Red Hat Enterprise Linux 9

配置和管理高可用性集群

使用红帽高可用性附加组件创建和维护 Pacemaker 集群

Last Updated: 2024-06-27

Red Hat Enterprise Linux 9 配置和管理高可用性集群

使用红帽高可用性附加组件创建和维护 Pacemaker 集群

法律通告

摘要

红帽高可用性附加组件配置使用 Pacemaker 集群资源管理器的高可用性集群。此标题提供了熟悉 Pacemaker 集群配置步骤的流程，以及配置主动/主动和主动/被动集群的示例流程。

目录

对红帽文档提供反馈

我们感谢您对我们文档的反馈。让我们了解如何改进它。

通过 Jira 提交反馈（需要帐户）

1. 登录到 Jira 网站。
2. 点顶部导航栏中的 **Create**
3. 在 **Summary** 字段中输入描述性标题。
4. 在 **Description** 字段中输入您对改进的建议。包括文档相关部分的链接。
5. 点对话框底部的 **Create**。

第 1 章 高可用性附加组件概述

高可用性附加组件是一个集群的系统，它为关键生产环境服务提供可靠性、可伸缩性以及高可用性。

集群由两个或者多个计算机（称为 *节点（node）* 或 *成员（member）*）组成来一起执行任务。集群可用于提供高可用性服务或资源。多个机器的冗余是用来在出现各种失败时对环境的保护。

高可用性型机器通过消除单点失效以及在一个节点停止运作时将服务从一个群集节点切换到另外一个节点来提供高可用性服务。通常，高可用性群集中的服务会进行读写数据的操作（使用通过读写方式挂载的文件系统）。因此，高可用性集群必须维护数据的完整性，因为一个集群节点可能需要接管另外一个集群节点的任务。对于集群外的客户端，高可用性集群中出现的节点故障是不可见的。（高可用性机器有时候被称为故障转移（failover）集群。）High Availability Add-On 通过它的高可用性服务管理组件 **Pacemaker** 为集群提供高可用性。

红帽提供了各种规划、配置和维护红帽高可用性集群的文档。有关向 Red Hat 集群文档的各个方面提供指导索引的文章列表，请参阅 红帽高可用性附加组件文档指南。

1.1. HIGH AVAILABILITY ADD-ON 附件

红帽高可用性附加组件由多个组件组成，它们提供高可用性服务。

High Availability Add-On 的主要组件如下：

- 集群基础结构 - 为节点以集群方式运行提供基本功能：配置文件管理、成员资格管理、锁管理和保护。
- 高可用性服务管理 - 提供在一个节点不可操作时，服务从一个集群节点切换到另外一个节点的功能。
- 集群管理工具 - 用于设置、配置和管理高可用性附加组件的配置和管理工具。这些工具用于集群基础结构组件、高可用性和服务管理组件以及存储。

您还可以为高可用性附加组件添加以下组件：

- Red Hat GFS2 (Global File System 2) – Resilient Storage Add-On 的一部分，提供和高可用性附加组件一起使用的集群文件系统。GFS2 允许多个节点进行块级别的存储共享，就好像每个群集节点都连接至本地存储空间一样。GFS2 集群文件系统需要一个集群基础结构。
- LVM locking Daemon(**lvmlockd**)- Resilient Storage Add-On 的一部分，它提供集群存储的卷管理功能。**lvmlockd** 还需要集群基础架构。
- HAProxy – 在第 4 层（TCP）和第 7 层（HTTP, HTTPS）服务中提供高可用性负载平衡和故障转移的路由软件。

1.2. 高可用性附加组件概念

红帽高可用性附加组件集群的一些主要概念如下：

1.2.1. 隔离

如果与集群中的单个节点通信失败，则集群中的其他节点必须能够限制或释放对故障集群节点可能有权访问的资源的访问。无法通过联系集群节点本身来实现。因为集群节点可能没有响应。反之，必须提供一个外部的方法来实现。这个方法为称为隔离（fencing）。隔离设备是一个外部设备，集群使用它用来限制错误节点对共享资源的访问，或对集群的节点执行硬重启。

如果没有配置隔离设备，您就无法知道以前被出现问题的集群节点使用的资源已被释放，这可能会阻止服务在集群的其他节点中运行。因此，该系统可能会错误地假设集群节点释放了它的资源，从而可能导致数据崩溃和数据丢失。没有隔离设备配置的数据完整性就无法保证，集群配置将不被支持。

当隔离进行时，不允许执行其他集群操作。在隔离完成前，或集群节点重启后重新加入集群前，集群的正常操作不能恢复。

有关隔离的更多信息，请参阅红帽高可用性集群中的保护。

1.2.2. 仲裁

为了保持集群的完整性和可用性，集群系统使用一个称为*仲裁（quorum）*的概念来防止数据崩溃和丢失。当超过一半的集群节点在线时，集群就已被"仲裁"。为减少由于失败造成数据崩溃的机会，在不满足仲裁数量的情况下，Pacemaker 会默认停止所有资源。

仲裁是通过一个投票（voting）系统来建立的。当一个集群节点工作不正常，或丢掉了与其他集群部分的通信，则大多数工作的节点可以通过投票来分离有问题的节点，如果需要，对节点进行隔离。

例如，在一个 6 个节点集群中，在至少有 4 个集群节点正常工作时就满足了仲裁。如果大多数节点离线或不可用，集群就不再具仲裁数量，Pacemaker 会停止集群的服务。

Pacemaker 仲裁的功能可以防止出现*脑裂（split-brain）*问题。当集群中出现无法相互通信的部分，而每个部分都可以在自己的部分中作为一个独立的集群运行，则代表集群出现了脑裂的问题。这可能会导致出现数据被破坏的问题。有关它脑裂状态的含义以及一般仲裁概念的更多信息，请参阅 探索 RHEL 高可用性集群的概念-仲裁。

Red Hat Enterprise Linux 高可用性附加组件集群使用 **votequorum** 服务，并结合隔离，以避免脑裂的情况。为集群中的每个系统分配一组投票机制，只能在大多数投票机制都存在时才允许执行集群操作。

1.2.3. 集群资源

*集群资源*是一个由集群服务管理的程序、数据或应用程序实例。这些资源通过*代理（agent）*作为一个标准接口，用来在集群环境中管理资源。

为确保资源健康，您可以在资源的定义中添加监控操作。如果您没有为资源指定监控操作，则会默认添加一个。

您可以通过配置*约束（constraint）*来决定集群中的资源行为。您可以配置以下约束类别：

- 位置约束 - 位置约束决定资源可在哪个节点上运行。
- 排序约束 - 排序约束决定资源运行的顺序。
- 共同位置约束 - 共同位置约束（colocation constraint）决定资源相对于其他资源的位置。

集集的一个最常见的元素是一组资源，这些资源需要放置在一起，并按顺序启动并按反顺序停止。为简化此配置，Pacemaker 支持*组*的概念。

1.3. PACEMAKER 概述

Pacemaker 是一个集群资源管理器。它通过使用集群基础结构的消息和成员资格功能来实现集群服务和资源的最大可用性。

1.3.1. Pacemaker 架构组件

使用 Pacemaker 配置的集群由独立的组件守护进程组成,这些守护进程监控集群成员资格、管理服务的脚本以及监控不同资源的管理子系统。

以下组件组成 Pacemaker 架构：

Cluster Information Base（CIB）

Pacemaker 信息守护进程使用 XML 在内部向所有其他集群节点分发和同步当前配置和状态信息。这些信息来自于 DC（Designated Coordinator），它是由 Pacemaker 分配的用来存储和发布集群状态和动作的节点。

集群资源管理守护进程（CRMd）

Pacemaker 集群资源操作通过这个守护进程进行路由。由 CRMd 管理的资源可由客户端系统查询，并在需要时进行移动、实例化和更改。
每个节点还包括一个本地资源管理器守护进程（LRMd），它充当 CRMd 和资源间的接口。LRMd 将命令从 CRMd 传递给代理，如启动和停止状态信息。

STONITH（Shoot the Other Node in the Head）

STONITH 是 Pacemaker 的隔离（fencing）实现。它在 Pacemaker 中作为集群资源使用,用于处理保护请求,强制关闭节点并从集群中移除它们以确保数据的完整性。STONITH 在 CIB 中配置，它可以作为普通的集群资源被监控。

corosync

corosync 是用于高可用性集群的核心成员资格和成员沟通需要的组件。它使用一个同名称的守护进程。它是 High Availability Add-On 正常运行所必需的功能。
除了成员资格和消息功能外，**corosync** 还 ：

- 管理仲裁规则并进行裁定。
- 为在集群的多个成员间协调或操作的应用程序提供消息功能，因此必须在实例间进行有状态或其他信息通信。
- 使用 **kronosnet** 库作为其网络传输，提供多个冗余链接和自动故障转移。

1.3.2. Pacemaker 配置和管理工具

High Availability Add-On 有两个配置工具用于集群部署、监控和管理。

pcs

pcs 命令行界面控制并配置 Pacemaker 和 **corosync** 心跳守护进程。基于命令行的程序 **pcs** 可以执行以下集群管理任务：

- 创建并配置 Pacemaker/Corosync 集群
- 在集群运行时修改集群配置
- 远程配置 Pacemaker 和 Corosync 以及启动、停止和显示集群状态信息

pcsd Web UI

用于创建和配置 Pacemaker/Corosync 集群的图形用户界面。

1.3.3. 集群和 Pacemaker 配置文件

红帽高可用性附加组件的配置文件是 **corosync.conf** 和 **cib.xml**。

corosync.conf 文件提供了 **corosync** 所使用的集群参数，corosync 是构建 Pacemaker 的集群管理器。通常，您不应该直接编辑 **corosync.conf**，而是使用 **pcs** 或 **pcsd** 接口。

cib.xml 文件是一个 XML 文件，代表集群配置和集群中所有资源的当前状态。Pacemaker 的集群信息基础（Cluster Information Base，CIB）会使用这个文件。CIB 的内容会在整个集群中自动保持同步。不要直接编辑 **cib.xml** 文件；要使用 **pcs** 或 **pcsd** 界面。

1.4. 红帽高可用性集群中的 LVM 逻辑卷

红帽高可用性附加组件以两个不同的集群配置为 LVM 卷提供支持。

您可以选择的集群配置如下：

- 带有主动/被动故障转换配置的高可用性 LVM 卷（HA-LVM），在任何一个时间点上，集群中只能有一个节点访问存储。
- 使用 **lvmlockd** 守护进程管理主动/主动配置中的存储设备的 LVM 卷，在任何一个时间点上，集群中可以有多个节点同时访问存储。**lvmlockd** 守护进程是 Resilient Storage 附加组件的一部分。

1.4.1. 选择 HA-LVM 或者共享卷

使用 HA-LVM 或者使用由 **lvmlockd** 守护进程管理的共享逻辑卷，应该根据要部署的应用程序或服务的要求来决定。

- 如果集群的多个节点需要同步对 active/active 系统中的 LVM 卷进行读/写访问，则必须使用 **lvmlockd** 守护进程并将您的卷配置为共享卷。**lvmlockd** 守护进程提供了一个系统，用于协调集群节点间 LVM 卷的激活和更改。**lvmlockd** 守护进程的锁定服务为 LVM 元数据提供了保护，因为集群的不同节点与卷交互并更改其布局。这种保护取决于将同时在多个群集节点间作为共享卷激活的卷组。
- 如果将高可用性集群配置为以主动/被动方式管理共享资源，同时只有一个成员需要访问给定 LVM 卷，那么您可以在没有 **lvmlockd** 锁定服务的情况下使用 HA-LVM。

因为大多数应用程序没有为与其他实例同时运行而设计或进行优化，所有它们以主动/被动配置的模式运行更佳。选择在共享逻辑卷上运行不是集群感知的应用程序可能会导致性能下降。这是因为，在这些情况下逻辑卷本身需要有集群通信的额外开销。针对集群设计的应用程序所获得的性能提高幅度必须大于因为集群文件系统和针对集群的逻辑卷所造成的性能降低的幅度。一些应用程序和工作负载会比其他应用程序和工作负载更容易实现这一点。确定集群的要求以及是否要为活跃/主动集群进行优化，从而在两个 LVM 配置间进行选择。大多数用户使用 HA-LVM 一般会获得最佳的 HA 结果。

HA-LVM 和使用 **lvmlockd** 的共享逻辑卷类似，它们可防止 LVM 元数据及其逻辑卷崩溃，否则在允许多个机器进行重叠更改时会出现这种情况。HA-LVM 会限制在一个时间点上只能有一个逻辑卷被激活，也就是说一次只在一个机器中激活。这意味着，只使本地（非集群）实现的存储驱动被使用。使用这种方法可以避免用于集群协调的额外开销，以提高性能。使用 **lvmlockd** 的共享卷不会强制实施这些限制，用户可以自由地在集群的所有机器上激活逻辑卷。这会强制使用支持集群功能的存储驱动，以允许支持集群功能的文件系统在上面运行。

1.4.2. 在集群中配置 LVM 卷

集群通过 Pacemaker 管理。只有与 Pacemaker 集群联合支持 HA-LVM 和共享逻辑卷，且必须配置为集群资源。

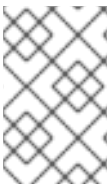
注意

如果 Pacemaker 集群使用的 LVM 卷组包含位于远程块存储（如 iSCSI 目标）上的一个或多个物理卷，则红帽建议您为目标配置一个 **systemd resource-agents-deps** 目标和一个 **systemd** 置入单元，以确保服务在 Pacemaker 启动之前启动。有关配置 **systemd resource-agents-deps** 目标的详情，请参阅 为不由 Pacemaker 管理的资源依赖项配置启动顺序。

- 有关将 HA-LVM 卷配置为 Pacemaker 集群一部分的流程的示例，请参阅 在红帽高可用性集群中配置一个主动/被动 Apache HTTP 服务器 和在红帽高可用性集群中配置一个主动/被动 NFS 服务器。
  请注意，这些步骤包括以下步骤：
  - 确保只有集群可以激活卷组
  - 配置 LVM 逻辑卷
  - 将 LVM 卷配置为集群资源
- 有关使用 **lvmlockd** 守护进程配置共享 LVM 卷来管理主动/主动配置下的存储设备的流程，请参阅 集群中的 GFS2 文件系统 和 在红帽高可用性集群中配置主动/主动 Samba 服务器。

第 2 章 PACEMAKER 入门

要熟悉您用来创建 Pacemaker 集群的工具和进程，您可以执行以下流程。这些内容适用于想了解集群软件以及如何管理它，而不需要配置集群的用户。



注意

这些步骤并不会创建受支持的红帽集群。受支持的红帽集群至少需要两个节点并配置隔离设备。有关红帽对 RHEL 高可用性集群的支持政策、要求和限制的详情，请参考 RHEL 高可用性集群的支持政策。

2.1. 学习使用 PACEMAKER

通过这个过程，您将了解如何使用 Pacemaker 设置集群、如何显示集群状态以及如何配置集群服务。这个示例创建了一个 Apache HTTP 服务器作为集群资源，并显示了集群在资源失败时如何响应。

在本例中：

- 节点为 **z1.example.com**。
- 浮动 IP 地址为 192.168.122.120。

先决条件

- 一个用于运行 RHEL 9 的单个节点
- 一个浮动的 IP 地址，它与一个节点静态分配的 IP 地址处于同一个网络。
- 运行的节点的名称位于 **/etc/hosts** 文件中

步骤

1. 从 High Availability 频道安装 Red Hat High Availability Add-On 软件包，然后启动并启用 **pcsd** 服务。

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

   如果您正在运行 **firewalld** 守护进程，启用红帽高可用性附加组件所需的端口。

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

2. 在集群的每个节点上为用户 **hacluster** 设置密码，并为您要从中运行 **pcs** 命令的集群中每个节点验证用户 **hacluster**。本例只使用一个节点，您要从这个节点中运行命令。把这一步包括在这个步骤的原因是，它是配置一个被支持的红帽高可用性多节点集群的一个必要步骤。

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

3. 创建名为 **my_cluster** 的集群，具有一个成员并检查集群的状态。这个命令会创建并启动集群。

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

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

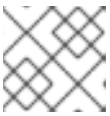
4. 红帽高可用性集群要求为集群配置隔离功能。需要满足这个要求的原因包括在 Red Hat High Availability 集群中的隔离中。但是，这里只显示如何使用基本的 Pacemaker 命令，因此将 **stonith-enabled** 集群选项设置为 **false** 来禁用隔离功能。

警告

对生产集群而言，不要使用 **stonith-enabled=false**。它通知集群，假设出现故障的节点已被安全隔离。

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

5. 在您的系统中配置网页浏览器并创建一个网页来显示简单文本信息。如果您正在运行 **firewalld** 守护进程，启用 **httpd** 所需的端口。



注意

不要使用 **systemctl enable** 启用任何由集群管理的服务在系统引导时启动。

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

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

要让 Apache 资源代理获得 Apache 状态，在现有配置中添加以下内容来启用状态服务器 URL。

```
# cat <<-END > /etc/httpd/conf.d/status.conf
```

```
<Location /server-status>
SetHandler server-status
Order deny,allow
Deny from all
Allow from 127.0.0.1
Allow from ::1
</Location>
END
```

6. 创建 **IPaddr2** 和 **apache** 资源，供集群管理。'IPaddr2' 资源是一个浮动 IP 地址，它不能是一个已经与物理节点关联的 IP 地址。如果没有指定 'IPaddr2' 资源的 NIC 设备，浮动 IP 必须位于与静态分配的 IP 地址相同的网络中。
   您可以使用 **pcs resource list** 命令显示所有可用资源类型的列表。您可以使用 **pcs resource describe *resourcetype*** 命令显示您可以为指定资源类型设置的参数。例如，以下命令显示您可以为类型为 **apache** 的资源设置的参数：

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

   在这个示例中，IP 地址资源和 apache 资源都配置为名为 **apachegroup** 的组的一部分，这样可确保在配置正常工作的多节点集群时让资源在同一节点中运行。

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

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

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

1 node configured
2 resources configured

Online: [ z1.example.com ]

Full list of resources:

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

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

   配置群集资源后，您可以使用 **pcs resource config** 命令显示为该资源配置的选项。

```
# pcs resource config WebSite
Resource: WebSite (class=ocf provider=heartbeat type=apache)
```

```
 Attributes: configfile=/etc/httpd/conf/httpd.conf statusurl=http://localhost/server-status
 Operations: start interval=0s timeout=40s (WebSite-start-interval-0s)
             stop interval=0s timeout=60s (WebSite-stop-interval-0s)
             monitor interval=1min (WebSite-monitor-interval-1min)
```

7. 将浏览器指向使用您配置的浮动 IP 地址创建的网站。这个命令会显示您定义的文本信息。

8. 停止 apache web 服务并检查集群的状态。使用 **killall -9** 模拟应用程序级别的崩溃。

```
# killall -9 httpd
```

检查集群状态。您应该看到停止 web 服务会导致操作失败，但集群软件会重启该服务，您应该仍然可以访问网站。

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

Online: [ z1.example.com ]

Full list of resources:

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

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

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

您可以在服务启动并再次运行后，清除失败的资源中的失败状态。当您查看集群状态时，失败的操作通知将不再出现。

```
# pcs resource cleanup WebSite
```

9. 当您完成查看集群和集群状态后，停止节点上的集群服务。虽然在这里只在一个节点中只启动这个服务，但包含了 **--all** 参数，它会停止在一个实际的多节点集群中的所有节点上的集群服务。

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

2.2. 了解配置故障转移

以下流程介绍了创建运行服务的一个 Pacemaker 集群，当节点上的服务变为不可用时，将其从一个节点切换到另一个节点上。通过这个步骤，您可以了解如何在双节点集群中创建服务，并可以查看在运行该服务的节点出现问题时会出现什么情况。

这个示例步骤配置一个运行 Apache HTTP 服务器的双节点 Pacemaker 集群。然后，您可以停止一个节点上的 Apache 服务来查看该服务仍然可用。

在本例中：

- 节点为 **z1.example.com** 和 **z2.example.com**。
- 浮动 IP 地址为 192.168.122.120。

先决条件

- 两个可以相互通讯的、运行 RHEL 9 的节点
- 一个浮动的 IP 地址，它与一个节点静态分配的 IP 地址处于同一个网络。
- 运行的节点的名称位于 **/etc/hosts** 文件中

步骤

1. 在这两个节点中，通过 High Availability 频道安装 Red Hat High Availability Add-On 软件包，并启动并启用 **pcsd** 服务。

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

   如果您正在运行 **firewalld** 守护进程，在两个节点上启用红帽高可用性附加组件所需的端口。

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

2. 在集群的两个节点上为用户 **hacluster** 设置密码。

```
# passwd hacluster
```

3. 在要运行 **pcs** 命令的节点上，为集群中的每个节点验证用户 **hacluster**。

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

4. 创建名为 **my_cluster** 的集群，两个节点都作为集群成员。这个命令会创建并启动集群。因为 **pcs** 配置命令对整个集群的影响，您只需要从集群的一个节点上运行。
   在集群的一个节点中运行以下命令。

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

5. 红帽高可用性集群要求为集群配置隔离功能。需要满足这个要求的原因包括在 Red Hat High Availability 集群中的隔离中。在这里，仅显示在这个配置中故障转移是如何工作的。把 **stonith-enabled** 集群选项设置为 **false** 来禁用隔离

警告

对生产集群而言，不要使用 **stonith-enabled=false**。它通知集群，假设出现故障的节点已被安全隔离。

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

6. 创建集群并禁用隔离后，检查集群的状态。

注意

运行 **pcs cluster status** 命令时，可能会显示与系统组件启动时稍有不同示例的输出。

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

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

7. 在这两个节点中，配置网页浏览器并创建一个网页来显示简单的文本信息。如果您正在运行 **firewalld** 守护进程，启用 **httpd** 所需的端口。

注意

不要使用 **systemctl enable** 启用任何由集群管理的服务在系统引导时启动。

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

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

要让 Apache 资源代理获得 Apache 状态，集群中的每个节点都会在现有配置之外创建一个新的配置来启用状态服务器 URL。

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

8. 创建 **IPaddr2** 和 **apache** 资源，供集群管理。'IPaddr2' 资源是一个浮动 IP 地址，它不能是一个已经与物理节点关联的 IP 地址。如果没有指定 'IPaddr2' 资源的 NIC 设备，浮动 IP 必须位于与静态分配的 IP 地址相同的网络中。
   您可以使用 **pcs resource list** 命令显示所有可用资源类型的列表。您可以使用 **pcs resource describe *resourcetype*** 命令显示您可以为指定资源类型设置的参数。例如，以下命令显示您可以为类型为 **apache** 的资源设置的参数：

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

   在这个示例中，IP 地址资源和 apache 资源都配置为名为 **apachegroup** 的组的一部分，这样可确保这些资源在同一节点中运行。

   在集群中的一个节点中运行以下命令：

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

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

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

2 nodes configured
2 resources configured

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

Full list of resources:

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

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

请注意，在这个实例中，**apachegroup** 服务在节点 z1.example.com 中运行。

9. 访问您创建的网站，在运行该服务的节点上停止运行该服务，查看该服务如何切换到第二个节点。

   a. 将浏览器指向使用您配置的浮动 IP 地址创建的网站。这会显示您定义的文本信息，显示运行网站的节点名称。

   b. 停止 apache web 服务。使用 **killall -9** 模拟应用程序级别的崩溃。

   ```
   # killall -9 httpd
   ```

   检查集群状态。您应该可以看到，停止 web 服务会导致操作失败，但集群软件在运行该服务的节点中重启该服务，所以您应该仍然可以访问网页浏览器。

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

   2 nodes configured
   2 resources configured

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

   Full list of resources:

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

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

   在服务启动并再次运行后，清除失败状态。

   ```
   # pcs resource cleanup WebSite
   ```

   c. 将运行该服务的节点设置为待机模式。请注意，由于禁用了隔离功能，因此我们无法有效地模拟节点级别的故障（比如拔掉电源电缆）。需要隔离功能集群才可以在出现这类问题时被恢复。

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

   d. 检查集群的状态并记录该服务正在运行的位置。

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

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

2 nodes configured
2 resources configured

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

Full list of resources:

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

e. 访问网站。服务应该仍然可用，显示信息应该指示服务正在运行的节点。

10. 要将集群服务恢复到第一个节点，让节点离开待机模式。这不一定将该服务转换到第一个节点。

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

11. 最后，进行清理，停止两个节点上的集群服务。

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

第 3 章 PCS 命令行界面

pcs 命令行界面通过为其配置文件提供一个简单的接口来控制并配置集群服务，如 **corosync**、**pacemaker**、**booth** 和 **sbd**。

请注意，您不应该直接编辑 **cib.xml** 配置文件。在大多数情况下，Pacemaker 会拒绝直接修改的 **cib.xml** 文件。

3.1. PCS HELP DISPLAY

您可以使用 **pcs** 的 **-h** 选项显示 **pcs** 命令的参数以及这些参数的描述。

以下命令显示 **pcs resource** 命令的参数。

```
# pcs resource -h
```

3.2. 查看原始集群配置

虽然您不应该直接编辑集群配置文件，但您可以使用 **pcs cluster cib** 命令查看原始集群配置。

您可以使用 **pcs cluster cib *filename*** 命令将原始集群配置保存到指定的文件中。如果您之前已经配置了集群，且已经有一个活跃的 CIB，则使用以下命令保存原始 xml 文件。

```
pcs cluster cib filename
```

例如，以下命令可将 CIB 中的原始 xml 保存到名为 **testfile** 的文件中：

```
# pcs cluster cib testfile
```

3.3. 将配置更改保存到工作文件中

配置集群时，您可以在不影响活跃 CIB 的情况下将配置更改保存到指定的文件中。这可让您在每次单独的更新时都指定配置更新而无需立即更新当前运行的集群配置。

有关将 CIB 保存到文件的详情，请参考 查看原始集群配置。创建该文件后，您可以使用 **pcs** 命令的 **-f** 选项将配置更改保存到该文件而不是活跃的 CIB 中。当您完成更改并准备好更新活跃 CIB 文件后，您可以使用 **pcs cluster cib-push** 命令来推送这些文件更新。

步骤

以下是将更改推送到 CIB 文件的建议步骤。这个过程创建原始保存的 CIB 文件的副本并修改该副本。将这些更改推送到活跃 CIB 时，这个过程指定了 **pcs cluster cib-push** 命令的 **diff-against** 选项，以便只有原始文件与更新文件之间的更改推送到 CIB。这允许用户并行进行更改而不会相互覆盖其内容，这可以减少 Pacemaker（它不需要解析整个配置文件）的负载。

1. 将活动的 CIB 保存到文件中。这个示例将 CIB 保存到名为 **original.xml** 的文件。

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

2. 将保存的文件复制到您要用于配置更新的工作文件中。

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

3. 根据需要更新您的配置。以下命令在 **update.xml** 文件中创建一个资源，但不将该资源添加到当前运行的集群配置中。

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

4. 将更新的文件推送到活跃的 CIB 中，指定您只推送对原始文件进行的更改。

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

另外，您可以使用以下命令推送 CIB 文件的整个内容。

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

在推送整个 CIB 文件时，Pacemaker 会检查这个版本，并不允许推送比集群中已存在的 CIB 文件更早的文件。如果您需要更新整个 CIB 文件，其版本早于集群中的当前版本，可以使用 **pcs cluster cib-push** 命令的 **--config** 选项。

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

3.4. 显示集群状态

您可以使用各种命令来显示集群及其组件的状态。

您可以使用以下命令显示集群和集群资源的状态。

```
# pcs status
```

您可以使用 **pcs status** 命令的 *command* 参数显示特定集群组件的状态，指定 **resources**, **cluster**, **nodes**, 或 **pcsd**。

```
pcs status commands
```

例如，以下命令显示集群资源的状态。

```
# pcs status resources
```

以下命令显示集群的状态，但不显示集群资源。

```
# pcs cluster status
```

3.5. 显示完整的集群配置

使用以下命令显示完整的集群的当前配置。

```
# pcs config
```

3.6. 使用 PCS 命令修改 COROSYNC.CONF 文件

您可以使用 **pcs** 命令修改 **corosync.conf** 文件中的参数。

以下命令修改 **corosync.conf** 文件中的参数。

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

以下示例命令会更新 **knet_pmtud_interval** 传输值，以及 **token** 和 **join** 令牌值。

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

其他资源

- 有关从现有集群中添加和删除节点的详情，请参考 管理集群节点。
- 有关在现有集群中添加和修改链接的详情，请参考 在现有集群中添加和修改链接。
- 有关修改集群中仲裁选项和管理仲裁设备设置的详情，请参考 配置集群仲裁 和 配置仲裁设备。

3.7. 使用 PCS 命令显示 COROSYNC.CONF 文件

以下命令显示 **corosync.conf** 集群配置文件的内容。

```
# pcs cluster corosync
```

您可以使用 **pcs cluster config** 命令以人类可读格式打印 **corosync.conf** 文件的内容，如下例所示：

如果集群是在 RHEL 9.1 或更高版本中创建的，或者 UUID 是手动添加的，如 通过 UUID 识别集群 中所述，则这个命令的输出包括集群的 UUID。

```
[root@r8-node-01 ~]# pcs cluster config
Cluster Name: HACluster
Cluster UUID: ad4ae07dcafe4066b01f1cc9391f54f5
Transport: knet
Nodes:
  r8-node-01:
    Link 0 address: r8-node-01
    Link 1 address: 192.168.122.121
    nodeid: 1
  r8-node-02:
    Link 0 address: r8-node-02
    Link 1 address: 192.168.122.122
    nodeid: 2
Links:
  Link 1:
    linknumber: 1
    ping_interval: 1000
    ping_timeout: 2000
    pong_count: 5
Compression Options:
  level: 9
  model: zlib
  threshold: 150
Crypto Options:
  cipher: aes256
```

```
  hash: sha256
Totem Options:
  downcheck: 2000
  join: 50
  token: 10000
Quorum Device: net
  Options:
    sync_timeout: 2000
    timeout: 3000
  Model Options:
    algorithm: lms
    host: r8-node-03
  Heuristics:
    exec_ping: ping -c 1 127.0.0.1
```

您可以使用 **--output-format=cmd** 选项运行 **pcs cluster config show** 命令，以显示可用于重新创建现有 **corosync.conf** 文件的 **pcs** 配置命令，如下例所示。

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

第 4 章 使用 PACEMAKER 创建红帽高可用性集群

通过以下流程，使用 **pcs** 命令行界面创建一个红帽高可用双节点集群。

在本例中配置集群需要您的系统包含以下组件：

- 2 个节点，用于创建集群。在本例中，所用的节点为 **z1.example.com** 和 **z2.example.com**。
- 专用网络的网络交换机。我们推荐使用专用网络用于集群节点和其它集群硬件（比如网络电源交换机和光线通道交换机）的通信，当这不是必须的。
- 集群中的每个节点上都有一个隔离设备。这个示例使用 APC 电源交换机的两个端口，主机名为 **zapc.example.com**。

4.1. 安装集群软件

安装集群软件，并使用以下流程配置您的系统以进行集群创建。

步骤

1. 在集群的每个节点中，启用与您的系统架构对应的高可用性存储库。例如，要为 x86_64 系统启用高可用性存储库，您可以输入以下 **subscription-manager** 命令：

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

2. 在集群的每个节点中，安装 Red Hat High Availability Add-On 软件包，以及 High Availability 性频道中的所有可用的隔离代理。

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

另外，您可以使用以下命令安装 Red Hat High Availability Add-On 软件包，并只安装您需要的隔离代理。

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

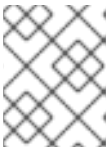
以下命令显示可用隔离代理列表。

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

警告

在安装 the Red Hat High Availability Add-On 软件包后，需要确定设置了软件更新首选项，以便不会自动安装任何软件。在正在运行的集群上安装可能会导致意外行为。如需更多信息，请参阅将软件更新应用到 RHEL High Availability 或弹性存储集群的建议实践。

3. 如果您正在运行 **firewalld** 守护进程，请执行以下命令启用红帽高可用性附加组件所需的端口。



注意

您可以使用 **rpm -q firewalld** 命令确定您的系统中是否安装了 **firewalld** 守护进程。如果已安装，您可以使用 **firewall-cmd --state** 命令确定它是否正在运行。

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

注意

集群组件的理想防火墙配置取决于本地环境，您可能需要考虑节点是否有多个网络接口或主机外防火墙是否存在。在此示例中打开 Pacemaker 集群通常所需的端口，您需要根据具体情况进行修改。为高可用性附加组件启用端口显示为红帽高可用性附加组件启用的端口，并提供每个端口的用途信息。

4. 要使用 **pcs** 配置集群并在节点间通信，您必须在每个节点中为用户 ID **hacluster**（即 **pcs** 管理帐户）设置密码。建议每个节点上的用户 **hacluster** 的密码都相同。

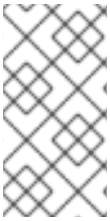
```
# passwd hacluster
Changing password for user hacluster.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
```

5. 在配置集群前，必须启动并启用 **pcsd** 守护进程，以便在每个节点上启动。此守护进程与 **pcs** 命令配合使用，以管理集群中跨节点的配置。
在集群的每个节点上执行以下命令启动 **pcsd** 服务并在系统启动时启用 **pcsd**。

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

4.2. 安装 PCP-ZEROCONF 软件包（推荐使用）

设置集群时，建议您为 Performance Co-Pilot(PCP)工具安装 **pcp-zeroconf** 软件包。PCP 是红帽推荐的 RHEL 系统资源监控工具。安装 **pcp-zeroconf** 软件包可让您运行 PCP，并收集 performance-monitoring 数据，以便调查隔离、资源故障以及破坏集群的其他事件。



注意

启用 PCP 的集群部署需要足够的空间可用于 PCP 捕获的文件系统上的数据，其中包含 **/var/log/pcp/**。PCP 空间使用量因部署而异，但在使用 **pcp-zeroconf** 默认设置时，10Gb 通常就足够了，有些环境的需要可能会更少。监控这个目录中的 14 天期间内的典型活动提供更准确的信息。

步骤

要安装 **pcp-zeroconf** 软件包，请运行以下命令。

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

这个软件包启用了 **pmcd**，并以 10 秒的间隔设置数据捕获。

有关查看 PCP 数据的详情，请查看 为什么 RHEL 高可用性集群节点重启 - 如何在红帽客户门户网站中防止其再次发生？

4.3. 创建高可用性集群

使用以下流程创建一个红帽高可用性附加组件集群。这个示例流程创建一个由节点 **z1.example.com** 和 **z2.example.com** 组成的集群。

步骤

1. 在要运行 **pcs** 的节点上，为集群中的每个节点验证 **pcs** 用户 **hacluster**。
   以下命令对由 **z1.example.com** 和 **z2.example.com** 组成的集群中的两个节点验证 **z1.example.com** 上的用户 **hacluster** ：

```
[root@z1 ~]# pcs host auth z1.example.com z2.example.com
Username: hacluster
Password:
z1.example.com: Authorized
z2.example.com: Authorized
```

2. 从 **z1.example.com** 执行以下命令，创建由 **z1.example.com** 和 **z2.example.com** 组成的双节点集群 **my_cluster**。这会将集群配置文件传播到集群中的两个节点。此命令包含 **--start** 选项，该选项将在集群的两个节点上启动群集服务。

```
[root@z1 ~]# pcs cluster setup my_cluster --start z1.example.com z2.example.com
```

3. 在节点引导时，启用集群服务在集群中的每个节点上运行。

注意

对于特定环境，您可以跳过这一步来禁用集群服务。这可让您确保在节点重新加入集群前解决集群或您的资源中的任何问题。如果禁用了集群服务，则需要在重新引导节点时手动启动该服务，方法是在该节点上执行 **pcs cluster start** 命令。

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

您可以使用 **pcs cluster status** 命令显示集群的当前状态。因为在使用 **pcs cluster setup** 命令的 **--start** 选项启动集群服务时，集群启动和运行可能有一些延迟，所以您应该确保在集群及其配置上执行后续操作前集群已启动并运行。

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

...
```

4.4. 创建使用多个链接的高可用性集群

您可以通过指定每个节点的所有链接，使用 **pcs cluster setup** 命令创建带有多个链接的红帽高可用性集群。

创建具有两个链接的双节点集群的基本命令格式如下：

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

有关此命令的完整语法，请查看 **pcs**(8)man page。

当创建具有多个链接的集群时，您应该考虑以下内容。

- **addr=*address*** 参数的顺序非常重要。节点名称后指定的第一个地址为 **link0**，第二个地址用于 **link1**，以此类推。
- 默认情况下，如果没有为链接指定 **link_priority**，则链接的优先级等于链接号。然后，链接优先级为 0、1、2、3 等，以此类推，0 是最高链路优先级。
- 默认链路模式是 **passive**，这代表带有最低编号链路优先级的主动链接。
- 使用 **link_mode** 和 **link_priority** 的默认值，指定的第一个链接将用作最高优先级链接，如果该链接失败，则将使用指定的下一个链接。
- 可以使用 **knet** 传输协议（即默认传输协议）指定最多 8 个链接。
- 所有节点必须具有相同数量的 **addr=** 参数。
- 可以使用 **pcs cluster link add**, **pcs cluster link remove**, **pcs cluster link delete**, 和 **pcs cluster link update** 命令在现有集群中添加、删除和更改链接。
- 与单链路集群一样，请勿将 IPv4 和 IPv6 地址混合到一个链接中，虽然您可以有一个链接运行 IPv4，另一个运行 IPv6。
- 与单链路集群一样，只要在一个单一的链接中没有混合使用 IPv4 和 IPv6，且名称可以被解析为 IPv4 或 IPv6 地址，就可以使用 IP 地址或名称来指定地址。

以下示例创建了名为 **my_twolink_cluster** 的双节点集群，它包括两个节点：**rh80-node1** 和 **rh80-node2**。**rh80-node1** 有两个接口，IP 地址 192.168.122.201 作为 **link0**，192.168.123.201 作为 **link1**。**rh80-node2** 有两个接口，IP 地址 192.168.122.202 作为 **link0**，192.168.123.202 作为 **link1**。

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

要将链接优先级设置为与默认值不同的值（即链接号），您可以使用 **pcs cluster setup** 命令的 **link_priority** 选项设置链接优先级。以下两个示例命令各自创建一个具有两个接口的双节点集群，其中第一个链接 0 具有链接优先级 1，而链接 1 的链接优先级为 0。首先使用链接 1，链接 0 将充当故障转移的链接。由于未指定链接模式，因此默认为被动（passive）。

这两个命令是等效的。如果您没有在 **link** 关键字之后指定链接号，**pcs** 接口会自动添加链接号，从最低未使用的链接编号开始。

```
# pcs cluster setup my_twolink_cluster rh80-node1 addr=192.168.122.201
```

```
addr=192.168.123.201 rh80-node2 addr=192.168.122.202 addr=192.168.123.202 transport knet
link link_priority=1 link link_priority=0

# pcs cluster setup my_twolink_cluster rh80-node1 addr=192.168.122.201
addr=192.168.123.201 rh80-node2 addr=192.168.122.202 addr=192.168.123.202 transport knet
link linknumber=1 link_priority=0 link link_priority=1
```

您可以将链接模式的值设置为与 **pcs cluster setup** 命令的 **link_mode** 选项的默认值 **passive** 不同的值，如下例所示：

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

以下示例同时设置链接模式和链接优先级。

```
# pcs cluster setup my_twolink_cluster rh80-node1 addr=192.168.122.201
addr=192.168.123.201 rh80-node2 addr=192.168.122.202 addr=192.168.123.202 transport knet
link_mode=active link link_priority=1 link link_priority=0
```

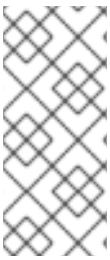
有关将节点添加到具有多个链接的现有集群中的详情，请参考 将一个节点添加到具有多个链接的集群中。

有关更改具有多个链接的现有集群中的链接的详情，请参考 在现有集群中添加和修改链接。

4.5. 配置隔离

您必须为集群中的每个节点配置保护设备。有关保护配置命令和选项的详情，请参考在红帽高可用性集群中配置隔离。

有关隔离的一般信息及其在红帽高可用性集群中的重要性，请参阅 红帽高可用性集群中的隔离。



注意

在配置隔离设备时，应该注意该设备是否与集群中的任何节点或者设备共享电源。如果某个节点及其隔离设备共享了电源，那么如果它的电源出现问题，集群可能就无法收到隔离功能的保护。这样的集群应该有冗余电源来保护设备和节点，或者具有没有和节点共享电源的额外的隔离设置。其他替代的隔离方法，比如 SBD 或存储隔离，也可以用来对电源问题提供冗余保护。

步骤

这个示例使用主机名 **zapc.example.com** 的 APC 电源开关来隔离节点，并使用 **fence_apc_snmp** 隔离代理。因为这两个节点都使用同一隔离代理实现隔离，所以您可以使用 **pcmk_host_map** 选项将这两个保护设备配置为单一资源。

您可以使用 **pcs stonith create** 命令将设备配置为 **stonith** 资源来创建隔离设备。以下命令配置名为 **myapc** 的 **stonith** 资源，它使用 **fence_apc_snmp** 节点 **z1.example.com** 和 **z2.example.com** 的隔离代理。**pcmk_host_map** 选项将 **z1.example.com** 映射到端口 1，**z2.example.com** 映射到端口 2。APC 设备的登录值和密码都是 **pc**。默认情况下，该设备对每个节点都使用 60 秒的监视间隔时间。

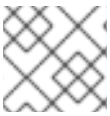
请注意，您可以在为节点指定主机名时使用 IP 地址。

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

以下命令显示现有隔离设备的参数。

