.

⑨ Defines all nodes of a mesh. The `hosts` parameter contains all host names that share the same DRBD setup.

⑩ The protocol to use for this connection. Protocol `C` is the default option. It provides better data availability and does not consider a write to be complete until it has reached all local and remote disks.

⑪ Specifies the fencing policy `resource-and-stonith` at the DRBD level. This policy immediately suspends active I/O operations until STONITH completes.

⑫ Enables resource-level fencing to prevent Pacemaker from starting a service with outdated data. If the DRBD replication link becomes disconnected, the **crm-fence-peer.9.sh** script stops the DRBD resource from being promoted to another node until the replication link becomes connected again and DRBD completes its synchronization process.

3. Check the syntax of your configuration files. If the following command returns an error, verify your files:

```
# drbdadm dump all
```

4. Copy the DRBD configuration files to all nodes:

```
# csync2 -xv
```

By default, the DRBD configuration file `/etc/drbd.conf` and the directory `/etc/drbd.d/` are already included in the list of files that Csync2 synchronizes.

23.3.3 Configuring DRBD with YaST

YaST can be used to start with an initial setup of DRBD. After you have created your DRBD setup, you can fine-tune the generated files manually.

However, when you have changed the configuration files, do not use the YaST DRBD module anymore. The DRBD module supports only a limited set of basic configuration. If you use it again, it is likely that the module will not show your changes.

To set up DRBD with YaST, proceed as follows:

1. Start YaST and select the configuration module *High Availability* › *DRBD*. If you already have a DRBD configuration, YaST warns you. YaST will change your configuration and save your old DRBD configuration files as `*.YaSTsave`.

2. Leave the booting flag in *Start-up Configuration* › *Booting* as it is (by default it is `off`); do not change that as Pacemaker manages this service.

3. If you have a firewall running, enable *Open Port in Firewall*.

4. Go to the *Resource Configuration* entry. Select *Add* to create a new resource (see *Figure 23.2, "Resource configuration"*).
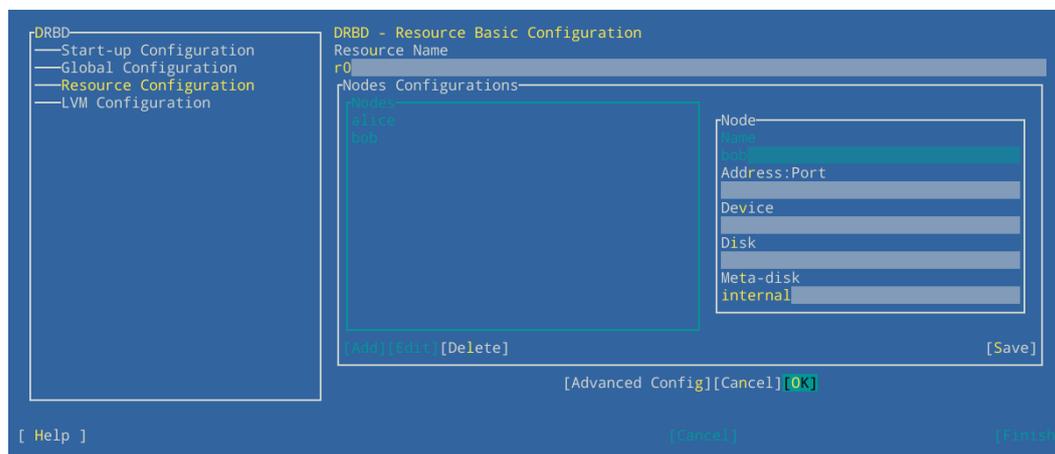
FIGURE 23.2: RESOURCE CONFIGURATION

The following parameters need to be set:

Resource Name
> The name of the DRBD resource (mandatory)

Name
> The host name of the relevant node

Address:Port
> The IP address and port number (default `7788`) for the respective node

Device

> The block device path that is used to access the replicated data. If the device contains a minor number, the associated block device is usually named `/dev/drbdX`, where `X` is the device minor number. If the device does not contain a minor number, make sure to add `minor 0` after the device name.

Disk

> The raw device that is replicated between both nodes. If you use LVM, insert your LVM device name.

Meta-disk

> The *Meta-disk* is either set to the value `internal` or specifies an explicit device extended by an index to hold the metadata needed by DRBD.
>
> A real device may also be used for multiple DRBD resources. For example, if your *Meta-Disk* is `/dev/disk/by-id/example-disk6[0]` for the first resource, you may use `/dev/disk/by-id/example-disk6[1]` for the second resource. However, there must be at least 128 MB space for each resource available on this disk. The fixed metadata size limits the maximum data size that you can replicate.

All these options are explained in the examples in the `/usr/share/doc/packages/drbd/drbd.conf` file and in the man page of **drbd.conf(5)**.

5. Click *Save*.

6. Click *Add* to enter the second DRBD resource and finish with *Save*.

7. Close the resource configuration with *Ok* and *Finish*.

8. If you use LVM with DRBD, it is necessary to change some options in the LVM configuration file (see the *LVM Configuration* entry). This change can be done by the YaST DRBD module automatically.

   The disk name of localhost for the DRBD resource and the default filter will be rejected in the LVM filter. Only `/dev/drbd` can be scanned for an LVM device.

   For example, if `/dev/disk/by-id/example-disk1` is used as a DRBD disk, the device name will be inserted as the first entry in the LVM filter. To change the filter manually, click the *Modify LVM Device Filter Automatically* check box.

9. Save your changes with *Finish*.

10. Copy the DRBD configuration files to all nodes:

```
# csync2 -xv
```

By default, the DRBD configuration file `/etc/drbd.conf` and the directory `/etc/drbd.d/` are already included in the list of files that Csync2 synchronizes.

23.3.4 Initializing and formatting DRBD resources

After you have prepared your system and configured DRBD, initialize your disk for the first time:

1. On *both* nodes (alice and bob), initialize the metadata storage:

```
# drbdadm create-md r0
# drbdadm up r0
```

2. To shorten the initial resynchronization of your DRBD resource check the following:

   - If the DRBD devices on all nodes have the same data (for example, by destroying the file system structure with the **dd** command as shown in *Section 23.3, "Setting up the DRBD service"*), then skip the initial resynchronization with the following command (on both nodes):

   ```
   # drbdadm new-current-uuid --clear-bitmap r0/0
   ```

   The state will be `Secondary/Secondary UpToDate/UpToDate`

   - Otherwise, proceed with the next step.

3. On the primary node alice, start the resynchronization process:

```
# drbdadm primary --force r0
```

4. Check the status with:

```
# drbdadm status r0
r0 role:Primary
  disk:UpToDate
  bob role:Secondary
  peer-disk:UpToDate
```

5. Create your file system on top of your DRBD device, for example:

```
# mkfs.ext3 /dev/drbd0
```

6. Mount the file system and use it:

```
# mount /dev/drbd0 /mnt/
```

23.3.5   Creating cluster resources for DRBD

After you have initialized your DRBD device, create a cluster resource to manage the DRBD device, and a promotable clone to allow this resource to run on both nodes:

PROCEDURE 23.4: CREATING CLUSTER RESOURCES FOR DRBD

1. Start the `crm` interactive shell:

```
# crm configure
```

2. Create a primitive for the DRBD resource `r0`:

```
crm(live)configure# primitive drbd-r0 ocf:linbit:drbd \
  params drbd_resource="r0" \
  op monitor interval=15 role=Promoted \
  op monitor interval=30 role=Unpromoted
```

3. Create a promotable clone for the `drbd-r0` primitive:

```
crm(live)configure# clone cl-drbd-r0 drbd-r0 \
  meta promotable="true" promoted-max="1" promoted-node-max="1" \
  clone-max="2" clone-node-max="1" notify="true" interleave=true
```

4. Commit this configuration:

```
crm(live)configure# commit
```

5. Exit the interactive shell:

```
crm(live)configure# quit
```

If you put the cluster in maintenance mode before configuring DRBD, you can now move it back to normal operation with `crm maintenance off`.

23.4 Creating a stacked DRBD device

A stacked DRBD device contains two other devices of which at least one device is also a DRBD resource. In other words, DRBD adds an additional node on top of an already existing DRBD resource (see *Figure 23.3, "Resource stacking"*). Such a replication setup can be used for backup and disaster recovery purposes.
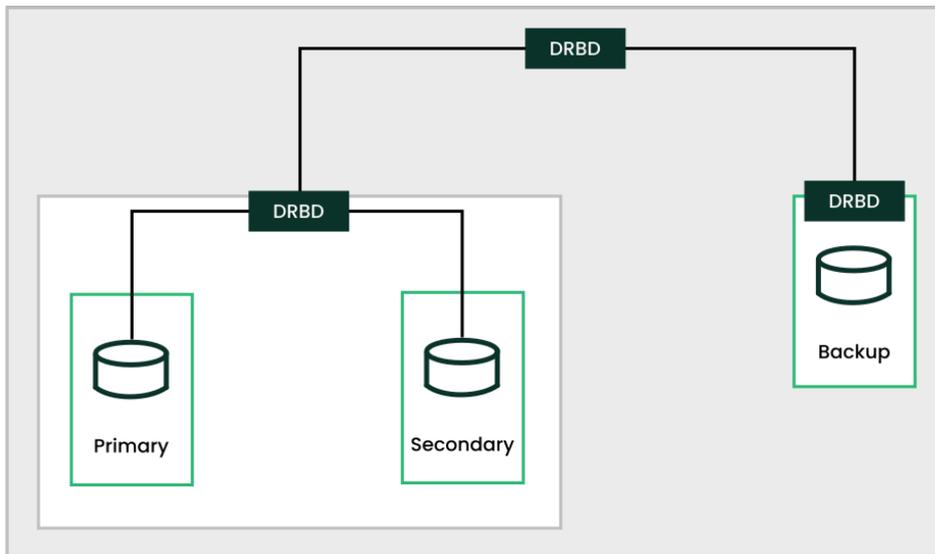
FIGURE 23.3: **RESOURCE STACKING**

Three-way replication uses asynchronous (DRBD protocol A) and synchronous replication (DRBD protocol C). The asynchronous part is used for the stacked resource whereas the synchronous part is used for the backup.

Your production environment uses the stacked device. For example, if you have a DRBD device `/dev/drbd0` and a stacked device `/dev/drbd10` on top, the file system will be created on `/dev/drbd10`, see *Example 23.1, "Configuration of a three-node stacked DRBD resource"* for more details.

EXAMPLE 23.1: **CONFIGURATION OF A THREE-NODE STACKED DRBD RESOURCE**

```
# /etc/drbd.d/r0.res
resource r0 {
  protocol C;
  device    /dev/drbd0;
  disk      /dev/disk/by-id/example-disk1;
  meta-disk internal;

  on amsterdam-alice {
```

```
    address     192.168.1.1:7900;
  }

  on amsterdam-bob {
    address     192.168.1.2:7900;
  }
}

resource r0-U {
  protocol A;
  device     /dev/drbd10;

  stacked-on-top-of r0 {
    address     192.168.2.1:7910;
  }

  on berlin-charlie {
    disk       /dev/disk/by-id/example-disk10;
    address     192.168.2.2:7910; # Public IP of the backup node
    meta-disk  internal;
  }
}
```

23.5   Testing the DRBD service

If the install and configuration procedures worked as expected, you are ready to run a basic test of the DRBD functionality. This test also helps with understanding how the software works.

1. Test the DRBD service on alice.

   a. Open a terminal console, then log in as `root`.

   b. Create a mount point on alice, such as `/srv/r0`:

   ```
   # mkdir -p /srv/r0
   ```

   c. Mount the **drbd** device:

   ```
   # mount -o rw /dev/drbd0 /srv/r0
   ```

   d. Create a file from the primary node:

   ```
   # touch /srv/r0/from_alice
   ```

e. Unmount the disk on alice:

```
# umount /srv/r0
```

f. Downgrade the DRBD service on alice by typing the following command on alice:

```
# drbdadm secondary r0
```

2. Test the DRBD service on bob.

a. Open a terminal console, then log in as `root` on bob.

b. On bob, promote the DRBD service to primary:

```
# drbdadm primary r0
```

c. On bob, check to see if bob is primary:

```
# drbdadm status r0
```

d. On bob, create a mount point such as `/srv/r0`:

```
# mkdir /srv/r0
```

e. On bob, mount the DRBD device:

```
# mount -o rw /dev/drbd0 /srv/r0
```

f. Verify that the file you created on alice exists:

```
# ls /srv/r0/from_alice
```

The `/srv/r0/from_alice` file should be listed.

3. If the service is working on both nodes, the DRBD setup is complete.

4. Set up alice as the primary again.

a. Dismount the disk on bob by typing the following command on bob:

```
# umount /srv/r0
```

b. Downgrade the DRBD service on bob by typing the following command on bob:

```
# drbdadm secondary r0
```

c. On alice, promote the DRBD service to primary:

```
# drbdadm primary r0
```

d. On alice, check to see if alice is primary:

```
# drbdadm status r0
```

5. To get the service to automatically start and fail over if the server has a problem, you can set up DRBD as a high availability service with Pacemaker/Corosync. For information about installing and configuring for SUSE Linux Enterprise 15 SP5 see *Part II, "Configuration and administration"*.

23.6 Monitoring DRBD devices

DRBD comes with the utility `drbdmon` which offers real time monitoring. It shows all the configured resources and their problems.

FIGURE 23.4: SHOWING A GOOD CONNECTION BY `drbdmon`

In case of problems, `drbdadm` shows an error message:

FIGURE 23.5: SHOWING A BAD CONNECTION BY `drbdmon`

23.7 Tuning DRBD

There are several ways to tune DRBD:

1. Use an external disk for your metadata. This might help, at the cost of maintenance ease.

2. Tune your network connection, by changing the receive and send buffer settings via `sysctl`.

3. Change the `max-buffers`, `max-epoch-size` or both in the DRBD configuration.

4. Increase the `al-extents` value, depending on your IO patterns.

5. If you have a hardware RAID controller with a BBU (*Battery Backup Unit*), you might benefit from setting `no-disk-flushes`, `no-disk-barrier` and/or `no-md-flushes`.

6. Enable read-balancing depending on your workload. See https://www.linbit.com/en/read-balancing/↗ for more details.

23.8 Troubleshooting DRBD

The DRBD setup involves many components and problems may arise from different sources. The following sections cover several common scenarios and recommend various solutions.

23.8.1 Configuration

If the initial DRBD setup does not work as expected, there is probably something wrong with your configuration.

To get information about the configuration:

1. Open a terminal console, then log in as `root`.

2. Test the configuration file by running **drbdadm** with the **-d** option. Enter the following command:

   ```
   # drbdadm -d adjust r0
   ```

   In a dry run of the **adjust** option, **drbdadm** compares the actual configuration of the DRBD resource with your DRBD configuration file, but it does not execute the calls. Review the output to make sure you know the source and cause of any errors.

3. If there are errors in the `/etc/drbd.d/*` and `drbd.conf` files, correct them before continuing.

4. If the partitions and settings are correct, run **drbdadm** again without the **-d** option.

   ```
   # drbdadm adjust r0
   ```

   This applies the configuration file to the DRBD resource.

23.8.2   Host names

For DRBD, host names are case-sensitive (`Node0` would be a different host than `node0`), and compared to the host name as stored in the Kernel (see the **uname -n** output).

If you have several network devices and want to use a dedicated network device, the host name will likely not resolve to the used IP address. In this case, use the parameter `disable-ip-verification`.

23.8.3   TCP port 7788

If your system cannot connect to the peer, this might be a problem with your local firewall. By default, DRBD uses the TCP port `7788` to access the other node. Make sure that this port is accessible on both nodes.

23.8.4   DRBD devices broken after reboot

In cases when DRBD does not know which of the real devices holds the latest data, it changes to a split-brain condition. In this case, the respective DRBD subsystems come up as secondary and do not connect to each other. In this case, the following message can be found in the logging data:

```
Split-Brain detected, dropping connection!
```

To resolve this situation, enter the following commands on the node which has data to be discarded:

```
# drbdadm disconnect r0
# drbdadm secondary r0
# drbdadm connect --discard-my-data r0
```

On the node which has the latest data, enter the following commands:

```
# drbdadm disconnect r0
# drbdadm connect r0
```

That resolves the issue by overwriting one node's data with the peer's data, therefore getting a consistent view on both nodes.

23.9 For more information

The following open source resources are available for DRBD:

- The project home page https://www.drbd.org ↗.

- See *Article "Highly Available NFS Storage with DRBD and Pacemaker"*.

- https://clusterlabs.org/wiki/DRBD_HowTo_1.0 ↗ by the Linux Pacemaker Cluster Stack Project.

- The following man pages for DRBD are available in the distribution: `drbd(8)`, `drbd-meta(8)`, `drbdsetup(8)`, `drbdadm(8)`, `drbd.conf(5)`.

- Find a commented example configuration for DRBD at `/usr/share/doc/packages/drbd-utils/drbd.conf.example`.

- Furthermore, for easier storage administration across your cluster, see the recent announcement about the *DRBD-Manager* at https://www.linbit.com/en/drbd-manager/ ↗.

