Adding Constraints to High Availability Cluster Resources

Resource constraints let you specify cluster node assignments, define the start order, and manage dependencies between resources. Because cluster resources sometimes depend on other resources or on a particular node configuration, they might need to run in a specific location or start in a specific order. Add resource constraints to specify exactly where and when High Availability cluster resources can run.

Prerequisites

- An existing SUSE Linux Enterprise High Availability cluster
- Existing cluster resources

If you still need to configure cluster resources, see *Configuring High Availability Cluster Resources* (https://documentation.suse.com/sle-ha/16.1/html/HA-resources-configuring/).

Publication Date: 06 Nov 2026

Contents

1 What are cluster resources? 3

2 What are resource constraints? 6

3 Adding order constraints 9

4 Adding location constraints 11

5 Adding colocation constraints 15

6 Colocating resources without adding dependency 18

7 Applying multiple changes at once 18

8 Legal Notice 25

A GNU Free Documentation License 26

HA glossary 34

1 What are cluster resources?

In a High Availability cluster, the applications and services that need to be highly available are called *resources*. Cluster resources can include Web sites, e-mail servers, databases, file systems, virtual machines or any other applications or services that need to be available to users at all times. You can start, stop, monitor and move resources as needed. You can also specify whether resources should run together on the same node, or start and stop in sequential order. If a node fails, the resources running on it *fail over* (move) to another node instead of being lost.

In SUSE Linux Enterprise High Availability, the *cluster resource manager* (CRM) is *Pacemaker*, which manages and coordinates all cluster services. Pacemaker uses *resource agents* (RAs) to start, stop and monitor resources. A resource agent abstracts the resource and presents its status to the cluster. This means that Pacemaker only interacts with the resource agent, not with the actual application or service. SUSE Linux Enterprise High Availability supports many different resource agents that are designed to manage specific types of resources.

1.1 Primitive resources

Primitive resources represent the applications or services that need to be highly available. Configuring a primitive specifies the resource agent and any other information required for the cluster to manage a resource. To see the supported resource agent classes, run **`crm ra classes`**. To see the resource agents available within a class, run **`crm ra list`** *`CLASS`*.

Primitive resources can have the following settings:

Properties

Resource properties are the basic details required by the cluster: a unique name for the resource, and the resource agent that manages the application or service.

Parameters

Parameters are specific to each resource agent and determine the details and behavior of an application or service. To show all the parameters available for a resource agent, run **`crm ra info`** *`RESOURCE_AGENT`*.

Operations

Operations are actions that the cluster can perform on a resource, such as `start`, `stop` and `monitor`. To show the supported operations for a resource agent, run **`crm ra info`** *`RESOURCE_AGENT`* and scroll to the bottom of the output.

Meta attributes

Meta attributes change how the cluster treats a resource. To show all the meta attributes available for primitives, run **`crm_resource --list-options primitive`**.

Utilization attributes

If the cluster nodes have utilization attributes configured, you can add these attributes to cluster resources to help with load balancing. For more information, see https://clusterlabs.org/projects/pacemaker/doc/3.0/Pacemaker_Explained/html/utilization.html.

1.2 Resource groups

Cluster resources might depend on other resources, such as a Web server that requires an IP address and a file system. You can combine these resources into a *resource group*. Groups contain multiple primitives that need to be located together, started sequentially and stopped in the reverse order.

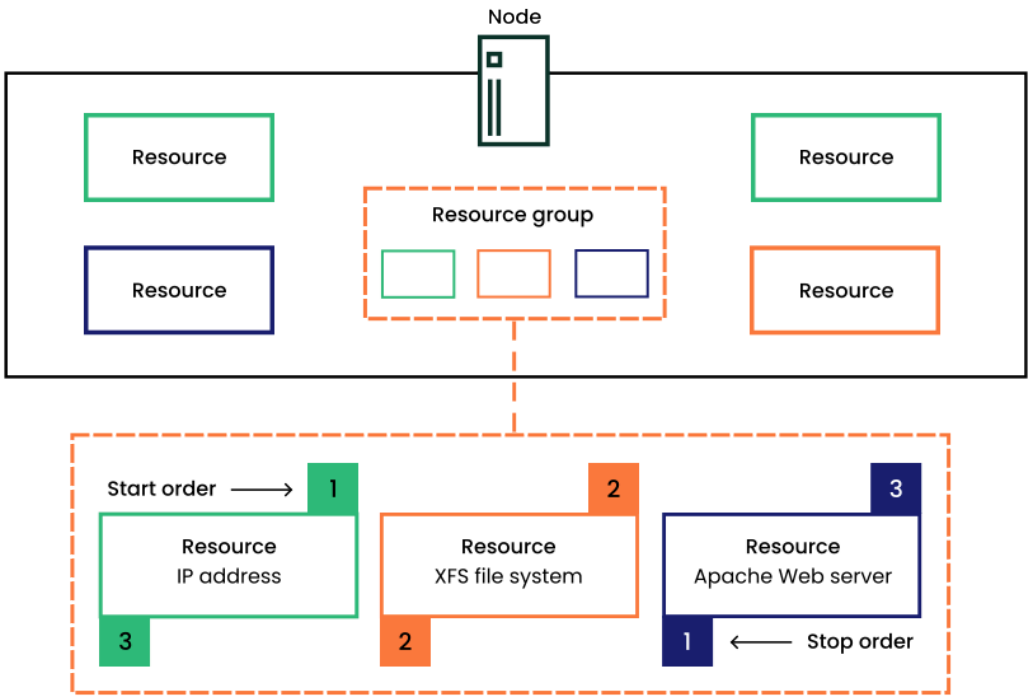


FIGURE 1: RESOURCE GROUP

Groups have the following properties:

Contents

Groups can only contain primitive resources, not clones or other groups.

Starting and stopping

Resources start in the order they appear and stop in the reverse order.