```
[root@rh7-1 ~]# pcs stonith config myapc
 Resource: myapc (class=stonith type=fence_apc_snmp)
  Attributes: ipaddr=zapc.example.com pcmk_host_map=z1.example.com:1;z2.example.com:2
login=apc passwd=apc
  Operations: monitor interval=60s (myapc-monitor-interval-60s)
```

配置了隔离设备后，您应该测试该设备。有关测试隔离设备的详情，请参考 测试隔离设备。



注意

不要通过禁用网络接口来测试您的隔离设备，因为这不会正确测试隔离功能。

注意

当配置了隔离功能，且启动集群后，网络重启会触发节点的隔离，即使没有超过超时时间也会重启网络。因此，不要在集群服务运行时重启网络服务，因为它将在节点上触发意外隔离。

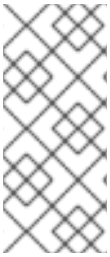
4.6. 备份和恢复集群配置

以下命令在 tar 归档中备份集群配置，并从备份中恢复所有节点上的集群配置文件。

步骤

使用以下命令在 tar 归档中备份集群配置。如果没有指定文件名，会使用标准输出。

```
pcs config backup filename
```



注意

pcs config backup 命令只备份 CIB 中配置的集群配置本身；资源守护进程的配置不在这个命令范围内。例如：如果您在集群中配置了 Apache 资源，则会备份资源设置（位于 CIB 中），而 Apache 守护进程设置（如'/etc/httpd' 中的设置）及其服务的文件不会被备份。同样，如果集群中配置了数据库资源，则不会备份数据库本身，而是备份数据库资源配置（CIB）。

使用以下命令从备份中恢复所有集群节点上的集群配置文件。指定 **--local** 选项只恢复您运行此命令的节点上的集群配置文件。如果没有指定文件名，将使用标准输入。

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

4.7. 为高可用性附加组件启用端口

集群组件的理想防火墙配置取决于本地环境，您可能需要考虑节点是否有多个网络接口或主机外防火墙是否存在。

如果您正在运行 **firewalld** 守护进程，请执行以下命令启用红帽高可用性附加组件所需的端口。

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

您可能需要修改开放端口以适合本地条件。

注意

您可以使用 **rpm -q firewalld** 命令确定您的系统中是否安装了 **firewalld** 守护进程。如果安装了 **firewalld** 守护进程，您可以使用 **firewall-cmd --state** 命令确定它是否正在运行。

下表显示了要为红帽高可用性附加组件启用的端口，并解释了该端口的用途。

表 4.1. 为高可用性附加组件启用的端口

端口	什么时候需要
TCP 2224	所有节点都需要默认的 **pcsd** 端口（pcsd Web UI 需要这个端口，节点到节点的通信也需要这个端口）。您可以使用 **/etc/sysconfig/pcsd** 文件中的 **PCSD_PORT** 参数配置 **pcsd** 端口。 关键是要以这样一种方式开放端口 2224，即来自任何节点的 **pcs** 可以与集群中的所有节点(包括它自己)通信。当使用 Booth 集群票据管理器或仲裁设备时，您必须在所有相关主机上打开端口 2224，如 Booth 仲裁器或仲裁设备主机。
TCP 3121	如果集群有 Pacemaker 远程节点，则所有节点都需要这个端口 整个集群节点上的 Pacemaker 的 **pacemaker-based** 守护进程将通过 Pacemaker 远程节点上的端口 3121 与 **pacemaker_remoted** 守护进程联系。如果使用一个单独的接口用于集群通信，则该端口只需要在那个接口上打开。至少应该在 Pacemaker 远程节点上完整集群节点打开这个端口。因为用户可以在完整节点和远程节点间转换主机，或使用主机网络在容器内运行远程节点，您可以为所有节点打开端口。不需要向节点以外的任何主机打开端口。
TCP 5403	当使用对仲裁设备使用 **corosync-qnetd** 时，仲裁设备主机上需要。可以使用 **corosync-qnetd** 命令的 **-p** 选项更改默认值。
UDP 5404-5412	corosync 节点需要这些端口以便在节点间进行通信。打开端口 5404-5412 非常重要，从而使来自任何节点的 **corosync** 都可以与集群中的所有节点（包括自身）进行通信。
TCP 21064	如果集群包含需要 DLM 的资源（如 **GFS2**），则在所有节点上都需要此项。

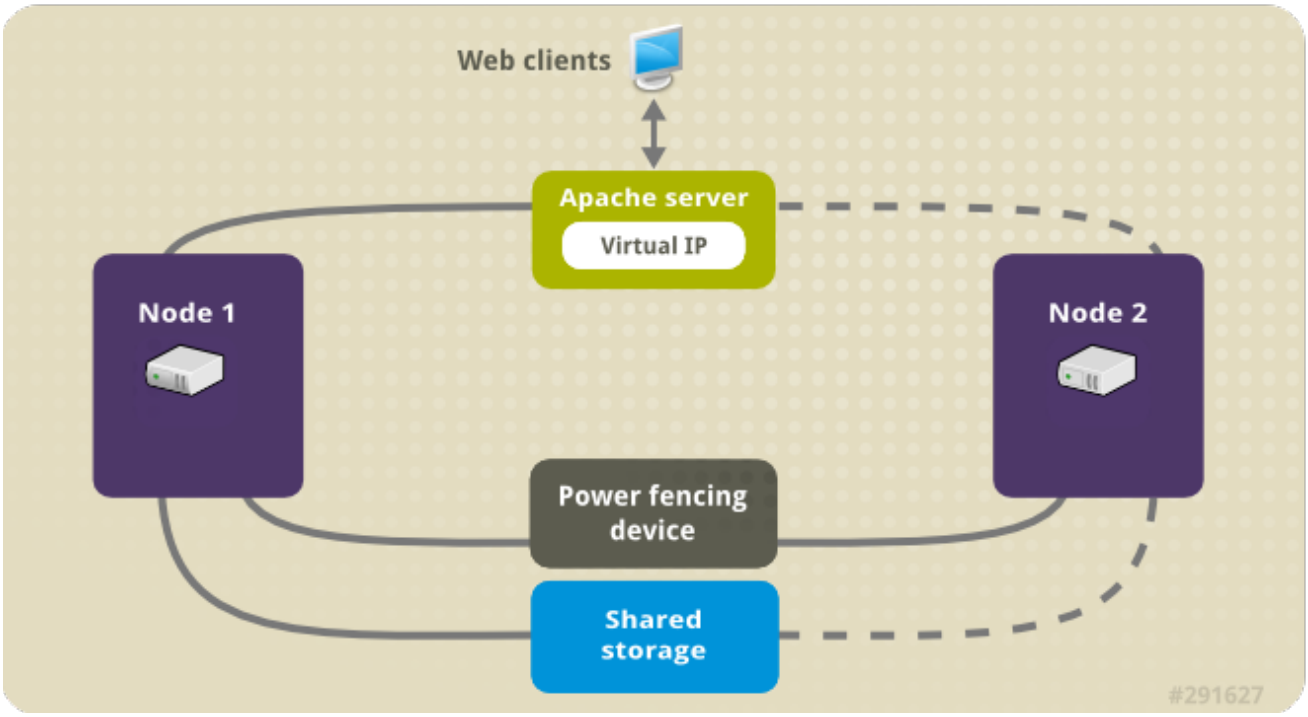
端口	什么时候需要
TCP 9929, UDP 9929	当使用 Booth ticket 管理器建立多站点集群时，需要在所有集群节点上打开 Booth 仲裁节点并从这些相同节点进行连接。

第 5 章 在红帽高可用性集群中配置主动/被动 APACHE HTTP 服务器

使用以下流程在双节点 Red Hat Enterprise Linux 高可用性附加组件集群中配置一个主动/被动 Apache HTTP 服务器。在这种情况下，客户端通过浮动 IP 地址访问 Apache HTTP 服务器。Web 服务器在集群的两个节点之一中运行。如果运行 web 服务器的节点出现问题，则 web 服务器会在集群的第二个节点上再次启动，以实现服务中断的最小化。

以下示例显示集群的高级概述，其中集群是双节点红帽高可用性集群，该集群使用网络电源交换机以及共享存储配置。集群节点连接到公用网络，以便客户端通过虚拟 IP 访问 Apache HTTP 服务器。Apache 服务器在 Node 1 或 Node 2 中运行，每个节点都可访问保存 Apache 数据的存储。本图例中，网页服务器在 Node 1 上运行，如果 Node 1 停止工作， Node 2 可以运行服务器。

图 5.1. Red Hat High Availability 双节点集群中的 Apache



这个用例需要您的系统包括以下组件：

- 一个双节点 Red Hat High Availability 集群，为每个节点配置了电源隔离功能。我们建议使用专用网络，但这不是必须的。此流程使用了创建带有 Pacemaker 的 Red Hat High-Availability 集群 中提供的集群示例。
- Apache 需要的公共虚拟 IP 地址。
- 集群中节点的共享存储，使用 iSCSI、光纤或其他共享网络块设备。

集群被配置为带有 Apache 资源组，其中包含 web 服务器所需的集群组件：LVM 资源、文件系统资源、IP 地址资源以及 web 服务器资源。这个资源组可以从集群的一个节点切换到另外一个节点，允许其中两个节点运行 web 服务器。在为集群创建资源组前，您将执行以下步骤：

1. 在逻辑卷 **my_lv** 上配置 XFS 文件系统。
2. 配置 web 服务器。

执行这些步骤后，您要创建资源组及其包含的资源。

5.1. 在 PACEMAKER 集群中配置具有 XFS 文件系统的 LVM 卷

使用以下流程在集群的节点之间共享的存储上创建一个 LVM 逻辑卷。

注意

LVM 卷以及集群节点使用的对应分区和设备必须只能连接到集群节点。

下面的流程创建了一个 LVM 逻辑卷，然后在那个卷上创建了一个 XFS 文件系统，以便在 Pacemaker 集群中使用。在这个示例中，使用共享分区 **/dev/sdb1** 来存储创建 LVM 逻辑卷的 LVM 物理卷。

步骤

1. 在集群的两个节点中，执行以下步骤将 LVM 系统 ID 的值设置为系统的 **uname** 标识符值。LVM 系统 ID 将用于确保只有集群可以激活卷组。

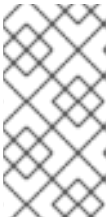
   a. 将 **/etc/lvm/lvm.conf** 配置文件中的 **system_id_source** 配置选项设置为 **uname**。

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

   b. 验证节点上的 LVM 系统 ID 是否与节点的 **uname** 匹配。

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

2. 创建 LVM 卷，并在那个卷上创建一个 XFS 文件系统。由于 **/dev/sdb1** 分区是共享的存储，因此您只在一个节点中执行这个操作过程。

   注意

   如果您的 LVM 卷组包含一个或多个位于远程块存储（如 iSCSI 目标）上的物理卷，则红帽建议您确保服务在 Pacemaker 启动之前启动。有关为 Pacemaker 集群使用的远程物理卷配置启动顺序的详情，请参考 为不由 Pacemaker 管理的资源依赖项配置启动顺序。

   a. 在分区 **/dev/sdb1** 上创建一个 LVM 物理卷。

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

   注意

   如果您的 LVM 卷组包含一个或多个位于远程块存储（如 iSCSI 目标）上的物理卷，则红帽建议您确保服务在 Pacemaker 启动之前启动。有关为 Pacemaker 集群使用的远程物理卷配置启动顺序的详情，请参考 为不由 Pacemaker 管理的资源依赖项配置启动顺序。

   b. 创建由物理卷 **/dev/sdb1** 组成的卷组 **my_vg**。

指定 **--setautoactivation n** 标志，以确保集群中 Pacemaker 管理的卷组在启动时不会自动激活。如果您要为要创建的 LVM 卷使用现有卷组，您可以使用 **vgchange --setautoactivation n** 命令为卷组重置此标记。

```
[root@z1 ~]# vgcreate --setautoactivation n my_vg /dev/sdb1
  Volume group "my_vg" successfully created
```

c. 确认新卷组带有您要运行的节点的系统 ID，并从这个节点中创建卷组。

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

d. 使用卷组 **my_vg** 创建逻辑卷。

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

您可以使用 **lvs** 命令显示逻辑卷。

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

e. 在逻辑卷 **my_lv** 上创建 XFS 文件系统。

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

3. 如果设备文件的使用是通过 **lvm.conf** 文件中的 **use_devicesfile = 1** 参数启用的，请将共享设备添加到集群中第二个节点上的设备文件中。此功能默认为启用。

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

5.2. 配置 APACHE HTTP 服务器

使用以下流程配置一个 Apache HTTP 服务器。

步骤

1. 确定在集群的每个节点中安装了 Apache HTTP Server。您还需要在集群中安装 **wget** 工具才能检查 Apache HTTP 服务器的状态。
在每个节点上执行以下命令。

```
# dnf install -y httpd wget
```

如果您在集群的每个节点上运行的 **firewalld** 守护进程启用了红帽高可用性附加组件所需的端口，并启用了运行 **httpd** 所需的端口。这个示例启用了 **httpd** 端口进行公共访问，但为 **httpd** 启用的特定端口可能会因生产用途而异。

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

2. 要让 Apache 资源代理获得 Apache 状态，集群中的每个节点都会在现有配置之外创建一个新的配置来启用状态服务器 URL。

```
# cat <<-END > /etc/httpd/conf.d/status.conf
<Location /server-status>
   SetHandler server-status
   Require local
</Location>
END
```

3. 为 Apache 创建网页供服务。
在集群的一个节点上，确保您在 配置具有 XFS 文件系统的 LVM 卷 时创建的逻辑卷已被激活，挂载您在该逻辑卷上创建的文件系统，在该文件系统上创建文件 **index.html**，然后卸载文件系统。

```
# lvchange -ay my_vg/my_lv
# mount /dev/my_vg/my_lv /var/www/
# mkdir /var/www/html
# mkdir /var/www/cgi-bin
# mkdir /var/www/error
# restorecon -R /var/www
# cat <<-END >/var/www/html/index.html
<html>
<body>Hello</body>
</html>
END
# umount /var/www
```

5.3. 创建资源和资源组

使用以下流程为集群创建资源。为确保这些资源在同一节点上运行，它们都配置为资源组 **apachegroup** 的一部分。要创建的资源如下，按其启动顺序列出。

1. 名为 **my_lvm** 的 **LVM-activate** 资源使用您在 配置具有 XFS 文件系统的 LVM 卷 时创建的 LVM 卷组。

2. 名为 **my_fs** 的 **Filesystem** 资源，它使用您在 配置具有 XFS 文件系统的 LVM 卷 时创建的文件系统设备 **/dev/my_vg/my_lv**。

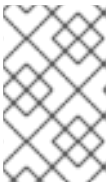
3. **IPaddr2** 资源，它是 **apachegroup** 资源组的浮动 IP 地址。IP 地址不能是一个已经与物理节点关联的 IP 地址。如果没有指定 **IPaddr2** 资源的 NIC 设备，浮动 IP 必须与节点静态分配的 IP 地址之一位于同一个网络中，否则无法正确检测到分配浮动 IP 地址的 NIC 设备。

4. 名为 **website** 的 **apache** 资源，它使用 **index.html** 文件和您在配置 Apache HTTP 服务器 中定义的 Apache 配置。

以下流程创建资源组 **apachegroup** 以及组包含的资源。资源将以您添加到组的顺序启动，并按照添加到组中的相反顺序停止。仅从集群的一个节点运行此步骤。

步骤

1. 以下命令创建 **LVM-activate** 资源 **my_lvm**。由于资源组 **apachegroup** 尚不存在，这个命令会创建资源组。



注意

不要配置多于一个的在主动/被动 HA 配置中使用相同 LVM 卷组的 **LVM-activate** 资源，因为这可能导致数据崩溃。另外，不要在主动/被动 HA 配置中将 **LVM-activate** 资源配置为克隆资源。

```
[root@z1 ~]# pcs resource create my_lvm ocf:heartbeat:LVM-activate vgname=my_vg
vg_access_mode=system_id --group apachegroup
```

当您创建资源时，会自动启动该资源。您可以使用以下命令确认资源已创建并启动。

```
# pcs resource status
 Resource Group: apachegroup
     my_lvm (ocf::heartbeat:LVM-activate): Started
```

您可以使用 **pcs resource disable** 和 **pcs resource enable** 命令手动停止和启动单独的资源。

2. 以下命令为配置创建剩余的资源，将它们添加到现有资源组 **apachegroup** 中。

```
[root@z1 ~]# pcs resource create my_fs Filesystem device="/dev/my_vg/my_lv"
directory="/var/www" fstype="xfs" --group apachegroup

[root@z1 ~]# pcs resource create VirtualIP IPaddr2 ip=198.51.100.3 cidr_netmask=24 --
group apachegroup

[root@z1 ~]# pcs resource create Website apache
configfile="/etc/httpd/conf/httpd.conf" statusurl="http://127.0.0.1/server-status" --
group apachegroup
```

3. 创建资源和包含这些资源的资源组后，您可以检查集群的状态。请注意，所有四个资源都在同一个节点上运行。

```
[root@z1 ~]# pcs status
Cluster name: my_cluster
Last updated: Wed Jul 31 16:38:51 2013
Last change: Wed Jul 31 16:42:14 2013 via crm_attribute on z1.example.com
Stack: corosync
Current DC: z2.example.com (2) - partition with quorum
Version: 1.1.10-5.el7-9abe687
2 Nodes configured
6 Resources configured

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

Full list of resources:
 myapc (stonith:fence_apc_snmp): Started z1.example.com
```

```
 Resource Group: apachegroup
     my_lvm (ocf::heartbeat:LVM-activate): Started z1.example.com
     my_fs (ocf::heartbeat:Filesystem): Started z1.example.com
     VirtualIP (ocf::heartbeat:IPaddr2): Started z1.example.com
     Website (ocf::heartbeat:apache): Started z1.example.com
```

请注意，如果您还没有为集群配置隔离设备，默认情况下资源不会启动。

4. 当集群启动并运行后，您可以将浏览器指向定义为 **IPaddr2** 资源，查看示例显示，包括一个简单的单词 "Hello"。

```
Hello
```

如果发现您配置的资源没有运行，您可以运行 **pcs resource debug-start *resource*** 命令来测试资源配置。

5. 当您使用 **apache** 资源代理来管理 Apache 时，它不会使用 **systemd**。因此，您必须编辑 Apache 提供的 **logrotate** 脚本，使其不使用 **systemctl** 重新加载 Apache。
在集群中的每个节点上删除 **/etc/logrotate.d/httpd** 文件中的以下行：

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

使用以下三行替换您删除的行，将 **/var/run/httpd-*website*.pid** 指定为 PID 文件路径，其中 *website* 是 Apache 资源的名称。在本例中，Apache 资源名称是 **Website**。

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

5.4. 测试资源配置

使用以下流程测试集群中的资源配置。

在创建资源和资源组中显示的集群状态，所有资源都在节点 **z1.example.com** 中运行。您可以使用以下步骤将第一个 **节点设置为待机模式** 来测试资源组是否切换到节点 **z2.example.com**，之后该节点将不再能够托管资源。

步骤

1. 以下命令将节点 **z1.example.com** 置于**待机模式**下。

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

2. 将节点 **z1** 置于**待机模式**后，检查集群状态。请注意，这些资源现在都应在 **z2** 中运行。

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

```
2 Nodes configured
6 Resources configured

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

Full list of resources:

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

定义的 IP 地址的网页仍会显示，而不中断。

3. 要从 **standby** 模式中删除 **z1**，请输入以下命令。

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

注意

从 **standby** 模式中删除节点不会自动导致资源恢复到该节点。这将依赖于 **resource-stickiness** 的值。有关 **resource-stickiness** 元属性的详情，请参考 配置资源以首选其当前节点。

第 6 章 在红帽高可用性集群中配置主动/被动模式的 NFS 服务器

红帽高可用性附加组件为在使用共享存储的 Red Hat Enterprise Linux 高可用性附加组件集群上运行高可用的主动/被动 NFS 服务器提供了支持。在以下示例中，您要配置一个双节点集群，其中客户端通过浮动 IP 地址访问 NFS 文件系统。NFS 服务器运行在集群中两个节点中的一个上。如果运行 NFS 服务器的节点出现问题，则 NFS 服务器会在集群的第二个节点上再次启动，以实现服务中断的最小化。

这个用例需要您的系统包括以下组件：

- 一个双节点 Red Hat High Availability 集群，为每个节点配置了电源隔离功能。我们建议使用专用网络，但这不是必须的。此流程使用了创建带有 Pacemaker 的 Red Hat High-Availability 集群 中提供的集群示例。
- NFS 服务器需要的一个公共虚拟 IP 地址。
- 集群中节点的共享存储，使用 iSCSI、光纤或其他共享网络块设备。

在现有双节点 Red Hat Enterprise Linux High Availability 集群中配置高可用性的主动/被动 NFS 服务器需要执行以下步骤：

1. 在共享存储的 LVM 逻辑卷中为集群中的节点配置文件系统。
2. 在 LVM 逻辑卷的共享存储中配置 NFS 共享。
3. 创建集群资源。
4. 测试您配置的 NFS 服务器。

6.1. 在 PACEMAKER 集群中配置具有 XFS 文件系统的 LVM 卷

使用以下流程在集群的节点之间共享的存储上创建一个 LVM 逻辑卷。

注意

LVM 卷以及集群节点使用的对应分区和设备必须只能连接到集群节点。

下面的流程创建了一个 LVM 逻辑卷，然后在那个卷上创建了一个 XFS 文件系统，以便在 Pacemaker 集群中使用。在这个示例中，使用共享分区 **/dev/sdb1** 来存储创建 LVM 逻辑卷的 LVM 物理卷。

步骤

1. 在集群的两个节点中，执行以下步骤将 LVM 系统 ID 的值设置为系统的 **uname** 标识符值。LVM 系统 ID 将用于确保只有集群可以激活卷组。
   a. 将 **/etc/lvm/lvm.conf** 配置文件中的 **system_id_source** 配置选项设置为 **uname**。

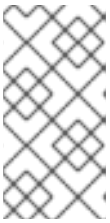
   ```
   # Configuration option global/system_id_source.
   system_id_source = "uname"
   ```

   b. 验证节点上的 LVM 系统 ID 是否与节点的 **uname** 匹配。

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

```
z1.example.com
```

2. 创建 LVM 卷，并在那个卷上创建一个 XFS 文件系统。由于 **/dev/sdb1** 分区是共享的存储，因此您只在一个节点中执行这个操作过程。

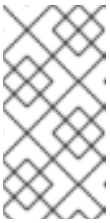


注意

如果您的 LVM 卷组包含一个或多个位于远程块存储（如 iSCSI 目标）上的物理卷，则红帽建议您确保服务在 Pacemaker 启动之前启动。有关为 Pacemaker 集群使用的远程物理卷配置启动顺序的详情，请参考 为不由 Pacemaker 管理的资源依赖项配置启动顺序。

a. 在分区 **/dev/sdb1** 上创建一个 LVM 物理卷。

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



注意

如果您的 LVM 卷组包含一个或多个位于远程块存储（如 iSCSI 目标）上的物理卷，则红帽建议您确保服务在 Pacemaker 启动之前启动。有关为 Pacemaker 集群使用的远程物理卷配置启动顺序的详情，请参考 为不由 Pacemaker 管理的资源依赖项配置启动顺序。

b. 创建由物理卷 **/dev/sdb1** 组成的卷组 **my_vg**。
指定 **--setautoactivation n** 标志，以确保集群中 Pacemaker 管理的卷组在启动时不会自动激活。如果您要为要创建的 LVM 卷使用现有卷组，您可以使用 **vgchange --setautoactivation n** 命令为卷组重置此标记。

```
[root@z1 ~]# vgcreate --setautoactivation n my_vg /dev/sdb1
  Volume group "my_vg" successfully created
```

c. 确认新卷组带有您要运行的节点的系统 ID，并从这个节点中创建卷组。

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

d. 使用卷组 **my_vg** 创建逻辑卷。

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

您可以使用 **lvs** 命令显示逻辑卷。

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

e. 在逻辑卷 **my_lv** 上创建 XFS 文件系统。

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

3. 如果设备文件的使用是通过 **lvm.conf** 文件中的 **use_devicesfile = 1** 参数启用的，请将共享设备添加到集群中第二个节点上的设备文件中。此功能默认为启用。

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

6.2. 配置一个 NFS 共享

使用以下流程为 NFS 服务故障转移配置一个 NFS 共享。

步骤

1. 在集群的两个节点上都创建 **/nfsshare** 目录。

```
# mkdir /nfsshare
```

2. 在集群的一个节点上执行以下步骤。

   a. 确定您在 配置带有 XFS 文件系统的 LVM 卷 中创建的的逻辑卷是否激活，然后将您在逻辑卷上创建的文件系统挂载到 **/nfsshare** 目录。

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

   b. 在 **/nfsshare** 目录上创建 **exports** 目录树。

```
[root@z1 ~]# mkdir -p /nfsshare/exports
[root@z1 ~]# mkdir -p /nfsshare/exports/export1
[root@z1 ~]# mkdir -p /nfsshare/exports/export2
```

   c. 将文件放到 **exports** 目录中，以便 NFS 客户端进行访问。在本例中，我们创建名为 **clientdatafile1** 和 **clientdatafile2** 的测试文件。

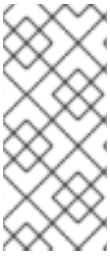
```
[root@z1 ~]# touch /nfsshare/exports/export1/clientdatafile1
[root@z1 ~]# touch /nfsshare/exports/export2/clientdatafile2
```

   d. 卸载文件系统，并停用 LVM 卷组。

```
[root@z1 ~]# umount /dev/my_vg/my_lv
[root@z1 ~]# vgchange -an my_vg
```

6.3. 为集群中的 NFS 服务器配置资源和资源组

使用以下流程为集群中的 NFS 服务器配置集群资源。



注意

如果您还没有为集群配置隔离设备，默认情况下资源不会启动。

如果发现您配置的资源没有运行，您可以运行 **pcs resource debug-start *resource*** 命令来测试资源配置。这会在集群控制之外启动服务。当配置的资源再次运行时，运行 **pcs resource cleanup *resource*** 以使集群已了解到更新。

步骤

以下步骤配置系统资源。为确保这些资源在同一节点上运行，它们都配置为资源组 **nfsgroup** 的一部分。资源将以您添加到组的顺序启动，并按照添加到组中的相反顺序停止。仅从集群的一个节点运行此步骤。

1. 创建名为 **my_lvm** 的 LVM 激活资源。由于资源组 **nfsgroup** 尚不存在，这个命令会创建资源组。

警告

不要配置多于一个的在主动/被动 HA 配置中使用相同 LVM 卷组的 **LVM-activate** 资源，因为这可能导致数据崩溃。另外，不要在主动/被动 HA 配置中将 **LVM-activate** 资源配置为克隆资源。

```
[root@z1 ~]# pcs resource create my_lvm ocf:heartbeat:LVM-activate vgname=my_vg
vg_access_mode=system_id --group nfsgroup
```

2. 检查集群的状态，以验证资源是否在运行。

```
root@z1 ~]#  pcs status
Cluster name: my_cluster
Last updated: Thu Jan  8 11:13:17 2015
Last change: Thu Jan  8 11:13:08 2015
Stack: corosync
Current DC: z2.example.com (2) - partition with quorum
Version: 1.1.12-a14efad
2 Nodes configured
3 Resources configured

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

Full list of resources:
 myapc  (stonith:fence_apc_snmp):       Started z1.example.com
 Resource Group: nfsgroup
     my_lvm     (ocf::heartbeat:LVM-activate):   Started z1.example.com

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

Daemon Status:
```

```
corosync: active/enabled
pacemaker: active/enabled
pcsd: active/enabled
```

3. 为集群配置 **Filesystem** 资源。
以下命令将名为 **nfsshare** 的 XFS **Filesystem** 资源配置为 **nfsgroup** 资源组的一部分。这个文件系统使用您在 配置具有 XFS 文件系统的 LVM 卷 中创建的 LVM 卷组和 XFS 文件系统，并将挂载到您在 配置 NFS 共享 中创建的 **/nfsshare** 目录上。

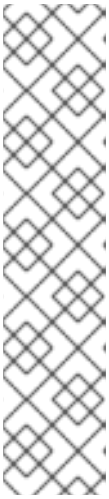
```
[root@z1 ~]# pcs resource create nfsshare Filesystem device=/dev/my_vg/my_lv
directory=/nfsshare fstype=xfs --group nfsgroup
```

您可以使用 **options=*options*** 参数将挂载选项指定为 **Filesystem** 资源的资源配置的一部分。运行 **pcs resource describe Filesystem** 命令以了解完整的配置选项。

4. 验证 **my_lvm** 和 **nfsshare** 资源是否正在运行。

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

5. 创建名为 **nfs-daemon** 的 **nfsserver** 资源，作为资源组 **nfsgroup** 的一部分。



注意

nfsserver 资源允许您指定 **nfs_shared_infodir** 参数，它是一个 NFS 服务器用于存储与 NFS 相关的有状态信息的目录。

建议将此属性设置为您在这个导出集合中创建的 **Filesystem** 资源的子目录。这样可确保 NFS 服务器将其有状态的信息存储在需要重新定位资源组时可供另一个节点使用的设备中。在这个示例中;

- **/nfsshare** 是由 **Filesystem** 资源管理的 shared-storage 目录
- **/nfsshare/exports/export1** 和 **/nfsshare/exports/export2** 是导出目录
- **/nfsshare/nfsinfo** 是 **nfsserver** 资源的共享目录

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

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

6. 添加 **exportfs** 资源以导出 **/nfsshare/exports** 目录。这些资源是 **nfsgroup** 资源组的一部分。这为 NFSv4 客户端构建了一个虚拟目录。NFSv3 客户端也可以访问这些导出。

注意

只有在您要为 NFSv4 客户端创建虚拟目录时才需要 **fsid=0** 选项。如需更多信息，请参阅 如何在 NFS 服务器的 /etc/exports 文件中配置 fsid 选项？。 //

```
[root@z1 ~]# pcs resource create nfs-root exportfs
clientspec=192.168.122.0/255.255.255.0 options=rw,sync,no_root_squash
directory=/nfsshare/exports fsid=0 --group nfsgroup

[root@z1 ~]# pcs resource create nfs-export1 exportfs
clientspec=192.168.122.0/255.255.255.0 options=rw,sync,no_root_squash
directory=/nfsshare/exports/export1 fsid=1 --group nfsgroup

[root@z1 ~]# pcs resource create nfs-export2 exportfs
clientspec=192.168.122.0/255.255.255.0 options=rw,sync,no_root_squash
directory=/nfsshare/exports/export2 fsid=2 --group nfsgroup
```

7. 添加 NFS 客户端用来访问 NFS 共享的浮动 IP 地址资源。这个资源是资源组 **nfsgroup** 的一部分。在本示例部署中，我们使用 192.168.122.200 作为浮动 IP 地址。

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

8. 添加 **nfsnotify** 资源，以便在整个 NFS 部署初始化后发送 NFSv3 重启通知。这个资源是资源组 **nfsgroup** 的一部分。

注意

为了正确处理 NFS 通知，浮动 IP 地址必须具有与其关联的主机名，在 NFS 服务器和 NFS 客户端中都一致。

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

9. 在创建资源和资源限制后，您可以检查集群的状态。请注意，所有资源都在同一个节点上运行。

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

6.4. 测试 NFS 资源配置

您可以按照以下步骤在高可用性集群中验证您的 NFS 资源配置。您应该可以使用 NFSv3 或 NFSv4 挂载导出的文件系统。

6.4.1. 测试 NFS 导出

1. 如果您在集群节点上运行 **firewalld** 守护进程，请确保在所有节点上都启用了 NFS 访问所需的端口。

2. 在与部署位于同一个网络中的、位于集群以外的一个节点中，通过挂载 NFS 共享来确定 NFS 共享。在本例中，我们使用 192.168.122.0/24 网络。

```
# showmount -e 192.168.122.200
Export list for 192.168.122.200:
/nfsshare/exports/export1 192.168.122.0/255.255.255.0
/nfsshare/exports         192.168.122.0/255.255.255.0
/nfsshare/exports/export2 192.168.122.0/255.255.255.0
```

3. 要验证您可以用 NFSv4 挂载 NFS 共享，将 NFS 共享挂载到客户端节点的目录中。挂载后，请确定导出目录的内容是可见的。测试后卸载共享。

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

4. 确定您可以用 NFSv3 挂载 NFS 共享。挂载后，验证测试文件 **clientdatafile1** 可见。和 NFSv4 不同，因为 NFSv3 不使用虚拟文件系统，所以您必须挂载一个特定的导出。测试后卸载共享。

```
# mkdir nfsshare
# mount -o "vers=3" 192.168.122.200:/nfsshare/exports/export2 nfsshare
# ls nfsshare
clientdatafile2
# umount nfsshare
```

6.4.2. 测试故障转移

1. 在集群外的节点上挂载 NFS 共享，并验证访问您在配置 NFS 共享 中创建的 **clientdatafile1** 文件。

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

2. 在集群的一个节点中，确定集群中的哪个节点正在运行 **nfsgroup**。在本例中，**nfsgroup** 在 **z1.example.com** 上运行。

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

```
    nfsshare   (ocf::heartbeat:Filesystem):    Started z1.example.com
    nfs-daemon (ocf::heartbeat:nfsserver):     Started z1.example.com
    nfs-root   (ocf::heartbeat:exportfs):      Started z1.example.com
    nfs-export1        (ocf::heartbeat:exportfs):      Started z1.example.com
    nfs-export2        (ocf::heartbeat:exportfs):      Started z1.example.com
    nfs_ip     (ocf::heartbeat:IPaddr2):       Started  z1.example.com
    nfs-notify (ocf::heartbeat:nfsnotify):     Started z1.example.com
...
```

3. 在集群的一个节点中，将运行 **nfsgroup** 的节点设置为待机模式。

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

4. 验证 **nfsgroup** 是否在其他集群节点上成功启动。

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

5. 在您挂载了 NFS 共享的集群之外的节点中，确认这个外部节点仍然可以访问 NFS 挂载中的测试文件。

```
# ls nfsshare
clientdatafile1
```

在故障转移的过程中，服务可能会在短暂时间内不可用，但可以在没有用户干预的情况下恢复。默认情况下，使用 NFSv4 的客户端可能最多需要 90 秒恢复该挂载。这个 90 秒代表服务器启动时观察到的 NFSv4 文件租期的宽限期。NFSv3 客户端应该在几秒钟内就可以恢复对该挂载的访问。

6. 从集群中的一个节点中，将最初运行 **nfsgroup** 的节点从待机模式中删除。

注意

从 **standby** 模式中删除节点不会自动导致资源恢复到该节点。这将依赖于 **resource-stickiness** 的值。有关 **resource-stickiness** 元属性的详情，请参考 配置资源以首选其当前节点。

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

第 7 章 集群中的 GFS2 文件系统

使用以下管理流程在红帽高可用性集群中配置 GFS2 文件系统。

7.1. 在集群中配置 GFS2 文件系统

您可以按照以下流程建立一个包含 GFS2 文件系统的 Pacemaker 集群。在这个示例中，您在双节点集群的三个逻辑卷上创建三个 GFS2 文件系统。

先决条件

- 在集群节点上安装并启动集群软件，并创建一个基本的双节点集群。
- 为集群配置隔离。

有关创建 Pacemaker 集群并为集群配置隔离的详情，请参考 使用 Pacemaker 创建红帽高可用性集群。

步骤

1. 在集群中的两个节点上，启用与您的系统架构对应的弹性存储存储库。例如，要为 x86_64 系统启用 Resilient Storage 仓库，您可以输入以下 **subscription-manager** 命令：

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

   请注意，弹性存储存储库是高可用性存储库的超集。如果启用弹性存储存储库，则不需要启用高可用性存储库。

2. 在集群的两个节点上安装 **lvm2-lockd**、**gfs2-utils** 和 **dlm** 软件包。要支持这些软件包，您必须订阅 AppStream 频道和 Resilient Storage 频道。

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

3. 在集群的两个节点上，将 **/etc/lvm/lvm.conf** 文件中的 **use_lvmlockd** 配置选项设置为 **use_lvmlockd=1**。

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

4. 将全局 Pacemaker 参数 **no-quorum-policy** 设置为 **freeze**。

注意

默认情况下，将 **no-quorum-policy** 的值设置为 **stop**，表示一旦仲裁丢失，剩余分区上的所有资源都会立即停止。通常，这个默认行为是最安全、最优的选项，但与大多数资源不同，GFS2 要求使用 quorum 才可以正常工作。当使用 GFS2 挂载的应用程序和 GFS2 挂载都丢失时，就无法正确停止 GFS2 挂载。任何在没有 quorum 的情况下停止这些资源的尝试都会失败，并最终会在每次 quorum 都丢失时保护整个集群。

要解决这个问题，请在使用 GFS2 时将 **no-quorum-policy** 设置为 **freeze**。这意味着，当 quorum 丢失时，剩余的分区将不会进行任何操作，直到 quorum 功能被恢复。

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

5. 设置 **dlm** 资源。这是在集群中配置 GFS2 文件系统所需的依赖软件包。这个示例创建 **dlm** 资源作为名为 **locking** 的资源组的一部分。

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

6. 克隆 **locking** 资源组，以便资源组可以在集群的两个节点上都活跃。

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

7. 建立 **lvmlockd** 资源，来作为 **locking** 资源组的一部分。

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

8. 检查集群的状态，以确保在集群的两个节点上启动了 **locking** 资源组。

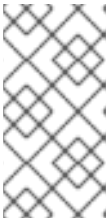
```
[root@z1 ~]# pcs status --full
Cluster name: my_cluster
[...]

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

Full list of resources:

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

9. 在集群的一个节点中，创建两个共享卷组。一个卷组将包含两个 GFS2 文件系统，另一个卷组将包含一个 GFS2 文件系统。



注意

如果您的 LVM 卷组包含一个或多个位于远程块存储（如 iSCSI 目标）上的物理卷，则红帽建议您确保服务在 Pacemaker 启动之前启动。有关为 Pacemaker 集群使用的远程物理卷配置启动顺序的详情，请参考 为不由 Pacemaker 管理的资源依赖项配置启动顺序。

以下命令在 **/dev/vdb** 上创建共享卷组 **shared_vg1**。

```
[root@z1 ~]# vgcreate --shared shared_vg1 /dev/vdb
  Physical volume "/dev/vdb" successfully created.
  Volume group "shared_vg1" successfully created
```

```
  VG shared_vg1 starting dlm lockspace
  Starting locking.  Waiting until locks are ready...
```

以下命令在 **/dev/vdc** 上创建共享卷组 **shared_vg2**。

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

10. 在集群中的第二个节点上：

    a. 如果设备文件的使用是通过 **lvm.conf** 文件中的 **use_devicesfile = 1** 参数启用的，请将共享设备添加到设备文件中，此功能默认启用。

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

    b. 为每个共享的卷组启动锁管理器。

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

11. 在集群的一个节点中，创建共享逻辑卷并使用 GFS2 文件系统格式化卷。每个挂载文件系统的节点都需要一个日志。确保为集群中的每个节点创建足够日志。锁表名称的格式为 *ClusterName:FSName*，其中 *ClusterName* 是为其创建 GFS2 文件系统的集群的名称，*FSName* 是文件系统名称，它对于所有集群上的 **lock_dlm** 文件系统必须是唯一的。

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

[root@z1 ~]# mkfs.gfs2 -j2 -p lock_dlm -t my_cluster:gfs2-demo1
/dev/shared_vg1/shared_lv1
[root@z1 ~]# mkfs.gfs2 -j2 -p lock_dlm -t my_cluster:gfs2-demo2
/dev/shared_vg1/shared_lv2
[root@z1 ~]# mkfs.gfs2 -j2 -p lock_dlm -t my_cluster:gfs2-demo3
/dev/shared_vg2/shared_lv1
```

12. 为每个逻辑卷创建一个 **LVM-activate** 资源，以便在所有节点中自动激活该逻辑卷。

    a. 为卷组 **shared_vg1** 中的逻辑卷 **shared_lv1** 创建名为 **sharedlv1** 的 **LVM-activate** 资源。此命令还会创建包含该资源的资源组 **shared_vg1**。在这个示例中，资源组的名称与包含逻辑卷的共享卷组的名称相同。

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

b. 为卷组 **shared_vg1** 中的逻辑卷 **shared_lv2** 创建名为 **sharedlv2** 的 **LVM-activate** 资源。此资源也是资源组 **shared_vg1** 的一部分。

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

c. 为卷组 **shared_vg2** 中的逻辑卷 **shared_lv1** 创建名为 **sharedlv3** 的 **LVM-activate** 资源。此命令还会创建包含该资源的资源组 **shared_vg2**。

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

13. 克隆两个新资源组。

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

14. 配置排序限制，以确保首先启动包含 **dlm** 和 **lvmlockd** 资源的 **locking** 资源组。

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

15. 配置共存限制，以确保 **vg1** 和 **vg2** 资源组在与 **locking** 资源组相同的节点上启动。

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

16. 在集群中的两个节点中，验证逻辑卷是否活跃。这可能会延迟几秒钟。

```
[root@z1 ~]# lvs
  LV        VG         Attr       LSize
  shared_lv1 shared_vg1  -wi-a----- 5.00g
  shared_lv2 shared_vg1  -wi-a----- 5.00g
  shared_lv1 shared_vg2  -wi-a----- 5.00g

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

17. 创建文件系统资源在所有节点中自动挂载每个 GFS2 文件系统。

您不应该将文件系统添加到 **/etc/fstab** 文件，因为它将作为 Pacemaker 集群资源进行管理。可以通过 **options=*options*** 将挂载选项指定为资源配置的一部分。运行 **pcs resource describe Filesystem** 命令显示完整的配置选项。

以下命令可创建文件系统资源。这些命令在包含该文件系统逻辑卷资源的资源组中添加每个资源。

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

验证步骤

1. 验证 GFS2 文件系统是否挂载到集群的两个节点中。

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

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

2. 检查集群的状态。

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

Full list of resources:

 smoke-apc      (stonith:fence_apc):    Started z1.example.com
 Clone Set: locking-clone [locking]
     Resource Group: locking:0
         dlm    (ocf::pacemaker:controld):      Started z2.example.com
         lvmlockd       (ocf::heartbeat:lvmlockd):      Started z2.example.com
     Resource Group: locking:1
         dlm    (ocf::pacemaker:controld):      Started z1.example.com
         lvmlockd       (ocf::heartbeat:lvmlockd):      Started z1.example.com
     Started: [ z1.example.com z2.example.com ]
 Clone Set: shared_vg1-clone [shared_vg1]
     Resource Group: shared_vg1:0
         sharedlv1      (ocf::heartbeat:LVM-activate):  Started z2.example.com
         sharedlv2      (ocf::heartbeat:LVM-activate):  Started z2.example.com
         sharedfs1      (ocf::heartbeat:Filesystem):    Started z2.example.com
         sharedfs2      (ocf::heartbeat:Filesystem):    Started z2.example.com
     Resource Group: shared_vg1:1
```

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

...
```

其他资源

- 配置 GFS2 文件系统
- 在 Microsoft Azure 上配置红帽高可用性集群
- 在 AWS 上配置红帽高可用性集群
- 在 Google Cloud Platform 上配置红帽高可用性集群

7.2. 在集群中配置加密的 GFS2 文件系统

您可以按照以下流程创建一个包含 LUKS 加密的 GFS2 文件系统的 Pacemaker 集群。在这个示例中，您在逻辑卷上创建一个 GFS2 文件系统，并加密文件系统。使用 **crypt** 资源代理支持加密 GFS2 文件系统，该代理支持 LUKS 加密。

此流程有三个部分：

- 在 Pacemaker 集群中配置共享逻辑卷
- 加密逻辑卷并创建一个 **crypt** 资源
- 使用 GFS2 文件系统格式化加密的逻辑卷并为集群创建文件系统资源

7.2.1. 在 Pacemaker 集群中配置共享逻辑卷

先决条件

- 在两个集群节点上安装并启动集群软件，并创建一个基本的双节点集群。
- 为集群配置隔离。

有关创建 Pacemaker 集群并为集群配置隔离的详情，请参考 使用 Pacemaker 创建红帽高可用性集群。

步骤

1. 在集群中的两个节点上，启用与您的系统架构对应的弹性存储存储库。例如，要为 x86_64 系统启用 Resilient Storage 仓库，您可以输入以下 **subscription-manager** 命令：

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

请注意，弹性存储存储库是高可用性存储库的超集。如果启用弹性存储存储库，则不需要启用高可用性存储库。

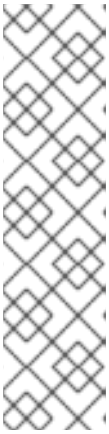
2. 在集群的两个节点上安装 **lvm2-lockd**、**gfs2-utils** 和 **dlm** 软件包。要支持这些软件包，您必须订阅 AppStream 频道和 Resilient Storage 频道。

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

3. 在集群的两个节点上，将 **/etc/lvm/lvm.conf** 文件中的 **use_lvmlockd** 配置选项设置为 **use_lvmlockd=1**。

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

4. 将全局 Pacemaker 参数 **no-quorum-policy** 设置为 **freeze**。



注意

默认情况下，**no-quorum-policy** 的值被设为 **stop**，这表示仲裁丢失时，剩余分区上的所有资源将立即停止。通常，这个默认行为是最安全、最优的选项，但与大多数资源不同，GFS2 要求使用 quorum 才可以正常工作。当使用 GFS2 挂载的应用程序和 GFS2 挂载都丢失时，就无法正确停止 GFS2 挂载。任何在没有 quorum 的情况下停止这些资源的尝试都会失败，并最终会在每次 quorum 都丢失时保护整个集群。

要解决这个问题，请在使用 GFS2 时将 **no-quorum-policy** 设置为 **freeze**。这意味着，当 quorum 丢失时，剩余的分区将不会进行任何操作，直到 quorum 功能被恢复。

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

5. 设置 **dlm** 资源。这是在集群中配置 GFS2 文件系统所需的依赖软件包。这个示例创建 **dlm** 资源作为名为 **locking** 的资源组的一部分。

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

6. 克隆 **locking** 资源组，以便资源组可以在集群的两个节点上都活跃。

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

7. 设置 **lvmlockd** 资源作为组 **locking** 的一部分。

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

8. 检查集群的状态，以确保在集群的两个节点上启动了 **locking** 资源组。

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

```
Cluster name: my_cluster
[...]

