Red Hat Enterprise Linux 7

High Availability Add-On 参考

配置和管理高可用性附加组件参考指南

Last Updated: 2021-09-14

Red Hat Enterprise Linux 7 High Availability Add-On 参考

配置和管理高可用性附加组件参考指南

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律通告

摘要

红帽高可用性附加组件参考提供有关为红帽企业 Linux 7 安装、配置和管理红帽高可用性附加组件的参考信息。

目录

第 1 章 红帽高可用性附加组件配置和管理参考概述

本文档提供了使用 Pacemaker 的红帽高可用性附加组件支持的选项和功能。*有关步骤基本配置示例的步骤，请参阅红帽高可用性附加组件管理。*

您可以使用 pcs 配置界面或使用 **pcs d GUI 界面配置红帽高可用性附加组件集群。**

1.1. 新的和更改的功能

本节列出了 Red Hat High Availability Add-On 自 Red Hat Enterprise Linux 7 初始版本之后的新功能。

1.1.1. Red Hat Enterprise Linux 7.1 的新功能和改变的功能

红帽企业 Linux 7.1 包含以下文档和功能更新和更改：

- **pcs resource cleanup** 命令现在可以重置所有资源的资源状态和故障计数，如 第 6.11 节 "集群资源清理" 所述。
- 您可以为 **pcs resource move** 命令指定一个 生命周期 参数，如 第 8.1 节 "手动在集群中移动资源" 所述。
- 从红帽企业 Linux 7.1 开始，您可以使用 **pcs acl** 命令为本地用户设置权限，以允许通过使用访问控制列表(ACL)对集群配置进行只读或读写访问。有关 ACL 的详情请参考 第 4.5 节 "设置用户权限"。
- 第 7.2.3 节 "排序的资源集" 并且 第 7.3 节 "资源共存" 已进行了广泛更新和修改。
- 第 6.1 节 "资源创建" 文档 **pcs resource create** 命令的 **disabled** 参数，以指示正在创建的资源没有自动启动。
- 第 10.1 节 "配置仲裁选项" 记录新的群集仲裁未阻塞功能，这会阻止集群在建立仲裁时等待所有节点。
- 第 6.1 节 "资源创建" 记录 **pcs resource create** 命令 前和 之后 参数的，该命令可用于配置资源组顺序。
- 从 Red Hat Enterprise Linux 7.1 发行版本开始，您可以使用 **pcs config** 命令的 备份和恢复 选项，在 tarball 中备份集群配置，并在 所有节点上恢复集群配置文件。有关这个功能的详情请参考 第 3.8 节 "备份和恢复集群配置"。
- 本文通篇给出了少量说明。

1.1.2. Red Hat Enterprise Linux 7.2 的新功能和改变的功能

红帽企业 Linux 7.2 包含以下文档和功能更新及更改：

- 现在，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、限制、资源位置和其他设置决定。有关这个命令的详情请参考 第 8.1.2 节 "将资源移动到首选节点"。
- 第 13.2 节 "使用监控资源的事件通知" 已修改并扩展，以更好地了解如何配置 **ClusterMon** 资源来执行外部程序，以确定如何处理群集通知。
- 在为冗余电源配置隔离时，现在只需要为每个设备定义一次，并指定两个设备都需要隔离该节点。有关为冗余电源配置隔离的详情请参考 第 5.10 节 "为冗余电源配置隔离"。

- 本文档现在提供了将节点添加到 第 4.4.3 节 "添加集群节点" 中现有集群的步骤。
- 新的 resource-discovery 位置约束选项允许您指定 Pacemaker 是否应该为指定资源在节点上执行资源发现，如 表 7.1 "简单位置限制选项" 所述。
- 本文通篇都进行少量说明和纠正。

1.1.3. Red Hat Enterprise Linux 7.3 的新功能和改变的功能

Red Hat Enterprise Linux 7.3 包含以下文档和功能更新和更改。

- 第 9.4 节 "pacemaker_remote 服务"已针对此版本的文档完全重写。
- 您可以使用警报代理来配置 Pacemaker 警报，它们是集群调用的外部程序，其方式与集群调用的资源代理相同，以处理资源配置和操作。Pacemaker 警报代理在 第 13.1 节 "Pacemaker 警报代理（红帽企业 Linux 7.3 及更新的版本）" 中描述。
- 此发行版本支持新的仲裁管理命令，允许您显示仲裁状态并更改 expected_votes 参数。这些命令在 第 10.2 节 "仲裁管理命令（Red Hat Enterprise Linux 7.3 及稍后）"中描述。
- 现在，您可以使用 pcs quorum update 命令修改集群的常规仲裁选项，如 第 10.3 节 "修改仲裁选项（红帽企业 Linux 7.3 及更新的版本）" 所述。
- 您可以配置作为集群的第三方设备的独立仲裁设备。这个功能的主要用途是允许集群保持比标准仲裁规则允许更多的节点故障。此功能仅提供给技术预览。有关仲裁设备的详情请参考 第 10.5 节 "仲裁设备"。
- Red Hat Enterprise Linux release 7.3 提供了通过使用 Booth 集群票据管理器配置跨多个站点的高可用性集群的功能。此功能仅提供给技术预览。有关 Booth 集群票据管理器的详情请参考 第 14 章 *使用 Pacemaker 配置多站点集群*。
- 在配置运行 pacemaker_remote 服务的 KVM 虚拟客户机节点时，您可以将客户机节点包含在组中，这允许您对存储设备、文件系统和虚拟机进行分组。有关配置 KVM 客户机节点的详情请参考 第 9.4.5 节 "配置概述：KVM 客户机节点"。

此外，本文通篇还进行少量说明和纠正。

1.1.4. Red Hat Enterprise Linux 7.4 的新功能和改变的功能

红帽企业 Linux 7.4 包括以下文档和功能更新及更改：

- Red Hat Enterprise Linux release 7.4 提供了全面支持，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。有关 Booth 集群票据管理器的详情请参考 第 14 章 *使用 Pacemaker 配置多站点集群*。
- Red Hat Enterprise Linux 7.4 完全支持配置作为集群的第三方设备的独立仲裁设备。这个功能的主要用途是允许集群保持比标准仲裁规则允许更多的节点故障。有关仲裁设备的详情请参考 第 10.5 节 "仲裁设备"。
- 现在，您可以通过在节点名称、节点属性及其值中应用的正则表达式，在隔离拓扑中指定节点。有关配置隔离级别的详情请参考 第 5.9 节 "配置隔离级别"。
- Red Hat Enterprise Linux 7.4 支持 NodeUtilization 资源代理，它可以检测可用 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。有关此资源代理的详情请参考 第 9.6.5 节 "NodeUtilization 资源代理（红帽企业 Linux 7.4 及更高版本）"。

- 对于红帽企业 Linux 7.4，群集节点 **add-guest** 和群集节点 **remove-guest** 命令取代了群集 **remote-node add** 和群集远程节点删除命令。**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**。有关更新的客户机和远程节点配置过程，请参阅 第 9.3 节 "将虚拟域配置为资源"。

- Red Hat Enterprise Linux 7.4 支持 **systemd resource-agents-deps** 目标。这可让您为集群配置适当的启动顺序，其中包含不是由集群管理的依赖项的资源，如 第 9.7 节 "为不由 Pacemaker 管理的资源依赖项配置启动顺序（Red Hat Enterprise Linux 7.4 及更新的版本）" 所述。

- 本发行版本中更改了将资源创建为主/从克隆的命令格式。有关创建 master/从克隆的详情请参考 第 9.2 节 "多状态资源：具有多个模式的资源"。

1.1.5. Red Hat Enterprise Linux 7.5 的新功能和改变的功能

红帽企业 Linux 7.5 包含以下文档和功能更新及更改：

- 从 Red Hat Enterprise Linux 7.5 开始，您可以使用 **pcs_snmp_agent** 守护进程通过 SNMP 查询 Pacemaker 集群的数据。有关使用 SNMP 查询集群的详情请参考 第 9.8 节 "使用 SNMP 查询 Pacemaker 集群（Red Hat Enterprise Linux 7.5 及更新的版本）"。

1.1.6. Red Hat Enterprise Linux 7.8 的新功能和改变的功能

Red Hat Enterprise Linux 7.8 包括以下文档和功能更新和更改。

- 从 Red Hat Enterprise Linux 7.8 开始，您可以配置 Pacemaker，以便在节点完全关闭时，附加到该节点的资源将锁定到该节点，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重新启动。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。有关将资源配置为在清理节点关闭时保持停止的详情请参考 第 9.9 节 " 配置资源以保持在 Clean Node Shutdown 上停止（红帽企业 Linux 7.8 及更新的版本） "。

1.2. 安装 PACEMAKER 配置工具

您可以使用以下 **yum install** 命令安装红帽高可用性附加组件软件包，以及 High Availability 频道中所有可用的隔离代理。

```
# yum 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 软件包以及只安装您需要的隔离代理。

```
# yum 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
...
```

lvm2-cluster 和 **gfs2-utils** 软件包是 ResilientStorage 频道的一部分。您可以根据需要使用以下命令安装它们：

```
# yum install lvm2-cluster gfs2-utils
```

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。

1.3. 配置 IPTABLES 防火墙以允许集群组件

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。

表 1.1 "为高可用性附加组件启用的端口" 显示要为红帽高可用性附加组件启用的端口，并解释该端口的用途。您可以通过执行下列命令，通过利用 firewalld 守护进程启用所有这些端口：

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

表 1.1. 为高可用性附加组件启用的端口

端口	什么时候需要
TCP 2224	**所有节点上都需要（pcsd** Web UI 需要且节点到节点的通信需要） 打开端口 2224 非常重要，从而使来自任何节点的 **pcs** 可以与群集中的所有节点（包括自身）进行通信。当使用 Booth 集群票据管理程序或一个 quorum 设备时，您必须在所有相关主机上打开端口 2224，比如 Booth abiter 或者 quorum 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 完整集群节点上的 Pacemaker 的 **crmd** 守护进程将在端口 3121 联系 Pacemaker 远程节点上的 **pacemaker_remoted** 守护进程。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。由于用户可以在完整节点和远程节点之间转换主机，或使用主机的网络在容器内运行远程节点，因此打开所有节点的端口会很有用。不需要向节点以外的任何主机打开端口。
TCP 5403	当使用带有 **corosync-qnetd** 的仲裁设备时，quorum 设备主机上需要此项。可以使用 **corosync-qnetd** 命令的 **-p** 选项更改默认值。
UDP 5404	如果为多播 UDP 配置 **corosync，则 corosync** 节点需要
UDP 5405	所有 corosync 节点上都需要（corosync 需要 ）

端口	什么时候需要
TCP 21064	如果群集包含任何需要 DLM 的资源（如 **clvm** 或 **GFS2**），则在所有节点上都需要这个端口。
TCP 9929, UDP 9929	需要在所有集群节点上打开，并在使用 Booth ticket 管理器建立多站点集群时引导节点从这些相同节点进行连接。

1.4. 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**（**Pacemaker** 构建的集群管理器）使用的集群参数。通常，您不应该直接编辑 **corosync.conf**，而是使用 **pcs** 或 **pcsd** 接口。但是，在某些情况下，您可能需要直接编辑此文件。有关编辑 **corosync.conf** 文件的详情，请参考在 Red Hat Enterprise Linux 7 中编辑 corosync.conf 文件。

cib.xml 文件是一个 XML 文件，它代表群集的配置和群集中所有资源的当前状态。Pacemaker 的集群信息基础(CIB)使用此文件。CIB 的内容在整个群集间自动保持同步，请勿直接编辑 **cib.xml** 文件；改为使用 **pcs** 或 **pcsd** 接口。

1.5. 集群配置注意事项

在配置 Red Hat High Availability Add-On 集群时，您必须考虑以下事项：

- 红帽不支持 RHEL 7.7（及更高版本）的集群部署超过 32 个节点。但是，通过运行 **pacemaker_remote** 服务的远程节点，有可能超出这一限制。有关 **pacemaker_remote** 服务的详情请参考 第 9.4 节 “pacemaker_remote 服务”。
- 不支持使用动态主机配置协议(DHCP)在 **corosync** 守护进程使用的网络接口上获取 IP 地址。DHCP 客户端可以在地址续订期间定期删除 IP 地址并重新为其分配接口重新添加 IP 地址。这将导致 **corosync** 检测 连接失败，这将导致对群集中其他任何节点进行 心跳 连接的隔离活动。

1.6. 更新红帽企业 LINUX 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*：从服务中删除一个节点，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。
- *更新整个集群*：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 集群执行软件更新步骤时，您必须确保在更新启动前，任何进行更新的节点都不是集群的活跃成员。

有关每个方法以及更新的步骤的完整描述，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。

1.7. RHEL 集群中实时迁移虚拟机的问题

有关使用虚拟化集群成员的 RHEL 高可用性集群支持政策的信息，请参阅 RHEL 高可用性集群的支持政策 - 虚拟化集群成员的一般条件。如前所述，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。

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，如果您有一个三个节点集群，并且删除了一个节点，则集群只能有一个节点失败。如果三个节点群集中的一个节点已经停机，删除第二个节点将丢失仲裁。

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. 将任何受管资源移出虚拟机。如果应当重新定位资源的具体要求或首选项，请考虑创建新的位置限制，以将资源放置在正确的节点上。
3. 将虚拟机置于待机模式以确保它不被视为服务，并导致任何剩余的资源重新定位到其他位置或停止。

   ```
   # pcs cluster standby VM
   ```

4. 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
   ```

5. 执行虚拟机的实时迁移。
6. 在虚拟机上启动集群服务。

   ```
   # pcs cluster start
   ```

7. 将虚拟机移出待机模式。

   ```
   # pcs cluster unstandby VM
   ```

8. 如果您在将虚拟机置于待机模式之前创建了任何临时位置限制，请调整或删除这些限制，以允许资源返回到通常首选的位置。

第 2 章 PCSD WEB UI

本章概述了使用 pcsd Web UI 配置红帽高可用性群集。

2.1. PCSD WEB UI 设置

要将您的系统设置为使用 pcsd Web UI 配置群集，请使用以下步骤：

1. 安装 Pacemaker 配置工具，如 第 1.2 节 “安装 Pacemaker 配置工具” 所述。
2. 在将成为群集一部分的每个节点上，使用 passwd 命令设置用户 hacluster 的密码，并且在每个节点上使用相同的密码。
3. 在每个节点中启动并启用 pcsd 守护进程：

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

4. 在集群的一个节点上，使用以下命令验证组成集群的节点。执行此命令后，系统将 提示您输入 用户名和密码。将 hacluster 指定为 Username。

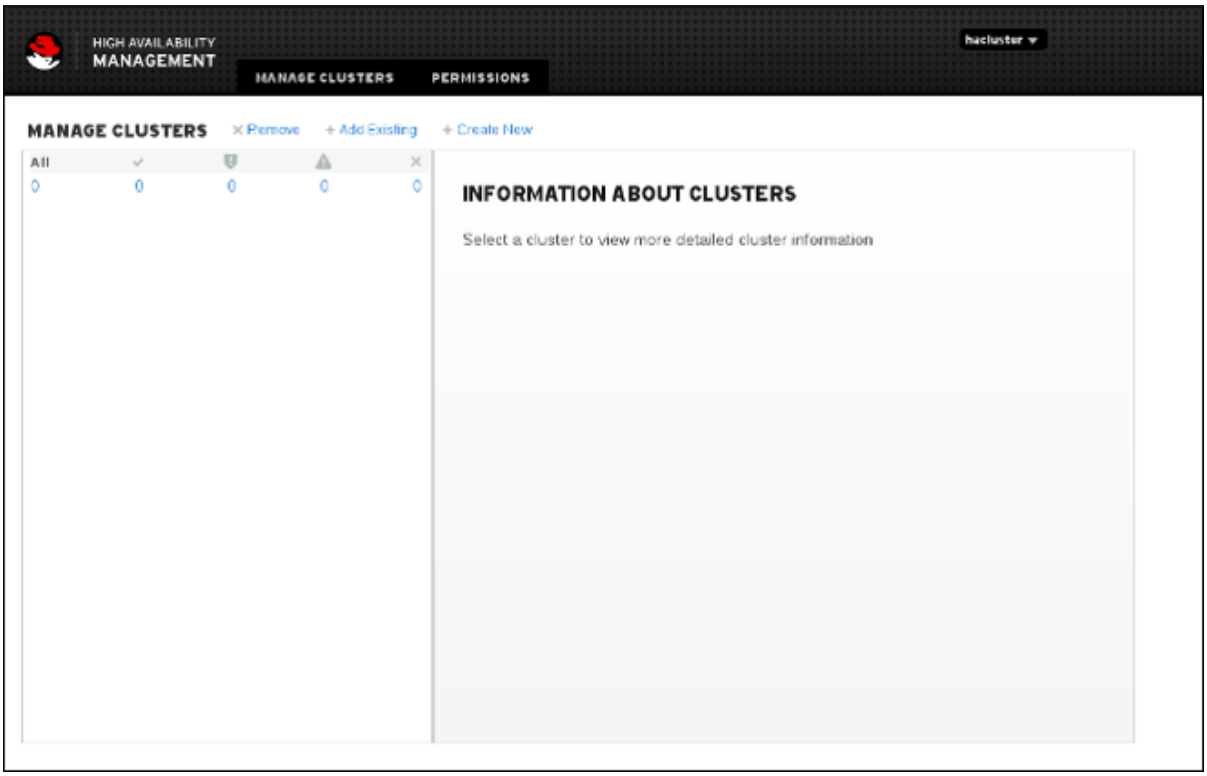
```
# pcs cluster auth node1 node2 ... nodeN
```

5. 在任意系统上，打开浏览器到以下 URL，指定您授权的一个节点（请注意，这使用 https 协议）。这将调出 pcsd Web UI 登录屏幕。

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

6. 以用户 hacluster 身份登录。此时会出现管理集群页面，如 图 2.1 “管理集群页面” 所示。

图 2.1. 管理集群页面



[D]

2.2. 使用 PCSD WEB UI 创建集群

在 Manage Clusters 页面中，您可以创建新集群，将现有集群添加到 Web UI 中，或者从 Web UI 中删除集群。

- 要创建集群，请点击 **Create New** 并输入要创建的集群的名称以及组成集群的节点。您还可以在此屏幕中配置高级集群选项，包括集群通信的传输机制，如 第 2.2.1 节 “高级集群配置选项” 所述。输入集群信息后，点 **Create Cluster**。
- 要将现有集群添加到 Web UI 中，请点击 **Add Existing**，并输入您要使用 Web UI 管理的集群中的节点的主机名或 IP 地址。

创建或添加集群后，会在管理集群页面中显示集群名称。选择集群会显示有关集群的信息。

注意

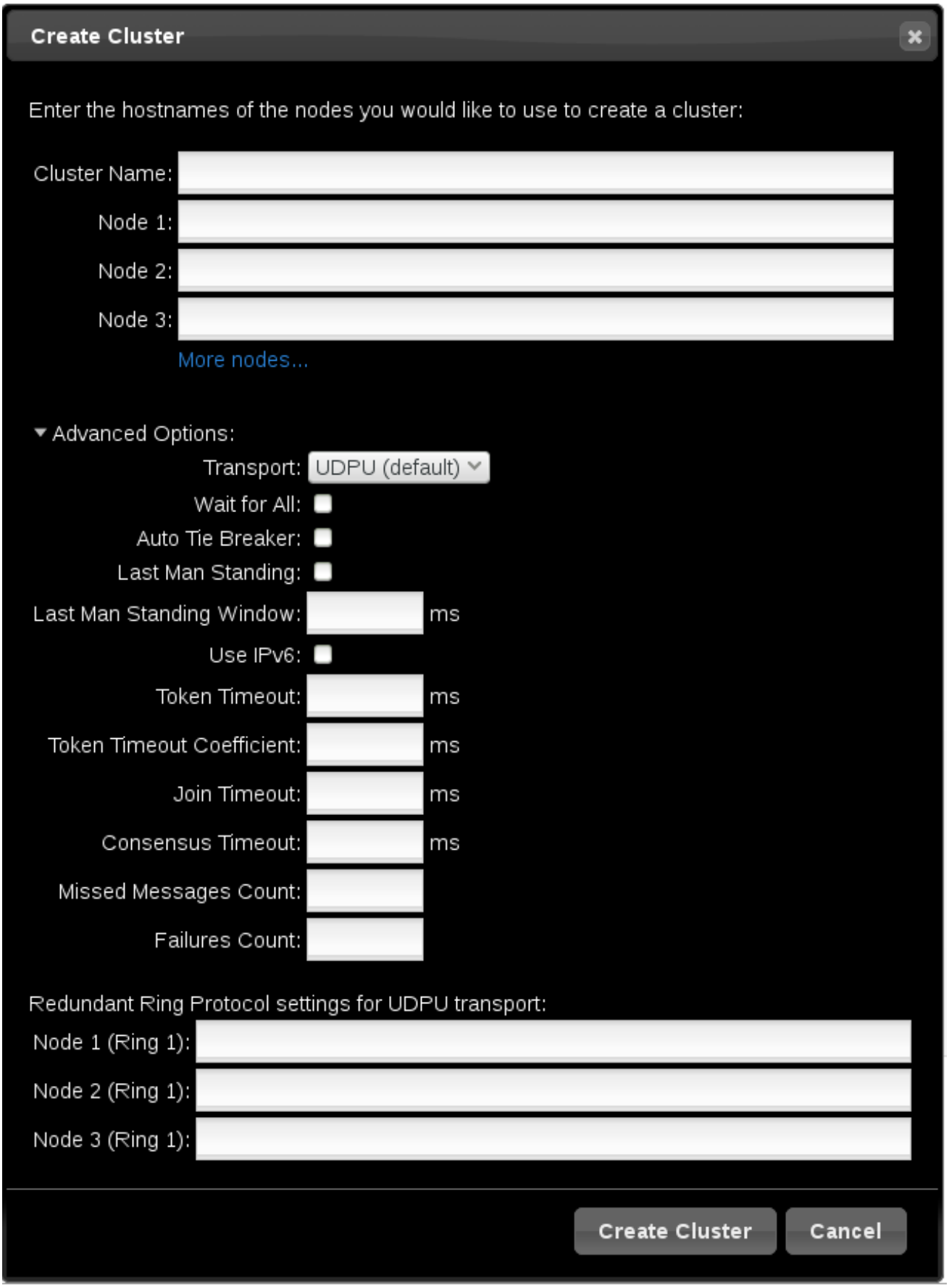
当使用 **pcsd** Web UI 配置集群时，您可以将鼠标移到文本描述中，以作为工具提示 显示这些选项的较长描述。