Location

Resources in the group must all run on the same node.

Dependency

If one of the resources can't run anywhere, then the resources that appear after it in the group can't run anywhere either.

Meta attributes

The meta attributes `target-role`, `priority`, `maintenance` and `is-managed` can be added directly to the group. All other attributes are inherited from the group's resources.

Resource stickiness

Stickiness in groups is additive. Every *active* resource in the group contributes its stickiness value to the group's total.

1.3 Resource clones

Resource clones can run simultaneously on multiple nodes. This is required for cluster file systems like GFS2, for example. You can clone a primitive or a group if the resource agent supports it. Clones can also have additional clone-specific meta attributes to change their behavior.

Clones can be one of the following types:

Anonymous clones

Anonymous clones are the simplest type. All instances of the clone behave identically. Therefore, only one instance of an anonymous clone can be active on each node.

Globally unique clones

Globally unique clones are distinct entities. An instance of the clone running on one node is not necessarily equivalent to another instance on another node.

Promotable clones

Promotable clones allow the instances to be in one of two operating modes: promoted or unpromoted (also known as primary and secondary, or active and passive). Promotable clones can be either anonymous or globally unique.

To make a clone promotable, the resource agent must support the *promote* and *demote* operations. The cluster determines which instance of the clone to promote using one of the following methods (or a combination of both):

- If a resource agent supports promotable clones, it might automatically set promotion scores based on its own preference for which instance to promote.
- You can manually set promotion rules or preferences by creating location or colocation constraints.

For more information, see https://clusterlabs.org/projects/pacemaker/doc/3.0/Pacemaker_Explained/html/collective.html#determining-which-instance-is-promoted

1.4 For more information

- Primitive resources: https://clusterlabs.org/projects/pacemaker/doc/3.0/Pacemaker_Explained/html/resources.html
- Groups and clones: https://clusterlabs.org/projects/pacemaker/doc/3.0/Pacemaker_Explained/html/collective.html
- Resource operations: https://clusterlabs.org/projects/pacemaker/doc/3.0/Pacemaker_Explained/html/operations.html
- Utilization attributes: https://clusterlabs.org/projects/pacemaker/doc/3.0/Pacemaker_Explained/html/utilization.html

2 What are resource constraints?

Resource constraints let you specify cluster node assignments, define the start order, and manage dependencies between resources.

2.1 Types of constraints

Order

Order constraints tell the cluster to start resources in a specific order and stop them in the reverse order. You can also start or stop a resource right before or after a different resource performs a certain action, such as being promoted.

Location

Location constraints tell the cluster where resources can or cannot run. You can specify that a resource *must* or must *not* run on a particular node, or give the resource an optional preference for or against a node.

Colocation

Colocation constraints tell the cluster which resources can or cannot run together on the same node. You can specify that the resources *must* or must *not* run together, or give a resource an optional preference for or against another resource.

Important: Restrictions for constraints and primitive resources

- Don't add colocation constraints to primitives in a resource group. Add a colocation constraint to the group instead.
- Don't add any constraints to a primitive that has been cloned. Add constraints to the clone instead.

2.2 Scores and kinds

Location and colocation constraints must have a *score* to indicate how important the constraint is. A score of `INFINITY` (positive or negative) means that the constraint *must* be applied. The score can also be a non-infinite number (positive or negative), which means the constraint is only a preference. In this case, the constraint might not be applied if its score is outweighed by other factors, such as other constraints, resource stickiness, failure thresholds, etc.

For location constraints, a positive score means the resource should (or must) run on the specified node. A negative score means the resource should *not* (or must not) run on the specified node. You can also add multiple location constraints with different scores to prioritize node preferences.

For colocation constraints, a positive score means the resources should (or must) run together on the same node. A negative score means the resources should *not* (or must not) run together on the same node.

Instead of a score, order constraints must have a *kind*, which defines whether the constraint is mandatory or optional.

2.3 Resource sets

Resource sets let you group multiple resources in a single constraint instead of needing multiple constraints. The effect depends on the type of constraint:

- In a *location* constraint, a resource set applies the constraint to multiple resources instead of just one, without creating any dependency between the resources.
  Location constraints use curly brackets to group the resources.

- In an *order* constraint, a resource set gives multiple resources the same position in the order. For example, a simple order constraint says that `res1` must start first, then `res2`, then `res3`. However, if you put `res2` and `res3` in a resource set, the constraint would say that `res1` must start first, and then `res2` and `res3` can start in any order.
  Order constraints can use round brackets or square brackets to group the resources.

- In a *colocation* constraint, a resource set gives multiple resources the same relationship to the next resource. For example, a simple colocation constraint says that `res1` must run on the same node as `res2`, which must run on the same node as `res3`. However, if you put `res1` and `res2` in a resource set, the constraint would say that `res1` and `res2` must run on the same node as `res3`, but aren't otherwise dependent on each other.
  Colocation constraints can use round brackets or square brackets to group the resources.

The syntax for resource sets can be complex, so before using them in a production environment, we recommend testing them to make sure the resources behave as expected.

For more information about the syntax of resource sets and the difference between round and square brackets, run **`crm help Resourcesets`**.

2.4 For more information

- Constraints: https://clusterlabs.org/projects/pacemaker/doc/3.0/Pacemaker_Explained/html/constraints.html
- Constraints for clones: https://clusterlabs.org/projects/pacemaker/doc/3.0/Pacemaker_Explained/html/collective.html#clone-constraints
- Scores: https://clusterlabs.org/projects/pacemaker/doc/3.0/Pacemaker_Explained/html/local-options.html#scores
- Resource sets: https://clusterlabs.org/projects/pacemaker/doc/3.0/Pacemaker_Explained/html/constraints.html#resource-sets

3 Adding order constraints

Order constraints tell the cluster to start resources in a specific order and stop them in the reverse order. You can also start or stop a resource right before or after a different resource performs a certain action, such as being promoted.