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

Full list of resources:

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

9. 在集群的一个节点中创建一个共享卷组。

注意

如果您的 LVM 卷组包含一个或多个位于远程块存储（如 iSCSI 目标）上的物理卷，则红帽建议您确保服务在 Pacemaker 启动之前启动。有关为 Pacemaker 集群使用的远程物理卷配置启动顺序的详情，请参考 为不由 Pacemaker 管理的资源依赖项配置启动顺序。

以下命令在 **/dev/sda1** 上创建共享卷组 **shared_vg1**。

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

10. 在集群中的第二个节点上：

    a. 如果设备文件的使用是通过 **lvm.conf** 文件中的 **use_devicesfile = 1** 参数启用的，请将共享设备添加到集群中第二个节点上的设备文件中。此功能默认为启用。

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

    b. 为共享卷组启动锁管理器。

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

11. 在集群的一个节点中创建共享逻辑卷。

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

12. 为逻辑卷创建一个 **LVM-activate** 资源，以便在所有节点中自动激活逻辑卷。

以下命令为卷组 **shared_vg1** 中的逻辑卷 **shared_lv1** 创建名为 **sharedlv1** 的 **LVM-activate** 资源。此命令还会创建包含该资源的资源组 **shared_vg1**。在这个示例中，资源组的名称与包含逻辑卷的共享卷组的名称相同。

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

13. 克隆新资源组。

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

14. 配置排序限制，以确保首先启动包含 **dlm** 和 **lvmlockd** 资源的 **locking** 资源组。

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

15. 配置 colocation 约束，以确保 **vg1** 和 **vg2** 资源组在与 **locking** 资源组相同的节点上启动。

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

验证步骤

在集群中的两个节点中，验证逻辑卷是否活跃。这可能会延迟几秒钟。

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

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

7.2.2. 加密逻辑卷并创建 crypt 资源

先决条件

- 您已在 Pacemaker 集群中配置了共享逻辑卷。

步骤

1. 在集群的一个节点中，创建一个包含 crypt 密钥的新文件，并在文件中设置权限，使其只能被 root 读取。

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

2. 创建 crypt 密钥。

```
[root@z1 ~]# dd if=/dev/urandom bs=4K count=1 of=/etc/crypt_keyfile
1+0 records in
```

```
1+0 records out
4096 bytes (4.1 kB, 4.0 KiB) copied, 0.000306202 s, 13.4 MB/s
[root@z1 ~]# scp /etc/crypt_keyfile root@z2.example.com:/etc/
```

3. 使用 **-p** 参数将 crypt keyfile 分发到集群中的其他节点，以保留您设置的权限。

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

4. 在要配置加密的 GFS2 文件系统的 LVM 卷中创建加密设备。

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

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

5. 创建 crypt 资源，作为 **shared_vg1** 卷组的一部分。

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

验证步骤

确保 crypt 资源已创建了 crypt 设备，本例中为 **/dev/mapper/luks_lv1**。

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

7.2.3. 使用 GFS2 文件系统格式化加密的逻辑卷，并为集群创建文件系统资源

先决条件

- 您已加密逻辑卷并创建 crypt 资源。

步骤

1. 在集群的一个节点中，使用 GFS2 文件系统格式化卷。每个挂载文件系统的节点都需要一个日志。确保为集群中的每个节点创建足够日志。锁表名称的格式为 *ClusterName:FSName*，其中 *ClusterName* 是为其创建 GFS2 文件系统的集群的名称，*FSName* 是文件系统名称，它对于所有集群上的 **lock_dlm** 文件系统必须是唯一的。

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

```
Creating quota file: Done
Writing superblock and syncing: Done
Device:                 /dev/mapper/luks_lv1
Block size:             4096
Device size:            4.98 GB (1306624 blocks)
Filesystem size:        4.98 GB (1306622 blocks)
Journals:               3
Journal size:           16MB
Resource groups:        23
Locking protocol:       "lock_dlm"
Lock table:             "my_cluster:gfs2-demo1"
UUID:                   de263f7b-0f12-4d02-bbb2-56642fade293
```

2. 创建文件系统资源在所有节点中自动挂载 GFS2 文件系统。
不要将该文件系统添加到 **/etc/fstab** 文件中，因为它将作为 Pacemaker 集群资源进行管理。可以通过 **options=*options*** 将挂载选项指定为资源配置的一部分。运行 **pcs resource describe Filesystem** 命令以了解完整的配置选项。

   以下命令创建文件系统资源。这个命令在包含该文件系统逻辑卷资源的资源组中添加资源。

```
[root@z1 ~]# pcs resource create sharedfs1 --group shared_vg1
ocf:heartbeat:Filesystem device="/dev/mapper/luks_lv1" directory="/mnt/gfs1"
fstype="gfs2" options=noatime op monitor interval=10s on-fail=fence
```

验证步骤

1. 验证 GFS2 文件系统是否挂载到集群的两个节点上。

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

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

2. 检查集群的状态。

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

Full list of resources:

  smoke-apc      (stonith:fence_apc):    Started z1.example.com
  Clone Set: locking-clone [locking]
    Resource Group: locking:0
      dlm    (ocf::pacemaker:controld):      Started z2.example.com
      lvmlockd       (ocf::heartbeat:lvmlockd):      Started z2.example.com
    Resource Group: locking:1
      dlm    (ocf::pacemaker:controld):      Started z1.example.com
      lvmlockd       (ocf::heartbeat:lvmlockd):      Started z1.example.com
    Started: [ z1.example.com z2.example.com ]
  Clone Set: shared_vg1-clone [shared_vg1]
    Resource Group: shared_vg1:0
        sharedlv1      (ocf::heartbeat:LVM-activate):  Started z2.example.com
```

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

其他资源

- 配置 GFS2 文件系统

第 8 章 在红帽高可用性集群中配置主动/主动 SAMBA 服务器

红帽高可用性附加组件为在主动/主动集群配置中配置 Samba 提供支持。在以下示例中，您要在双节点 RHEL 集群上配置一个主动/主动 Samba 服务器。

有关 Samba 的支持政策的详情，请参考红帽客户门户网站上的 RHEL 高可用性的支持策略 - ctdb 通用策略 和 RHEL 弹性存储的支持策略 - 通过其他协议导出 gfs2 内容 。

要在主动/主动集群中配置 Samba：

1. 配置 GFS2 文件系统及其关联的集群资源。
2. 在集群节点上配置 Samba。
3. 配置 Samba 集群资源。
4. 测试您配置的 Samba 服务器。

8.1. 为高可用性集群中的 SAMBA 服务配置 GFS2 文件系统

在 Pacemaker 集群中配置主动/主动 Samba 服务前，请为集群配置 GFS2 文件系统。

先决条件

- 为每个节点配置了隔离功能的双节点红帽高可用性集群
- 每个集群节点可用的共享存储
- 每个集群节点的 AppStream 渠道和弹性存储渠道的订阅

有关创建 Pacemaker 集群并为集群配置隔离的详情，请参考 使用 Pacemaker 创建红帽高可用性集群。

步骤

1. 在集群的两个节点上都执行以下初始设置步骤。
   a. 为与您的系统架构对应的弹性存储启用存储库。例如，要为 x86_64 系统启用弹性存储存储库，请输入以下 **subscription-manager** 命令：

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

   弹性存储存储库是高可用性存储库的超集。如果启用了弹性存储存储库，则不需要启用高可用性存储库。

   b. 安装 **lvm2-lockd**、**gfs2-utils** 和 **dlm** 软件包。

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

   c. 将 **/etc/lvm/lvm.conf** 文件中的 **use_lvmlockd** 配置选项设为 **use_lvmlockd=1**。

   ```
   ...

   use_lvmlockd = 1

   ...
   ```

2. 在集群中的一个节点上，将全局 Pacemaker 参数 **no-quorum-policy** 设为 **freeze**。

注意

默认情况下，将 **no-quorum-policy** 的值设置为 **stop**，表示一旦仲裁丢失，剩余分区上的所有资源都会立即停止。通常，这个默认行为是最安全、最优的选项，但与大多数资源不同，GFS2 要求使用 quorum 才可以正常工作。当使用 GFS2 挂载的应用程序和 GFS2 挂载都丢失时，就无法正确停止 GFS2 挂载。任何在没有 quorum 的情况下停止这些资源的尝试都会失败，并最终会在每次 quorum 都丢失时保护整个集群。

要解决这个问题，请在使用 GFS2 时将 **no-quorum-policy** 设置为 **freeze**。这意味着，当 quorum 丢失时，剩余的分区将不会进行任何操作，直到 quorum 功能被恢复。

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

3. 设置 **dlm** 资源。这是在集群中配置 GFS2 文件系统所需的依赖软件包。这个示例创建 **dlm** 资源作为名为 **locking** 的资源组的一部分。如果您之前没有为集群配置隔离，则此步骤会失败，**pcs status** 命令显示资源失败信息。

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

4. 克隆 **locking** 资源组，以便资源组可以在集群的两个节点上都活跃。

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

5. 建立 **lvmlockd** 资源，来作为 **locking** 资源组的一部分。

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

6. 在共享设备 **/dev/vdb** 上创建物理卷和共享卷组。这个示例创建共享卷组 **csmb_vg**。

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

7. 在集群中的第二个节点上：

8. 如果设备文件的使用是通过 **lvm.conf** 文件中的 **use_devicesfile = 1** 参数启用的，请将共享设备添加到集群中第二个节点上的设备文件中。此功能默认为启用。

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

   a. 为共享卷组启动锁管理器。

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

9. 在集群中的一个节点上，创建一个逻辑卷并使用 GFS2 文件系统格式化卷，该文件系统将被 CTDB 专门用于内部锁定。即使部署导出多个共享，集群中只需要一个这样的文件系统。
当使用 **mkfs.gfs2** 命令的 **-t** 选项指定锁表名称时，请确保您指定的 *clustername:filesystemname* 的第一个组件与集群名称匹配。在本例中，集群名称为 **my_cluster**。

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

10. 为每个通过 Samba 共享的 GFS2 文件系统创建一个逻辑卷，并使用 GFS2 文件系统格式化卷。这个示例创建了一个 GFS2 文件系统和 Samba 共享，但您可以创建多个文件系统和共享。

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

11. 建立 **LVM_Activate** 资源，以确保所需的共享卷被激活。这个示例创建 **LVM_Activate** 资源，来作为资源组 **shared_vg** 的一部分，然后克隆该资源组，使其在集群中的所有节点上运行。
将资源创建为禁用，以便它们在已配置的必要的顺序约束前不自动启动。

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

12. 配置排序约束，以在 **shared_vg** 资源组的成员之前启动 **locking** 资源组的所有成员。

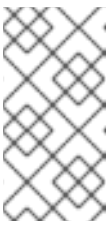
```
[root@z1 ~]# pcs constraint order start locking-clone then shared_vg-clone
Adding locking-clone shared_vg-clone (kind: Mandatory) (Options: first-action=start then-action=start)
```

13. 启用 **LVM-activate** 资源。

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

14. 在集群的一个节点上，执行以下步骤来创建您需要的 **Filesystem** 资源。

    a. 使用之前在 LVM 卷上配置的 GFS2 文件系统将 **Filesystem** 资源创建为克隆的资源。这将配置 Pacemaker，以挂载和管理文件系统。



注意

您不应该将文件系统添加到 **/etc/fstab** 文件，因为它将作为 Pacemaker 集群资源进行管理。您可以使用 **options=*options*** 将挂载选项指定为资源配置的一部分。运行 **pcs resource describe Filesystem** 命令显示完整的配置选项。

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

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

b. 配置排序约束，以确保 Pacemaker 在共享卷组 **shared_vg** 启动后挂载文件系统。

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

8.2. 在高可用性集群中配置 SAMBA

要在 Pacemaker 集群中配置 Samba 服务，请在集群的所有节点上配置该服务。

先决条件

- 使用 GFS2 文件系统配置的双节点红帽高可用性集群，如在 为高可用性集群中的 Samba 服务配置一个 GFS2 文件系统 中所述。
- 在 GFS2 文件系统上创建的用于 Samba 共享的公共目录。在本例中，目录为 **/srv/samba/share1**。
- 可用于访问此集群导出的 Samba 共享的公共虚拟 IP 地址。

步骤

1. 在集群的两个节点上，配置 Samba 服务，并建立共享定义：

   a. 安装 Samba 和 ctdb 软件包。

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

   b. 确保 **ctdb**、**smb**、**nmb** 和 **winbind** 服务没有运行，且不会在引导时启动。

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

   c. 在 **/etc/samba/smb.conf** 文件中，配置 Samba 服务，并建立共享定义，如下例具有一个共享的独立服务器所示：

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

   d. 验证 **/etc/samba/smb.conf** 文件。

```
# testparm
```

2. 在集群的两个节点上配置 CTDB：

   a. 创建 **/etc/ctdb/nodes** 文件并添加集群节点的 IP 地址，如示例节点文件中所示。

   ```
   192.0.2.11
   192.0.2.12
   ```

   b. 创建 **/etc/ctdb/public_addresses** 文件，并将集群公共接口的 IP 地址和网络设备名称添加到该文件中。在 **public_addresses** 文件中分配 IP 地址时，请确保这些地址没有被使用，并且这些地址可从预期的客户端路由。**/etc/ctdb/public_addresses** 文件的每个条目中的第二个字段是集群机器上用于相应公共地址的接口。在这个 **public_addresses** 示例文件中，接口 **enp1s0** 用于所有公共地址。

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

   集群的公共接口是客户端用于从其网络访问 Samba 的公共接口。出于负载均衡目的，将集群的每个公共 IP 地址的 A 记录添加到您的 DNS 区域。这些记录的每一个都必须解析为同样的主机名。客户端使用主机名访问 Samba，DNS 将客户端分发到集群的不同节点。

   c. 如果您正在运行 **firewalld** 服务，启用 **ctdb** 和 **samba** 服务所需的端口。

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

3. 对集群的一个节点更新 SELinux 上下文：

   a. 更新 GFS2 共享的 SELinux 上下文。

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

   b. 更新 Samba 中共享的目录的 SELinux 上下文。

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

其他资源

- 有关将 Samba 配置为独立服务器的详情，请参考 配置和使用网络文件服务 中的 使用 Samba 作为服务器 章节。
- 在 BIND 主服务器上建立转发区域。

8.3. 配置 SAMBA 集群资源

在双节点高可用性集群的两个节点上配置 Samba 服务后，请为集群配置 Samba 集群资源。

先决条件

- 使用 GFS2 文件系统配置的双节点红帽高可用性集群，如在 为高可用性集群中的 Samba 服务配置一个 GFS2 文件系统 中所述。
- Samba 服务已在两个集群节点上配置了，如 在高可用性集群中配置 Samba 中所述。

步骤

1. 在集群的一个节点上配置 Samba 集群资源：

   a. 在组 **samba-group** 中创建 CTDB 资源。CTDB 资源代理使用 **pcs** 命令指定的**ctdb_*** 选项来创建 CTDB 配置文件。将资源创建为禁用，以便其在配置必要的顺序约束前不自动启动。

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

   b. 克隆 **samba-group** 资源组。

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

   c. 创建排序约束，以确保所有 **Filesystem** 资源都在 **samba-group** 中的资源之前运行。

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

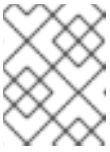
   d. 在资源组 **samba-group** 中创建**samba** 资源。这会根据它们添加的顺序在 CTDB 和 Samba 之间创建一个隐式排序约束。

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

   e. 启用 **ctdb** 和 **samba** 资源。

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

   f. 检查是否所有服务都已成功启动。

   注意

   可能需要几分钟 CTDB 才能启动 Samba、导出共享并稳定下来。如果在此过程完成前检查集群状态，您可能会看到 **samba** 服务还没有运行。

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

   ...

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

```
 * Clone Set: ctdb_fs-clone [ctdb_fs]:
* Started: [ z1.example.com z2.example.com ]
 * Clone Set: csmb_fs1-clone [csmb_fs1]:
* Started: [ z1.example.com z2.example.com ]
  * Clone Set: samba-group-clone [samba-group]:
* Started: [ z1.example.com z2.example.com ]
```

2. 在集群的两个节点上，为 test 共享目录添加本地用户。

   a. 添加用户。

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

   b. 设置用户的密码。

   ```
   # passwd example_user
   ```

   c. 为用户设置 SMB 密码。

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

   d. 激活 Samba 数据库中的用户。

   ```
   # smbpasswd -e example_user
   ```

   e. 为 Samba 用户更新 GFS2 共享的文件所有者和权限。

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

8.4. 验证集群的 SAMBA 配置

如果您的集群 Samba 配置成功，您可以挂载 Samba 共享。挂载共享后，如果导出 Samba 共享的集群节点不可用，您可以测试 Samba 恢复。

步骤

1. 在可以访问集群节点上 **/etc/ctdb/public_addresses** 文件中配置的一个或多个公共 IP 地址的系统上，使用其中一个公共 IP 地址挂载 Samba 共享。

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

2. 验证文件系统是否已挂载。

   ```
   [root@testmount ~]# mount | grep /mnt/sambashare
   //192.0.2.201/public on /mnt/sambashare type cifs
   (rw,relatime,vers=1.0,cache=strict,username=example_user,domain=LINUXSERVER,uid=0,nof
   ```

```
orceuid,gid=0,noforcegid,addr=192.0.2.201,unix,posixpaths,serverino,mapposix,acl,rsize=1048576,wsize=65536,echo_interval=60,actimeo=1,user=example_user)
```

3. 验证您是否可以在挂载的文件系统上创建文件。

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

4. 确定哪个集群节点正在导出 Samba 共享：

   a. 在每个集群节点上，显示分配给 **public_addresses** 文件中指定的接口的 IP 地址。以下命令显示分配给每个节点上 **enp1s0** 接口的 IPv4 地址。

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

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

   b. 在 **ip** 命令的输出中，找到在挂载共享时使用 **mount** 命令指定的 IP 地址的节点。在本例中，mount 命令中指定的 IP 地址为 192.0.2.201。**ip** 命令的输出显示 IP 地址 192.0.2.201 被分配给 **z1.example.com**。

5. 将导出 Samba 共享的节点置于 **standby** 模式，这会导致节点无法托管任何集群资源。

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

6. 在挂载文件系统的系统上，确定您仍可在文件系统上创建文件。

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

7. 删除您创建的文件，以验证文件系统是否已成功挂载。如果您不再需要挂载文件系统，此时卸载文件系统。

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

8. 从集群的一个节点，将集群服务恢复到您之前置于待机模式的节点。这不一定将该服务转换到第一个节点。

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

第 9 章 PCSD WEB UI 入门

pcsd Web UI 是一个图形用户界面，用于创建和配置 Pacemaker/Corosync 集群。

9.1. 设置 PCSD WEB UI

按照以下流程设置您的系统以使用 **pcsd** Web UI 来配置集群。

先决条件

- Pacemaker 配置工具已安装。
- 已为集群配置设置了您的系统。

有关安装集群软件并为集群配置设置您的系统的说明，请参阅 安装集群软件。

步骤

1. 在任何系统上，打开浏览器到以下 URL，指定集群的一个节点（请注意，这会使用 **https** 协议）。这将打开 **pcsd** Web UI 登录界面。

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

2. 以用户 **hacluster** 身份登录。这将打开 **Clusters** 页面。

9.2. 配置高可用性 PCSD WEB UI

使用 **pcsd** Web UI 时，您可以连接到集群的一个节点来显示集群管理页面。如果您要连接的节点停机或不可用，可以在浏览器使用指向集群中不同节点的 URL 来重新连接到集群。但是，有可能将 **pcsd** Web UI 本身配置为具有高可用性，这样您就可以继续管理集群而无需输入新的 URL。

步骤

要为 **pcsd** Web UI 配置高可用性，请执行以下步骤：

1. 通过将 **/etc/sysconfig/pcsd** 配置文件中的 **PCSD_SSL_CERT_SYNC_ENABLED** 设置为 **true**，确保 **pcsd** 证书在集群节点间同步。启用证书同步会导致 **pcsd** 同步集群设置和节点添加命令的证书。默认情况下，**PCSD_SSL_CERT_SYNC_ENABLED** 被设置为 **false**。

2. 创建一个 **IPaddr2** 集群资源，它是您将用来连接到 **pcsd** Web UI 的浮动 IP 地址。IP 地址不能是一个已经与物理节点关联的 IP 地址。如果没有指定 **IPaddr2** 资源的 NIC 设备，浮动 IP 必须与节点静态分配的 IP 地址之一位于同一个网络中，否则无法正确检测到分配浮动 IP 地址的 NIC 设备。

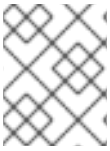
3. 为使用 **pcsd** 创建自定义 SSL 证书，并确保它们对连接到 **pcsd** Web UI 的节点的地址有效。

   a. 要创建自定义 SSL 证书，您可以使用通配符证书，或者使用主题替代名称证书扩展。有关红帽认证系统的详情，请查看 红帽认证系统管理指南。

   b. 使用 **pcs pcsd certkey** 命令来为 **pcsd** 安装自定义证书。

   c. 使用 **pcs pcsd sync-certificates** 命令将 **pcsd** 证书同步到集群中的所有节点上。

4. 使用您配置为集群资源的浮动 IP 地址连接到 **pcsd** Web UI。



注意

即使您将 **pcsd** Web UI 配置为高可用性，当您要连接的节点停机时，也会要求您再次登录。

第 10 章 在 RED HAT HIGH AVAILABILITY 集群中配置隔离功能

不响应的节点可能仍然在访问数据。确定您的数据安全的唯一方法是使用 STONITH 保护节点。STONITH 是"Shoot The Other Node In The Head"的缩写，它保护您的数据不受有问题的节点或并发访问的影响。使用 STONITH 可以确保，在允许从另一个节点访问数据前确定节点真正离线。

当无法停止集群的服务时，STONITH 也会有意义。在这种情况下，集群使用 STONITH 来强制整个节点离线，从而使在其他位置可以安全地启动该服务。

有关隔离的一般信息及其在红帽高可用性集群中的重要性，请参阅 红帽高可用性集群中的隔离。

您可以在 Pacemaker 集群中为集群的节点配置隔离设备，从而在 Pacemaker 集群中实施 STONITH。

10.1. 显示可用的隔离代理及其选项

以下命令可用于查看可用的隔离代理和特定隔离代理的可用选项。

注意

您的系统硬件决定了用于集群的隔离设备的类型。有关支持的平台和架构以及不同的隔离设备的详情，请参考 RHEL 高可用性集群中的支持策略 文章的 集群平台和架构 部分。

运行以下命令列出所有可用的隔离代理。当您指定过滤器时，这个命令只会显示与过滤器匹配的隔离代理。

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

运行以下命令显示指定隔离代理的选项。

```
pcs stonith describe [stonith_agent]
```

例如：以下命令显示 APC 通过 telnet/SSH 的隔离代理的选项。

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

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

警告

对于提供 **方法** 选项的隔离代理，但 **fence_sbd** 代理除外，它不被支持，不应指定 **循环**，因为它可能会导致数据崩溃。即使是 **fence_sbd**，但不应指定方法，而是使用默认值。

10.2. 创建隔离设备

创建隔离设备的命令格式如下。有关可用隔离设备创建选项的列表，请参阅 **pcs stonith -h** 显示。

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

以下命令为单一节点创建一个隔离设备。

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

有些隔离设备只能隔离一个节点，其他设备则可能隔离多个节点。您创建隔离设备时指定的参数取决于您的隔离设备的支持和要求。

- 有些隔离设备可自动决定它们可以隔离哪些节点。
- 您可以在创建隔离设备时使用 **pcmk_host_list** 参数，以指定由该隔离设备控制的所有机器。
- 有些隔离设备需要主机名与隔离设备可识别的规格映射。在创建隔离设备时，您可以使用 **pcmk_host_map** 参数来映射主机名。

有关 **pcmk_host_list** 和 **pcmk_host_map** 参数的详情，请参考 隔离设备的常规属性。

配置隔离设备后，您必须测试该设备以保证其可以正常工作。有关测试隔离设备的详情，请参考 测试隔离设备。

10.3. 隔离设备的常规属性

您可以为隔离设备设置很多常规属性，以及决定隔离行为的各种集群属性。

任何集群节点都可以使用任何隔离设备隔离保护其它集群节点，无论隔离资源是启动还是停止。资源是否启动只控制设备的重复监控，而不控制是否使用资源，但以下情况除外：

- 您可以通过运行 **pcs stonith disable *stonith_id*** 命令来禁用隔离设备。这将阻止任何节点使用该设备。

- 要防止特定节点使用隔离设备，您可以为隔离资源配置位置约束，使用 **pcs constraint location ... avoids** 命令。
- 配置 **stonith-enabled=false** 将完全禁用隔离。但请注意，红帽不支持隔离功能被禁用的集群，因为它不适用于生产环境。

下表描述了您可以为隔离设备设置的一般属性。

表 10.1. 隔离设备的常规属性

项	类型	默认值	描述
pcmk_host_map	字符串		用于不支持主机名的设备的主机名到端口号的映射。例如：**node1:1;node2:2,3** 告知集群将端口 1 用于 node1，端口 2 和 3 用于 node2。**pcmk_host_map** 属性支持在值前面使用反斜杠的 **pcmk_host_map** 值中的特殊字符。例如，您可以指定 **pcmk_host_map="node3:plug\ 1"**，来在主机别名中包含一个空格。
pcmk_host_list	字符串		此设备控制的机器列表（可选，除非 **pcmk_host_check=static-list**）。
pcmk_host_check	字符串	* **static-list**（如果设置了 **pcmk_host_list** 或 **pcmk_host_map**） * 否则，**dynamic-list**（如果隔离设备支持 **list** 操作） * 否则，**status**（如果隔离设备支持 **status** 操作） * 否则，**none**.	如何确定被设备控制的机器。允许的值：**dynamic-list**（查询设备）、**static-list**（检查 **pcmk_host_list** 属性）、none（假设每个设备都可以隔离每台机器）

下表总结了您可以为隔离设备设置的其他属性。请注意，这些属性仅适用于高级使用。

表 10.2. 隔离设备的高级属性

项	类型	默认值	描述
pcmk_host_argument	字符串	port	提供端口的一个替代参数。有些设备不支持标准端口参数，或者可能会提供额外的端口。使用这个参数指定一个替代的、针对于具体设备的参数，它代表要被隔离的机器。值 **none** 可用于告之集群不要提供任何额外的参数。

项	类型	默认值	描述
pcmk_reboot_action	字符串	reboot	要运行的一个替代命令，而不是 **reboot**。有些设备不支持标准命令或者可能需要提供额外的命令。使用这个选项指定可执行 reboot 操作的替代的、特定于具体设备的命令。
pcmk_reboot_timeout	time	60s	指定用于重新启动操作的替代超时，而不是 **stonith-timeout**。和一般的设备相比，有些设备需要更长或更短的时间完成。使用此选项指定替代的、重启操作使用的、特定于设备的超时时间。
pcmk_reboot_retries	整数	2	在超时时间内重试 **reboot** 命令的次数上限。有些设备不支持多个连接。如果设备忙碌了处理另一个任务，操作可能会失败，因此如果还有剩余时间，Pacemaker 会自动重试操作。使用这个选项更改 Pacemaker 在放弃前重试重启动作的次数。
pcmk_off_action	字符串	off	要运行的一个替代命令，而不是 **off**。有些设备不支持标准命令或者可能需要提供额外的命令。使用这个选项指定可执行 off 操作的替代的、特定于具体设备的命令。
pcmk_off_timeout	time	60s	指定用于 off 操作的替代超时，而不是 **stonith-timeout**。和一般的设备相比，有些设备需要更长或更短的时间完成。使用此选项指定替代的、off 操作使用的、特定于设备的超时时间。
pcmk_off_retries	整数	2	在超时时间内重试 off 命令的次数上限。有些设备不支持多个连接。如果设备忙碌了处理另一个任务，操作可能会失败，因此如果还有剩余时间，Pacemaker 会自动重试操作。使用这个选项更改 Pacemaker 在放弃前重试操作的次数。
pcmk_list_action	字符串	list	要运行的一个替代命令，而不是 **list**。有些设备不支持标准命令或者可能需要提供额外的命令。使用这个选项指定可执行 list 操作的替代的、特定于具体设备的命令。
pcmk_list_timeout	time	60s	指定 list 操作使用的特代的超时时间。和一般的设备相比，有些设备需要更长或更短的时间完成。使用此选项指定替代的、list 操作使用的、特定于设备的超时时间。

项	类型	默认值	描述
pcmk_list_retries	整数	2	在超时时间内重试 **list** 命令的次数上限。有些设备不支持多个连接。如果设备忙碌了处理另一个任务，操作可能会失败，因此如果还有剩余时间，Pacemaker 会自动重试操作。使用这个选项更改 Pacemaker 在放弃前 list 操作的次数。
pcmk_monitor_action	字符串	monitor	要运行的一个替代命令，而不是 **monitor**。有些设备不支持标准命令或者可能需要提供额外的命令。使用这个选项指定可执行 monitor 操作的替代的、特定于具体设备的命令。
pcmk_monitor_timeout	time	60s	指定用于 monitor 操作的替代超时，而不是 **stonith-timeout**。和一般的设备相比，有些设备需要更长或更短的时间完成。使用此选项指定替代的、monitor 操作使用的、特定于设备的超时时间。
pcmk_monitor_retries	整数	2	在超时时间内重试 **monitor** 命令的次数上限。有些设备不支持多个连接。如果设备忙碌了处理另一个任务，操作可能会失败，因此如果还有剩余时间，Pacemaker 会自动重试操作。使用这个选项更改 Pacemaker 在放弃前 monitor 操作的次数。
pcmk_status_action	字符串	status	要运行的一个替代命令，而不是 **status**。有些设备不支持标准命令或者可能需要提供额外的命令。使用这个选项指定可执行 status 操作的替代的、特定于具体设备的命令。
pcmk_status_timeout	time	60s	指定用于 status 操作的替代超时，而不是 **stonith-timeout**。和一般的设备相比，有些设备需要更长或更短的时间完成。使用此选项指定替代的、status 操作使用的、特定于设备的超时时间。
pcmk_status_retries	整数	2	在超时时间内重试 status 命令的次数上限。有些设备不支持多个连接。如果设备忙碌了处理另一个任务，操作可能会失败，因此如果还有剩余时间，Pacemaker 会自动重试操作。使用这个选项更改 Pacemaker 在放弃前 status 操作的次数。

项	类型	默认值	描述
pcmk_delay_base	字符串	0s	为隔离操作启用基本延迟，并指定基本延迟值。您可以使用 **pcmk_delay_base** 参数为不同的节点指定不同的值。有关隔离延迟参数及其相互作用的一般信息，请查看 隔离延迟。
pcmk_delay_max	time	0s	为隔离操作启用一个随机延迟，并指定最大延迟，即组合基本延迟和随机延迟的最大值。例如，如果基本延迟为 3，并且 **pcmk_delay_max** 为 10，则随机延迟将在 3 和 10 之间。有关隔离延迟参数及其相互作用的一般信息，请查看 隔离延迟。
pcmk_action_limit	整数	1	在这个设备上可并行执行的最大操作数量。需要首先配置集群属性 **concurrent-fencing=true**（这是默认值）。值为 -1 代表没有限制。
pcmk_on_action	字符串	on	仅用于高级使用：替代 **on** 的一个要运行的命令。有些设备不支持标准命令或者可能需要提供额外的命令。使用它来指定一个执行 **on** 操作的替代的、特定于设备的命令。
pcmk_on_timeout	time	60s	仅用于高级使用：指定用于 **on** 操作的、替代 **stonith-timeout** 的超时设置。和一般的设备相比，有些设备需要更长或更短的时间完成。使用它来为 **on** 操作指定一个替代的、特定于设备的超时。
pcmk_on_retries	整数	2	仅用于高级使用：在超时时间内重试 **on** 命令的次数上限。有些设备不支持多个连接。如果设备忙碌了处理另一个任务，操作可能会**失败**，因此如果还有剩余时间，Pacemaker 会自动重试操作。使用这个选项来更改 Pacemaker 在放弃前重试 **on** 操作的次数。

除了可以为单独的隔离设备设置的属性外，您还可以设置用来决定隔离行为的集群属性，如下表所述。

表 10.3. 确定隔离行为的集群属性

选项	默认值	描述

选项	默认值	描述
stonith-enabled	true	表示失败的节点以及带有资源无法停止的节点应该被隔离。保护数据需要将此设置为 **true**。 如果为 **true** 或未设置，集群将拒绝启动资源，除非也配置了一个或多个 STONITH 资源。 红帽只支持将这个值设置为 **true** 的集群。
stonith-action	reboot	发送到隔离设备的操作。允许的值：**reboot**、**off**。也允许使用值 **poweroff**，但只适用于旧的设备。
stonith-timeout	60s	等待 STONITH 操作完成的时间。
stonith-max-attempts	10	在集群不再立即重新尝试之前，隔离可以失败的次数。
stonith-watchdog-timeout		在认为某个节点被硬件 wathcdog 终止前等待的最长时间。建议将此值设置为硬件 watchdog 超时值的两倍。只有在使用 watchdog-only SBD 配置进行隔离时才需要这个选项。
concurrent-fencing	true	允许并行执行隔离操作。
fence-reaction	stop	决定集群节点在收到其自身隔离通知时应如何响应。如果错误配置了隔离，或者使用 fabric 隔离方式当没有中断集群的通信，集群节点可能会收到其自身隔离的通知信息。允许的值为 **stop**，来尝试立即停止 Pacemaker 并保持停止状态，或者为 **panic**，来尝试立即重启本地节点，并在失败时退回到停止状态。 虽然此属性的默认值是 **stop**，但这个值的最安全选择是 **panic**，它尝试立即重启本地节点。如果您希望使用 stop（通常是使用 fabric 隔离方式时），建议对这个参数进行明确设定。
priority-fencing-delay	0（禁用）	设置一个允许您配置双节点集群的隔离延迟，以便在脑裂情况下，运行最少或最不重要资源的节点是被隔离的节点。有关隔离延迟参数及其相互作用的一般信息，请查看 隔离延迟。

有关设置集群属性的详情，请参考 设置和删除集群属性。

10.4. 隔离延迟

当集群通信在双节点集群中丢失时，一个节点可能首先检测到此节点，并隔离其他节点。但是，如果两个节点同时检测到这个，则每个节点都可以启动其他节点的隔离，使两个节点关闭或重置。通过设置隔离延迟，您可以降低两个集群节点相互隔离的可能性。您可以在具有超过两个节点的集群中设置延迟，但这通常不会有任何好处，因为只有具有仲裁的分区将启动隔离。

您可以根据系统要求设置不同类型的隔离延迟。

- **静态隔离延迟**
  静态隔离延迟是一个固定的、预先确定的延迟。在一个节点上设置静态延迟使该节点更有可能被隔离，因为它增加了检测到丢失的通信后，其他节点首先启动隔离的机会。在主动/被动集群中，在被动节点上设置一个延迟，使得在通信中断时，被动节点将更有可能被隔离。您可以使用 **pcs_delay_base** 集群属性配置静态延迟。当单独的隔离设备用于每个节点时，或者一个隔离设备用于所有节点时，您可以设置此属性。

- **动态隔离延迟**
  动态隔离延迟是随机的。它可能会有所不同，并在需要隔离时确定。您可以配置一个随机延迟，并使用 **pcs_delay_max** 集群属性为组合的基本延迟和随机延迟指定最大值。当每个节点的隔离延迟是随机的时，被隔离的节点也是随机的。如果您的集群被配置为在主动/主动设计中具有所有节点的单个隔离设备，则可能会发现此功能很有用。

- **优先级隔离延迟**
  优先级隔离延迟基于活跃的资源优先级。如果所有资源具有相同的优先级，则运行最少资源的节点是被隔离的节点。在大多数情况下，您只使用一个与延迟相关的参数，但可以组合它们。合并与延迟相关的参数，将资源的优先级值加在一起，以创建总延迟。您可以使用 **priority-fencing-delay** 集群属性配置优先级隔离延迟。您可以在主动/主动集群设计中发现此功能，因为它可以在节点之间的通信丢失时让运行最少资源的节点更有可能被隔离。

pcmk_delay_base 集群属性

设置 **pcmk_delay_base** 集群属性为隔离启用了基本延迟，并指定了基本延迟值。

当您除了设置 **pcmk_delay_base** 属性，又设置了 **pcmk_delay_max** 集群属性时，整个延迟会从添加到此静态延迟的随机延迟值中派生，以便总和保持在最大延迟之下。当您设置了 **pcmk_delay_base**，但没有设置 **pcmk_delay_max** 时，延迟没有随机的组件，它将是 **pcmk_delay_base** 的值。

您可以使用 **pcmk_delay_base** 参数为不同的节点指定不同的值。这允许在双节点集群中使用单个隔离设备，每个节点有不同的延迟。您不需要配置两个单独的设备来使用单独的延迟。要为不同的节点指定不同的值，您可以使用与 **pcmk_host_map** 类似的语法，将主机名映射到该节点的延迟值。例如，在隔离 **node1** 时，**node1:0;node2:10s** 将不会使用延迟，在隔离 **node2** 时使用 10 秒的延迟。

pcmk_delay_max 集群属性

设置 **pcmk_delay_max** 集群属性启用了隔离操作的随机延迟，并指定了最大延迟，这是组合的基本延迟和随机延迟的最大值。例如，如果基本延迟为 3，并且 **pcmk_delay_max** 为 10，则随机延迟将在 3 和 10 之间。

当您除了设置 **pcmk_delay_max** 属性外，还设置了 **pcmk_delay_base** 集群属性时，整个延迟是从添加到此静态延迟的随机延迟值中派生的，以便总和保持在最大延迟之下。当您设置 **pcmk_delay_max**，但没有设置 **pcmk_delay_base** 时，这个延迟没有静态组件。

priority-fencing-delay 集群属性

设置 **priority-fencing-delay** 集群属性允许您配置一个双节点集群，以便在脑裂的情况下，运行最少或最不重要资源的节点是被隔离的节点。

priority-fencing-delay 属性可以被设置为一个持续时间。这个属性的默认值为 0（禁用）。如果将此属性被设置为非零值，并且优先级 meta-attribute 为至少一个资源进行了配置，那么在脑裂的情况下，在其上运行的具有所有资源的最高组合优先级的节点将更有可能保持正常运行。例如，如果您设置了 **pcs resource defaults update priority=1** 和 **pcs property set priority-fencing-delay=15s**，且没有设置其他优先级，那么运行最多资源的节点将更有可能保持正常运行，因为其他节点在启动隔离前将等待 15 秒。如果特定资源比其他资源更重要，您可以赋予它更高的优先权。

如果为该克隆配置了优先级，运行可升级的克隆 master 角色的节点会得到额外的 1 点。

隔离延迟的交互

设置多种类型的隔离延迟会产生以下结果：

- 使用 **priority-fencing-delay** 属性设置的任何延迟都被添加到 **pcmk_delay_base** 和 **pcmk_delay_max** 隔离设备属性中的任何延迟中。当两个节点具有相等的优先级时，或者两个节点因为节点丢失以外的原因需要隔离时，这种行为允许一些延迟，例如当为资源监控器操作设置了 **on-fail=fencing** 时。当设置这些延迟的组合时，将 **priority-fencing-delay** 属性设置为一个明显大于**pcmk_delay_base** 和 **pcmk_delay_max** 中的最大延迟的值，以确保优先级的节点是首选的。将此属性设置为此值的两倍通常是安全的。

- 只有 Pacemaker 本身调度的隔离才能观察到隔离延迟。由外部代码（如 **dlm_controld**）调度的隔离和由 **pcs stonith fence** 命令实现的隔离不会向隔离设备提供必要信息。

- 有些单独的隔离代理实现一个延迟参数，其名称由代理决定，并且其独立于使用 **pcmk_delay_*** 属性配置的延迟。如果同时配置了这些延迟，它们会被加在一起，通常不会一起使用。

10.5. 测试隔离设备

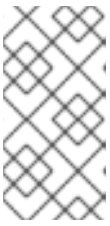
隔离(fencing)是红帽集群基础结构的基本部分，对于验证或测试隔离是否正常运行至关重要。

步骤

使用以下步骤测隔离护设备。

1. 使用 ssh、telnet、HTTP 或者任何远程协议连接到该设备以便手动登录并测试隔离设备或者查看给出的输出。例如，如果您要为启用了 IPMI 的设备配置隔离，则尝试使用 **ipmitool** 远程登录。记录手动登录时使用的选项，因为在使用隔离代理时可能需要使用这些选项。
   如果您无法登录到隔离设备，请确定设备是可以被 ping 到的，没有因为如防火墙等配置限制对隔离设备的访问，在隔离设备中启用了远程访问，且有正确的凭证。

2. 使用隔离代理脚本手动运行隔离代理。这不需要集群服务正在运行，因此您可以在集群配置该设备前执行这个步骤。这可保证在继续前隔离设备响应正常。



注意

这些示例将为 iLO 设备使用 **fence_ipmilan** 隔离代理脚本。您使用的实际隔离代理以及调用代理的命令取决于服务器硬件。您应该参考您使用的隔离保护代理的 man 页来确定要指定的选项。您通常需要了解隔离设备的登录和密码，以及其它与该隔离设备相关的信息。

以下示例显示了使用 **-o status** 参数运行 **fence_ipmilan** 隔离代理脚本的格式，来检查另一个节点上的隔离设备接口的状态，而实际上并没有对该节点进行隔离。这可让您在尝试重新引导节点前测试该设备并使其可用。在运行这个命令时，您可以为 iLO 设备指定打开和关闭权限的 iLO 用

户的名称和密码。

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

以下示例显示了使用 **-o reboot** 参数运行 **fence_ipmilan** 隔离代理脚本的格式。在一个节点上运行此命令可重新引导此 iLO 设备管理的节点。

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

如果隔离代理无法正确地执行 status、off、on 或 reboot 操作，您应该检查硬件、隔离设备的配置以及命令的语法。另外，您可以运行启用了 debug 输出的隔离代理脚本。调试输出会记录隔离设备时失败的事件，对于一些隔离代理，这个信息可能非常有用。

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

当诊断发生的故障时，您应该确定手动登录到隔离设备时指定的选项与您使用隔离代理传递给隔离代理的操作相同。

对于支持加密连接的隔离代理，您可能会因为证书验证失败而看到一个错误，这需要您信任主机或使用隔离代理的 **ssl-insecure** 参数。同样，如果在目标设备上禁用了 SSL/TLS，可能需要在为隔离代理设置 SSL 参数时考虑此事项。

注意

如果正在测试的隔离代理是 **fence_drac** 、**fence_ilo** 或系统管理设备的其他一些隔离代理，并且一直出现故障，则返回尝试 **fence_ipmilan**。大多数系统管理卡支持 IPMI 远程登录，唯一支持的隔离代理是 **fence_ipmilan**。

3. 一旦隔离设备在集群中配置了与手动操作相同的选项，并且集群已经启动，则可以从任何节点（或者多次从不同的节点）使用 **pcs stonith fence** 命令来测试隔离，如下例所示。**pcs stonith fence** 命令从 CIB 中读取集群配置，并调用配置的隔离代理来执行隔离操作。这会验证集群配置是否正确。

