KYLINSOFT
麒麟软件

银河麒麟高可用集群软件 V10[1]

管理员手册

麒麟软件有限公司

2024 年 4 月

[1]本手册适用于银河麒麟高可用集群软件 V10（SP3-2403）

目录

银河麒麟最终用户使用许可协议

尊敬的银河麒麟高可用集群软件 V10 的用户：

首先感谢您选用由麒麟软件有限公司开发并制作发行的银河麒麟高可用集群软件 V10（下称“麒麟高可用集群软件”）产品。

请在打开本软件介质包之前，仔细阅读本协议条款以及所提供的所有补充许可条款（统称“协议”）。一旦您打开本软件介质包，即表明您已接受本协议的条款，本协议将立即生效，对您和本公司双方具有法律约束力。

1. 使用许可

按照已为之支付费用的用户数目及计算机硬件类型，麒麟软件有限公司向您授予非排他、不可转让的许可，仅允许内部使用由麒麟软件有限公司提供的随附软件和文档以及任何错误纠正（统称“本软件”）。

— 软件使用许可

在遵守本协议的条款和条件的情况下，麒麟软件有限公司给予贵机构非独占、不可转让、有限的许可，允许贵机构至多使用软件的五（5）份完整及未经修改的二进制格式副本，而此种软件副本仅可安装于贵机构操作的电脑中。

— 教育机构使用许可

在遵守本协议的条款和条件的情况下，如果贵机构是教育机构，麒麟软件有限公司给予贵机构非独占、不可转让的许可，允许贵机构仅在内部使用随附的未经修改的二进制格式的软件。此处的“在内部使用”是指由在贵机构入学的学生、教员和员工使用软件。

— 字型软件使用

软件中包含生成字体样式的软件（“字型软件”）。贵机构不可从软件中分离字型软件。贵机构不可改动字型软件，以新增此等字型软件被作为软件的一部分交付予贵机构时所不具备的任何功能。贵机构不可将字型软件嵌入作为商业产品提供以换取收费或其他报酬的文件。

2. 限制

本软件受到版权（著作权）法、商标法和其他法律及国际知识产权公约的保

护。麒麟软件有限公司和/或其许可方保留对本软件的所有权及所有相关的知识产权。对于麒麟软件有限公司或其许可方的任何商标、服务标记、标识或商号的任何权利、所有权或利益，本协议均不作任何授权。

3. 关于复制、修改及分发

如果在所有复制品中维持本协议书不变，您可以且必须根据《GNU GPL-GNU 通用公共许可证》复制、修改及分发麒麟高可用集群软件产品中遵守《GNU GPL-GNU 通用公共许可证》协议的部分，其他不遵守《GNU GPL-GNU 通用公共许可证》协议的麒麟高可用集群软件产品必须根据符合相关法律之其他许可协议进行复制、修改及分发，但任何以麒麟高可用集群软件产品为基础的衍生发行版未经麒麟软件有限公司的书面授权不能使用任何麒麟软件有限公司的商标或其他任何标志。

特别注意：该复制、修改及分发不包括本产品中包含的任何不适用《GNU GPL-GNU 通用公共许可证》的软件。除非适用法律禁止实施，否则您不得对上述软件进行复制、修改（包括反编译或反向工程）、分发。

4. 有限担保

麒麟软件有限公司向您担保，自购买之日起九十（90）天内（以收据副本为凭证），本软件的存储介质（如果有的话）在正常使用的情况下无材料和工艺方面的缺陷。除上述内容外，本软件按“原样”提供。在本有限担保项下，您的所有补偿及麒麟软件有限公司的全部责任为由麒麟软件有限公司选择更换本软件介质或退还本软件的购买费用。

5. 担保的免责声明

除非在本协议中有明确规定，否则对于任何明示或默示的条件、陈述及担保，包括对适销性、对特定用途的适用性或非侵权性的任何默示的担保，均不予负责，但上述免责声明被认定为法律上无效的情况除外。

6. 责任限制

在法律允许范围内，无论在何种情况下，无论采用何种有关责任的理论，无论因何种方式导致，对于因使用或无法使用本软件引起的或与之相关的任何收益损失、利润或数据损失，或者对于特殊的、间接的、后果性的、偶发的或惩罚性

的损害赔偿，麒麟软件有限公司或其许可方均不承担任何责任（即使麒麟软件有限公司已被告知可能出现上述损害赔偿）。根据本协议，在任何情况下，无论是在合同、侵权行为（包括过失）方面，还是在其他方面，麒麟软件有限公司对您的责任将不超过您就本软件所支付的金额。即使上述担保未能达到其基本目的，上文所述的限制仍然适用。