Instead of a score, order constraints have a *kind*. The kind can be one of the following:

- `Mandatory`: Resources *must* start in this order.
- `Optional`: The start order is only a preference.
- `Serialize`: A resource must finish starting before the next resource can begin starting. This is useful for resources that put a high load on the node during start-up, for example.

You can add order constraints with the CRM Shell (`crmsh`) or with the Hawk Web interface.

CRM Shell

You can perform this procedure on any node in the cluster.

1. Log in either as the `root` user or as a user with **`sudo`** privileges.
2. Add an order constraint. The basic command requires a name, a kind, and at least two resources:

   ```
   > sudo crm configure order ORDER_NAME KIND: RESOURCE1 RESOURCE2
   ```

   The *`KIND`* can be `Mandatory`, `Optional` or `Serialize`.

ADDITIONAL OPTIONS

Resource sets

You can use round brackets or square brackets to group resources in an order constraint. In this example, `res1` must start first, and then `res2` and `res3` can start in any order:

```
> sudo crm configure order res1-first Mandatory: res1 [ res2 res3 ]
```

Round brackets are treated as special characters in most shells, so for **`crm`** commands that aren't in interactive mode, you must escape the brackets with `\`:

```
> sudo crm configure order res1-first Mandatory: res1 \( res2 res3 \)
```

For more information about the difference between round and square brackets, run **`crm help Resourcesets`**.

Symmetry

In most cases, you should keep the default `symmetrical=true`, which means that resources stop in the reverse of their start order. However, if necessary, you can change it to `false`.

Actions

You can require a resource to perform an action (`start`, `stop`, `promote` or `demote`) before or after another resource starts. In this example, `res1` must start before `clone2` can be promoted:

```
> sudo crm configure order res1-pr-clone2 Mandatory: res1 clone2:promote
```

3. You can view all order constraints with the following command:

```
> sudo crm configure show type:order
```

Hawk

You can use Hawk on any machine with a Web browser and network access to the nodes.

1. In a Web browser, go to `https://HAWKSERVER:7630/` and log in.
2. From the left navigation bar, select *Configuration* › *Add Constraint*.
3. Click *Order*.
4. Enter a unique *Constraint ID*.

5. Select a *Kind* from the drop-down list.
6. In most cases, you should keep the option *Symmetrical* enabled. This means that resources stop in the reverse of their start order.
7. From the drop-down list in the *Resources* section, select a resource.
   A new drop-down list appears. Repeat this step to add more resources.
   The resources will start in the order shown on the screen. To change the order, click the up arrow icon to swap a resource with the one above it.
   To remove a resource from the constraint, click the minus icon.
8. *(Optional)* To create a resource set, click the chain icon to link a resource to the one above it. You can link multiple resources into a set or create multiple resource sets.
   To remove a resource from the set, click the scissors icon.
9. *(Optional)* If required, select an action (`Start`, `Stop`, `Promote` or `Demote`) from the drop-down list next to a resource. The resource must perform this action before or after another resource starts, depending on the order.
10. Click *Create* to finish the configuration.
    A message at the top of the screen shows if the action was successful.

FOR MORE INFORMATION

- **`crm configure help order`**
- **`crm help Resourcesets`**
- https://clusterlabs.org/projects/pacemaker/doc/3.0/Pacemaker_Explained/html/constraints.html#specifying-the-order-in-which-resources-should-start-stop

4 Adding location constraints

Location constraints tell the cluster where resources can or cannot run. You can specify that a resource *must* or must *not* run on a particular node, or give the resource an optional preference for or against a node.

Location constraints use a *score* to indicate how important the constraint is. A score of `INFINITY` (positive or negative) means that the constraint *must* be applied. The score can also be a non-infinite number (positive or negative), which means the constraint is only a preference. You can also add multiple location constraints with different scores to prioritize node preferences.

You can add location constraints with the CRM Shell (`crmsh`) or with the Hawk Web interface.

CRM Shell

You can perform this procedure on any node in the cluster.

1. Log in either as the `root` user or as a user with **`sudo`** privileges.
2. Add a location constraint. The basic command requires a name, at least one resource, a score, and a node:

```
> sudo crm configure location LOCATION_NAME RESOURCE SCORE: NODE
```

The *`SCORE`* can be `inf`, `-inf` or a number (positive or negative).

ADDITIONAL OPTIONS

Resource sets

You can use curly brackets to include multiple resources in a location constraint. In this example, `res1` and `res2` must both run on node `alice`:

```
> sudo crm configure location res1-2-alice { res1 res2 } inf: alice
```

Resource patterns

You can use a resource pattern (a regular expression between two slashes) to refer to multiple resources. If you add new resources with the same name pattern, they will also be included in the location constraint. In this example, any resources beginning with `res` must run on node `alice`:

```
> sudo crm configure location all-res-alice /res.*/ inf: alice
```

Resource discovery

In most cases, you should keep the default `resource-discovery=always`. This means that the cluster checks whether the resource is already running, to help avoid unwanted instances of the resource. However, if necessary, you can change it to `never` or `exclusive`.

Rules

Instead of specifying a single node, you can use a rule to refer to multiple nodes. If you add more nodes that meet the rule, they will also be included in the location constraint. In this example, `res1` can't run on any node where the custom node attribute `memory` is less than `4096`:

```
> sudo crm configure location res1-no-small-nodes res1 \
  rule -inf: memory lt 4096
```

3. Check the status to see which node the resources are now running on:

```
> sudo crm status
```

Hawk

You can use Hawk on any machine with a Web browser and network access to the nodes.