```
# pcs stonith fence node_name
```

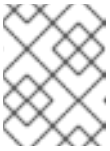
如果 **pcs stonith fence** 命令正常工作，这意味着发生隔离事件时集群的隔离配置应该可以正常工作。如果命令失败，这意味着集群管理无法通过它获取的配置调用隔离设备。检查以下问题并根据需要更新集群配置。

- 检查您的隔离配置。例如，如果您使用了主机映射，则应该确保系统可以使用您提供的主机名查找节点。
- 检查该设备的密码和用户名是否包含 bash shell 可能会错误解析的特殊字符。请确定，使用引号来包括您输入的密码和用户名是否可以解决这个问题。
- 检查是否可以使用您在 **pcs stonith** 命令中指定的 IP 地址或主机名连接到设备。例如：如果您在 stonith 命令中给出主机名，但使用 IP 地址进行测试，则这不是一个有效的测试。
- 如果您可以访问您的隔离设备使用的协议，使用那个协议尝试连接该设备。例如，很多代理都使用 ssh 或者 telnet。您应该尝试使用您在配置该设备时提供的凭证连接到该设备，查看是否收到有效提示符并登录该设备。
  如果您确定所有参数都正确，但仍无法连接到隔离设备，则可以查看隔离设备的日志信息（如果隔离设备提供了日志）。这会显示该用户是否已连接以及该用户发出什么命令。您还

可以在 **/var/log/messages** 文件中搜索 stonith 实例和错误，这可以了解发生了什么，但有些代理可以提供额外的信息。

4. 隔离设备测试正常工作并启动并运行集群后，测试实际故障。要做到这一点，在集群中执行应启动令牌丢失的操作。

   - 关闭网络。如何关闭网络取决于您的具体配置。在很多情况下，您可以从主机中物理拔掉网线或电源电缆。有关模拟网络故障的详情，请参考 在 RHEL 集群上模拟网络故障的正确方法是什么？

     注意

     不推荐通过在本地主机中禁用网络接口而不是物理断开网线或者电源电缆的方法进行测试，因为这无法准确模拟典型的实际失败。

   - 使用本地防火墙的阻塞 corosync 的入站和出站网络流落。
     以下示例会阻止 corosync，假设使用默认的 corosync 端口，**firewalld** 用作本地防火墙，corosync 使用的网络接口位于默认防火墙区内：

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

   - 使用 **sysrq-trigger** 模拟崩溃，并使您的机器死机。请注意，触发内核 panic 可能会导致数据丢失；建议首先禁用集群资源。

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

10.6. 配置隔离级别

Pacemaker 通过一个称为隔离拓扑的功能实现有多个设备的节点的隔离。要实现拓扑结构，根据常规创建独立设备，然后在配置中的隔离拓扑部分定义一个或多个隔离级别。

Pacemaker 会按如下方式处理隔离级别：

- 级别以整数形式递增，从 1 开始。
- 如果设备失败，对当前级别的处理会中断。不会执行该级别的其他设备，而是尝试下一个级别。
- 如果所有设备被成功隔离，那么该级别已成功，且不会尝试其他级别。
- 当一个级别被通过（success）或所有级别都已经被尝试（failed）后，操作就会完成。

使用以下命令为节点添加隔离级别。这些设备以用逗号分开的 **stonith** id 列表的形式提供，这些设备是用于该级别的节点进行尝试的。

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

以下命令列出目前配置的所有隔离级别。

```
pcs stonith level
```

在以下示例中，为节点 **rh7-2** 配置了两个隔离设备：名为 **my_ilo** 的 ilo 隔离设备，以及名为 **my_apc** 的 apc 隔离设备。这些命令设置了隔离级别，如果设备 **my_ilo** 失败且无法隔离该节点，Pacemaker 会尝试使用设备 **my_apc**。本例还显示了配置级别后， **pcs stonith level** 命令的输出。

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

以下命令删除指定节点和设备的隔离级别。如果没有指定节点或设备，则您指定的隔离级别会从所有节点中删除。

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

以下命令清除指定节点或者 stonith id 的隔离级别。如果您没有指定节点或 stonith id，则会清除所有隔离级别。

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

如果您指定一个以上的 stonith id，则必须用逗号分开（不要有空格），如下例所示。

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

以下命令可验证所有在隔离级别指定的隔离设备和节点是否存在。

```
pcs stonith level verify
```

您可以在隔离拓扑中通过在节点名称上应用的正则表达式、节点属性及其值来指定节点。例如，以下命令将节点 **node1**、**node2** 和 **node3** 配置为使用隔离设备 **apc1** 和 **apc2**，节点 **node4**、**node5** 和 **node6** 使用隔离设备 **apc3** 和 **apc4**。

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

以下命令通过使用节点属性匹配得到同样的结果。

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

10.7. 配置冗余电源的隔离

当为冗余电源配置隔离时，集群必须确保在尝试重启主机时，在恢复电源前两个电源都关闭。

如果节点永远无法完全断电，则该节点可能无法释放其资源。这可能会导致同时访问这些资源，并导致节点崩溃的问题。

您需要为每个设备定义一次，并指定它们需要隔离该节点，如下例所示。

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

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

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

10.8. 显示配置的隔离设备

以下命令显示所有目前配置的隔离设备。如果指定了 *stonith_id*，命令仅显示那个配置的隔离设备的选项。如果指定了 **--full** 选项，则会显示所有配置的隔离选项。

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

10.9. 使用 PCS 命令导出隔离设备

从 Red Hat Enterprise Linux 9.1 开始，您可以显示 **pcs** 命令，该命令可用于使用 **pcs stonith config** 命令的 **--output-format=cmd** 选项在不同的系统上重新创建配置的隔离设备。

以下命令创建一个 **fence_apc_snmp** 隔离设备，并显示您可以用来重新创建该设备的 **pcs** 命令。

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

10.10. 修改和删除隔离设备

使用以下命令修改或添加当前配置的隔离设备。

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

使用 **pcs stonith update** 命令更新 SCSI 隔离设备会导致在隔离资源运行的同一节点上运行的所有资源重启。您可以使用以下命令的任一版本来更新 SCSI 设备，而不会重启其他集群资源。从 RHEL 9.1 开始，SCSI 隔离设备可配置为多路径设备。

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

使用以下命令从当前的配置中删除隔离设备。

```
pcs stonith delete stonith_id
```

10.11. 手动隔离一个集群节点

您可以使用以下命令手动隔离节点。如果您指定了 **--off**，这将使用 **off** API 调用 stonith，这将关闭节点，而不是重启节点。

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

如果隔离设备无法隔离节点，即使它不再活跃，集群可能无法恢复节点上的资源。如果发生了这种情况，在手动确定该节点已关闭后，您可以输入以下命令向集群确认节点已关闭，并释放其资源以用于恢复。

警告

如果您指定的节点实际上没有关闭，但运行了通常由集群控制的集群软件或服务，则数据崩溃/集群失败将发生。

```
pcs stonith confirm node
```

10.12. 禁用隔离设备

要禁用隔离设备/资源，请运行 **pcs stonith disable** 命令。

以下命令禁用隔离设备 **myapc**。

```
# pcs stonith disable myapc
```

10.13. 防止节点使用隔离设备

要防止特定节点使用隔离设备，您可以为隔离资源配置位置限制。

以下示例阻止隔离设备 **node1-ipmi** 在 **node1** 上运行。

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

10.14. 配置 ACPI 以用于集成的隔离设备

如果您的集群使用集成的隔离设备，必须配置 ACPI（高级配置和电源界面）以保证迅速和完全的隔离。

如果将集群节点配置为使用集成的隔离设备保护，则为该节点禁用 ACPI Soft-Off。禁用 ACPI Soft-Off 允许集成的隔离设备立即并完全关闭节点，而不是尝试彻底关闭（例如，**shutdown -h now**）。否则，如果启用了 ACPI Soft-Off，集成的隔离设备可能需要 4 秒以上的时间来关闭节点（请参阅下面的备注）。另外，如果启用了 ACPI Soft-Off，且在关闭过程中有一个节点 panic 或停滞，则集成的保护设备可能无法关闭该节点。在这些情况下，隔离会被延迟或者失败。因此，当使用集成隔离设备隔离节点并启用 ACPI Soft-Off时，集群恢复会很慢，或者需要管理员进行干预才能恢复。

注意

保护节点所需时间取决于所使用的集成的保护设备。有些集成的保护设备性能与按住电源按钮相当，因此隔离设备可在 4-5 秒内关闭该节点。其他集成的隔离设备性能与按电源开关一致，依靠操作系统关闭该节点，因此隔离设备关闭该节点的时间要大大超过 4-5 秒。

- 禁用 ACPI Soft-Off 的首选方法是将 BIOS 设置改为 "instant-off" 或一个在不延迟关闭该节点的对等设置，如下为 "Disabling ACPI Soft-Off"。

使用 BIOS 禁用 ACPI Soft-Off 可能不适用于某些系统。如果无法使用 BIOS 禁用 ACPI Soft-Off，您可以使用以下备选方法之一禁用 ACPI Soft-Off：

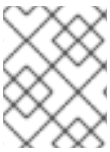
- 在 **/etc/systemd/logind.conf** 文件中设置 **HandlePowerKey=ignore**，并在隔离时验证节点节点立即关闭，如 logind.conf 文件中的"Disabling ACPI Soft-Off"中所述。这是禁用 ACPI Soft-Off 的第一个替代方法。
- 在内核引导命令行中附加 **acpi=off**，如"在 GRUB 2 文件中完全禁用 ACPI"中所述。这是禁用 ACPI Soft-Off 的第二个替代方法，如果首选方法或第一个替代方法不可用。

重要

这个方法可完全禁用 ACPI。当 ACPI 被完全禁用时，以下计算机可能无法正确引导。*只有在*其他方法无法在您的集群中使用时，才使用这个方法。

10.14.1. 使用 BIOS 禁用 ACPI Soft-Off

您可以按照以下步骤配置每个集群节点的 BIOS 来禁用 ACPI Soft-Off。

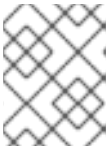


注意

使用 BIOS 禁用 ACPI Soft-Off 的步骤可能因服务器系统而异。您应该在您的硬件文档中验证此步骤。

步骤

1. 重启节点，并启动 **BIOS CMOS 设置工具** 程序。
2. 进入 Power 菜单（或者对等的电源管理菜单）。
3. 在 Power 菜单中，将 **Soft-Off by PWR-BTTN** 功能（或等效的功能）设置为 **Instant-Off**（或使用无延迟电源按钮关闭节点的对等设置）。以下 **BIOS CMOS Setup Utiliy** 示例显示 **ACPI Function** 设置为 **Enabled**， **Soft-Off by PWR-BTTN** 设置为 **Instant-Off**。



注意

与 **ACPI Function** 等效，**Soft-Off by PWR-BTTN** 和 **Instant-Off** 可能因计算机而异。但这个过程的目的是配置 BIOS，以便计算机能无延迟地关闭电源按钮。

4. 退出 **BIOS CMOS 设置工具** 程序，保存 BIOS 配置。
5. 验证在隔离时该节点是否立即关闭。有关测试隔离设备的详情，请参考 测试隔离设备。

BIOS CMOS 设置实用程序 ：

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

```
+---------------------------------------------|-------------------+
|   ACPI Function            [Enabled]     |    Item Help      |
|   ACPI Suspend Type        [S1(POS)]     |-------------------|
| x Run VGABIOS if S3 Resume   Auto        |   Menu Level   *  |
|   Suspend Mode             [Disabled]    |                   |
|   HDD Power Down           [Disabled]    |                   |
|   Soft-Off by PWR-BTTN     [Instant-Off  |                   |
|   CPU THRM-Throttling      [50.0%]       |                   |
|   Wake-Up by PCI card      [Enabled]     |                   |
|   Power On by Ring         [Enabled]     |                   |
|   Wake Up On LAN           [Enabled]     |                   |
| x USB KB Wake-Up From S3     Disabled    |                   |
|   Resume by Alarm          [Disabled]    |                   |
| x  Date(of Month) Alarm       0          |                   |
| x  Time(hh:mm:ss) Alarm       0 :  0 :   |                   |
|   POWER ON Function        [BUTTON ONLY  |                   |
| x KB Power ON Password       Enter       |                   |
| x Hot Key Power ON           Ctrl-F1     |                   |
|                                 |                |
|                                 |                |
+---------------------------------------------|-------------------+
```

本例展示了 **ACPI Function** 设置为 **Enabled**， **Soft-Off by PWR-BTTN** 设置为 **Instant-Off**。

10.14.2. 在 logind.conf 文件中禁用 ACPI Soft-Off

要禁用 **/etc/systemd/logind.conf** 文件中的电源密钥处理，请使用以下流程。

步骤

1. 在 **/etc/systemd/logind.conf** 文件中定义以下配置：

   ```
   HandlePowerKey=ignore
   ```

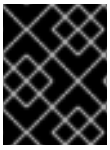
2. 重启 **systemd-logind** 服务：

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

3. 验证在隔离时该节点是否立即关闭。有关测试隔离设备的详情，请参考 测试隔离设备。

10.14.3. 在 GRUB 2 文件中完全禁用 ACPI

您可以通过在内核的 GRUB 菜单条目中添加 **acpi=off** 来禁用 ACPI Soft-Off。



重要

这个方法可完全禁用 ACPI。当 ACPI 被完全禁用时，以下计算机可能无法正确引导。*只有在*其他方法无法在您的集群中使用时，才使用这个方法。

步骤

在 GRUB 2 文件中使用以下步骤禁用 ACPI：

1. 将 **--args** 选项与 **grubby** 工具的 **--update-kernel** 选项结合使用，来更改每个集群节点的 **grub.cfg** 文件，如下所示：

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

2. 重新引导节点。
3. 验证在隔离时该节点是否立即关闭。有关测试隔离设备的详情，请参考 测试隔离设备。

第 11 章 配置集群资源

使用以下命令创建和删除集群资源。

创建集群资源的命令格式如下：

> pcs resource create *resource_id* [*standard*:[*provider*:]]*type* [*resource_options*] [op *operation_action operation_options* [*operation_action operation options*]...] [meta *meta_options*...] [clone [*clone_id*] [*clone_options*] | promotable [*clone_id*] [*clone_options*] [--wait[=*n*]]

集群资源创建的关键选项包括：

- **--before** 和 **--after** 选项指定添加的资源相对于资源组中已存在的资源的位置。
- 指定 **--disabled** 选项表示资源不会自动启动。

对可在集群中创建的资源数量没有限制。

您可以通过配置该资源的约束来决定集群中资源的行为。

资源创建示例

以下命令创建一个 standard的 **ocf**、名为 **VirtualIP** 的资源，provider 为 **heartbeat**，类型为 **IPaddr2**。这个资源的浮动地址是 192.168.0.120，系统会每 30 秒检查一次这个资源是否在运行。

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

另外，您可以省略 *standard* 和 *provider* 字段，并使用以下命令：这将默认为 **ocf** 标准， **heartbeat** 的 provider。

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

删除配置的资源

使用以下命令删除配置的资源。

> pcs resource delete *resource_id*

例如，以下命令删除具有 **VirtualIP** 资源 ID 的现有资源。

> **# pcs resource delete VirtualIP**

11.1. 资源代理标识符

您为资源定义的标识符告诉集群用于该资源的代理，在哪里找到代理及其合规标准。

下表描述了资源代理的这些属性。

表 11.1. 资源代理标识符

项	描述
standard	代理符合的标准。允许的值及其含义： * **ocf** - 指定的 *type* 是符合 Open Cluster Framework Resource Agent API 的可执行文件名称，位于 **/usr/lib/ocf/resource.d/*provider*** 下 * **lsb** - 指定的 *type* 是符合 Linux Standard Base Init Script Actions 的可执行文件名称。如果类型没有指定完整路径，则系统将在 **/etc/init.d** 目录中查找该路径。 * **systemd** - 指定的 *type* 是已安装的 **systemd** 单元的名称 * **service** - pacemaker 将搜索指定的 *类型*，首先作为一个 **lsb** 代理，然后作为 **systemd** 代理 * **nagios** - 指定的 *类型* 是符合 Nagios Plugin API 的可执行文件名称，位于 **/usr/libexec/nagios/plugins** 目录中，其中 OCF 风格的元数据单独存储在 **/usr/share/nagios/plugins-metadata** 目录中（由某些常见插件的 **nagios-agents-metadata** 软件包提供）。
type	您要使用的资源代理的名称，如 **IPaddr** 或 **Filesystem**
provider	OCF spec 允许多个厂商提供相同的资源代理。红帽提供的大多数代理都使用 **heartbeat** 作为provider。

下表总结了显示可用资源属性的命令。

表 11.2. 显示资源属性的命令

pcs Display 命令	输出。
pcs resource list	显示所有可用资源的列表。
pcs 资源标准	显示可用资源代理标准列表。
pcs resource provider	显示可用资源代理供应商列表。
pcs resource list *string*	显示根据指定字符串过滤的可用资源列表。您可以使用这个命令显示根据标准名称、供应商或类型过滤的资源。

11.2. 显示特定于资源的参数

对于任何单独的资源，您可以使用以下命令显示资源描述、您可以为该资源设置的参数以及为资源设置的默认值。

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

例如，以下命令显示类型为 **apache** 的资源的信息。

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

...
```

11.3. 配置资源 META 选项

除了特定于资源的参数外，您还可以为任何资源配置其他资源选项。集群会使用这些选项来决定您的资源的行为。

下表描述了资源 meta 选项。

表 11.3. 资源元数据选项

项	默认值	描述
priority	**0**	如果不是所有资源都处于活跃状态，集群将停止较低优先级的资源，以便保持优先权更高的资源的活跃状态。
target-role	**Started**	指明集群应尝试将此资源保留在什么状态。允许的值： * **Stopped** - 强制停止资源 * **Started** - 允许启动资源（如果是可提升的克隆，则进行提升（如果适用） * **Promoted** - 允许启动资源，并在可能的情况下提升资源 * **Unpromoted** - 允许资源启动，但仅在资源可提升但却处于未提升模式
is-managed	**true**	指明是否允许集群启动和停止资源。允许的值： **true**,**false**
resource-stickiness	1	指示资源倾向于保留在当前位置的程度。

项	默认值	描述
requires	Calculated	指示可在什么情况下启动资源。 除非在下列情况下，否则默认为 **fencing**。可能的值： * **nothing** - 集群总是可以启动资源。 * **quorum** - 只有在大多数配置的节点处于活动状态时，集群才能启动此资源。如果 **stonith-enabled** 为 **false**，或资源的 **standard** 是 **stonith**，则这是默认值。 * **fencing** - 只有大多数配置的节点活跃*并且*隔离所有失败或未知节点时，集群才可以启动此资源。 * **取消隔离** - 只有大多数配置的节点活跃 *且* 所有失败或未知节点都被隔离，*且* 只能在未*隔离的* 节点上，集群才能启动此资源。如果为隔离设备设置了 **provides=unfencing stonith** 元选项，则这是默认值。
migration-threshold	**INFINITY**	在将这个节点标记为不允许托管此资源之前，节点上可能会发生多少个故障。值 0 表示禁用了此功能（节点永远不会标记为无效）；相反，集群将 **INFINITY**（默认值）视为非常大但有限的数。只有在失败的操作有 **on-fail=restart**（默认值）时，这个选项才会生效，如果集群属性 **start-failure-is-fatal** 为 **false**，则此选项还可用于失败的启动操作。
failure-timeout	**0**（禁用）	与 **migration-threshold** 选项结合使用，可指示在采取操作之前要等待多少秒，就像没有发生故障一样，并可能允许资源返回到失败的节点上。

项	默认值	描述
multiple-active	**stop_start**	代表当在多个节点上都找到活跃的资源时集群应该做什么。允许的值： * **block** - 将资源标记为非受管 * **stop_only** - 停止所有活动的实例，并使其保持这种状态 * **stop_start** - 停止所有活跃的实例，并仅在一个位置启动资源 * **stop_unexpected** - (RHEL 9.1 及更新版本)只停止资源的意外实例，而无需完全重启。用户负责验证服务及其资源代理是否可以与额外的活跃实例一起正常工作，而无需全面重启。
critical	**true**	为涉及资源作为从属资源(*target_resource*)的所有 colocation 约束设置 **influence** 选项的默认值，包括在资源属于资源组时创建的隐式 colocation 限制。**influence** colocation 约束选项决定在依赖资源达到失败的迁移阈值时，集群是否会将主和依赖资源移动到另一个节点，或者集群是否使依赖资源离线，而不会导致服务切换。**critical** 资源 meta 选项的值可以是 **true** 或 **false**，默认值为 **true**。
allow-unhealthy-nodes	**false**	(RHEL 9.1 及更高版本)当设置为 **true** 时，节点不会因为降级的健康状况强制关闭该节点。当健康资源设置了此属性时，集群可以自动检测节点的健康状态恢复，并将资源移回节点。节点的运行状况由基于本地条件的健康资源代理设置的健康属性和决定集群如何对这些条件做出响应的与策略相关的选项的组合决定的。

11.3.1. 更改资源选项的默认值

您可以使用 **pcs resource defaults update** 命令更改所有资源的资源选项默认值。以下命令将资源粘性的默认值重置为 **100**。

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

原始的 **pcs resource defaults *name=value*** 命令（在以前的版本中为所有资源设置默认值）仍然被支持，除非设置了多组默认值。但是，**pcs resource defaults update** 现在是该命令的首选版本。

11.3.2. 更改一组资源选项的默认值

您可以使用 **pcs resource defaults set create** 命令创建多个资源默认值，该命令允许您指定包含 **resource** 表达式的规则。在使用此命令指定的规则中，仅允许 **resource** 和 **date** 表达式（包括 **and**, **or** 和括号）。

使用 **pcs resource defaults set create** 命令，您可以为特定类型的所有资源配置默认值。例如，如果您正在运行数据库，需要很长时间才能停止，可以提高数据库类型的 **resource-stickiness** 默认值，以防止这些资源更频繁地迁移到其他节点。

以下命令将类型为 **pqsql** 的所有资源的 **resource- sticky** 设置为 100。

- **id** 选项（即资源默认值的名称）不是强制性的。如果您没有设定此选项， **pcs** 则会自动生成 ID。设置这个值可让您提供更描述性的名称。
- 在这个示例中，**::pgsql** 代表一个任何类、任何厂商、类型为 **pgsql** 的资源。
  - **ocf:heartbeat:pgsql** 代表类为 **ocf**，厂商为 **heartbeat**，类型为 **pgsql**。
  - **ocf:pacemaker:** 代表类为 **ocf**，厂商为 **pacemaker**，类型为任意的资源。

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

要更改现有集合中的默认值，请使用 **pcs resource defaults set update** 命令。

11.3.3. 显示当前配置的资源默认设置

pcs resource defaults 命令显示目前配置的资源选项默认值列表，包括您指定的规则。

以下示例显示了在将 **resource-stickiness** 的默认值重置为 100 后，此命令的输出。

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

以下示例显示，在将类型为 **pqsql** 的所有资源的 **resource-stickiness** 默认值重新设置为 100，把 **id** 选项设置为 **id=pgsql-stickiness** 后的输出。

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

11.3.4. 在创建资源时设置 meta 选项

是否重置资源 meta 选项的默认值，您可以在创建资源时将特定资源的资源选项设置为默认值，而不是默认值。以下显示了在为资源元选项指定值时使用的 **pcs resource create** 命令的格式。

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

例如，以下命令创建一个 **resource-stickiness** 值为 50 的资源。

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

您还可以使用以下命令为现有资源、组或克隆的资源设置资源 meta 选项的值。

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

在以下示例中，有一个名为 **dummy_resource** 的现有资源。此命令将 **failure-timeout** 元选项设置为 20 秒，因此资源可在 20 秒内尝试在同一节点上重启。

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

执行此命令后，可以显示资源的值，以验证是否设置了 **failure-timeout=20s**。

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

11.4. 配置资源组

集集的一个最常见的元素是一组资源，这些资源需要放置在一起，并按顺序启动并按反顺序停止。为简化此配置，Pacemaker 支持资源组的概念。

11.4.1. 创建资源组

您可以使用以下命令创建资源组，指定要包含在组中的资源。如果组不存在，这个命令会创建组。如果组存在，这个命令会向组群添加其他资源。这些资源将按您使用此命令指定的顺序启动，并以相反的顺序停止。

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

您可以使用此命令的 **--before** 和 **--after** 选项来指定添加的资源相对于组中已存在的资源的位置。

您还可以使用以下命令在创建新资源时，将新资源添加到现有组中。您创建的资源会添加到名为 *group_name* 的组中。如果 *group_name* 不存在，则会创建它。

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

对组可以包含的资源数量没有限制。组群的基本属性如下。

- 资源在一个组中在一起。
- 资源按照您指定的顺序启动。如果组中的资源无法在任何位置运行，则不允许在该资源之后指定资源运行。
- 资源按照您指定的顺序的相反顺序停止。

以下示例创建了一个名为 **shortcut** 的资源组，其中包含现有的资源 **IPaddr** 和 **Email**。

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

在本例中：

- **IPaddr** 首先启动，然后是 **Email**。
- **Email** 资源首先停止，然后是 **IPAddr**。
- 如果 **IPaddr** 无法在任何地方运行，则 **Email** 也无法运行。
- 但是，如果 **Email** 无法在任何地方运行，这不会影响 **IPaddr**。

11.4.2. 删除资源组

您可以使用以下命令从组中删除资源。如果组中没有剩余资源，这个命令会删除组本身。

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

11.4.3. 显示资源组

以下命令列出所有目前配置的资源组。

```
pcs resource group list
```

11.4.4. 组选项

您可以为资源组设置以下选项，它们的含义与为单个资源设置时相同： **priority**、**target-role**, **is-managed**。有关资源 meta 选项的详情，请参考 配置资源 meta 选项。

11.4.5. 组粘性

粘性（stickiness）在组中是可选的，它代表一个资源倾向于停留在组中的程度。组的每个活跃资源都会为组的总数贡献其粘性值。因此，如果默认的 **resource-stickiness** 为 100，并且组有 7 个成员，其中 5 个处于活动状态，则整个组将首选其当前位置，分数为 500。

11.5. 确定资源行为

您可以通过配置该资源的约束来决定集群中资源的行为。您可以配置以下约束类别：

- **location** 约束 - 位置约束决定资源可在哪个节点上运行。有关配置位置约束的详情，请参阅 确定资源可在哪些节点上运行。
- **order** 限制 - 排序约束决定资源运行的顺序。有关配置排序约束的详情，请参考 确定集群资源的运行顺序。
- **colocation** 约束 - 共处约束决定了资源相对于其他资源将被放置在何处。有关托管约束的详情，请参考 托管集群资源。

简而言之，配置一组限制会将一组资源放在一起，并确保资源按顺序启动并按相反顺序停止，Pacemaker 支持资源组的概念。创建资源组后，您可以像为单个资源配置限制一样，对组本身配置限制。

第 12 章 确定资源可在哪些节点上运行

位置限制决定资源可在哪些节点上运行。您可以配置位置限制，以确定资源是否首选或避免指定节点。

除了位置约束外，资源运行的节点还受到该资源的 **resource-stickiness** 值的影响，这决定了资源在当前运行的节点上的保留程度。有关设置 **resource-stickiness** 值的详情，请参考 配置资源以首选其当前节点。

12.1. 配置位置限制

您可以配置基本的位置约束，以指定资源是首选某个节点还是避免某个节点，使用可选的 **score** 值来指示约束的相对首选程度。

以下命令为资源创建一个位置约束，以偏好指定节点。请注意，可以使用单个命令为多个节点在特定资源上创建限制。

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

以下命令为资源创建一个位置约束,以避免指定节。

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

下表总结了配置位置限制的基本选项的含义。

表 12.1. 位置限制选项

项	描述
rsc	资源名称
node	节点的名称
分数	正整数值来指示给定资源应首选的资源还是避免给定节点的首选程度。**INFINITY** 是资源位置约束的默认 **score** 值。 **pcs contraint location *rsc* prefers** 命令中的 **score** 值为 **INFINITY**，表示该节点首选该节点（如果节点可用），但不会阻止资源在其它节点上运行（如果指定的节点不可用）。 **pcs contraint location *rsc* avoids** 命令中的 **score** 值为 **INFINITY**，表示资源将永远不会运行在那个节点上，即使其它节点都不可用。这等同于设置了 score 为 **-INFINITY** 的 **pcs constraint location add** 命令。 一个数字分数（即不是 **INFINITY**）意味着这个约束是可选的，除了其它因素外强于它，此约束将会被遵守。例如，如果资源已放置到另一个节点上，其 **resource-stickiness** 分数高于 **prefers** 位置约束的分数，则该资源将保留在其中。

以下命令创建了一个位置约束，以指定资源 **Webserver** 首选节点 **node1**。

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

pcs 支持命令行中的位置限制中的正则表达式。这些限制适用于基于正则表达式匹配资源名称的多个资源。这可让您使用单一命令行配置多个位置限制。

以下命令创建了一个位置约束，从资源 **dummy0** 到 **dummy9** 都首选 **node1**。

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

因为 Pacemaker 使用 POSIX 扩展正则表达式，如 http://pubs.opengroup.org/onlinepubs/9699919799/basedefs/V1_chap09.html#tag_09_04 所述，您可以使用以下命令指定相同的约束。

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

12.2. 将资源发现限制为节点子集

Pacemaker 在任何位置启动资源前，它首先在每个节点上执行一次性 monitor 操作（通常称为"探测"），以了解资源是否已在运行。这种资源发现的过程可能会导致无法执行 monitor 的节点出现错误。

在节点上配置位置约束时，您可以使用 **pcs constraint location** 命令的 **resource-discovery** 选项，来为 Pacemaker 是否应该为指定资源在该节点上执行资源发现指示首选。将资源发现限制到物理上能够运行的节点子集可能会在有大量节点时显著提高性能。当使用 **pacemaker_remote** 来将节点数扩展到数百个节点范围时，应考虑此选项。

以下命令显示了为 **pcs constraint location** 命令指定 **resource-discovery** 选项的格式。在这个命令中，一个正的*分值*值对应一个基本位置约束，它配置为首选节点，而 *分数* 的负数值对应配置资源以避免节点的基本位置约束。与基本位置限制一样，您也可以使用这些限制的资源使用正则表达式。

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

下表总结了为资源发现配置约束的基本参数的含义。

表 12.2. 资源发现约束参数

项	描述
id	约束本身的用户选择的名称。
rsc	资源名称
node	节点的名称

分数	整数值来指示给定资源应首选的资源还是避免给定节点的首选程度。一个正的分值对应一个基本位置约束，它配置为首选节点，而分数的负数值对应配置资源以避免节点的基本位置约束。 **score** 的 **INFINITY** 值表示该节点可用时该节点首选，但不会阻止资源在指定节点不可用时在另一节点上运行。**score** 的 **-INFINITY** 值表示该资源永远不会在该节点上运行，即使没有其它节点可用。 一个数字分数（即不是 **INFINITY** 或 **-INFINITY**）意味着这个约束是可选的，除了其它因素外强于它，此约束将会被遵守。例如，如果资源已放置到另一个节点上，其 **resource-stickiness** 分数高于 **prefers** 位置约束的分数，则该资源将保留在其中。
resource-discovery 选项	* **always** - 始终为此节点上的指定资源执行资源发现。这是资源位置约束的默认 **resource-discovery** 值。 * **never** - 永远不会为这个节点上的指定资源执行资源发现。 * **exclusive** - 仅在此节点（以及其他标记为 **exclusive** 的节点）上对指定的资源执行资源发现。对跨不同节点的同一资源使用 **exclusive** 进行多次位置约束，会创建 **resource-discovery** 独占的节点子集。如果某个资源在一个或多个节点上标记为 **exclusive** 发现，则该资源仅被允许放在那个节点的子集中。

警告

将 **resource-discovery** 设置为 **never** 或 **exclusive** 的 Pacemaker 能够检测和停止运行不需要的服务实例（在不应该运行）。系统管理员可以确保该服务永远无法在没有资源发现的情况下在节点上活跃（比如卸载相关的软件）。

12.3. 配置位置约束策略

在使用位置限制时，您可以配置常规策略来指定资源可在哪些节点上运行：

- Opt-In 集群 - 配置一个集群，默认情况下，任何资源都无法在任何位置运行，然后有选择地为特定资源启用允许的节点。
- Opt-Out 集群 - 配置一个集群，默认情况下，所有资源都可在任何位置运行，然后为不允许在特定节点上运行的资源创建位置限制。

根据您的需要以及集群的组成，把集群设置为 opt-in 集群还是 opt-out 集群。如果大多数资源可以在大多数节点上运行，那么如果没有选择的协议则可能会导致配置更简单。另一方面，如果大多数资源只能在一小部分节点中运行，那么选择的配置可能比较简单。