24 Cluster logical volume manager (Cluster LVM)

When managing shared storage on a cluster, every node must be informed about changes to the storage subsystem. Logical Volume Manager (LVM) supports transparent management of volume groups across the whole cluster. Volume groups shared among multiple hosts can be managed using the same commands as local storage.

24.1 Conceptual overview

Cluster LVM is coordinated with different tools:

Distributed lock manager (DLM)

Coordinates access to shared resources among multiple hosts through cluster-wide locking.

Logical Volume Manager (LVM)

LVM provides a virtual pool of disk space and enables flexible distribution of one logical volume over several disks.

Cluster logical volume manager (Cluster LVM)

The term `Cluster LVM` indicates that LVM is being used in a cluster environment. This needs some configuration adjustments to protect the LVM metadata on shared storage. From SUSE Linux Enterprise 15 onward, the cluster extension uses lvmlockd, which replaces clvmd. For more information about lvmlockd, see the man page of the `lvmlockd` command (`man 8 lvmlockd`).

> **!** lvmlockd with sanlock is not officially supported.

Volume group and logical volume

Volume groups (VGs) and logical volumes (LVs) are basic concepts of LVM. A volume group is a storage pool of multiple physical disks. A logical volume belongs to a volume group, and can be seen as an elastic volume on which you can create a file system. In a cluster environment, there is a concept of shared VGs, which consist of shared storage and can be used concurrently by multiple hosts.

24.2 Configuration of Cluster LVM

Make sure the following requirements are fulfilled:

- A shared storage device is available, provided by a Fibre Channel, FCoE, SCSI, iSCSI SAN, or DRBD*, for example.

- Make sure the following packages have been installed: `lvm2` and `lvm2-lockd`.

- From SUSE Linux Enterprise 15 onward, the cluster extension uses lvmlockd, which replaces clvmd. Make sure the clvmd daemon is not running, otherwise lvmlockd will fail to start.

24.2.1 Creating the cluster resources

Perform the following basic steps on one node to configure a shared VG in the cluster:

- *Creating a DLM resource*

- *Creating an lvmlockd resource*

- *Creating a shared VG and LV*

- *Creating an LVM-activate resource*

PROCEDURE 24.1: CREATING A DLM RESOURCE

1. Start a shell and log in as `root`.

2. Check the current configuration of the cluster resources:

   ```
   # crm configure show
   ```

3. If you have already configured a DLM resource (and a corresponding base group and base clone), continue with *Procedure 24.2, "Creating an lvmlockd resource"*.
   Otherwise, configure a DLM resource and a corresponding base group and base clone as described in *Procedure 20.1, "Configuring a base group for DLM"*.

PROCEDURE 24.2: CREATING AN LVMLOCKD RESOURCE

1. Start a shell and log in as `root`.

2. Run the following command to see the usage of this resource:

   ```
   # crm configure ra info lvmlockd
   ```

3. Configure a `lvmlockd` resource as follows:

```
# crm configure primitive lvmlockd lvmlockd \
        op start timeout="90" \
        op stop timeout="100" \
        op monitor interval="30" timeout="90"
```

4. To ensure the `lvmlockd` resource is started on every node, add the primitive resource to the base group for storage you have created in *Procedure 24.1, "Creating a DLM resource"*:

```
# crm configure modgroup g-storage add lvmlockd
```

5. Review your changes:

```
# crm configure show
```

6. Check if the resources are running well:

```
# crm status full
```

PROCEDURE 24.3: CREATING A SHARED VG AND LV

1. Start a shell and log in as `root`.

2. Assuming you already have two shared disks, create a shared VG with them:

```
# vgcreate --shared vg1 /dev/disk/by-id/DEVICE_ID1 /dev/disk/by-id/DEVICE_ID2
```

3. Create an LV and do not activate it initially:

```
# lvcreate -an -L10G -n lv1 vg1
```

PROCEDURE 24.4: CREATING AN LVM-ACTIVATE RESOURCE

1. Start a shell and log in as `root`.

2. Run the following command to see the usage of this resource:

```
# crm configure ra info LVM-activate
```

This resource manages the activation of a VG. In a shared VG, LV activation has two different modes: exclusive and shared mode. The exclusive mode is the default and should be used normally, when a local file system like `ext4` uses the LV. The shared mode should only be used for cluster file systems like OCFS2.

3. Configure a resource to manage the activation of your VG. Choose one of the following options according to your scenario:

- Use exclusive activation mode for local file system usage:

```
# crm configure primitive vg1 LVM-activate \
    params vgname=vg1 vg_access_mode=lvmlockd \
    op start timeout=90s interval=0 \
    op stop timeout=90s interval=0 \
    op monitor interval=30s timeout=90s
```

- Use shared activation mode for OCFS2:

```
# crm configure primitive vg1 LVM-activate \
    params vgname=vg1 vg_access_mode=lvmlockd activation_mode=shared \
    op start timeout=90s interval=0 \
    op stop timeout=90s interval=0 \
    op monitor interval=30s timeout=90s
```

4. Make sure the VG can only be activated on nodes where the DLM and `lvmlockd` resources are already running:

- **Exclusive activation mode:**
  Because this VG is only active on a single node, do *not* add it to the cloned `g-storage` group. Instead, add constraints directly to the resource:

  ```
  # crm configure colocation col-vg-with-dlm inf: vg1 cl-storage
  # crm configure order o-dlm-before-vg Mandatory: cl-storage vg1
  ```

  For multiple VGs, you can add constraints to multiple resources at once:

  ```
  # crm configure colocation col-vg-with-dlm inf: ( vg1 vg2 ) cl-storage
  # crm configure order o-dlm-before-vg Mandatory: cl-storage ( vg1 vg2 )
  ```

- **Shared activation mode:**
  Because this VG is active on multiple nodes, you can add it to the cloned `g-storage` group, which already has internal colocation and order constraints:

  ```
  # crm configure modgroup g-storage add vg1
  ```

  Do *not* add multiple VGs to the group, because this creates a dependency between the VGs. For multiple VGs, clone the resources and add constraints to the clones:

  ```
  # crm configure clone cl-vg1 vg1 meta interleave=true
  ```

```
# crm configure clone cl-vg2 vg2 meta interleave=true
# crm configure colocation col-vg-with-dlm inf: ( cl-vg1 cl-vg2 ) cl-storage
# crm configure order o-dlm-before-vg Mandatory: cl-storage ( cl-vg1 cl-vg2 )
```

5. Check if the resources are running well:

```
# crm status full
```

24.2.2 Scenario: Cluster LVM with iSCSI on SANs

The following scenario uses two SAN boxes which export their iSCSI targets to several clients. The general idea is displayed in *Figure 24.1, "Setup of a shared disk with Cluster LVM"*.
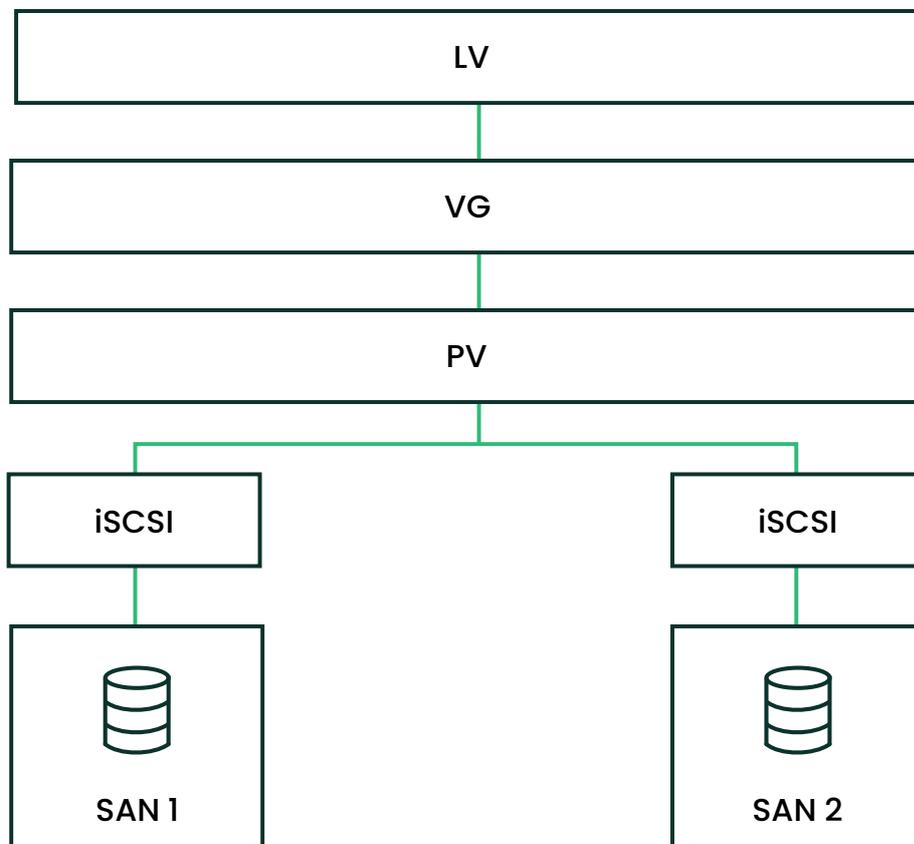
FIGURE 24.1: SETUP OF A SHARED DISK WITH CLUSTER LVM

✋ Warning: Data loss

The following procedures will destroy any data on your disks.

Configure only one SAN box first. Each SAN box needs to export its own iSCSI target. Proceed as follows:

1. Run YaST and click *Network Services* › *iSCSI LIO Target* to start the iSCSI Server module.

2. If you want to start the iSCSI target whenever your computer is booted, choose *When Booting,* otherwise choose *Manually*.

3. If you have a firewall running, enable *Open Port in Firewall*.

4. Switch to the *Global* tab. If you need authentication, enable incoming or outgoing authentication or both. In this example, we select *No Authentication*.

5. Add a new iSCSI target:

   a. Switch to the *Targets* tab.

   b. Click *Add*.

   c. Enter a target name. The name needs to be formatted like this:

      ```
      iqn.DATE.DOMAIN
      ```

      For more information about the format, refer to *Section 3.2.6.3.1. Type "iqn." (iSCSI Qualified Name)* at https://www.ietf.org/rfc/rfc3720.txt ↗.

   d. If you want a more descriptive name, you can change it if your identifier is unique for your different targets.

   e. Click *Add*.

   f. Enter the device name in *Path* and use a *Scsiid*.

   g. Click *Next* twice.

6. Confirm the warning box with *Yes*.

7. Open the configuration file `/etc/iscsi/iscsid.conf` and change the parameter `node.startup` to `automatic`.

Now set up your iSCSI initiators as follows:

1. Run YaST and click *Network Services* › *iSCSI Initiator*.

2. If you want to start the iSCSI initiator whenever your computer is booted, choose *When Booting*, otherwise set *Manually*.

3. Change to the *Discovery* tab and click the *Discovery* button.

4. Add the IP address and the port of your iSCSI target (see *Procedure 24.5, "Configuring iSCSI targets (SAN)"*). Normally, you can leave the port as it is and use the default value.

5. If you use authentication, insert the incoming and outgoing user name and password, otherwise activate *No Authentication*.

6. Select *Next*. The found connections are displayed in the list.

7. Proceed with *Finish*.

8. Open a shell, log in as `root`.

9. Test if the iSCSI initiator has been started successfully:

   ```
   # iscsiadm -m discovery -t st -p 192.168.3.100
   192.168.3.100:3260,1 iqn.2010-03.de.jupiter:san1
   ```

10. Establish a session:

    ```
    # iscsiadm -m node -l -p 192.168.3.100 -T iqn.2010-03.de.jupiter:san1
    Logging in to [iface: default, target: iqn.2010-03.de.jupiter:san1, portal:
     192.168.3.100,3260]
    Login to [iface: default, target: iqn.2010-03.de.jupiter:san1, portal:
     192.168.3.100,3260]: successful
    ```

    See the device names with `lsscsi`:

    ```
    ...
    [4:0:0:2]    disk    IET    ...    0    /dev/sdd
    [5:0:0:1]    disk    IET    ...    0    /dev/sde
    ```

    Look for entries with `IET` in their third column. In this case, the devices are `/dev/sdd` and `/dev/sde`.

1. Open a `root` shell on one of the nodes you have run the iSCSI initiator from *Procedure 24.6, "Configuring iSCSI initiators"*.

2. Create the shared volume group on disks `/dev/sdd` and `/dev/sde`, using their stable device names (for example, in `/dev/disk/by-id/`):

   ```
   # vgcreate --shared testvg /dev/disk/by-id/DEVICE_ID /dev/disk/by-id/DEVICE_ID
   ```

3. Create logical volumes as needed:

   ```
   # lvcreate --name lv1 --size 500M testvg
   ```

4. Check the volume group with **vgdisplay**:

   ```
   --- Volume group ---
       VG Name               testvg
       System ID
       Format                lvm2
       Metadata Areas        2
       Metadata Sequence No  1
       VG Access             read/write
       VG Status             resizable
       MAX LV                0
       Cur LV                0
       Open LV               0
       Max PV                0
       Cur PV                2
       Act PV                2
       VG Size               1016,00 MB
       PE Size               4,00 MB
       Total PE              254
       Alloc PE / Size       0 / 0
       Free  PE / Size       254 / 1016,00 MB
       VG UUID               UCyWw8-2jqV-enuT-KH4d-NXQI-JhH3-J24anD
   ```

5. Check the shared state of the volume group with the command `vgs`:

   ```
   # vgs
     VG        #PV #LV #SN Attr   VSize     VFree
     vgshared    1   1    0 wz--ns 1016.00m  1016.00m
   ```

   The `Attr` column shows the volume attributes. In this example, the volume group is writable (`w`), resizeable (`z`), the allocation policy is normal (`n`), and it is a shared resource (`s`). See the man page of **vgs** for details.

After you have created the volumes and started your resources you should have new device names under `/dev/testvg`, for example `/dev/testvg/lv1`. This indicates the LV has been activated for use.

24.2.3   Scenario: Cluster LVM with DRBD

The following scenarios can be used if you have data centers located in different parts of your city, country, or continent.

PROCEDURE 24.8: **CREATING A CLUSTER-AWARE VOLUME GROUP WITH DRBD**

1. Create a primary/primary DRBD resource:

   a. First, set up a DRBD device as primary/secondary as described in *Procedure 23.2, "Manually configuring DRBD"*. Make sure the disk state is `up-to-date` on both nodes. Check this with **drbdadm status**.

   b. Add the following options to your configuration file (usually something like `/etc/drbd.d/r0.res`):

   ```
   resource r0 {
     net {
         allow-two-primaries;
     }
     ...
   }
   ```

   c. Copy the changed configuration file to the other node, for example:

   ```
   # scp /etc/drbd.d/r0.res venus:/etc/drbd.d/
   ```

   d. Run the following commands on *both* nodes:

   ```
   # drbdadm disconnect r0
   # drbdadm connect r0
   # drbdadm primary r0
   ```

   e. Check the status of your nodes:

   ```
   # drbdadm status r0
   ```

2. Include the lvmlockd resource as a clone in the pacemaker configuration, and make it depend on the DLM clone resource. See *Procedure 24.1, "Creating a DLM resource"* for detailed instructions. Before proceeding, confirm that these resources have started successfully on your cluster. Use `crm status` or the Web interface to check the running services.

3. Prepare the physical volume for LVM with the command `pvcreate`. For example, on the device `/dev/drbd_r0` the command would look like this:

```
# pvcreate /dev/drbd_r0
```

4. Create a shared volume group:

```
# vgcreate --shared testvg /dev/drbd_r0
```

5. Create logical volumes as needed. For example, create a 4 GB logical volume with the following command:

```
# lvcreate --name lv1 -L 4G testvg
```

6. The logical volumes within the VG are now available as file system mounts for raw usage. Ensure that services using them have proper dependencies to collocate them with and order them after the VG has been activated.

After finishing these configuration steps, the LVM configuration can be done like on any stand-alone workstation.

24.3   Configuring eligible LVM devices explicitly

When several devices seemingly share the same physical volume signature (as can be the case for multipath devices or DRBD), we recommend to explicitly configure the devices which LVM scans for PVs.

For example, if the command `vgcreate` uses the physical device instead of using the mirrored block device, DRBD will be confused. This may result in a split-brain condition for DRBD.

To deactivate a single device for LVM, do the following:

1. Edit the file `/etc/lvm/lvm.conf` and search for the line starting with `filter`.

2. The patterns there are handled as regular expressions. A leading "a" means to accept a device pattern to the scan, a leading "r" rejects the devices that follow the device pattern.

3. To remove a device named `/dev/sdb1`, add the following expression to the filter rule:

```
"r|^/dev/sdb1$|"
```

The complete filter line will look like the following:

```
filter = [ "r|^/dev/sdb1$|", "r|/dev/.*/by-path/.*|", "r|/dev/.*/by-id/.*|",
 "a/.*/" ]
```

A filter line that accepts DRBD and MPIO devices but rejects all other devices would look like this:

```
filter = [ "a|/dev/drbd.*|", "a|/dev/.*/by-id/dm-uuid-mpath-.*|", "r/.*/" ]
```

4. Write the configuration file and copy it to all cluster nodes.

24.4   Online migration from mirror LV to cluster MD

Starting with SUSE Linux Enterprise High Availability 15, `cmirrord` in Cluster LVM is deprecated. We highly recommend to migrate the mirror logical volumes in your cluster to cluster MD. Cluster MD stands for cluster multi-device and is a software-based RAID storage solution for a cluster.