2.2.1. 高级集群配置选项

在创建集群时，您可以点 Advanced Options 来配置附加集群选项，如 图 2.2 “创建 Clusters 页面” 所示。有关所显示选项的信息，请将鼠标移到该选项的文本上。

请注意，您可以通过为每个节点指定接口来配置使用冗余环协议的集群。如果您选择 UDP，而不是作为群集的传输机制的 UDP U 的默认值，则将显示冗余环协议设置。

图 2.2. 创建 Clusters 页面



[D]

2.2.2. 设置集群管理权限

您可以向用户授予两组集群权限：

- 使用 Web UI 管理集群的权限，它还授予运行通过网络连接到节点的 pcs 命令的权限。本节论述了如何使用 Web UI 配置这些权限。
- 本地用户使用 ACL 允许只读或读写访问集群配置的权限。第 2.3.4 节 "配置 ACL" 中描述了使用 Web UI 配置 ACL。

有关用户权限的详情请参考 第 4.5 节 "设置用户权限"。

您可以为用户 hacluster 以外的特定用户授予权限，以便通过 Web UI 管理集群，并运行pcs 命令通过将它们添加到组 haclient 来运行通过网络连接到节点的 pcs 命令。然后，您可以通过单击 Manage Clusters 页面上的 Permissions 选项卡，并在结果屏幕上设置权限，为组 haclient 的单个成员配置权限集。在这个页面中，您还可以为组群设置权限。

您可以授予以下权限：

- 查看集群设置的读取权限
- 写入权限，修改集群设置（权限和 ACL 除外）
- 授予权限以修改集群权限和 ACL
- 对集群的不受限制访问（包括添加和删除节点）的所有权限，并可访问密钥和证书

2.3. 配置集群组件

要配置集群的组件和属性，请点击 Manage Clusters 屏幕上显示的集群名称。这会显示 Nodes 页面，如 第 2.3.1 节 "集群节点" 所述。此页面在页面顶部显示一个菜单，如 图 2.3 "集群组件菜单" 所示，包括以下条目：

- 节点，如 所述 第 2.3.1 节 "集群节点"
- 资源，如 所述 第 2.3.2 节 "集群资源"
- 隔离设备，如 所述 第 2.3.3 节 "隔离设备"
- ACL，如 所述 第 2.3.4 节 "配置 ACL"
- 集群属性，如 所述 第 2.3.5 节 "集群属性"

图 2.3. 集群组件菜单

[D]

2.3.1. 集群节点

从集群管理页面顶部的菜单中选择 Nodes 选项会显示当前配置的节点和当前选定节点的状态，包括节点上运行哪些资源以及资源位置首选项。这是从 Manage Clusters 屏幕中选择集群时显示的默认页面。

您可以在此页面中添加或删除节点，您可以启动、停止、重启或将节点设置为待机模式。有关待机模式的详情请参考 第 4.4.5 节 "待机模式"。

您还可以直接在这个页面中配置隔离设备，如 第 2.3.3 节 "隔离设备" 所述，选择 Configure Fencing。

2.3.2. 集群资源

在集群管理页面顶部的菜单中选择 **Resources** 选项显示当前为集群配置的资源，并根据资源组进行组织。选择组或资源会显示该组或资源的属性。

在本页中，您可以添加或删除资源，您可以编辑现有资源的配置，您可以创建资源组。

若要在集群中添加新资源，请单击 **Add**。这会显示 **Add Resource** 屏幕。从类型下拉菜单中选择资源类型时，必须为该资源指定的参数将显示在菜单中。您可以点击可选参数来显示您可以为您要定义的资源指定的其他参数。为您要创建的资源输入参数后，点 **Create Resource**。

当为资源配置参数时，会在菜单中显示参数的简单描述。如果您将光标移动到字段，就会显示一个较长的帮助信息。

您可以将 作为资源定义为克隆的资源，或定义为主/从资源。有关这些资源类型的详情请参考 第 9 章 *高级配置*。

至少创建了一个资源后，您可以创建一个资源组。有关资源组的详情请参考 第 6.5 节 “资源组”。

若要创建资源组，可从 **Resources** 屏幕选择属于组的资源，然后单击 **Create Group**。这将显示 **Create Group** 屏幕。输入组名称，再单击 **Create Group**。这会返回到 **Resources** 屏幕，现在显示资源的组名称。创建资源组后，您可以在创建或修改其他资源时将组名称指定为资源参数。

2.3.3. 隔离设备

在集群管理页面顶部的菜单中选择 **Fence Devices** 选项会显示 **Fence Devices** 屏幕，显示当前配置的隔离设备。

要在集群中添加新隔离设备，点 **Add**。这会显示 **Add Fence Device** 屏幕。当您从 **Type** 下拉菜单中选择隔离设备类型时，您必须为该隔离设备指定的参数会出现在菜单中。您可以点 **Optional Arguments** 来显示您可以为您要定义的隔离设备指定的附加参数。为新隔离设备输入参数后，点 **Create Fence Instance**。

有关使用 Pacemaker 配置隔离设备的详情请参考 第 5 章 *隔离：配置 STONITH*。

2.3.4. 配置 ACL

在集群管理页面顶部的菜单中选择 **ACLS** 选项会显示一个屏幕，您可以在其中为本地用户设置权限，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。

要分配 ACL 权限，您可以创建一个角色并为该角色定义访问权限。每个角色都可以有无限数量的、适用于 XPath 查询或者一个特定元素的 ID 的权限（读/写/拒绝）。定义角色后，您可以将其分配给现有用户或组群。

2.3.5. 集群属性

在集群管理页面顶部的菜单中选择 **Cluster Properties** 选项会显示集群属性，并允许您从默认值中修改这些属性。有关 Pacemaker 集群属性的详情请参考 第 12 章 *Pacemaker 集群属性*。

2.4. 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以显示集群管理页面。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，可以配置 pcsd Web UI 本身以实现高可用性，在这种情况下，您可以继续管理集群而无需输入新 URL。

要配置 pcsd Web UI 以实现高可用性，请执行以下步骤：

1. 确保在 /etc/sysconfig/pcsd 配置文件中将 PCSD_SSL_CERT_SYNC_ENABLED 设置为 true，这是 RHEL 7 中的默认值。启用证书同步会导致 pcsd 为群集设置和节点添加命令同步 pcsd 证书。

2. 创建一个 IPaddr2 群集资源，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 必须位于与节点静态分配的 IP 地址之一相同的网络中，否则无法正确检测到分配浮动 IP 地址的 NIC 设备。

3. 为 pcsd 创建自定义 SSL 证书，并确保它们对用于连接到pcsd Web UI 的节点地址有效。

   a. 要创建自定义 SSL 证书，您可以使用通配符证书，或者使用 Subject 备用名称证书扩展。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 命令安装 pcsd 的自定义证书。

   c. 使用 pcs pcsd sync-certificates 命令将 pcsd 证书同步到群集中的所有节点。

4. 使用您配置为群集资源的浮动 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 3 章 PCS 命令行界面

pcs 命令行界面通过提供 corosync .conf 和 cib.xml 文件的接口来控制和配置 corosync 和 Pacemaker。

pcs 命令的一般格式如下：

```
pcs [-f file] [-h] [commands]...
```

3.1. PCS 命令

pcs 命令如下所示：

- **cluster**

  配置群集选项和节点.有关 pcs cluster 命令的详情请参考 第 4 章 *集群创建和管理*。

- **resource**

  创建和管理群集资源。有关 pcs cluster 命令的详情请参考 第 6 章 *配置集群资源*、第 8 章 *管理集群资源* 和 第 9 章 *高级配置*。

- **stonith**

  配置隔离设备以用于 Pacemaker。有关 pcs stonith 命令的详情请参考 第 5 章 *隔离：配置 STONITH*。

- **constraint**

  管理资源限制。有关 pcs constraint 命令的详情请参考 第 7 章 *资源约束*。

- **属性**

  设置 Pacemaker 属性。有关使用 pcs property 命令设置属性的详情请参考 第 12 章 *Pacemaker 集群属性*。

- **status**

  查看当前集群和资源状态.有关 pcs status 命令的详情请参考 第 3.5 节 "显示状态"。

- **config**

  以用户可读形式显示完整的集群配置。有关 pcs config 命令的详情请参考 第 3.6 节 "显示完全集群配置"。

3.2. PCS USAGE HELP 显示

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说明。例如，以下命令显示 pcs resource 命令的参数。输出中仅显示一部分。

```
# pcs resource -h
Usage: pcs resource [commands]...
Manage pacemaker resources
Commands:
   show [resource id] [--all]
```

```
        Show all currently configured resources or if a resource is specified
        show the options for the configured resource.  If --all is specified
        resource options will be displayed

    start <resource id>
        Start resource specified by resource_id
...
```

3.3. 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 第 3.4 节 "将配置更改保存到文件" 所述。

3.4. 将配置更改保存到文件

使用 **pcs** 命令时，您可以使用 **-f** 选项将配置更改保存到文件，而不影响活动的 CIB。

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
```

以下命令在 **testfile** 文件中创建一个资源，但不将该资源添加到当前正在运行的集群配置中。

```
# pcs -f testfile resource create VirtualIP ocf:heartbeat:IPaddr2 ip=192.168.0.120 cidr_netmask=24
op monitor interval=30s
```

您可以使用以下命令将 **testfile** 的当前内容推送到 CIB 中。

```
# pcs cluster cib-push testfile
```

3.5. 显示状态

您可以使用以下命令显示集群和集群资源的状态。

```
pcs status commands
```

如果没有指定 command *参数，这个命令会显示有关集群和资源的所有信息*。您可以通过指定 资源、组、集群、 节点 或 **pcsd** 只显示特定集群 组件的状态。

3.6. 显示完全集群配置

使用以下命令显示完整的集群的当前配置。

```
pcs config
```

3.7. 显示当前 PCS 版本

以下命令显示正在运行的 pcs 的当前版本。

```
pcs --version
```

3.8. 备份和恢复集群配置

自红帽企业 Linux 7.1 发行版本起，您可以使用以下命令在 tarball 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
```

使用以下命令从备份中恢复所有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。指定 --local 选项仅恢复当前节点上的文件。

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

第 4 章 集群创建和管理

本章论述了如何使用 Pacemaker 执行基本集群管理，包括创建集群、管理集群组件和显示集群状态。

4.1. 创建集群

要创建正在运行的集群，请执行以下步骤：

1. 在群集的每个节点上启动 pcsd。
2. 验证组成集群的节点。
3. 配置和同步群集节点。
4. 启动群集节点上的群集服务。

以下小节介绍了用于执行这些步骤的命令。

4.1.1. 启动 pcsd 守护进程

以下命令启动 pcsd 服务并在系统启动时启用 pcsd。这些命令应在群集中的每个节点上运行。

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

4.1.2. 对集群节点进行身份验证

以下命令 向 群集节点上的 pcs 守护进程验证 pcs。

- pcs 管理员的用户名必须是每个节点上的 hacluster。建议每个节点上的用户 hacluster 的密码都相同。
- 如果没有指定 用户名 或 密码，系统在执行该命令时会提示您为每个节点输入这些参数。
- 如果没有指定任何节点，这个命令将在使用 pcs cluster setup 命令指定的节点上验证 pcs （如果您之前已执行过该命令）。

```
pcs cluster auth [node] [...] [-u username] [-p password]
```

例如，以下命令可针对由 z1.example.com 和 z2. example.com 群集中的两个节点验证z1.example.com 上的用户 hacluster ：此命令会在集群节点上提示输入用户 hacluster 的密码。

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

授权令牌存储在文件 ~/.pcs/tokens （或 /var/lib/pcsd/tokens）中。

4.1.3. 配置和启动集群节点

以下命令配置集群配置文件，并将配置同步到指定的节点。

- 如果指定了 **--start** 选项，命令也会在指定节点上启动集群服务。如果需要，您还可以使用单独的 **pcs cluster start** 命令启动集群服务。

  当您使用 **pcs cluster setup --start** 命令创建群集时，或使用 **pcs cluster start** 命令启动群集服务时，在群集启动并运行前可能会稍有延迟。在对群集及其配置执行任何后续操作之前，建议您使用 **pcs cluster status** 命令确保群集已启动并运行。

- 如果指定了 **--local** 选项，命令将仅在本地节点上执行更改。

```
pcs cluster setup [--start] [--local] --name cluster_ name node1 [node2] [...]
```

以下命令在指定节点或节点上启动集群服务。

- 如果指定了 **--all** 选项，命令将在所有节点上启动群集服务。

- 如果没有指定任何节点，则仅在本地节点上启动集群服务。

```
pcs cluster start [--all] [node] [...]
```

4.2. 为集群配置超时值

使用 **pcs cluster setup** 命令创建群集时，群集的超时值被设置为默认值，适合大多数群集配置。但是，如果您的系统需要不同的超时值，您可以使用 **pcs cluster setup** 选项修改这些值，如下所示：表 4.1 “超时选项”

表 4.1. 超时选项

选项	描述
--token ***timeout***	以毫秒为单位设置时间，直到在未接收令牌后声明令牌丢失（默认为 1000 毫秒）
--join ***timeout***	以毫秒为单位设置等待加入消息的时间（默认值 50 ms）

选项	描述
--consensus *timeout*	以毫秒为单位设置在启动新成员发货配置前等待达成一致的时间（默认值为 1200 ms）
--miss_count_const *count*	在发生重新传输前检查消息以进行重新传输前，设置收到令牌的最大次数（默认 5 消息）
--fail_recv_const *failures*	指定在生成新配置前可能会发生消息时发生的令牌轮转数量，但不接收任何消息（默认值 2500 失败）

例如，以下命令创建群集 new_cluster，并将令牌超时值设置为 10000 毫秒（10 秒），并将加入超时值设置为 100 毫秒。

```
# pcs cluster setup --name new_cluster nodeA nodeB --token 10000 --join 100
```

4.3. 配置冗余环网协议(RRP)

注意

红帽支持在群集中配置冗余环协议(RRP)，具体取决于 RHEL 高可用性群集支持政策 - 集群互连网络接口的"冗余环协议(RRP)"部分中描述的条件。

使用 pcs cluster setup 命令创建群集时，您可以通过为每个节点指定两个接口，使用冗余环协议配置群集。使用默认的 udpu 传输时，当指定集群节点时，您可以指定环 0 地址，后跟 ','，然后是环 1 地址。

例如，以下命令将名为 my_rrp_clusterM 的集群配置为节点 A 和节点 B。节点 A 有两个接口：node A-0 和 nodeA-1。节点 B 有两个接口，node B-0 和 nodeB-1。若要利用 RRP 将这些节点配置为群集，请执行以下命令：

```
# pcs cluster setup --name my_rrp_cluster nodeA-0,nodeA-1 nodeB-0,nodeB-1
```

有关在使用sud p 传输的集群中配置 RRP 的详情，请查看 pcs cluster setup 命令的帮助屏幕。

4.4. 管理集群节点

以下小节介绍了用来管理集群节点的命令，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。

4.4.1. 停止集群服务

以下命令在指定节点或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群集服务，如果没有指定任何节点，则仅在本地节点上停止群集服务。

```
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
```

4.4.2. 启用和禁用集群服务

使用以下命令，将群集服务配置为在指定节点或节点上启动时运行。

- 如果指定了 **--all** 选项，该命令在所有节点上启用集群服务。
- 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。

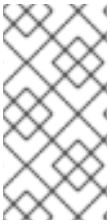
```
pcs cluster enable [--all] [node] [...]
```

使用以下命令配置在指定节点或节点的启动时不要运行的群集服务。

- 如果指定了 **--all** 选项，该命令将禁用所有节点上的群集服务。
- 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] [...]
```

4.4.3. 添加集群节点



注意

强烈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。

使用以下步骤将新节点添加到现有集群中。在本例中，现有群集节点为 clusternode-01.example.com、cluster node-02.example.com 和 clusternode-03.example.com。新节点为 newnode.example.com。

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 ticket 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 ~]# yum install -y pcs fence-agents-all
```

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. 设置用户 ID hacluster 的密码。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. 在新群集节点上验证用户 hacluster。

```
[root@clusternode-01 ~]# pcs cluster auth newnode.example.com
Username: hacluster
Password:
newnode.example.com: Authorized
```

–

2. 在现有集群中添加新节点。此命令还会将群集配置文件 corosync.conf 同步到集群中的所有节点，包括您要添加的新节点。

```
[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。有关配置隔离设备的详情请参考 第 5 章 *隔离：配置 STONITH*。

4.4.4. 删除集群节点

以下命令关闭指定节点，并将其从群集配置文件 corosync.conf 中删除至群集配置文件 corosync.conf 中。有关从集群节点完全删除集群的所有信息的详情，请参考 第 4.6 节 “删除集群配置”。

```
pcs cluster node remove node
```

4.4.5. 待机模式

以下命令将指定节点设置为待机模式。指定节点无法再托管资源。该节点上所有当前活跃的资源都将移至另一节点。如果您指定了 --all，这个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 cluster 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，这个命令会将所有节点从待机模式中删除。

```
pcs cluster unstandby node | --all
```

请注意，当执行 pcs cluster standby 命令时，这会阻止资源在指定节点上运行。执行 pcs cluster 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。有关资源限制的详情请参考 第 7 章 *资源约束*。

4.5. 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集群。您可以为独立的用户授予两组权限：

- 允许单个用户通过 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，如 第 4.5.1 节 “设置通过网络访问节点的权限” 所述。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，如 第 4.5.2 节 “使用 ACL 设置本地权限” 所述。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。

4.5.1. 设置通过网络访问节点的权限

要授予特定用户通过 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，请将这些用户添加到组 haclient。然后，您可以使用 Web UI 为这些用户授予权限，如 第 2.2.2 节 “设置集群管理权限” 所述。

4.5.2. 使用 ACL 设置本地权限

从红帽企业 Linux 7.1 开始，您可以使用 pcs acl 命令为本地用户设置权限，以允许通过使用访问控制列表(ACL)对集群配置进行只读或读写访问。您还可以使用 pcsd Web UI 配置 ACL，如 第 2.3.4 节 “配置 ACL” 所述。默认情况下，root 用户和属于 haclient 组成员的任何用户都拥有对集群配置的完整本地读/写访问权限。

为本地用户设置权限分为两个步骤：

1. 执行 **pcs acl 角色 create...** *命令创建定义该角色权限的角色。*

2. **使用 pcs acl user create 命令将您创建的角色分配给用户。**

以下示例步骤提供集群配置到名为 rouser 的本地用户的只读访问权限。

1. **此流程要求本地系统上存在 rouser 用户，并且 rouser 是组 haclient 的成员。**

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

2. **使用 enable-acl 集群属性启用 Pacemaker ACL。**

```
# pcs property set enable-acl=true --force
```

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 只读 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)
```

以下示例步骤提供集群配置到名为 wuser 的本地用户的写入访问权限。

1. **此流程要求本地系统上存在 wuser 用户，并且用户 wuser 是组 haclient 的成员。**

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

2. **使用 enable-acl 集群属性启用 Pacemaker ACL。**

```
# pcs property set enable-acl=true --force
```

3. **创建名为 write-access 的角色，其具有 cib 的写入权限。**

```
# pcs acl role create write-access description="Full access" write xpath /cib
```

4. **在 pcs ACL 系统中创建用户 wuser，并为该用户分配 write-access 角色。**

```
# pcs acl user create wuser write-access
```

5. **查看当前的 ACL。**

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

有关集群 ACL 的详情请参考 pcs acl 命令的帮助屏幕。

4.6. 删除集群配置

要删除所有集群配置文件并停止所有群集服务，从而永久销毁集群，请使用以下命令：

警告

此命令会永久删除已创建的任何集群配置。建议您在销毁群集之前运行 pcs cluster stop。

```
pcs cluster destroy
```

4.7. 显示集群状态

以下命令显示集群的当前状态和集群资源。

```
pcs status
```

您可以使用以下命令显示集群当前状态的信息子集。

以下命令显示集群的状态，但不显示集群资源。

```
pcs cluster status
```

以下命令显示集群资源的状态。

```
pcs status resources
```

4.8. 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 第 4.4.5 节 “待机模式”。

- 如果您需要在不停止该资源的情况下将单独的资源从当前运行的节点中移动，您可以使用 pcs resource move 命令将资源移到其他节点。有关 pcs resource move 命令的详情请参考 第 8.1 节 “手动在集群中移动资源”。

  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中运行。当您准备好重新移动资源时，可以执行 pcs resource clear 或 pcs constraint delete 命令以移除约束。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指定节点，如 第 8.1.1 节 “从当前节点移动资源” 所述。

- 如果您需要停止正在运行的资源并阻止集群再次启动，您可以使用 pcs resource disable 命令。有关 pcs resource disable 命令的详情请参考 第 8.4 节 “启用、禁用和禁止集群资源”。

- 如果要防止 Pacemaker 对资源执行任何操作（例如，要在资源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 第 8.6 节 “受管资源” 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没有启动或停止服务的状态，您可以设置 维护模式集群 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。有关设置集群属性的详情请参考 表 12.1 “集群属性”。

- 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 第 9.4.8 节 “系统升级和 pacemaker_remote” 所述。

第 5 章 隔离：配置 STONITH

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。

仅仅因为节点不响应，这并不表示它不会访问您的数据。完全确保您的数据安全的唯一方法是使用 STONITH 隔离节点，以便我们能够在允许从另一个节点访问数据前确保节点真正离线。

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程度，请参阅红帽高可用性集群中的隔离。

5.1. 可用的 STONITH（隔离）代理

使用以下命令查看所有可用的 STONITH 代理列表。您可以指定一个过滤器，这个命令只显示与过滤器匹配的 STONITH 代理。

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

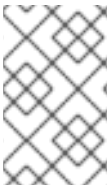
5.2. 隔离设备的常规属性

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 *disablestonith_id*命令来禁用隔离设备。这会阻止任何节点使用该设备
- 要防止特定节点使用隔离设备，您可以使用 pcs constraint location 为隔离资源配置位置限制... 避免命令。
- configurationing 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。

表 5.1 “隔离设备的常规属性” 描述您可以为隔离设备设置的一般属性。有关您可以为特定隔离设备设置

的隔离属性的信息，请参阅 第 5.3 节 “显示设备特定隔离选项”。



注意

有关更高级隔离配置属性的详情，请参考 第 5.8 节 “其他隔离配置选项”

表 5.1. 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：node **1:1;node2:2，3 告知** 集群将端口 1 用于 node1，端口 2 和端口 3 用于 node2。
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	dynamic-list	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）

5.3. 显示设备特定隔离选项

使用以下命令查看指定 STONITH 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，不支持 循环 值且不应指定，因为它可能导致数据崩溃。