12.3.1. 配置 "Opt-In" 集群

要创建一个 opt-in 集群，请将 **symmetric-cluster** 集群属性设置为 **false**，以防止资源默认在任何地方运行。

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

为单个资源启用节点。以下命令配置位置约束，以便资源 **Webserver** 首选节点 **example-1**，资源 **Database** 首选节点 **example-2**，如果它们的首选节点都出现故障，则这两个资源都可以切换到节点 **example-3**。当为 opt-in 集群配置位置限制时，设置零分数可允许资源在节点上运行，而不表示首选或避免该节点。

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

12.3.2. 配置 "Opt-Out" 集群

要创建一个 opt-out 集群，请将 **symmetric-cluster** 集群属性设置为 **true**，以允许资源默认可在任何地方运行。如果没有明确设置 **symmetric-cluster**，则这是默认配置。

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

然后，以下命令将生成一个配置，该配置等同于 "Opt-In" 集群中的示例。如果它们的首选节点出现故障，这两个资源都可以切换到节点 **example-3**，因为每个节点都有一个隐含的为 0 的 score 。

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

请注意，不需要在这些命令中指定 INFINITY 分数，因为这是分数的默认值。

12.4. 配置资源以首选其当前节点

资源具有 **resource-stickiness** 值，您可以在创建资源时设置一个 meta 属性，如配置资源 meta 选项所述。**resource-stickiness** 值决定资源要在当前运行的节点上保留多少。Pacemaker 会将 **resource-stickiness** 值与其他设置（例如，位置约束的 score 值）一起考虑，以确定是否将资源移至其它节点还是将其留在原地。

resource-stickiness 值为 0 时，集群可以根据需要移动资源，以在节点之间平衡资源。这可能会导致资源在不相关的资源启动或停止时移动。对于正的粘性，资源更倾向于保持原处，只有在其他情况超出粘性时才会移动。这可能会导致新添加的节点在没有管理员干预的情况下无法获取分配给它们的任何资源。

RHEL 9 中新创建的集群将 **resource-stickiness** 的默认值设置为 1。这个小值可以被您创建的其他限制轻松覆盖，但可以防止 Pacemaker 在集群中无用地移动处于健康状态的资源。如果您希望从 **resource-stickiness** 值为 0 的集群行为，您可以使用以下命令 **resource-stickiness** 默认值改为 0：

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

如果您将现有集群升级到 RHEL 9，且您没有明确为 **resource-stickiness** 设置默认值，则 **resource-stickiness** 值保留 0，**pcs resource defaults** 命令将不会显示任何的粘性值。

因为 **resource-stickiness** 值为正，任何资源都不会移至新添加的节点。如果此时需要资源平衡，您可以临时将资源 **resource-stickiness** 设置为 0。

请注意，如果位置约束 score 高于 **resource-stickiness** 值，则集群仍然可以将健康资源移至位置约束所指向的节点。

如需有关 Pacemaker 如何确定资源的放置位置的更多信息，请参阅配置节点放置策略。

12.5. 使用 PCS 命令导出资源约束

从 Red Hat Enterprise Linux 9.3 开始，您可以使用 **pcs constraint** 命令的 **--output-format=cmd** 选项显示可在不同系统上重新创建配置的资源约束的 **pcs**命令。

以下命令创建一个 **IPaddr2** 资源和一个 **apache** 资源。

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

以下命令为两个资源配置位置约束、托管约束和顺序约束。

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

创建资源和约束后，以下命令显示可用于在不同系统上重新创建约束的 **pcs** 命令。

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

第 13 章 确定运行集群资源的顺序

要确定资源运行的顺序，您需要配置一个顺序约束。

以下显示了命令配置排序约束的格式。

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

下表总结了配置排序约束的属性和选项。

表 13.1. 顺序约束的属性

项	描述
resource_id	执行某个操作的资源的名称。
action	资源操作。*action* 属性可能的值如下： * **start** - 资源启动操作。 * **stop** - 资源停止操作。 * **promote** - 将资源从一个未提升资源提升为一个提升的资源。 * **demote** - 将资源从提升的资源降级到未提升的资源。 如果没有指定操作，则 **start** 为默认操作。
kind 选项	如何强制实施约束。**kind** 选项可能的值如下： * **Optional** - 仅在两个资源都执行指定操作时才适用。有关可选排序的详情，请参考 配置咨询排序。 * **Mandatory** - 总是强制限制（默认值）。如果您指定的第一个资源是停止或无法启动，则您指定的第二个资源必须停止。有关强制排序的详情，请参考 配置强制排序。 * **Serialize** - 确定您指定的资源不会同时出现两个 stop/start 动作。您指定的第一个和第二个资源可以按其中顺序启动，但必须在启动另一个资源前完成。一个典型的用例是资源启动在主机上造成高负载。
symmetrical 选项	如果为 true，则代表反向约束适用于相反的操作（例如，如果 B 在 A 启动后启动，则 B 会在 A 停止前停止）。对于 **kind** 为 **Serialize** 的排序约束不能为 symmetrical。对于 **Mandatory** 和 **Optional** kind，默认值是 **true**，对于 **Serialize**，默认值为 **false**。

使用以下命令从任何排序约束中删除资源。

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

13.1. 配置必须的排序

一个强制的顺序约束表示，在第一次操作成功完成了第一个资源之前，不应该对第二个资源启动第二个操作。可以排序的操作是 **stop**、**start**，对于可升级克隆，还有 **demote** 和 **promote**。例如："A then B"（相当于 "start A then start B"）表示 B 不会被启动，直到 A 成功启动为止。如果约束的 **kind** 选项被设置为 **Mandatory**，或保留为默认值，则排序约束是必须的。

如果 **symmetrical** 选项设定为 **true** 或保留为默认值，则相反的操作将遵循相反的排序。**start** 和 **stop** 操作是相反的操作，**demote** 和 **promote** 是相反的操作。例如：一个对称 "promote A" 排序意味着 "stop B then demote A" 表示 A 不能被降级，直到 B 成功停止。对称排序表示 A 状态的改变可能会导致操作调度到 B。例如，给定为 "A then B"，如果出现故障，B 将首先停止，A 将被停止，A 将启动，然后启动 A，那么 B 将启动。

请注意，集群会响应每个状态的更改。如果第一个资源在第二个资源启动停止操作前再次处于启动状态，则不需要重启第二个资源。

13.2. 配置公告顺序

当为顺序约束指定 **kind=Optional** 选项时，约束被视为可选的，且仅在两个资源都执行指定操作时才适用。您指定的第一个资源的状态更改不会对您指定的第二个资源起作用。

以下命令为名为 **VirtualIP** 和 **dummy_resource** 的资源配置咨询排序约束。

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

13.3. 配置排序的资源集

管理员创建排序的资源链的一个常见情况，例如，资源 A 在资源 B 之前启动，而资源 B 在资源 C 之前启动。如果您的配置需要创建一组资源，这些资源是按顺序共处和启动的，则您可以配置包含这些资源的资源组。

然而，在有些情况下，配置资源需要以指定顺序启动，因为资源组不合适：

- 您可能需要配置资源以启动，而且资源不一定是在一起的。
- 您可能有一个资源 C，它必须在资源 A 或 B 启动后启动，但 A 和 B 之间没有关系。
- 您可能有资源 C 和 D 在资源 A 和 B 启动时必须启动，但 A 和 B 之间没有关系，C 和 D 之间没有关系。

在这些情况下，您可以使用 **pcs constraint set** 命令对一组或几组资源创建顺序约束。

您可以使用 **pcs constraint order set** 命令为一组资源设置以下选项。

- **sequential**，可以设为 **true** 或 **false**，以指示资源集合是否可以相互排序。默认值为 **true**。将 **sequential** 设为 **false**，允许一个集合相对于排序约束中的其他集合进行排序，而不对其成员进行相互排序。因此，只有在约束里列出了多个集合时才有意义；否则，约束无效。
- **require-all**，它可以设为 **true** 或 **false**，以指示集合中的所有资源在继续前是否处于活动状态。将 **require-all** 设为 **false**，表示集合中只有一个资源需要启动，然后才能继续下一个集合。将 **require-all** 设为 **false** 没有任何作用，除非与未排序的集合一起使用，未排序的集合是那些 **sequential** 设置为 **false** 的集合。默认值为 **true**。
- **action**，可以被设置为 **start**, **promote**, **demote** 或 **stop**，如确定群集资源的运行顺序 中" Order Constraint"表中的"属性"中所述。
- **role**，可以被设置为 **Stopped**, **Started**, **Promoted**, 或 **Unpromoted**。

您可以按照 **pcs constraint set** 命令的 **setoptions** 参数，为一组资源设置以下约束选项。

- **ID**，为您定义的约束提供名称：
- **kind**，代表如何强制约束，如确定集群资源的运行顺序中的 "Properties of an Order Constraint" 表所述。
- **symmetrical**，设置约束的反向是否适用于相反操作，如在确定集群资源运行顺序 中" Order Constraint"表中的"Properties 所述。

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

如果您有三个名为 **D1**、**D2** 和 **D3** 的资源，以下命令将它们配置为排序的资源集。

> **# pcs constraint order set D1 D2 D3**

如果您有六个名为 **A**、**B**、**C**、**D**、**E**、和 **F** 的资源，则示例为一组启动的资源配置了排序约束，如下所示：

- **A** 和 **B** 的启动是相互独立的
- **C** 在 **A** 或 **B** 启动后启动
- **D** 在 **C** 启动后启动
- **D** 启动后，**E** 和 **F** 的启动是相互独立的

如果设置了 **symmetrical=false**，则停止的顺序不受这个约束的影响。

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

13.4. 为不由 PACEMAKER 管理的资源依赖项配置启动顺序

集群可能包含不是由集群管理的依赖项的资源。在这种情况下，您必须确保在 Pacemaker 停止后启动这些依赖项，然后才能停止 Pacemaker。

您可以通过 **systemd resource-agents-deps** target 配置您的启动顺序来应对这种情况。您可以为此目标创建一个 **systemd** drop-in 单元，Pacemaker 会根据这个目标自行排序。

例如，如果集群包含依赖于集群管理的外部服务 **foo** 的资源，请执行以下步骤。

1. 创建 drop-in 单元 **/etc/systemd/system/resource-agents-deps.target.d/foo.conf**，其中包含以下内容：

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

2. 运行 **systemctl daemon-reload** 命令。

用这种方法指定的集群依赖项可以是服务以外的其它依赖项。例如：您在 **/srv** 中挂载一个文件系统可能会需要依赖项，这时您可执行以下步骤：

1. 确定 **/srv** 列在 **/etc/fstab** 文件中。这样，当重新载入系统管理器的配置时，会在启动时自动转换为 **systemd** 文件 **srv.mount**。如需更多信息，请参阅 **systemd.mount**(5)和 **systemd-fstab-generator**(8)man page。

2. 要确保 Pacemaker 在挂载磁盘后启动，请创建 drop-in 单元 **/etc/systemd/system/resource-agents-deps.target.d/srv.conf**，其中包含以下内容：

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

3. 运行 **systemctl daemon-reload** 命令。

如果 Pacemaker 集群使用的 LVM 卷组包含位于远程块存储上的一个或多个物理卷，如 iSCSI 目标，则您可以为目标配置 **systemd resource-agents-deps** 目标和一个 **systemd** 置入单元，以确保服务在 Pacemaker 启动之前启动。

以下流程将 **blk-availability.service** 配置为依赖项。**blk-availability.service** 服务是一个包装程序，它包含 **iscsi.service** 以及其他服务。如果您的部署需要它，则您可以将 **iscsi.service**（仅用于 iSCSI）或 **remote-fs.target** 配置为依赖项，而不是 **blk-availability**。

1. 创建包含以下内容的置入单元 **/etc/systemd/system/resource-agents-deps.target.d/blk-availability.conf**：

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

2. 运行 **systemctl daemon-reload** 命令。

第 14 章 COLOCATING 集群资源

要指定一个资源的位置取决于另一个资源的位置，您需要配置 colocation 约束。

在两个资源间创建 colocation 约束具有重要的副作用：它会影响分配给节点资源的顺序。这是因为您无法相对于资源 B 来放置资源 A，除非您知道资源 B 的位置。因此，当创建 colocation 约束时，您必须考虑是将资源 A 与资源 B 共处，还是将资源 B 与资源 A 共处。

在创建 colocation 约束时要记住的是，假设资源 A 与资源 B 在一起，在决定哪个节点要选择资源 B 时，集群也会考虑资源 A 的首选项。

以下命令创建了 colocation 约束。

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

下表总结了配置 colocation 约束的属性和选项。

表 14.1. Colocation 约束的参数

参数	描述
source_resource	共处源。如果约束不满意，集群可能决定完全不允许该资源运行。
target_resource	共处目标。集群将决定优先放置此资源的位置，然后决定放置源资源的位置。
分数	正数值表示资源应该在同一个节点上运行。负值表示资源不应在同一节点上运行。+**INFINITY** 值（默认值）表示 *source_resource* 必须在与 *target_resource* 运行在相同的节点上。-**INFINITY** 值表示 *source_resource* 不得在与 *target_resource* 运行在相同的节点上。
influence 选项	确定当依赖资源达到其迁移失败阈值时，集群是否同时将主资源 (*source_resource*) 和依赖资源 (*target_resource*) 移到另一节点，或者集群允许依赖资源离线而不造成服务切换。 **influence** colocation 约束选项的值为 **true** 或 **false**。此选项的默认值由依赖资源的 **critical** 资源 meta 选项的值确定，其默认值为 **true**。 当这个选项的值为 **true** 时，Pacemaker 会同时尝试保持主资源和依赖的资源活跃状态。如果依赖资源达到迁移失败的阈值，则两个资源都会移到另一个节点。 当这个选项的值为 **false** 时，Pacemaker 会避免因为依赖资源的状态而移动主资源。在这种情况下，如果依赖资源达到失败的迁移阈值，则当主资源活跃并可以保持在当前节点上时，它将停止。

14.1. 指定资源的强制放置

每当约束的 score 为 **+INFINITY** 或 **-INFINITY** 时，就会发生强制放置。在这种情况下，如果约束无法满足，则不允许 *source_resource* 运行。对于 **score=INFINITY**，这包括 *target_resource* 未处于活动状态的情况。

如果您需要 **myresource1** 始终与 **myresource2** 运行在同一台机器上，则您可以添加以下约束：

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

由于使用了 **INFINITY**，如果 **myresource2** 无法在任何一个集群节点上运行（出于某种原因），则将不允许 **myresource1** 运行。

或者，您可能想要配置相反的情况，在集群中，**myresource1** 不能与 **myresource2** 运行在同一台机器上。在这种情况下，请使用 **score=-INFINITY**

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

同样，通过指定 **-INFINITY**，约束是绑定的。因此，如果只剩下 **myresource2** 所在的位置可以运行，那么 **myresource1** 可能无法在任何地方运行。

14.2. 指定资源的公告放置

资源的公告放置表示资源的放置是首选项，但并不是强制的。对于 score 大于 **-INFINITY** 且小于 **INFINITY** 的约束，集群将尝试满足您的要求，但如果替代方案是停止某些集群资源，则可能会忽略它们。

14.3. 资源共存集合

如果您的配置需要您创建一组资源，它们按顺序共处并启动，您可以配置包含这些资源的资源组。然而，在有些情况下，配置需要作为资源组共存的资源是不合适的：

- 您可能需要托管一组资源，但这些资源不一定要按顺序启动。
- 您可能有一个资源 C，它必须和资源 A 或 B 在一起，但 A 和 B 之间没有关系。
- 您可能有资源 C 和 D 必须和资源 A 和 B 在一起，但 A 和 B 之间没有关系，C 和 D 之间没有关系。

在这些情况下，您可以使用 **pcs constraint colocation set** 命令对一组或几组资源创建共处约束。

您可以使用 **pcs constraint colocation set** 命令为一组资源设置以下选项。

- **sequential**，它可以设为 **true** 或 **false**，以指示集合成员是否必须相互共处。
  将 **sequential** 设为 **false**，允许此集合的成员与约束后面列出的另一个集合共处，而无论此集合中哪些成员处于活动状态。因此，只有在约束里列出另一个集合之后，这个选项才有意义，否则约束无效。
- **role**，可以被设置为 **Stopped**, **Started**, **Promoted**, 或 **Unpromoted**。

您可以按照 **pcs constraint colocation set** 命令的 **setoptions** 参数为一组资源设置以下约束选项。

- **ID**，为您定义的约束提供名称：
- **score** 表示此约束的首选程度。有关这个选项的详情，请参考 配置位置约束中的 "Location Constraint Options" 表

当列出集合的成员时，每个成员都与其前一个成员共处。例如："set A B" 表示 "B 与 A 共存"。但是，当列出多个集合时，每个集合都与后面的组在一起。例如："set C D sequential=false set A B" 表示 "set C D（其中 C 和 D 间没有关系）与 set A B 在一起（其中 B 与 A 在一起）"。

以下命令在一组或一组资源上创建了 colocation 约束。

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

使用以下命令删除带有 *source_resource* 的共处约束。

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

第 15 章 显示资源限制和资源依赖项

您可以使用一些命令来显示已经配置的约束。您可以显示所有配置的资源约束，也可以将资源约束的显示限制为特定类型的资源限制。另外，您还可以显示配置的资源依赖项。

显示所有配置的限制

以下命令列出所有当前位置、顺序和 colocation 约束。如果指定了 **--full** 选项，则显示内部约束 ID。

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

默认情况下，列出资源约束不会显示过期的限制。要在列表中包含已过期的 constaints，请使用 **pcs constraint** 命令的 **--all** 选项。这将列出已过期的限制，请注意显示中的限制及其关联的规则为 **(expired)**。

显示位置限制

以下命令列出所有当前位置限制。

- 如果指定了 **resources**，则会按资源显示位置约束。这是默认的行为。
- 如果指定了 **nodes**，则按节点显示位置约束。
- 如果指定了特定资源或节点，则只显示那些资源或节点的信息。

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

显示排序限制

以下命令列出所有当前排序限制。

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

显示 colocation 约束

以下命令列出所有当前的 colocation 约束。

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

显示特定于资源的约束

以下命令列出引用特定资源的约束。

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

显示资源依赖项

以下命令显示树结构中集群资源间的关系。

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

如果使用 **--full** 选项，命令会显示附加信息，包括约束 ID 和资源类型。

在以下示例中，有 3 个配置的资源：C、D 和 E。

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

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

在以下示例中，有 2 个配置的资源：A 和 B。资源 A 和 B 是资源组 G 的一部分。

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

第 16 章 使用规则决定资源位置

对于更复杂的位置限制，您可以使用 Pacemaker 规则来确定资源的位置。

16.1. PACEMAKER 规则

Pacemaker 规则可用于使您的配置更动态。规则的一个用法可能是根据时间将机器分配给不同的处理组（使用 node 属性），然后在创建位置约束时使用该属性。

每个规则都可以包含多个表达式、日期表达式甚至其它规则。表达式的结果根据规则的 **boolean-op** 字段来合并，以确定规则是否最终评估为 **true** 或 **false**。接下来的操作要看规则使用的上下文而定。

表 16.1. 规则的属性

项	描述
role	只有在资源位于该角色时才会应用该规则。允许的值：**Started**, **Unpromoted,** 和 **Promoted**。注：具有 **role="Promoted"** 的规则无法决定克隆实例的初始位置。它只会影响哪些活跃的实例将会被提升。
分数	规则评估为 **true** 时要应用的 score 。仅限于作为位置约束一部分的规则使用。
score-attribute	如果规则评估为 **true**，则要查找并用作 score 的 node 属性。仅限于作为位置约束一部分的规则使用。
boolean-op	如何组合多个表达式对象的结果。允许的值： **and** 和 **or**。默认值为 **and**。

16.1.1. 节点属性表达式

节点属性表达式用于根据节点或节点定义的属性控制资源。

表 16.2. 表达式的属性

项	描述
attribute	要测试的节点属性
type	决定值应该如何进行测试。允许的值：**string**, **integer**, **number**, **version**。默认值为 **string**。

项	描述
operation	执行的对比。允许的值： * **lt** - 如果节点属性的值比 **value** 小，则为 True * **gt** - 如果节点属性的值比 **value** 大，则为 True * **lte** - 如果节点属性的值小于或等于 **value**，则为 True * **gte** - 如果节点属性的值大于或等于 **value**，则为 True * **eq** - 如果节点属性的值等于 **value**，则为 True * **ne** - 如果节点属性的值不等于 **value**，则为 True * **defined** - 如果节点具有命名属性，则为 True * **not_defined** - 如果节点没有命名属性，则为 True
值	用户提供要比较的值（必需，除非 **operation** 为 **defined** 或 **not_defined**）

除了管理员添加的任何属性外，集群还会为每个节点定义特殊的内置节点属性，如下表中所述。

表 16.3. 内置节点属性

名称	描述
#uname	节点名
#id	节点 ID
#kind	节点类型。可能的值有 **cluster** 、**remote** 和 **container**。对于使用 **ocf:pacemaker:remote** 资源创建的 Pacemaker 远程节点，**kind** 的值为 **remote** ，对于 Pacemaker 远程客户机和捆绑节点，其值为 **container**。
#is_dc	如果此节点是指定控制器(DC)，则为 **true**，否则为 **false**
#cluster_name	**cluster-name** 集群属性的值（如果设置了）
#site_name	**site-name** 节点属性的值（如果设置了），否则与 **#cluster-name** 相同
#role	此节点上相关的可远程克隆的角色。仅在可转发克隆的位置约束的规则内有效。

名称	描述

16.1.2. 基于时间/日期的表达式

日期表达式用于根据当前的日期/时间控制资源或集群选项。它们可以包含可选的日期规格。

表 16.4. 日期表达式的属性

项	描述
开始	符合 ISO8601 规范的日期/时间。
end	符合 ISO8601 规范的日期/时间。
operation	根据上下文，将当前日期/时间与开始或结束日期进行比较。允许的值： * **gt** - 如果当前日期/时间晚于 **start**，则为 True * **lt** - 如果当前日期/时间早于 **end**，则为 True * **in_range** - 如果当前日期/时间在 **start** 后，在 **end** 前，则为 True * **date-spec** - 对当前日期/时间执行类似于 cron 的比较

16.1.3. 日期规格

日期规格用于创建与时间相关的类似 cron 的表达式。每个字段可以包含一个数字或一个范围。任何未提供的字段都会被忽略，而不是使用默认值 0。

例如，**monthdays="1"** 匹配每月的第一天，**hours="09-17"** 匹配上午 9 点到下午 5 点（包含）之间的小时数。但是，您不能指定 **weekdays="1,2"** 或 **weekdays="1-2,5-6"**，因为它们包含多个范围。

表 16.5. 日期规格的属性

项	描述
id	日期的唯一名称
hours	允许的值：0-23
monthdays	允许的值：0-31（取决于月份和年）
weekdays	允许的值：1-7（1代表星期一，7 代表星期日）

项	描述
yeardays	允许的值：1-366（根据年而定）
months	允许的值：1-12
weeks	允许的值：1-53（取决于 **weekyear**）
years	根据 Gregorian 日历年
weekyears	可能不同于 Gregorian 年；例如，**2005-001 Ordinal** 也是 **2005-01-01 Gregorian**，也是 **2004-W53-6 Weekly**
moon	允许的值：0-7（0 是新月，4 为满月）。

16.2. 使用规则配置 PACEMAKER 位置约束

使用以下命令配置使用规则的 Pacemaker 约束。如果省略 **score**，则默认为 INFINITY。如果省略 **resource-discovery**，则默认为 **always**。

有关 **resource-discovery** 选项的信息，请参阅 将资源发现限制到节点的子集。

与基本位置限制一样，您也可以使用这些限制的资源使用正则表达式。

当使用规则来配置位置约束时，**score** 的值可以是正数，也可以是负数，正数表示"首选"，负数表示"避免"。

> pcs constraint location *rsc* rule [resource-discovery=*option*] [role=promoted|unpromoted] [score=*score* | score-attribute=*attribute*] *expression*

expression 选项可以是以下之一，其中 *duration_options* 和 *date_spec_options* 为 ：hours, monthdays, weekdays, yeardays, months, weeks, years, weekyears，如数据规格中的 "Properties of a Date Specification" 表中所述。

- **defined|not_defined *attribute***
- ***attribute* lt|gt|lte|gte|eq|ne [string|integer|number|version] *value***
- **date gt|lt *date***
- **date in_range *date* to *date***
- **date in_range *date* to duration *duration_options* …**
- **date-spec *date_spec_options***
- ***表达式* 和|或 *表达式***
- **（*表达式*）**

请注意，持续时间是通过计算方式为 **in_range** 操作指定结束的方法。例如，您可以指定 19 个月的持续时间。

下面的位置约束配置一个满足以下位置的表达式（如果现在是 2018 年）。

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

以下命令配置一个周一到周五从上午 9 点下午 5 点为 true 的表达式。请注意，小时值为 16 可以匹配到 16:59:59，因为小时数仍然匹配。

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

下面的命令配置一个表达式，当周五且为 13 号并为一个满月时，这个表达式为 true。

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

要删除某个规则，使用以下命令：如果您要删除的规则是其约束中的最后一规则，则约束将被删除。

```
pcs constraint rule remove rule_id
```

第 17 章 管理集群资源

您可以使用各种命令来显示、修改和管理集群资源。

17.1. 显示配置的资源

要显示所有配置的资源列表，使用以下命令。

```
pcs resource status
```

例如，如果您的系统配置了名为 **VirtualIP** 的资源和名为 **WebSite** 的资源，则 **pcs resource show** 命令将产生以下输出：

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

要显示资源配置的参数，请使用以下命令。

```
pcs resource config resource_id
```

例如，以下命令显示了资源 **VirtualIP** 当前配置的参数。

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

要显示单个资源的状态，请使用以下命令：

```
pcs resource status resource_id
```

例如，如果系统配置了名为 **VirtualIP** 的资源，**pcs resource status VirtualIP** 命令会生成以下输出：

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

要显示特定节点上运行的资源的状态，请使用以下命令：您可以使用此命令显示集群和远程节点上资源的状态。

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

例如，如果 **node-01** 正在运行名为 **VirtualIP** 的资源，并且 **WebSite** 使用 **pcs resource status node=node-01** 命令可能会生成以下输出：

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

17.2. 使用 PCS 命令导出集群资源

从 Red Hat Enterprise Linux 9.1 开始，您可以显示 **pcs** 命令，该命令可用于使用 **pcs resource config** 命令的 **--output-format=cmd** 选项在不同的系统上重新创建配置的集群资源。

以下命令为红帽高可用性集群中的主动/被动 Apache HTTP 服务器创建四个资源：**LVM-activate** 资源、**Filesystem** 资源、**IPaddr2** 资源和 **Apache** 资源。

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

创建资源后，以下命令会显示可用来在不同系统上重新创建这些资源的 **pcs** 命令。

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

要显示 **pcs** 命令或您可以用来只重新创建一个配置资源的命令，请指定该资源的资源 ID。

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

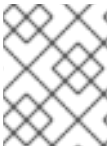
17.3. 修改资源参数

要修改配置的资源的参数，请使用以下命令：

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

以下命令序列显示了资源 **VirtualIP** 配置的参数的初始值、更改 **ip** 参数值的命令，以及 update 命令后的值。

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



注意

当您使用 **pcs resource update** 命令更新资源操作时，您没有专门调用的任何选项都将重置为它们的默认值。

17.4. 清除集群资源的失败状态

如果资源失败，则使用 **pcs status** 命令显示集群状态时会出现失败信息。尝试解决失败原因后，您可以通过再次运行 **pcs status** 命令检查资源的更新状态，您可以使用 **pcs resource failcount show --full** 命令检查集群资源的失败数。

您可以使用 **pcs resource cleanup** 命令清除资源的失败状态。**pcs resource cleanup** 命令重置资源状态和资源的 **failcount** 值。此命令还会删除资源的操作历史记录，并重新检测其当前状态。

以下命令重置 *resource_id* 指定的资源的状态和 **failcount** 值。

```
pcs resource cleanup resource_id
```

如果没有指定 *resource_id*，**pcs resource cleanup** 命令会为所有具有失败计数的资源重置资源状态和 **failcount** 值。

除了 **pcs resource cleanup *resource_id*** 命令外，您还可以重置资源状态，并使用 **pcs resource refresh *resource_id*** 命令清除资源的操作历史记录。与 **pcs resource cleanup** 命令一样，您可以运行没有指定选项的 **pcs resource refresh** 命令，来为所有资源重置资源状态和 **failcount** 值。

pcs resource cleanup 和 **pcs resource refresh** 命令都清除资源的操作历史记录，并重新检测资源的当前状态。**pcs resource cleanup** 命令仅对集群状态中显示为失败操作的资源进行操作，而 **pcs resource refresh** 命令则会对资源进行操作，而无论其当前状态如何。

17.5. 在集群中移动资源

Pacemaker 提供了各种机制来将资源配置为从一个节点迁移到另一个节点，并在需要时手动移动资源。

您可以使用 **pcs resource move** 和 **pcs resource relocate** 命令手动移动集群中的资源，如手动移动群集资源中所述。除了这些命令外，您还可以通过启用、禁用和禁止资源来控制集群资源的行为，如启用、禁用和禁止集群资源中所述。

您可以配置资源以便在定义多个故障后移到新节点，您可以在外部连接丢失时配置集群来移动资源。

17.5.1. 因为失败而移动资源

当您创建资源时，您可以通过为该资源设置 **migration-threshold** 选项来配置资源，使其在达到定义的故障次数后移至新节点。达到阈值后，这个节点将不再被允许运行失败的资源，直到：

- 已达到资源的 **failure-timeout** 值。
- 管理员使用 **pcs resource cleanup** 命令手动重置资源的失败计数。

migration-threshold 的值默认设置为 **INFINITY**。**INFINITY** 在内部被定义为一个非常大的数，但是有限的。值 0 会禁用 **migration-threshold** 功能。

注意

为资源设置 **migration-threshold** 与为迁移配置资源不同，其中资源移动可以到另一个位置，而不会丢失状态。

以下示例为名为 **dummy_resource** 的资源添加了一个迁移阈值 10，这表示资源将在 10 次故障后移到一个新节点。

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

您可以使用以下命令为整个集群的默认值添加迁移阈值。

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

要确定资源当前的故障状态和限值，请使用 **pcs resource failcount show** 命令。

迁移阈值概念有两个例外，当资源无法启动或无法停止时会出现这种情况。如果集群属性 **start-failure-is-fatal** 被设为 **true**（默认值），启动失败会导致 **failcount** 被设为 **INFINITY**，并总是导致资源被立即移动。

停止失败会稍有不同，且非常关键。如果资源无法停止且启用了 STONITH，则集群将隔离该节点以便可在其他位置启动该资源。如果没有启用 STONITH，那么集群就无法继续，也不会尝试在其他位置启动资源，而是会在失败超时后尝试再次停止它。

17.5.2. 由于连接更改而移动资源

将集群设置为在外部连接丢失时移动资源分为两个步骤。

1. 在集群中添加 **ping** 资源。**ping** 资源使用同名的系统工具来测试是否可以访问（由 DNS 主机名或 IPv4/IPv6 地址指定）一系列机器，并使用结果来维护名为 **pingd** 的节点属性。
2. 为资源配置位置约束，该限制将在连接丢失时将资源移动到不同的节点。

下表描述了您可以为 **ping** 资源设置的属性。

表 17.1. ping 资源的属性

项	描述
dampen	等待（强化）时间进一步发生更改。这会防止，当集群节点在稍有不同的时间发现连接丢失时资源在集群中移动。
multiplier	连接的 ping 节点数量乘以这个值来获得分数。在配置了多个 ping 节点时很有用。
host_list	要联系的机器决定当前的连接状态。允许的值包括可解析 DNS 主机名、IPv4 和 IPv6 地址。主机列表中的条目是空格分开的。

以下示例命令会创建一个 **ping** 资源，来验证与 **gateway.example.com** 的连接。在实践中，您可以验证到网络网关/路由器的连接。您可以将 **ping** 资源配置为克隆，以便资源可以在所有集群节点上运行。

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

以下示例为名为 **Webserver** 的现有资源配置位置约束规则。如果当前运行的主机无法 ping **gateway.example.com**，这将导致 **Webserver** 资源移至能够 ping **gateway.example.com** 的主机。

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

17.6. 禁用 MONITOR 操作

停止重复 monitor 的最简单方法是删除它。然而，在有些情况下，您可能只想临时禁用它。在这种情况下，将 **enabled="false"** 添加到操作的定义中。当您想重新使用 monitor 操作时，将 **enabled="true"** 设置为操作的定义。

当您使用 **pcs resource update** 命令更新资源操作时，您没有专门调用的任何选项都将重置为它们的默认值。例如，如果您已经配置了自定义超时值为 600 的监控操作，运行以下命令会将超时值重置为默认值 20（或通过 **pcs resource op defaults** 命令设置的任何默认值）。

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

为了保持这个选项的原始值 600，当您重新启用 monitor 控操作时，必须指定那个值，如下例所示。

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

17.7. 配置和管理集群资源标签

您可以使用 **pcs** 命令标记集群资源。这允许您使用单个命令启用、禁用、管理或取消管理指定的一组资源。

17.7.1. 为管理标记集群资源，按类别标记

以下流程使用资源标签标记两个资源，并禁用标记的资源。在本例中，要标记的现有资源命名为 **d-01** 和 **d-02**。

步骤

1. 为 **d-01** 和 **d-02** 资源创建一个名为 **special-resources** 的标签。

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

2. 显示资源标签配置。

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

3. 禁用带有 **special-resources** 标签的所有资源。

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

4. 显示资源状态，以确认资源 **d-01** 和 **d-02** 已被禁用。

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

除了 **pcs resource disable** 命令，**pcs resource enable**、**pcs resource manage** 和 **pcs resource unmanage** 命令还支持对带标记资源的管理。

创建资源标签后：

- 您可以使用 **pcs tag delete** 命令删除资源标签。
- 您可以使用 **pcs tag update** 命令修改现有资源标签的资源标签配置。

17.7.2. 删除标记的集群资源

您不能使用 **pcs** 命令删除标记的群集资源。要删除标记的资源，请使用以下步骤。

步骤

1. 删除资源标签。

   a. 以下命令从具有该标签的所有资源中删除资源标签 **special-resources**，

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

   b. 以下命令仅从资源 **d-01** 中删除资源标签 **special-resources**。

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

2. 删除资源。

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

第 18 章 创建在多个节点上活跃的集群资源（克隆的资源）

您可以克隆集群资源，以便在多个节点上激活该资源。例如，您可以使用克隆的资源配置 IP 资源的多个实例来分布到群集中以进行节点均衡。您可以克隆资源代理支持的任何资源。克隆由一个资源或一个资源组组成。

注意

只有同时可在多个节点上活跃的资源才适用于克隆。例如：从挂载非集群文件系统（如共享内存设备的 **ext4** ）的 **Filesystem** 资源不应被克隆。由于 **ext4** 分区不支持集群，因此此文件系统不适用于同时发生在多个节点上的读写操作。

18.1. 创建和删除克隆的资源

可以同时创建资源以及该资源的克隆。

通过以下单一命令创建资源并克隆资源：

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

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

默认情况下，克隆的名称将是 ***resource_id*-clone**。您可以通过为 *clone_id* 选项指定值来为克隆设置自定义名称。

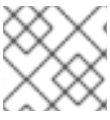
您不能在单个命令中创建资源组以及该资源组的克隆。

另外，您可以使用以下命令创建之前创建的资源或资源组的克隆。

pcs resource clone *resource_id* | *group_id* [*clone_id*][*clone options*]...