24.4.1   Example setup before migration

Let us assume you have the following example setup:

- You have a two-node cluster consisting of the nodes `alice` and `bob`.

- A mirror logical volume named `test-lv` was created from a volume group named `clus-ter-vg2`.

- The volume group `cluster-vg2` is composed of the disks `/dev/vdb` and `/dev/vdc`.

```
# lsblk
NAME                            MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
vda                             253:0    0   40G  0 disk
├─vda1                          253:1    0    4G  0 part [SWAP]
└─vda2                          253:2    0   36G  0 part /
vdb                             253:16   0   20G  0 disk
```

```
├─cluster--vg2-test--lv_mlog_mimage_0 254:0    0    4M  0 lvm
│ └─cluster--vg2-test--lv_mlog         254:2    0    4M  0 lvm
│   └─cluster--vg2-test--lv            254:5    0   12G  0 lvm
└─cluster--vg2-test--lv_mimage_0       254:3    0   12G  0 lvm
  └─cluster--vg2-test--lv              254:5    0   12G  0 lvm
vdc                                    253:32   0   20G  0 disk
├─cluster--vg2-test--lv_mlog_mimage_1 254:1    0    4M  0 lvm
│ └─cluster--vg2-test--lv_mlog         254:2    0    4M  0 lvm
│   └─cluster--vg2-test--lv            254:5    0   12G  0 lvm
└─cluster--vg2-test--lv_mimage_1       254:4    0   12G  0 lvm
  └─cluster--vg2-test--lv              254:5    0   12G  0 lvm
```

❗ Important: Avoiding migration failures

Before you start the migration procedure, check the capacity and degree of utilization of your logical and physical volumes. If the logical volume uses 100% of the physical volume capacity, the migration might fail with an `insufficient free space` error on the target volume. How to prevent this migration failure depends on the options used for mirror log:

- **Is the mirror log itself mirrored (`mirrored` option) and allocated on the same device as the mirror leg?** (For example, this might be the case if you have created the logical volume for a `cmirrord` setup on SUSE Linux Enterprise High Availability 11 or 12 as described in the Administration Guide for those versions (https://documentation.suse.com/sle-ha/12-SP5/html/SLE-HA-all/cha-ha-clvm.html#sec-ha-clvm-config-cmirrord)↗.)

  By default, `mdadm` reserves a certain amount of space between the start of a device and the start of array data. During migration, you can check for the unused padding space and reduce it with the `data-offset` option as shown in *Step 1.d* and following.

  The `data-offset` must leave enough space on the device for cluster MD to write its metadata to it. However, the offset must be small enough for the remaining capacity of the device to accommodate all physical volume extents of the migrated volume. Because the volume may have spanned the complete device minus the mirror log, the offset must be smaller than the size of the mirror log.

We recommend to set the `data-offset` to 128 kB. If no value is specified for the offset, its default value is 1 kB (1024 bytes).

- **Is the mirror log written to a different device (`disk` option) or kept in memory (`core` option)?** Before starting the migration, either enlarge the size of the physical volume or reduce the size of the logical volume (to free more space for the physical volume).

24.4.2 Migrating a mirror LV to cluster MD

The following procedure is based on *Section 24.4.1, "Example setup before migration"*. Adjust the instructions to match your setup and replace the names for the LVs, VGs, disks, and the cluster MD device accordingly.

The migration does not involve any downtime. The file system can still be mounted during the migration procedure.

1. On node `alice`, execute the following steps:

   a. Convert the mirror logical volume `test-lv` to a linear logical volume:

   ```
   # lvconvert -m0 cluster-vg2/test-lv /dev/vdc
   ```

   b. Remove the physical volume `/dev/vdc` from the volume group `cluster-vg2`:

   ```
   # vgreduce cluster-vg2 /dev/vdc
   ```

   c. Remove this physical volume from LVM:

   ```
   # pvremove /dev/vdc
   ```

   When you run **lsblk** now, you get:

   ```
   NAME                           MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
   vda                     253:0    0   40G  0 disk
   ├─vda1                  253:1    0    4G  0 part [SWAP]
   └─vda2                  253:2    0   36G  0 part /
   vdb                     253:16   0   20G  0 disk
   └─cluster--vg2-test--lv 254:5    0   12G  0 lvm
   vdc                     253:32   0   20G  0 disk
   ```

d. Create a cluster MD device `/dev/md0` with the disk `/dev/vdc`:

```
# mdadm --create /dev/md0 --bitmap=clustered \
    --metadata=1.2 --raid-devices=1 --force --level=mirror \
    /dev/vdc --data-offset=128
```

For details on why to use the `data-offset` option, see *Important: Avoiding migration failures*.

2. On node `bob`, assemble this MD device:

```
# mdadm --assemble md0 /dev/vdc
```

If your cluster consists of more than two nodes, execute this step on all remaining nodes in your cluster.

3. Back on node `alice`:

a. Initialize the MD device `/dev/md0` as physical volume for use with LVM:

```
# pvcreate /dev/md0
```

b. Add the MD device `/dev/md0` to the volume group `cluster-vg2`:

```
# vgextend cluster-vg2 /dev/md0
```

c. Move the data from the disk `/dev/vdb` to the `/dev/md0` device:

```
# pvmove /dev/vdb /dev/md0
```

d. Remove the physical volume `/dev/vdb` from the volume `group cluster-vg2`:

```
# vgreduce cluster-vg2 /dev/vdb
```

e. Remove the label from the device so that LVM no longer recognizes it as physical volume:

```
# pvremove /dev/vdb
```

f. Add `/dev/vdb` to the MD device `/dev/md0`:

```
# mdadm --grow /dev/md0 --raid-devices=2 --add /dev/vdb
```

24.4.3 Example setup after migration

When you run `lsblk` now, you get:

```
NAME                     MAJ:MIN RM  SIZE RO TYPE  MOUNTPOINT
vda                      253:0    0   40G  0 disk
├─vda1                   253:1    0    4G  0 part  [SWAP]
└─vda2                   253:2    0   36G  0 part  /
vdb                      253:16   0   20G  0 disk
└─md0                      9:0    0   20G  0 raid1
  └─cluster--vg2-test--lv 254:5   0   12G  0 lvm
vdc                      253:32   0   20G  0 disk
└─md0                      9:0    0   20G  0 raid1
  └─cluster--vg2-test--lv 254:5   0   12G  0 lvm
```

25 Cluster multi-device (Cluster MD)

The cluster multi-device (Cluster MD) is a software based RAID storage solution for a cluster. Currently, Cluster MD provides the redundancy of RAID1 mirroring to the cluster. With SUSE Linux Enterprise High Availability 15 SP5, RAID10 is included as a technology preview. If you want to try RAID10, replace `mirror` with `10` in the related **mdadm** command. This chapter shows you how to create and use Cluster MD.

25.1 Conceptual overview

The Cluster MD provides support for use of RAID1 across a cluster environment. The disks or devices used by Cluster MD are accessed by each node. If one device of the Cluster MD fails, it can be replaced at runtime by another device and it is re-synced to provide the same amount of redundancy. The Cluster MD requires Corosync and Distributed Lock Manager (DLM) for coordination and messaging.

A Cluster MD device is not automatically started on boot like the rest of the regular MD devices. A clustered device needs to be started using resource agents to ensure the DLM resource has been started.

25.2 Creating a clustered MD RAID device

REQUIREMENTS

- A running cluster with pacemaker.

- A resource agent for DLM (see *Section 20.2, "Configuring DLM cluster resources"*).

- At least two shared disk devices. You can use an additional device as a spare that fails over automatically in case of device failure.

- An installed package `cluster-md-kmp-default.`

⛔ Warning: Always use persistent device names

Always use cluster-wide persistent device names, such as `/dev/disk/by-id/DEVICE_ID`.
Unstable device names like `/dev/sdX` or `/dev/dm-X` might become mismatched on different nodes, causing major problems across the cluster.

1. Make sure the DLM resource is up and running on every node of the cluster and check the resource status with the command:

   ```
   # crm_resource -r dlm -W
   ```

2. Create the Cluster MD device:

   - If you do not have an existing normal RAID device, create the Cluster MD device on the node running the DLM resource with the following command:

     ```
     # mdadm --create /dev/md0 --bitmap=clustered \
     --metadata=1.2 --raid-devices=2 --level=mirror \
     /dev/disk/by-id/DEVICE_ID1 /dev/disk/by-id/DEVICE_ID2
     ```

     As Cluster MD only works with version 1.2 of the metadata, it is recommended to specify the version using the `--metadata` option. For other useful options, refer to the man page of **mdadm**. Monitor the progress of the re-sync in `/proc/mdstat`.

   - If you already have an existing normal RAID, first clear the existing bitmap and then create the clustered bitmap:

     ```
     # mdadm --grow /dev/mdX --bitmap=none
     # mdadm --grow /dev/mdX --bitmap=clustered
     ```

   - Optionally, to create a Cluster MD device with a spare device for automatic failover, run the following command on one cluster node:

     ```
     # mdadm --create /dev/md0 --bitmap=clustered --raid-devices=2 \
     --level=mirror --spare-devices=1 --metadata=1.2 \
     /dev/disk/by-id/DEVICE_ID1 /dev/disk/by-id/DEVICE_ID2 /dev/disk/by-
     id/DEVICE_ID3
     ```

3. Get the UUID and the related md path:

   ```
   # mdadm --detail --scan
   ```

The UUID must match the UUID stored in the superblock. For details on the UUID, refer to the `mdadm.conf` man page.

4. Open `/etc/mdadm.conf` and add the md device name and the devices associated with it. Use the UUID from the previous step:

```
DEVICE /dev/disk/by-id/DEVICE_ID1 /dev/disk/by-id/DEVICE_ID2
ARRAY /dev/md0 UUID=1d70f103:49740ef1:af2afce5:fcf6a489
```

5. Open Csync2's configuration file `/etc/csync2/csync2.cfg` and add `/etc/mdadm.conf`:

```
group ha_group
{
   # ... list of files pruned ...
   include /etc/mdadm.conf
}
```

25.3   Configuring a resource agent

Configure a CRM resource as follows:

1. Create a `Raid1` primitive:

```
crm(live)configure# primitive raider Raid1 \
  params raidconf="/etc/mdadm.conf" raiddev=/dev/md0 \
  force_clones=true \
  op monitor timeout=20s interval=10 \
  op start timeout=20s interval=0 \
  op stop timeout=20s interval=0
```

2. Add the `raider` resource to the base group for storage that you have created for DLM:

```
crm(live)configure# modgroup g-storage add raider
```

The **add** sub-command appends the new group member by default.

If not already done, clone the `g-storage` group so that it runs on all nodes:

```
crm(live)configure# clone cl-storage g-storage \
    meta interleave=true target-role=Started
```

3. Review your changes with **show**.

4. If everything is correct, submit your changes with **commit**.

25.4 Adding a device

To add a device to an existing, active Cluster MD device, first ensure that the device is "visible" on each node with the command `cat /proc/mdstat`. If the device is not visible, the command fails.

Use the following command on one cluster node:

```
# mdadm --manage /dev/md0 --add /dev/disk/by-id/DEVICE_ID
```

The behavior of the new device added depends on the state of the Cluster MD device:

- If only one of the mirrored devices is active, the new device becomes the second device of the mirrored devices and a recovery is initiated.

- If both devices of the Cluster MD device are active, the new added device becomes a spare device.

25.5 Re-adding a temporarily failed device

Often the failures are transient and limited to a single node. If any of the nodes encounters a failure during an I/O operation, the device is marked as failed for the entire cluster.

This could happen, for example, because of a cable failure on one of the nodes. After correcting the problem, you can re-add the device. Only the outdated parts are synchronized as opposed to synchronizing the entire device by adding a new one.

To re-add the device, run the following command on one cluster node:

```
# mdadm --manage /dev/md0 --re-add /dev/disk/by-id/DEVICE_ID
```

25.6 Removing a device

Before removing a device at runtime for replacement, do the following:

1. Make sure the device is failed by introspecting `/proc/mdstat`. Look for an `(F)` before the device.

2. Run the following command on one cluster node to make a device fail:

   ```
   # mdadm --manage /dev/md0 --fail /dev/disk/by-id/DEVICE_ID
   ```

3. Remove the failed device using the command on one cluster node:

```
# mdadm --manage /dev/md0 --remove /dev/disk/by-id/DEVICE_ID
```

25.7 Assembling Cluster MD as normal RAID at the disaster recovery site

In the event of disaster recovery, you might face the situation that you do not have a Pacemaker cluster stack in the infrastructure on the disaster recovery site, but applications still need to access the data on the existing Cluster MD disks, or from the backups.

You can convert a Cluster MD RAID to a normal RAID by using the `--assemble` operation with the `-U no-bitmap` option to change the metadata of the RAID disks accordingly.

Find an example below of how to assemble all arrays on the data recovery site:

```
while read i; do
    NAME=`echo $i | sed 's/.*name=//'|awk '{print $1}'|sed 's/.*://'`
    UUID=`echo $i | sed 's/.*UUID=//'|awk '{print $1}'`
    mdadm -AR "/dev/md/$NAME" -u $UUID -U no-bitmap
    echo "NAME =" $NAME ", UUID =" $UUID ", assembled."
done < <(mdadm -Es)
```

26 Samba clustering

A clustered Samba server provides a High Availability solution in your heterogeneous networks. This chapter explains some background information and how to set up a clustered Samba server.

26.1 Conceptual overview

Trivial Database (TDB) has been used by Samba for many years. It allows multiple applications to write simultaneously. To make sure all write operations are successfully performed and do not collide with each other, TDB uses an internal locking mechanism.

Cluster Trivial Database (CTDB) is a small extension of the existing TDB. CTDB is described by the project as a "cluster implementation of the TDB database used by Samba and other projects to store temporary data".

Each cluster node runs a local CTDB daemon. Samba communicates with its local CTDB daemon instead of writing directly to its TDB. The daemons exchange metadata over the network, but actual write and read operations are done on a local copy with fast storage. The concept of CTDB is displayed in *Figure 26.1, "Structure of a CTDB cluster"*.

> ✏ **Note: CTDB for Samba only**
>
> The current implementation of the CTDB Resource Agent configures CTDB to only manage Samba. Everything else, including IP failover, should be configured with Pacemaker.
>
> CTDB is only supported for completely homogeneous clusters. For example, all nodes in the cluster need to have the same architecture. You cannot mix x86 with AMD64.
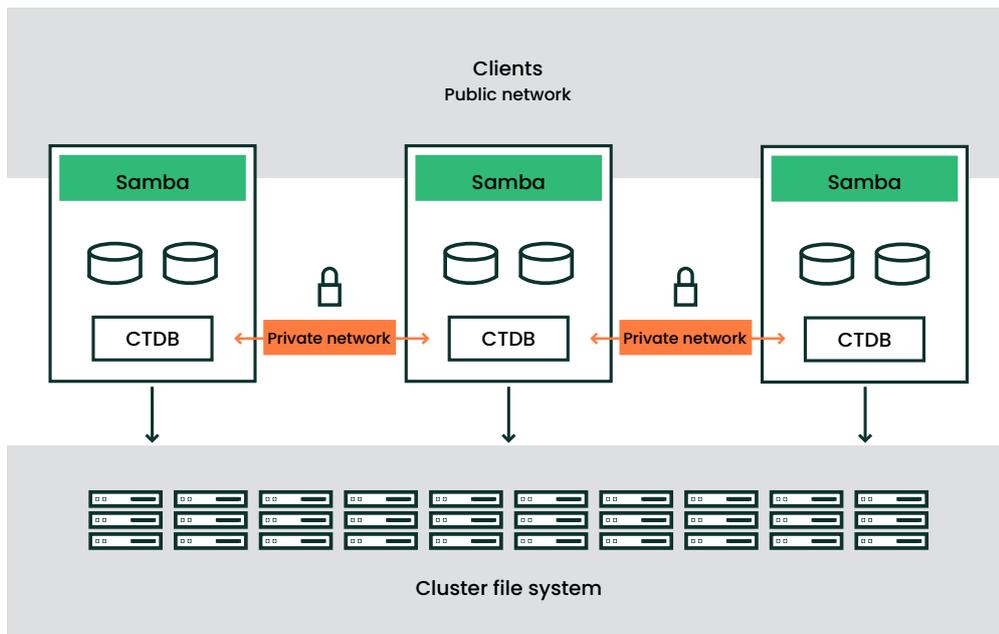
FIGURE 26.1: STRUCTURE OF A CTDB CLUSTER

A clustered Samba server must share certain data:

- Mapping table that associates Unix user and group IDs to Windows users and groups.

- The user database must be synchronized between all nodes.

- Join information for a member server in a Windows domain must be available on all nodes.

- Metadata needs to be available on all nodes, like active SMB sessions, share connections, and various locks.