7. 终止

本协议在终止之前有效。您可以随时终止本协议，但必须销毁本软件的全部正本和副本。如果您未遵守本协议的任何规定，则本协议将不经麒麟软件有限公司发出通知立即终止。终止时，您必须销毁本软件的全部正本和副本。

8. 管辖法律

与本协议相关的任何诉讼均受适用的中华人民共和国法律管辖。任何其它国家和地区的选择法律的规则不予适用。

9. 可分割性

如果本协议中有任何规定被认定为无法执行，则删除相应规定，本协议仍然有效，除非删除妨碍各方愿望的实现（在这种情况下，本协议将立即终止）

10. 完整性

本协议是您与麒麟软件有限公司就其标的达成的完整协议。它取代此前或同期的所有口头或书面往来信息、建议、陈述和担保。在本协议期间，有关报价、订单、回执或各方之间就本协议标的进行的其他往来通信中的任何冲突条款或附加条款，均以本协议为准。对本协议的任何修改均无约束力，除非通过书面进行修改并由每一方的授权代表签字。

11. 商标和标识

贵机构承认并与麒麟软件有限公司有着以下共识，即麒麟软件有限公司拥有麒麟软件、银河麒麟商标，以及所有与麒麟软件、银河麒麟相关的商标、服务标记、标识及其他品牌标识（“麒麟软件标记”）。贵机构对麒麟软件标记的任何使用都应有利于麒麟软件有限公司。

12. 源代码

本软件可能包含源代码，其提供之唯一目的是在符合本协议条款之规定时供

参考之用。源代码不可再分发，除非在本协议中有明确规定。

13. 因侵权而终止

如果本软件成为或在任一方看来可能成为任何知识产权侵权索赔之标的，则任一方可立即终止本协议。

14. Java 技术限制

贵机构不可更改“Java 平台界面”（简称“JPI”，即指明为“java”包或“java”包的任何子包中的类），无论通过在 JPI 中创建额外的类，还是通过其他方式导致对 JPI 中的类进行增添或更动，均为不可。如果贵机构创建一个额外的类以及一个或多个相关的 API，而它们：（i）扩展 Java 平台的功能，（ii）可供第三方软件开发者用于开发可调用上述额外 API 的额外软件，则贵机构必须迅即广泛公布对此种 API 的准确说明，以供所有开发者免费使用。贵机构不可创建、或授权贵机构的被许可人创建以任何方式标示为“java”、“javax”、“sun”的额外的类、界面、子包或 Sun 在任何命名约定中指明的类似约定。参见 Java 运行时环境二进制代码许可的适当版本，以了解可与 Java 小程序和应用程序共同分发的运行时代码的可供情况。

本协议受中华人民共和国法律管辖。

麒麟软件有限公司保留对本协议的最终解释权。

麒麟高可用集群软件介绍

欢迎您使用麒麟高可用集群软件！

麒麟软件有限公司专注于 Linux 系统开发，提供全方位的 Linux 解决方案，麒麟高可用集群软件是基于国产银河麒麟高级服务器操作系统开发的高可用性产品，能够给用户提供多种灵活的高可用组合解决方案，保护用户的服务器集群对外提供不间断的运行环境，我们始终本着功能可靠、使用简单、运行稳定、响应及时的产品宗旨，为企业关键性业务应用达到 7×24 小时不间断运行提供强大的支撑平台。麒麟高可用集群软件的产品特性包括以下几个方面：

（1）多种灵活的集群援备模式

- 双机热备模式

在这种模式下，一台服务器作为主服务器，正常情况下其承担所有的服务。另外一台服务器作为待机服务器，正常情况下除了监控主服务器的状态，不进行其他的操作。一旦主服务器宕机，待机服务器就接手工作，成为新的主服务器。客户仍然可以拥有同样的服务器 IP 地址、NFS、数据、数据库及其它。安装在主机上的 HA 软件通过心跳来实时监测对方的运行状态，一旦正在工作的主机 A 因为各种硬件故障导致系统发生故障，主机 B 立即投入工作。

- 双机互备模式

在这种模式下，两台主机都作为主服务器，各自承担一部分服务。例如服务器 A 在执行应用 A，服务器 B 在执行应用 B，两个主机在正常情况下各自独立运行自己的应用；两个主机同时又都作为对方的待机服务器，通过心跳监控对方的状态。一旦某一服务器宕机，另一台服务器就承担所有的服务，这是一种互为冗余的模式。