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

默认情况下，克隆的名称将是 ***resource_id*-clone** 或 ***group_name*-clone**。您可以通过为 *clone_id* 选项指定值来为克隆设置自定义名称。



注意

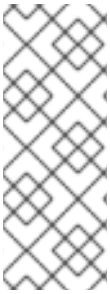
您只需要在一个节点中配置资源配置更改。

注意

在配置限制时，始终使用组或克隆的名称。

当您创建资源的克隆时，默认情况下，克隆将使用资源名称，并在名称后附加 **-clone** 。以下命令创建一个名为 **webfarm** 、类型为 **apache** 的资源，以及名为 **webfarm-clone** 资源的克隆。

 # pcs resource create webfarm apache clone



注意

当您创建一个资源或资源组克隆，其将在另一个克隆后排序时，您几乎应该始终设置 **interleave=true** 选项。这样可保证当依赖克隆的克隆停止或启动时，依赖克隆的副本可以停止或启动。如果没有设置这个选项，克隆的资源 B 依赖于克隆的资源 A，且节点离开集群，当节点返回到集群并在该节点上启动资源 A，那么所有节点上的资源 B 的副本都将会重启。这是因为，当依赖的克隆资源没有设置 **interleave** 选项时，该资源的所有实例都依赖于它所依赖的资源的任何正在运行的实例。

使用以下命令删除资源或资源组的克隆。这不会删除资源或资源组本身。

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

下表描述了您可以为克隆资源指定的选项。

表 18.1. 资源克隆选项

项	描述
priority, target-role, is-managed	选项继承自正在克隆的资源，如配置资源 meta 选项中的" 资源元数据"表中所述。
clone-max	要启动的资源副本数量。默认为集群中的节点数量。
clone-node-max	在一个节点上可以启动资源的副本数 ; 默认值为 **1**。
notify	当停止或启动克隆的副本时，预先并在操作成功时告知所有其他副本。允许的值： **false**，**true**.默认值为：**false**。
globally-unique	克隆的每个副本是否会执行不同的功能？允许的值：**false**,**true** 如果此选项的值为 **false**，则这些资源在任何位置的行为都相同，因此每台机器只能有一个克隆活跃副本。 如果此选项的值为 **true**，则运行在一台机器上的克隆的副本并不等于另一个实例，无论该实例是运行在另一个节点上还是运行在同一节点上。如果 **clone-node-max** 的值大于 1，则默认值为 **true** ；否则默认值为 **false**。
排序的	是否应该以系列的方式启动副本（而不是并行的）。允许的值： **false**，**true**.默认值为：**false**。
interleave	更改排序限制的行为（克隆之间）的行为，以便在相同节点中的同一节点中的副本立即启动或停止（而不是等到第二个克隆的每个实例启动或停止）。允许的值： **false**，**true**.默认值为：**false**。

项	描述
clone-min	如果指定了值，则在此克隆后排序的任何克隆都将无法启动，直到原始克隆指定数量的实例都在运行，即使 **interleave** 选项设为 **true**。

要实现稳定的分配模式，默认情况下克隆具有稍微的粘贴性，这意味着它们更喜欢保留在运行的节点中。如果未提供 **resource-stickiness** 值，克隆将使用值 1。作为一个小的值，它会对其他资源分数计算最小，但足以防止 Pacemaker 在集群间不必要地移动副本。有关设置 **resource-stickiness** 资源 meta-option 的详情，请参考 配置资源 meta 选项。

18.2. 配置克隆资源限制

在大多数情况下，克隆将在每个活跃集群节点上都有一个副本。但是，您可以将资源克隆的 **clone-max** 设置为一个小于集群中节点总数的值。如果情况如此，您可以指定集群使用资源位置约束来优先分配哪些节点。这些限制与用于常规资源的条件相同，除了必须使用克隆的 id 以外。

以下命令为集群创建了一个位置约束，以优先将资源克隆 **webfarm-clone** 分配给 **node1**。

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

排序限制对克隆的行为稍有不同。在下例中，由于 **interleave** 克隆选项保留为 **false**，因此在启动需要启动的所有 **webfarm-clone** 的实例之前，不会启动任何 **webfarm-stats** 的实例。只有任何 **webfarm-clone** 的副本都无法启动时，才会阻止 **webfarm-stats** 处于活动状态。此外，**webfarm-clone** 在停止其自身之前，将等待 **webfarm-stats** 停止。

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

将常规（或组）资源与克隆在一起，意味着该资源可在任何有克隆活跃副本的机器中运行。集群将根据克隆运行情况以及资源自己的位置首选项选择一个副本。

克隆之间的并发位置也是有可能的。在这种情况下，克隆允许的位置集合仅限于克隆要激活的节点。然后分配可以正常执行。

以下命令创建了一个共处约束，以确保资源 **webfarm-stats** 与 **webfarm-clone** 的活动副本运行在同一节点上。

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

18.3. 可升级克隆资源

可升级克隆资源是将 **promotable** meta 属性设置为 **true** 的克隆资源。它们允许实例处于两种工作模式之一；称为 **promoted** 和 **unpromoted**。模式的名称没有特定的含义，除了一个限制，即实例启动时，它必须处于 **Unpromoted** 状态。备注：Promoted 和 Unpromoted 角色名称的功能等同于之前的 RHEL 版本中 Master 和 Slave Pacemaker 角色。

18.3.1. 创建可升级的克隆资源

您可以用下列单一命令将资源创建为可升级的克隆。

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

默认情况下，可升级克隆的名称为 ***resource_id*-clone**。

您可以通过为 *clone_id* 选项指定值来为克隆设置自定义名称。

另外，您可以使用以下命令从之前创建的资源或资源组中创建可升级的资源。

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

默认情况下，可升级克隆的名称为 ***resource_id*-clone** 或 ***group_name*-clone**。

您可以通过为 *clone_id* 选项指定值来为克隆设置自定义名称。

下表描述了您可以为可升级资源指定的额外克隆选项。

表 18.2. 为可升级克隆提供了额外的克隆选项

项	描述
promoted-max	可以升级的资源副本数；默认为 1。
promoted-node-max	在一个节点中可升级的资源副本数；默认为 1。

18.3.2. 配置可升级资源限制

在大多数情况下，可升级的资源在每个活跃的集群节点上都有一个副本。如果情况不同，您可以指定集群使用资源位置约束来优先分配哪些节点。这些限制与常规资源的写法不同。

您可以创建一个 colocation 约束，指定资源是否在升级或未升级的角色中运行。以下命令创建了资源 colocation 约束。

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

有关托管约束的详情，请参考 托管集群资源。

在配置包含可升级资源的排序约束时，您可以为资源指定的其中一个操作是 **promote** 的，这表示资源从未升级的角色提升为提升角色。另外，您可以指定 **demote** 操作，这表示资源从提升角色降级为未升级的角色。

配置顺序约束的命令如下。

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

有关资源顺序约束的详情，请参阅 确定集群资源的运行顺序。

18.4. 失败时降级升级的资源

您可以配置可升级资源，以便在 **promote** 和 **monitor** 操作失败时，或者该资源运行的分区丢失仲裁（quorum）时，资源将会降级，但不会完全停止。这可避免在完全停止资源时需要的人工干预。

- 要将可升级资源配置为在 **promote** 操作失败时降级，请将 **on-fail** 操作 meta 选项设置为 **demote**，如下例所示。

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

- 要将可升级的资源配置为在 **monitor** 操作失败时降级，将 **interval** 设置为非零值，将 **on-fail** 操作 meta 选项设置为 **demote**，并将 **role** 设置为 **Promoted**，如下例所示。

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

- 要配置集群，当集群分区丢失仲裁时，任何升级的资源都会降级但会继续运行，所有其他资源都将停止，将 **no-quorum-policy** 集群属性设置为 **demote**

将操作的 **on-fail** meta-attribute 设置为 **demote** 不会影响如何确定资源的提升。如果受影响的节点仍然具有最高的升级分数，则会选择再次提升。

第 19 章 管理集群节点

您可以使用各种 **pcs** 命令来管理群集节点，包括启动和停止集群服务以及添加和删除集群节点的命令。

19.1. 停止集群服务

以下命令在指定的一个节点或几个节点上停止集群服务。与 **pcs cluster start** 一样，**--all** 选项会停止所有节点上的集群服务，如果没有指定任何节点，则只停止本地节点上的集群服务。

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

您可以使用以下命令强制停止本地节点上的集群服务，该命令会执行 **kill -9** 命令。

```
pcs cluster kill
```

19.2. 启用和禁用集群服务

使用以下命令启用集群服务。这会将集群服务配置为在指定的节点上启动时运行。

启用允许节点在集集被隔离后自动重新加入集群，从而减少集群性能小于满额性能的时间。如果没有启用集群服务，管理员可以在手动启动集群服务前手动调查出了什么问题，例如：当有硬件问题的节点可能会再次失败时无法重新访问该集群。

- 如果指定了 **--all** 选项，该命令将启用所有节点上的集群服务。
- 如果您没有指定任何节点，则仅在本地节点上启用集群服务。

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

使用以下命令将集群服务配置为在指定的一个节点或几个节点启动时不运行。

- 如果指定了 **--all** 选项，该命令将禁用所有节点上的集群服务。
- 如果没有指定任何节点，则仅在本地节点上禁用集群服务。

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

19.3. 添加集群节点

使用以下流程向现有集群添加新节点。

这个过程添加了运行 **corosync** 的标准集群节点。有关将非 corosync 节点整合到集群中的详情，请参考 将非 corosync 节点整合到集群中：pacemaker_remote 服务。

注意

建议您仅在生产环境维护窗口期间将节点添加到现有集群中。这可让您对新节点及其保护配置执行适当的资源和部署测试。

在本例中，现有的集群节点为 **clusternode-01.example.com**、**clusternode-02.example.com** 和 **clusternode-03.example.com**。新节点为 **newnode.example.com**。

步骤

在加入到集群中的新节点上，执行以下任务。

1. 安装集群软件包。如果集群使用 SBD、Booth 票据管理器或仲裁设备，则必须在新节点上手动安装相应的软件包（**sbd**、**booth-site**、**corosync-qdevice**）。

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

除了集群软件包外，还需要安装并配置在集群中运行的所有服务（已安装在现有集群节点上）。例如：如果您在红帽高可用性集群中运行 Apache HTTP 服务器，则需要在您要添加的节点中安装该服务器，以及检查服务器状态的 **wget** 工具。

2. 如果您正在运行 **firewalld** 守护进程，请执行以下命令启用红帽高可用性附加组件所需的端口。

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

3. 为用户 ID **hacluster** 设置密码。建议您为集群中的每个节点使用相同的密码。

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

4. 执行以下命令启动 **pcsd** 服务，并在系统启动时启用 **pcsd** ：

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

在现有集群中的一个节点上执行以下任务。

1. 在新集群节点上验证用户 **hacluster**。

```
[root@clusternode-01 ~]# pcs host auth newnode.example.com
Username: hacluster
Password:
newnode.example.com: Authorized
```

2. 在现有集群中添加新节点。此命令还会将集群配置文件 **corosync.conf** 同步到集群中的所有节点上，包括您添加的新节点。

```
[root@clusternode-01 ~]# pcs cluster node add newnode.example.com
```

在加入到集群中的新节点上，执行以下任务。

1. 在新节点上启动并启用集群服务。

```
[root@newnode ~]# pcs cluster start
Starting Cluster...
[root@newnode ~]# pcs cluster enable
```

2. 确保您为新集群节点配置并测试隔离设备。

19.4. 删除集群节点

以下命令关闭指定的节点，并将其从集群中其它节点上的配置文件 **corosync.conf** 中删除。

```
pcs cluster node remove node
```

19.5. 使用多个链接在集群中添加节点

当将节点添加到有多个链接的集群时，您必须为所有链接指定地址。

以下示例将节点 **rh80-node3** 添加到集群中，为第一个链接指定 IP 地址 192.168.122.203，第二个链接为 192.168.123.203。

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

19.6. 在现有集群中添加和修改链接

在大多数情况下，您可以在不重启集群的情况下在现有集群中添加或修改链接。

19.6.1. 在现有集群中添加和删除链接

要向正在运行的集群添加新链接，请使用 **pcs cluster link add** 命令。

- 在添加链接时，必须为每个节点指定一个地址。
- 只有在您使用 **knet** 传输协议时，才能添加和删除链接。
- 在任何时候，集群中至少都需要有一个链接被定义。
- 集群中的最多链接数量为 8 个，编号为 0-7。定义了哪些链接无关紧要，例如，您可以只定义链接 3、6 和 7。
- 在不指定链接号的情况下添加链接时，**pcs** 使用最低可用的链接。
- 当前配置的链接链接号包括在 **corosync.conf** 文件中。要显示 **corosync.conf** 文件，请运行 **pcs cluster corosync** 命令或 **pcs cluster config show** 命令。

以下命令将链接号 5 添加到三个节点集群中。

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

要删除现有链接，请使用 **pcs cluster link delete** 或 **pcs cluster link remove** 命令。下列指令之一可以将编号为 5 的链接删除。

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

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

19.6.2. 使用多个链接修改集群中的链接

如果集群中有多个链接，并且要更改其中的一个链接，请执行以下步骤。

步骤

1. 删除您要更改的链接。

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

2. 使用更新的地址和选项将链接重新添加到集群。

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

19.6.3. 使用单个链接修改集群中的链接地址

如果您的集群只使用一个链接，且您想要修改该链接以使用不同的地址，请执行以下步骤。在这个示例中，原始链接是链接 1。

1. 添加新地址和选项的链接。

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

2. 删除原始链接。

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

请注意，您不能指定在向集群添加链接时正被使用的地址。例如，这表示如果您有一个双节点集群，它有一个链接，而您希望只为一个节点更改地址，则无法使用上述流程添加指定一个新地址和一个现有地址的新链接。反之，您可以在删除现有链接并通过更新的地址将其重新添加前添加临时链接，如下例所示。

在本例中：

- 现有集群的链接为 link 1，节点 1 使用地址 10.0.5.11，节点 2 使用地址 10.0.5.12。
- 您要将节点 2 的地址改为 10.0.5.31。

步骤

要只为带有单一链接的双节点集群更新其中一个地址，请使用以下步骤。

1. 使用当前没有使用的地址为现有集群添加新临时链接。

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

2. 删除原始链接。

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

3. 添加新的修改的链接。

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

4. 删除您创建的临时链接

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

19.6.4. 使用单一链接修改集群中链接的选项

如果您的集群只使用一个链接，且您想要修改该链接的选项，但您不想更改要使用的地址，可以在删除和更新链接前添加临时链接以进行修改。

在本例中：

- 现有集群的链接为 link 1，节点 1 使用地址 10.0.5.11，节点 2 使用地址 10.0.5.12。
- 您要将链接选项 **link_priority** 更改为 11。

步骤

使用以下流程，在具有单链接的集群中修改 link 选项。

1. 使用当前没有使用的地址为现有集群添加新临时链接。

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

2. 删除原始链接。

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

3. 使用更新的选项为原始链接添加后端。

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

4. 删除临时链接。

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

19.6.5. 不可能在添加新链接时修改链接

如果由于某种原因在配置中不可能添加新链接，且您的唯一选项是修改单个现有链接，您可以使用以下步骤来关闭集群。

步骤

以下示例步骤更新集群中的链接 1，并将链接的 **link_priority** 选项设置为 11。

1. 停止集群的集群服务。

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

2. 更新链接地址和选项。
   pcs cluster link update 命令不要求您指定所有节点地址和选项。反之，您只能指定要更改的地址。这个示例只修改 **node1** 和 **node3** 的地址和 **link_priority** 选项。

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

要删除某个选项，您可以使用 ***option=*** 格式将选项设置为 null 值。

3. 重启集群

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

19.7. 配置节点健康策略

节点可能运行良好，足以维护其集群成员身份，但在某些情况下可能会处于不健康状态，使它成为资源不受欢迎的位置。例如，磁盘驱动器可能会报告 SMART 错误，或者 CPU 可能负载过高。从 RHEL 9.1 开始，您可以在 Pacemaker 中使用节点健康状况策略，以自动将资源从不健康节点移出。

您可以使用以下健康节点资源代理监控节点的健康状况，该代理根据 CPU 和磁盘状态设置节点属性：

- **ocf:pacemaker:HealthCPU**，其监控 CPU 空闲
- **ocf:pacemaker:HealthIOWait**，其监控 CPU I/O 等待
- **ocf:pacemaker:HealthSMART**，其监控磁盘驱动器的 SMART 状态
- **ocf:pacemaker:SysInfo** 其使用本地系统信息设置各种节点属性，并作为健康代理监控磁盘空间的使用情况

另外，任何资源代理都可能会提供可用于定义健康节点策略的节点属性。

步骤

以下流程为集群配置了健康节点策略，该策略会将资源从任何 CPU I/O 等待超过 15% 的节点移出。

1. 设置 **health-node-strategy** 集群属性，以定义 Pacemaker 如何对节点健康状况中的变化做出响应。

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

2. 创建使用健康节点资源代理的克隆的集群资源，设置 **allow-unhealthy-nodes** 资源元选项，以定义集群是否将检测节点的健康状态是否恢复了，并将资源移回该节点。使用重复的监控操作配置此资源，以持续检查所有节点的健康状况。
本例创建了一个 **HealthIOWait** 资源代理来监控 CPU I/O 等待，为将资源从节点移出的红色限制设置为 15%。此命令将 **allow-unhealthy-nodes** 资源元选项设为 **true**，并配置 10 秒的重复监控间隔。

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

19.8. 使用许多资源配置大型集群

如果要部署的集群由大量节点和许多资源组成，您可能需要为集群修改以下参数的默认值。

cluster-ipc-limit 集群属性

cluster-ipc-limit 集群属性是在一个集群守护进程断开连接前的最大 IPC 消息积压。当在大型集群中同时清理或修改大量资源时，大量 CIB 更新会一次性进行。如果 Pacemaker 服务没有时间处理 CIB 事件队列阈值前的所有配置更新，这可能会导致速度较慢的客户端被驱除。
在大型集群中推荐的 **cluster-ipc-limit** 值是集群中的资源数量乘以节点的数量。如果您在日志中看到"Evicting client"消息，则可能会引发该值。

您可以使用 **pcs property set** 命令从默认值 500 增加 **cluster-ipc-limit** 的值。例如，对于具有 200 个资源的十个节点集群，您可以使用以下命令将 **cluster-ipc-limit** 的值设置为 2000。

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

PCMK_ipc_buffer Pacemaker 参数

在非常大的部署中，内部 Pacemaker 消息可能会超过消息缓冲区的大小。当发生这种情况时，您会看到以下格式系统日志中的信息：

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

当您看到此消息时，您可以增加每个节点上的 **/etc/sysconfig/pacemaker** 配置文件中的 **PCMK_ipc_buffer** 的值。例如，要将 **PCMK_ipc_buffer** 的值从默认值提高为 13396332 字节，请按如下所示更改集群中每个节点的 **/etc/sysconfig/pacemaker** 文件中没有被注册掉的 **PCMK_ipc_buffer** 字段：

```
PCMK_ipc_buffer=13396332
```

要应用此更改，请运行以下命令：

```
# systemctl restart pacemaker
```

第 20 章 为 PACEMAKER 集群设置用户权限

您可以为用户 **hacluster** 以外的特定用户授予权限来管理 Pacemaker 集群。您可以为独立的用户授予两组权限：

- 允许个人用户通过 Web UI 管理集群的权限，并运行通过网络连接到节点的 **pcs** 命令。通过网络连接到节点的命令包括设置集群、从集群中添加或删除节点的命令。
- 本地用户允许只读或读写访问集群配置的权限。不需要通过网络连接的命令包括编辑集群配置的命令，比如那些创建资源和配置限制的命令。

当分配了两组权限时，首先应用通过网络连接的命令的权限，然后应用在本地节点中编辑集群配置的权限。大多数 **pcs** 命令不需要网络访问，在这种情况下，网络权限将不适用。

20.1. 设置通过网络访问节点的权限

要对特定用户授予权限来通过 Web UI 管理集群，并运行通过网络连接到节点的 **pcs** 命令，请将这些用户添加到组 **haclient** 中。这必须在集群的每个节点中完成。

20.2. 使用 ACL 设置本地权限

您可以使用 **pcs acl** 命令，为本地用户设置权限，以允许使用访问控制列表(ACL)对集群配置进行只读或读写访问。

默认情况下不启用 ACL。如果没有启用 ACL，则属于所有节点上的 **haclient** 组成员的任何用户都具有对集群配置的完整本地读/写权限，而不属于 **haclient** 成员的用户则无访问权限。当启用 ACL 时，即使属于 **haclient** 组成员的用户也只能访问 ACL 为该用户授予的内容。root 和 **hacluster** 用户帐户始终对集群配置有完全访问权限，即使启用了 ACL。

为本地用户设置权限分为两个步骤：

1. 执行 **pcs acl role create...** 创建一个 *role* 来定义该角色的权限。
2. 使用 **pcs acl user create** 命令将您创建的角色分配给用户。如果为同一用户分配多个角色，则任何 **deny** 权限优先于 **write**，然后 **read**。

步骤

以下示例流程为名为 **rouser** 的本地用户提供了对集群配置的只读权限。请注意，也可能限制对配置某些部分的访问。

警告

以根用户身份执行这个步骤或者保存所有配置更新到工作文件是很重要的，然后在完成后将其推送到活跃 CIB。否则，您可以锁定自己以阻止做任何进一步的更改。有关保存对工作文件的配置更新的详情，请参考为可工作的文件保存配置更改。

1. 此流程要求本地系统上 **rouser** 用户存在，并且 **rouser** 是组 **haclient** 的成员。

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

2. 使用 **pcs acl enable** 命令启用 Pacemaker ACL。

```
# pcs acl enable
```

3. 为 cib 创建名为 **read-only** 且具有只读权限的角色。

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

4. 在 pcs ACL 系统中创建用户 **rouser**，并为该用户分配 **read-only** 角色。

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

5. 查看当前的 ACL。

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

6. 在每个 **rouser** 将运行 **pcs** 命令的节点上，以 **rouser** 用户身份登录并进行本地 **pcsd** 服务身份验证。这是为了以 ACL 用户身份运行特定的 **pcs** 命令（如 **pcs status**）。

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

第 21 章 资源监控操作

为确保资源健康，您可以在资源的定义中添加监控操作。如果您没有对资源指定监控操作，默认情况下，**pcs** 命令将创建一个监控操作，间隔由资源代理决定。如果资源代理不提供默认的监控间隔，pcs 命令将创建监控操作，间隔为 60 秒。

下表总结了资源监控操作的属性。

表 21.1. 操作的属性

项	描述
id	操作的唯一名称。系统在配置操作时分配这个值。
名称	要执行的操作。常见值：**monitor**、**start**、**stop**
interval	如果设置为非零值，则会以这个频率（以秒为单位）重复操作。只有在操作 **name** 设为 **monitor** 时，非零值才有意义。资源启动后，将立即执行重复的 monitor 操作，并在上一个监控动作完成后调度后续的 monitor 操作。例如，如果 monitor 操作的 **interval=20s** 在 01:00:00 执行，则下一次 monitor 操作不会发生在 01:00:20，而是在第一个 monitor 操作完成后的 20 秒发生。 如果设置为零（默认值为零），则此参数允许您为集群创建的操作提供值。例如，如果 **interval** 设为零，则操作的 **name** 设置为 **start**，**timeout** 值设为 40，则 Pacemaker 在启动此资源时将使用 40 秒超时。带有零间隔的 **monitor** 操作允许您为 Pacemaker 在启动时的探测设置 **timeout**/**on-fail**/**enabled** 值，以便在不需要默认值时获取所有资源的当前状态。
timeout	如果在此参数设置的时间内操作没有完成，操作会被终止并认为它失败。默认值是使用 **pcs resource op defaults** 命令设置的 **timeout** 值，如果未设置，则为 20 秒。如果您发现您的系统所包含的资源比系统允许执行操作的时间更长（如 **start**、**stop** 或 **monitor**），请调查其原因，如果您预计需要较长的执行时间，则可以增加这个值。 **timeout** 值不是任何类型的延迟，如果操作在超时时间用完后返回，则集群也不会等待整个超时时间。

项	描述
on-fail	在这个操作失败时要执行的操作。允许的值： * **ignore** - 假装资源没有失败 * **block** - 不对资源执行任何进一步的操作 * **stop** - 停止资源，且不在其它地方启动它 * **restart** - 停止资源，并重新启动它（可能在不同的节点上） * **fence** - STONITH 资源失败的节点 * **standby** - 将 *所有* 资源从资源失败的节点移出 * **demote** - 当资源的 **promote** 操作失败时，该资源会被降级但不会被完全停止。当资源 **monitor** 操作失败时，如果 **interval** 设置为一个非零值，并且 **role** 设置为 **Promoted**，则资源将会降级但不会被完全停止。 当启用了 STONITH 时，**stop** 操作的默认值为 **fence**，否则为 **block**。所有其他操作默认为 **restart**。
enabled	如果为 **false**，则该操作被视为不存在。允许的值： **true**,**false**

21.1. 配置资源监控操作

您可以在使用以下命令创建资源时配置监控操作。

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

例如，以下命令创建了一个带有监控操作的 **IPaddr2** 资源：新资源称为 **VirtualIP**，**eth2** 的 IP 地址为 192.168.0.99，子网掩码为 24。每 30 秒将执行一次监控操作。

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

另外，您可以使用以下命令在现有资源中添加监控操作。

> pcs resource op add *resource_id operation_action* [*operation_properties*]

使用以下命令删除配置的资源操作。

> pcs resource op remove *resource_id operation_name operation_properties*

注意

您必须指定准确的操作属性才能正确地删除现有的操作。

要更改监控选项的值，您可以更新资源。例如，您可以使用以下命令创建 **VirtualIP** ：

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

默认情况下，这个命令会创建这些操作。

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

要改变停止超时操作，请执行以下命令。

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

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

21.2. 配置全局资源操作默认

您可以使用 **pcs resource op defaults update** 命令为所有资源更改资源操作的默认值。

以下命令为所有监控操作设置 **timeout** 值 240 秒。

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

原始的 **pcs resource defaults *name=value*** 命令（在以前的版本中为所有资源设置默认值）仍然被支持，除非设置了多组默认值。但是，**pcs resource op defaults update** 是该命令的首选版本。

21.2.1. 覆盖特定于资源的操作值

请注意，只有在集群资源定义中没有指定该选项时，集群资源才会使用全局默认值。默认情况下，资源代理为所有操作定义 **timeout** 选项。要接受全局操作超时值，您必须明确地创建没有 **timeout** 选项的集群资源，或者您必须通过更新集群资源来删除 **timeout** 选项，如以下命令所示。

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

例如，在为所有监控操作设置了一个 240 秒的 **timeout** 值，并更新集群资源 **VirtualIP** 来删除 **monitor** 操作的超时值后，资源 **VirtualIP** 的 **start**、**stop** 和 **monitor** 操作的超时值将分别为 20s、40s 和 240s。这里，超时操作的全局默认值仅应用于 **monitor** 操作，其中默认的 **timeout** 选项已被上一条命令删除。

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

21.2.2. 更改一组资源操作的默认值

您可以使用 **pcs resource op defaults set create** 命令创建多组资源操作默认值，该命令允许您指定包含 **resource** 和操作表达式的规则。Pacemaker 支持的所有规则表达式都被允许。

使用这个命令，您可以为特定类型的所有资源配置默认资源操作值。例如，在使用捆绑包时，可以配置 Pacemaker 创建的隐式 **podman** 资源。

以下命令为所有 **podman** 资源操作设定默认的超时值 90s。在本例中，**::podman** 是指类型为 **podman** 的任何类、任何供应商的资源。

id 选项用来命名资源操作默认选项，当并不强制使用。如果您没有设定此选项， **pcs** 则会自动生成 ID。设置这个值可让您提供更描述性的名称。

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

以下命令为所有资源的 **stop** 操作设置默认的超时值 120s。

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

对于特定类型的所有资源，可以为特定的操作设置默认值。以下示例为所有 **podman** 资源设置 **stop** 操作的默认超时值 120s。

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

21.2.3. 显示当前配置的资源操作默认值

pcs resource op defaults 命令显示目前配置的资源操作默认值列表，包括您指定的规则。

以下命令显示集群的默认操作值，为所有 **podman** 资源的所有操作都设置了默认超时值 90s，并为 ID 设置了一组资源操作默认值为 **podman-timeout**。

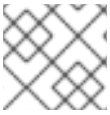
```
# pcs resource op defaults
Meta Attrs: podman-timeout
  timeout=90s
  Rule: boolean-op=and score=INFINITY
    Expression: resource ::podman
```

以下命令显示集群的默认操作值，为所有 **podman** 资源的 **stop** 操作都设置了默认超时值 120s，并为 ID 设置了一组资源操作默认值为 **podman-stop-timeout**。

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

21.3. 配置多个监控操作

您可以根据资源代理支持，使用多个监控操作配置单个资源。这样，您可以每分钟执行一次一般的健康检查，而以更高的间隔执行其他更大型的健康检查。



注意

当配置多个监控器操作时，您必须确保不会同时执行两个操作。

要为支持在不同级别上更深入检查的资源配置额外的监控操作，您需要添加 **OCF_CHECK_LEVEL=*n*** 选项。

例如，如果您配置了以下 **IPaddr2** 资源，默认情况下，这会创建一个监控操作，间隔为 10 秒，超时值为 20 秒。

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

如果虚拟 IP 支持不同的检查，且深度为 10 秒，以下命令可让 Pacemaker 每 10 秒执行一次常规的虚拟 IP 检查，每 60 秒执行更高级的监控检查。（如前所述，您不应该配置额外的监控操作，间隔为 10 秒。）

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

第 22 章 PACEMAKER 集群属性

集群属性控制在集群操作过程中遇到可能发生的情况时集群的行为。

22.1. 集群属性和选项概述

下表总结了 Pacemaker 集群属性，显示属性的默认值以及您可以为这些属性设置的可能值。

另外，还有额外的用于隔离功能的集群属性。有关这些属性的详情，请查看确定 隔离设备常规属性 中的隔离行为的集群属性表。

注意

除了本表格中描述的属性外，还有一些由集群软件公开的集群属性。对于这些属性，建议您不要修改其默认值。

表 22.1. 集群属性

选项	默认值	描述
batch-limit	0	集群可以并行执行的资源操作数量。"正确的"值取决于网络和集群节点的速度和负载。默认值为 0，表示当任何节点有高 CPU 负载时，集群会动态强制限制。
migration-limit	-1（无限）	集群允许在节点上并行执行的迁移作业数量。
no-quorum-policy	stop	当集群没有仲裁（quorum）时该做什么。允许的值： * ignore - 继续所有资源管理 * freeze - 继续管理资源，但不会从受影响分区以外的节点中恢复资源 * stop - 停止受影响集群分区中的所有资源 * suicide - 隔离受影响集群分区中的所有节点 * demote - 如果集群分区缺少仲裁，降级任何提升的资源并停止所有其他资源
symmetric-cluster	true	指明资源是否可以默认在任何节点上运行。
cluster-delay	60s	在网络间进行往返延时（不包括操作执行）。"正确的"值取决于网络和集群节点的速度和负载。
dc-deadtime	20s	在启动期间等待来自其他节点的响应时间。"正确的"值将取决于您网络的速度和负载，以及所使用的交换机的类型。

选项	默认值	描述
stop-orphan-resources	true	指明是否应该停止删除的资源。
stop-orphan-actions	true	指明是否应该取消删除的动作。
start-failure-is-fatal	true	指明某个节点上启动资源失败是否防止了在该节点上进一步启动尝试。当设置为 **false** 时，集群会根据资源当前的故障计数和迁移阈值决定是否在同一节点上再次启动。有关为资源设置 **migration-threshold** 选项的详情，请参考 配置资源 meta 选项。 将 **start-failure-is-fatal** 设置为 **false** 会带来风险，即可能会导致一个无法启动资源的节点耽搁所有依赖的操作。这就是为什么 **start-failure-is-fatal** 默认为 true。可以通过设置低迁移阈值来降低设置 **start-failure-is-fatal=false** 的风险，以便其他操作可在多次失败后能够继续。
pe-error-series-max	-1（全部）	调度程序输入的数量会导致要保存 ERRORs。报告问题时使用。
pe-warn-series-max	-1（全部）	调度程序输入的数量会导致 WARNINGs 保存。报告问题时使用。
pe-input-series-max	-1（全部）	要保存的 "normal" 调度程序输入数。报告问题时使用。
cluster-infrastructure		当前运行的 Pacemaker 的消息堆栈。用于信息和诊断目的，用户不能配置。
dc-version		集群的 Designated Controller（DC）上的 Pacemaker 版本。用于诊断目的，用户不能配置。
cluster-recheck-interval	15 分钟	Pacemaker 主要由事件驱动，并提前了解何时重新检查集群的故障超时和大多数基于时间的规则。Pacemaker 还会在此属性指定的不活跃的时间后重新检查集群。此集群重新检查有两个目的：具有 **date-spec** 的规则保证频繁检查，而对于某些类型的调度程序错误，它充当故障安全保护。值 0 禁用这个轮询；正值表示时间间隔。
maintenance-mode	false	Maintenance Mode 让集群进入"手动关闭"模式，而不要启动或停止任何服务，直到有其他指示为止。当维护模式完成后，集群会对任何服务的当前状态进行完整性检查，然后停止或启动任何需要它的状态。

选项	默认值	描述
shutdown-escalation	20min	在经过这个时间后，放弃安全关闭并直接退出。只用于高级使用。
stop-all-resources	false	集群是否应该停止所有资源。
enable-acl	false	指明集群是否可以使用 **pcs acl** 命令设置的访问控制列表。
placement-strategy	default	指定在决定集群节点上资源放置时集群是否以及如何考虑使用属性。
node-health-strategy	none	与健康资源代理一起使用时，控制 Pacemaker 如何对节点健康状况中的变化做出响应。允许的值： * **none** - 不跟踪节点健康状况。 * **migrate-on-red** - 根据代理监控的本地条件，将资源从健康代理确定的节点状态为**red** 的任何节点移出。 * **only-green** - 根据代理监控的本地条件，将资源代理确定的节点状态为 **yellow** 或 **red** 的资源从任何节点移出。 * **progressive**，**custom** - 高级节点健康策略，其根据健康属性的内部数字值，提供集群对健康状况响应的精细控制。

22.2. 设置和删除集群属性

要设置集群属性的值，使用以下 **pcs** 命令。

```
pcs property set property=value
```

例如，若要将 **symmetric-cluster** 的值设置为 **false**，请使用以下命令：

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

您可以使用以下命令从配置中删除集群属性。

```
pcs property unset property
```

另外，您可以通过将 **pcs property set** 命令的 value 字段留空来从配置中删除集群属性。这会将该属性恢复为默认值。例如，如果您之前将 **symmetric-cluster** 属性设置为 **false**，以下命令会从配置中删除您设置的值，并将 **symmetric-cluster** 的值恢复为 **true**，这是它的默认值。

```
# pcs property set symmetic-cluster=
```

22.3. 查询集群属性设置

在大多数情况下，当使用 **pcs** 命令来显示各种集群组件的值时，您可以互换使用 **pcs list** 或 **pcs show**。在以下示例中，**pcs list** 是用来显示多个属性的所有设置的完整列表的格式，而 **pcs show** 是用来显示特定属性值的格式。

要显示为集群设置的属性设置的值，使用以下 pcs 命令。

```
pcs property list
```

要显示集群属性设置的所有值，包括未明确设置的属性设置的默认值，请使用以下命令。

```
pcs property list --all
```

要显示特定集群属性的当前值，请使用以下命令。

```
pcs property show property
```

例如，要显示 **cluster-infrastructure** 属性的当前值，请执行以下命令：

```
# pcs property show cluster-infrastructure
Cluster Properties:
 cluster-infrastructure: cman
```

为方便起见，您可以通过下列命令，显示这些属性的所有默认值，无论是否将其设置为非默认值。

```
pcs property [list|show] --defaults
```

22.4. 使用 PCS 命令导出集群属性

从 Red Hat Enterprise Linux 9.3 开始，您可以使用 **pcs property config** 命令的 **--output-format=cmd** 选项显示可用于在不同系统上重新创建配置的集群属性的 **pcs** 命令。

以下命令将 **migration-limit** 集群属性设置为 10。

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

设置集群属性后，以下命令显示可用于在不同系统上设置集群属性的 **pcs** 命令。

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

第 23 章 配置资源以在清理节点关闭时保持停止

当集群节点关闭时，Pacemaker 的默认响应是停止在该节点上运行的所有资源，并在其它位置恢复这些资源，即使关闭是一个"干净"的关闭。您可以配置 Pacemaker，在节点关闭时，附加到节点的资源将锁定到节点，且无法在其他位置启动，直到节点关闭后重新加入集群时才会再次启动。这样，您可以在维护窗口期间关闭节点，这样可在接受服务中断时关闭节点，而不会导致节点资源切换到集群中的其他节点。

23.1. 集群属性配置资源以在清理节点关闭时保持停止

防止资源在干净节点关闭中进行故障的功能是通过下列集群属性实现的。

shutdown-lock

当将此集群属性设置为默认值 **false** 时，集群将恢复在被完全关闭的节点上活动的资源。当此属性设为 **true** 时，在被完全关闭的节点上活动的资源将无法在其它地方启动，直到它们在重新加入集群后在该节点上再次启动。
shutdown-lock 属性适用于集群节点或远程节点，但不适用于客户机节点。

如果 **shutdown-lock** 设为 **true**，您可以在节点关闭时删除一个集群资源上的锁，以便可通过使用以下命令在节点上手动刷新来在其它地方启动资源。

pcs resource refresh *resource* node=*nodename*

请注意，资源被解锁后，集群就可以自由地将资源移至其他位置。您可以使用粘性值或位置首选项来控制发生这种情况的可能性。

注意

只有在您第一次运行以下命令时，手动刷新才可以在远程节点中使用：

1. 在远程节点上运行 **systemctl stop pacemaker_remote** 命令来停止该节点。
2. 运行 **pcs resource disable *remote-connection-resource*** 命令。

然后您可以在远程节点上手动进行刷新。

shutdown-lock-limit

当将此集群属性设置为默认值 0 以外的其他值时，如果节点在启动关闭后的指定时间内没有重新加入，则资源将在其他节点上可用。

注意

只有在您第一次运行以下命令时，**shutdown-lock-limit** 属性才能用于远程节点：

1. 在远程节点上运行 **systemctl stop pacemaker_remote** 命令来停止该节点。
2. 运行 **pcs resource disable *remote-connection-resource*** 命令。

运行这些命令后，当 **shutdown-lock-limit** 指定的时间过后，远程节点上运行的资源将可用于在其他节点上恢复。

23.2. 设置 SHUTDOWN-LOCK 集群属性

以下示例将示例集群中的 **shutdown-lock** 集群属性设置为 **true**，并显示节点关闭并再次启动时的它的效果。这个示例集群由三个节点组成：**z1.example.com**、**z2.example.com** 和 **z3.example.com**。

步骤

1. 将 **shutdown-lock** 属性设为 **true**，并验证其值。在本例中，**shutdown-lock-limit** 属性保持其默认值 0。

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

2. 检查集群的状态。在本例中，资源 **third** 和 **fifth** 运行在 **z1.example.com** 上。

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

3. 关闭 **z1.example.com**，这将停止在该节点上运行的资源。

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

4. 运行 **pcs status** 命令显示节点 **z1.example.com** 下线了，并且 **z1.example.com** 上运行的资源在节点停机时为 **LOCKED**。

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

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

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

...
```

5. 在 **z1.example.com** 上再次启动集群服务，使其重新加入集群。锁住的资源应该在这个节点上启动，但当它们启动后，它们不一定会停留在同一个节点上。

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

6. 在本例中，在节点 **z1.example.com** 上恢复了资源 **third** 和 **fifth**。

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

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

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

...
```

第 24 章 配置节点放置策略

Pacemaker 根据资源分配分数来决定在每个节点上放置资源的位置。资源将分配给资源分数最高的节点。此分配 score 源自因素的组合，包括资源约束、**resource-stickiness** 设置、各个节点上资源以前的故障历史记录以及每个节点的利用率。

如果所有节点上的资源分配分数相等，默认的放置策略Pacemaker 将选择一个分配的资源最少的节点来平衡负载。如果每个节点中的资源数量相等，则会选择 CIB 中列出的第一个有资格的节点来运行该资源。

但通常不同的资源使用会对节点容量有很大不同（比如内存或者 I/O）。您始终无法通过只考虑分配给节点的资源数量来平衡负载。另外，如果资源的放置使其组合要求超过所提供的容量，则它们可能无法完全启动，或者可能以较低的性能运行。要考虑以上因素，Pacemaker 允许您配置以下组件：

- 特定节点提供的能力
- 特定资源需要的容量
- 资源放置的整体策略

24.1. 使用属性和放置策略

要配置节点提供或需要资源的容量，您可以使用节点和资源 *使用属性*。您可以通过为资源设置使用变量，并将值分配给该变量以指示资源需要，然后为节点设置相同的使用变量，并为该变量分配一个值来指示节点提供的内容。

您可以根据喜好命名使用属性，并根据您的配置定义名称和值对。使用属性的值必须是整数。

24.1.1. 配置节点和资源容量

以下示例为两个节点配置 CPU 容量的使用属性，将此属性设置为变量 **cpu**。它还配置 RAM 容量的使用属性，将此属性设置为变量 **memory**。在本例中：

- 节点 1 定义为提供 2 个 CPU 和 2048 RAM
- 节点 2 定义为提供 4 个 CPU 和 2048 RAM

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

以下示例指定三个不同资源需要的相同的使用属性。在本例中：

- 资源 **dummy-small** 需要 1 个 CPU 容量和 1024 个 RAM 容量
- 资源 **dummy-medium** 需要 2 个 CPU 容量和 2048 个 RAM 容量
- 资源 **dummy-large** 需要 1 个 CPU 容量和 3072 个 RAM 容量

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

如果节点有足够的可用容量以满足资源的要求，则节点被视为有资格获得资源。

24.1.2. 配置放置策略

在配置了节点提供的容量以及资源需要的容量后，您需要设置 **placement-strategy** 集群属性，否则容量配置无效。

placement-strategy 集群属性有四个值：

- **default** - 根本不考虑 Utilization 值。根据分配分数分配资源。如果分数相等，则在节点间平均分配资源。
- **utilization** - 只有在决定节点是否被视为有资格时才会考虑 Utilization 值（即，它是否有足够的可用容量来满足资源的要求）。负载均衡仍会根据分配给节点的资源数量进行。
- **balanced** - 在决定节点是否有资格提供资源以及负载平衡时，会考虑 Utilization 值，因此会尝试以优化资源性能的方式分散资源。
- **minimal** - 只有在决定节点是否有资格为资源提供服务时才会考虑 Utilization 值。对于负载平衡，会尝试尽可能将资源集中到几个节点上，从而在剩余的节点上启用以实现节电的目的。

以下示例命令将 **placement-strategy** 的值设为 **balanced**。运行此命令后，Pacemaker 会确保在整个集群中平均分配来自您的资源负载，而无需使用复杂的托管限制集合。

```
# pcs property set placement-strategy=balanced
```

24.2. PACEMAKER 资源分配

Pacemaker 根据节点首选项、节点容量和资源分配首选项分配资源。

24.2.1. 节点首选项

Pacemaker 根据以下策略决定在分配资源时首选哪个节点。

- 节点权重最高的节点会首先被消耗。节点 weight 是集群维护的分数，以表示节点健康状况。
- 如果多个节点具有相同的节点权重：
  - 如果 **placement-strategy** 集群属性是 **default** 或 **utilization** ：
    - 分配的资源最少的节点会首先被消耗。
    - 如果分配的资源数量相等，在 CIB 中列出的第一个有资格的节点会首先被消耗。
  - 如果 **placement-strategy** 集群属性是 **balanced** ：
    - 具有最多可用容量的节点会首先被消耗。
    - 如果节点的空闲容量相等，首先消耗分配的资源数量最少的节点。
    - 如果节点的空闲容量相等，并且分配的资源数量相等，在 CIB 中列出的第一个有资格的节点会首先被消耗。
  - 如果 **placement-strategy** 集群属性是 **minimal**，则 CIB 中列出的第一个有资格的节点会首先被使用。

24.2.2. 节点容量

Pacemaker 根据以下策略决定哪个节点拥有最多的可用容量。

- 如果只定义了一种类型的使用属性，那么空闲容量就是一个简单的数字比较。
- 如果定义了多个类型的使用属性,那么在最多属性类型中数字最高的节点具有最大的可用容量。例如：
  - 如果 NodeA 有更多可用 CPU，而 NodeB 拥有更多可用内存，则它们的可用容量是相等的。
  - 如果 NodeA 有更多可用 CPU，而 NodeB 有更多可用内存和存储，则 NodeB 具有更多可用容量。

24.2.3. 资源分配首选项

Pacemaker 根据以下策略决定优先分配哪些资源。

- 优先级最高的资源会首先被分配。您可以在创建资源时设置资源优先级。
- 如果资源优先级相等，运行该资源的节点中分数最高的资源会首先被分配，以防止资源 shuffling 的问题。
- 如果资源在运行资源的节点中的分数相等，或者资源没有运行，则首选节点上具有最高分数的资源会被首先分配。如果首选节点上的资源分数相等，则 CIB 中列出的第一个可运行资源会首先被分配。

24.3. 资源放置策略指南

为确保 Pacemaker 对资源放置策略最有效的工作，在配置系统时请考虑以下事项。

- 请确定您有足够的物理容量。
  如果节点在通常情况下使用的物理容量接近近似的最大值，那么在故障切换过程中可能会出现问题。即使没有使用功能，您仍可能会遇到超时和二级故障。
- 在您为节点配置的功能中构建一些缓冲。
  假设 Pacemaker 资源不会使用 100% 配置的 CPU 和内存量，所以所有时间都比您的物理资源稍多。这种方法有时被称为过量使用。
- 指定资源优先级。
  如果集群需要牺牲一些服务，则这些服务应该是对您最不重要的。确保正确设置资源优先级，以便首先调度最重要的资源。

24.4. NODEUTILIZATION 资源代理

NodeUtilization 资源代理可以检测可用的 CPU、主机内存可用性和虚拟机监控程序内存可用性的系统参数，并将这些参数添加到 CIB 中。您可以将代理作为克隆资源运行，使其在每个节点上自动填充这些参数。

有关 **NodeUtilization** 资源代理和此代理的资源选项的详情，请运行 **pcs resource describe NodeUtilization** 命令。

第 25 章 将虚拟域配置为资源

您可以使用 **pcs resource create** 命令将 **libvirt** 虚拟化框架管理的虚拟域配置为集群资源，并将 **VirtualDomain** 指定为资源类型。