The goal is that a clustered Samba server with N + 1 nodes is faster than with only N nodes. One node is not slower than an unclustered Samba server.

26.2 Basic configuration

> **Note: Changed configuration files**
>
> The CTDB Resource Agent automatically changes `/etc/sysconfig/ctdb`. Use `crm ra info CTDB` to list all parameters that can be specified for the CTDB resource.

To set up a clustered Samba server, proceed as follows:

PROCEDURE 26.1: SETTING UP A BASIC CLUSTERED SAMBA SERVER

1. Prepare your cluster:

   a. Make sure the following packages are installed before you proceed: `ctdb`, `tdb-tools`, and `samba` (needed for `smb` and `nmb` resources).

   b. Configure your cluster (Pacemaker, OCFS2) as described in this guide in *Part II, "Configuration and administration"*.

   c. Configure a shared file system, like OCFS2, and mount it, for example, on `/srv/clusterfs`. See *Chapter 21, OCFS2* for more information.

   d. If you want to turn on POSIX ACLs, enable it:

      - For a new OCFS2 file system use:

        ```
        # mkfs.ocfs2 --fs-features=xattr ...
        ```

      - For an existing OCFS2 file system use:

        ```
        # tunefs.ocfs2 --fs-feature=xattr DEVICE
        ```

        Make sure the `acl` option is specified in the file system resource. Use the **crm** shell as follows:

        ```
        crm(live)configure# primitive ocfs2-3 ocf:heartbeat:Filesystem params
         options="acl" ...
        ```

   e. Make sure the services `ctdb`, `smb`, and `nmb` are disabled:

      ```
      # systemctl disable ctdb
      # systemctl disable smb
      # systemctl disable nmb
      ```

   f. Open port `4379` of your firewall on all nodes. This is needed for CTDB to communicate with other cluster nodes.

2. Create a directory for the CTDB lock on the shared file system:

   ```
   # mkdir -p /srv/clusterfs/samba/
   ```

3. In `/etc/ctdb/nodes` insert all nodes which contain all private IP addresses of each node in the cluster:

```
192.168.1.10
192.168.1.11
```

4. Configure Samba. Add the following lines in the `[global]` section of `/etc/samba/smb.conf`. Use the host name of your choice in place of "CTDB-SERVER" (all nodes in the cluster will appear as one big node with this name, effectively):

```
[global]
    # ...
    # settings applicable for all CTDB deployments
    netbios name = CTDB-SERVER
    clustering = yes
    idmap config * : backend = tdb2
    passdb backend = tdbsam
    ctdbd socket = /var/lib/ctdb/ctdbd.socket
    # settings necessary for CTDB on OCFS2
    fileid:algorithm = fsid
    vfs objects = fileid
    # ...
```

5. Copy the configuration file to all of your nodes by using `csync2`:

```
# csync2 -xv
```

For more information, see *Procedure 4.9, "Synchronizing the configuration files with Csync2"*.

6. Add a CTDB resource to the cluster:

```
# crm configure
crm(live)configure# primitive ctdb CTDB params \
    ctdb_manages_winbind="false" \
    ctdb_manages_samba="false" \
    ctdb_recovery_lock="/srv/clusterfs/samba/ctdb.lock" \
    ctdb_socket="/var/lib/ctdb/ctdbd.socket" \
      op monitor interval="10" timeout="20" \
      op start interval="0" timeout="90" \
      op stop interval="0" timeout="100"
crm(live)configure# primitive nmb systemd:nmb \
    op start timeout="60" interval="0" \
    op stop timeout="60" interval="0" \
    op monitor interval="60" timeout="60"
crm(live)configure# primitive smb systemd:smb \
    op start timeout="60" interval="0" \
```

```
        op stop timeout="60" interval="0" \
        op monitor interval="60" timeout="60"
crm(live)configure# group g-ctdb ctdb nmb smb
crm(live)configure# clone cl-ctdb g-ctdb meta interleave="true"
crm(live)configure# colocation col-ctdb-with-clusterfs inf: cl-ctdb cl-clusterfs
crm(live)configure# order o-clusterfs-then-ctdb Mandatory: cl-clusterfs cl-ctdb
crm(live)configure# commit
```

7. Add a clustered IP address:

```
crm(live)configure# primitive ip IPaddr2 params ip=192.168.2.222 \
        unique_clone_address="true" \
        op monitor interval="60" \
        meta resource-stickiness="0"
crm(live)configure# clone cl-ip ip \
        meta interleave="true" clone-node-max="2" globally-unique="true"
crm(live)configure# colocation col-ip-with-ctdb 0: cl-ip cl-ctdb
crm(live)configure# order o-ip-then-ctdb 0: cl-ip cl-ctdb
crm(live)configure# commit
```

If `unique_clone_address` is set to `true`, the IPaddr2 resource agent adds a clone ID to the specified address, leading to three different IP addresses. These are usually not needed, but help with load balancing. For further information about this topic, see *Section 17.2, "Configuring load balancing with Linux Virtual Server"*.

8. Commit your change:

```
crm(live)configure# commit
```

9. Check the result:

```
# crm status
Clone Set: cl-storage [dlm]
     Started: [ factory-1 ]
     Stopped: [ factory-0 ]
Clone Set: cl-clusterfs [clusterfs]
     Started: [ factory-1 ]
     Stopped: [ factory-0 ]
 Clone Set: cl-ctdb [g-ctdb]
     Started: [ factory-1 ]
     Started: [ factory-0 ]
 Clone Set: cl-ip [ip] (unique)
     ip:0        (ocf:heartbeat:IPaddr2):        Started factory-0
     ip:1        (ocf:heartbeat:IPaddr2):        Started factory-1
```

10. Test from a client machine. On a Linux client, run the following command to see if you can copy files from and to the system:

```
# smbclient //192.168.2.222/myshare
```

26.3 Joining an Active Directory domain

Active Directory (AD) is a directory service for Windows server systems.

The following instructions outline how to join a CTDB cluster to an Active Directory domain:

1. Create a CTDB resource as described in *Procedure 26.1, "Setting up a basic clustered Samba server"*.

2. Install the `samba-winbind` package.

3. Disable the `winbind` service:

   ```
   # systemctl disable winbind
   ```

4. Define a winbind cluster resource:

   ```
   # crm configure
   crm(live)configure# primitive winbind systemd:winbind \
       op start timeout="60" interval="0" \
       op stop timeout="60" interval="0" \
       op monitor interval="60" timeout="60"
   crm(live)configure# commit
   ```

5. Edit the `g-ctdb` group and insert `winbind` between the `nmb` and `smb` resources:

   ```
   crm(live)configure# edit g-ctdb
   ```

   Save and close the editor with `: ─ w` (`vim`).

6. Consult your Windows Server documentation for instructions on how to set up an Active Directory domain. In this example, we use the following parameters:

AD and DNS server	win2k3.2k3test.example.com
AD domain	2k3test.example.com

Cluster AD member NetBIOS name	`CTDB-SERVER`

7. *Procedure 26.2, "Joining Active Directory"*

Finally, join your cluster to the Active Directory server:

PROCEDURE 26.2: JOINING ACTIVE DIRECTORY

1. Make sure the following files are included in Csync2's configuration to become installed on all cluster hosts:

```
/etc/samba/smb.conf
/etc/security/pam_winbind.conf
/etc/krb5.conf
/etc/nsswitch.conf
/etc/security/pam_mount.conf.xml
/etc/pam.d/common-session
```

You can also use YaST's *Configure Csync2* module for this task, see *Section 4.7, "Transferring the configuration to all nodes"*.

2. Run YaST and open the *Windows Domain Membership* module from the *Network Services* entry.

3. Enter your domain or workgroup settings and finish with *Ok*.

26.4 Debugging and testing clustered Samba

To debug your clustered Samba server, the following tools which operate on different levels are available:

`ctdb_diagnostics`

Run this tool to diagnose your clustered Samba server. Detailed debug messages should help you track down any problems.

The **`ctdb_diagnostics`** command searches for the following files which must be available on all nodes:

```
/etc/krb5.conf
/etc/hosts
/etc/ctdb/nodes
/etc/sysconfig/ctdb
```

```
/etc/resolv.conf
/etc/nsswitch.conf
/etc/sysctl.conf
/etc/samba/smb.conf
/etc/fstab
/etc/multipath.conf
/etc/pam.d/system-auth
/etc/sysconfig/nfs
/etc/exports
/etc/vsftpd/vsftpd.conf
```

If the files `/etc/ctdb/public_addresses` and `/etc/ctdb/static-routes` exist, they will be checked as well.

`ping_pong`

Check whether your file system is suitable for CTDB with **`ping_pong`**. It performs certain tests of your cluster file system like coherence and performance (see https://wiki.samba.org/index.php/Ping_pong ⬈) and gives some indication how your cluster may behave under high load.

`send_arp` tool and `SendArp` resource agent

The `SendArp` resource agent is located in `/usr/lib/heartbeat/send_arp` (or `/usr/lib64/heartbeat/send_arp`). The **`send_arp`** tool sends out a gratuitous ARP (Address Resolution Protocol) packet and can be used for updating other machines' ARP tables. It can help to identify communication problems after a failover process. If you cannot connect to a node or ping it although it shows the clustered IP address for Samba, use the **`send_arp`** command to test if the nodes only need an ARP table update.

For more information, refer to https://gitlab.com/wireshark/wireshark/-/wikis/home ⬈.

To test certain aspects of your cluster file system proceed as follows:

PROCEDURE 26.3: TEST COHERENCE AND PERFORMANCE OF YOUR CLUSTER FILE SYSTEM

1. Start the command **`ping_pong`** on one node and replace the placeholder _N_ with the amount of nodes plus one. The file _ABSPATH_`/data.txt` is available in your shared storage and is therefore accessible on all nodes (_ABSPATH_ indicates an absolute path):

   ```
   # ping_pong ABSPATH/data.txt N
   ```

   Expect a very high locking rate as you are running only one node. If the program does not print a locking rate, replace your cluster file system.

2. Start a second copy of **`ping_pong`** on another node with the same parameters.

Expect to see a dramatic drop in the locking rate. If any of the following applies to your cluster file system, replace it:

- `ping_pong` does not print a locking rate per second,

- the locking rates in the two instances are not almost equal,

- the locking rate did not drop after you started the second instance.

3. Start a third copy of `ping_pong`. Add another node and note how the locking rates change.

4. Kill the `ping_pong` commands one after the other. You should observe an increase of the locking rate until you get back to the single node case. If you did not get the expected behavior, find more information in *Chapter 21, OCFS2*.

26.5 For more information

- https://wiki.samba.org/index.php/CTDB_Setup ↗

- https://ctdb.samba.org ↗

- https://wiki.samba.org/index.php/Samba_%26_Clustering ↗

27 Disaster recovery with ReaR (Relax-and-Recover)

Relax-and-Recover ("ReaR") is a disaster recovery framework for use by system administrators. It is a collection of Bash scripts that need to be adjusted to the specific production environment that is to be protected in case of disaster.

No disaster recovery solution works out-of-the-box. Therefore you must take preparations *before* any disaster happens.

27.1 Conceptual overview

The following sections describe the general disaster recovery concept and the basic steps you need to execute for successful recovery with ReaR. They also provide guidance on ReaR requirements, some limitations to be aware of, and scenarios and backup tools.

27.1.1 Creating a disaster recovery plan

Before the worst scenario happens, take action: analyze your IT infrastructure for any substantial risks, evaluate your budget, and create a disaster recovery plan. If you do not already have a disaster recovery plan at hand, find information on each step below:

- **Risk analysis.** Conduct a solid risk analysis of your infrastructure. List all the possible threats and evaluate how serious they are. Determine how likely these threats are and prioritize them. It is recommended to use a simple categorization: probability and impact.

- **Budget planning.** The outcome of the analysis is an overview, which risks can be tolerated and which are critical for your business. Ask yourself how you can minimize risks and how much it costs. Depending on how big your company is, spend two to fifteen percent of the overall IT budget on disaster recovery.

- **Disaster recovery plan development.** Make checklists, test procedures, establish and assign priorities, and inventory your IT infrastructure. Define how to deal with a problem when some services in your infrastructure fail.

- **Test.** After defining an elaborate plan, test it. Test it at least once a year. Use the same testing hardware as your main IT infrastructure.

27.1.2 What does disaster recovery mean?

If a system in a production environment has been destroyed (for whatever reasons—be it broken hardware, a misconfiguration or software problems), you need to re-create the system. The recreation can be done either on the same hardware or on compatible replacement hardware. Re-creating a system means more than restoring files from a backup. It also includes preparing the system's storage (with regard to partitioning, file systems, and mount points), and reinstalling the boot loader.

27.1.3 How does disaster recovery with ReaR work?

While the system is up and running, create a backup of the files and create a recovery system on a recovery medium. The recovery system contains a recovery installer.

In case the system has been destroyed, replace broken hardware (if needed), boot the recovery system from the recovery medium and launch the recovery installer. The recovery installer re-creates the system: First, it prepares the storage (partitioning, file systems, mount points), then it restores the files from the backup. Finally, it reinstalls the boot loader.

27.1.4 ReaR requirements

To use ReaR you need at least two identical systems: the machine that runs your production environment and an identical test machine. "Identical" in this context means that you can, for example, replace a network card with another one using the same Kernel driver.

> ## ✋ Warning: Identical drivers required
>
> If a hardware component does not use the same driver as the one in your production environment, it is not considered identical by ReaR.

27.1.5 ReaR version updates

To see which versions of ReaR are available with SUSE Linux Enterprise High Availability 15 SP5, run the following command:

```
# zypper search --type package --verbose rear
```

> 🗒 **Note: Find important information in changelogs**
>
> Any information about bug fixes, incompatibilities, and other issues can be found in the changelogs of the packages. It is recommended to review also later package versions of ReaR in case you need to re-validate your disaster recovery procedure.

Be aware of the following issues with ReaR:

- To allow disaster recovery on UEFI systems, you need the package `xorriso`, which provides the helper tool `/usr/bin/xorrisofs`. This helper tool is used to create a UEFI-bootable ReaR recovery system ISO image.

- If you have a tested and fully functional disaster recovery procedure with one ReaR version, do not update ReaR. Keep the ReaR package and do not change your disaster recovery method.

- Version updates for ReaR are provided as separate packages that intentionally conflict with each other to prevent your installed version getting accidentally replaced with another version.

In the following cases you need to completely re-validate your existing disaster recovery procedure:

- For each ReaR version update.

- When you update ReaR manually.

- For each software that is used by ReaR.

- If you update low-level system components such as **parted**, **btrfs** and similar.

27.1.6 Limitations with Btrfs

The following limitations apply if you use Btrfs.

Your system includes subvolumes, but no snapshot subvolumes

At least ReaR version 1.17.2.a is required. This version supports re-creating "normal" Btrfs subvolume structure (no snapshot subvolumes).

Your system includes snapshot subvolumes

> ## ✋ Warning
>
> Btrfs snapshot subvolumes *cannot* be backed up and restored as usual with file-based backup software.

While recent snapshot subvolumes on Btrfs file systems need almost no disk space (because of Btrfs's copy-on-write functionality), those files would be backed up as complete files when using file-based backup software. They would end up twice in the backup with their original file size. Therefore, it is impossible to restore the snapshots as they have been before on the original system.

Your SLE system needs matching ReaR configuration

For example, the setups in SLE12 GA, SLE12 SP1, and SLE12 SP2 have several incompatible Btrfs default structures. As such, it is crucial to use a matching ReaR configuration file. See the example files `/usr/share/rear/conf/examples/SLE12*-btrfs-example.conf`.

27.1.7 Scenarios and backup tools

ReaR can create a disaster recovery system (including a system-specific recovery installer) that can be booted from a local medium (like a hard disk, a flash disk, a DVD/CD-R) or via PXE. The backup data can be stored on a network file system, for example NFS, as described in *Example 27.1*.

ReaR does not replace a file backup, but complements it. By default, ReaR supports the generic `tar` command, and several third-party backup tools (such as Tivoli Storage Manager, QNetix Galaxy, Symantec NetBackup, EMC NetWorker, or HP DataProtector). Refer to *Example 27.3* for an example configuration of using ReaR with EMC NetWorker as backup tool.

27.1.8 Basic steps

For a successful recovery with ReaR in case of disaster, you need to execute the following basic steps:

Setting up ReaR and your backup solution

This includes tasks like editing the ReaR configuration file, adjusting the Bash scripts, and configuring the backup solution that you want to use.

Creating the recovery installation system

While the system to be protected is up and running, create a file backup and generate a recovery system that contains a system-specific ReaR recovery installer.

For basic backups with `tar`, use the `rear mkbackup` command to create both the backup and the recovery system.

For third-party backup tools, use the `rear mkrescue` command to create the recovery system, and use the third-party backup tool to create the backup.

Testing the recovery process

Whenever you have created a disaster recovery medium with ReaR, test the disaster recovery process thoroughly. You must use a test machine that has *identical* hardware like the one that is part of your production environment. For details, refer to *Section 27.1.4, "ReaR requirements"*.

Recovering from disaster

After a disaster has occurred, replace any broken hardware (if necessary). Then boot the ReaR recovery system and start the recovery installer with the `rear recover` command.