5.4. 创建隔离设备

以下命令创建一个 stonith 设备。

```
pcs stonith create stonith_id stonith_device_type [stonith_device_options]
```

```
# 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 参数映射主机名。

有关 pcmk_host_list 和 pcmk_host_map 参数的详情请参考 表 5.1 “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 第 5.12 节 “测试隔离设备”。

5.5. 显示隔离设备

以下命令显示所有当前配置的隔离设备。如果指定了 *astonith_id*，命令仅显示为该 stonith 设备配置的选项。如果指定了 --full 选项，则会显示所有配置的 stonith 选项。

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

5.6. 修改和删除隔离设备

使用以下命令修改或者添加当前配置的隔离设备选项。

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

使用以下命令从当前的配置中删除隔离设备。

```
pcs stonith delete stonith_id
```

5.7. 使用隔离设备管理节点

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 来关闭节点，而不是重启节点。

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

如果 stonith 设备无法隔离节点，即使它不再活跃，集群可能无法恢复该节点中的资源。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
```

5.8. 其他隔离配置选项

表 5.2 “隔离设备的高级属性” 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 5.2. 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。使用这个选项指定一个替代的、特定于具体设备的参数，该参数应指示要隔离的计算机。值 **none** 可用于告诉集群不提供任何额外参数。
pcmk_reboot_action	字符串	reboot	运行 的另一个命令，而不是 **重新启动**。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	指定替代了重启操作的超时时间，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。

项	类型	默认值	描述
pcmk_off_action	字符串	off	运行另一个命令，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	指定一个替代 off 操作使用的超时时间而不是 **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	运行另一个命令，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	指定替代了 list 操作使用的超时时间而不是 **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、list 操作使用的、特定于设备的超时时间。
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	运行另一个命令，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	指定替代了 monitor 操作使用的超时时间，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。

项	类型	默认值	描述
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	运行另一个命令，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	指定替代 status 操作使用的超时时间，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。
pcmk_delay_base	time	0s	为 stonith 操作启用基础延迟并指定一个基本延迟值。在带有偶数节点的集群中，配置延迟有助于避免在平等分割时同时出现节点相互隔离的问题。当同一个隔离设备被所有节点使用时，使用一个随机延迟会很有用，而如果每个节点都使用单独的设备时，使用不同的静态延迟会很有用。总的延迟是根据一个随机延迟值再加上这个静态延迟而获得的，这样可以保持总和低于最大延迟。如果您设置了 **pcmk_delay_base**，但没有设置 **pcmk_delay_max**，则没有随机组件，它将是 **pcmk_delay_base** 的值。 有些隔离代理使用一个 "delay" 参数，它独立于使用 **pcmk_delay_*** 属性配置的延迟。如果同时配置了这两个延迟，它们会被相加。因此，一般不要同时使用它们。

项	类型	默认值	描述
pcmk_delay_max	time	0s	为 stonith 动作启用随机延迟并指定最大随机延迟。在带有偶数节点的集群中，配置延迟有助于避免在平等分割时同时出现节点相互隔离的问题。当同一个隔离设备被所有节点使用时，使用一个随机延迟会很有用，而如果每个节点都使用单独的设备时，使用不同的静态延迟会很有用。总的延迟是根据一个随机延迟值再加上这个静态延迟而获得的，这样可以保持总和低于最大延迟。如果您设置了 **pcmk_delay_max**，但没有设置 **pcmk_delay_base**，则延迟没有静态组件。 有些隔离代理使用一个 "delay" 参数，它独立于使用 **pcmk_delay_*** 属性配置的延迟。如果同时配置了这两个延迟，它们会被相加。因此，一般不要同时使用它们。
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 **并发-fencing=true**。值为 -1 代表没有限制。
pcmk_on_action	字符串	on	仅供高级使用：要运行的一个替代命令，而不是 **on**。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 **on** 操作的替代的、特定于具体设备的命令。
pcmk_on_timeout	time	60s	仅供高级使用：指定用于操作的替代超时时间，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。使用这个选项指定替代的、**操作使用的、特定于设备的超时时间**。
pcmk_on_retries	整数	2	仅供高级使用：**在** 超时时间内重试命令的次数上限。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。

您可以设置 fence-reaction 集群 属性，如 表 12.1 “集群属性” 中所示，决定集群节点在其自身隔离通知时应如何做出反应。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。虽然此属性的默认值为 stop，它会尝试立即停止 Pacemaker 并保持停止，但这个值的最安全选择是 panic，它会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。

5.9. 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。

- 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。这些设备以使用用逗号分开的 stonith id 列表形式提供，它们是该级别要尝试的节点。

```
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 配置两个隔离设备：一个名为 my_ilo 的 ilo 隔离设备，以及名为 my_apc 的 apc 隔离设备。这些命令设置隔离级别，以便在设备 my_ilo 失败且无法隔离节点时，Pacemaker 将尝试使用设备 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
```

从 Red Hat Enterprise Linux 7.4 开始，您可以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隔离拓扑中的节点。例如，以下命令将节点 node1、node2 和 ' node 3 配置为使用隔离设备 apc1 和 'apc2，以及节点 'node4、node5 和 'node6，以使用隔离设备 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
```

5.10. 为冗余电源配置隔离

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。

在 Red Hat Enterprise Linux 7.2 之前，您需要明确配置使用 'on' 或 'off' 操作的设备的不同版本。由于 Red Hat Enterprise Linux 7.2，现在只需要定义每个设备一次，并指定它们都需要隔离该节点，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
```

5.11. 配置 ACPI 以用于集成隔离设备

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 可让集成的隔离设备立即完全关闭节点，而不是尝试彻底关闭（例如，现在的 shutdown -h）。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"或无延迟关闭该节点的对等设置，如 第 5.11.1 节 “使用 BIOS 禁用 ACPI Soft-Off” 所述。

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，并验证隔离时节点是否立即关闭，如 第 5.11.2 节 “在 logind.conf 文件中禁用 ACPI Soft-Off” 所述。这是禁用 ACPI Soft-Off 的第一个备用方法。

-

在内核引导命令行中附加 acpi=off，如 第 5.11.3 节 “在 GRUB 2 文件中完全禁用 ACPI” 所述。这是禁用 ACPI Soft-Off 的第二个备用方法，如果首选方法或第一个备用方法不可用时使用。

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。

5.11.1. 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。

1. 重新引导节点并启动 BIOS CMOS 设置实用程序程序。

2. 导航到 Power 菜单（或对等的电源管理菜单）。

3. 在 Power 菜单中，将 PWR-BTTN 功能（或等效）的 Soft-Off 设置为 Instant-Off （或者使用电源按钮无延迟关闭节点的对等设置）。例 5.1 “BIOS CMOS 设置实用程序 ：Soft-Off by PWR-BTTN 设置为 Instant-Off” 显示 Power 菜单，并将 ACPI Function 设置为 Enabled ，Soft-Off by PWR-BTTN 设置为 Instant-Off。

   注意

   与 ACPI Function、Soft-Off by PWR-BTTN 和 Instant-Off 等效的功能可能因计算机而异。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 设置实用程序程序，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 第 5.12 节 “测试隔离设

备”。

例 5.1. BIOS CMOS 设置实用程序 ：Soft-Off by PWR-BTTN 设置为 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      |                   |
|                                           |                   |
|                                           |                   |
+---------------------------------------------|-------------------+
```

本例演示了 ACPI Function 设置为 Enabled ， Soft-Off by PWR-BTTN 设置为 Instant-Off。

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

要禁用 /etc/systemd/logind.conf 文件中的 power-key 握手，请使用以下步骤。

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

```
HandlePowerKey=ignore
```

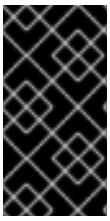
2. 重新载入 systemd 配置：

```
# systemctl daemon-reload
```

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 第 5.12 节 “测试隔离设备”。

5.11.3. 在 GRUB 2 文件中完全禁用 ACPI

您可以通过在内核的 GRUB 菜单条目中附加 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。

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，以更改每个群集节点的 grub.cfg 文件，如下所示：

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

   有关 GRUB 2 的常规信息，请参阅《系统管理员指南》的使用 GRUB 2 章节。

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 第 5.12 节 “测试隔离设备”。

5.12. 测试隔离设备

隔离（Fencing）是红帽集群基础结构的基础部分，因此验证或者测试隔离服务是否正常至关重要。

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 IPMI 的设备配置隔离，请尝试使用 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，请确认该设备是可以被 ping 到的，没有如防火墙配置阻止对隔

离设备的访问，隔离代理中启用了远程访问，且凭证正确。

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

本节中的示例将 fence_ilo 隔离代理脚本用于 iLO 设备。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_ilo 隔离代理脚本的格式，以检查另一个节点上的隔离设备接口的状态，而不实际对其进行隔离。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_ilo -a ipaddress -l username -p password -o status
```

以下示例显示了使用 -o reboot 参数运行 fence_ilo 隔离代理脚本的格式。在一个节点上运行这个命令会重启另一个配置了隔离代理的节点。

```
# fence_ilo -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_ilo -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，并且仍会尝试 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. 在群集中使用手动运行并启动群集相同的选项配置隔离设备后，可以从任何节点（或者多次来自不同节点）使用 pcs stonith fence 命令测试隔离，如下例所示。pcs stonith fence 命令从 CIB 中读取群集配置，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群集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该设备。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，使用该协议尝试连接到该设备。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 和 error 实例，它们可以让大家了解正在转换的内容，但有些代理可以提供更多信息。

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。

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。

以下示例会阻塞 corosync，假设使用默认的 corosync 端口，firewall d 用作本地防火墙，corosync 使用的网络接口位于默认防火墙区中：

```
# 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 模拟崩溃并导致您的计算机崩溃。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
```

第 6 章 配置集群资源

本章提供有关在集群中配置资源的信息。

6.1. 资源创建

使用以下命令来创建集群资源。

```
pcs resource create resource_id [standard:[provider:]]type [resource_options] [op operation_action
operation_options [operation_action operation options]...] [meta meta_options...] [clone
[clone_options] | master [master_options] | --group group_name [--before resource_id | --after
resource_id] | [bundle bundle_id] [--disabled] [--wait[=n]]
```

指定 **--group** 选项时，资源将添加到资源组中。如果组不存在，这会创建组并将这些资源添加到组中。有关资源组的详情请参考 第 6.5 节 “资源组”。

--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** 选项表示资源不会被自动启动。

以下命令创建了名称为 **VirtualIP** 标准 **ocf**、**provider heartbeat** 和类型 **IPaddr2** 的资源。此资源的浮动地址是 **192.168.0.120**，系统将每 **30** 秒检查一次该资源是否在运行。

```
# 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** 和 **heartbeat** 供应商的标准 。

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

使用以下命令删除配置的资源。

```
pcs resource delete resource_id
```

例如，以下命令删除资源 ID 为 VirtualIP 的现有资源

```
# pcs resource delete VirtualIP
```

- *有关 pcs resource create 命令的 resource _id、standard、provider 和 type 字段的详情请参考 第 6.2 节 “资源属性”。*
- *有关为单个资源定义资源参数的详情请参考 第 6.3 节 “资源特定参数”。*
- *有关定义资源 meta 选项的详情，集群用来决定资源的行为方式，请参阅 第 6.4 节 “资源元数据选项”。*
- *有关定义要在资源上执行的操作的详情请参考 第 6.6 节 “资源操作”。*
- *指定克隆选项可创建克隆资源。指定 master 选项会创建一个 master/slave 资源。有关使用多个模式的资源克隆和资源的详情请参考 第 9 章 高级配置。*

6.2. 资源属性

您为资源定义的属性告诉集群要用于该资源的脚本，在哪里找到该脚本及其符合的标准。表 6.1 “资源属性” 描述这些属性。

表 6.1. 资源属性

项	描述
resource_id	您的资源名称
standard	脚本符合的标准。允许的值： **ocf**,**service** , **upstart**, **systemd**,**lsb**,**stonith**
type	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** 作为提供商。

表 6.2 “显示资源属性的命令”. 总结了显示可用资源属性的命令。

表 6.2. 显示资源属性的命令

pcs Display 命令	Output
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	显示可用资源代理提供程序列表。
pcs resource list *字符串*	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。

6.3. 资源特定参数

对于任何单独的资源，您可以使用以下命令显示您可以为该资源设置的参数。

```
# pcs resource describe standard:provider:type|type
```

例如：以下命令显示您可以为 LVM 类型的资源设置的参数。

```
# pcs resource describe LVM
Resource options for: LVM
  volgrpname (required): The name of volume group.
  exclusive: If set, the volume group will be activated exclusively.
  partial_activation: If set, the volume group will be activated even
  only partial of the physical volumes available. It helps to set to
  true, when you are using mirroring logical volumes.
```

6.4. 资源元数据选项

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。表 6.3 “资源元数据选项” 描述这些选项。

表 6.3. 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。

项	默认值	描述
target-role	**Started**	集群应该将这个资源维持为什么状态？允许的值： * *Stopped* - 强制停止资源 * *Started* - 允许启动资源（当为 multistate 资源时，不会将其提升为 master） * *Master* - 允许启动资源，并在可能的情况下提升资源
is-managed	**true**	集群是否允许启动和停止资源？允许的值：**true**,**false**
resource-stickiness	0	指示资源倾向于保留在当前位置的程度。
Requires	Calculated	指示可在什么情况下启动资源。 除非 **满足以下条件，否则默认为隔离**。可能的值： * **无** - 集群总是可以启动该资源。 * **仲裁** - 集群只能在大多数配置的节点活跃时启动此资源。如果 **stonith-enabled** 为 **false** 或资源 **的标准** is **stonith**，则这是默认值。 * **隔离** - 只有大多数配置的节点活跃 *且* 任何失败或未知节点都已关闭时，集群才能启动此资源。 * **取消** 隔离 - 只有大多数配置的节点活跃且任何失败或未知节点都已关闭，*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 provided* ***=unfencing stonith*** *meta 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，群集将 **INFINITY** （默认值）视为非常大但有上限的数字。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。有关配置 **migration-threshold** 选项的详情请参考 第 8.2 节 “因为失败而移动资源”。有关 **start-failure-is-fatal** 选项的详情请参考 表 12.1 “集群属性”。
failure-timeout	**0** （禁用）	与 **migration-threshold** 选项结合使用，可指示在作为故障发生前要等待的秒数，并可能允许资源返回到失败的节点。与任何基于时间的操作一样，无法保证检查的频率高于 **cluster-recheck-interval** 集群参数的值。有关配置 **failure-timeout** 选项的详情请参考 第 8.2 节 “因为失败而移动资源”。

项	默认值	描述
multiple-active	**stop_start**	如果这个资源在多个节点上找到活跃的资源，集群该怎么办。允许的值： * **block** - 将资源标记为非受管 * **stop_only** - 停止所有活跃的实例，并以这种方式保留它们 * **stop_start** - 停止所有活跃的实例并在一个位置中只启动该资源

要更改资源选项的默认值，请使用以下命令：

```
pcs resource defaults options
```

*例如，以下命令会将 **resource-stickiness** 的默认值重置为 **100**：*

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

*省略 **pcs resource defaults** 中的 **options** 参数会显示资源选项当前配置的默认值的列表。以下示例显示了在将 **resource-stickiness** 重置为 **100** 后此命令的输出。*

```
# pcs resource defaults
resource-stickiness:100
```

*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 **meta** 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|type [resource options] [meta
meta_options...]
```

*例如，以下命令创建一个资源 粘性值为 **50** 的资源。*

```
# pcs resource create VirtualIP ocf:heartbeat:IPaddr2 ip=192.168.0.120 cidr_netmask=24 meta
resource-stickiness=50
```

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

```
pcs resource meta resource_id | group_id | clone_id | master_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 meta** 选项设置为 **20** 秒，以便资源可在 **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，您可以显示资源的值以覆盖设置了 **failure-timeout=20s** 的值。

```
# pcs resource show dummy_resource
 Resource: dummy_resource (class=ocf provider=heartbeat type=Dummy)
  Meta Attrs: failure-timeout=20s
  Operations: start interval=0s timeout=20 (dummy_resource-start-timeout-20)
              stop interval=0s timeout=20 (dummy_resource-stop-timeout-20)
              monitor interval=10 timeout=20 (dummy_resource-monitor-interval-10)
```

有关资源克隆 **meta** 选项的详情请参考 第 9.1 节 “资源克隆”。有关资源 **master meta** 选项的详情请参考 第 9.2 节 “多状态资源：具有多个模式的资源”。

6.5. 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** 支持组的概念。

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** 选项指定与组中已存在的资源相关的添加资源的位置。

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。

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

您可以使用以下命令从组中删除资源。如果组中没有资源，这个命令会删除组本身。

```
pcs resource group remove group_name resource_id...
```

以下命令列出所有目前配置的资源组。

```
pcs resource group list
```

以下示例创建名为 快捷 方式的资源组，其中包含现有资源 IPaddr 和 Email。

```
# pcs resource group add shortcut IPaddr Email
```

对组可以包含的资源数量没有限制。组群的基本属性如下。

- 资源按照您指定的顺序启动（在本示例中，首先 IPaddr，然后是 电子邮件）。
- 资源按照您指定的顺序的相反顺序停止。（首先发送电子邮件，再发送 IPaddr）。

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。

- 如果 IPaddr 无法在任何位置运行，则无法 电子邮件.
- 但是，如果 电子邮件 无法在任何位置运行，这不会影响 IPaddr。

显然，随着该组的规模不断增长，创建资源组时减少的配置工作量可能会变得非常显著。

6.5.1. 组选项

资源组从其包含的资源继承以下选项：优先级、target-role 和is-managed。有关资源选项的详情请参考 表 6.3 “资源元数据选项”。

6.5.2. 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 资源粘性 为 100，并且组有 7 个成员（其中五个处于活动状态），则整个组将首选其当前位置，分数为 500。

6.6. 资源操作

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

表 6.4 "操作的属性" 总结资源监控操作的属性。

表 6.4. 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。
name	要执行的操作。常见值：**监控**、**启动**、**停止**
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。非零值只有在操作 **名称** 设为 **monitor** 时才有意义。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 操作不是在 01:00:20 时发生，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 **间隔** 设为零，则操作 **的名称** 被设置为 **start**，**超时** 值设为 40，则 Pacemaker 在启动此资源时将使用 40 秒超时。通过零间隔的 **monitor** 操作，您可以为 Pacemaker 在启动时使用的探测设置 **超时**/**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 **超时** 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，并调查您预计较长的执行时间可以增加这个值。 **超时** 值不是任何类型的延迟，如果操作在超时期限完成后返回，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** - Pretend resource 没有失败 * **block** - 不要对资源执行任何进一步的操作 * **stop** - 停止资源且不在其它位置启动它 * **restart** - 停止资源并重新启动（可能在不同的节点上） * **fence** - 资源失败的节点 STONITH * **standby** - 从资源失败的节点中移出 *所有资源* * **migrate** - 如果可能，将资源迁移到另一个节点。这等同于将 **migration-threshold** 资源 meta 选项设置为 1。 当启用 STONITH 并阻止其他 **时**，**停止** 操作的默认设置 **会被隔离**。所有其他操作默认为 **重新启动**。
enabled	如果为 **false**，则操作将被视为不存在。允许的值： **true**,**false**

6.6.1. 配置资源操作

您可以使用以下命令在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，以下命令使用监控操作创建 IPaddr2 资源：新资源称为 VirtualIP，IP 地址为 192.168.0.99，eth 2 上的子网掩码为 24。每 30 秒将执行一次监控操作。

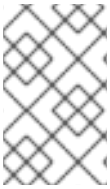
```
# pcs resource create VirtualIP ocf:heartbeat:IPaddr2 ip=192.168.0.99 cidr_netmask=24 nic=eth2 op
monitor interval=30s
```

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** ：

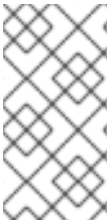
```
# pcs resource create VirtualIP ocf:heartbeat:IPaddr2 ip=192.168.0.99 cidr_netmask=24 nic=eth2
```

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 show 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)
```



注意

当您使用 **pcs resource update** 命令更新资源操作时，您没有特别调用的任何选项都将重置为默认值。

6.6.2. 配置全局资源操作默认值

您可以使用以下命令为监控操作设置全局默认值。

```
pcs resource op defaults [options]
```

例如，以下命令为所有监控操作设置 超时 值 **240** 秒的全局默认值。

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

- 要显示当前配置的监控操作的默认值，请在执行 **pcs resource op defaults** 命令时不要指定任何选项。

例如，以下命令显示集群的默认监控操作值，其 超时 值配置为 **240** 秒。

```
# pcs resource op defaults
timeout: 240s
```

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。要满足全局操作超时值，您必须明确在没有 超时 选项的情况下创建集群资源，或者您必须通过更新集群资源来删除 超时 选项，如下命令所示。

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

例如，在为所有监控操作设置一个 超时 值 **240** 秒，并更新集群资源 **VirtualIP** 以删除 **monitor** 操作的超时值后，资源 **VirtualIP** 将分别具有 **start**、**stop** 和 **monitor** 操作的 超时值，分别为 **20s**、**40s** 和 **240s**。超时操作的全局默认值仅在 **monitor** 操作中应用，上一命令已删除了默认的 超时 选项。

```
# pcs resource show 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)
```

6.7. 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 show
```

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** 命令将生成以下输出：

```
# pcs resource show
 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 show resource_id
```

例如，以下命令显示资源 **VirtualIP** 当前配置的参数。

```
# pcs resource show VirtualIP
 Resource: VirtualIP (type=IPaddr2 class=ocf provider=heartbeat)
  Attributes: ip=192.168.0.120 cidr_netmask=24
  Operations: monitor interval=30s
```

6.8. 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]
```

以下命令显示为资源 **VirtualIP** 配置的参数的初始值、更改 **ip** 参数值的命令，以及 **update** 命令中的值。

```
# pcs resource show 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 show VirtualIP
 Resource: VirtualIP (type=IPaddr2 class=ocf provider=heartbeat)
  Attributes: ip=192.169.0.120 cidr_netmask=24
  Operations: monitor interval=30s
```

6.9. 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，您需要添加一个 OCF_CHECK_LEVEL=n 选项。

例如，如果您配置以下 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
```

6.10. 启用和禁用集群资源

以下命令启用由 resource_id 指定的资源。

```
pcs resource enable resource_id
```

以下命令禁用 resource_id 指定的资源。

```
pcs resource disable resource_id
```

6.11. 集群资源清理

如果资源失败，则显示集群状态时会出现一个失败信息。如果解析该资源，您可以使用 pcs resource cleanup 命令清除该故障状态。此命令会重置资源状态和 故障计数，指示集群忘记资源的操作历史记录并重新检测其当前状态。

以下命令清理由 resource_id 指定的资源。

```
pcs resource cleanup resource_id
```

如果没有指定 resource_id，这个命令会重置所有资源的资源状态和 故障计数。

从 Red Hat Enterprise Linux 7.5 开始，pcs resource cleanup 命令只会探测显示为失败操作的资源。要探测所有节点上的所有资源，使用以下命令：

```
pcs resource refresh
```