1. In a Web browser, go to `https://HAWKSERVER:7630/` and log in.
2. From the left navigation bar, select *Configuration* › *Add Constraint*.
3. Click *Location*.
4. Enter a unique *Constraint ID*.
5. Select one or more *Resources* to apply the constraint to.
   To remove a resource from the constraint, press `Ctrl` and click the resource again.
6. Enter a numeric *Score* or select a value from the drop-down list:
   - `Always` sets the score to `INFINITY`, which means that the resources must run on the specified node.
   - `Never` sets the score to `-INFINITY`, which means that the resources must *not* run on the specified node.
   - `Advisory` sets the score to `0`, which disables the constraint. This can be useful when you want to set resource discovery but don't want to constrain the resources.
7. Select a *Node* to apply the constraint to.
8. *(Optional)* Click the *Advanced* tab if you need to set further parameters.

9. *(Optional)* In most cases, you should keep the option *Resource Discovery* enabled. This means that the cluster checks whether the resource is already running, to help avoid unwanted instances of the resource. If you do need to change this option, select a new value from the drop-down list.

10. *(Optional)* Instead of specifying a single node, you can use a *Rule* to refer to multiple nodes. If you add more nodes that meet the rule, they will also be included in the location constraint. Configure the rule with the following settings:

    - Add a *Score* for the rule.
    - Optionally, select a *Role* from the drop-down list. The rule will only apply when the resource is in this role.
    - If you intend to add multiple expressions, select a *Boolean Operator* from the drop-down list.
    - Add one or more *Expressions*. Select an expression type from the drop-down list, then click the plus icon.
      The required fields for that expression type appear. This could be a node's `uname`, or a custom node attribute and value. Fill in the fields and select an operation (such as =, <, >, etc.) to create the expression.

11. Click *Create* to finish the configuration.
    A message at the top of the screen shows if the action was successful.

FOR MORE INFORMATION

- **`crm configure help location`**
- **`crm help Resourcesets`**
- **`crm help RuleExpressions`**
- https://clusterlabs.org/projects/pacemaker/doc/3.0/Pacemaker_Explained/html/constraints.html#deciding-which-nodes-a-resource-can-run-on
- https://clusterlabs.org/projects/pacemaker/doc/3.0/Pacemaker_Explained/html/rules.html

5 Adding colocation constraints

Colocation constraints tell the cluster which resources can or cannot run together on the same node. You can specify that the resources *must* or must *not* run together, or give a resource an optional preference for or against another resource.

Colocation constraints use a *score* to indicate how important the constraint is. A score of `INFINITY` (positive or negative) means that the constraint *must* be applied. The score can also be a non-infinite number (positive or negative), which means the constraint is only a preference.



Important: Resource start order

Colocation constraints *don't* affect the order in which the resources start. If you want resources to run on the same node *and* start in a specific order, create a resource group.

You can add colocation constraints with the CRM Shell (`crmsh`) or with the Hawk Web interface.

CRM Shell

You can perform this procedure on any node in the cluster.

1. Log in either as the `root` user or as a user with **`sudo`** privileges.
2. Add a colocation constraint. The basic command requires a name, a score and at least two resources:

```
> sudo crm configure colocation COLOCATION_NAME SCORE: RESOURCE1 RESOURCE2
```

The *`SCORE`* can be `inf`, `-inf` or a number (positive or negative).
`RESOURCE2` is assigned to a node *first*, and *`RESOURCE1`* is placed relative to *`RESOURCE2`*.

ADDITIONAL OPTIONS

Resource sets

You can use round brackets or square brackets to group resources in a colocation constraint. In this example, `res1` and `res2` must run on the same node as `res3`, but aren't otherwise dependent on each other:

```
> sudo crm configure colocation res1-2-with-res3 \
  inf: [ res1 res2 ] res3
```

Round brackets are treated as special characters in most shells, so for `crm` commands that aren't in interactive mode, you must escape the brackets with `\`:

```
> sudo crm configure colocation res1-2-with-res3 \
  inf: \( res1 res2 \) res3
```

For more information about the difference between round and square brackets, run `crm help Resourcesets`.

Roles

You can require a role (`Started`, `Stopped`, `Promoted` or `Unpromoted`) for one or more resources. The colocation constraint only applies when the resource is in this role. In this example, `res1` must run on the same node where `clone2` is in the `Promoted` role:

```
> sudo crm configure colocation res1-with-pr-clone2 \
  inf: res1 clone2:Promoted
```

Node attributes

You can use a node attribute to colocate resources on a group of nodes instead of one node. In this example, `res1` and `res2` don't have to run on the same node, but must run on nodes with the same value for the `datacenter` node attribute:

```
> sudo crm configure colocation res1-2-same-dc \
  inf: res1 res2 node-attribute=datacenter
```

3. Check the status to see which node the resources are now running on:

```
> sudo crm status
```

Hawk

You can use Hawk on any machine with a Web browser and network access to the nodes.

1. In a Web browser, go to `https://HAWKSERVER:7630/` and log in.
2. From the left navigation bar, select *Configuration* › *Add Constraint*.
3. Click *Colocation*.
4. Enter a unique *Constraint ID*.

5. Enter a numeric *Score* or select a value from the drop-down list:

   - `Always` sets the score to `INFINITY`, which means that the resources must run on the same node.
   - `Never` sets the score to `-INFINITY`, which means that the resources must run on different nodes.

6. From the drop-down list in the *Resources* section, select a resource.
   A new drop-down list appears. Repeat this step to add more resources.
   The resource that appears *last* in the list is assigned to a node *first*, and the other resources are placed relative to that resource. To change the order of the resources, click the up arrow icon to swap a resource with the one above it.
   To remove a resource from the constraint, click the minus icon.

7. *(Optional)* To create a resource set, click the chain icon to link a resource to the one above it. You can link multiple resources into a set or create multiple resource sets.
   To remove a resource from the set, click the scissors icon.

8. *(Optional)* If required, select a role (`Started`, `Stopped`, `Promoted` or `Promotable`) from the drop-down list next to a resource. The colocation constraint only applies when the resource is in this role.