当将虚拟域配置为资源时，请考虑以下事项：

- 在将虚拟域配置为集群资源之前，应停止它。
- 一旦虚拟域是集群资源，除了通过集群工具外，它不应该启动、停止或迁移。
- 不要配置您已配置为集群资源的虚拟域，使其在主机引导时启动。
- 所有允许运行虚拟域的节点都必须有权访问该虚拟域所需的配置文件和存储设备。

如果您希望集群管理虚拟域本身中的服务，可以将该虚拟域配置为客户机节点。

25.1. 虚拟域资源选项

下表描述了您可以为 **VirtualDomain** 资源配置的资源选项。

表 25.1. 虚拟域资源资源选项

项	默认值	描述
config		（必需）到此虚拟域的 **libvirt** 配置文件的绝对路径。
hypervisor	依赖系统	要连接的虚拟机管理器 URI。您可以通过运行 **virsh --quiet uri** 命令来确定系统的默认 URI。
force_stop	**0**	在停止时总是强制关闭（"destroy"）域。默认的行为是仅在安全关闭尝试失败后强制关闭。只有在您的虚拟域（或您的虚拟化后端）不支持安全关闭时，才应将其设置为 **true**。
migration_transport	依赖系统	迁移时用来连接到远程管理程序的传输。如果省略此参数，资源将使用 **libvirt** 的默认传输连接到远程 hypervisor。
migration_network_suffix		使用专用的迁移网络。迁移 URI 由在节点名称末尾添加此参数的值组成。如果节点名称是一个完全限定域名（FQDN），在 FQDN 的第一个句点（.）前插入后缀。确定由此组成的主机名可在本地被解析，相关的 IP 地址可以通过网络被访问。

项	默认值	描述
monitor_scripts		要额外监控虚拟域中的服务，请使用要监控的脚本列表添加这个参数。*注:*当使用监控脚本时，只有所有监控脚本都成功完成时，**start** 和 **migrate_from** 操作才会完成。请确定设置这些操作的超时时间，以适应这个延迟
autoset_utilization_cpu	**true**	如果设置为 **true**，代理将通过 **virsh** 检测 **domainU** 的 **vCPU** 数，并在执行监控时将其置于资源的 CPU 使用率中。
autoset_utilization_hv_memory	**true**	如果设置为 true，代理会通过 **virsh** 检测 **Max memory** 的数量，并在执行监控将其置于源的 **hv_memory** 使用率中。
migrateport	随机高端口	此端口将用在 **qemu** 迁移 URI 中。如果未设置，则端口将是一个随机高端口。
snapshot		保存虚拟机镜像的快照目录的路径。设定此参数后，虚拟机的 RAM 状态将在停止后保存在快照目录中的文件。如果启动了某个域的状态文件，域将在最后停止之前恢复到正确的状态。此选项与 **force_stop** 选项不兼容。

除了 **VirtualDomain** 资源选项外，您还可以配置 **allow-migrate** 元数据选项，以允许将资源实时迁移到另一节点上。当此选项设为 **true** 时，可以迁移资源，而且不丢失状态。当此选项设为 **false** 时（这是默认状态），虚拟域将在第一节点上关闭，然后在其从一个节点移到另一个节点时，在第二个节点上重新启动。

25.2. 创建虚拟域资源

以下流程为之前创建的虚拟机在集群中创建一个 **VirtualDomain** 资源。

步骤

1. 要创建 **VirtualDomain** 资源代理来管理虚拟机，Pacemaker 需要虚拟机的 **xml** 配置文件转储到磁盘上的文件中。例如，如果您创建了名为 **guest1** 的虚拟机，请将 **xml** 文件转储到允许运行该客户端的一个集群节点中的某个文件中。您可以使用您选择的文件名；本例使用 **/etc/pacemaker/guest1.xml**。

   ```
   # virsh dumpxml guest1 > /etc/pacemaker/guest1.xml
   ```

2. 将虚拟机的 **xml** 配置文件复制到允许运行客户机的所有其它集群节点中，在每个节点中位于同一位置。

3. 请确定所有允许运行虚拟域的节点都可访问该虚拟域所需的存储设备。

4. 单独测试虚拟域是否可以在每个运行虚拟域的节点中启动和停止。

5. 如果正在运行，请关闭该客户机节点。Pacemaker 会在集群中配置时启动节点。不应该将虚拟机配置为在主机引导时自动启动。

6. 使用 **pcs resource create** 命令配置 **VirtualDomain** 资源。例如，以下命令配置了一个名为 **VM** 的 **VirtualDomain** 资源。由于 **allow-migrate** 选项设置为 **true**，因此 **pcs resource move VM *nodeX*** 命令将作为实时迁移执行。
   在本例中，**migration_transport** 设置为 **ssh**。请注意，要使 SSH 迁移正常工作，无密钥日志记录必须可以在节点间正常工作。

   ```
   # pcs resource create VM VirtualDomain config=/etc/pacemaker/guest1.xml
   migration_transport=ssh meta allow-migrate=true
   ```

第 26 章 配置集群仲裁(QUORUM)

Red Hat Enterprise Linux 高可用性附加组件集群使用 **votequorum** 服务，并结合隔离，以避免脑裂的情况。为集群中的每个系统分配一组投票机制，只能在大多数投票机制都存在时才允许执行集群操作。该服务必须被加载到所有节点或无节点；如果服务被载入到集群节点的一个子集，则结果将无法预计。有关 **votequorum** 服务的配置和操作的详情，请查看 **votequorum**(5)手册页。

26.1. 配置仲裁选项

使用 **pcs cluster setup** 命令创建集群时，可以设置仲裁配置的一些特殊功能。下表总结了这些选项。

表 26.1. 仲裁选项

选项	描述
auto_tie_breaker	启用后，集群可能会以确定的方式达到 50% 个节点同时失败的情况。集群分区或仍与 **auto_tie_breaker_node** 中配置的 **nodeid** 保持联系的节点集合（如果未设置则为最低的 **nodeid** ）将保持仲裁状态。其他节点将为 inquorate。 **auto_tie_breaker** 选项主要用于具有偶数节点的集群，因为它允许集群使用平均分割继续操作。对于更复杂的故障，如多个、不均匀的分割，建议您使用仲裁设备。 **auto_tie_breaker** 选项与仲裁设备不兼容。
wait_for_all	在启用后，只有在所有节点都最少同时可见一次后，集群才会第一次处于仲裁状态。 **wait_for_all** 选项主要用于双节点集群，以及使用仲裁设备 **lms** (last man standing)算法的偶数节点集群。 当集群具有两个节点，不使用仲裁设备，且禁用 **auto_tie_breaker** 时，**wait_for_all** 选项会自动启用。您可以通过将 **wait_for_all** 明确设置为 0 来覆盖它。
last_man_standing	启用后，集群可以在特定情况下重新动态计算 **expected_votes** 和仲裁。启用这个选项时，您必须启用 **wait_for_all**。**last_man_standing** 选项与仲裁设备不兼容。
last_man_standing_window	在集群丢失节点后，在重新计算 **expected_votes** 和仲裁前需要等待的时间（毫秒）。

有关配置和使用这些选项的详情，请查看 **votequorum**(5)手册页。

26.2. 修改仲裁选项

您可以使用 **pcs quorum update** 命令修改集群的常规仲裁选项。您可以在正在运行的系统上修改 **quorum.two_node** 和 **quorum.expected_votes** 选项。对于所有其他仲裁选项，执行此命令需要集群停止。有关仲裁选项的信息，请查看 **votequorum**(5)手册页。

pcs quorum update 命令的格式如下。

```
pcs quorum update [auto_tie_breaker=[0|1]] [last_man_standing=[0|1]] [last_man_standing_window=
[time-in-ms] [wait_for_all=[0|1]]
```

以下一系列命令修改 **wait_for_all** 仲裁选项，并显示选项的更新状态：请注意，系统不允许在集群运行时执行这个命令。

```
[root@node1:~]# pcs quorum update wait_for_all=1
Checking corosync is not running on nodes...
Error: node1: corosync is running
Error: node2: corosync is running

[root@node1:~]# pcs cluster stop --all
node2: Stopping Cluster (pacemaker)...
node1: Stopping Cluster (pacemaker)...
node1: Stopping Cluster (corosync)...
node2: Stopping Cluster (corosync)...

[root@node1:~]# pcs quorum update wait_for_all=1
Checking corosync is not running on nodes...
node2: corosync is not running
node1: corosync is not running
Sending updated corosync.conf to nodes...
node1: Succeeded
node2: Succeeded

[root@node1:~]# pcs quorum config
Options:
  wait_for_all: 1
```

26.3. 显示制裁配置和状态

集群运行后，您可以输入以下集群仲裁命令来显示仲裁配置和状态。

以下命令显示制裁配置。

```
pcs quorum [config]
```

以下命令显示制裁运行时状态。

```
pcs quorum status
```

26.4. 运行非仲裁的集群

如果您将节点长时间移出集群，且这些节点的丢失会导致仲裁丢失，则您可以使用 **pcs quorum expected-votes** 命令更改实时集群的 **expected_votes** 参数的值。这可让集群在没有仲裁的情况下继续操作。

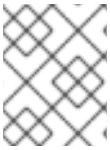
警告

在 Live 集群中更改预期投票时应特别小心。如果因为您手动更改了预期的投票，集群的少于 50% 的部分在运行，那么集群中的其他节点就可以单独启动并运行集群服务，从而导致数据崩溃和其他意外结果。如果更改了这个值，您应该确保启用了 **wait_for_all** 参数。

以下命令将 live 集群中的预期 vote 设置为指定的值。这只会影响实时集群，不会更改配置文件；如果重新加载，则 **expected_votes** 的值将重置为配置文件中的值。

```
pcs quorum expected-votes votes
```

在您知道集群不问题但您希望集群进行资源管理时，您可以使用 **pcs quorum unblock** 命令防止集群在建立仲裁时等待所有节点。



注意

使用这个命令时需要特别小心。在运行此命令前，请确定关闭没有在集群中的节点，并确保无法访问共享资源。

```
# pcs quorum unblock
```

第 27 章 配置仲裁设备

您可以通过配置作为集群第三方设备的独立仲裁设备，让集群出现比标准仲裁规则处理更多节点故障的情况。对于偶数节点的集群，建议使用仲裁设备。对于双节点集群，使用仲裁设备可以更好地决定在脑裂情况下保留哪些节点。

在配置仲裁设备，您必须考虑以下内容。

- 建议您在与使用该仲裁设备的集群相同的站点中的不同的物理网络中运行仲裁设备。理想情况下，仲裁设备主机应该独立于主集群，或者至少位于一个独立的 PSU，而不要与 corosync 环或者环位于同一个网络网段。
- 您不能同时在集群中使用多个仲裁设备。
- 虽然您不能同时在集群中使用多个仲裁设备，但多个集群可能同时使用一个仲裁设备。每个使用这个仲裁设备的集群都可以使用不同的算法和仲裁选项，因为它们保存在集群节点本身。例如，单个仲裁设备可由具有**ffsplit** (50/50 均分)算法的一个集群和具有 **lms** (last man standing)算法的第二个集群来使用。
- 不应在现有集群节点中运行制裁设备。

27.1. 安装制裁设备软件包

为集群配置仲裁设备需要您安装以下软件包：

- 在现有集群的节点上安装 **corosync-qdevice**。

```
[root@node1:~]# dnf install corosync-qdevice
[root@node2:~]# dnf install corosync-qdevice
```

- 在仲裁设备主机上安装 **pcs** 和 **corosync-qnetd**。

```
[root@qdevice:~]# dnf install pcs corosync-qnetd
```

- 在仲裁设备主机上启动 **pcsd** 服务，并在系统启动时启用 **pcsd**。

```
[root@qdevice:~]# systemctl start pcsd.service
[root@qdevice:~]# systemctl enable pcsd.service
```

27.2. 配置仲裁设备

按照以下流程配置仲裁设备，并将其添加到集群中。

在本例中：

- 用于仲裁设备的节点是 **qdevice**。
- 仲裁设备模型是 **net**，这是目前唯一支持的模型。**net** 模型支持以下算法：
  - **ffsplit** ：50-50 均分。这为拥有最多活跃节点的分区提供一个投票。
  - **lMS** :last-man-standing。如果节点是集群中唯一可以看到 **qnetd** 服务器的节点，则它将返回一个投票。

警告

LMS 算法允许在集群中只剩下一个节点时仍保持仲裁，但也意味着制裁设备的投票权利更大，它等同于 number_of_nodes - 1。丢失与制裁设备的连接意味着丢失了 number_of_nodes - 1 个投票，就是说只有所有节点都处于活动状态的集群才能保持仲裁（通过对仲裁设备进行过度投票），其它任何集群都变为不可仲裁。

有关这些算法实现的详情，请查看 **corosync-qdevice**(8)手册页。

- 集群节点是 **node1** 和 **node2**。

步骤

1. 在您要用来托管仲裁设备的节点中，使用以下命令配置仲裁设备。这个命令配置并启动仲裁设备模型 **net**，并将设备配置为在引导时启动。

```
[root@qdevice:~]# pcs qdevice setup model net --enable --start
Quorum device 'net' initialized
quorum device enabled
Starting quorum device...
quorum device started
```

配置制裁设备后，您可以检查其状态。这应该显示 **corosync-qnetd** 守护进程正在运行，此时没有客户端连接上来。**--full** 命令选项提供详细的输出。

```
[root@qdevice:~]# pcs qdevice status net --full
QNetd address:                  *:5403
TLS:                            Supported (client certificate required)
Connected clients:              0
Connected clusters:             0
Maximum send/receive size:      32768/32768 bytes
```

2. 使用以下命令在 **firewalld** 上启用 **high-availability** 服务，从而在防火墙上启用 **pcsd** 守护进程所需的端口和 **net** 仲裁设备：

```
[root@qdevice:~]# firewall-cmd --permanent --add-service=high-availability
[root@qdevice:~]# firewall-cmd --add-service=high-availability
```

3. 在现有集群中的某个节点上，对托管仲裁设备的节点上的用户 **hacluster** 进行身份验证。这允许集群上的 **pcs** 连接到 **qdevice** 主机上的 **pcs**，但不允许 **qdevice** 主机上的 **pcs** 连接到集群上的 **pcs**。

```
[root@node1:~] # pcs host auth qdevice
Username: hacluster
Password:
qdevice: Authorized
```

4. 在集群中添加仲裁设备。

在添加仲裁设备前，您可以检查当前的配置以及仲裁设备的状态以便稍后进行比较。这些命令的输出表示集群还没有使用仲裁设备，每个节点的 **Qdevice** 成员资格状态为 **NR** （未注册）。

```
[root@node1:~]# pcs quorum config
Options:
```

```
[root@node1:~]# pcs quorum status
Quorum information
------------------
Date:             Wed Jun 29 13:15:36 2016
Quorum provider:  corosync_votequorum
Nodes:            2
Node ID:          1
Ring ID:          1/8272
Quorate:          Yes

Votequorum information
----------------------
Expected votes:   2
Highest expected: 2
Total votes:      2
Quorum:           1
Flags:            2Node Quorate

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

以下命令添加您之前在集群中创建的仲裁设备。您不能同时在集群中使用多个仲裁设备。但是，一个仲裁设备可以被多个集群同时使用。这个示例命令将仲裁设备配置为使用 **ffsplit** 算法。有关仲裁设备的配置选项的详情，请查看 **corosync-qdevice**(8)手册页。

```
[root@node1:~]# pcs quorum device add model net host=qdevice algorithm=ffsplit
Setting up qdevice certificates on nodes...
node2: Succeeded
node1: Succeeded
Enabling corosync-qdevice...
node1: corosync-qdevice enabled
node2: corosync-qdevice enabled
Sending updated corosync.conf to nodes...
node1: Succeeded
node2: Succeeded
Corosync configuration reloaded
Starting corosync-qdevice...
node1: corosync-qdevice started
node2: corosync-qdevice started
```

5. 检查仲裁设备的配置状态。
   在集群一端，您可以执行以下命令查看如何更改配置。

   pcs quorum config 显示已配置的仲裁设备。

```
[root@node1:~]# pcs quorum config
Options:
Device:
  Model: net
    algorithm: ffsplit
    host: qdevice
```

pcs quorum status 命令显示仲裁运行时状态，表示仲裁设备正在使用中。每个集群节点的 **Qdevice** 成员资格信息状态值的含义如下：

- **A/NA** - 仲裁设备处于活动状态，代表 **qdevice** 和 **corosync** 之间是否存在心跳。这应该总是表示仲裁设备处于活动状态。
- **V/NV** - 当仲裁设备为某个节点投票时，会设置 **V**。在本例中，两个节点都设置为 **V**，因为它们可以相互通信。如果集群被分成两个单节点集群，其中一个节点将设置为 **V**，其他节点将设置为 **NV**。
- **MW/NMW** - 内部仲裁设备标记(**MW**)或未设置(**NMW**)。默认情况下不设置 标志，值为 **NMW**。

```
[root@node1:~]# pcs quorum status
Quorum information
------------------
Date:             Wed Jun 29 13:17:02 2016
Quorum provider:  corosync_votequorum
Nodes:            2
Node ID:          1
Ring ID:          1/8272
Quorate:          Yes

Votequorum information
----------------------
Expected votes:   3
Highest expected: 3
Total votes:      3
Quorum:           2
Flags:            Quorate Qdevice

Membership information
----------------------
    Nodeid      Votes    Qdevice Name
         1          1    A,V,NMW node1 (local)
         2          1    A,V,NMW node2
         0          1            Qdevice
```

pcs quorum device status 显示仲裁设备运行时状态。

```
[root@node1:~]# pcs quorum device status
Qdevice information
-------------------
Model:                  Net
Node ID:                1
Configured node list:
    0   Node ID = 1
    1   Node ID = 2
```

```
Membership node list:   1, 2

Qdevice-net information
----------------------
Cluster name:          mycluster
QNetd host:            qdevice:5403
Algorithm:             ffsplit
Tie-breaker:           Node with lowest node ID
State:                 Connected
```

在仲裁设备一侧，您可以执行以下状态命令，其显示 **corosync-qnetd** 守护进程的状态：

```
[root@qdevice:~]# pcs qdevice status net --full
QNetd address:                  *:5403
TLS:                            Supported (client certificate required)
Connected clients:              2
Connected clusters:             1
Maximum send/receive size:      32768/32768 bytes
Cluster "mycluster":
   Algorithm:          ffsplit
   Tie-breaker:        Node with lowest node ID
   Node ID 2:
      Client address:         ::ffff:192.168.122.122:50028
      HB interval:            8000ms
      Configured node list:   1, 2
      Ring ID:                1.2050
      Membership node list:   1, 2
      TLS active:             Yes (client certificate verified)
      Vote:                   ACK (ACK)
   Node ID 1:
      Client address:         ::ffff:192.168.122.121:48786
      HB interval:            8000ms
      Configured node list:   1, 2
      Ring ID:                1.2050
      Membership node list:   1, 2
      TLS active:             Yes (client certificate verified)
      Vote:                   ACK (ACK)
```

27.3. 管理仲裁设备服务

PCS 提供了在本地主机上管理仲裁设备服务 (**corosync-qnetd**)的能力，如下例所示。请注意，这些命令只影响 **corosync-qnetd** 服务。

```
[root@qdevice:~]# pcs qdevice start net
[root@qdevice:~]# pcs qdevice stop net
[root@qdevice:~]# pcs qdevice enable net
[root@qdevice:~]# pcs qdevice disable net
[root@qdevice:~]# pcs qdevice kill net
```

27.4. 管理集群中的仲裁设备

您可以使用各种 **pcs** 命令来更改群集中的仲裁设备设置、禁用仲裁设备并删除仲裁设备。

27.4.1. 更改仲裁设备设置

您可以使用 **pcs quorum device update** 命令更改仲裁设备的设置。

警告

要更改仲裁设备模型 **net** 的 **host** 选项，请使用 **pcs quorum device remove** 和 **pcs quorum device add** 命令来正确设置配置，除非旧主机和新主机是同一台机器。

以下命令将仲裁设备算法改为 **lms**。

```
[root@node1:~]# pcs quorum device update model algorithm=lms
Sending updated corosync.conf to nodes...
node1: Succeeded
node2: Succeeded
Corosync configuration reloaded
Reloading qdevice configuration on nodes...
node1: corosync-qdevice stopped
node2: corosync-qdevice stopped
node1: corosync-qdevice started
node2: corosync-qdevice started
```

27.4.2. 删除仲裁设备

以下命令删除在集群节点中配置的仲裁设备。

```
[root@node1:~]# pcs quorum device remove
Sending updated corosync.conf to nodes...
node1: Succeeded
node2: Succeeded
Corosync configuration reloaded
Disabling corosync-qdevice...
node1: corosync-qdevice disabled
node2: corosync-qdevice disabled
Stopping corosync-qdevice...
node1: corosync-qdevice stopped
node2: corosync-qdevice stopped
Removing qdevice certificates from nodes...
node1: Succeeded
node2: Succeeded
```

删除仲裁设备后，您应该在显示仲裁设备状态时看到以下出错信息。

```
[root@node1:~]# pcs quorum device status
Error: Unable to get quorum status: corosync-qdevice-tool: Can't connect to QDevice socket (is
QDevice running?): No such file or directory
```

27.4.3. 销毁仲裁设备

以下命令禁用和停止仲裁设备主机上的仲裁设备并删除它的所有配置文件。

```
[root@qdevice:~]# pcs qdevice destroy net
Stopping quorum device...
quorum device stopped
quorum device disabled
Quorum device 'net' configuration files removed
```

第 28 章 为集群事件触发脚本

Pacemaker 集群是一个事件驱动的系统，其中事件可能是资源或节点故障、配置更改或资源启动或停止。您可以将 Pacemaker 集群警报配置为在集群事件发生时执行一些外部操作，警报代理是集群调用的外部程序，其方式与集群调用的资源代理处理资源配置和操作相同。

集群使用环境变量将事件信息传递给代理。代理可以执行任何操作，比如发送电子邮件信息或登录到某个文件或更新监控系统。

- Pacemaker 提供了几个示例报警代理，这些代理默认安装在 **/usr/share/pacemaker/alerts** 中。这些样本脚本可以像现在一样复制和使用，或者可作为模板使用，以适应您的目的。关于它们支持的所有属性，请参考样本代理的源代码。
- 如果示例警报代理不满足您的需要，您可以编写自己的警报代理来调用。

28.1. 安装并配置示例警报代理

当使用示例警报代理时，您应该检查该脚本以确保它适合您的需要。这些示例代理是作为特定集群环境自定义脚本的起点。请注意，虽然红帽支持报警代理脚本用来与 Pacemaker 进行通信的接口，但红帽并不支持自定义代理。

要使用示例报警代理中的一个，您必须在集群中的每个节点上安装代理。例如，以下命令安装 **alert_file.sh.sample** 脚本来作为 **alert_file.sh**。

```
# install --mode=0755 /usr/share/pacemaker/alerts/alert_file.sh.sample
/var/lib/pacemaker/alert_file.sh
```

安装脚本后，您可以创建使用该脚本的报警。

以下示例配置了一个报警，其使用安装的 **alert_file.sh** 报警代理将事件记录到文件中。报警代理是以用户 **hacluster** 身份运行的，该用户具有最小的权限集。

这个示例创建了一个日志文件 **pcmk_alert_file.log**，该文件将用于记录事件。然后，它会创建报警代理，并将路径添加到日志文件来作为其接收者。

```
# touch /var/log/pcmk_alert_file.log
# chown hacluster:haclient /var/log/pcmk_alert_file.log
# chmod 600 /var/log/pcmk_alert_file.log
# pcs alert create id=alert_file description="Log events to a file."
path=/var/lib/pacemaker/alert_file.sh
# pcs alert recipient add alert_file id=my-alert_logfile value=/var/log/pcmk_alert_file.log
```

以下示例安装了 **alert_snmp.sh.sample** 脚本来作为 **alert_snmp.sh**，并配置了一个报警，其使用安装的 **alert_snmp.sh** 报警代理来将集群事件作为 SNMP 陷阱发送。默认情况下，该脚本会发送除成功监控调用 SNMP 服务器外的所有事件。这个示例将时间戳格式配置为 meta 选项。配置报警后，本例为报警配置了一个接收者，并显示报警配置。

```
# install --mode=0755 /usr/share/pacemaker/alerts/alert_snmp.sh.sample
/var/lib/pacemaker/alert_snmp.sh
# pcs alert create id=snmp_alert path=/var/lib/pacemaker/alert_snmp.sh meta timestamp-
format="%Y-%m-%d,%H:%M:%S.%01N"
# pcs alert recipient add snmp_alert value=192.168.1.2
# pcs alert
Alerts:
```

```
 Alert: snmp_alert (path=/var/lib/pacemaker/alert_snmp.sh)
  Meta options: timestamp-format=%Y-%m-%d,%H:%M:%S.%01N.
  Recipients:
   Recipient: snmp_alert-recipient (value=192.168.1.2)
```

以下示例安装了 **alert_smtp.sh** 代理，然后配置了一个报警，其使用安装的报警代理来将集群事件作为电子邮件消息发送。配置报警后，本示例配置了一个接收者，并显示报警配置。

```
# install --mode=0755 /usr/share/pacemaker/alerts/alert_smtp.sh.sample
/var/lib/pacemaker/alert_smtp.sh
# pcs alert create id=smtp_alert path=/var/lib/pacemaker/alert_smtp.sh options
email_sender=donotreply@example.com
# pcs alert recipient add smtp_alert value=admin@example.com
# pcs alert
Alerts:
 Alert: smtp_alert (path=/var/lib/pacemaker/alert_smtp.sh)
  Options: email_sender=donotreply@example.com
  Recipients:
   Recipient: smtp_alert-recipient (value=admin@example.com)
```

28.2. 创建集群警报

以下命令创建集群警报。您配置的选项是特定于代理的配置值，这些值会被传递给您指定为额外环境变量的路径上的报警代理脚本。如果没有为 **id** 指定值，则会生成一个。

```
pcs alert create path=path [id=alert-id] [description=description] [options [option=value]...] [meta
[meta-option=value]...]
```

可能会配置多个报警代理，集群会对每个事件调用它们。报警代理只会在集群节点上调用。会为涉及 Pacemaker 远程节点的事件调用它们，但不会在这些节点上调用它们。

以下示例创建了一个简单的报警，它将对每个事件调用 **myscript.sh**。

```
# pcs alert create id=my_alert path=/path/to/myscript.sh
```

28.3. 显示、修改和删除集群警报

您可以使用各种 **pcs** 命令来显示、修改和删除集群警报。

以下命令显示了所有配置的报警以及配置选项的值。

```
pcs alert [config|show]
```

以下命令使用指定的 *alert-id* 值更新一个现有报警。

```
pcs alert update alert-id [path=path] [description=description] [options [option=value]...] [meta [meta-
option=value]...]
```

以下命令使用指定的 *alert-id* 值删除一个报警。

```
pcs alert remove alert-id
```

或者，您可以运行 **pcs alert delete** 命令，该命令与 **pcs alert remove** 命令相同。**pcs alert delete** 和 **pcs alert remove** 命令都允许您指定要删除的多个报警。

28.4. 配置集群警报接收者

通常，报警是指向一个接收者的。因此，每个报警可能会被额外配置一个或多个接收者。集群将为每个接收者单独调用代理。

接收者可以是警告代理可识别的任何内容：IP 地址、电子邮件地址、文件名或特定代理支持的任何内容。

以下命令为指定报警添加一个新的接收者。

```
pcs alert recipient add alert-id value=recipient-value [id=recipient-id] [description=description] [options [option=value]...] [meta [meta-option=value]...]
```

以下命令更新现有的报警接收者。

```
pcs alert recipient update recipient-id [value=recipient-value] [description=description] [options [option=value]...] [meta [meta-option=value]...]
```

以下命令移除指定的报警接收者。

```
pcs alert recipient remove recipient-id
```

或者，您可以运行 **pcs alert receiver delete** 命令，该命令与 **pcs alert receiver remove** 命令相同。**pcs alert receiver remove** 和 **pcs alert receiver delete** 命令都允许您删除多个报警接收者。

以下示例命令将接收者 ID 为 **my-recipient-id** 的 **my-alert-recipient** 报警接收者添加到报警 **my-alert** 中。这会配置集群来调用报警脚本，该脚本已对每个事件的 **my-alert** 进行了配置，并将接收者 **some-address** 作为环境变量传递。

```
# pcs alert recipient add my-alert value=my-alert-recipient id=my-recipient-id options value=some-address
```

28.5. 警报 META 选项

与资源代理一样，可以对报警代理配置 meta 选项来影响 Pacemaker 调用它们的方式。下表描述了警报 meta 选项。meta 选项可以为每个报警代理和每个接收者配置。

表 28.1. 报警 Meta 选项

meta-Attribute	默认值	描述
enabled	**true**	(RHEL 9.3 及更新的版本)如果将警报设置为 **false**，则不会使用警报。如果将警报设置为 **true**，将特定接收者设置为**false**，则不会使用该接收者。
timestamp-format	%H:%M:%S.%06N	将事件时间戳发送到代理时，集群将使用的格式。这是与 **date**(1)命令一起使用的字符串。

meta-Attribute	默认值	描述
timeout	30s	如果报警代理没有在这段时间内完成，它将被终止。

以下示例配置了一个报警，其调用脚本 **myscript.sh** ，然后为报警添加两个接收者。第一个接收者 ID 为 **my-alert-recipient1**，第二个收件者的 ID 为 **my-alert-recipient2**。这个脚本会为每个事件调用两次，每个调用都使用 15 秒超时。一个调用将被传递给接收者 **someuser@example.com**，时间戳格式为 %D %H:%M，另一个调用将被传递给接收者 **otheruser@example.com**，时间戳格式为 %c。

```
# pcs alert create id=my-alert path=/path/to/myscript.sh meta timeout=15s
# pcs alert recipient add my-alert value=someuser@example.com id=my-alert-recipient1 meta timestamp-format="%D %H:%M"
# pcs alert recipient add my-alert value=otheruser@example.com id=my-alert-recipient2 meta timestamp-format="%c"
```

28.6. 集群警报配置命令示例

以下连续示例展示了一些基本的报警配置命令，以显示用于创建报警、添加接收者和显示配置的报警的格式。

请注意，虽然您必须在集群中的每个节点上安装警报代理，但您需要仅运行一次 **pcs** 命令。

以下命令创建了一个简单的报警，为报警添加两个接收者，并显示配置的值。

- 由于没有指定报警 ID 值，系统会创建 **alert** 的报警 ID 值。
- 第一个接收者创建命令指定 **rec_value** 的接收者。由于这个命令没有指定接收者 ID，因此 **alert-recipient** 的值被用作接收者 ID。
- 第二个接收者创建命令指定了 **rec_value2** 的接收者。此命令为接收者指定 **my-recipient** 的接收者 ID。

```
# pcs alert create path=/my/path
# pcs alert recipient add alert value=rec_value
# pcs alert recipient add alert value=rec_value2 id=my-recipient
# pcs alert config
Alerts:
 Alert: alert (path=/my/path)
  Recipients:
   Recipient: alert-recipient (value=rec_value)
   Recipient: my-recipient (value=rec_value2)
```

以下命令添加第二个报警，以及该报警的接收者。第二个报警的报警 ID 是 **my-alert**，接收者的值是 **my-other-recipient**。因为没有指定接收者 ID，系统会提供接收者 ID **my-alert-recipient**。

```
# pcs alert create id=my-alert path=/path/to/script description=alert_description options option1=value1 opt=val meta timeout=50s timestamp-format="%H%B%S"
# pcs alert recipient add my-alert value=my-other-recipient
# pcs alert
Alerts:
 Alert: alert (path=/my/path)
```

```
  Recipients:
   Recipient: alert-recipient (value=rec_value)
   Recipient: my-recipient (value=rec_value2)
 Alert: my-alert (path=/path/to/script)
  Description: alert_description
  Options: opt=val option1=value1
  Meta options: timestamp-format=%H%B%S timeout=50s
  Recipients:
   Recipient: my-alert-recipient (value=my-other-recipient)
```

以下命令修改报警 **my-alert** 和接收者 **my-alert-recipient** 的报警值。

```
# pcs alert update my-alert options option1=newvalue1 meta timestamp-format="%H%M%S"
# pcs alert recipient update my-alert-recipient options option1=new meta timeout=60s
# pcs alert
Alerts:
 Alert: alert (path=/my/path)
  Recipients:
   Recipient: alert-recipient (value=rec_value)
   Recipient: my-recipient (value=rec_value2)
 Alert: my-alert (path=/path/to/script)
  Description: alert_description
  Options: opt=val option1=newvalue1
  Meta options: timestamp-format=%H%M%S timeout=50s
  Recipients:
   Recipient: my-alert-recipient (value=my-other-recipient)
    Options: option1=new
    Meta options: timeout=60s
```

以下命令从 **alert** 中删除接收者 **my-alert-recipient**。

```
# pcs alert recipient remove my-recipient
# pcs alert
Alerts:
 Alert: alert (path=/my/path)
  Recipients:
   Recipient: alert-recipient (value=rec_value)
 Alert: my-alert (path=/path/to/script)
  Description: alert_description
  Options: opt=val option1=newvalue1
  Meta options: timestamp-format="%M%B%S" timeout=50s
  Recipients:
   Recipient: my-alert-recipient (value=my-other-recipient)
    Options: option1=new
    Meta options: timeout=60s
```

以下命令将从配置中删除 **myalert**。

```
# pcs alert remove myalert
# pcs alert
Alerts:
 Alert: alert (path=/my/path)
  Recipients:
   Recipient: alert-recipient (value=rec_value)
```

28.7. 编写集群警报代理

Pacemaker 集群警报有三种类型：节点警报、隔离警报和资源警报。传递给报警代理的环境变量可能会由于报警类型而有所不同。下表描述了传递给警报代理的环境变量，并指定环境变量何时与特定警报类型关联。

表 28.2. 传递给报警代理的环境变量

环境变量	描述
CRM_alert_kind	报警类型（节点、隔离或资源）
CRM_alert_version	发送报警的 Pacemaker 版本
CRM_alert_recipient	配置的接收者
CRM_alert_node_sequence	每当在本地节点上发出报警时，序列号就会增加，它可以用来参考 Pacemaker 所发出报警的顺序。发生时间较晚的事件的报警序列号比发生时间较早的事件的报警序列号要高。请注意，这个数字没有集群范围的含义。
CRM_alert_timestamp	执行代理前创建的时间戳，使用由 **timestamp-format** meta 选项指定的格式。这可以确保在事件发生时代理有一个可靠、高度准确的时间，无论代理本身何时被调用（这可能会因为系统负载或其他情况而延迟）。
CRM_alert_node	受影响节点的名称
CRM_alert_desc	有关事件的详情。对于节点报警，这是节点的当前状态（成员或丢失）。对于隔离报警，这是请求的隔离操作的总结，其中包括起源、目标以及隔离操作错误代码（若有的话）。对于资源报警，这是等同于 **CRM_alert_status** 的可读字符串。
CRM_alert_nodeid	状态更改了的节点 ID（仅由节点报警提供）
CRM_alert_task	请求的隔离或资源操作（仅由隔离和资源报警提供）
CRM_alert_rc	保护或资源操作的数字返回代码（仅由隔离和资源警告提供）
CRM_alert_rsc	受影响的资源的名称（仅限资源报警）
CRM_alert_interval	资源操作的时间间隔（仅限资源报警）
CRM_alert_target_rc	操作的预期数字返回码（仅用于资源报警）
CRM_alert_status	Pacemaker 用来表示操作结果的数字码（仅用于资源报警）

在编写报警代理时，您必须考虑以下问题。

- 警告代理可以在没有接收者的情况下被调用（如果没有配置任何接收者），因此代理必须能够处理这种情况，即使它只在那种情况下才会退出。用户可以修改配置阶段，并在以后添加一个接收者。

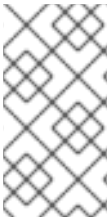
- 如果为报警配置了多个接收者，则会为每个接收者调用一次报警代理。如果代理无法同时运行，则应该只使用单个的接收者进行配置。不过，代理可以自由地将接收者解析为一个列表。

- 当发生集群事件时，所有报警作为单独的进程同时被触发。根据配置了多少报警和接收方以及报警代理中的操作，可能会发生严重的负载突发。可以编写代理来考虑这一点，例如将资源密集型操作排队到其他实例中，而不是直接执行。

- 报警代理以 **hacluster 用户身份运行**，该用户具有最小权限集。如果代理需要额外的特权，建议配置 **sudo** 以允许代理以具有适当特权的另一用户的身份运行必要的命令。

- 请小心地验证和清理用户配置的参数，如 **CRM_alert_timestamp**（其内容由用户配置的 **timestamp-format** 指定）、**CRM_alert_recipient** 和所有报警选项。这是防止配置错误所必需的。此外，如果某些用户可以在不具有 **hacluster** 级别权限访问集群节点的情况下修改了 CIB，则这也是一个潜在的安全问题，您应该避免注入代码的可能性。

- 如果集群包含资源，而对资源操作的 **on-fail** 参数设为 **fence**，则失败时会有多个隔离通知，每个设置了此参数的资源都有一个通知，另外还有一个通知。**pacemaker-fenced** 和 **pacemaker-controld** 将发送通知。pacemaker 在这种情况下只能执行一个实际隔离操作，无论发送了多少条通知。



注意

报警接口设计为与 **ocf:pacemaker:ClusterMon** 资源使用的外部脚本接口向后兼容。为了保持这种兼容性，传递给报警代理的环境变量会带有 **CRM_notify_** 和 **CRM_alert_** 前缀。兼容性方面的一个问题是 **ClusterMon** 资源以 root 用户身份运行外部脚本，而报警代理则以 **hacluster** 用户身份运行。

第 29 章 多站点 PACEMAKER 集群

当集群跨越多个站点时，站点间网络连接的问题可能会导致崩溃问题。当连接断开时，某个位置的节点无法判断位于另一个站点中的某个节点是否失败，或者仍然能够使用失败的站点间连接。此外，在两个站点间提供高可用性服务可能会有问题。为解决这些问题，Pacemaker 完全支持通过使用 Booth 集群票据管理器配置跨多个站点的高可用性集群。

29.1. BOOTH 集群票据管理器概述

Booth *票据管理器（ticket manager）* 是一个分布式服务，它应该在与在特定站点连接集群节点的网络不同的物理网络中运行。它会产生另一个松散集群，*Booth formation*，位于站点的常规集群之上。这可整合沟通层，为独立的 Booth ticket 采用基于认可的决策流程。

Booth *ticket* 是 Booth formation 中的单例，代表一个对时间敏感、可移动的授权单元。资源可以被配置为需要运行某个 ticket。这样可保证资源一次只在一个站点运行，并为其提供 ticket。

您可以将 Booth 看成一个覆盖集群，由在不同站点中运行的集群组成，所有原始集群相互独立。这是与集群沟通的 Booth 服务，它是否获得一个 ticket，而 Pacemaker 会根据 Pacemaker ticket 约束决定是否在集群中运行资源。这意味着，在使用 ticket 管理器时，每个集群都可以运行自己的资源和共享资源。例如，在一个集群中只能运行资源 A、B 和 C，资源 D、E 和 F 仅在另一个集群中运行，且在这两个集群中之一运行的资源 G 和 H 由 ticket 决定。也可以按照一个单独的 ticket 来决定在两个集群中运行的额外资源 J。

29.2. 使用 PACEMAKER 配置多站点集群

您可以使用以下流程配置使用 Booth 票据管理器的多站点配置。

这些示例命令使用以下协议：

- 集群 1 由节点 **cluster1-node1** 和 **cluster1-node2** 组成
- 集群 1 具有为其分配的浮动 IP 地址 192.168.11.100
- 集群 2 由 **cluster2-node1** 和 **cluster2-node2** 组成
- 集群 2 具有为其分配的浮动 IP 地址 192.168.22.100
- 仲裁节点是 **arbitrator-node**，其 IP 地址为 192.168.99.100
- 此配置使用的 Booth ticket 的名称是 **apacheticket**

这些示例命令假定已将 Apache 服务的集群资源配置为每个集群的资源组 **apachegroup** 的一部分。不需要每个集群上的资源和资源组为这些资源配置一个 ticket 约束，因为每个集群的 Pacemaker 实例都是独立的，但这是一个常见故障转移的场景。

请注意，在配置过程中，您可以随时输入 **pcs booth config** 命令来显示当前节点或集群的 booth 配置，或输入 **pcs booth status** 命令来显示本地节点上 booth 的当前状态。

步骤

1. 在两个集群的每个节点上都安装 **booth-site** Booth ticket manager 软件包。

```
[root@cluster1-node1 ~]# dnf install -y booth-site
[root@cluster1-node2 ~]# dnf install -y booth-site
[root@cluster2-node1 ~]# dnf install -y booth-site
```

```
[root@cluster2-node2 ~]# dnf install -y booth-site
```

2. 在仲裁节点上安装 **pcs**、**booth-core** 和 **booth-arbitrator** 软件包。