默认情况下，pcs resource refresh 命令只会探测到已知资源状态的节点。要探测所有资源，包括状态未知的资源，使用以下命令：

```
pcs resource refresh --full
```

第 7 章 资源约束

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：

- 位置 限制 - 位置约束决定资源可在哪个节点上运行。位置限制在 第 7.1 节 “位置限制” 中描述。
- 顺序 约束 - 顺序约束决定资源运行的顺序。在 第 7.2 节 “顺序限制” 中描述了顺序限制。
- 共存 约束 - 共同位置约束(colocation constraint)决定资源相对于其他资源的位置。在 第 7.3 节 “资源共存” 中描述了 colocation 约束。

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。有关资源组的详情请参考 第 6.5 节 “资源组”。

7.1. 位置限制

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。

除了位置限制外，资源运行的节点还受到 该资源的资源粘性 值的影响，这决定了资源是否首选保留在当前运行的节点中。有关设置 资源粘性 值的详情请参考 第 7.1.5 节 “配置资源以首选其当前节点”。

7.1.1. 基本位置限制

您可以配置基本位置约束，以指定资源首选项或避免节点，使用可选 分数 值来指示约束的首选程度。

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,以避免指定节点。

```
pcs constraint location rsc avoids node[=score] [node[=score]] ...
```

表 7.1 "简单位置限制选项" 以最简单的形式总结了配置位置限制的选项的含义。

表 7.1. 简单位置限制选项

项	描述
rsc	资源名称
node	节点的名称
分数	动态整数值，用于指示资源应首选的资源还是避免节点。INFINITY 是资源位置约束的默认 分 数值。 pcs contraint 位置 器 命令 中的 分数 值为 INFINITY 表示该节点首选该节点（如果节点可用），但不会阻止资源在指定节点不可用时在另一节点上运行。 pcs contraint 位置 器c 中的 分数 值为 INFINITY 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。这等同于设置分数为 -INFINITY 的 pcs constraint location add 命令。

以下命令创建了位置约束，以指定资源 Web 服务器首选 节点 node1。

```
# pcs constraint location Webserver prefers node1
```

从 Red Hat Enterprise Linux 7.4 开始，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。

以下命令创建一个位置约束，将资源 dummy0 指定为 dummy 9 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
```

7.1.2. 高级位置限制

在节点上配置位置限制时，您可以使用 pcs constraint location 命令的 resource-discovery 选项指示 Pacemaker 是否应该为指定资源在该节点上执行资源发现。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 将节点数扩展到数百个节点范围时，应考虑此选项。

以下命令显示为 pcs constraint location 命令指定 resource-discovery 选项的格式。请注意，id 是约束 id。在 表 7.1 “简单位置限制选项” 中总结了rsc、 node 和分数的含义。在这个命令中，正分数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]
```

表 7.2 “资源发现值” 总结了您可以为 resource-discovery 选项指定的值的含义。

表 7.2. 资源发现值

值	描述
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	永不为这个节点上的指定资源执行资源发现。
专用	仅在此节点上对指定资源执行资源发现（及其他标记为 **专用**的节点）。在不同节点间使用 **专用** 发现同一资源的多个位置限制可创建节点 **资源发现的** 子集。如果某个资源在一个或多个节点上标记为 **独** 占发现，则该资源仅被允许放置到节点的子集中。

值	描述

请注意，将 resource-discovery 选项设置为 never 或 专用 选项可在这些位置中激活资源，而无需了解集群的知识。如果服务在集群控制之外启动（如 systemd 或管理员），则可能会导致资源在多个位置处于活跃状态。如果 部分 群集发生故障或遭遇脑裂，或者资源在该节点上活跃时更改了 resource-discovery 属性，则也会发生这种情况。因此，只有在有超过八个节点时才使用这个选项，并可以保证只能在特定位置运行该资源（例如，当所需的软件没有在其它任何位置安装时）。

7.1.3. 使用规则确定资源位置

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。有关 Pacemaker 规则以及您可以设置的属性的一般信息，请参阅 第 11 章 Pacemaker 规则。

使用以下命令配置使用规则的 Pacemaker 约束。如果省略 分数，则默认为 INFINITY。如果省略 resource-discovery，则默认为 always。有关 resource-discovery 选项的详情请参考 第 7.1.2 节 “高级位置限制”。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。

使用规则配置位置限制时，分 数值可以是正数或负数，正值表示"prefers"，负值表示"avoids"。

> **pcs constraint location** *rsc* **rule [resource-discovery=***option***] [role=master|slave] [score=***score* **|**
> **score-attribute=***attribute***]** *expression*

expression 选项可以是以下之一，其中 duration_options 和 date_spec_options 是：hours、monthdays、workerdays、yeardays、monthdays、month、weeks、weekyears、moon，如 表 11.5 “日期规格的属性” 所述。

- defined|not_defined attribute
- attribute lt|gt|lte|gte|eq|ne [string|integer|version] 值
- 日期 gt|lt date

- 至今为止 的日期
- 持续时间为 duration_options 的日期...
- date-spec date_spec_options
- expression and|or 表达式
- （表达式）

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
```

7.1.4. 位置限制策略

使用 第 7.1.1 节 “基本位置限制”、第 7.1.2 节 “高级位置限制” 和 第 7.1.3 节 “使用规则确定资源位置” 中描述的任何位置限制，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。配置 opt-in 集群的步骤请参考 第 7.1.4.1 节 “配置 "Opt-In" 集群”。
- Opt-Out 集群s - 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。配置 opt-out 集群的步骤请参考 第 7.1.4.2 节 “配置

"Opt-Out" 集群"。这是默认的 Pacemaker 策略。

是否应选择将集群配置为 opt-in 或 opt-out 集群，取决于您的个人偏好和集群的构建。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。

7.1.4.1. 配置 "Opt-In" 集群

要创建一个 opt-in 集群，将 symmetric-cluster 集群属性设置为 false，以防止资源默认在任何位置运行。

```
# pcs property set symmetric-cluster=false
```

为单个资源启用节点。以下命令配置位置限制，以便资源 Web 服务器首选 节点 example-1、资源 数据库 首选节点 example-2，如果首选节点出现故障，这两个资源都可切换到节点 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
```

7.1.4.2. 配置 "Opt-Out" 集群

要创建一个 opt-out 集群，将 symmetric-cluster 集群属性设置为 true，以允许资源默认随处运行。

```
# pcs property set symmetric-cluster=true
```

以下命令将生成一个与 第 7.1.4.1 节 "配置 "Opt-In" 集群" 中的示例对应的配置。如果首选节点失败，这两个资源都可切换到节点 example-3，因为每个节点都有隐式分数 0。

```
# pcs constraint location Webserver prefers example-1=200
# pcs constraint location Webserver avoids example-2=INFINITY
# pcs constraint location Database avoids example-1=INFINITY
# pcs constraint location Database prefers example-2=200
```

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。

7.1.5. 配置资源以首选其当前节点

资源具有 资源粘性 值，您可以在创建资源时将其设置为 meta 属性，如 第 6.4 节 “资源元数据选项” 所述。resource-stickiness 值决定资源要保留在当前运行节点中的有多少。Pacemaker 与其他设置（如位置限制的分数）一起考虑资源 粘性 值，以确定是否将资源移至另一节点还是保留原位。

默认情况下，创建资源 粘性 值为 0。当 资源粘性 设置为 0 时，Pacemaker 的默认行为是移动资源，以便在集群节点中平均分配这些资源。这可能导致健康的资源变化频率超过您的要求。要防止这种行为，您可以将默认 资源粘性 值设置为 1。此默认值将应用到集群中的所有资源。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。

以下命令将默认资源粘性值设置为 1。

```
# pcs resource defaults resource-stickiness=1
```

如果设置了 资源粘性 值，则没有资源移至新添加的节点。如果此时需要资源平衡，您可以临时将资源粘性 值设置为 0。

请注意，如果位置约束分数高于资源粘性值，集群仍然可以将健康资源移至位置约束点的节点。

有关 Pacemaker 如何确定资源放置位置的更多信息，请参阅 第 9.6 节 “使用和放置策略”。

7.2. 顺序限制

顺序限制决定资源运行的顺序。

使用以下命令配置顺序约束：

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

表 7.3 “顺序约束的属性”. 总结了配置顺序约束的属性和选项。

表 7.3. 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	对资源执行的操作。action 属性的可能值如下： * start - 启动资源。 * stop - 停止资源。 * Prop rate - 将资源从 slave 资源提升到主资源。 * demote - 将资源从主资源降级到从资源。 如果没有指定操作，则启动默认操作。有关 master 和从资源的详情请参考 第 9.2 节 “多状态资源：具有多个模式的资源”。

项	描述
kind 选项	如何强制实施约束。kind 选项的可能值如下： * 可选 - 仅在两个资源都执行指定操作时才应用。有关可选排序的详情请参考 第 7.2.2 节 “公告排序”。 * 强制 - Always （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 第 7.2.1 节 “强制排序”。 * serialize - 确保一组资源不会同时发生两个 stop/start 操作。
对称 选项	如果为 true（默认值），按相反顺序停止资源。默认值为： true

7.2.1. 强制排序

强制限制表示您指定的第二个资源在没有您指定的第一个资源处于活跃状态的情况下无法运行。这是 kind 选项的默认值。保留默认值可确保您指定的第二个资源会在您指定更改状态的第一个资源时响应。

•

如果您指定的第一个资源正在运行并且已停止，则您指定的第二个资源也会停止（如果它正在运行）。

- 如果您指定的第一个资源没有运行，且无法启动，则您指定的资源将会停止（如果正在运行）。

- 如果您指定的第一个资源在您指定的第二个资源正在运行时启动，则您指定的第二个资源将会停止并重启。

但请注意，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。

7.2.2. 公告排序

当为顺序约束指定 **kind=Optional** 选项时，约束被视为可选，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** 的资源配置公告排序约束。

```
# pcs constraint order VirtualIP then dummy_resource kind=Optional
```

7.2.3. 排序的资源集

一个常见的情况是，管理员创建排序的资源链，例如资源 **A** 在资源 **C** 之前启动。如果您的配置需要创建一组在一起并启动的资源，您可以配置包含这些资源的资源组，如 第 6.5 节 “资源组” 所述。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** 命令在一组或一组资源中创建顺序约束。

您可以使用 **pcs constraint order set** 命令为一组资源设置以下选项。

- **sequential**，它可以设为 **true** 或 **false**，以指示资源集合是否相互排序。

  将 **sequential** 设置为 **false** 允许在顺序约束中相对于其他集合对集合进行排序，而不对成员进行排序。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是否在继续前处于活动状态。将 **require-all** 设置为 **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设置。将 **require-all** 设置为 **false** 无效，除非与未排序的集合结合使用，这些集合的 **序列** 设置为 **false**。

- **操作**，它可以设置为 启动、提升、降级 或停止，如 表 7.3 “顺序约束的属性” 所述。

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：

- **分数** 表示此约束的首选程度。有关这个选项的详情请参考 表 7.4 “Colocation 约束的属性”。

```
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
```

7.2.4. 从排序约束中删除资源

使用以下命令从任何排序约束中删除资源。

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

7.3. 资源共存

共存约束决定一个资源的位置取决于另一个资源的位置。

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 [master|slave] source_resource with [master|slave]
target_resource [score] [options]
```

有关 master 和从资源的详情请参考 第 9.2 节 “多状态资源：具有多个模式的资源”。

表 7.4 “Colocation 约束的属性”. 总结了配置 colocation 约束的属性和选项。

表 7.4. Colocation 约束的属性

项	描述
source_resource	colocation 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	colocation 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。

项	描述
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。值 +INFINITY （默认值）表示 source_resource 必须在与 target_resource 相同的节点上运行。值 -INFINITY 表示 source_resource 不得在与 target_resource 相同的节点上运行。

7.3.1. 强制放置

当约束分数为 +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 没有激活的情况。

如果您需要 myresource1 始终与 myresource2 在同一台机器中运行，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 无法在任何群集节点上运行（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集群，其中 myresource1 无法与 myresource2 在同一计算机上运行。在这种情况下，使用 分数=-INFINITY

```
# pcs constraint colocation add myresource1 with myresource2 score=-INFINITY
```

同样，通过指定 -INFINITY，约束会绑定。因此，如果唯一要运行的地方是 myresource2 已经是，则 myresource1 可能无法在任何位置运行。

7.3.2. 公告放置

如果强制放置是 "must" 和 "must not"，则公告放置是 "I would prefer if" 的替代。对于分数大于 -INFINITY 且少于 INFINITY 的限制，群集将尝试满足您的希望，但如果您的替代方案是停止某些集群资源，则可能会忽略它们。公告共存限制可与配置的其他元素组合，以像强制一样运作。

7.3.3. 资源共存集合

如果您的配置需要创建一组在一起并启动的资源，您可以配置包含这些资源的资源组，如 第 6.5 节 “资源组” 所述。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，它必须与资源 A 或 B 共同启动，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 命令在一组或一组资源中创建 colocation 约束。

您可以使用 pcs constraint colocation set 命令为一组资源设置以下选项。

- 顺序，它可以设为 true 或 false，以指示集合成员是否必须相互在一起。

  将 sequential 设置为 false 允许此集合的成员与稍后列出的另一个集合在一起，无论此集合中的哪个成员处于活动状态。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。

- 角色，它可以设置为 Stopped、Started、master 或 Slave。有关多状态资源的详情请参考 第 9.2 节 “多状态资源：具有多个模式的资源”。

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。

- kind，以指示如何强制实施约束。有关这个选项的详情请参考 表 7.3 “顺序约束的属性”。
- 对称，指示停止资源的顺序。如果为 true（默认值），按相反顺序停止资源。默认值为：true

- ID，为您定义的约束提供名称：

当列出集合的成员时，每个成员都与其前一个处于共同位置。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]]
```

7.3.4. 删除重新定位限制

使用以下命令删除使用 source_resource 的 colocation 约束。

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

7.4. 显示限制

您可以使用一些命令来显示已经配置的约束。

以下命令列出所有当前位置、顺序和 colocation 约束。

```
pcs constraint list|show
```

以下命令列出所有当前位置限制。

- 如果指定了资源，则每个资源会显示位置限制。这是默认的行为。
- 如果指定了节点，则每个节点会显示位置限制。
- 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|nodes [specific nodes|resources]] [--full]
```

-

以下命令列出所有当前排序限制。如果指定了 --full 选项，显示内部约束 ID。

```
pcs constraint order show [--full]
```

以下命令列出所有当前的 colocation 约束。如果指定了 --full 选项，显示内部约束 ID。

```
pcs constraint colocation show [--full]
```

以下命令列出引用特定资源的约束。

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

第 8 章 管理集群资源

本章介绍了可以用来管理集群资源的各种命令。它提供关于以下步骤的信息。

- 第 8.1 节 “手动在集群中移动资源”
- 第 8.2 节 “因为失败而移动资源”
- 第 8.4 节 “启用、禁用和禁止集群资源”
- 第 8.5 节 “禁用 monitor 操作”

8.1. 手动在集群中移动资源

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。有关将集群节点放在待机节点的详情请参考 第 4.4.5 节 “待机模式”。

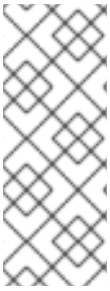
您可以用下列方式之一移动独立指定的资源。

- 您可以使用 pcs resource move 命令将资源从当前运行的节点中移出，如 第 8.1.1 节 “从当前节点移动资源” 所述。
- 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、限制、资源位置和其他设置决定。有关这个命令的详情请参考 第 8.1.2 节 “将资源移动到首选节点”。

8.1.1. 从当前节点移动资源

要将资源从当前运行的节点中移动，请使用以下命令，指定定义的 resource_id。如果要指定在哪个节点上运行您要移动的资源，指定 destination_node。

```
pcs resource move resource_id [destination_node] [--master] [lifetime=lifetime]
```



注意

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中运行。您可以执行 pcs resource clear 或 pcs constraint delete 命令删除约束。这不一定将资源重新移到原始节点；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 move 命令的 --master 参数，则约束的范围仅限于 master 角色，您必须指定 master_id 而不是 resource_id。

您可选择为 pcs resource move 命令配置 Life 参数，以指示约束应保留的时间。根据 ISO 8601 中定义的格式指定 Life 参数 的单元，它要求您将单位指定为大写字母，例如 Y（年）、M（月）、W（周）、D（天）、H（小时）、M（分钟）和 S（秒）。

为了将分钟（M）与月（M）区分开，需要在分钟值前添加 PT 来指定。例如，生命周期 参数为 5M 表示 5 个月的间隔，而 PT5M 的生命周期 参数则表示间隔为五分钟。

Life 参数 按照 cluster-recheck-interval 集群属性定义的间隔进行检查。默认值为 15 分钟。如果您的配置需要更频繁地检查这个参数，您可以使用以下命令重置这个值。

```
pcs property set cluster-recheck-interval=value
```

您可以选择为 pcs resource move 命令配置 --wait[=n] 参数，以指示在返回 0（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。

以下命令将资源 resource1 移到 node example-node2，并阻止它重新移至最初运行 1 小时和 30 分钟的节点。

```
pcs resource move resource1 example-node2 lifetime=PT1H30M
```

以下命令将资源 resource1 移到 node example-node2，并阻止它重新移至最初运行 30 分钟的节点。

```
pcs resource move resource1 example-node2 lifetime=PT30M
```

有关资源限制的详情请参考 第 7 章 资源约束。

8.1.2. 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 命令创建的所有限制，您可以输入 pcs resource relocate clear 命令。要显示资源的当前状态及其最佳节点忽略资源粘性，请输入 pcs resource relocate show 命令。

8.2. 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，使其在定义多个故障后移至新节点。达到阈值后，这个节点将不再被允许运行失败的资源，直到：

- 管理员使用 pcs resource failcount 命令手动重置资源的故障计数。
- 达到资源的 failure-timeout 值。

migration-threshold 的值默认设置为 INFINITY。INFINITY 在内部被定义为一个非常大但有限的数字。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，其中资源移动到另一个位置而不丢失状态。

以下示例在名为 **dummy_resource** 的资源中添加了一个迁移阈值 **10**，这表示资源将在 **10** 个故障后移到新节点。

```
# 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 migration-threshold=10
```

要确定资源当前的故障状态和限值，请使用 **pcs resource failcount** 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** 设为 **true** （默认值），启动失败会导致故障 计数 设置为 **INFINITY**，因此始终会导致资源立即移动。有关 **start-failure-is-fatal** 选项的详情请参考 表 12.1 “集群属性”。

停止失败会稍有不同，且非常关键。如果资源无法停止，并且启用了 **STONITH**，那么集群将隔离该节点以便可以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。

8.3. 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** 资源使用相同名称的系统实用程序来测试是否可以访问（由 **DNS** 主机名或 **IPv4/IPv6** 地址指定）的计算机列表，并使用结果维护名为 **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。

表 6.1 “资源属性” 描述您可以为 **ping** 资源设置的属性。

表 8.1. 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	要联系的机器以确定当前的连接状态。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，以便资源在所有集群节点中运行。

```
# 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
```

8.4. 启用、禁用和禁止集群资源

除了 第 8.1 节 “手动在集群中移动资源” 中描述的 pcs resource move 和 pcs resource relocate 命令外，您还可以使用其他各种命令来控制集群资源的行为。

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 将等待 'n' 秒以便资源停止，然后如果资源停止，则返回 0 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]]
```

您可以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 将等待 'n' 秒以便资源启动，然后如果资源启动，则返回 0 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]]
```

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] [--master] [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，以防止其在指定节点上运行。您可以执行 pcs resource clear 或 pcs constraint delete 命令删除约束。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。有关资源限制的详情请参考 第 7 章 资源约束。

如果您指定 pcs resource ban 命令的 --master 参数，则约束的范围仅限于 master 角色，您必须指定 master_id 而不是 resource_id。

您可选择为 pcs resource ban 命令配置 Life 参数，以指示约束应保留的时间。有关为 Life 参数 指定单位以及指定要检查 生命周期 参数的间隔的详情请参考 第 8.1 节 “手动在集群中移动资源”。

您可以选择为 pcs resource ban 命令配置 --wait[=n] 参数，以指示在返回 0（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。

您可以使用 pcs resource 命令的 debug-start 参数强制指定的资源在当前节点上启动，忽略群集建议并打印启动资源的输出。这主要用于调试资源；群集上启动资源总是（几乎）由 Pacemaker 完成，而不是直接通过 pcs 命令完成。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
```

8.5. 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，使用 pcs resource update 命令将 enabled="false" 添加到操作的定义中。当您要重新恢复监控操作时，请将 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，您没有特别调用的任何选项都将重置为默认值。例如，如果您已经配置了自定义超时值 600 的监控操作，运行以下命令可将超时值重置为默认值 20（或通过 pcs resource ops default 命令将默认值设置为 ）。

```
# 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
```

8.6. 受管资源

您可以将资源设置为 非受管 模式，这表示资源仍然在配置中，但 Pacemaker 不管理该资源。

以下命令将指定的资源设置为 非受管 模式。

```
pcs resource unmanage resource1  [resource2] ...
```

以下命令将资源设置为 受管 模式，这是默认状态。

```
pcs resource manage resource1  [resource2] ...
```

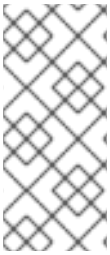
您可以使用 pcs resource manage 或 pcs resource unmanage 命令来指定资源组的名称。命令将对组中的所有资源执行操作，以便您可以通过单个命令将组中的所有资源设置为 受管 或非 受管 模式，然后单独管理包含的资源。

第 9 章 高级配置

本章论述了 Pacemaker 支持的高级资源类型和高级配置功能。

9.1. 资源克隆

您可以克隆资源，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：从共享内存设备挂载非集群文件系统（如 ext4 ）的 Filesystem 资源不应克隆。由于 ext4 分区不知道集群，因此此文件系统不适用于同时从多个节点发生的读写操作。

9.1.1. 创建和删除克隆的资源

您可以使用以下命令同时创建资源以及该资源的克隆。

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

克隆的名称为 resource_id-clone。

您不能在单个命令中创建资源组以及该资源组的克隆。

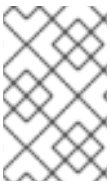
另外，您可以使用以下命令创建之前创建的资源或资源组的克隆。

```
pcs resource clone resource_id | group_name [clone_options]...
```

克隆的名称为 resource_id-clone 或 group_name-clone。

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。

当您创建资源克隆时，克隆使用附加至名称中的 -clone 资源名称。以下命令创建名为 webfarm 的类型为 apache 的资源，以及名为 webfarm-clone 的克隆。

```
# pcs resource create webfarm apache clone
```