27.2 Setting up ReaR and your backup solution

To install the latest version of ReaR, run the following command:

```
# zypper install rear
```

If you need to install an earlier version of ReaR, you can specify the package version. For example:

```
# zypper install rear23a
```

To configure ReaR, add your settings to the ReaR configuration file `/etc/rear/local.conf` and, if needed, edit the Bash scripts that are part of the ReaR framework. In particular, you need to define the following tasks that ReaR should do:

- **How to back up files and how to create and store the disaster recovery system.** This needs to be configured in `/etc/rear/local.conf`.

- **What to re-create (partitioning, file systems, mount points, etc.).** This can be defined in `/etc/rear/local.conf`. To re-create non-standard systems, you may need to enhance the Bash scripts.

- **How the recovery process works.** To change how ReaR generates the recovery installer, or to adapt what the ReaR recovery installer does, you need to edit the Bash scripts.

All ReaR configuration variables and their default values are explained in `/usr/share/rear/conf/default.conf`.

Below are some examples of useful configuration options. You can also find example files for different scenarios in the `/usr/share/rear/conf/examples/` subdirectory. Use these to help you create your `/etc/rear/local.conf` file.

EXAMPLE 27.1: USING AN NFS SERVER TO STORE THE FILE BACKUP

1. Set up an NFS server with YaST as described in the Administration Guide for SUSE Linux Enterprise Server 15 SP5 (https://documentation.suse.com/sles-15/html/SLES-all/cha-nfs.html) ↗.

2. Define the configuration for your NFS server in the `/etc/exports` file. Make sure the directory on the NFS server has the right mount options. For example, you might need `no_root_squash` as the **rear mkbackup** command runs as `root`. For more information, see **man exports**.

3. Adjust the various `BACKUP` parameters in the configuration file `/etc/rear/local.conf` to make ReaR store the file backup on the respective NFS server. Find examples in your installed system under `/usr/share/rear/conf/examples/SLE*-example.conf`.

EXAMPLE 27.2: BACKING UP BTRFS SUBVOLUMES WITH `tar`

Btrfs is set up with subvolumes, but **tar** creates backups with the `--one-file-system` option, which excludes subvolumes. Therefore, the mount points for the subvolumes must be explicitly included in the ReaR configuration.

Add this configuration snippet to `/etc/rear/local.conf` and adjust it according to your setup. For more information and additional options, see the example files at `/usr/share/rear/conf/examples/SLE*-btrfs-example.conf`.

```
OUTPUT=ISO
BACKUP=NETFS
BACKUP_URL=nfs://host.example.com/path/to/rear/backup
BACKUP_PROG_INCLUDE=( /root /boot/grub2/i386-pc /tmp /opt /var /boot/grub2/x86_64-
efi /srv /usr/local )
```

> 💡 **Tip: Mount points for BACKUP_PROG_INCLUDE**
>
> You can find the mount points for the subvolumes on your specific system by running the following command:
>
> ```
> # findmnt -n -r -o TARGET -t btrfs | grep -v '^/$' | egrep -v 'snapshots|
> crash'
> ```

EXAMPLE 27.3: USING THIRD-PARTY BACKUP TOOLS LIKE EMC NETWORKER

Using third-party backup tools instead of `tar` requires appropriate settings in the ReaR configuration file.

The following is an example configuration for EMC NetWorker. Add this configuration snippet to `/etc/rear/local.conf` and adjust it according to your setup:

```
BACKUP=NSR
OUTPUT=ISO
BACKUP_URL=nfs://host.example.com/path/to/rear/backup
OUTPUT_URL=nfs://host.example.com/path/to/rear/backup
NSRSERVER=backupserver.example.com
RETENTION_TIME="Month"
```

For more information about supported third-party backup tools, see `man rear`, section *BACKUP SOFTWARE INTEGRATION*.

EXAMPLE 27.4: BACKING UP MULTIPATH DEVICES

By default, ReaR ignores file systems on multipath devices because it assumes they are on remote storage, not part of the local system. You can make ReaR include these file systems by adding the following line to `/etc/rear/local.conf`:

```
AUTOEXCLUDE_MULTIPATH=n
```

If your system boots with a UEFI boot loader, additional configuration is required:

1. Install the package `xorriso`:

   ```
   # zypper install xorriso
   ```

2. Add the following line to `/etc/rear/local.conf`:

   ```
   ISO_MKISOFS_BIN="/usr/bin/xorrisofs"
   ```

3. If your system boots with UEFI Secure Boot, you must also add the following line:

   ```
   SECURE_BOOT_BOOTLOADER="/boot/efi/EFI/sles/shim.efi"
   ```

For more information about ReaR configuration variables for UEFI, see the `/usr/share/rear/conf/default.conf` file.

27.3 Creating the recovery installation system

After you have configured ReaR as described in *Section 27.2*, create the recovery installation system (including the ReaR recovery installer) plus the file backup.

EXAMPLE 27.6: CREATING A RECOVERY SYSTEM WITH A BASIC `tar` BACKUP

Run the `rear mkbackup` command:

```
# rear -d -D mkbackup
```

This command performs the following steps:

- Analyzing the target system and gathering information, in particular about the disk layout (partitioning, file systems, mount points) and about the boot loader.

- Creating a bootable recovery system with the information gathered in the first step. The resulting ReaR recovery installer is *specific* to the system that you want to protect from disaster. It can only be used to re-create this specific system.

- Calling the internal backup tool to back up system files and user files.

EXAMPLE 27.7: CREATING A RECOVERY SYSTEM WITH A THIRD-PARTY BACKUP

1. Run the `rear mkrescue` command:

   ```
   # rear -d -D mkrescue
   ```

This command analyzes the target and creates a recovery system, but does *not* create a backup of the files.

2. Create a file backup using your third-party backup tool.

27.4 Testing the recovery process

After having created the recovery system, test the recovery process on a test machine with identical hardware. See also *Section 27.1.4, "ReaR requirements"*. Make sure the test machine is correctly set up and can serve as a replacement for your main machine.

🤚 Warning: Extensive testing on identical hardware

Thorough testing of the disaster recovery process on machines is required. Test the recovery procedure on a regular basis to ensure everything works as expected.

PROCEDURE 27.1: PERFORMING A DISASTER RECOVERY ON A TEST MACHINE

1. Create a recovery medium by burning the recovery system that you have created in *Section 27.3* to a DVD or CD. Alternatively, you can use a network boot via PXE.

2. Boot the test machine from the recovery medium.

3. From the menu, select *Recover*.

4. Log in as `root` (no password needed).

5. Enter the following command to start the recovery installer:

```
rear -d -D recover
```

For details about the steps that ReaR takes during the process, see *Recovery process*.

6. After the recovery process has finished, check whether the system has been successfully re-created and can serve as a replacement for your original system in the production environment.

27.5 Recovering from disaster

In case a disaster has occurred, replace any broken hardware if necessary. Then proceed as described in *Procedure 27.1*, using either the repaired machine (or a tested, identical machine that can serve as a replacement for your original system).

The `rear recover` command executes the following steps:

RECOVERY PROCESS

1. Restoring the disk layout (partitions, file systems, and mount points).

2. Restoring the system and user files from the backup.

3. Restoring the boot loader.

27.6 For more information

- https://en.opensuse.org/SDB:Disaster_Recovery ↗

- `rear` man page

- `/usr/share/doc/packages/rear/`

IV Maintenance and upgrade

28 Executing maintenance tasks

To perform maintenance tasks on the cluster nodes, you might need to stop the resources running on that node, to move them, or to shut down or reboot the node. It might also be necessary to temporarily take over the control of resources from the cluster, or even to stop the cluster service while resources remain running.

This chapter explains how to manually take down a cluster node without negative side-effects. It also gives an overview of different options the cluster stack provides for executing maintenance tasks.

28.1 Preparing and finishing maintenance work

Use the following commands to start, stop, or view the status of the cluster:

`crm cluster start [--all]`
> Start the cluster services on one node or all nodes

`crm cluster stop [--all]`
> Stop the cluster services on one node or all nodes

`crm cluster restart [--all]`
> Restart the cluster services on one node or all nodes

`crm cluster status`
> View the status of the cluster stack

Execute the above commands as user `root`, or as a user with the required privileges.

When you shut down or reboot a cluster node (or stop the cluster services on a node), the following processes are triggered:

- The resources that are running on the node are stopped or moved off the node.

- If stopping a resource fails or times out, the STONITH mechanism fences the node and shuts it down.

🖐 Warning: Risk of data loss

> If you need to do testing or maintenance work, follow the general steps below.

Otherwise, you risk unwanted side effects, like resources not starting in an orderly fashion, unsynchronized CIBs across the cluster nodes, or even data loss.

1. Before you start, choose the appropriate option from *Section 28.2, "Different options for maintenance tasks"*.

2. Apply this option with Hawk2 or crmsh.

3. Execute your maintenance task or tests.

4. After you have finished, put the resource, node or cluster back to "normal" operation.

28.2 Different options for maintenance tasks

Pacemaker offers the following options for performing system maintenance:

Putting the cluster into maintenance mode

The global cluster property `maintenance-mode` puts all resources into maintenance state at once. The cluster stops monitoring them and becomes oblivious to their status. *Only* the resource management by Pacemaker is disabled. Corosync and SBD are still functional. Use maintenance mode for any tasks involving cluster resources. For any tasks involving infrastructure, such as storage or networking, the safest method is to stop the cluster services completely. See *Stopping the cluster services for the whole cluster*.

Stopping the cluster services for the whole cluster

Stopping the cluster services on all nodes at once allows you to shut down a cluster while avoiding the mass migration of resources that would happen if you shut down each node one by one. Because there are no nodes to migrate to, all resources will be stopped.

Putting a node into maintenance mode

This option allows you to put all resources running on a specific node into maintenance state at once. The cluster will cease monitoring them and thus become oblivious to their status.

Putting a node into standby mode

A node that is in standby mode can no longer run resources. Any resources running on the node are moved away or stopped (if no other node is eligible to run the resource). Also, all monitoring operations are stopped on the node (except for those with `role="Stopped"`). You can use this option if you need to stop a node in a cluster while continuing to provide the services running on another node.

Stopping the cluster services on a node

This option stops all the cluster services on a single node. Any resources running on the node are moved away or stopped (if no other node is eligible to run the resource). If stopping a resource fails or times out, the node is fenced.

Putting a resource into maintenance mode

When this mode is enabled for a resource, no monitoring operations will be triggered for the resource.

Use this option if you need to manually touch the service that is managed by this resource and do not want the cluster to run any monitoring operations for the resource during that time.

Putting a resource into unmanaged mode

The `is-managed` meta attribute allows you to temporarily "release" a resource from being managed by the cluster stack. This means you can manually touch the service that is managed by this resource (for example, to adjust any components). However, the cluster will continue to *monitor* the resource and to report any failures.

If you want the cluster to also cease *monitoring* the resource, use the per-resource maintenance mode instead (see *Putting a resource into maintenance mode*).

28.3 Putting the cluster into maintenance mode

⛔ Warning: Maintenance mode only disables Pacemaker

When putting a cluster into maintenance mode, only the resource management by Pacemaker is disabled. Corosync and SBD are still functional. Depending on your maintenance tasks, this might lead to fence operations.

Use maintenance mode for any tasks involving cluster resources. For any tasks involving infrastructure, such as storage or networking, the safest method is to stop the cluster services completely. See *Section 28.4, "Stopping the cluster services for the whole cluster"*.

To put the cluster into maintenance mode on the crm shell, use the following command:

```
# crm maintenance on
```

To put the cluster back to normal mode after your maintenance work is done, use the following command:

```
# crm maintenance off
```

PROCEDURE 28.1: PUTTING THE CLUSTER INTO MAINTENANCE MODE WITH HAWK2

1. Start a Web browser and log in to the cluster as described in *Section 5.4.2, "Logging in"*.

2. In the left navigation bar, select *Configuration* › *Cluster Configuration*.

3. Select the *maintenance-mode* attribute from the empty drop-down box.

4. From the `maintenance-mode` drop-down box, select *Yes*.

5. Click *Apply*.

6. After you have finished the maintenance task for the whole cluster, select *No* from the `maintenance-mode` drop-down box, then click *Apply*.
   From this point on, High Availability will take over cluster management again.

28.4 Stopping the cluster services for the whole cluster

To stop the cluster services on all nodes at once, use the following command:

```
# crm cluster stop --all
```

To start the cluster services again after your maintenance work is done, use the following command:

```
# crm cluster start --all
```

Warning: Graceful shutdown not guaranteed

The `--all` option alone does not guarantee graceful shutdown of the cluster, because of the unexpected fencing that might be triggered by resource stop-failure at the application level. If applications are critical, consider stopping them before stopping the cluster services for the whole cluster.

28.5   Putting a node into maintenance mode

To put a node into maintenance mode on the crm shell, use the following command:

```
# crm node maintenance NODENAME
```

To put the node back to normal mode after your maintenance work is done, use the following command:

```
# crm node ready NODENAME
```

PROCEDURE 28.2: PUTTING A NODE INTO MAINTENANCE MODE WITH HAWK2

1. Start a Web browser and log in to the cluster as described in *Section 5.4.2, "Logging in"*.

2. In the left navigation bar, select *Cluster Status*.

3. In one of the individual nodes' views, click the wrench icon next to the node and select *Maintenance*.

4. After you have finished your maintenance task, click the wrench icon next to the node and select *Ready*.

28.6   Putting a node into standby mode

To put a node into standby mode on the crm shell, use the following command:

```
# crm node standby NODENAME
```

To bring the node back online after your maintenance work is done, use the following command:

```
# crm node online NODENAME
```

1. Start a Web browser and log in to the cluster as described in *Section 5.4.2, "Logging in"*.

2. In the left navigation bar, select *Cluster Status*.

3. In one of the individual nodes' views, click the wrench icon next to the node and select *Standby*.

4. Finish the maintenance task for the node.

5. To deactivate the standby mode, click the wrench icon next to the node and select *Ready*.

28.7   Stopping the cluster services on a node

You can move the services off the node in an orderly fashion before shutting down or rebooting the node. This allows services to migrate off the node without being limited by the shutdown timeout of the cluster services.

PROCEDURE 28.4: **MANUALLY REBOOTING A CLUSTER NODE**

1. On the node you want to reboot or shut down, log in as `root` or equivalent.

2. Put the node into `standby` mode:

   ```
   # crm node standby
   ```

   By default, the node will remain in `standby` mode after rebooting. Alternatively, you can set the node to come back online automatically with `crm node standby reboot`.

3. Check the cluster status:

   ```
   # crm status
   ```

   It shows the respective node in `standby` mode:

   ```
   [...]
   Node bob: standby
   [...]
   ```

4. Stop the cluster services on that node:

   ```
   # crm cluster stop
   ```

5. Reboot the node.

To check if the node joins the cluster again:

1. After the node reboots, log in to it again.

2. Check if the cluster services have started:

   ```
   # crm cluster status
   ```

   This might take some time. If the cluster services do not start again on their own, start them manually:

   ```
   # crm cluster start
   ```

3. Check the cluster status:

   ```
   # crm status
   ```

4. If the node is still in `standby` mode, bring it back online:

   ```
   # crm node online
   ```

28.8 Putting a resource into maintenance mode

To put a resource into maintenance mode on the crm shell, use the following command:

```
# crm resource maintenance RESOURCE_ID true
```

To put the resource back into normal mode after your maintenance work is done, use the following command:

```
# crm resource maintenance RESOURCE_ID false
```

PROCEDURE 28.5: PUTTING A RESOURCE INTO MAINTENANCE MODE WITH HAWK2

1. Start a Web browser and log in to the cluster as described in *Section 5.4.2, "Logging in"*.

2. In the left navigation bar, select *Resources*.

3. Select the resource you want to put in maintenance mode or unmanaged mode, click the wrench icon next to the resource and select *Edit Resource*.

4. Open the *Meta Attributes* category.

5. From the empty drop-down list, select the *maintenance* attribute and click the plus icon to add it.

6. Activate the check box next to `maintenance` to set the maintenance attribute to `yes`.

7. Confirm your changes.

8. After you have finished the maintenance task for that resource, deactivate the check box next to the `maintenance` attribute for that resource.
   From this point on, the resource will be managed by the High Availability software again.

28.9 Putting a resource into unmanaged mode

To put a resource into unmanaged mode on the crm shell, use the following command:

```
# crm resource unmanage RESOURCE_ID
```

To put it into managed mode again after your maintenance work is done, use the following command:

```
# crm resource manage RESOURCE_ID
```

PROCEDURE 28.6: PUTTING A RESOURCE INTO UNMANAGED MODE WITH HAWK2

1. Start a Web browser and log in to the cluster as described in *Section 5.4.2, "Logging in"*.

2. From the left navigation bar, select *Status* and go to the *Resources* list.

3. In the *Operations* column, click the arrow down icon next to the resource you want to modify and select *Edit*.
   The resource configuration screen opens.

4. Below *Meta Attributes*, select the *is-managed* entry from the empty drop-down box.

5. Set its value to `No` and click *Apply*.

6. After you have finished your maintenance task, set *is-managed* to `Yes` (which is the default value) and apply your changes.
   From this point on, the resource will be managed by the High Availability software again.