9. *(Optional)* You can add a *Node Attribute* to colocate the resources on a group of nodes instead of one specific node. If the nodes have an attribute assigned, the resources in this constraint will all be placed on nodes with the same value for that attribute.

10. Click *Create* to finish the configuration.
    A message at the top of the screen shows if the action was successful.

FOR MORE INFORMATION

- **`crm configure help colocation`**
- **`crm help Resourcesets`**
- https://clusterlabs.org/projects/pacemaker/doc/3.0/Pacemaker_Explained/html/constraints.html#placing-resources-relative-to-other-resources
- https://clusterlabs.org/projects/pacemaker/doc/3.0/Pacemaker_Explained/html/constraints.html#colocating-sets-of-resources

6 Colocating resources without adding dependency

A colocation constraint creates dependency between the resources. If one resource fails, the cluster also restarts the other resource. However, sometimes you might need to place resources together *without* any dependency. You can do this with the CRM Shell's **`weak-bond`** command. This creates a `Dummy` resource, then creates a colocation constraint between the `Dummy` resource and the resources you specify.

You can perform this procedure on any node in the cluster.

1. Log in either as the `root` user or as a user with **`sudo`** privileges.
2. Run the following command, specifying two or more primitive resources:

```
> sudo crm configure assist weak-bond RESOURCE1 RESOURCE2
Create weak bond / independent colocation
The following elements will be created:
   * Colocation constraint, ID: place-constraint-1
   * Dummy resource, ID: place-dummy-1
Create resources (y/n)?
```

3. Enter `y` to create the `Dummy` resource and colocation constraint.
4. View the new constraint:

```
> sudo crm configure show place-constraint-1
```

5. Check which node the resources are running on:

```
> sudo crm status
```

FOR MORE INFORMATION

- **`crm assist weak-bond --help`**

7 Applying multiple changes at once

SUSE Linux Enterprise High Availability provides different ways to apply multiple cluster changes simultaneously. This is sometimes necessary when you make changes that are dependent on each other. For example, a group of related resources might need to have colocation or order constraints in place before the resources start.

You can use one of the following methods to make batch changes:

- *Section 7.1, "Using the CRM Shell in interactive mode"*
- *Section 7.2, "Using Hawk in batch mode"*
- *Section 7.3, "Loading configuration changes from a file"*

7.1 Using the CRM Shell in interactive mode

The CRM Shell's interactive mode is a dedicated, internal command shell that allows you to run multiple commands without needing to repeatedly type the **`crm`** prefix. You can also move down a level to a subcommand, such as **`crm configure`**, to run multiple commands at the same level. Changes you make in interactive mode are only applied to the live cluster after you **`commit`** them, which can be useful when configuring resources that are dependent on one another and require constraints. Interactive mode also supports tab completion.

1. Log in either as the `root` user or as a user with **`sudo`** privileges.
2. Start interactive mode:

   ```
   > sudo crm
   ```

   The prompt changes to `crm(live/HOSTNAME)#`. However, for better readability we omit the host name in our documentation:

   ```
   crm(live)#
   ```

   ### Tip: Starting a subcommand directly

   You can also start interactive mode directly at the subcommand level. For example:

   ```
   > sudo crm configure
   crm(live)configure#
   ```

3. You can navigate interactive mode with the following options:

   - List the available subcommands:

     ```
     crm(live)# ls
     ```

   - Enter the name of a subcommand to switch to that level. For example:

     ```
     crm(live)# resource
     crm(live)resource#
     ```

   - Switch between subcommands with **`cd`**. For example:

     ```
     crm(live)resource# cd ../node
     crm(live)node#
     ```

   - Go back one level with **`up`** or **`cd`**:

     ```
     crm(live)node# up
     crm(live)#
     ```

4. Make your changes. This example shows commands for configuring a virtual IP address and adding a location constraint to keep the IP address running on a node named `alice`. Because both commands are under the **`crm configure`** subcommand, you can switch to the **`configure`** level to shorten the actual commands you need to type.

   EXAMPLE 1: ADDING A VIRTUAL IP ADDRESS WITH A LOCATION CONSTRAINT FOR NODE ALICE

   ```
   crm(live)# configure
   crm(live)configure# primitive vip1 IPaddr2 params ip=192.168.1.5
   crm(live)configure# location loc-vip1-on-alice vip1 inf: alice
   crm(live)configure# show changed
   ```

5. Commit your changes to apply them to the live cluster:

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

   You must commit your changes before you exit interactive mode, switch back to the **`crm`** level, or switch to another subcommand. The changes will be lost if you don't commit them, but the CRM Shell will warn you before this happens.

6. Exit interactive mode:

   ```
   crm(live)configure# quit
   ```

FOR MORE INFORMATION

- https://crmsh.github.io/man-5.0/#topics.Introduction.Interface

7.2 Using Hawk in batch mode

Hawk's *batch mode* allows you to stage configuration changes and apply them all as a single transaction, instead of having each change take effect immediately. This can be useful when configuring resources that are dependent on one another and require constraints.

Batch mode also includes a *cluster simulator*, which shows the expected outcome of each change without affecting the live cluster. For example, if you stop a resource while in batch mode, the user interface shows the resource as stopped, but in the live cluster the resource is still running.

You can also *inject* cluster events, such as node changes and resource operations, to explore potential failure scenarios. These changes are only simulations, and won't be applied to the live cluster along with the configuration changes.

Important: Using configuration wizards in batch mode

Certain wizards include actions beyond cluster configuration, such as installing packages. 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 can't be used while in batch mode.

1. In a Web browser, go to `https://HAWKSERVER:7630/` and log in.
2. In the top-level row, click *Batch*.
   An additional bar appears below the top-level row to indicate that batch mode is active. This bar also includes the options to show, discard or apply your changes.



FIGURE 2: HAWK BATCH MODE BAR

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 but are not applied to the live cluster yet.