注意

当您创建在另一个克隆后排序的资源或资源组克隆时，您应该始终设置 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 | group_name
```

有关资源选项的详情请参考 第 6.1 节 “资源创建”。

表 9.1 “资源克隆选项” 描述您可以为克隆的资源指定的选项。

表 9.1. 资源克隆选项

项	描述
优先级, target-role, is-managed	选项从正在克隆的资源继承，如 表 6.3 “资源元数据选项” 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。

项	描述
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，则在一台机器上运行的克隆的副本不等于另一个实例，无论该实例是在另一个节点上运行还是在同一节点上运行。如果 clone-node-max 值大于一，则默认值为 true ；否则默认值为 false。
ordered	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/主控机之间）的行为，以便在第二个克隆的同一节点上的副本立即启动或停止（而不是等待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。
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在指定数量的原始克隆实例运行后启动，即使 interleave 选项设为 true。

9.1.2. 克隆限制

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。这些限制与常规资源的写法不同，除非必须使用克隆的 id。

以下命令为集群创建一个位置约束，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 实例之前，不会启动任何 webfarm- stats 实例。只有无法启动 webfarm-clone 的副本时，才会阻止 webfarm-stats 处于活动状态。此外，webfarm-clone 在 停止 webfarm-stats 之前将等待停止。

```
# 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的位置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 colocation 约束，以确保资源 webfarm-stats 在与 webfarm-clone 活动副本相同的节点上运行。

```
# pcs constraint colocation add webfarm-stats with webfarm-clone
```

9.1.3. 克隆粘性

为实现稳定的分配模式，克隆默认为稍有粘性。如果未提供资源 粘性 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。

9.2. 多状态资源：具有多个模式的资源

多状态资源是克隆资源的专业化。它们允许实例处于以下两种操作模式之一：它们称为 Master 和 Slave。模式的名称没有特定含义，除了实例启动时的限制外，它必须处于 Slave 状态。

您可以使用以下单个命令将资源创建为主/从克隆：

> **pcs resource create** resource_id standard:provider:type|type **[**resource options**] master [**master_options**]**

master/slave 克隆的名称为 **resource_id-master**。

注意

对于 **Red Hat Enterprise Linux** 版本 **7.3** 及更早版本，请使用以下格式创建主/从克隆：

> **pcs resource create** resource_id standard:provider:type|type **[**resource options**] --master [meta** master_options**]**

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 **master/slave** 资源：使用此命令时，您可以为 **master/slave** 克隆指定一个名称。如果没有指定名称，**master/slave** 克隆的名称将是 **resource_id-master** 或 **group_name-master**。

> **pcs resource master** master/slave_name resource_id|group_name **[**master_options**]**

有关资源选项的详情请参考 第 6.1 节 “资源创建”。

表 9.2 “多状态资源的属性” 描述您可以为多状态资源指定的选项。

表 9.2. 多状态资源的属性

项	描述
id	多状态资源的名称

项	描述
优先级,target-role, is-managed	请参阅 表 6.3 “资源元数据选项”。
clone-max,clone-node-max,notify,globally-unique,order,interleave	请参阅 表 9.1 “资源克隆选项”。
master-max	可以提升资源副本数到 master 状态；默认值 1。
master-node-max	在单个节点上可提升资源副本数到 master 状态；默认值 1。

9.2.1. 监控多状态资源

要仅为 master 资源添加监控操作，您可以在资源中添加额外的 monitor 操作。但请注意，资源中的每个 monitor 操作都必须具有不同的间隔。

以下示例为 ms_resource 配置一个监控器操作，间隔为 11 秒。除了默认的 monitor 操作外，默认监控间隔为 10 秒。

```
# pcs resource op add ms_resource interval=11s role=Master
```

9.2.2. 多状态约束

在大多数情况下，多状态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。

有关资源位置限制的详情请参考 第 7.1 节 “位置限制”。

您可以创建一个 colocation 约束来指定资源是 master 资源还是从资源。以下命令创建了资源 colocation 约束。

```
pcs constraint colocation add [master|slave] source_resource with [master|slave]
target_resource [score] [options]
```

有关 colocation 约束的详情请参考 第 7.3 节 “资源共存”。

在配置包含 multistate 资源的顺序约束时，您可以为资源指定的一个动作被提升， 这表示该资源从 slave 提升到 master。另外，您可以指定 降级 操作，表示资源从主卷降级到从设备。

配置顺序约束的命令如下。

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

有关资源顺序限制的详情请参考 第 7.2 节 “顺序限制”。

9.2.3. 多状态粘性

为实现稳定的分配模式，默认情况下多状态资源会稍微粘性。如果未提供资源 粘性 值，则多状态资源将使用值 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。

9.3. 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。*
- *所有节点都必须有权访问每个受管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。有关配置客户机节点的详情请参考。 第 9.4 节 "pacemaker_remote 服务"

有关配置虚拟主机的详情请参考 虚拟化部署和管理指南。

表 9.3 "虚拟域资源资源选项" 描述您可以为 VirtualDomain 资源配置的资源选项。

表 9.3. 虚拟域资源资源选项

项	默认值	描述
config		（必需）指向此虚拟域的 **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虚拟机监控程序。
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。
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。*注意*：当使用监控脚本时，只有所有监控脚本都成功完成时，**启动 和迁移_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

项	默认值	描述
autoset_utilization_cpu	**true**	如果设置为 **true**，代理将从 **virsh** 检测到 **domainU** 的**vCPU**数，并在执行监控器时将其置于资源的 CPU 使用率中。
autoset_utilization_hv_memory	**true**	如果设置为 true，代理会从 **virsh** 检测到 **Max 内存** 数量，并在执行监控时将其置于源的 **hv_memory** 使用率中。
migrateport	随机高端口	此端口将在 **qemu** 迁移 URI 中使用。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。设置此参数后，虚拟机的 RAM 状态将在停止后保存到快照目录中的文件。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 metadata 选项，以允许将资源实时迁移到另一节点。当此选项设为 true 时，可以迁移资源而不丢失状态。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，然后在第二个节点从节点移动到另一个节点时重新启动。

使用以下步骤创建 VirtualDomain 资源：

1. 要创建 VirtualDomain 资源代理来管理虚拟机，Pacemaker 需要将虚拟机的 xml 配置文件转储到磁盘上的一个文件中。例如，如果您创建了名为 guest1 的虚拟机，请将 xml 转储到主机上的某个位置。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. 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。

3. 使用 pcs resource create 命令配置 VirtualDoman 资源。例如，以下命令配置名为 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进行。

```
# pcs resource create VM VirtualDomain config=.../vm.xml \
      migration_transport=ssh meta allow-migrate=true
```

9.4. PACEMAKER_REMOTE 服务

pacemaker_remote 服务允许没有运行 corosync 的节点集成到集群中，让群集像实际群集节点一样

管理其资源。

pacemaker_remote 服务提供的功能包括：

- pacemaker_remote 服务允许您在超过红帽支持 RHEL 7.7 的 32 个节点支持范围内进行扩展。
- pacemaker_remote 服务允许您将虚拟环境作为集群资源进行管理，还可作为集群资源管理虚拟环境中的单个服务。

以下术语用于描述 pacemaker_remote 服务：

- 群集节点 - 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，无需 corosync 群集成员资格。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 - 在一个可以在 Pacemaker 集群环境中的远程节点和客户机节点（KVM 和 LXC）内执行远程应用程序管理的服务守护进程。这个服务是 Pacemaker 的本地资源管理守护进程(LRMD)的改进版本，它可以在没有运行 corosync 的节点上远程管理资源。
- LXC - 由 libvirt-lxc Linux 容器驱动程序定义的 Linux 容器。

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（虚拟机上不需要的配置）。
- 在群集节点上运行的群集堆栈（pacemaker 和 corosync）连接到远程节点上的 pacemaker_remote 服务，允许它们集成到群集中。

- 在群集节点上运行的群集堆栈（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，允许它们集成到群集中。

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)
- 它们本身不运行完整的 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。集群完全能够在每个远程和客户机节点上管理和监控资源：您可以针对它们构建限制，将它们置于待机状态，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。

9.4.1. 主机和客户机身份验证

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。

从红帽企业 Linux 7.4 开始，pcs cluster node add-guest 命令为客户机节点设置 authkey，而 pcs cluster node add-remote 命令可为远程节点设置 authkey。

9.4.2. 客户机节点资源选项

将虚拟机或 LXC 资源配置为充当客户机节点时，您可以创建一个 VirtualDomain 资源来管理虚拟机。有关您可以为 VirtualDomain 资源设置的选项描述，请参阅 表 9.3 “虚拟域资源资源选项”。

除了 VirtualDomain 资源选项外，元数据选项将资源定义为客户机节点，再定义连接参数。从 Red Hat Enterprise Linux 7.4 开始，您应使用 pcs cluster node add-guest 命令设置这些资源选项。在早于 7.4 的版本中，您可以在创建资源时设置这些选项。表 9.4 “将 KVM/LXC 资源配置为远程节点的元数据选项” 描述这些元数据选项。

表 9.4. 将 KVM/LXC 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。*WARNING*：这个值不能与任何资源或节点 ID 重叠。
remote-port	3121	配置用于 guest 连接 **pacemaker_remote**的自定义端口
remote-addr	**remote-node** 值用作主机名	如果远程节点的名称不是客户机的主机名，则要连接的 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

9.4.3. 远程节点资源选项

远程节点定义为将 ocf:pacemaker:remote 用作 资源代理的集群资源。在 Red Hat Enterprise Linux 7.4 中，您应该使用 pcs cluster node add-remote 命令创建此资源。在早于 7.4 的版本中，您可以使用 pcs resource create 命令创建此资源。表 9.5 “远程节点的资源选项” 描述您可以为 远程 资源配置的资源选项。

表 9.5. 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		要连接的服务器位置。这可以是 IP 地址或主机名。
port		要连接的 TCP 端口。

9.4.4. 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。可以通过将变量放在 /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群集资源也必须使用相同的端口号（对客户机节点使用 remote-port 元数据选项，或远程节点的 port 选项）。

9.4.5. 配置概述：KVM 客户机节点

本节概述了使用 libvirt 和 KVM 虚拟机执行 Pacemaker 启动虚拟机并将该虚拟机整合为客户机节点的步骤。

1. 配置 VirtualDomain 资源，如 第 9.3 节 “将虚拟域配置为资源” 所述。

2. 在运行 Red Hat Enterprise Linux 7.3 及更早版本的系统上，按照以下步骤将路径 /etc/pacemaker/authkey 放置到每个集群节点和虚拟机上。这可保护远程通信和身份验证。

   a. 在每个节点上输入以下一组命令，以创建具有安全权限的 authkey 目录。

   ```
   # mkdir -p --mode=0750 /etc/pacemaker
   # chgrp haclient /etc/pacemaker
   ```

   b. 以下命令显示了创建加密密钥的一种方法：您应该仅创建一次密钥，然后将它复制到所有节点。

   ```
   # dd if=/dev/urandom of=/etc/pacemaker/authkey bs=4096 count=1
   ```

3. 对于 Red Hat Enterprise Linux 7.4，在每个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，并通过防火墙允许 TCP 端口 3121。

   ```
   # yum 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
```

对于 **Red Hat Enterprise Linux 7.3** 和更早版本，在每个虚拟机上运行以下命令，以安装 **pacemaker_remote** 软件包、启动 **pacemaker_remote** 服务并使其在启动时运行，并允许 **TCP** 端口 **3121** 通过防火墙。

```
# yum install pacemaker-remote resource-agents pcs
# systemctl start pacemaker_remote.service
# systemctl enable pacemaker_remote.service
# firewall-cmd --add-port 3121/tcp --permanent
# firewall-cmd --add-port 2224/tcp --permanent
# firewall-cmd --reload
```

4. 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。有关为客户机虚拟机设置静态 **IP** 地址的详情，请参考 虚拟化部署和管理指南。

5. 对于 **Red Hat Enterprise Linux 7.4** 及更高版本，请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中运行，而不必在要添加的客户端节点中运行。除了转换资源外，这个命令会将 **/etc/pacemaker/authkey** 复制到客户机节点，并在客户机节点上启动并启用 **pacemaker_remote** 守护进程。

```
pcs cluster node add-guest hostname resource_id [options]
```

对于 **Red Hat Enterprise Linux 7.3** 及更早版本，使用以下命令将现有的 **VirtualDomain** 资源转换为客户机节点。这个命令必须在集群节点中运行，而不必在要添加的客户端节点中运行。

```
pcs cluster remote-node add hostname resource_id [options]
```

6. 创建 **VirtualDomain** 资源后，您可以像在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。从 **Red Hat Enterprise Linux 7.3** 开始，您可以在组中包括客户机节点，它们允许您对存储设备、文件系统和虚拟机进行分组。

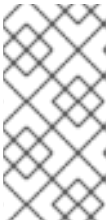
```
# pcs resource create webserver apache configfile=/etc/httpd/conf/httpd.conf op monitor
interval=30s
# pcs constraint location webserver prefers guest1
```

9.4.6. 配置概述：远程节点（红帽企业 Linux 7.4）

本节概述了配置 Pacemaker 远程节点并将该节点集成到 Red Hat Enterprise Linux 7.4 的现有 Pacemaker 集群环境中的步骤。

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 之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。

```
# yum 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中验证 pcs。

```
# pcs cluster 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，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，您可以像在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。

9.4.7. 配置概述：远程节点（红帽企业 Linux 7.3 及更早版本）

本节概述了配置 Pacemaker 远程节点以及将该节点集成到 Red Hat Enterprise Linux 7.3（及更早）系统中的现有 Pacemaker 集群环境中的步骤。

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 之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。

```
# yum install -y pacemaker-remote resource-agents pcs
```

3. 所有节点（集群节点和远程节点）必须安装相同的身份验证密钥才能使通信正常工作。如果

您在现有节点上已有密钥，请使用该密钥并将其复制到远程节点。否则，在远程节点上创建新密钥。

在远程节点上输入以下一组命令，为具有安全权限的身份验证密钥创建目录。

```
# mkdir -p --mode=0750 /etc/pacemaker
# chgrp haclient /etc/pacemaker
```

以下命令显示一种在远程节点上创建加密密钥的方法。

```
# dd if=/dev/urandom of=/etc/pacemaker/authkey bs=4096 count=1
```

4. *在远程节点上启动并启用 **pacemaker_remote** 守护进程。*

```
# systemctl enable pacemaker_remote.service
# systemctl start pacemaker_remote.service
```

5. *在集群节点中，使用与远程节点上的身份验证密钥相同的路径为共享身份验证密钥创建一个位置，并将密钥复制到该目录中。在本例中，密钥从创建密钥的远程节点复制。*

```
# mkdir -p --mode=0750 /etc/pacemaker
# chgrp haclient /etc/pacemaker
# scp remote1:/etc/pacemaker/authkey /etc/pacemaker/authkey
```

6. *在集群节点中输入以下命令来创建 远程 资源。在本例中，远程节点是 **remote1**。*

```
# pcs resource create remote1 ocf:pacemaker:remote
```

7. *创建 远程 资源后，您可以像在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。

8. *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。*

9.4.8. 系统升级和 pacemaker_remote

从 Red Hat Enterprise Linux 7.3 开始，如果 pacemaker_remote 服务在活跃的 Pacemaker 远程节点上停止，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，群集将立即尝试重新连接。如果在资源监控器超时内没有重启 pacemaker_remote，集群会将监控器操作视为失败。

如果要避免在活跃的 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。

警告

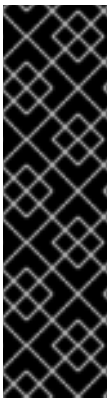
对于 Red Hat Enterprise Linux 版本 7.2 及更早版本，如果 pacemaker_remote 在当前集成到群集的节点中停止，则群集将隔离该节点。如果作为 yum 更新 过程的一部分自动发生停止，则系统可能会处于不可用状态（特别是内核也与 pacemaker_remote同时升级）。对于 Red Hat Enterprise Linux 版本 7.2 及更早版本，您必须使用以下步骤使节点退出集群，然后才能执行任何可能停止 pacemaker_remote 的系统管理。

1. *使用 pcs resource disable resourcename 停止节点的连接资源，这将将所有服务移出该节点。对于客户机节点，这也会停止虚拟机，因此虚拟机必须在集群外启动（例如，使用 virsh）来执行任何维护。*

2. 执行所需的维护。

3. 当准备好将节点返回到群集时，请使用 pcs resource enable 重新启用该资源。

9.5. DOCKER 容器的 PACEMAKER 支持（技术预览）



重要

对 Docker 容器的 Pacemaker 支持仅用于技术预览。有关"技术预览"含义的详情，请参阅 技术预览功能支持范围。

这个功能有一个例外是技术预览：与 Red Hat Enterprise Linux 7.4 一样，红帽完全支持在 Red Hat Openstack Platform(RHOSP)部署中使用 Pacemaker 捆绑包。

Pacemaker 支持使用任何所需的基础架构启动 Docker 容器的特殊语法：该捆绑包。创建 Pacemaker 捆绑包后，您可以创建一个捆绑包封装的 Pacemaker 资源。

- 第 9.5.1 节 “配置 Pacemaker 捆绑包资源” 描述创建 Pacemaker 捆绑包的命令语法，并提供表总结您可以为每个捆绑包参数定义的参数。
- 第 9.5.2 节 “在捆绑包中配置 Pacemaker 资源” 提供有关配置 Pacemaker 捆绑包中包含的资源的信息。
- 第 9.5.3 节 “Pacemaker 捆绑包的限制” 请注意 Pacemaker 捆绑包的限制。
- 第 9.5.4 节 “Pacemaker 捆绑包配置示例” 提供 Pacemaker 捆绑包配置示例。

9.5.1. 配置 Pacemaker 捆绑包资源

为 Docker 容器创建 Pacemaker 捆绑包的命令语法如下：此命令会创建一个捆绑包，封装其他资源。有关在捆绑包中创建集群资源的详情请参考 第 9.5.2 节 “在捆绑包中配置 Pacemaker 资源”。

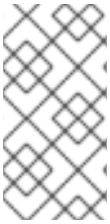
> *pcs resource bundle create bundle_id container docker [container_options] [network network_options] [port-map port_options]... [storage-map storage_options]... [meta meta_options] [--disabled] [--wait[=n]]*

*所需的 **bundle_id** 参数必须是捆绑包的唯一名称。如果指定了 **--disabled** 选项，则捆绑包不会自动启动。如果指定了 **--wait** 选项，**Pacemaker** 将等待最多 **n** 秒以启动捆绑包，然后成功返回 **0** 或 **1** 出错。如果未指定 **n**，则默认为 **60** 分钟。*

*以下小节描述了您可以为 **Pacemaker** 捆绑包的每个元素配置的参数。*

9.5.1.1. Docker 参数

*表 9.6 "Docker 容器参数" 描述您可以为捆绑包设置的 **docker** 容器选项。*



注意

*在 **Pacemaker** 中配置 **docker bundle** 前，您必须安装 **Docker**，并在允许运行捆绑包的每个节点上提供完全配置的 **Docker** 镜像。*

表 9.6. Docker 容器参数

项	默认值	描述
Image		Docker 镜像标签（必需）
replicas	如果这是正则，则为 **promote-max** 值，否则为 1。	指定一个正整数，指定要启动的容器实例数
replicas-per-host	1	指定允许在一个节点上运行的容器实例的正整数
promoted-max	0	一个非负整数，如果为正，则表示容器化服务应被视为多状态服务，且此副本数允许在 master 角色中运行该服务
网络		如果指定，它将传递到 **docker run** 命令，作为 Docker 容器的网络设置。
run-command	如果捆绑包包含资源，则 **/usr/sbin/pacemaker_remoted**	启动后，该命令将在容器内运行("PID 1")。如果捆绑包中包含资源，此命令必须启动 **pacemaker_remoted** 守护进程（但也可以是执行其他任务的脚本）。

项	默认值	描述
选项		传递给 **docker run** 命令的额外命令行选项

9.5.1.2. 捆绑包网络参数

表 9.7 "捆绑包资源网络参数" 描述您可以为捆绑包设置的 网络 选项。

表 9.7. 捆绑包资源网络参数

项	默认值	描述
add-host	TRUE	如果使用 TRUE 和 **ip-range-start**，Pacemaker 将自动确保容器内的 **/etc/hosts** 文件为每个副本名称及其分配的 IP 具有条目。
ip-range-start		如果指定，Pacemaker 将为每个容器实例创建一个隐式 **ocf:heartbeat:IPaddr2** 资源，从这个 IP 地址开始，使用指定为 Docker 元素 **replicas** 参数的任意连续地址。这些地址可以从主机的网络访问容器内的服务，尽管无法在容器本身中看到。目前仅支持 IPv4 地址。
host-netmask	32	如果指定了 **ip-range-start**，则会使用此 CIDR 子网掩码（以位数为单位）创建 IP 地址。
host-interface		如果指定了 **ip-range-start**，则会在此主机接口上创建 IP 地址（默认情况下，它将从 IP 地址确定）。
control-port	3121	如果捆绑包包含 Pacemaker 资源，集群将使用这个整数 TCP 端口与容器内的 Pacemaker 远程进行通信。当容器无法侦听默认端口时（当容器使用主机的网络而不是 **ip-range-start（在这种情况下，replicas- per-host** 必须为 1）或捆绑包可以在已侦听默认端口的 Pacemaker 远程节点上运行时，更改此设置非常有用。在主机上或容器中设置的任何 **PCMK_remote_port** 环境变量都会在捆绑包连接中被忽略。 当 Pacemaker 捆绑包配置使用 **control-port** 参数时，如果捆绑包有其自身的 IP 地址，则需要在该 IP 地址和所有运行 corosync 的完整群集节点上打开端口。如果捆绑包设置了 **network="host"** 容器参数，则需要在每个集群节点的 IP 地址上打开该端口。

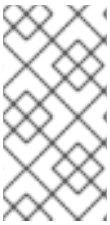
注意

副本通过捆绑包 ID 加上破折号和整数计数器命名，以零开头。例如，如果名为 httpd-bundle 的捆绑包配置了 replicas=2，则其容器将命名为 httpd-bundle-0 和 httpd-bundle-1。

除了网络参数外，您还可以为捆绑包指定 port-map 参数。表 9.8 "捆绑包资源端口映射参数" 描述这些 port-map 参数。

表 9.8. 捆绑包资源端口映射参数

项	默认值	描述
id		端口映射的唯一名称（必需）
port		如果指定，则主机网络上此 TCP 端口号的连接（容器分配的 IP 地址上，如果指定了 **ip-range-start** ）将转发到容器网络。正好一个 **端口** 或 **范围** 必须在端口映射中指定。
internal-port	**端口**值	如果指定了 **端口 和内部 端口，则** 到主机网络上的端口的连接将转发到容器网络上的此端口。
范围		如果指定了 **范围**，则主机网络上（如果指定了 **ip -range-start，则表示为 first_port-last_port** ）的连接将转发到容器网络中相同的端口。正好一个 **端口** 或 **范围** 必须在端口映射中指定。