28.10 Rebooting a cluster node while in maintenance mode

Note: Implications

If the cluster or a node is in maintenance mode, you can use tools external to the cluster stack (for example, `systemctl`) to manually operate the components that are managed by the cluster as resources. The High Availability software will not monitor them or attempt to restart them.

If you stop the cluster services on a node, all daemons and processes (originally started as Pacemaker-managed cluster resources) will continue to run.

If you attempt to start cluster services on a node while the cluster or node is in maintenance mode, Pacemaker will initiate a single one-shot monitor operation (a "probe") for every resource to evaluate which resources are currently running on that node. However, it will take no further action other than determining the resources' status.

PROCEDURE 28.7: REBOOTING A CLUSTER NODE WHILE THE CLUSTER OR NODE IS IN MAINTENANCE MODE

1. On the node you want to reboot or shut down, log in as `root` or equivalent.

2. If you have a DLM resource (or other resources depending on DLM), make sure to explicitly stop those resources before stopping the cluster services:

   ```
   crm(live)resource# stop RESOURCE_ID
   ```

   The reason is that stopping Pacemaker also stops the Corosync service on whose membership and messaging services DLM depends. If Corosync stops, the DLM resource will assume a split-brain scenario and trigger a fencing operation.

3. Stop the cluster services on that node:

   ```
   # crm cluster stop
   ```

4. Shut down or reboot the node.

29 Upgrading your cluster and updating software packages

This chapter covers two different scenarios: upgrading a cluster to another version of SUSE Linux Enterprise High Availability (either a major release or a service pack) as opposed to updating individual packages on cluster nodes. See *Section 29.2, "Upgrading your cluster to the latest product version"* versus *Section 29.3, "Updating software packages on cluster nodes"*.

If you want to upgrade your cluster, check *Section 29.2.1, "Supported upgrade paths for SLE HA and SLE HA Geo"* and *Section 29.2.2, "Required preparations before upgrading"* before starting to upgrade.

29.1 Terminology

In the following, find definitions of the most important terms used in this chapter:

Major release,
General Availability (GA) version

A major release is a new product version that brings new features and tools, and decommissions previously deprecated components. It comes with backward incompatible changes.

Cluster offline upgrade

If a new product version includes major changes that are backward incompatible, the cluster needs to be upgraded by a cluster offline upgrade. You need to take all nodes offline and upgrade the cluster as a whole, before you can bring all nodes back online.

Cluster rolling upgrade

In a cluster rolling upgrade one cluster node at a time is upgraded while the rest of the cluster is still running. You take the first node offline, upgrade it and bring it back online to join the cluster. Then you continue one by one until all cluster nodes are upgraded to a major version.

Service Pack (SP)

Combines several patches into a form that is easy to install or deploy. Service packs are numbered and usually contain security fixes, updates, upgrades or enhancements of programs.

Update

> Installation of a newer *minor* version of a package, which usually contains security fixes and other important fixes.

Upgrade

> Installation of a newer *major* version of a package or distribution, which brings *new features*. See also *Cluster offline upgrade* versus *Cluster rolling upgrade*.

29.2   Upgrading your cluster to the latest product version

Which upgrade path is supported, and how to perform the upgrade, depends on the current product version and on the target version you want to migrate to.

29.2.1   Supported upgrade paths for SLE HA and SLE HA Geo

SUSE Linux Enterprise High Availability has the same supported upgrade paths as the underlying base system. For a complete overview, see the section *Supported Upgrade Paths to SUSE Linux Enterprise Server 15 SP5* (https://documentation.suse.com/sles-15/html/SLES-all/cha-upgrade-path-s.html#sec-upgrade-paths-supported)↗ in the SUSE Linux Enterprise Server Upgrade Guide.

In addition, the following rules apply, as the High Availability cluster stack offers two methods for upgrading the cluster:

- *Cluster rolling upgrade*. A cluster rolling upgrade is only supported within the same major release (from one service pack to the next, or from the GA version of a product to SP1).

- *Cluster offline upgrade*. A cluster offline upgrade is required to upgrade from one major release to the next (for example, from SLE HA 12 to SLE HA 15) or from a service pack within one major release to the next major release (for example, from SLE HA 12 SP5 to SLE HA 15).

> **⓵ Important: No support for mixed clusters and reversion after upgrade**
>
> - Mixed clusters running on SUSE Linux Enterprise High Availability 12/SUSE Linux Enterprise High Availability 15 are *not* supported.
>
> - After the upgrade process to product version 15, reverting back to product version 12 is *not* supported.

> **🔖 Note: Skipping service packs**
>
> The easiest upgrade path is consecutively installing all service packs. For the SUSE Linux Enterprise 15 product line (GA and the subsequent service packs), skipping up to two service packs is also supported when upgrading. For example, upgrading from SLE HA 15 SP2 to 15 SP5 is supported (as long as SLE HA 15 SP2 is supported).
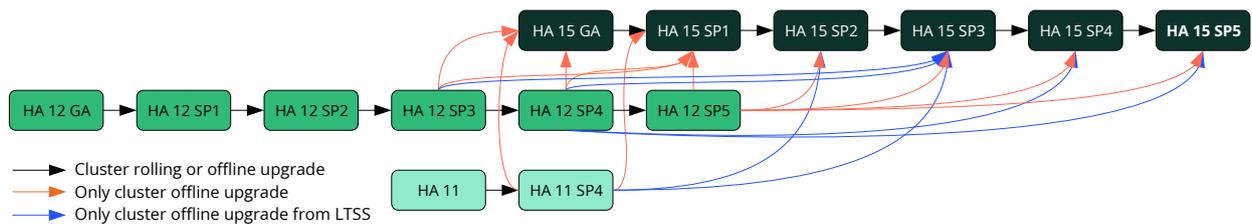
FIGURE 29.1: OVERVIEW OF SUPPORTED UPGRADE PATHS

29.2.2 Required preparations before upgrading

Backup

Ensure that your system backup is up to date and restorable.

Testing

Test the upgrade procedure on a staging instance of your cluster setup first, before performing it in a production environment. This gives you an estimation of the time frame required for the maintenance window. It also helps to detect and solve any unexpected problems.

29.2.3　Cluster offline upgrade

This section describes upgrading from SLE HA 12 to SLE HA 15. Check the supported service packs for upgrade in *Figure 29.1, "Overview of supported upgrade paths"*. If your cluster is still based on an older service pack, first upgrade it to a version of SLES and SLE HA that can be used as a source for upgrading to SLE HA 15. See Upgrading your Cluster to the Latest Product Version (https://documentation.suse.com/sle-ha/12-SP5/html/SLE-HA-all/cha-ha-migration.html#sec-ha-migration-upgrade) ↗ for SLE HA 12 SP5.

PROCEDURE 29.1: UPGRADING FROM PRODUCT VERSION 12 TO 15: CLUSTER OFFLINE UPGRADE

> ❗ **Important: Installation from scratch**
>
> If you decide to install the cluster nodes from scratch (instead of upgrading them), see *Section 2.2, "Software requirements"* for the list of modules required for SUSE Linux Enterprise High Availability 15 SP5. Find more information about modules, extensions, and related products in the release notes for SUSE Linux Enterprise Server 15. They are available at https://www.suse.com/releasenotes/ ↗.

1. Before starting the offline upgrade to SUSE Linux Enterprise High Availability 15, manually upgrade the CIB syntax in your current cluster as described in *Note: Upgrading the CIB syntax version*.

2. Log in to each cluster node and stop the cluster stack with:

   ```
   # crm cluster stop
   ```

3. For each cluster node, perform an upgrade to the desired target version of SUSE Linux Enterprise Server and SUSE Linux Enterprise High Availability—see *Section 29.2.1, "Supported upgrade paths for SLE HA and SLE HA Geo"*.

4. After the upgrade process has finished, log in to each node and boot it with the upgraded version of SUSE Linux Enterprise Server and SUSE Linux Enterprise High Availability.

5. If you use Cluster LVM, you need to migrate from clvmd to lvmlockd. See the man page of `lvmlockd`, section *Changing a clvm/clustered VG to a shared VG*.
   If you also use `cmirrord`, we highly recommend migrating to Cluster MD. See *Section 24.4, "Online migration from mirror LV to cluster MD"*.

6. Log in to a cluster node and start the cluster stack on all nodes:

```
# crm cluster start --all
```

> 📝 **Note: `--all` option only available in version 15 SP4**
>
> The `--all` option was added in SUSE Linux Enterprise High Availability 15 SP4. In earlier versions, you must run the `crm cluster start` command on each node one by one.

7. Check the cluster status with `crm status` or with Hawk2.

> 📝 **Note: Upgrading the CIB syntax version**
>
> Sometimes new features are only available with the latest CIB syntax version. When you upgrade to a new product version, your CIB syntax version will *not* be upgraded by default.
>
> 1. Check your version with:
>
>    ```
>    cibadmin -Q | grep validate-with
>    ```
>
> 2. Upgrade to the latest CIB syntax version with:
>
>    ```
>    # cibadmin --upgrade --force
>    ```

29.2.4 Cluster rolling upgrade

This section describes upgrading to a new SLE HA 15 service pack. Check the supported service packs for upgrade in *Figure 29.1, "Overview of supported upgrade paths"*.

Use one of the following procedures for your scenario:

- *Procedure 29.2, "Performing a cluster rolling upgrade"*

- *Procedure 29.3, "Performing a cluster-wide fresh installation of a new service pack"*

> 🖐 **Warning: Active cluster stack**
>
> Before starting an upgrade for a node, *stop* the cluster stack *on that node*.

If the cluster resource manager on a node is active during the software update, this can lead to results such as fencing of active nodes.

> ⓘ **Important: Time limit for cluster rolling upgrade**
>
> The new features shipped with the latest product version will only be available after *all* cluster nodes have been upgraded to the latest product version. Mixed version clusters are only supported for a short time frame during the cluster rolling upgrade. Complete the cluster rolling upgrade within one week.
>
> Once all the online nodes are running the upgraded version, it is not possible for any other nodes with the old version to (re-)join without having been upgraded.

PROCEDURE 29.2: PERFORMING A CLUSTER ROLLING UPGRADE

1. Log in as `root` on the node that you want to upgrade and stop the cluster stack:

   ```
   # crm cluster stop
   ```

2. Perform an upgrade to the desired target version of SUSE Linux Enterprise Server and the SUSE Linux Enterprise High Availability extension as described in Upgrade Guide for SLES 15 SP5 (https://documentation.suse.com/sles/html/SLES-all/book-upgrade.html) ↗.
   If you need to upgrade a Geo cluster, see *Book "Geo Clustering Guide", Chapter 10 "Upgrading to the latest product version"*.

3. Start the cluster stack on the upgraded node to make the node rejoin the cluster:

   ```
   # crm cluster start
   ```

4. Take the next node offline and repeat the procedure for that node.

5. Check the cluster status with `crm status` or with Hawk2.
   The Hawk2 *Status* screen also shows a warning if different CRM versions are detected for your cluster nodes.

> ⓘ **Important: Time limit for rolling upgrade**
>
> The new features shipped with the latest product version will only be available after *all* cluster nodes have been upgraded to the latest product version. Mixed version clusters are only supported for a short time frame during the rolling upgrade. Complete the rolling upgrade within one week.

The Hawk2 *Status* screen also shows a warning if different CRM versions are detected for your cluster nodes.

Beside an in-place upgrade, many customers prefer a fresh installation even for moving to the next service pack. The following procedure shows a scenario where a two-node cluster with the nodes alice and bob is upgraded to the next service pack (SP):

PROCEDURE 29.3: PERFORMING A CLUSTER-WIDE FRESH INSTALLATION OF A NEW SERVICE PACK

1. Make a backup of your cluster configuration. A minimum set of files is shown in the following list:

    ```
    /etc/corosync/corosync.conf
    /etc/corosync/authkey
    /etc/sysconfig/sbd
    /etc/modules-load.d/watchdog.conf
    /etc/hosts
    /etc/chrony.conf
    ```

    Depending on your resources, you may also need the following files:

    ```
    /etc/services
    /etc/passwd
    /etc/shadow
    /etc/groups
    /etc/drbd/*
    /etc/lvm/lvm.conf
    /etc/mdadm.conf
    /etc/mdadm.SID.conf
    ```

2. Start with node alice.

    a. Put the node into standby node. That way, resources can move off the node:

       ```
       # crm --wait node standby alice reboot
       ```

With the option `--wait`, the command returns only when the cluster finishes the transition and becomes idle. The `reboot` option has the effect that the node will be already out of standby mode when it is online again. Despite its name, the `reboot` option works as long as the node goes offline and online.

    b. Stop the cluster services on node alice:

```
# crm cluster stop
```

    c. At this point, alice does not have running resources anymore. Upgrade the node alice and reboot it afterward. Cluster services are assumed not to start on boot.

    d. Copy your backup files from *Step 1* to the original places.

    e. Bring back node alice into cluster:

```
# crm cluster start
```

    f. Check that resources are fine.

3. Repeat *Step 2* for node bob.

29.3   Updating software packages on cluster nodes

✋ Warning: Active cluster stack

Before starting a package update for a node, either *stop* the cluster stack *on that node* or put the *node into maintenance mode,* depending on whether the cluster stack is affected or not. See *Step 1* for details.

If the cluster resource manager on a node is active during the software update, this can lead to results such as fencing of active nodes.

1. Before installing any package updates on a node, check the following:

   - Does the update affect any packages belonging to SUSE Linux Enterprise High Availability? If `yes`, stop the cluster stack on the node before starting the software update:

     ```
     # crm cluster stop
     ```

   - Does the package update require a reboot? If `yes`: Stop the cluster stack on the node before starting the software update:

     ```
     # crm cluster stop
     ```

   - If none of the situations above apply, you do not need to stop the cluster stack. In that case, put the node into maintenance mode before starting the software update:

     ```
     # crm node maintenance NODE_NAME
     ```

     For more details on maintenance mode, see *Section 28.2, "Different options for maintenance tasks"*.

2. Install the package update using either YaST or Zypper.

3. After the update has been successfully installed:

   - Either start the cluster stack on the respective node (if you stopped it in *Step 1*):

     ```
     # crm cluster start
     ```

   - or remove the maintenance flag to bring the node back to normal mode:

     ```
     # crm node ready NODE_NAME
     ```

4. Check the cluster status with `crm status` or with Hawk2.

29.4 For more information

For detailed information about any changes and new features of the product you are upgrading to, refer to its release notes. They are available from https://www.suse.com/releasenotes/ ↗.

V   Appendix

A   Troubleshooting

Strange problems may occur that are not easy to understand, especially when starting to experiment with High Availability. However, there are several utilities that allow you to take a closer look at the High Availability internal processes. This chapter recommends various solutions.

A.1   Installation and first steps

Troubleshooting difficulties when installing the packages or bringing the cluster online.

Are the HA packages installed?

> The packages needed for configuring and managing a cluster are included in the `High Availability` installation pattern, available with SUSE Linux Enterprise High Availability.
>
> Check if SUSE Linux Enterprise High Availability is installed on each of the cluster nodes and if the *High Availability* pattern is installed on each of the machines as described in the Installation and Setup Quick Start.

Is the initial configuration the same for all cluster nodes?

> To communicate with each other, all nodes belonging to the same cluster need to use the same `bindnetaddr`, `mcastaddr` and `mcastport` as described in *Chapter 4, Using the YaST cluster module*.
>
> Check if the communication channels and options configured in `/etc/corosync/corosync.conf` are the same for all cluster nodes.
>
> In case you use encrypted communication, check if the `/etc/corosync/authkey` file is available on all cluster nodes.
>
> All `corosync.conf` settings except for `nodeid` must be the same; `authkey` files on all nodes must be identical.

Does the firewall allow communication via the `mcastport`?

> If the mcastport used for communication between the cluster nodes is blocked by the firewall, the nodes cannot see each other. When doing the initial setup with YaST or the bootstrap scripts (as described in *Chapter 4, Using the YaST cluster module* or the *Article "Installation and Setup Quick Start"*, respectively), the firewall settings are usually automatically adjusted.

To make sure the mcastport is not blocked by the firewall, check the firewall settings on each node.

Are Pacemaker and Corosync started on each cluster node?

Usually, starting Pacemaker also starts the Corosync service. To check if both services are running:

```
# crm cluster status
```

In case they are not running, start them by executing the following command:

```
# crm cluster start
```

A.2   Logging

Where to find the log files?

Pacemaker writes its log files into the `/var/log/pacemaker` directory. The main Pacemaker log file is `/var/log/pacemaker/pacemaker.log`. In case you cannot find the log files, check the logging settings in `/etc/sysconfig/pacemaker`, Pacemaker's own configuration file. If `PCMK_logfile` is configured there, Pacemaker uses the path that is defined by this parameter.

If you need a cluster-wide report showing all relevant log files, see *How can I create a report with an analysis of all my cluster nodes?* for more information.

I enabled monitoring but there is no trace of monitoring operations in the log files?