4. To view details of your changes, select *Show* from the batch mode bar.
   The *Batch Mode* window opens and shows the difference between the live cluster and the simulated changes in `crmsh` syntax. Lines starting with a `-` character represent the current state, whereas lines starting with `+` show the proposed state.

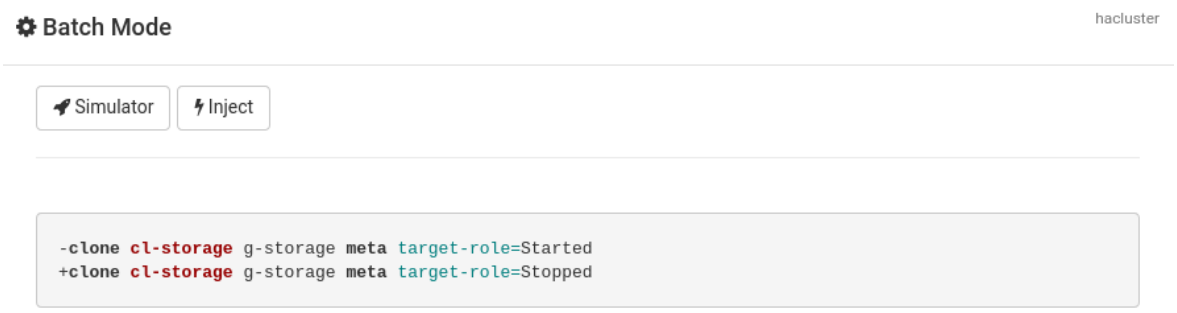


FIGURE 3: HAWK BATCH MODE WINDOW

5. *(Optional)* To test potential failures or to see how the cluster behaves in a specific scenario, you can simulate node changes and resource operations:

   - Simulate a node failure or status change:

     a. Click *Inject* › *Node Event*.

     b. Select a *Node* and the node *State* to simulate.

     c. Click *OK*.

   - Simulate a resource operation:

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

     b. Select a *Resource* and the *Operation* to simulate.

     c. If necessary, define an *Interval*.

     d. Select the *Node* on which to run the operation.

     e. Select the operation *Result* to simulate.

     f. Click *OK*.

The *Batch Mode* window shows a new line per injected event. Any event listed here is simulated immediately and is reflected on the *Status* screen.
To remove an injected event, click the *x* icon next to it. Hawk updates the *Status* screen accordingly.

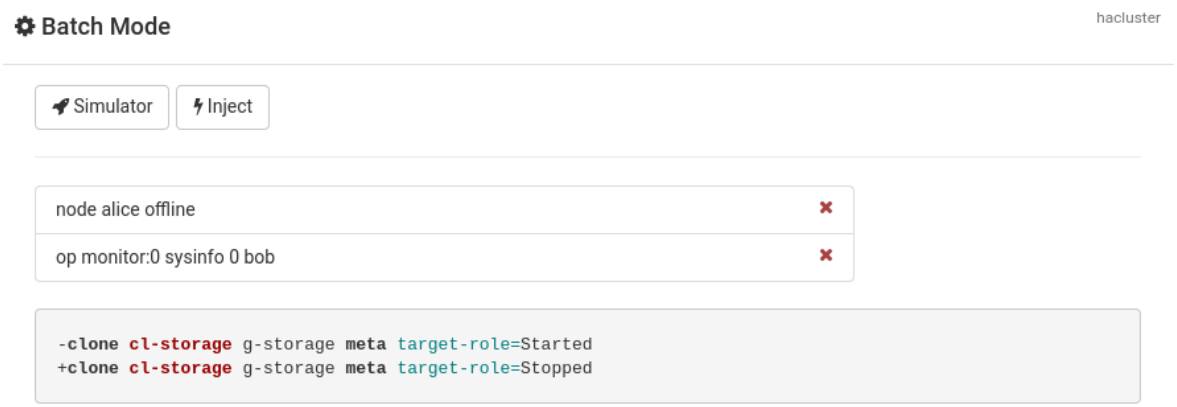


FIGURE 4: HAWK BATCH MODE WINDOW WITH INJECTED EVENTS

6. *(Optional)* To view more details about the changes, click *Simulator* and choose one of the following:

   - *Summary*: Shows a detailed summary.
   - *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.

7. After reviewing your changes, click *Close* to exit the *Batch Mode* window.

8. Choose whether to keep your changes or discard them:

   - To keep your changes, click *Apply* in the batch mode bar. The *Apply Configuration* window opens and shows a list of the changes. Only configuration changes are applied; injected events are not. Click *Apply* to confirm.
   - To discard your changes, click *Discard* in the batch mode bar, then click *OK* to confirm.

Both options deactivate batch mode and take you back to normal mode.

7.3 Loading configuration changes from a file

The **`crm configure load`** command allows you to load configuration changes from a local file or a network URL. This command is useful if you prefer to prepare your configuration changes in advance. The CRM Shell validates the changes and warns you if the new configuration is invalid.

You can load the new configuration with one of the following methods:

- `update`: Imports the new configuration into the current configuration by adding new items or updating existing items. This method can't remove any existing items.
- `replace`: Replaces the current configuration with the new configuration. If any resources are running, this method still adds new items and updates existing items, but doesn't remove any existing items.
- `push`: Imports the new configuration into the current configuration by adding new items or updating existing items, and also removes existing items that are not present in the new configuration. This method can remove existing items even when resources are running, but only if the removed item isn't a running resource. If you try to remove a running resource, the command exits with an error.

You can perform this procedure on any node in the cluster:

1. Log in either as the `root` user or as a user with **`sudo`** privileges.
2. Show the cluster's current configuration:
   - Show the current configuration in CLI format:

     ```
     > sudo crm configure show
     ```

   - Show the current configuration in XML format:

     ```
     > sudo crm configure show xml
     ```

3. Copy the current configuration into a new file, for example, `hacluster-config.xml`. You can copy the full configuration or just the portion you want to change.
4. Make your changes to the new configuration.