注意

如果捆绑包包含资源，Pacemaker 将自动映射 control-port，因此不需要在端口映射中指定该端口。

9.5.1.3. 捆绑包存储参数

您可选择为捆绑包配置 storage-map 参数。表 9.9 "捆绑包资源存储映射参数" 描述这些参数。

表 9.9. 捆绑包资源存储映射参数

项	默认值	描述
id		存储映射的唯一名称（必需）
source-dir		将映射到容器中的主机文件系统的绝对路径。在配置 **storage -map 参数时，必须指定 source-dir 和 source-dir- root** 参数之一。
source-dir-root		主机文件系统上的路径的开头，该路径将映射到容器，每个容器实例在主机上使用不同的子目录。子目录的名称与捆绑包名称相同，外加破折号和整数计数器（以 0 开头）。在配置 **storage -map 参数时，必须仅指定一个 source-dir 和 source-dir- root** 参数。
target-dir		映射主机存储的容器内的路径名称（必需）

项	默认值	描述
选项		映射存储时使用的文件系统挂载选项

例如，如何使用 source-dir- root 参数命名主机上的子目录，如果 source-dir -root=/path/to/my/directory,target-dir=/srv/appdata，捆绑包将命名为 mybundle 且 replicas=2，集群将创建两个容器 主机名为 mybundle-0 和 mybundle-1 的实例，并在运行容器的主机上创建两个目录：/path/to/my/directory/mybundle-0 和 /path/to/my/directory/mybundle-1。每个容器将获得其中一个目录，容器内运行的任何应用程序都将该目录视为 /srv/appdata。

注意

如果主机上还没有源目录，Pacemaker 不会定义行为。但是，在此情况下，容器技术或其资源代理应该会创建源目录。

注意

如果捆绑包包含 Pacemaker 资源，Pacemaker 将自动将相当于 source-dir=/etc/pacemaker/authkeytarget-dir=/etc/pacemaker/authkey 和 source-dir-root=/var/log/pacemaker/bundlestarget-dir=/var/log 映射到容器中，因此在配置 storage-map 参数时不需要指定其中的路径。

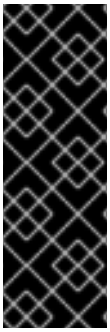
重要

在群集的任何节点上，PCMK_authkey_location 环境变量不得设置为 /etc/pacemaker/authkey 默认值。

9.5.2. 在捆绑包中配置 Pacemaker 资源

捆绑包可以选择性地包含一个 Pacemaker 集群资源。与捆绑包中未包含的资源一样，集群资源可能定义有操作、实例属性和元数据属性。如果捆绑包包含资源，容器镜像必须包含 Pacemaker Remote 守护进程，并且必须在捆绑包中配置 ip-range -start 或 control-port。Pacemaker 会为连接创建一个隐式 ocf:pacemaker:remote 资源，在容器内启动 Pacemaker Remote，并通过 Pacemaker Remote 监控和管理资源。如果捆绑包有多个容器实例（副本），Pacemaker 资源将充当隐式克隆，如果捆绑包将 提升-max 选项配置为大于零，则该克隆将是一个多状态克隆。

您可以通过为命令指定 bundle 参数以及包含该资源的捆绑包 ID，使用 pcs resource create 命令在 Pacemaker 捆绑包中创建资源。有关创建包含资源的 Pacemaker 捆绑包的示例，请参阅 第 9.5.4 节 “Pacemaker 捆绑包配置示例”。



重要

包含资源的捆绑包中的容器必须具有可访问的网络环境，以便集群节点上的 Pacemaker 可以与容器内的 Pacemaker 远程联系。例如，docker 选项 --net=none 不应用于资源。默认值（使用容器内的不同网络空间）与 ip-range-start 参数相结合。如果使用 docker 选项 --net=host （使容器共享主机的网络空间），则应为每个捆绑包指定一个唯一的 control-port 参数。任何防火墙都必须允许访问 control-port。

9.5.2.1. 节点属性和捆绑包资源

如果捆绑包包含集群资源，则资源代理可能需要设置节点属性，如 master 分数。但是，对于容器而言，哪一个节点应该获取 属性并不明显。

如果容器使用的共享存储相同，无论容器托管在哪个节点上，都可在捆绑包节点上使用 master 分数。另一方面，如果容器使用从底层主机导出的存储，那么在底层主机上使用 master 分数可能更为适当。由于这取决于特定的情况，因此 container-attribute-target 资源 metadata 属性允许用户指定要使用的方法。如果设置为 host，则在底层主机上检查用户定义的节点属性。如果是任何其他节点，则使用本地节点（本例中为捆绑包节点）。这个行为只适用于用户定义的属性；集群总是检查本地节点是否有集群定义的属性，如 #uname。

如果将 container-attribute-target 设置为 host，集群会将额外的环境变量传递给资源代理，以便它能够适当地设置节点属性。

9.5.2.2. 元数据属性和捆绑包资源

捆绑包上设置的任何元数据属性都将由捆绑包中包含的资源继承，以及 Pacemaker 为捆绑包创建的任何资源。这包括 优先级、target-role 和 is-managed 等选项。

9.5.3. Pacemaker 捆绑包的限制

Pacemaker 捆绑包在以下限制下运行：

- 捆绑包不能包含在组中，或者通过 pcs 命令显式克隆。这包括捆绑包包含的资源，以及 Pacemaker 为捆绑包隐式创建的任何资源。但请注意，如果捆绑包配置了大于一 的副本 值，则捆绑包的行为就像是一个克隆一样。
- 当捆绑包不是非受管或集群处于维护模式时重启 Pacemaker 可能会导致捆绑包失败。

- 捆绑包没有实例属性、使用属性或操作，尽管捆绑包中包含的资源可能有它们。

- 只有捆绑包使用不同的 control-port 时，包含资源的捆绑包才能在 Pacemaker 远程节点上运行。

9.5.4. Pacemaker 捆绑包配置示例

以下示例创建一个 Pacemaker 捆绑包 资源，其捆绑包 ID 为 httpd-bundle，其中包含资源 ID 为 httpd 的 ocf:heartbeat:apache 资源。

此流程需要以下先决条件配置：

- 在群集的每个节点上安装并启用了 Docker。

- 现有 Docker 镜像名为 pcmktest:http

- 容器镜像包括 Pacemaker 远程守护进程。

- 容器镜像包含已配置的 Apache Web 服务器。

- 集群中的每个节点都有目录 /var/local/containers/httpd-bundle-0、/var/local/containers/httpd-bundle-1 和 /var/local/containers/httpd-bundle-2，其中包含 web 服务器 root 的 index.html 文件。在生产中，更有可能使用一个共享的文档根目录，但示例中，此配置允许您使每个主机上的 index.html 文件与众不同，以便您可以连接到 Web 服务器并验证是否提供了 index.html 文件。

此流程为 Pacemaker 捆绑包配置以下参数：

- 捆绑包 ID 是 httpd-bundle。

- 之前配置的 Docker 容器镜像为 pcmktest:http。

- 本例将启动三个容器实例：

- 本例会将命令行选项 --log-driver=journald 传递给 docker run 命令。此参数不是必需的，但用于演示如何将额外选项传递给 docker 命令。值为 --log-driver=journald 表示容器内的系统日志将记录在底层主机的 systemd 日志中。

- Pacemaker 将创建三个连续隐式 ocf:heartbeat:IPaddr2 资源，每个容器镜像一个，从 IP 地址 192.168.122.131 开始。

- IP 地址在主机接口 eth0 上创建。

- IP 地址使用 CIDR 子网掩码 24 创建。

- 这个示例创建了一个端口映射 ID http-port ；到容器分配的 IP 地址上的端口 80 的连接将转发到容器网络。

- 本例创建存储映射 ID httpd-root。对于这个存储映射：

  - source-dir-root 的值是 /var/local/containers，它使用每个容器实例主机上的不同子目录指定主机文件系统上的路径的开头。

  - target-dir 的值是 /var/www/html，它指定要映射主机存储的容器中的路径名称。

  - 在映射存储时，将使用文件系统 rw 挂载选项。

  - 由于此示例容器包含一个资源，Pacemaker 将自动在容器中映射 source-dir=/etc/pacemaker/authkey 等效项，因此您不需要在存储映射中指定该路径。

在本例中，现有集群配置被放入名为 temp-cib.xml 的临时文件中，然后复制到名为 temp-cib.xml.deltasrc 的文件中。对集群配置的所有修改都对tmp-cib.xml 文件进行。当 udpates 完成后，这个过程使用 pcs cluster cib-push 命令的 diff-against 选项，以便只有配置文件的更新推送到活动的配置文件。

```
# pcs cluster cib tmp-cib.xml
# cp tmp-cib.xml tmp-cib.xml.deltasrc
# pcs -f tmp.cib.xml resource bundle create httpd-bundle \
container docker image=pcmktest:http replicas=3 \
options=--log-driver=journald \
network ip-range-start=192.168.122.131 host-interface=eth0 \
host-netmask=24 port-map id=httpd-port port=80 \
storage-map id=httpd-root source-dir-root=/var/local/containers \
target-dir=/var/www/html options=rw \
# pcs -f tmp-cib.xml resource create httpd ocf:heartbeat:apache \
statusurl=http://localhost/server-status bundle httpd-bundle
# pcs cluster cib-push tmp-cib.xml diff-against=tmp-cib.xml.deltasrc
```

9.6. 使用和放置策略

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。此分配分数源自因素的组合，包括 资源限制、资源粘性 设置、各个节点上的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，如果将资源设置为其合并要求超过提供容量，则可能无法完全启动，或者可能会以降低性能运行。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

以下小节描述了如何配置这些组件。

9.6.1. 利用率属性

要配置节点提供或需要资源的容量，您可以对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。

从 Red Hat Enterprise Linux 7.3 开始，您可以使用 pcs 命令设置使用属性。

以下示例为两个节点配置 CPU 容量的使用属性，命名属性 cpu。它还配置 RAM 容量的使用属性，命名属性 内存。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，RAM 容量为 1024
- 资源 dummy-medium 需要 2 个 CPU，2048 RAM
- 资源 dummy-large 需要 1 个 CPU 和 3072 RAM

```
# 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。

9.6.2. 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。有关设置集群属性的详情请参考 第 12 章 Pacemaker 集群属性。

placement-strategy 集群属性有四个值：

- 默认值 - 根本不考虑使用值。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。
- 利用率 - 只有在决定节点是否被视为有资格时才会考虑使用值（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。
- balance - 在决定节点是否有资格提供资源以及负载平衡时，会考虑使用值，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时才会考虑使用值。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 的值设置为 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
```

9.6.3. 资源分配

以下小节概述了 Pacemaker 如何分配资源。

9.6.3.1. 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 均衡 ：
  - 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 最小，则 CIB 中列出的第一个有资格的节点会首先被消耗。

9.6.3.2. 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。

9.6.3.3. 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。有关为资源设置优先级的详情请参考 表 6.3 “资源元数据选项”。
- 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。

9.6.4. 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，在配置系统时应考虑以下事项。

- 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。有关设置资源优先级的详情请参考 表 6.3 “资源元数据选项”。

9.6.5. NodeUtilization 资源代理（红帽企业 Linux 7.4 及更高版本）

红帽企业 Linux 7.4 支持 NodeUtilization 资源代理。NodeUtilization 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信息，请运行 pcs resource describe NodeUtilization 命令。

9.7. 为不由 PACEMAKER 管理的资源依赖项配置启动顺序（RED HAT ENTERPRISE LINUX 7.4 及更新的版本）

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。

从 Red Hat Enterprise Linux 7.4 开始，您可以通过 systemd resource-agents-deps 目标将您的启动顺序配置为在这种情况下。您可以为此目标创建一个 systemd 置入单元，Pacemaker 会根据这个目标自行排序。

例如，如果集群包含不受集群管理的外部服务 foo 的资源，您可以创建包含以下内容的 drop-in 单元 /etc/systemd/system/resource-agents-deps.target.d/foo.conf ：

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

创建置入单元后，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，您可能依赖于在 /srv 中挂载文件系统，在这种情况下，根据 systemd 文档 为其创建一个 systemd file srv.mount，然后创建一个置入单元，如 .conf 文件中使用 srv.mount 而不是 foo.service 文件所述，以确保 Pacemaker 在挂载磁盘后启动。

9.8. 使用 SNMP 查询 PACEMAKER 集群（RED HAT ENTERPRISE LINUX 7.5 及更新的版本）

从 Red Hat Enterprise Linux 7.5 开始，您可以使用 pcs_snmp_agent 守护进程通过 SNMP 查询 Pacemaker 集群的数据。pcs_snmp_agent 守护进程是一个 SNMP 代理，通过代理 x 协议连接到主代理 (snmpd)。pcs_snmp_agent 代理不充当独立代理，因为它仅向主代理提供数据。

以下流程为系统设置基本配置，以便在 Pacemaker 集群中使用 SNMP。您可以在集群的每个节点上运行此步骤，您将使用 SNMP 为集群获取数据。

1. 在群集的每个节点上安装 pcs-snmp 软件包。这还将安装提供 sn mp 守护进程的 net-sn mp 软件包。

```
# yum install pcs-snmp
```

2. 将以下行添加到 /etc/snmp/snmpd.conf 配置文件，以将 snmpd 守护进程设置为 主代理x。

```
master agentx
```

3. 将以下行添加到 /etc/snmp/snmpd.conf 配置文件，以在同一 SNMP 配置中启用 pcs_snmp_agent。

```
view    systemview    included   .1.3.6.1.4.1.32723.100
```

4. 启动 pcs_snmp_agent 服务。

```
# systemctl start pcs_snmp_agent.service
# systemctl enable pcs_snmp_agent.service
```

5. 要检查配置，请使用 pcs status 显示群集的状态，然后尝试从 SNMP 获取数据，以检查它是否与输出对应。请注意，当使用 SNMP 获取数据时，只会提供原始资源。

   以下示例显示了在运行中的群集上使用失败操作的 pcs status 命令的输出结果。

```
# pcs status
Cluster name: rhel75-cluster
Stack: corosync
Current DC: rhel75-node2 (version 1.1.18-5.el7-1a4ef7d180) - partition with quorum
Last updated: Wed Nov 15 16:07:44 2017
Last change: Wed Nov 15 16:06:40 2017 by hacluster via cibadmin on rhel75-node1

2 nodes configured
14 resources configured (1 DISABLED)

Online: [ rhel75-node1 rhel75-node2 ]

Full list of resources:
```

```
fencing        (stonith:fence_xvm):    Started rhel75-node1
dummy5 (ocf::pacemaker:Dummy): Stopped (disabled)
dummy6 (ocf::pacemaker:Dummy): Stopped
dummy7 (ocf::pacemaker:Dummy): Started rhel75-node2
dummy8 (ocf::pacemaker:Dummy): Started rhel75-node1
dummy9 (ocf::pacemaker:Dummy): Started rhel75-node2
Resource Group: group1
    dummy1     (ocf::pacemaker:Dummy): Started rhel75-node1
    dummy10    (ocf::pacemaker:Dummy): Started rhel75-node1
Clone Set: group2-clone [group2]
    Started: [ rhel75-node1 rhel75-node2 ]
Clone Set: dummy4-clone [dummy4]
    Started: [ rhel75-node1 rhel75-node2 ]

Failed Actions:
* dummy6_start_0 on rhel75-node1 'unknown error' (1): call=87, status=complete,
exitreason='',
   last-rc-change='Wed Nov 15 16:05:55 2017', queued=0ms, exec=20ms
```

```
# snmpwalk -v 2c -c public localhost PACEMAKER-PCS-V1-MIB::pcmkPcsV1Cluster
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterName.0 = STRING: "rhel75-cluster"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterQuorate.0 = INTEGER: 1
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterNodesNum.0 = INTEGER: 2
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterNodesNames.0 = STRING: "rhel75-node1"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterNodesNames.1 = STRING: "rhel75-node2"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterCorosyncNodesOnlineNum.0 = INTEGER: 2
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterCorosyncNodesOnlineNames.0 = STRING:
"rhel75-node1"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterCorosyncNodesOnlineNames.1 = STRING:
"rhel75-node2"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterCorosyncNodesOfflineNum.0 = INTEGER: 0
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterPcmkNodesOnlineNum.0 = INTEGER: 2
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterPcmkNodesOnlineNames.0 = STRING:
"rhel75-node1"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterPcmkNodesOnlineNames.1 = STRING:
"rhel75-node2"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterPcmkNodesStandbyNum.0 = INTEGER: 0
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterPcmkNodesOfflineNum.0 = INTEGER: 0
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterAllResourcesNum.0 = INTEGER: 11
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterAllResourcesIds.0 = STRING: "fencing"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterAllResourcesIds.1 = STRING: "dummy5"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterAllResourcesIds.2 = STRING: "dummy6"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterAllResourcesIds.3 = STRING: "dummy7"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterAllResourcesIds.4 = STRING: "dummy8"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterAllResourcesIds.5 = STRING: "dummy9"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterAllResourcesIds.6 = STRING: "dummy1"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterAllResourcesIds.7 = STRING: "dummy10"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterAllResourcesIds.8 = STRING: "dummy2"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterAllResourcesIds.9 = STRING: "dummy3"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterAllResourcesIds.10 = STRING: "dummy4"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterRunningResourcesNum.0 = INTEGER: 9
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterRunningResourcesIds.0 = STRING:
"fencing"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterRunningResourcesIds.1 = STRING:
"dummy7"
```

```
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterRunningResourcesIds.2 = STRING:
"dummy8"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterRunningResourcesIds.3 = STRING:
"dummy9"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterRunningResourcesIds.4 = STRING:
"dummy1"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterRunningResourcesIds.5 = STRING:
"dummy10"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterRunningResourcesIds.6 = STRING:
"dummy2"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterRunningResourcesIds.7 = STRING:
"dummy3"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterRunningResourcesIds.8 = STRING:
"dummy4"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterStoppedResroucesNum.0 = INTEGER: 1
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterStoppedResroucesIds.0 = STRING:
"dummy5"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterFailedResourcesNum.0 = INTEGER: 1
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterFailedResourcesIds.0 = STRING: "dummy6"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterFailedResourcesIds.0 = No more variables
left in this MIB View (It is past the end of the MIB tree)
```

9.9. 配置资源以保持在 CLEAN NODE SHUTDOWN 上停止（红帽企业 LINUX 7.8 及更新的版本）

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。从 Red Hat Enterprise Linux 7.8 开始，您可以配置 Pacemaker，以便在节点完全关闭时，附加到该节点的资源将锁定到该节点，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重新启动。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。

9.9.1. 配置资源在 Clean Node Shutdown 上停止的集群属性

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

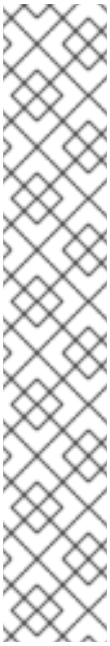
当将此集群属性设置为 false 的默认值时，集群将恢复在被完全关闭的节点上活跃的资源。当此属性设为 true 时，在被完全关闭的节点上活跃的资源将无法在其它位置启动，直到它们在重新加入集群后在该节点上再次启动。

shutdown-lock 属性适用于群集节点或远程节点，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的锁定，以便可通过使用以下命令在节点上手动刷新来在其它位置启动资源。

```
pcs resource refresh resource --node node
```

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，以停止该节点。
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。但请注意，时间间隔不会比 **cluster-recheck-interval** 集群属性的值更频繁地检查。

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，以停止该节点。
2. 运行 **pcs resource disable remote-connection-resource** 命令。

运行这些命令后，当因 **shutdown-lock-limit** 指定的时间已过后，远程节点上运行的资源将可用于在其他节点上恢复。

9.9.2. 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，并显示在关闭并再次启动节点时的影响。这个示例集群由三个节点组成：z 1.example.com、z2.example.com 和 z3.example.com。

1. 将 shutdown-lock 属性设为 true 并验证其值。在本例中，shutdown-lock-limit 属性维护其默认值 0。

```
[root@z3.example.com ~]# pcs property set shutdown-lock=true
[root@z3.example.com ~]# 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 三 和 第五个 在 z1.example.com 上运行。

```
[root@z3.example.com ~]# 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，这将停止该节点上运行的资源。

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

运行 pcs status 命令可显示节点 z1.example.com 脱机，并且 z1.example.com 上运行的资源在节点停机时为 LOCKED。

```
[root@z3.example.com ~]# 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)
...
```

4. **在 z1.example.com 上再次启动群集服务，使其重新加入集群。锁定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.example.com ~]# pcs cluster start z1.example.com
Starting Cluster...
```

在本例中，会在节点 z1.example.com 上恢复第三和第五个节点。

```
[root@z3.example.com ~]# 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 10 章 集群仲裁

红帽企业 Linux 高可用性附加组件群集使用 votequorum 服务和隔离以避免脑裂问题。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。

10.1. 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。表 10.1 "仲裁选项" 总结了这些选项。

表 10.1. 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。群集分区或仍与 **auto_tie_breaker_node** 中配置的 **nodeid** 联系的节点集合（如果未设置则为最低的 **nodeid** ）将保持法定状态。其他节点将为 inquorate。 **auto_tie_breaker** 选项主要用于具有偶数节点的群集，因为它允许群集继续使用平均分割操作。对于更复杂的故障，如多个不一致的分割，建议您使用仲裁设备，如 第 10.5 节 "仲裁设备" 所述。**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群集，以及用于使用仲裁设备 **lms(** last man standing)算法的双向群集。 当群集具有两个节点并且不使用仲裁设备并且禁用 **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动态重新计算 **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)man page。

10.2. 仲裁管理命令（RED HAT ENTERPRISE LINUX 7.3 及稍后）

在集群运行时，您可以输入以下的集群仲裁命令。

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
```

如果您将节点长时间移出集群，且这些节点丢失会导致仲裁丢失，您可以使用 pcs quorum expected-votes 命令更改实时群集的 expected_ votes 参数值。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，且不会更改配置文件；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
```

10.3. 修改仲裁选项（红帽企业 LINUX 7.3 及更新的版本）

从 Red Hat Enterprise Linux 7.3 开始，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，执行此命令要求停止群集。有关仲裁选项的详情，请查看 votequorum(5)man page。

pcs quorum update 命令的格式如下。

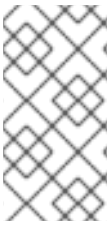
```
pcs quorum update [auto_tie_breaker=[0|1]] [last_man_standing=[0|1]] [last_man_standing_window=
[time-in-ms] [wait_for_all=[0|1]]
```