The `pacemaker-execd` daemon does not log recurring monitor operations unless an error occurred. Logging all recurring operations would produce too much noise. Therefore recurring monitor operations are logged only once an hour.

I only get a `failed` message. Is it possible to get more information?

Add the `--verbose` parameter to your commands. If you do that multiple times, the debug output becomes more verbose. See the logging data (**sudo journalctl -n**) for useful hints.

How can I get an overview of all my nodes and resources?

Use the **crm_mon** command. The following displays the resource operation history (option `-o`) and inactive resources (`-r`):

```
# crm_mon -o -r
```

The display is refreshed when the status changes (to cancel this press `Ctrl`–`C`). An example may look like:

EXAMPLE A.1: STOPPED RESOURCES

```
Last updated: Fri Aug 15 10:42:08 2014
Last change: Fri Aug 15 10:32:19 2014
 Stack: corosync
Current DC: bob (175704619) - partition with quorum
Version: 1.1.12-ad083a8
2 Nodes configured
3 Resources configured


Online: [ alice bob ]


Full list of resources:


my_ipaddress    (ocf:heartbeat:Dummy): Started bob
my_filesystem   (ocf:heartbeat:Dummy): Stopped
my_webserver    (ocf:heartbeat:Dummy): Stopped


Operations:
* Node bob:
   my_ipaddress: migration-threshold=3
     + (14) start: rc=0 (ok)
     + (15) monitor: interval=10000ms rc=0 (ok)
     * Node alice:
```

The *Pacemaker Explained* PDF, available at https://www.clusterlabs.org/pacemaker/doc/ ↗, covers three different recovery types in the *How are OCF Return Codes Interpreted?* section.

How to view logs?

For a more detailed view of what is happening in your cluster, use the following command:

```
# crm history log [NODE]
```

Replace *NODE* with the node you want to examine, or leave it empty. See *Section A.5, "History"* for further information.

A.3 Resources

How can I clean up my resources?

Use the following commands:

```
# crm resource list
```

```
# crm resource cleanup rscid [node]
```

If you leave out the node, the resource is cleaned on all nodes. More information can be found in *Section 8.5.2, "Cleaning up cluster resources with crmsh"*.

How can I list my currently known resources?

Use the command `crm resource list` to display your current resources.

I configured a resource, but it always fails. Why?

To check an OCF script use `ocf-tester`, for example:

```
ocf-tester -n ip1 -o ip=YOUR_IP_ADDRESS \
   /usr/lib/ocf/resource.d/heartbeat/IPaddr
```

Use `-o` multiple times for more parameters. The list of required and optional parameters can be obtained by running `crm ra info AGENT`, for example:

```
# crm ra info ocf:heartbeat:IPaddr
```

Before running ocf-tester, make sure the resource is not managed by the cluster.

Why do resources not fail over and why are there no errors?

The terminated node might be considered unclean. Then it is necessary to fence it. If the STONITH resource is not operational or does not exist, the remaining node waits for the fencing to happen. The fencing timeouts are typically high, so it might take a while to see any obvious sign of problems (if ever).

Yet another possible explanation is that a resource is simply not allowed to run on this node. That may be because of a failure which happened in the past and which was not "cleaned". Or it may be because of an earlier administrative action, that is a location constraint with a negative score. Such a location constraint is inserted by the `crm resource move` command, for example.

Why can I never tell where my resource will run?

If there are no location constraints for a resource, its placement is subject to an (almost) random node choice. You are well advised to always express a preferred node for resources. That does not mean that you need to specify location preferences for *all* resources. One preference suffices for a set of related (colocated) resources. A node preference looks like this:

```
location rsc-prefers-alice rsc 100: alice
```

A.4 STONITH and fencing

Why does my STONITH resource not start?

A start (or enable) operation includes checking the status of the device. If the device is not ready, the STONITH resource fails to start.

At the same time, the STONITH plugin is asked to produce a host list. If this list is empty, there is no point in running a STONITH resource which cannot shoot anything. The name of the host on which STONITH is running is filtered from the list, since the node cannot shoot itself.

To use single-host management devices such as lights-out devices, make sure that the STONITH resource is *not* allowed to run on the node which it is supposed to fence. Use an infinitely negative location node preference (constraint). The cluster will move the STONITH resource to another place where it can start, but not before informing you.

Why does fencing not happen, although I have the STONITH resource?

Each STONITH resource must provide a host list. This list may be inserted by hand in the STONITH resource configuration or retrieved from the device itself from outlet names, for example. That depends on the nature of the STONITH plugin. `pacemaker-fenced` uses the list to find out which STONITH resource can fence the target node. Only if the node appears in the list can the STONITH resource shoot (fence) the node.

If `pacemaker-fenced` does not find the node in any of the host lists provided by running STONITH resources, it asks `pacemaker-fenced` instances on other nodes. If the target node does not show up in the host lists of other `pacemaker-fenced` instances, the fencing request ends in a timeout at the originating node.

Why does my STONITH resource fail occasionally?

Power management devices may give up if there is too much broadcast traffic. Space out the monitor operations. Given that fencing is necessary only occasionally (and hopefully never), checking the device status once every few hours is more than enough.

Also, some of these devices may refuse to talk to more than one party at the same time. This may be a problem if you keep a terminal or browser session open while the cluster tries to test the status.

A.5 History

How to retrieve status information or a log from a failed resource?

Use the `history` command and its subcommand `resource`:

```
# crm history resource NAME1
```

This gives you a full transition log for the given resource only. However, it is possible to investigate more than one resource. Append the resource names after the first.

If you followed naming conventions (see *Appendix B, Naming conventions*), the `resource` command makes it easier to investigate a group of resources. For example, this command investigates all primitives starting with `db`:

```
# crm history resource db*
```

View the log file in `/var/cache/crm/history/live/alice/ha-log.txt`.

How can I reduce the history output?

There are two options for the `history` command:

- Use `exclude`

- Use `timeframe`

The `exclude` command let you set an additive regular expression that excludes certain patterns from the log. For example, the following command excludes all SSH, `systemd`, and kernel messages:

```
# crm history exclude ssh|systemd|kernel.
```

With the `timeframe` command you limit the output to a certain range. For example, the following command shows all the events on August 23 from 12:00 to 12:30:

```
# crm history timeframe "Aug 23 12:00" "Aug 23 12:30"
```

How can I store a "session" for later inspection?

When you encounter a bug or an event that needs further examination, it is useful to store all the current settings. This file can be sent to support or viewed with `bzless`. For example:

```
crm(live)history# timeframe "Oct 13 15:00" "Oct 13 16:00"
crm(live)history# session save tux-test
crm(live)history# session pack
```

```
Report saved in '/root/tux-test.tar.bz2'
```

A.6 Hawk2

Replacing the self-signed certificate

To avoid the warning about the self-signed certificate on first Hawk2 start-up, replace the automatically created certificate with your own certificate (or a certificate that was signed by an official Certificate Authority, CA):

1. Replace `/etc/hawk/hawk.key` with the private key.

2. Replace `/etc/hawk/hawk.pem` with the certificate that Hawk2 should present.

3. Restart the Hawk2 services to reload the new certificate:

   ```
   # systemctl restart hawk-backend hawk
   ```

Change ownership of the files to `root:haclient` and make the files accessible to the group:

```
# chown root:haclient /etc/hawk/hawk.key /etc/hawk/hawk.pem
# chmod 640 /etc/hawk/hawk.key /etc/hawk/hawk.pem
```

A.7 Miscellaneous

How can I run commands on all cluster nodes?

Use the command `crm cluster run` for this task. For example:

```
# crm cluster run "ls -l /etc/corosync/*.conf"
INFO: [alice]
-rw-r--r-- 1 root root 812 Oct 27 15:42 /etc/corosync/corosync.conf
INFO: [bob]
-rw-r--r-- 1 root root 812 Oct 27 15:42 /etc/corosync/corosync.conf
INFO: [charlie]
-rw-r--r-- 1 root root 812 Oct 27 15:42 /etc/corosync/corosync.conf
```

By default, the specified command runs on all nodes in the cluster. Alternatively, you can run the command on a specific node or group of nodes:

```
# crm cluster run "ls -l /etc/corosync/*.conf" alice bob
```

What is the state of my cluster?

To check the current state of your cluster, use one of the programs `crm_mon` or `crm status`. This displays the current DC and all the nodes and resources known by the current node.

Why can several nodes of my cluster not see each other?

There could be several reasons:

- Look first in the configuration file `/etc/corosync/corosync.conf`. Check if the multicast or unicast address is the same for every node in the cluster (look in the `interface` section with the key `mcastaddr`).

- Check your firewall settings.

- Check if your switch supports multicast or unicast addresses.

- Check if the connection between your nodes is broken. Most often, this is the result of a badly configured firewall. This also may be the reason for a *split-brain* condition, where the cluster is partitioned.

Why can an OCFS2 device not be mounted?

Check the log messages (`sudo journalctl -n`) for the following line:

```
Jan 12 09:58:55 alice pacemaker-execd: [3487]: info: RA output: [...]
  ERROR: Could not load ocfs2_stackglue
Jan 12 16:04:22 alice modprobe: FATAL: Module ocfs2_stackglue not found.
```

In this case the Kernel module `ocfs2_stackglue.ko` is missing. Install the package `ocfs2-kmp-default`, `ocfs2-kmp-pae` or `ocfs2-kmp-xen`, depending on the installed Kernel.

How can I create a report with an analysis of all my cluster nodes?

On the crm shell, use `crm report` to create a report. This tool compiles:

- Cluster-wide log files,

- Package states,

- DLM/OCFS2 states,

- System information,

- CIB history,

- Parsing of core dump reports, if a debuginfo package is installed.

Usually run **crm report** with the following command:

```
# crm report -f 0:00 -n alice -n bob
```

The command extracts all information since 0am on the hosts alice and bob and creates a `*.tar.bz2` archive named `crm_report-DATE.tar.bz2` in the current directory, for example, `crm_report-Wed-03-Mar-2012`. If you are only interested in a specific time frame, add the end time with the `-t` option.

> ✋ ## Warning: Remove sensitive information
>
> The **crm report** tool tries to remove any sensitive information from the CIB and the PE input files, however, it cannot know everything. If you have more sensitive information, supply additional patterns with the `-p` option (see man page). The log files and the **crm_mon**, **ccm_tool**, and **crm_verify** output are *not* sanitized.
>
> Before sharing your data in any way, check the archive and remove all information you do not want to expose.

Customize the command execution with further options. For example, if you have a Pacemaker cluster, you certainly want to add the option `-A`. In case you have another user who has permissions to the cluster, use the `-u` option and specify this user (in addition to `root` and `hacluster`). In case you have a non-standard SSH port, use the `-X` option to add the port (for example, with the port 3479, use `-X "-p 3479"`). Further options can be found in the man page of **crm report**.

After **crm report** has analyzed all the relevant log files and created the directory (or archive), check the log files for an uppercase `ERROR` string. The most important files in the top level directory of the report are:

`analysis.txt`

Compares files that should be identical on all nodes.

`corosync.txt`

Contains a copy of the Corosync configuration file.

`crm_mon.txt`

Contains the output of the **crm_mon** command.

`description.txt`

Contains all cluster package versions on your nodes. There is also the `sysinfo.txt` file which is node specific. It is linked to the top directory.

This file can be used as a template to describe the issue you encountered and post it to https://github.com/ClusterLabs/crmsh/issues ↗.

`members.txt`

A list of all nodes

`sysinfo.txt`

Contains a list of all relevant package names and their versions. Additionally, there is also a list of configuration files which are different from the original RPM package.

Node-specific files are stored in a subdirectory named by the node's name. It contains a copy of the directory `/etc` of the respective node.

In case you need to simplify the arguments, set your default values in the configuration file `/etc/crm/crm.conf`, section `report`. Further information is written in the man page **man 8 crmsh_hb_report**.

A.8 For more information

For additional information about high availability on Linux, including configuring cluster resources and managing and customizing a High Availability cluster, see https://clusterlabs.org/wiki/Documentation ↗.

B Naming conventions

This guide uses the following naming conventions for cluster nodes and names, cluster resources, and constraints.

Cluster nodes

Cluster nodes use first names:

alice, bob, charlie, doro, and eris

Cluster site names

Cluster sites are named after cities:

amsterdam, berlin, canberra, dublin, fukuoka, gizeh, hanoi, and istanbul

Cluster resources

Primitives	No prefix
Groups	Prefix `g-`
Clones	Prefix `cl-`
Promotable Clones	(formerly known as multi-state resources) Prefix `ms-`

Constraints

Order constraints	Prefix `o-`
Location constraints	Prefix `loc-`
Colocation constraints	Prefix `col-`

C Cluster management tools (command line)

SUSE Linux Enterprise High Availability ships with a comprehensive set of tools to assists you in managing your cluster from the command line. This chapter introduces the tools needed for managing the cluster configuration in the CIB and the cluster resources. Other command line tools for managing resource agents or tools used for debugging (and troubleshooting) your setup are covered in *Appendix A, Troubleshooting*.

> ### Note: Use crmsh
>
> This tool is for experts only. Usually the crm shell (crmsh) is the recommended way of managing your cluster.

The following list presents several tasks related to cluster management and briefly introduces the tools to use to accomplish these tasks:

Monitoring the cluster's status

The `crm_mon` command allows you to monitor your cluster's status and configuration. Its output includes the number of nodes, uname, UUID, status, the resources configured in your cluster, and the current status of each. The output of `crm_mon` can be displayed at the console or printed into an HTML file. When provided with a cluster configuration file without the status section, `crm_mon` creates an overview of nodes and resources as specified in the file. See the `crm_mon` man page for a detailed introduction to this tool's usage and command syntax.

Managing the CIB

The `cibadmin` command is the low-level administrative command for manipulating the CIB. It can be used to dump all or part of the CIB, update all or part of it, modify all or part of it, delete the entire CIB, or perform miscellaneous CIB administrative operations. See the `cibadmin` man page for a detailed introduction to this tool's usage and command syntax.

Managing configuration changes

The `crm_diff` command assists you in creating and applying XML patches. This can be useful for visualizing the changes between two versions of the cluster configuration or saving changes so they can be applied at a later time using `cibadmin`. See the `crm_diff` man page for a detailed introduction to this tool's usage and command syntax.

Manipulating CIB attributes

The `crm_attribute` command lets you query and manipulate node attributes and cluster configuration options that are used in the CIB. See the `crm_attribute` man page for a detailed introduction to this tool's usage and command syntax.

Validating the cluster configuration

The `crm_verify` command checks the configuration database (CIB) for consistency and other problems. It can check a file containing the configuration or connect to a running cluster. It reports two classes of problems. Errors must be fixed before the High Availability software can work properly while warning resolution is up to the administrator. `crm_verify` assists in creating new or modified configurations. You can take a local copy of a CIB in the running cluster, edit it, validate it using `crm_verify`, then put the new configuration into effect using `cibadmin`. See the `crm_verify` man page for a detailed introduction to this tool's usage and command syntax.

Managing resource configurations

The `crm_resource` command performs various resource-related actions on the cluster. It lets you modify the definition of configured resources, start and stop resources, or delete and migrate resources between nodes. See the `crm_resource` man page for a detailed introduction to this tool's usage and command syntax.

Managing resource fail counts

The `crm_failcount` command queries the number of failures per resource on a given node. This tool can also be used to reset the fail count, allowing the resource to again run on nodes where it had failed too often. See the `crm_failcount` man page for a detailed introduction to this tool's usage and command syntax.

Managing a node's standby status

The `crm_standby` command can manipulate a node's standby attribute. Any node in standby mode is no longer eligible to host resources and any resources that are there must be moved. Standby mode can be useful for performing maintenance tasks, such as Kernel updates. Remove the standby attribute from the node for it to become a fully active member of the cluster again. See the `crm_standby` man page for a detailed introduction to this tool's usage and command syntax.

D Running cluster reports without `root` access

All cluster nodes must be able to access each other via SSH. Tools like `crm report` (for troubleshooting) and Hawk2's *History Explorer* require passwordless SSH access between the nodes, otherwise they can only collect data from the current node.

If your security policy does not allow passwordless `root` SSH login, or prevents `root` SSH login entirely, then running `crm report` as `root` fails on all remote nodes. If the cluster was initialized by a non-root user with `sudo` privileges, this user can run cluster reports successfully under these restrictions. However, if the cluster was initialized by the `root` user, you must use the following workaround to run cluster reports:

1. Creating a dedicated local user account (for running `crm report`).

2. Configuring passwordless SSH access for that user account, ideally by using a non-standard SSH port.

3. Configuring `sudo` for that user.

4. Running `crm report` as that user.

D.1 Creating a local user account

The following example creates a local user named `hareport` from command line. The password can be anything that meets your security requirements. Alternatively, you can create the user account and set the password with YaST.

PROCEDURE D.1: CREATING A DEDICATED USER ACCOUNT FOR RUNNING CLUSTER REPORTS

1. Start a shell and create a user `hareport` with a home directory `/home/hareport`:

   ```
   # useradd -m -d /home/hareport -c "HA Report" hareport
   ```

2. Set a password for the user:

   ```
   # passwd hareport
   ```

3. When prompted, enter and re-enter a password for the user.

> **❗ Important: Same user is required on each cluster node**
>
> To create the same user account on all nodes, repeat the steps above on each cluster node.