5. Load the new configuration from the file to the live cluster:

   - Load the new configuration in CLI format:

     ```
     > sudo crm configure load METHOD FILE
     ```

   - Load the new configuration in XML format:

     ```
     > sudo crm configure load xml METHOD FILE
     ```

   METHOD can be one of `replace`, `update` or `push`. *FILE* can be a path to a local file or a URL to a remote file.
   The CRM Shell checks the new configuration. If your changes would break the cluster configuration, you'll see the following output:

   ```
   error: CIB did not pass schema validation
   Configuration invalid
   Do you still want to commit (y/n)?
   ```

   If this happens, enter `n` to discard your changes.

6. Check the live cluster configuration for any errors:

   ```
   > sudo crm configure show
   ```

7. Check the status of the cluster to make sure everything is still running as expected:

   ```
   > sudo crm status
   ```

FOR MORE INFORMATION

- **`crm configure help load`**

8 Legal Notice

Copyright© 2006– 2026 SUSE LLC and contributors. All rights reserved.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or (at your option) version 1.3; with the Invariant Section being this copyright notice and license. A copy of the license version 1.2 is included in the section entitled “GNU Free Documentation License”.

For SUSE trademarks, see https://www.suse.com/company/legal/. All other third-party trademarks are the property of their respective owners. Trademark symbols (®, ™ etc.) denote trademarks of SUSE and its affiliates. Asterisks (*) denote third-party trademarks.

All information found in this book has been compiled with utmost attention to detail. However, this does not guarantee complete accuracy. Neither SUSE LLC, its affiliates, the authors, nor the translators shall be held liable for possible errors or the consequences thereof.

A 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.

HA glossary

ACL (access control list)

Access control lists can be used to limit or fine-tune a user's access to the CIB. An ACL consists of an ordered set of access rules. Each rule allows or denies read or write access to a part of the CIB. Rules can be combined into specific roles, and users can be assigned to a role that matches their tasks.

active/active, active/passive

How resources run on the nodes. Active/passive means that resources only run on the active node, but can move to the passive node if the active node fails. Active/active means that all nodes are active at once, and resources can run on (and move to) any node in the cluster.

arbitrator

An *arbitrator* is a machine running outside the cluster to provide an additional instance for cluster calculations. For example, *QNetd* provides a vote to help *QDevice* participate in *quorum* decisions.

CIB (cluster information base)

An XML representation of the whole cluster configuration and status (cluster options, nodes, resources, constraints, etc.). The CIB manager (`pacemaker-based`) keeps the CIB synchronized across the cluster nodes and handles requests to modify it.

clone

In the context of a cluster *resource*, a clone is a resource that can be active on multiple nodes. Any resource can be cloned if its resource agent supports it.

cluster

A *high-availability* cluster is a group of servers (physical or virtual) designed primarily to secure the highest possible availability of data, applications and services. Not to be confused with a *high-performance* cluster, which shares the application load to achieve faster results.

Cluster LVM (Cluster logical volume manager)

The term *Cluster LVM* indicates that LVM is being used in a cluster environment. This requires configuration adjustments to protect the LVM metadata on shared storage.

cluster offline upgrade

If a new product version includes major changes that are backward incompatible, the cluster requires a cluster offline upgrade. You must stop the cluster services on all nodes and upgrade the whole cluster before bringing the nodes back online.

For minor version upgrades, see *cluster rolling upgrade*.

cluster partition

A cluster partition occurs when communication fails between one or more nodes and the rest of the cluster. The nodes are split into partitions but are still active. They can only communicate with nodes in the same partition and are unaware of the separated nodes. This is known as a *split brain* scenario.

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. Stop the cluster services on a node, upgrade it, then bring it back online to join the cluster. Continue one by one until all cluster nodes are upgraded.

For major version upgrades, see *cluster offline upgrade*.

cluster services

The services that run the cluster. Typically this refers to Pacemaker and Corosync, but might also include SBD and QDevice if those services are used.

cluster stack

The ensemble of software technologies and components that make up a cluster.

colocation constraint

A type of *resource constraint* that specifies which resources can or cannot run together on a node.

concurrency violation

A resource that should be running on only one node in the cluster is running on several nodes.

Corosync

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

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 the CRM. It interacts with several components: local executors on its own node and on the other nodes, non-local CRMs, administrative commands, the fencing functionality, and the membership layer.

`crmsh` (CRM Shell)

The command-line utility `crmsh` manages the cluster, nodes and resources.

Csync2

A synchronization tool for replicating configuration files across all nodes in the cluster.

DC (designated coordinator)

The `pacemaker-controld` daemon is the cluster controller, which coordinates all actions. This daemon has an instance on each cluster node, but only one instance is elected to act as the DC. The DC is elected when the cluster services start, or if the current DC fails or leaves the cluster. The DC decides whether a cluster-wide change must be performed, such as fencing a node or moving resources.

disaster

An unexpected interruption of critical infrastructure caused by nature, humans, hardware failure, or software bugs.

disaster recovery

The process by which a function is restored to the normal, steady state after a disaster.

Disaster Recovery Plan

A strategy to recover from a disaster with the minimum impact on IT infrastructure.

DLM (Distributed Lock Manager)

DLM coordinates accesses to shared resources in a cluster, for example, managing file locking in clustered file systems to increase performance and availability.