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
```

10.4. 仲裁 UNBLOCK 命令

在您知道集群不仲裁但您希望集群进行资源管理的情况下，您可以使用以下命令来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 cluster quorum unblock
```

10.5. 仲裁设备

Red Hat Enterprise Linux 7.4 完全支持配置作为集群的第三方设备的独立仲裁设备。它的主要用途是允许集群保持比标准仲裁规则允许更多的节点故障。建议在具有偶数节点的集群中使用仲裁设备。对于双

节点群集，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，单个仲裁设备可由一个具有破坏 (fifty/fifty split)算法的集群和具有 lms (last man standing)算法的第二个群集使用。
- 不应在现有集群节点中运行制裁设备。

10.5.1. 安装仲裁设备软件包

为集群配置仲裁设备需要您安装以下软件包：

- 在现有群集的节点上安装 corosync-qdevice。

```
[root@node1:~]# yum install corosync-qdevice
[root@node2:~]# yum install corosync-qdevice
```

- 在仲裁设备主机上安装 pcs 和 corosync-qnetd。

```
[root@qdevice:~]# yum 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
```

10.5.2. 配置仲裁设备

本节提供了在红帽高可用性集群中配置仲裁设备的示例步骤。以下流程配置了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 ：5-fifty split.这为拥有最多活跃节点的分区提供一个投票。
  - lms：le -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，就是说只有具有所有活跃节点的集群才能保持仲裁（因为仲裁设备的投票权利更大），其它任何群集都每以处于仲裁状态。

  有关实施这些算法的详情，请查看 corosync-qdevice(8)man page。

- 集群节点是 node1 和 node2。

下面步骤配置一个仲裁设备，并将仲裁设备添加到集群中。

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，此时没有连接的客户端。**--full** 命令选项提供详细输出。

```
[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 高可用性 服务，从而在防火墙上启用 **pcsd** 守护进程和网络 仲裁 设备所需的端口：

```
[root@qdevice:~]# firewall-cmd --permanent --add-service=high-availability
[root@qdevice:~]# firewall-cmd --add-service=high-availability
```

3. 从现有集群中的某个节点中，在托管仲裁设备的节点上验证用户 **hacluster**。

```
[root@node1:~] # pcs cluster 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。这些命令的输出表明集群还没有使用仲裁设备。

```
[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)man page**。

```
[root@node1:~]# pcs quorum device add model net host=qdevice algorithm=ffsplit
Setting up qdevice certificates on nodes...
node2: Succeeded
node1: Succeeded
Enabling corosync-qdevice...
node1: corosync-qdevice enabled
node2: corosync-qdevice enabled
Sending updated corosync.conf to nodes...
node1: Succeeded
node2: Succeeded
Corosync configuration reloaded
Starting corosync-qdevice...
node1: corosync-qdevice started
node2: corosync-qdevice started
```

5. 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，这表示仲裁设备正在使用中。

```
[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 设备状态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
```

从仲裁设备一侧，您可以执行以下 **status** 命令，该命令显示 **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)
```

10.5.3. 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**)的功能，如下例所示。请注意，这些命令仅影响 **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
```

10.5.4. 管理集群中的仲裁设备设置

下面的部分描述了可以用来管理集群中的仲裁设备设置的 **PCS** 命令，显示了 第 10.5.2 节 “配置仲裁设备” 中基于仲裁设备配置的示例。

10.5.4.1. 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 的主机 选项 net，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
```

10.5.4.2. 删除仲裁设备

使用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
```

10.5.4.3. 销毁仲裁设备

要禁用和停止仲裁设备主机上的仲裁设备并删除其所有配置文件，请使用以下命令。

```
[root@qdevice:~]# pcs qdevice destroy net
Stopping quorum device...
quorum device stopped
quorum device disabled
Quorum device 'net' configuration files removed
```

第 11 章 PACEMAKER 规则

通过使用规则可以使您的配置更动态。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 字段合并，以确定规则最终评估为 true 或 false。接下来的操作要看规则使用的上下文而定。

表 11.1. 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、Slave 和 Master。注意：带有 role="Master" 的规则无法确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 时要应用的分数。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，则要查找并用作分数的节点属性。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： 和 和 或.默认值为 and.

11.1. 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 11.2. 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：字符串、整数、version。默认值为 **string**
操作	执行的对比。允许的值： * **lt** - 如果节点属性 的值小于值，则为 **True** * **gt** - 如果节点属性 的值大于值，则为 **True** * **LTE** - 如果节点属性的值小于或等于值，则为 **True**

项	描述 * G TE - 如果节点属性的值大于或等于值，则为 True
	* eq - 如果节点属性 的值等于值，则为 True * ne - 如果节点属性的值不等于值，则为 True * 已定义 - 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
value	用户提供用于比较的值（必需）

除了管理员添加的任何属性外，集群还为每个节点定义特殊的内置节点属性，如 表 11.3 “内置节点属性” 所述。

表 11.3. 内置节点属性

名称	描述
#uname	节点名称

名称	描述
#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，以及 Pacemaker 远程客户机节点和捆绑包节点 的容器，kind 的值是 remote。
#is_dc	如果此节点是 Designated Controller(DC)，则为 true，否则 为 false
#cluster_name	cluster-name 集群属性的值（如果设置）
#site_name	site-name node 属性的值（如果设置），否则与 #cluster-name相同
#role	此节点上相关的多状态资源的角色。仅在多状态资源的位置约束的规则内有效。

11.2. 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 11.4. 日期表达式的属性

项	描述
start	符合 ISO8601 规范的日期/时间。
end	符合 ISO8601 规范的日期/时间。

项	描述
操作	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/时间在启动后，则为 True * lt - 如果当前日期/时间在 结束前，则为 True * in-range - 如果当前日期/时间在 开始之后 和 结束前，则为 True * date-spec - 执行与当前日期/时间类似的 cron 比较

11.3. 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，月日="1" 与每月第一天和 小时="09-17" 匹配上午 9 点到下午 5 点（包含）之间的小时数。但是，您无法指定 weekdays="1,2" 或 weekdays="1-2,5-6"，因为它们包含多个范围。

表 11.5. 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）
年日	允许的值： 1-366（根据年而定）
months	允许的值： 1-12
周	允许的值： 1-53（取决于 星期年）

项	描述
年	根据 Gregorian 日历年
周年	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 为新月，4 为满月）。

11.4. 持续时间

持续时间用于计算当一个值未提供给 in_range 操作时 的末尾 值。它们包含与 date_spec 对象相同的字段，但没有限制（例如，您可以持续 19 个月）。与 date_specs 一样，任何未提供的字段都会被忽略。

11.5. 使用 PCS 配置规则

要使用 pcs 配置规则，您可以配置使用规则的位置约束，如 第 7.1.3 节 “使用规则确定资源位置” 所述。

要删除规则，可使用以下内容：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
```

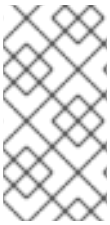
第 12 章 PACEMAKER 集群属性

集群属性用于控制，当遇到在操作时可能会发生的情况时，集群会如何处理。

- 表 12.1 “集群属性” 描述集群属性选项。
- 第 12.2 节 “设置和删除集群属性” 描述如何设置集群属性。
- 第 12.3 节 “查询集群属性设置” 描述如何列出当前设置的集群属性。

12.1. 集群属性和选项概述

表 12.1 “集群属性” 总结 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。



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 12.1. 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。
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
symmetric-cluster	true	指明资源是否可以默认在任何节点上运行。

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 unset，除非同时配置了一个或多个 STONITH 资源，否则集群将拒绝启动资源。
stonith-action	reboot	发送到 STONITH 设备的操作。允许的值：**reboot**、**off.**也允许使用 value **poweroff**，但只适用于旧的设备。
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。
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，集群将根据资源当前的故障数和迁移阈值决定是否在同一节点中再次启动。有关为资源设置 **migration-threshold** 选项的详情请参考第 8.2 节 "因为失败而移动资源"。 将 **start-failure-is-fatal** 设置为 **false** 的风险会导致一个无法启动资源的节点无法执行所有依赖的操作的风险。这就是 **start-failure-is-fatal** 默认为 **true** 的原因。可以通过设置低迁移阈值来降低设置 **start-failure-is-fatal=false** 的风险，以便其他操作可在很多失败后继续。
pe-error-series-max	-1（全部）	PE 输入数导致要保存的 ERRORs。报告问题时使用。
pe-warn-series-max	-1（全部）	PE 输入数导致 WARNINGs 要保存。报告问题时使用。
pe-input-series-max	-1（全部）	要保存的 "normal" PE 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。
last-lrm-refresh		最后一次刷新本地资源管理器，自 epoca 起以秒为单位。用于诊断目的，用户不能配置。
cluster-recheck-interval	15 分钟	对选项、资源参数和限制进行基于时间的更改轮询间隔。允许的值：零代表禁用轮询，正数值代表以秒为单位的间隔（除非指定了其它单位，如 5min）。请注意，这个值是不同检查之间的最长时间 ；如果集群事件发生的时间早于这个值指定的时间，则会更早地进行检查。

选项	默认值	描述
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。
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。
stonith-timeout	60s	等待 STONITH 操作完成的时间。
stop-all-resources	false	集群是否应该停止所有资源。
enable-acl	false	（红帽企业 Linux 7.1 及更高版本）指明群集是否可以使用访问控制列表，如 **pcs acl** 命令所设置。
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。有关使用属性和放置策略的详情请参考 第 9.6 节 “使用和放置策略”。
fence-reaction	**stop**	（Red Hat Enterprise Linux 7.8 及更新的版本）决定在收到其自身隔离通知时集群节点应如何做出反应。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。允许的值会 **停止**，它会停止 Pacemaker 并保持停止状态，或者 **panic** 来尝试立即重启本地节点，并在失败后退回到停止状态。

12.2. 设置和删除集群属性

*要设置集群属性的值，请使用以下 **pcs** 命令。*

```
pcs property set property=value
```

*例如，若要将 **symmetric-cluster** 的值设置为 **false**，可使用以下命令：*

```
# 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=
```

12.3. 查询集群属性设置

在大多数情况下，当使用 pcs 命令显示各种群集组件的值时，您可以互换使用 pcs list 或 pcs show。在以下示例中，pcs list 的格式用于显示多个属性的所有设置的完整列表，而 pcs show 是用于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，请使用以下 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
```

第 13 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采取一些外部操作。您可以通过以下两种方式之一配置集群警报：

- 从 Red Hat Enterprise Linux 7.3 开始，您可以使用警报代理来配置 Pacemaker 警报，它们是集群调用的外部程序，其方式与集群调用的资源代理来处理资源配置和操作相同。这是配置群集警报的首选、更简单的方法。Pacemaker 警报代理在 第 13.1 节 “Pacemaker 警报代理（红帽企业 Linux 7.3 及更新的版本）” 中描述。

- ocf:pacemaker:ClusterMon 资源可以监控集群状态，并触发每个集群事件的警报。此资源在后台以固定间隔运行 crm_mon 命令。有关 ClusterMon 资源的详情请参考 第 13.2 节 “使用监控资源的事件通知”。

13.1. PACEMAKER 警报代理（红帽企业 LINUX 7.3 及更新的版本）

您可以创建 Pacemaker 警报代理，以便在集群事件发生时采取一些外部操作。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 提供几个示例警报代理，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。有关配置使用示例警报代理的警报的基本步骤示例，请参阅 第 13.1.1 节 “使用示例警报代理”。

- 第 13.1.2 节 “创建警报”、第 13.1.3 节 “显示、修改和删除警报”、第 13.1.4 节 “警报 Recipients”、第 13.1.5 节 “警报元数据选项” 和 第 13.1.6 节 “警报配置命令示例” 中提供了有关配置和管理警报代理的一般信息。

- 您可以为 Pacemaker 警报编写自己的警报代理来调用。有关编写警报代理的详情请参考 第 13.1.7 节 “编写警报代理”。

13.1.1. 使用示例警报代理

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，红帽支持警报代理脚本用来与 Pacemaker 通信的界面，但红帽并不支持自定义代理本身。

要使用示例警报代理中的一个，您必须在集群中的每个节点上安装代理。例如，以下命令将 alert_file.sh.sample 脚本安装为 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警报。

以下示例配置了使用安装的 alert_file.sh 警报代理将事件记录到文件中的警报。以用户 hacluster 身份运行的警报代理，该用户具有最小权限集。

这个示例创建日志文件 pcmk_alert_file.log，该文件将用于记录事件。然后，它会创建警报代理，并添加到日志文件的路径作为其接收者。

```
# 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
```

以下示例将 alert_snmp.sh.sample 脚本安装为 alert_snmp.sh，并配置使用安装的 alert_snmp.sh 警报代理将集群事件作为 SNMP 陷阱发送的警报。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。有关 meta 选项的详情请参考 第 13.1.5 节 “警报元数据选项”。配置警报后，本例配置警报的接收者并显示警报配置。

```
# 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 alert_smtp.sh 代理，然后配置使用安装的警报代理将集群事件作为电子邮件消息发送的警报。配置警报后，本示例配置了接收方并显示警报配置。

```
# 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)
```

有关 **pcs alert create** 和 **pcs alert receiver add** 命令格式的更多信息，请参阅 第 13.1.2 节 “创建警报” 和 第 13.1.4 节 “警报 Recipients”。

13.1.2. 创建警报

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文件，这些值会被传递给您指定为额外环境变量的路径的警报代理脚本。如果没有为 **id** 指定值，则会生成一个值。如需关于警报 **meta** 选项的信息，请参阅 第 13.1.5 节 “警报元数据选项”。

```
pcs alert create path=path [id=alert-id] [description=description] [options [option=value]...] [meta
[meta-option=value]...]
```

可能会配置多个警报代理，集群会在每个事件中调用它们。只有集群节点上才会调用警报代理。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警报，它将为每个事件调用 **myscript.sh**。

```
# pcs alert create id=my_alert path=/path/to/myscript.sh
```

有关如何创建使用其中一个示例警报代理的集群警报的示例，请参考 第 13.1.1 节 “使用示例警报代理”。

13.1.3. 显示、修改和删除警报

以下命令显示所有配置的警报以及配置选项的值。

```
pcs alert [config|show]
```

以下命令使用指定的 **alert-id** 值更新现有警报。

```
pcs alert update alert-id [path=path] [description=description] [options [option=value]...] [meta [meta-
option=value]...]
```

- 以下命令移除具有指定 alert-id 值的警报。

```
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警报。

13.1.4. 警报 Recipients

通常，警报是针对接收方的。因此，每个警报可能被额外配置为一个或多个接收方。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。

以下命令为指定警报添加新的接收者。

```
pcs alert recipient add alert-id value=recipient-value [id=recipient-id] [description=description] [options
[option=value]...] [meta [meta-option=value]...]
```

以下命令更新现有警报接收者。

```
pcs alert recipient update recipient-id [value=recipient-value] [description=description] [options
[option=value]...] [meta [meta-option=value]...]
```

以下命令移除指定警报接收者。

```
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警报接收者。

以下示例命令将警报接收者 my-alert-reci pipient -id 添加到警报 my-alert 中。这会将群集配置为调用为每个事件配置了 my-alert 的警报脚本，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
```

13.1.5. 警报元数据选项

与资源代理一样，可以对警报代理配置 meta 选项来影响 Pacemaker 调用它们的方式。表 13.1 “警报元数据选项” 描述警报 meta 选项。meta 选项可以为每个警报代理和接收者配置。

表 13.1. 警报元数据选项

meta-Attribute	默认值	描述
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。
timeout	30s	如果警报代理没有在这段时间内完成，它将被终止。

以下示例配置了调用脚本 myscript.sh 的警报，然后为警报添加两个接收者。第一个接收者 ID 为 my-alert-recipient1，第二个收件人的 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，格式为 %D %H:%M，另一个调用将被传递给接收者 otheruser@example.com，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
```

13.1.6. 警报配置命令示例

以下后续示例演示了一些基本警报配置命令，以显示用于创建警报、添加接收方和显示配置的警报的格式。请注意，虽然您必须在集群中的每个节点上安装警报代理，但您需要只运行一次 'pcs' 命令。

以下命令创建了一个简单的警报，为警报添加两个接受者，并显示配置的值。

- 由于没有指定警报 ID 值，系统会创建警报的警报 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，alert-recipient 的值被用作接收者 ID。
-

第二个接收者创建命令指定 **rec_value2** 的接收者。此命令 为接收者指定 **my- repient** 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警报以及该警报的接收者。第二个警报的警报 **ID** 是 **my-alert**，接收者值为 **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)
```

以下命令修改警报 **my-alert** 和接收者 **my-alert -recipient** 的警报值。

```
# 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 警报 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
  Meta options: timestamp-format="%M%B%S" timeout=50s
  Meta options: m=newval meta-option1=2
  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 my-alert
# pcs alert
Alerts:
 Alert: alert (path=/my/path)
  Recipients:
   Recipient: alert-recipient (value=rec_value)
```

13.1.7. 编写警报代理

***Pacemaker** 警报有三种类型：节点警报、保护警报和资源警报。传递给警报代理的环境变量可能会根据警报类型而有所不同。表 13.2 “传递给警报代理的环境变量” 描述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 13.2. 传递给警报代理的环境变量

环境变量	描述
CRM_alert_kind	警报类型（节点、保护或资源）
CRM_alert_version	Pacemaker 发送警报的版本
CRM_alert_recipient	配置的接收者
CRM_alert_node_sequence	每当在本地节点上发出警报时，序列数量会增加，它可以用来引用 Pacemaker 发出警报的顺序。稍后发生事件警告的序列号比之前的事件的警报要高。请注意，这个数字没有集群范围的含义。

环境变量	描述
CRM_alert_timestamp	执行代理前创建的时间戳，采用由 **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。对于节点警报，这是节点的当前状态（成员或丢失）。对于隔离警报，这是请求的隔离操作的总结，其中包括原始数据、目标以及隔离操作错误代码（若有）。对于资源警报，这是等同于 **CRM_alert_status** 的可读字符串。
CRM_alert_nodeid	状态更改的节点 ID（仅由节点警报提供）
CRM_alert_task	请求的隔离或资源操作（仅由隔离和资源警报提供）
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	受影响资源的名称（仅限资源警报）
CRM_alert_interval	资源操作的时间间隔（仅限资源警报）
CRM_alert_target_rc	操作的预期数字返回代码（仅用于资源警报）
CRM_alert_status	Pacemaker 用来表示操作结果的数字代码（仅用于资源警报）

在编写警报代理时，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。*

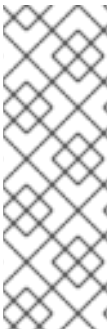
- *如果为警报配置了多个接收者，则会为每个接收者调用一个警报代理。如果代理无法同时运行，则应该只使用单个的接收者进行配置。不过，代理可以自由地将接收者解析为一个列表。*

- *当发生集群事件时，所有警报都会与独立进程同时触发。根据配置了警报和接收方的数量以及警报代理中的操作，可能会发生大量负载。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。*

- *警报代理以 **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 （由用户配置的 timestamp-format）、CRM_alert_recipient 和所有警报选项指定的内容。这是防止配置错误所必需的。此外，如果某些用户可以在没有 hacluster-level 访问集群节点的情况下修改 CIB，则也是潜在的安全问题，您应该避免注入代码的可能性。

- 如果群集包含将 on-fail 参数设置为 隔离 的操作的资源，则失败时会有多个隔离通知，每个资源都有一个用于设置此参数的资源，再加上一个附加通知。STONITH 守护进程和 crmd 守护进程都将发送通知。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

警报接口设计为与 ocf:pacemaker:ClusterMon 资源使用 的外部脚本界面向后兼容。为了保持这种兼容性，传递给警报代理的环境变量会预先带有 CRM_notify_ 和 CRM_alert_。兼容性问题之一是 ClusterMon 资源以 root 用户身份运行外部脚本，而警报代理则以 hacluster 用户身份运行。有关配置由 ClusterMon 触发的脚本的详情请参考 第 13.2 节 “使用监控资源的事件通知”。

13.2. 使用监控资源的事件通知

ocf:pacemaker:ClusterMon 资源可以监控集群状态，并触发每个集群事件的警报。此资源在后台以固定间隔运行 crm_mon 命令。

默认情况下，cr m_mon 命令仅侦听资源事件；若要启用隔离事件列表，您可以在配置 ClusterMon 资源时为 命令提供 --watch-fencing 选项。crm_mon 命令不会监控成员资格问题，而是在启动隔离以及为该节点启动监控时打印一条消息，这意味着成员刚加入群集。

ClusterMon 资源可以执行外部程序，以确定如何使用 extra_options 参数来使用集群通知。表 13.3 “传递给外部监控程序的环境变量” 列出传递给该程序的环境变量，以描述发生的集群事件类型。

表 13.3. 传递给外部监控程序的环境变量

环境变量	描述
CRM_notify_recipient	资源定义的静态外部通知
CRM_notify_node	发生状态更改的节点
CRM_notify_rsc	更改状态的资源名称
CRM_notify_task	导致状态更改的操作

环境变量	描述
CRM_notify_desc	导致状态更改的操作的文本输出相关错误代码（如果有）
CRM_notify_rc	操作的返回代码
CRM_target_rc	操作的预期返回代码
CRM_notify_status	操作状态的数字表示

以下示例配置了一个 ClusterMon 资源，用于执行外部程序 crm_logger.sh，它将记录程序中指定的事件通知。

以下流程创建此资源 要使用的 crm_logger.sh 程序。

1. *在集群的一个节点上，创建将记录事件通知的程序。*

```
# cat <<-END >/usr/local/bin/crm_logger.sh
#!/bin/sh
logger -t "ClusterMon-External" "${CRM_notify_node} ${CRM_notify_rsc} \
${CRM_notify_task} ${CRM_notify_desc} ${CRM_notify_rc} \
${CRM_notify_target_rc} ${CRM_notify_status} ${CRM_notify_recipient}";
exit;
END
```

2. *设置程序的所有权和权限。*

```
# chmod 700 /usr/local/bin/crm_logger.sh
# chown root.root /usr/local/bin/crm_logger.sh
```

3. *使用 scp 命令将 crm_logger.sh 程序复制到集群的其他节点上，将程序放置在同一位置上，并为程序设置相同的所有权和权限。*

以下示例配置名为 ClusterMon -External 的 ClusterMon 资源，该资源运行程序 /usr/local/bin/crm_logger.sh。ClusterMon 资源将集群状态输出到 a html 文件，在这个示例中是 /var/www/html/cluster_mon.html。The pidfile 检测 ClusterMon 是否已 在运行；在本示例中，该文件为 /var/run/crm_mon-external.pid。此资源作为克隆创建，以便其在群集中的每个节点上运行。指定了 watch-fencing，除了资源事件（包括 start/stop/monitor、start/monitor）和停止隔离资源外，还启用对隔离事件的监控。