D.2   Configuring a passwordless SSH account

By default, the SSH daemon and the SSH client talk and listen on port `22`. If your network security guidelines require the default SSH port to be changed to an alternate high numbered port, you need to modify the daemon's configuration file `/etc/ssh/sshd_config`.

1. To modify the default port, search the file for the `Port` line, uncomment it and edit it according to your wishes. For example, set it to:

```
Port 5022
```

2. If your organization does not permit the `root` user to access other servers, search the file for the `PermitRootLogin` entry, uncomment it and set it to `no`:

```
PermitRootLogin no
```

3. Alternatively, add the respective lines to the end of the file by executing the following commands:

```
# echo "PermitRootLogin no" >> /etc/ssh/sshd_config
# echo "Port 5022" >> /etc/ssh/sshd_config
```

4. After modifying `/etc/ssh/sshd_config`, restart the SSH daemon to make the new settings take effect:

```
# systemctl restart sshd
```

> ❗ **Important: Same settings are required on each cluster node**
>
> Repeat the SSH daemon configuration above on each cluster node.

PROCEDURE D.3: CONFIGURING THE SSH CLIENT FOR A NON-STANDARD PORT

If the SSH port change is going to be made on all nodes in the cluster, it is useful to modify the SSH configuration file, `/etc/ssh/sshd_config`.

1. To modify the default port, search the file for the `Port` line, uncomment it and edit it according to your wishes. For example, set it to:

```
Port 5022
```

2. Alternatively, add the respective line to the end of the file by executing the following commands:

```
# echo "Port 5022" >> /etc/ssh/ssh_config
```

Note: Settings only required on one node

The SSH client configuration above is only needed on the node on which you want to run the cluster report.

Alternatively, you can use the `-X` option to run the `crm report` with a custom SSH port or even make `crm report` use your custom SSH port by default. For details, see *Procedure D.5, "Generating a cluster report using a custom SSH port"*.

PROCEDURE D.4: CONFIGURING SHARED SSH KEYS

You can access other servers using SSH and not be asked for a password. While this may appear insecure, it is actually a secure access method since the users can only access servers that their public key has been shared with. The shared key must be created as the user that will use the key.

1. Log in to one of the nodes with the user account that you have created for running cluster reports (in our example above, the user account was `hareport`).

2. Generate a new key:

```
hareport > ssh-keygen -t rsa
```

This command generates a 2048 bit key by default. The default location for the key is `~/.ssh/`. You are asked to set a passphrase on the key. However, do not enter a passphrase because for passwordless login there must not be a passphrase on the key.

3. After the keys have been generated, copy the public key to *each* of the other nodes (*including* the node where you created the key):

```
hareport > ssh-copy-id -i ~/.ssh/id_rsa.pub HOSTNAME_OR_IP
```

In the command, you can either use the DNS name for each server, an alias, or the IP address. During the copy process you will be asked to accept the host key for each node, and you will need to provide the password for the `hareport` user account (this will be the only time you need to enter it).

4. After the key is shared to all cluster nodes, test if you can log in as user `hareport` to the other nodes by using passwordless SSH:

```
hareport > ssh HOSTNAME_OR_IP
```

You should be automatically connected to the remote server without being asked to accept a certificate or enter a password.

Note: Settings only required on one node

If you intend to run the cluster report from the same node each time, it is sufficient to execute the procedure above on this node only. Otherwise repeat the procedure on each node.

D.3 Configuring **sudo**

The **sudo** command allows a regular user to quickly become `root` and issue a command, with or without providing a password. Sudo access can be given to all root-level commands or to specific commands only. Sudo typically uses aliases to define the entire command string.

To configure sudo either use **visudo** (*not* vi) or YaST.

Warning: Do not use vi

For sudo configuration from command line, you must edit the sudoers file as `root` with **visudo**. Using any other editor may result in syntax or file permission errors that prevent sudo from running.

1. Log in as `root`.

2. To open the `/etc/sudoers` file, enter **visudo**.

3. Look for the following categories: `Host alias specification`, `User alias specification`, `Cmnd alias specification`, and `Runas alias specification`.

4. Add the following entries to the respective categories in `/etc/sudoers`:

```
Host_Alias CLUSTER = alice,bob,charlie ❶
User_Alias HA = hareport ❷
Cmnd_Alias HA_ALLOWED = /bin/su, /usr/sbin/crm report *❸
```

```
Runas_Alias R = root ❹
```

❶ The host alias defines on which server (or range of servers) the sudo user has rights to issue commands. In the host alias you can use DNS names, or IP addresses, or specify an entire network range (for example, `172.17.12.0/24`). To limit the scope of access you should specify the host names for the cluster nodes only.

❷ The user alias allows you to add multiple local user accounts to a single alias. However, in this case you could avoid creating an alias since only one account is being used. In the example above, we added the `hareport` user which we have created for running cluster reports.

❸ The command alias defines which commands can be executed by the user. This is useful if you want to limit what the non-root user can access when using **sudo**. In this case the `hareport` user account will need access to the commands **crm report** and **su**.

❹ The `runas` alias specifies the account that the command will be run as. In this case `root`.

5. Search for the following two lines:

```
Defaults targetpw
ALL     ALL=(ALL) ALL
```

As they would conflict with the setup we want to create, disable them:

```
#Defaults targetpw
#ALL     ALL=(ALL) ALL
```

6. Look for the `User privilege specification` category. After having defined the aliases above, you can now add the following rule there:

```
HA CLUSTER = (R) NOPASSWD:HA_ALLOWED
```

The `NOPASSWORD` option ensures that the user `hareport` can execute the cluster report without providing a password.

7. *(Optional)* To allow the user `hareport` to run cluster reports using your local SSH keys, add the following line to the `Defaults specification` category. This preserves the `SSH_AUTH_SOCK` environment variable, which is required for SSH agent forwarding.

```
Defaults!HA_ALLOWED env_keep+=SSH_AUTH_SOCK
```

When you log in to a node as the user `hareport` via **ssh -A** and use **sudo** to run **crm report**, your local SSH keys are passed to the node for authentication.

> ◈ **Note: SSH agent forwarding for crm report only**
>
> SSH agent forwarding is only supported for `crm report`, not for any other crmsh functions.

> ❗ **Important: Same sudo configuration is required on each cluster node**
>
> This sudo configuration must be made on all nodes in the cluster. No other changes are needed for sudo and no services need to be restarted.

D.4 Generating a cluster report

To run cluster reports with the settings you have configured above, you need to be logged in to one of the nodes as user `hareport`. To start a cluster report, use the **crm report** command. For example:

```
hareport > sudo crm report -f 0:00 -n "alice bob charlie"
```

This command extracts all information since `0 am` on the named nodes and create a `*.tar.bz2` archive named `pcmk-DATE.tar.bz2` in the current directory.

PROCEDURE D.5: GENERATING A CLUSTER REPORT USING A CUSTOM SSH PORT

1. When using a custom SSH port, use the `-X` with **crm report** to modify the client's SSH port. For example, if your custom SSH port is `5022`, use the following command:

   ```
   # crm report -X "-p 5022" [...]
   ```

2. To set your custom SSH port permanently for **crm report**, start the interactive crm shell:

   ```
   # crm options
   ```

3. Enter the following:

   ```
   crm(live)options#  set core.report_tool_options "-X -oPort=5022"
   ```

Glossary

active/active, active/passive

A concept of how services are running on nodes. An active-passive scenario means that one or more services are running on the active node and the passive node waits for the active node to fail. Active-active means that each node is active and passive at the same time. For example, it has *some* services running, but can take over other services from the other node. Compare with primary/secondary and dual-primary in DRBD speak.

arbitrator

Additional instance in a Geo cluster that helps to reach consensus about decisions such as failover of resources across sites. Arbitrators are single machines that run one or more booth instances in a special mode.

AutoYaST

AutoYaST is a system for installing one or more SUSE Linux Enterprise systems automatically and without user intervention.

bindnetaddr (bind network address)

The network address the Corosync executive should bind to.

booth

The instance that manages the failover process between the sites of a Geo cluster. It aims to get multi-site resources active on one and only one site. This is achieved by using so-called tickets that are treated as failover domain between cluster sites, in case a site should be down.

boothd (booth daemon)

Each of the participating clusters and arbitrators in a Geo cluster runs a service, the `boothd`. It connects to the booth daemons running at the other sites and exchanges connectivity details.

CCM (consensus cluster membership)

The CCM determines which nodes make up the cluster and shares this information across the cluster. Any new addition and any loss of nodes or quorum is delivered by the CCM. A CCM module runs on each node of the cluster.

CIB (cluster information base)

A representation of the whole cluster configuration and status (cluster options, nodes, resources, constraints and the relationship to each other). It is written in XML and resides in memory. A primary CIB is kept and maintained on the *DC (designated coordinator)* and replicated to the other nodes. Normal read and write operations on the CIB are serialized through the primary CIB.

cluster

A *high-performance* cluster is a group of computers (real or virtual) sharing the application load to achieve faster results. A *high-availability* cluster is designed primarily to secure the highest possible availability of services.

cluster partition

Whenever communication fails between one or more nodes and the rest of the cluster, a cluster partition occurs. The nodes of a cluster are split into partitions but still active. They can only communicate with nodes in the same partition and are unaware of the separated nodes. As the loss of the nodes on the other partition cannot be confirmed, a split-brain scenario develops (see also *split-brain*).

cluster site

In Geo clustering, a cluster site (or just "site") is a group of nodes in the same physical location, managed by *booth*.

cluster stack

The ensemble of software technologies and components that compose a cluster.

concurrency violation

A resource that should be running on only one node in the cluster is running on several nodes.

conntrack tools

Allow interaction with the in-kernel connection tracking system for enabling *stateful* packet inspection for iptables. Used by SUSE Linux Enterprise High Availability to synchronize the connection status between cluster nodes.

CRM (cluster resource manager)

The management entity responsible for coordinating all non-local interactions in a High Availability cluster. SUSE Linux Enterprise High Availability uses Pacemaker as CRM. The CRM is implemented as `pacemaker-controld`. It interacts with several components: local

resource managers, both on its own node and on the other nodes, non-local CRMs, administrative commands, the fencing functionality, and the membership layer.

crmsh

The command line utility crmsh manages your cluster, nodes, and resources.

See *Section 5.5, "Introduction to crmsh"* for more information.

Csync2

A synchronization tool that can be used to replicate configuration files across all nodes in the cluster, and even across Geo clusters.

DC (designated coordinator)

The DC is elected from all nodes in the cluster. This happens if there is no DC yet or if the current DC leaves the cluster for any reason. The DC is the only entity in the cluster that can decide that a cluster-wide change needs to be performed, such as fencing a node or moving resources around. All other nodes get their configuration and resource allocation information from the current DC.

Disaster

Unexpected interruption of critical infrastructure induced by nature, humans, hardware failure, or software bugs.

Disaster Recover Plan

A strategy to recover from a disaster with minimum impact on IT infrastructure.

Disaster Recovery

Disaster recovery is the process by which a business function is restored to the normal, steady state after a disaster.

DLM (distributed lock manager)

DLM coordinates disk access for clustered file systems and administers file locking to increase performance and availability.

DRBD

DRBD® is a block device designed for building high availability clusters. The whole block device is mirrored via a dedicated network and is seen as a network RAID-1.

existing cluster

The term "existing cluster" is used to refer to any cluster that consists of at least one node. Existing clusters have a basic Corosync configuration that defines the communication channels, but they do not necessarily have resource configuration yet.

failover

Occurs when a resource or node fails on one machine and the affected resources are started on another node.

failover domain

A named subset of cluster nodes that are eligible to run a cluster service if a node fails.

fencing

Describes the concept of preventing access to a shared resource by isolated or failing cluster members. There are two classes of fencing: resource level fencing and node level fencing. Resource level fencing ensures exclusive access to a given resource. Node level fencing prevents a failed node from accessing shared resources entirely and prevents resources from running on a node whose status is uncertain. This is usually done in a simple and abrupt way: reset or power off the node.

Geo cluster (geographically dispersed cluster)

Consists of multiple, geographically dispersed sites with a local cluster each. The sites communicate via IP. Failover across the sites is coordinated by a higher-level entity, the booth. Geo clusters need to cope with limited network bandwidth and high latency. Storage is replicated asynchronously.

load balancing

The ability to make several servers participate in the same service and do the same work.

local cluster

A single cluster in one location (for example, all nodes are located in one data center). Network latency can be neglected. Storage is typically accessed synchronously by all nodes.

location

In the context of a *location constraint,* "location" refers to the nodes on which a resource can or cannot run.

LRM (local resource manager)

The local resource manager is located between the Pacemaker layer and the resources layer on each node. It is implemented as `pacemaker-execd` daemon. Through this daemon, Pacemaker can start, stop, and monitor resources.

mcastaddr (multicast address)

IP address to be used for multicasting by the Corosync executive. The IP address can either be IPv4 or IPv6.

mcastport (multicast port)

The port to use for cluster communication.

metro cluster

A single cluster that can stretch over multiple buildings or data centers, with all sites connected by fibre channel. Network latency is usually low (<5 ms for distances of approximately 20 miles). Storage is frequently replicated (mirroring or synchronous replication).

multicast

A technology used for a one-to-many communication within a network that can be used for cluster communication. Corosync supports both multicast and unicast.

node

Any computer (real or virtual) that is a member of a cluster and invisible to the user.

pacemaker-controld (cluster controller daemon)

The CRM is implemented as daemon, pacemaker-controld. It has an instance on each cluster node. All cluster decision-making is centralized by electing one of the pacemaker-controld instances to act as a primary. If the elected pacemaker-controld process fails (or the node it ran on), a new one is established.

PE (policy engine)

The policy engine is implemented as `pacemaker-schedulerd` daemon. When a cluster transition is needed, based on the current state and configuration, `pacemaker-schedulerd` calculates the expected next state of the cluster. It determines what actions need to be scheduled to achieve the next state.

quorum

In a cluster, a cluster partition is defined to have quorum (be "quorate") if it has the majority of nodes (or votes). Quorum distinguishes exactly one partition. It is part of the algorithm

to prevent several disconnected partitions or nodes from proceeding and causing data and service corruption (split brain). Quorum is a prerequisite for fencing, which then ensures that quorum is indeed unique.

RA (resource agent)

A script acting as a proxy to manage a resource (for example, to start, stop, or monitor a resource). SUSE Linux Enterprise High Availability supports different kinds of resource agents. For details, see *Section 6.2, "Supported resource agent classes"*.

ReaR (Relax and Recover)

An administrator tool set for creating disaster recovery images.

resource

Any type of service or application that is known to Pacemaker. Examples include an IP address, a file system, or a database.

The term "resource" is also used for DRBD, where it names a set of block devices that are using a common connection for replication.

RRP (redundant ring protocol)

Allows the use of multiple redundant local area networks for resilience against partial or total network faults. This way, cluster communication can still be kept up as long as a single network is operational. Corosync supports the Totem Redundant Ring Protocol.

SBD (STONITH Block Device)

Provides a node fencing mechanism through the exchange of messages via shared block storage (SAN, iSCSI, FCoE, etc.). Can also be used in diskless mode. Needs a hardware or software watchdog on each node to ensure that misbehaving nodes are really stopped.

SFEX (shared disk file exclusiveness)

SFEX provides storage protection over SAN.

split-brain

A scenario in which the cluster nodes are divided into two or more groups that do not know of each other (either through a software or hardware failure). STONITH prevents a split-brain situation from badly affecting the entire cluster. Also known as a "partitioned cluster" scenario.

The term `split brain` is also used in DRBD but means that the two nodes contain different data.

SPOF (single point of failure)

Any component of a cluster that, should it fail, triggers the failure of the entire cluster.

STONITH

The acronym for "Shoot the other node in the head". It refers to the fencing mechanism that shuts down a misbehaving node to prevent it from causing trouble in a cluster. In a Pacemaker cluster, the implementation of node level fencing is STONITH. For this, Pacemaker comes with a fencing subsystem, `pacemaker-fenced`.

switchover

Planned, on-demand moving of services to other nodes in a cluster. See *failover*.

ticket

A component used in Geo clusters. A ticket grants the right to run certain resources on a specific cluster site. A ticket can only be owned by one site at a time. Resources can be bound to a certain ticket by dependencies. Only if the defined ticket is available at a site, the respective resources are started. Vice versa, if the ticket is removed, the resources depending on that ticket are automatically stopped.

unicast

A technology for sending messages to a single network destination. Corosync supports both multicast and unicast. In Corosync, unicast is implemented as UDP-unicast (UDPU).

E GNU licenses

This appendix contains the GNU Free Documentation License version 1.2.

GNU Free Documentation License

Copyright (C) 2000, 2001, 2002 Free Software Foundation, Inc. 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA. Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or non-commercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or non-commercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See https://www.gnu.org/copyleft/ ↗.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

```
Copyright (c) YEAR YOUR NAME.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.2
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
A copy of the license is included in the section entitled "GNU
Free Documentation License".
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

```
with the Invariant Sections being LIST THEIR TITLES, with the
Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.