```
[root@arbitrator-node ~]# dnf install -y pcs booth-core booth-arbitrator
```

3. 如果您正在运行 **firewalld** 守护进程，请在两个集群的所有节点上以及仲裁程序节点上执行以下命令，以启用红帽高可用性附加组件所需的端口。

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

您可能需要修改开放端口以适合本地条件。有关红帽高可用性附加组件所需端口的更多信息，请参阅 为高可用性附加组件启用端口。

4. 在一个集群的一个节点上创建 Booth 配置。您为每个集群和地区指定的地址必须是 IP 地址。对于每个集群，您可以指定一个浮动 IP 地址。

```
[cluster1-node1 ~] # pcs booth setup sites 192.168.11.100 192.168.22.100 arbitrators
192.168.99.100
```

这个命令会在运行它的节点上创建配置文件 **/etc/booth/booth.conf** 和 **/etc/booth/booth.key**。

5. 为 Booth 配置创建 ticket。这是您要用来定义资源约束的票据，允许仅在向集群授予这个票据时运行资源。
这个基本故障转移配置过程只使用一个 ticket，但您可以为每个复杂情况创建额外的 ticket，因为每个 ticket 都与不同的资源或资源关联。

```
[cluster1-node1 ~] # pcs booth ticket add apacheticket
```

6. 将 Booth 配置同步至当前集群中的所有节点。

```
[cluster1-node1 ~] # pcs booth sync
```

7. 在仲裁机构（arbitrator）节点中，将 Booth 配置拉取到仲裁机构中。如果您之前还没有这样做，则必须首先向要提取配置的节点验证 **pcs**。

```
[arbitrator-node ~] # pcs host auth cluster1-node1
[arbitrator-node ~] # pcs booth pull cluster1-node1
```

8. 将 Booth 配置拉取到其他集群，并同步到该集群的所有节点。与仲裁节点一样，如果您之前还没有这样做，您必须首先向要提取配置的节点验证 **pcs**。

```
[cluster2-node1 ~] # pcs host auth cluster1-node1
[cluster2-node1 ~] # pcs booth pull cluster1-node1
[cluster2-node1 ~] # pcs booth sync
```

9. 在仲裁机构中开启并启动 Booth。

注意

您不能在集群的任何节点上手动启动或启用 Booth，因为 Booth 作为这些集群中的 Pacemaker 资源运行。

```
[arbitrator-node ~] # pcs booth start
[arbitrator-node ~] # pcs booth enable
```

10. 使用分配给每个集群的浮动 IP 地址，将 Booth 配置为在两个集群站点上作为集群资源运行。这将创建一个资源组，并将 **booth-ip** 和 **booth-service** 作为该组的成员。

```
[cluster1-node1 ~] # pcs booth create ip 192.168.11.100
[cluster2-node1 ~] # pcs booth create ip 192.168.22.100
```

11. 为您为每个集群定义的资源组添加一个 ticket 约束。

```
[cluster1-node1 ~] # pcs constraint ticket add apacheticket apachegroup
[cluster2-node1 ~] # pcs constraint ticket add apacheticket apachegroup
```

    您可以输入以下命令来显示当前配置的 ticket 约束。

```
pcs constraint ticket [show]
```

12. 为第一个集群授予您为此设置创建的 ticket。
    请注意，在授予 ticket 前不需要定义 ticket 约束。一旦您最初向集群授予了一个票据，Booth 会接管票据管理，除非您使用 **pcs booth ticket revoke** 命令手动覆盖了此票据。有关 **pcs booth** 管理命令的详情，请查看 **pcs booth** 命令的 PCS 帮助屏幕。

```
[cluster1-node1 ~] # pcs booth ticket grant apacheticket
```

可在任何时间添加或删除票据，即使完成此步骤后也是如此。但是，添加或删除一个 ticket 后，您必须将配置文件同步到其他节点和集群，并赋予这个问题单。

有关可用于清理和删除 Booth 配置文件、票据和资源的其他 Booth 管理命令的详情，请参考 **pcs booth** 命令的 PCS 帮助屏幕。

第 30 章 将非 COROSYNC 节点整合到集群中： PACEMAKER_REMOTE 服务

pacemaker_remote 服务允许没有运行 **corosync** 的节点集成到集群中，让集群管理它们的资源，就像它们是实际的集群节点一样。

pacemaker_remote 服务提供的功能如下：

- **pacemaker_remote** 服务允许您扩展到红帽支持超过 32 个节点的限制。
- **pacemaker_remote** 服务允许您将虚拟环境作为集群资源来管理，还可以将虚拟环境中的单个服务作为集群资源来管理。

以下术语用于描述 **pacemaker_remote** 服务：

- *集群节点* - 运行高可用性服务（**pacemaker 和 corosync**）的节点。
- *远程节点* - 运行 **pacemaker_remote** 的节点，用于远程集成到集群中，而无需 **corosync** 集群成员资格。远程节点被配置为使用 **ocf:pacemaker:remote** 资源代理的集群资源。
- *客户机节点* - 运行 **pacemaker_remote** 服务的虚拟客户机节点。虚拟客体资源由集群管理，它由集群启动，并作为远程节点集成到集群中。
- *pacemaker_remote* - 是一个可在远程节点和 Pacemaker 集群环境中 KVM 客户机节点中执行远程应用程序管理的服务守护进程。这个服务是 Pacemaker 的本地 executor 守护进程(**pacemaker-execd**)的改进版本，能够在没有运行 corosync 的节点中远程管理资源。

运行 **pacemaker_remote** 服务的 Pacemaker 集群具有以下特征：

- 远程节点和客户机节点运行 **pacemaker_remote** 服务（虚拟机端只需要很少的配置）。
- 在集群节点上运行的集群堆栈（**pacemaker** 和 **corosync**）连接到远程节点上的 **pacemaker_remote** 服务，允许它们集成到集群中。
- 在集群节点上运行的集群堆栈（**pacemaker** 和 **corosync**）可启动客户机节点，并立即连接到客户机节点上的 **pacemaker_remote** 服务，允许它们集成到集群中。

集群节点与集群节点管理的远程和客户机节点之间的关键区别在于远程和客户机节点没有运行集群堆栈。这意味着远程和虚拟机节点有以下限制：

- 它们不会在仲裁里进行
- 它们不执行隔离设备操作
- 他们没有有资格成为集群的指定控制器（DC）
- 它们本身不运行所有的 **pcs** 命令

另外，远程节点和客户机节点不与与集群堆栈关联的可扩展性限制绑定。

除这些限制外，远程和客户机节点的行为与集群节点在资源管理方面的行为类似，且远程和虚拟机节点本身也可被保护。集群完全能够管理和监控每个远程和客户机节点上的资源：您可以针对它们构建限制，将其置于备用状态，或使用 **pcs** 命令在群集节点上执行任何其他操作。远程和虚拟机节点如集群节点一样显示在集群状态输出中。

30.1. PACEMAKER_REMOTE 节点的主机和虚拟机验证

集群节点与 pacemaker_remote 之间的连接是使用传输层安全（TLS）进行安全保护，使用预共享密钥（PSK）加密和验证 TCP（默认使用端口 3121）进行验证。这意味着集群节点和运行 **pacemaker_remote** 的节点必须共享相同的私钥。默认情况下，此密钥必须放在集群节点和远程节点上的 **/etc/pacemaker/authkey** 中。

pcs cluster node add-guest 命令为客户机节点设置 **authkey**，而 **pcs cluster node add-remote** 命令则为远程节点设置 **authkey**。

30.2. 配置 KVM 客户机节点

Pacemaker 客户机节点是运行 **pacemaker_remote** 服务的虚拟客户机节点。虚拟客户机节点由集群管理。

30.2.1. 客户端节点资源选项

当将虚拟机配置为作为客户机节点时，您可以创建一个 **VirtualDomain** 资源，用于管理该虚拟机。有关您可以为 **VirtualDomain** 资源设置的选项的描述，请参阅虚拟域资源选项中的"虚拟域资源 资源选项 "表。

除了 **VirtualDomain** 资源选项外，元数据选项将资源定义为客户机节点，并定义了连接参数。您可以使用 **pcs cluster node add-guest** 命令设置这些资源选项。下表描述了这些元数据选项。

表 30.1. 将 KVM 资源配置为远程节点的元数据选项

项	默认值	描述
remote-node	<none>	此资源定义的客户机节点的名称。这可让资源作为客户机节点启用，并定义用于识别客户端节点的唯一名称。*警告*：这个值不能与任何资源或节点 ID 重叠。
remote-port	3121	配置一个自定义端口，用于到 **pacemaker_remote** 的客户机连接
remote-addr	**pcs host auth** 命令提供的地址	要连接的 IP 地址或主机名
remote-connect-timeout	60s	待处理的客户端连接超时前的时间

30.2.2. 将虚拟机整合为客户机节点

以下流程是有关 Pacemaker 启动虚拟机以及将机器作为客户机节点集成的步骤的高级概述，使用 **libvirt** 和 KVM 虚拟机。

步骤

1. 配置 **VirtualDomain** 资源。
2. 在每一虚拟机上输入以下命令来安装 **pacemaker_remote** 软件包，启动 **pcsd** 服务并启用它在启动时运行，并允许 TCP 端口 3121 通过防火墙。

```
# dnf install pacemaker-remote resource-agents pcs
```

```
# systemctl start pcsd.service
# systemctl enable pcsd.service
# firewall-cmd --add-port 3121/tcp --permanent
# firewall-cmd --add-port 2224/tcp --permanent
# firewall-cmd --reload
```

3. 为每个虚拟机分配一个静态网络地址和唯一主机名，适用于所有节点。

4. 如果您还没有这样做，在要整合为最优节点的节点上验证 **pcs**。

```
# pcs host auth nodename
```

5. 使用以下命令将现有 **VirtualDomain** 资源转换为客户机节点。这个命令必须在集群节点上运行，而不是在要添加的客户端节点上运行。除了转换资源外，这个命令会将 **/etc/pacemaker/authkey** 复制到客户机节点上，并在客户机节点上启动并启用 **pacemaker_remote** 守护进程。客户机节点的节点名称（您可以随意定义）可以与节点的主机名不同。

```
# pcs cluster node add-guest nodename resource_id [options]
```

6. 创建 **VirtualDomain** 资源后，您可以像对待集群中的任何其他节点一样对待客户机节点。例如，您可以创建资源并在客户机节点中运行的资源上放置资源约束，如下命令可在集群节点中运行。您可以在组群中包含客户机节点，它们允许您对存储设备、文件系统和虚拟机进行分组。

```
# pcs resource create webserver apache configfile=/etc/httpd/conf/httpd.conf op
monitor interval=30s
# pcs constraint location webserver prefers nodename
```

30.3. 配置 PACEMAKER 远程节点

远程节点被定义为将 **ocf:pacemaker:remote** 作为资源代理的集群资源。您可以使用 **pcs cluster node add-remote** 命令创建此资源。

30.3.1. 远程节点资源选项

下表描述了您可以为 **remote** 资源配置的资源选项。

表 30.2. 远程节点的资源选项

项	默认值	描述
reconnect_interval	0	在到远程节点活跃连接断开后，在尝试重新连接到远程节点前等待的时间（以秒为单位）。这个等待是重复的。如果在等待时间过后重新连接失败，会在观察等待时间后进行一个新的重新连接尝试。当使用这个选项时，Pacemaker 会在每次等待的时间段内一直尝试退出并连接到远程节点。
server	使用 **pcs host auth** 命令指定的地址	要连接的服务器。这可以是 IP 地址或主机名。

项	默认值	描述
端口		要连接的 TCP 端口。

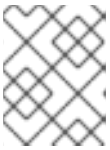
30.3.2. 远程节点配置概述

以下概述了配置 Pacemaker 远程节点并将该节点整合到现有 Pacemaker 集群环境中的步骤。

步骤

1. 在您要配置为远程节点的节点上，允许通过本地防火墙与集群相关的服务。

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



注意

如果您直接使用 **iptables**，或者 **firewalld** 以外的其他防火墙解决方案，只需打开以下端口：TCP 端口 2224 和 3121。

2. 在远程节点上安装 **pacemaker_remote** 守护进程。

```
# dnf install -y pacemaker-remote resource-agents pcs
```

3. 在远程节点上启动并启用 **pcsd**。

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

4. 如果您还没有这样做，请在要添加为远程节点的节点上验证 **pcs**。

```
# pcs host auth remote1
```

5. 使用以下命令在集群中添加远程节点资源。此命令还会将所有相关配置文件同步到新节点上，启动节点，并将其配置为在引导时启动 **pacemaker_remote**。这个命令必须运行在集群节点中，而不必在要添加的远程节点中运行。

```
# pcs cluster node add-remote remote1
```

6. 在集群中添加 **远程** 资源后，您可以像对待集群中的任何其他节点一样对待远程节点。例如，您可以创建资源并在远程节点中运行的资源上放置资源约束，如下命令可在集群节点中运行。

```
# pcs resource create webserver apache configfile=/etc/httpd/conf/httpd.conf op
monitor interval=30s
# pcs constraint location webserver prefers remote1
```

警告

资源组、colocation 约束或顺序约束中永远不会涉及远程节点连接资源。

7. 为远程节点配置保护资源。远程节点的隔离方式与集群节点相同。配置保护资源，以便使用与集群节点相同的远程节点。但请注意，远程节点永远不会启动隔离操作。只有群集节点能够真正对另一节点执行隔离操作。

30.4. 更改默认端口位置

如果您需要更改 Pacemaker 或 **pacemaker_remote** 的默认端口位置，您可以设置影响这两个守护进程的 **PCMK_remote_port** 环境变量。可以通过将其放在 **/etc/sysconfig/pacemaker** 文件中来启用该变量，如下所示：

```
\#==#==# Pacemaker Remote
...
#
# Specify a custom port for Pacemaker Remote connections
PCMK_remote_port=3121
```

当更改特定客户机节点或远程节点使用的默认端口时，必须在该节点的 **/etc/sysconfig/pacemaker** 文件中设置 **PCMK_remote_port** 变量，创建客户机节点或远程节点连接的集群资源也必须使用同样的端口号来配置（对客户机节点使用 **remote-port** 元数据选项，对远程节点使用 **port** 选项）。

30.5. 使用 PACEMAKER_REMOTE 节点升级系统

如果在活跃的 Pacemaker 远程节点上停止 **pacemaker_remote** 服务，集群将在停止节点前安全地迁移该节点的资源。这可让您在不从集群中删除节点的情况下执行软件升级和其他常规维护流程。关闭 **pacemaker_remote** 后，集群将立即尝试重新连接。如果 **pacemaker_remote** 在资源的监控器超时内没有重启，集群会将 monitor 操作视为失败。

如果您希望避免在活跃的 Pacemaker 远程节点上停止 **pacemaker_remote** 服务时监控失败，您可以在执行任何可能停止 **pacemaker_remote** 的系统管理前使用以下步骤使节点退出集群。

步骤

1. 使用 **pcs resource disable *resourcename*** 命令停止节点的连接资源，这样可将所有服务移出该节点。连接资源是远程节点的 **ocf:pacemaker:remote** 资源，通常为客户机节点的 **ocf:heartbeat:VirtualDomain** 资源。对于客户机节点，此命令也会停止虚拟机，因此虚拟机必须在集群外启动（例如，使用 **virsh**）以执行任何维护。

   ```
   pcs resource disable resourcename
   ```

2. 执行所需的维护。

3. 当准备好将节点返回集群时，使用 **pcs resource enable** 命令重新启用该资源。

   ```
   pcs resource enable resourcename
   ```

第 31 章 执行集群维护

要在集群的节点上执行维护，您可能需要停止或移动该集群中运行的资源和服务。或者，在不影响服务的同时，您可能需要停止集群软件。pacemaker 提供各种执行系统维护的方法。

- 如果您需要停止集群中的节点，同时继续提供在另一个节点中运行的服务，您可以让该集群节点处于待机模式。处于待机模式的节点无法再托管资源。该节点上任何当前活跃的资源都将移至另一节点，如果没有其他节点有资格运行该资源，则停止。有关待机模式的详情，请参考 将节点置于待机模式。

- 如果您需要在不停止该资源的情况下将资源从当前运行的节点中移除，您可以使用 **pcs resource move** 命令将资源移到其他节点上。
  执行 **pcs resource move** 命令时，这会向资源添加一个约束，以防止其在当前运行的节点上运行。当您准备重新移回资源时，可以执行 **pcs resource clear** 或 **pcs constraint delete** 命令来删除约束。这不一定将资源回原始节点，因为此时可以在哪里运行这些资源取决于您最初配置的资源。您可以使用 **pcs resource relocate run** 命令将资源重新定位到首选节点。

- 如果您需要完全停止正在运行的资源，并防止集群再次启动它，您可以使用 **pcs resource disable** 命令。有关 **pcs resource disable** 命令的详情，请参考 禁用、启用和禁止群集资源。

- 如果要防止 Pacemaker 对资源执行任何操作（例如，在对资源进行维护时禁用恢复操作，或者需要重新载入 **/etc/sysconfig/pacemaker** 设置），请使用 **pcs resource unmanage** 命令，如将资源设置为 unmanaged 模式所述。pacemaker 远程连接资源应该永远不是非受管状态。

- 如果您需要将集群置于不启动或停止服务的状态，您可以设置 **maintenance-mode** 集群属性。将集群放入维护模式会自动使所有资源为非受管状态。有关将集群置于维护模式的详情，请参考 将集群置于维护模式。

- 如果您需要更新组成 RHEL 高可用性和弹性存储附加组件的软件包，您可以一次在一个节点上更新软件包，或在整个集群中一起更新软件包，如更新 Red Hat Enterprise Linux 高可用性集群 中所述。

- 如果您需要在 Pacemaker 远程节点上执行维护操作，可以通过禁用远程节点资源从集群中删除该节点，如升级远程节点和客户机节点中所述。

- 如果您需要在 RHEL 集群中迁移虚拟机，首先需要停止虚拟机上的集群服务以从集群中删除该节点，然后在执行迁移后启动集群备份。如在 RHEL 集群中迁移虚拟机中所述。

31.1. 把节点设置为待机模式

当集群节点处于待机模式时，节点将不再能够托管资源。该节点上所有当前活跃的资源都将移至另一节点。

以下命令将指定节点设置为待机模式。如果您指定了 **--all**，该命令会将所有节点置于待机模式。

您可以在更新资源的软件包时使用此命令。您还可以在测试配置时使用此命令模拟恢复，而无需实际关闭节点。

```
pcs node standby node | --all
```

以下命令将指定节点从待机模式中删除。运行此命令后，指定节点就可以托管资源。如果您指定了 **--all**，该命令会将所有节点从待机模式中删除。

```
pcs node unstandby node | --all
```

请注意，当您执行 **pcs node standby** 命令时，这会阻止资源在指定节点上运行。当您执行 **pcs node unstandby** 命令时，这允许资源在指定节点上运行。这不一定将资源回指定节点；此时可以在哪里运行这些资源取决于您最初配置的资源。

31.2. 手动移动集群资源

您可以覆盖集群并强制资源从其当前位置移动。当您要做到这一点时有两个问题：

- 当某个节点处于维护状态时，您需要将该节点上运行的所有资源移至不同节点
- 当需要移动单独指定的资源时

要将节点上运行的所有资源移动到另一个节点，需要使该节点处于待机模式。

您可以用下列方式之一移动独立指定的资源。

- 您可以使用 **pcs resource move** 命令将资源从当前运行的节点中移出。
- 您可以使用 **pcs resource relocate run** 命令将资源移至首选节点，具体由当前的集群状态、约束、资源位置和其他设置来决定。

31.2.1. 将资源从其当前节点移动

要将资源从其当前运行的节点上移出，请使用以下命令，指定定义的资源的 *resource_id*。如果要指定在哪个节点上运行您要移动的资源，请指定 **destination_node**。

```
pcs resource move resource_id [destination_node] [--promoted] [--strict] [--wait[=n]]
```

执行 **pcs resource move** 命令时，这会向资源添加一个约束，以防止其在当前运行的节点上运行。默认情况下，在资源被移动后，命令创建的位置约束会被自动删除。如果删除约束将导致资源移回到原始节点，就像资源的 **resource-stickiness** 值为 0 才会发生的那样，那么 **pcs resource move** 命令将失败。如果您要移动资源并保留生成的约束，请使用 **pcs resource move-with-constraint** 命令。

如果您指定了 **pcs resource move** 命令的 **--promoted** 参数，则约束仅适用于提升资源实例。

如果您指定 **pcs resource move** 命令的 **--strict** 参数，则命令将失败，如果命令中指定的其他资源不受影响。

您可以选择为 **pcs resource move** 命令配置 **--wait[=*n*] 参数**，以指示在返回 0（如果资源已启动）或 1（如果资源尚未启动）之前，在目标节点上等待资源启动的秒数。如果没有指定 n，则默认为 60 分钟。

31.2.2. 将资源移动到首选节点

由于故障转移或管理员手动移动节点，在资源移动后，即使解决了造成故障转移的情况，它也不一定会迁移到其原始的节点。要将资源重新定位到首选节点，请使用以下命令。首选节点由当前的集群状态、约束、资源位置和其他设置决定,并可能随时间变化。

```
pcs resource relocate run [resource1] [resource2] ...
```

如果没有指定任何资源，则所有资源都会重新定位到首选节点。

此命令在忽略资源粘性时为每个资源计算首选的节点。在计算首选节点后，它会创建位置限制，导致资源移至首选节点。移动资源后，这些限制会自动被删除。要删除由 **pcs resource relocate run** 命令创建的所有约束，您可以输入 **pcs resource relocate clear** 命令。要显示资源的当前状态及其忽略资源粘性的最佳节点，请输入 **pcs resource relocate show** 命令。

31.3. 禁用、启用和禁止集群资源

除了 **pcs resource move** 和 **pcs resource relocate** 命令外，您还可以使用各种其他命令来控制集群资源的行为。

禁用集群资源
您可以手动停止正在运行的资源，并使用以下命令防止集群再次启动它。根据其他配置（约束、选项、失败等）配置，资源可能会继续启动。如果您指定了 **--wait** 选项，**pcs** 将最多等待资源停止 'n' 秒，然后返回 0 （如果资源停止）或 1（如果资源尚未停止）。如果没有指定 'n'，则默认为 60 分钟。

```
pcs resource disable resource_id [--wait[=n]]
```

在禁用资源不会影响其他资源的情况下，您可以指定一个资源被禁用。在具有复杂的资源关系时，这可能无法通过手动设置来完成。

- **pcs resource disable --simulate** 命令显示在更改集群配置的同时禁用资源的效果。
- **pcs resource disable --safe** 命令仅在没有任何方式影响任何其他资源时禁用资源，比如从一个节点迁移到另一个节点。**pcs resource secure-disable** 命令是 **pcs resource disable --safe** 命令的别名。
- **pcs resource disable --safe --no-strict** 命令仅在没有其他资源无法停止或降级时才禁用资源

您可以为 **pcs resource disable --safe** 命令指定 **--brief** 选项，以仅打印错误。如果安全禁用操作失败包含受影响的资源 ID，则 **pcs resource disable --safe** 命令会生成错误报告。如果您只需要知道受禁用资源影响的资源 ID，请使用 **--brief** 选项，这不提供完整的模拟结果。

启用集群资源
使用以下命令来允许集群启动资源。根据其余配置，资源可能会继续停止。如果您指定了 **--wait** 选项，**pcs** 将最多等待资源启动 'n' 秒，然后返回 0 （如果资源启动）或 1（如果资源尚未启动）。如果没有指定 'n'，则默认为 60 分钟。

```
pcs resource enable resource_id [--wait[=n]]
```

防止资源在特定节点上运行
使用以下命令来防止资源在指定节点上运行，如果没有指定节点则在当前节点上运行。

```
pcs resource ban resource_id [node] [--promoted] [lifetime=lifetime] [--wait[=n]]
```

请注意，当执行 **pcs resource ban** 命令时，这会向资源添加 -INFINITY 位置约束，以防止其在指定的节点上运行。您可以执行 **pcs resource clear** 或 **pcs constraint delete** 命令来删除约束。这不一定将资源回指定节点 ; 此时可以在哪里运行这些资源取决于您最初配置的资源。

如果您指定 **pcs resource ban** 命令的 **--promoted** 参数，则约束的范围仅限于提升的角色，您必须指定 *promotable_id* 而不是 *resource_id*。

您可选择为 **pcs resource ban** 命令配置 **lifetime** 参数，以指示约束应保留的时间。

您可以选择为 **pcs resource ban** 命令配置 **--wait[=*n*] 参数**，以指示在返回 0（如果资源已启动）或 1（如果资源尚未启动）之前，在目标节点上等待资源启动的秒数。如果没有指定 n，将使用默认的资源超时时间。

强制资源在当前节点上启动
使用 **pcs resource** 命令的 **debug-start** 参数强制指定资源在当前节点上启动，忽略集群建议并打印启动资源的输出。这主要用于调试资源；在集群上启动资源总是（几乎）由 Pacemaker 来完成，而不是直接

使用 **pcs** 命令。如果您的资源没有启动，这通常是由于资源配置错误（您在系统日志中调试）、阻止资源启动的限制，或者禁用资源。您可以使用这个命令来测试资源配置，但通常不应该用来启动集群中的资源。

debug-start 命令的格式如下：

```
pcs resource debug-start resource_id
```

31.4. 将资源设置为非受管模式

当资源处于 **unmanaged** 模式时，该资源仍然处于配置中，但 Pacemaker 不管理该资源。

以下命令将指定的资源设置为 **unmanaged** 模式。

```
pcs resource unmanage resource1 [resource2] ...
```

以下命令将资源设置为 **managed** 模式，这是默认状态。

```
pcs resource manage resource1 [resource2] ...
```

您可以使用 **pcs resource manage** 或 **pcs resource unmanage** 命令来指定资源组的名称。命令将对组中的所有资源执行操作，以便您可以通过单个命令将组中的所有资源设置为 **managed** 或 **unmanaged** 模式，然后单独管理包含的资源。

31.5. 将集群设置为维护模式

当集群处于维护模式时，除非有特别说明，集群不会启动或停止任何服务。当维护模式完成后，集群会对任何服务的当前状态进行完整性检查，然后停止或启动任何需要它的状态。

要将集群设置为维护模式，请使用以下命令将 **maintenance-mode** 集群属性设置为 **true**。

```
# pcs property set maintenance-mode=true
```

要从维护模式中删除集群，请使用以下命令将 **maintenance-mode** 集群属性设置为 **false**。

```
# pcs property set maintenance-mode=false
```

您可以使用以下命令从配置中删除集群属性。

```
pcs property unset property
```

另外，您可以通过将 **pcs property set** 命令的 value 字段留空来从配置中删除集群属性。这会将该属性恢复为默认值。例如，如果您之前将 **symmetric-cluster** 属性设置为 **false**，以下命令会从配置中删除您设置的值，并将 **symmetric-cluster** 的值恢复为 **true**，这是它的默认值。

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

31.6. 更新 RHEL 高可用性集群

可使用以下两种通用方法之一更新组成 RHEL High Availability 和 Resilient Storage 附加组件的软件包：

- *滚动更新*：一次从服务中移出一个节点，更新其软件，然后将其重新集成到集群中。这可让集群在更新每个节点时继续提供服务和管理资源。
- *整个集群更新*：停止整个集群，对所有节点应用更新，然后重新启动集群。

警告

在为 Red Hat Enterprise Linux High Availability 和 Resilient Storage 集群执行软件更新流程时，您必须确保在更新启动前，将进行更新的任何节点都不是集群的活跃成员。

有关以上方法及更新的流程的完整描述，请参阅将软件更新应用到 RHEL High Availability 或弹性存储集群的建议实践。

31.7. 升级远程节点和客户机节点

如果在活动远程节点或客户机节点上停止 **pacemaker_remote** 服务，则该集群将在停止该节点前安全地迁移该节点的资源。这可让您在不从集群中删除节点的情况下执行软件升级和其他常规维护流程。关闭 **pacemaker_remote** 后，集群将立即尝试重新连接。如果 **pacemaker_remote** 在资源的监控器超时内没有重启，集群会将 monitor 操作视为失败。

如果您希望避免在活跃的 Pacemaker 远程节点上停止 **pacemaker_remote** 服务时监控失败，您可以在执行任何可能停止 **pacemaker_remote** 的系统管理前使用以下步骤使节点退出集群。

步骤

1. 使用 **pcs resource disable *resourcename*** 命令停止节点的连接资源，这样可将所有服务移出该节点。连接资源是远程节点的 **ocf:pacemaker:remote** 资源，通常为客户机节点的 **ocf:heartbeat:VirtualDomain** 资源。对于客户机节点，此命令也会停止虚拟机，因此虚拟机必须在集群外启动（例如，使用 **virsh**）以执行任何维护。

   ```
   pcs resource disable resourcename
   ```

2. 执行所需的维护。
3. 当准备好将节点返回集群时，使用 **pcs resource enable** 命令重新启用该资源。

   ```
   pcs resource enable resourcename
   ```

31.8. 在 RHEL 集群中迁移虚拟机

红帽不支持在虚拟机监控程序或主机间实时迁移活跃集群节点，如 RHEL High Availability 集群支持政策 - 虚拟化集群成员的一般条件。如果需要执行实时迁移，首先需要停止虚拟机上的集群服务从集群中删除该节点，然后在执行迁移后启动集群备份。以下步骤概述了从集群中删除虚拟机、迁移虚拟机以及将虚拟机恢复到集群的步骤。

以下步骤概述了从集群中删除虚拟机、迁移虚拟机以及将虚拟机恢复到集群的步骤。

此流程适用于用作完整集群节点的虚拟机，不适用于作为集群资源（包括用作客户机节点的虚拟机）管理

的虚拟机，这些虚拟机可以在不需要特别计划的情况下进行实时迁移。有关单独或整体更新组成 RHEL 高可用性和弹性存储附加组件的软件包所需的完整流程的常规信息，请参阅将 将软件更新应用到 RHEL 高可用性或弹性存储集群的推荐实践。

注意

执行此步骤前，请考虑删除集群节点对集群仲裁的影响。例如，如果您有一个三节点集群，并且删除了一个节点，则集群无法承受任何节点故障。这是因为，如果三节点集群中的一个节点已经停机，删除第二个节点将会丢失仲裁。

步骤

1. 如果需要在停止或移动虚拟机上运行的资源或软件进行迁移前进行准备，请执行这些步骤。
2. 在虚拟机上运行以下命令来停止虚拟机上的集群软件。

```
# pcs cluster stop
```

3. 执行虚拟机的实时迁移。
4. 在虚拟机上启动集群服务。

```
# pcs cluster start
```

31.9. 通过 UUID 识别集群

从 Red Hat Enterprise Linux 9.1 开始，当您创建集群时，它有一个关联的 UUID。因为集群名称不是一个唯一的集群标识符，因此第三方工具，如管理具有相同名称的多个集群的配置管理数据库可以通过其 UUID 来唯一标识集群。您可以使用 **pcs cluster config [show]** 命令显示当前的集群 UUID，该命令在其输出中包含集群 UUID。

要向现有集群添加 UUID，请运行以下命令：

```
# pcs cluster config uuid generate
```

要为具有现有 UUID 的集群重新生成 UUID，请运行以下命令：

```
# pcs cluster config uuid generate --force
```

第 32 章 配置灾难恢复集群

为高可用性集群提供灾难恢复的一种方法是配置两个集群。然后，您可以将一个集群配置为主站点集群，第二个集群是灾难恢复集群。

在通常情况下，主集群在生产环境模式下运行资源。灾难恢复集群还配置了所有资源，且以降级模式运行或根本不运行。例如，在主集群中以提升模式运行数据库，且以降级模式在灾难恢复集群中运行。这个设置中的数据库将被配置，以便数据从主站点与灾难恢复网站同步。这是通过数据库配置本身进行的，而不是通过 **pcs** 命令界面完成。

当主集群停机时，用户可以使用 **pcs** 命令界面手动将资源切换到灾难恢复站点。然后，登录到灾难网站，升级并启动这些资源。主集群恢复后，用户可以使用 **pcs** 命令界面手动将资源重新移到主站点。

您可以使用 **pcs** 命令从其中一个站点的单个节点中显示主站点和灾难恢复站点集群的状态。

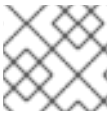
32.1. 灾难恢复集群的注意事项

在计划和配置要使用 **pcs** 命令管理并监控的灾难恢复网站时，请注意以下注意事项。

- 灾难恢复网站必须是一个集群。这样便可使用和主要站点相同的工具和相似过程配置它。
- 主集群和灾难恢复集群由独立的 **pcs cluster setup** 命令创建。
- 必须配置集群及其资源以便同步数据并可以进行故障切换。
- 恢复站点中的集群节点和主站点中的节点的名称不能相同。
- 在要运行 pcs 命令的节点上，必须针对两个集群中每个节点的 **pcs** 用户 **hacluster** 进行身份验证。

32.2. 显示恢复集群的状态

要配置主集群和灾难恢复集群，以便可以显示这两个集群的状态，请执行以下步骤。



注意

设置灾难恢复集群不会自动配置资源或复制数据。这些项目必须由用户手动配置。

在本例中：

- 主集群将命名为 **PrimarySite**，它由 **z1.example.com** 和 **z2.example.com** 组成。
- 灾难恢复站点集群将命名为 **DRsite**，它由 **z3.example.com** 和 **z4.example.com** 组成。

这个示例设置了一个没有配置资源或保护保护的基本集群。

步骤

1. 身份验证将用于这两个集群的所有节点。

```
[root@z1 ~]# pcs host auth z1.example.com z2.example.com z3.example.com
z4.example.com -u hacluster -p password
z1.example.com: Authorized
```

```
z2.example.com: Authorized
z3.example.com: Authorized
z4.example.com: Authorized
```

2. 创建用作集群的主集群并为集群启动集群服务的集群。

```
[root@z1 ~]# pcs cluster setup PrimarySite z1.example.com z2.example.com --start
{...}
Cluster has been successfully set up.
Starting cluster on hosts: 'z1.example.com', 'z2.example.com'...
```

3. 创建用作灾难恢复集群的集群，为集群启动集群服务。

```
[root@z1 ~]# pcs cluster setup DRSite z3.example.com z4.example.com --start
{...}
Cluster has been successfully set up.
Starting cluster on hosts: 'z3.example.com', 'z4.example.com'...
```

4. 从主集群中的一个节点中，将第二个集群设置为恢复站点。该恢复站点由其中一个节点的名称定义。

```
[root@z1 ~]# pcs dr set-recovery-site z3.example.com
Sending 'disaster-recovery config' to 'z3.example.com', 'z4.example.com'
z3.example.com: successful distribution of the file 'disaster-recovery config'
z4.example.com: successful distribution of the file 'disaster-recovery config'
Sending 'disaster-recovery config' to 'z1.example.com', 'z2.example.com'
z1.example.com: successful distribution of the file 'disaster-recovery config'
z2.example.com: successful distribution of the file 'disaster-recovery config'
```

5. 检查灾难恢复配置。

```
[root@z1 ~]# pcs dr config
Local site:
  Role: Primary
Remote site:
  Role: Recovery
  Nodes:
    z3.example.com
    z4.example.com
```

6. 检查主集群的状态以及主集群中节点的灾难恢复集群。

```
[root@z1 ~]# pcs dr status
--- Local cluster - Primary site ---
Cluster name: PrimarySite

WARNINGS:
No stonith devices and stonith-enabled is not false

Cluster Summary:
  * Stack: corosync
  * Current DC: z2.example.com (version 2.0.3-2.el8-2c9cea563e) - partition with quorum
  * Last updated: Mon Dec  9 04:10:31 2019
  * Last change:  Mon Dec  9 04:06:10 2019 by hacluster via crmd on z2.example.com
```

```
  * 2 nodes configured
  * 0 resource instances configured

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

Full List of Resources:
  * No resources

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


--- Remote cluster - Recovery site ---
Cluster name: DRSite

WARNINGS:
No stonith devices and stonith-enabled is not false

Cluster Summary:
  * Stack: corosync
  * Current DC: z4.example.com (version 2.0.3-2.el8-2c9cea563e) - partition with quorum
  * Last updated: Mon Dec  9 04:10:34 2019
  * Last change:  Mon Dec  9 04:09:55 2019 by hacluster via crmd on z4.example.com
  * 2 nodes configured
  * 0 resource instances configured

Node List:
  * Online: [ z3.example.com z4.example.com ]

Full List of Resources:
  * No resources

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

有关灾难恢复配置的其他显示选项，请参阅 **pcs dr** 命令的帮助页面。

第 33 章 解释资源代理 OCF 返回代码

Pacemaker 资源代理符合 Open Cluster Framework(OCF)资源代理 API。下表描述了 OCF 返回代码，以及 Pacemaker 如何解释它们。

当代理返回代码时，集群要做的第一件事是针对预期结果检查返回码。如果结果与预期值不匹配，则操作被视为失败，并启动恢复操作。

对于任何调用，资源代理必须以定义的返回码退出，该返回码告知调用者调用操作的结果。

故障恢复有三种类型，如下表所述。

表 33.1. 集群执行的恢复类型

Type	描述	集群采取的操作
soft	发生了短暂错误。	重新启动资源或将其移到新位置。
hard	发生了一个可能特定于当前节点的非短暂错误。	将资源移到其他地方，并阻止其在当前节点上被重试。
fatal	发生了一个对所有集群节点都常见的非短暂错误（例如，指定了一个错误的配置）。	停止资源，并阻止其在任何集群节点上启动。

下表提供了 OCF 返回码，以及收到失败代码时集群将启动的恢复类型。请注意，如果 0 不是预期的返回值，即使返回 0 (OCF 别名 **OCF_SUCCESS**)的操作也可被视为失败。

表 33.2. OCF 返回码

返回码	OCF 标签	描述
0	**OCF_SUCCESS**	* 操作成功完成。这是任何成功的 start、stop、promote 和 demote 命令的预期返回码。 * Type if unexpected: soft
1	**OCF_ERR_GENERIC**	* 操作返回一个通用错误。 * 类型： soft * 资源管理器将尝试恢复资源或将其移到新位置。
2	**OCF_ERR_ARGS**	* 资源的配置在此机器中无效。例如，它指向了一个节点上未找到的位置。 * Type: hard * 资源管理器会将资源移到其他位置，并阻止其在当前节点上重试
3	**OCF_ERR_UNIMPLEMENTED**	* 请求的操作没有实施。 * Type: hard

返回码	OCF 标签	描述
4	**OCF_ERR_PERM**	* 资源代理没有足够的权限来完成该任务。例如，这可能是因为代理无法打开特定文件、监听特定套接字或写入目录。 * Type: hard * 除非另有特殊配置，否则资源管理器将通过在其他节点（权限问题可能不存在）上重启资源来尝试恢复出错的资源。
5	**OCF_ERR_INSTALLED**	* 执行操作的节点中缺少所需的组件。这可能是因为所需的二进制文件不可执行，或者重要的配置文件不可读。 * Type: hard * 除非另有特殊配置，否则资源管理器将通过在其他节点（可能存在所需的文件或二进制文件）上重启资源来尝试恢复出错的资源。
6	**OCF_ERR_CONFIGURED**	* 本地节点上的资源配置无效。 * type: fatal * 当返回此码时，Pacemaker 将阻止资源在集群中的任何节点上运行，即使服务配置在某些其他节点上是有效的。
7	**OCF_NOT_RUNNING**	* 资源被安全地停止。这意味着资源已正常关闭，或者从未启动。 * Type if unexpected: soft * 集群不会尝试停止在任何操作中返回此功能的资源。
8	**OCF_RUNNING_PROMOTED**	* 资源在提升的角色中运行。 * Type if unexpected: soft
9	**OCF_FAILED_PROMOTED**	* 资源是（或可能）在提升角色中，但失败。 * 类型： soft * 资源将被降级、停止，然后再次启动（并可能升级）。
190		* 发现该服务正确活跃，但在这种情况下，未来故障的可能性更大。
191		* 资源代理支持角色和服务，发现在提升的角色中正确活跃，但在这样的情况下，未来故障的可能性更大。

返回码	OCF 标签	描述
其他	不适用	自定义错误码.

第 34 章 使用 IBM Z/VM 实例将红帽高可用性集群配置为集群成员

红帽提供了几篇文章，在设计、配置和管理在 z/VM 虚拟机上运行的红帽高可用性集群时非常有用。

- RHEL 高可用性集群设计指南 - IBM z/VM 实例作为群集成员
- RHEL 高可用性集群的管理流程 - 为 RHEL 7 或 8 IBM z 系统集群成员配置带有 fence_zvmip 的 z/VM SMAPI 隔离
- IBM z Systems 上的 RHEL High Availability 集群节点在夜间经历 STONITH-device 超时
- RHEL 高可用性集群的管理程序 - 为 IBM z Systems 成员集群准备 dasd 存储设备

一般情况下，您可能还会找到以下在设计红帽高可用性集群时有用的文章。

- RHEL 高可用性集群的支持政策
- 探索 RHEL 高可用性集群的概念 - Fencing/STONITH