```
# pcs resource create ClusterMon-External ClusterMon user=root \
update=10 extra_options="-E /usr/local/bin/crm_logger.sh --watch-fencing" \
htmlfile=/var/www/html/cluster_mon.html \
pidfile=/var/run/crm_mon-external.pid clone
```

注意

以下是此资源执行以及可以手动运行的 crm_mon 命令：

```
# /usr/sbin/crm_mon -p /var/run/crm_mon-manual.pid -d -i 5 \
-h /var/www/html/crm_mon-manual.html -E "/usr/local/bin/crm_logger.sh" \
--watch-fencing
```

以下示例显示了本示例生成的监控通知输出格式。

```
Aug  7 11:31:32 rh6node1pcmk ClusterMon-External: rh6node2pcmk.examplerh.com ClusterIP
st_notify_fence Operation st_notify_fence requested by rh6node1pcmk.examplerh.com for peer
rh6node2pcmk.examplerh.com: OK (ref=b206b618-e532-42a5-92eb-44d363ac848e) 0 0 0 #177
Aug  7 11:31:32 rh6node1pcmk ClusterMon-External: rh6node1pcmk.examplerh.com ClusterIP start
OK 0 0 0
Aug  7 11:31:32 rh6node1pcmk ClusterMon-External: rh6node1pcmk.examplerh.com ClusterIP
monitor OK 0 0 0
Aug  7 11:33:59 rh6node1pcmk ClusterMon-External: rh6node1pcmk.examplerh.com fence_xvms
monitor OK 0 0 0
Aug  7 11:33:59 rh6node1pcmk ClusterMon-External: rh6node1pcmk.examplerh.com ClusterIP
monitor OK 0 0 0
Aug  7 11:33:59 rh6node1pcmk ClusterMon-External: rh6node1pcmk.examplerh.com ClusterMon-
External start OK 0 0 0
Aug  7 11:33:59 rh6node1pcmk ClusterMon-External: rh6node1pcmk.examplerh.com fence_xvms
start OK 0 0 0
Aug  7 11:33:59 rh6node1pcmk ClusterMon-External: rh6node1pcmk.examplerh.com ClusterIP start
OK 0 0 0
Aug  7 11:33:59 rh6node1pcmk ClusterMon-External: rh6node1pcmk.examplerh.com ClusterMon-
External monitor OK 0 0 0
Aug  7 11:34:00 rh6node1pcmk crmd[2887]:   notice: te_rsc_command: Initiating action 8: monitor
ClusterMon-External:1_monitor_0 on rh6node2pcmk.examplerh.com
Aug  7 11:34:00 rh6node1pcmk crmd[2887]:   notice: te_rsc_command: Initiating action 16: start
ClusterMon-External:1_start_0 on rh6node2pcmk.examplerh.com
Aug  7 11:34:00 rh6node1pcmk ClusterMon-External: rh6node1pcmk.examplerh.com ClusterIP stop
OK 0 0 0
Aug  7 11:34:00 rh6node1pcmk crmd[2887]:   notice: te_rsc_command: Initiating action 15: monitor
ClusterMon-External_monitor_10000 on rh6node2pcmk.examplerh.com
Aug  7 11:34:00 rh6node1pcmk ClusterMon-External: rh6node2pcmk.examplerh.com ClusterMon-
External start OK 0 0 0
Aug  7 11:34:00 rh6node1pcmk ClusterMon-External: rh6node2pcmk.examplerh.com ClusterMon-
External monitor OK 0 0 0
Aug  7 11:34:00 rh6node1pcmk ClusterMon-External: rh6node2pcmk.examplerh.com ClusterIP start
OK 0 0 0
Aug  7 11:34:00 rh6node1pcmk ClusterMon-External: rh6node2pcmk.examplerh.com ClusterIP
monitor OK 0 0 0
```

-

第 14 章 使用 PACEMAKER 配置多站点集群

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。

为了解决这些问题，Red Hat Enterprise Linux release 7.4 提供了全面支持，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。Booth 票据管理器是一个分布式服务，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，一个 Booth 组成，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 Boothship 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。

以下流程概述了配置使用 Booth ticket 管理器的多站点配置的步骤。

这些示例命令使用以下协议：

- 集群 1 由 node 1 和 cluster1-node2 节点组成
- 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
- 仲裁节点是具有 IP 地址 192.168.99.100 的仲裁节点

- 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群集的资源组 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。

有关在集群中配置 Apache 服务的完整集群配置步骤，请参阅高可用性附加组件管理示例。

请注意，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，或使用 pcs booth status 命令在本地节点上显示 booth 的当前状态。

1. 在两个集群的每个节点上安装 booth-site Booth ticket manager 软件包。

```
[root@cluster1-node1 ~]# yum install -y booth-site
[root@cluster1-node2 ~]# yum install -y booth-site
[root@cluster2-node1 ~]# yum install -y booth-site
[root@cluster2-node2 ~]# yum 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 ~]# yum install -y pcs booth-core booth-arbitrator
```

3. 确保在所有群集节点和仲裁节点上打开端口 9929/tcp 和 9929/udp。

   例如，在两个集群的所有节点上和临时节点上运行以下命令，允许访问这些节点上的端口 9929/tcp 和 9929/udp。

```
# firewall-cmd --add-port=9929/udp
# firewall-cmd --add-port=9929/tcp
# firewall-cmd --add-port=9929/udp --permanent
# firewall-cmd --add-port=9929/tcp --permanent
```

   请注意，这个过程本身允许任何机器访问节点上的端口 9929。您应该确保主机上仅对需要节点的节点开放。

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，您必须首先将 pcs 身份验证到您要拉取配置的节点。

```
[arbitrator-node ~] # pcs cluster 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，您必须首先向要拉取配置的节点验证 pcs。

```
[cluster2-node1 ~] # pcs cluster 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. 将 Booth 配置为作为集群资源在这两个集群站点运行。这将创建一个资源组，并将 booth-ip 和 booth-service 用作该组的成员。

```
[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。最初为集群授予一个 ticket 后，booth 会接管票据管理，除非您使用 pcs booth ticket revoke 命令手动覆盖此票据。有关 pcs booth 管理命令的详情请参考 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。

有关您可用于清理和删除 Booth 配置文件、票据和资源的其他 Booth 管理命令的详情，请查看 pcs booth 命令的 PCS 帮助屏幕。

附录 A. OCF 返回代码

本附录描述了 OCF 返回代码，以及如何由 Pacemaker 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代码。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代码退出，该代码告知调用者调用的操作的结果。

如 表 A.1 “集群恢复执行的类型” 所述，有三种类型的故障恢复。

表 A.1. 集群恢复执行的类型

类型	描述	集群抓取的操作
soft	发生瞬态错误.	重新启动资源 或将其移到新位置。
难	发生非临时错误，可能特定于当前节点。	将资源移到其他位置，并阻止其在当前节点上重试。
fatal	发生非临时错误，适用于所有集群节点（例如，指定了一个错误的配置）。	停止资源，并阻止其在任何群集节点上启动。

表 A.2 “OCF 返回代码” 提供 OCF 返回代码，以及群集在收到失败代码时将启动的恢复类型。请注意，如果 0 不是预期返回值，即使返回 0（OCF 别名 OCF 别名 OCF_SUCCESS）的操作也被视为失败。

表 A.2. OCF 返回代码

返回代码	OCF Label	描述
0	OCF_SUCCESS	该操作成功完成。这是任何成功启动、停止、提升和降级命令的预期返回代码。 如果意外： soft 则键入
1	OCF_ERR_GENERIC	该操作返回一个通用错误。 类型：软 资源管理器将尝试恢复资源或将其移动到新位置。
2	OCF_ERR_ARGS	资源的配置在此计算机上无效。例如，它引用节点上未找到的位置。 类型： hard 资源管理器将在其他位置移动资源，并阻止其在当前节点上重试
3	OCF_ERR_UNIMPLEMENTED	请求的操作未实施。 类型： hard

返回代码	OCF Label	描述
4	OCF_ERR_PERM	资源代理没有足够的特权来完成该任务。这可能是因为代理无法打开特定文件、侦听特定套接字或写入目录。 类型： hard 除非另有特殊配置，否则资源管理器将通过在其他节点上重启资源来尝试恢复出错的资源（其中权限问题可能不存在）。
5	OCF_ERR_INSTALLED	执行该操作的节点上缺少所需的组件。这可能是因为所需的二进制文件不可执行，或者重要配置文件不可读取。 类型： hard 除非另有特殊配置，否则资源管理器将尝试通过在其他节点上重启资源（可能存在所需的文件或二进制文件）来恢复发生此错误的资源。
6	OCF_ERR_CONFIGURED	本地节点上的资源配置无效。 类型：fatal 当返回此代码时，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有效。

返回代码	OCF Label	描述
7	OCF_NOT_RUNNING	资源已被安全停止。这意味着资源已正常关闭，或者从未启动。 如果意外： soft 则键入 对于任何操作，集群不会尝试停止返回此值的资源。
8	OCF_RUNNING_MASTER	资源在 master 模式下运行。 如果意外： soft 则键入
9	OCF_FAILED_MASTER	资源处于 master 模式，但失败。 类型：软 资源将被降级、停止，然后再次启动（可能升级）。
其他	不适用	自定义错误代码.

附录 B. 在 RED HAT ENTERPRISE LINUX 6 和 RED HAT ENTERPRISE LINUX 7 中创建集群

使用 Pacemaker 在 Red Hat Enterprise Linux 7 中配置红帽高可用性集群需要一组不同的配置工具，其管理界面与在 Red Hat Enterprise Linux 6 中使用 rgmanager 配置集群不同。第 B.1 节 "使用 rgmanager 和 Pacemaker 创建集群" 总结了不同集群组件的配置差异。

Red Hat Enterprise Linux 6.5 及更新的版本使用 pcs 配置工具支持使用 Pacemaker 的群集配置。第 B.2 节 "Red Hat Enterprise Linux 6 和 Red Hat Enterprise Linux 7 中的 Pacemaker 安装" 总结了 Red Hat Enterprise Linux 6 和 Red Hat Enterprise Linux 7 之间的 Pacemaker 安装差异。

B.1. 使用 RGMANAGER 和 PACEMAKER 创建集群

表 B.1 "集群配置与 rgmanager 和 Pacemaker 的比较" 提供了有关如何在 Red Hat Enterprise Linux 6 和 Red Hat Enterprise Linux 7 中使用 Pacemaker 配置带有 rgmanager 的集群组件的比较概述。

表 B.1. 集群配置与 rgmanager 和 Pacemaker 的比较

配置组件	rgmanager	pacemaker
集群配置文件	每个节点上的集群配置文件是 **cluster.conf** 文件，可以直接编辑该文件。否则，使用 luci **orccs** 接口来定义集群配置。	群集和 Pacemaker 配置文件为 **corosync.conf** 和 **cib.xml**。不要直接编辑 **cib.xml** 文件；改为使用 **pcs** 或 **pcsd** 接口。
网络设置	在配置集群前配置 IP 地址和 SSH。	在配置集群前配置 IP 地址和 SSH。
集群配置工具	Luci, ccs **命令，手动编辑 cluster.conf 文件.**	pcs 或 pcsd.
安装	Install **rgmanager** （拉取所有依赖项，包括 **ricci**、**luci** 以及资源和隔离代理）。如果需要，请安装 **lvm2-cluster** 和 **gfs2-utils**。	安装 **pcs** 以及您需要的隔离代理。如果需要，请安装 **lvm2-cluster** 和 **gfs2-utils**。
启动集群服务	使用以下流程启动并启用集群服务： 1. Start **rgmanager**、**cman** 和（如果需要）c **lvmd** 和 **gfs2**。 2. Start **ricci**，如果使用 **luci** 接口，则启动 **luci**。 3. **为所需服务运行 Runchkconfig**，以便在每个运行时启动。 **另外，您可以输入ccs --start** 以启动并启用集群服务。	使用以下流程启动并启用集群服务： 1. 在每个节点上，执行 **systemctl start pcsd.service**，然后 **systemctl enable pcsd.service** 以启用 **pcsd** 在运行时启动。 2. 在群集的一个节点上，输入 **pcs cluster start --all** 以启动 **corosync** 和 **pacemaker**。

配置组件	rgmanager	pacemaker
控制对配置工具的访问	对于 **luci**，root 用户或具有 **luci 权限的用户可以访问 luci**。所有访问都需要节点的 **ricci** 密码。	**pcsd** gui 要求您以用户 **hacluster**（即通用系统用户）进行身份验证。root 用户可以设置 **hacluster** 的密码。
创建集群	将集群命名为，并使用 **luci orccs** 定义集群中要包含哪些节点，或者直接编辑 **cluster.conf** 文件。	使用 **pcs cluster setup 命令或使用 pcs d** Web UI 将群集命名为并包含节点。您可以使用 **pcs cluster node add 命令或 pcs d** Web UI 将节点添加到现有群集中。
将集群配置传播到所有节点	使用 **luci** 配置集群时，会自动传播。**Withccs**，使用 **--sync** 选项。您还可以使用 **cman_tool version -r** 命令。	集群和 Pacemaker 配置文件 **corosync.conf** 和 **cib.xml** 的传播会在群集设置或添加节点或资源时自动传播。
全局集群属性	Red Hat Enterprise Linux **6** 中支持以下功能： * 您可以配置系统，以便系统选择哪个多播地址用于集群网络中的 IP 多播。 * 如果 IP 多播不可用，您可以使用 UDP 单播传输机制。 * 您可以将集群配置为使用 RRP 协议。	Red Hat Enterprise Linux 7 中的 Pacemaker 支持集群的以下功能： * 您可以为集群设置 **no-quorum-policy**，以指定当集群没有仲裁时系统应执行的操作。 * 有关您可以设置的其他集群属性，请参阅 表 12.1 “集群属性”。
日志	您可以设置全局和特定于守护进程的日志配置。	有关如何手动配置日志记录的详情，请查看文件 **/etc/sysconfig/pacemaker**。
验证集群	集群验证通过 **luci** 和 **withccs** 自动使用集群架构。集群在启动时自动验证。	集群在启动时自动验证，或者您可以使用 **pcs cluster verify 验证群集**。
双节点集群中的仲裁	对于双节点集群，您可以配置系统如何决定仲裁： * 配置仲裁磁盘 * Use **ccs** 或编辑 **cluster.conf** 文件以设置 **two_node=1** 和 **expected_votes=1**，以允许单个节点维护仲裁。	**pcs** 将自动为双节点群集添加必要的选项到 **corosync**。
集群状态	在 **luci** 上，集群的当前状态在界面的不同组件中可见，这些组件可以刷新。您可以使用 **theccs** 命令的 **--getconf** 选项来查看当前的配置文件。您可以使用 **clustat** 命令显示集群状态。	您可以使用 **pcs status** 命令显示当前集群状态。
资源	您可以使用 **luci** 或 **ccs** 命令添加定义类型的资源并配置特定于资源的属性，或者编辑 **cluster.conf** 配置文件。	您可以使用 **pcs resource create 命令或使用 pcs d** Web UI 添加已定义类型的资源并配置特定于资源的属性。有关使用 Pacemaker 配置集群资源的常规信息，请参阅 第 6 章 *配置集群资源*。

配置组件	rgmanager	pacemaker
资源行为、分组和启动/停止顺序	*定义群集服务*，以配置资源交互方式。	使用 Pacemaker 时，您可以使用资源组作为定义一组资源的简写方法，这些资源需要放在一起并按顺序启动和停止。另外，您可以定义资源的行为方式，并通过以下方式进行交互： * 您可以将资源行为的一些方面设置为资源选项。 * 您可以使用位置限制来确定资源可在哪些节点上运行。 * 您可以使用顺序限制来确定资源运行的顺序。 * 您可以使用 colocation 约束来确定一个资源的位置取决于另一个资源的位置。 有关这些主题的详情请参考 第 6 章 *配置集群资源* 和 第 7 章 *资源约束*。
资源管理：移动、启动和停止资源	使用 luci，您可以管理集群、独立集群节点和集群服务。**使用ccs** 命令，您可以管理集群。您可以使用 **clusvadm** 管理集群服务。	您可以临时禁用节点，使其无法使用 **pcs cluster standby** 命令托管资源，这会导致资源迁移。您可以使用 **pcs resource disable** 命令停止资源。
完全删除集群配置	使用 luci，您可以选择集群中的所有节点进行删除，从而彻底删除集群。您还可以从集群中的每个节点中删除 **cluster.conf**。	您可以使用 **pcs cluster destroy** 命令删除集群配置。
在多个节点上活跃的资源，多个节点上活跃的资源	无等效.	通过 Pacemaker，您可以克隆资源以便在多个节点中运行，并将克隆的资源定义为 master 和 slave 资源，以便它们可以在多个模式下运行。有关克隆资源和 master/slave 资源的详情请参考 第 9 章 *高级配置*。
隔离 -- 每个节点一个隔离设备	全局或本地创建隔离设备，并将它们添加到节点。您可以为整个集群定义故障 **后 延迟 和加入后延迟** 值。	使用 pcs **stonith create 命令或使用 pcs d** Web UI 为每个节点创建隔离设备。对于可以隔离多个节点的设备，您需要为每个节点只定义一次，而不是单独定义它们。您还可以定义 **pcmk_host_map** 以使用单个命令为所有节点配置隔离设备；有关 **pcmk_host_map** 的信息，请参阅 表 5.1 “隔离设备的常规属性”。您可以为整个集群定义 **stonith-timeout** 值。
每个节点多个(backup)隔离设备	使用 luci **或 ccs 命令或通过直接编辑 cluster.conf 文件来定义备份设备。**	配置隔离级别.

B.2. RED HAT ENTERPRISE LINUX 6 和 RED HAT ENTERPRISE LINUX 7 中的 PACEMAKER 安装

Red Hat Enterprise Linux 6.5 及更新的版本使用 pcs 配置工具支持使用 Pacemaker 的群集配置。但是，在使用 Pacemaker 时，Red Hat Enterprise Linux 6 和 Red Hat Enterprise Linux 7 的集群安装存在一些差异。

以下命令安装 Pacemaker 在红帽企业 Linux 6 中所需的红帽高可用性附加组件软件包，并防止 corosync 在不使用 cman 的情况下启动。您必须在集群的每个节点中输入这些命令。

```
[root@rhel6]# yum install pacemaker cman pcs
[root@rhel6]# chkconfig corosync off
[root@rhel6]# chkconfig cman off
```

在集群的每个节点中，您将为名为 hacluster 的 pcs 管理帐户设置密码，并且您启动并启用 pcsd 服务。

```
[root@rhel6]# passwd hacluster
[root@rhel6]# service pcsd start
[root@rhel6]# chkconfig pcsd on
```

然后，在集群中的一个节点上验证集群节点的管理帐户。

```
[root@rhel6]# pcs cluster auth [node] [...] [-u username] [-p password]
```

在 Red Hat Enterprise Linux 7 中，您可以在集群的每个节点中运行以下命令来安装 Pacemaker 所需的红帽高可用性附加组件软件包，为名为 hacluster 的 pcs 管理帐户设置密码，并启动并启用 pcsd 服务，

```
[root@rhel7]# yum install pcs pacemaker fence-agents-all
[root@rhel7]# passwd hacluster
[root@rhel7]# systemctl start pcsd.service
[root@rhel7]# systemctl enable pcsd.service
```

在 Red Hat Enterprise Linux 7 中，与在 Red Hat Enterprise Linux 6 中一样，您可以通过在集群的一个节点上运行以下命令来验证集群节点的管理帐户。

```
[root@rhel7]# pcs cluster auth [node] [...] [-u username] [-p password]
```

有关在 Red Hat Enterprise Linux 7 中安装的详情请参考 第 1 章 红帽高可用性附加组件配置和管理参考概述 和 第 4 章 集群创建和管理。

附录 C. 修订历史记录

修订 8.1-1 发布 7.8 Beta 的文档版本.	Fri Feb 28 2020	Steven Levine
修订 7.1-1 发布 7.7 GA 的文档版本.	Wed Aug 7 2019	Steven Levine
修订 6.1-1 发布 7.6 GA 的文档版本.	Thu Oct 4 2018	Steven Levine
修订 5.1-2 发布 7.5 GA 的文档版本.	Thu Mar 15 2018	Steven Levine
修订 5.1-0 7.5 Beta 版出版物文档版本.	Thu Dec 14 2017	Steven Levine
修订 4.1-9 7.4 的更新版本.	Tue Oct 17 2017	Steven Levine
修订 4.1-5 发布 7.4 GA 的文件版本.	Wed Jul 19 2017	Steven Levine
修订 4.1-2 为 7.4 Beta 版出版物准备文档.	Wed May 10 2017	Steven Levine
修订 3.1-10 更新至 7.3 GA 发布的版本。	Tue May 2 2017	Steven Levine
修订 3.1-4 7.3 GA 发布版本.	Mon Oct 17 2016	Steven Levine
修订 3.1-3 为 7.3 Beta 发布准备文档.	Wed Aug 17 2016	Steven Levine
修订 2.1-8 为 7.2 GA 发布准备文档	Mon Nov 9 2015	Steven Levine
修订 2.1-5 为 7.2 Beta 版出版物准备文档.	Mon Aug 24 2015	Steven Levine
修订 1.1-9 7.1 GA 版本	Mon Feb 23 2015	Steven Levine
修订 1.1-7 7.1 Beta 发行版本的版本	Thu Dec 11 2014	Steven Levine
修订 0.1-41 7.0 GA 发行版本的版本	Mon Jun 2 2014	Steven Levine
修订 0.1-2 首次打印初稿	Thu May 16 2013	Steven Levine

索引

基于时间的表达式，基于时间/日期的表达式

多状态

属性

id，多状态资源：具有多个模式的资源

选项

master-max，多状态资源：具有多个模式的资源

master-node-max，多状态资源：具有多个模式的资源

多状态属性，多状态资源：具有多个模式的资源

多状态选项，多状态资源：具有多个模式的资源

对称，顺序限制

顺序限制，顺序限制

属性

enabled，资源操作

id，资源属性，资源操作，多状态资源：具有多个模式的资源

interval，资源操作

name，资源操作

on-fail，资源操作

provider，资源属性

standard，资源属性

timeout，资源操作

type，资源属性

属性表达式，节点属性表达式

attribute，节点属性表达式

type，节点属性表达式

value，节点属性表达式

操作，节点属性表达式

工作日，日期规格

日期规格，日期规格

年，日期规格

日期规格，日期规格

年日，日期规格

A