DRBD

DRBD® is a block device designed for building High Availability clusters. It replicates data on a primary device to secondary devices in a way that ensures all copies of the data remain identical.

existing cluster

The term *existing cluster* is used to refer to any cluster that consists of at least one node. An existing cluster has a basic *Corosync* configuration that defines the communication channels, but does not necessarily have resource configuration yet.

failover

Occurs when a resource or node fails on one machine and the affected resources move to another node.

failover domain

A named subset of cluster nodes that are eligible to run a resource if a node fails.

fencing

Prevents 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 resource. Node-level fencing prevents a failed node from accessing shared resources and

prevents resources from running on a node with an uncertain status. This is usually done by resetting or powering off the node. In a *Pacemaker* cluster, node fencing is managed by the fencing subsystem `pacemaker-fenced`.

GFS2

Global File System 2 (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.

group

Resource groups contain multiple resources that need to be located together, started sequentially and stopped in the reverse order.

Hawk (High Availability Web Konsole)

A user-friendly Web-based interface for monitoring and administering a High Availability cluster from Linux or non-Linux machines. Hawk can be accessed from any machine that can connect to the cluster nodes, using a graphical Web browser.

heuristics

QDevice supports using a set of commands (*heuristics*) that run locally on start-up of cluster services, cluster membership change, successful connection to the *QNetd* server, or optionally at regular times. The result is used in calculations to determine which partition should have *quorum*.

knet (kronosnet)

A network abstraction layer supporting redundancy, security, fault tolerance, and fast fail-over of network links. In SUSE Linux Enterprise High Availability 16, *knet* is the default transport protocol for the *Corosync* communication channels.

local cluster

A single cluster in one location (for example, all nodes are located in one data center). Network latency is minimal. Storage is typically accessed synchronously by all nodes.

local executor

The local executor is located between *Pacemaker* and the resources on each node. Through the `pacemaker-execd` daemon, Pacemaker can start, stop and monitor resources.

location

In the context of a whole cluster, *location* can refer to the physical location of nodes (for example, all nodes might be located in the same data center). In the context of a *location constraint*, *location* refers to the nodes on which a resource can or cannot run.

location constraint

A type of *resource constraint* that defines the nodes on which a resource can or cannot run.

meta attribute

Parameters that tell the *CRM (cluster resource manager)* how to treat a specific *resource*. For example, you might define a resource's priority or target role.

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. Storage is frequently replicated using mirroring or synchronous replication.

network device bonding

Network device bonding combines two or more network interfaces into a single bonded device to increase bandwidth and/or provide redundancy. When using *Corosync*, the bonded device is not managed by the cluster software. Therefore, the bonded device must be configured on every cluster node that might need to access it.

node

Any server (physical or virtual) that is a member of a cluster.

order constraint

A type of *resource constraint* that defines the sequence of actions.

Pacemaker

Pacemaker is the *CRM (cluster resource manager)* in SUSE Linux Enterprise High Availability, or the "brain" that reacts to events occurring in the cluster. Events might be nodes that join or leave the cluster, failure of resources, or scheduled activities such as maintenance, for example. The `pacemakerd` daemon launches and monitors all other related daemons.

parameters (instance attributes)

Parameters determine which instance of a service the *resource* controls.

primitive

A primitive resource is the most basic type of cluster resource.

promotable clone

Promotable clones are a special type of *clone* resource that can be promoted. Active instances of these resources are divided into two states: promoted and unpromoted (also known as "active and passive" or "primary and secondary").

QDevice

QDevice and *QNetd* participate in *quorum* decisions. The `corosync-qdevice` daemon runs on each cluster node and communicates with QNetd to provide a configurable number of votes, allowing a cluster to sustain more node failures than the standard quorum rules allow.

QNetd

QNetd is an *arbitrator* that runs outside the cluster. The `corosync-qnetd` daemon provides a vote to the `corosync-qdevice` daemon on each node to help it participate in quorum decisions.

quorum

A *cluster partition* is defined to have quorum (be *quorate*) if it has the majority of nodes (or "votes"). Quorum distinguishes exactly one partition. This is part of the algorithm to prevent several disconnected partitions or nodes ("split brain") from proceeding and causing data and service corruption. Quorum is a prerequisite for fencing, which then ensures that quorum is 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.

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*, for example, 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 use a common connection for replication.

resource constraint

Resource constraints specify which cluster nodes a resource can run on, what order the resources should start in, and which other resources a specific resource is dependent on.

See also *colocation constraint*, *location constraint* and *order constraint*.

resource set

As an alternative format for defining location, colocation or order constraints, you can use *resource sets*, where primitives are grouped together in one set. When creating a constraint, you can specify multiple resources for the constraint to apply to.

resource template

To help create many resources with similar configurations, you can define a resource template. After being defined, it can be referenced in primitives or in certain types of 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.

SBD (STONITH Block Device)

SBD provides a node *fencing* mechanism through the exchange of messages via shared block storage. Alternatively, it can be used in diskless mode. In either case, it needs a hardware or software *watchdog* on each node to ensure that misbehaving nodes are really stopped.

scheduler

The scheduler is implemented as `pacemaker-schedulerd`. When a cluster transition is needed, `pacemaker-schedulerd` calculates the expected next state of the cluster and determines what actions need to be scheduled to achieve the next state.

split brain

A scenario in which the cluster nodes are divided into two or more groups that do not know about each other (either through a software or hardware failure). Also known as a *partitioned cluster* scenario.

The term *split brain* is also used in *DRBD* but means that the nodes contain different data.

SPOF (single point of failure)

Any component of a cluster that, if it fails, triggers the failure of the entire cluster.

stickiness

Resource stickiness is a *meta attribute* that determines how much a resource prefers to stay on its current node.

switchover

The planned moving of resources to other nodes in a cluster. See also *failover*.

utilization

Tells the CRM what capacity a certain *resource* requires from a node.

watchdog

SBD (STONITH Block Device) needs a watchdog on each node to ensure that misbehaving nodes are really stopped. SBD “feeds” the watchdog by regularly writing a service pulse to it. If SBD stops feeding the watchdog, the hardware enforces a system restart. This protects against failures of the SBD process itself, such as becoming stuck on an I/O error.