- N+M 模式

在这种模式下，有 N 台主机运行业务，M 台主机作为待机服务器，正常情况下除了监控主服务器的状态，不进行其他的操作。一旦运行业务的主机出现故障，待机服务器依据设定的顺序接管业务，成为新的主机，继续对外提供服务。

（2）WEB 集群管理界面

麒麟高可用集群软件提供友好、直观、易操作的集群管理界面，支持资源操作、节点管理、集群配置等。同时管理界面中提供日志下载、集群快捷操作、脚

本生成器等快捷工具，简化用户操作。

日志下载：支持一键下载麒麟高可用集群软件的完整日志及相关调试信息，便于系统管理员进行监控、管理。

脚本生成器：填写脚本名称、启动脚本命令、停止脚步命令和监控进程后，自动生成资源保护脚本并同步到集群内其他节点，优化适配第三方软件的流程，提升产品扩展性。

集群快捷操作：支持在界面上快速执行关键命令查看集群状态。

（3）秒级切换能力

本产品错误检测时间小于 10 秒，故障切换触发时间达到秒级。

（4）极低的系统资源占用

本软件占用系统资源极低，不会与被保护应用争抢系统资源。

（5）多种硬件平台支持

麒麟高可用集群软件同源支持飞腾、鲲鹏、龙芯、兆芯、海光等国产 CPU 和 x86_64 平台，能够最大限度的满足不同平台的需求，并支持多种文件系统及多种存储设备，使其可以灵活的部署。

（6）麒麟高可用集群软件最大支持的节点数

麒麟高可用集群软件可支持的节点数是与用户网络带宽相关，若用户的集群所在网络是光纤千兆网那么我们推荐最高 32 个节点；若用户的网络是近似千兆网和百兆网速之间我们建议用户最高部署 16 个节点。若是百兆网我们推荐 16 个节点以内性能最佳。

文档约定标识

【界面上的文本】或**【屏幕、窗口中的按钮】**

在 GUI 界面屏幕或窗口中的标题、词汇、或短语、菜单选项等会用全角方括号“**【】**”括起来。它用来标明某个 GUI 屏幕或 GUI 屏幕上的某个元素（譬如与复选框或字段相关的文本）。例如：请点击**【确定】**按钮等。

可替代的文字

用在例子中的文本，如使用这种*斜体*方式，表明该文本应被用户提供的数据所代替。

命令

带字符底纹的表明是命令。

 窍门：即一些有用的信息、小技巧等；

 重要：提示请您需要格外重视的内容；

 注记：提醒您关注的事项、注释；

 警告：警示信息，告诫您采取或防止哪些操作；

 小心：情况可能稍有复杂，请您谨慎操作。

技术支持

本产品正式授权默认提供的技术支持服务均以远程方式执行，包括：

- 5*8 小时远程电话、邮件、网站、传真等支持服务：只针对银河麒麟相关产品的安装、使用问题提供支持，不包含对第三方软硬件的支持服务；
- 同版本补丁升级服务。

服务期为合同规定起止日期。如果您有现场服务、培训等其它额外的技术支持需求，请致电麒麟软件有限公司，我们承诺为您提供优质的服务。

地址：天津市滨海高新区塘沽海洋科技园信安创业广场 3 号楼（300450）

北京市海淀区北四环西路 9 号银谷大厦（100190）

长沙市开福区芙蓉中路一段 303 号 CFC 富兴世界金融中心 T3 栋（410028）

上海市番禺路 1028 号数娱大厦 10 层 （200030）

电话：天津（022）58955650 北京（010）51659955 长沙（0731）88280170

上海（021）51098866

传真：天津（022）58955651 北京（010）62800607 长沙（0731）88280166

上海（021）51062866

公司网站：www.kylinos.cn

第一部分 安装与配置

1 环境规划与设置

1.1 硬件环境

- 至少需要 2 个节点，节点可以是物理服务器和虚拟机，每个节点至少需要 2 块网卡做心跳与连接公网使用；
- 建议使用一台 SAN/NAS/ISCSI/NFS 存储作为数据共享存储空间；
- 提供电源管理设备。

1.2 软件环境

- 需要在每台服务器上安装银河麒麟高级服务器操作系统 V10；
- 需要将麒麟高可用集群软件分别安装在每一台服务器上。

1.3 网络环境

- 必须保证每个网卡有一个固定的 IP 地址，该地址用于对外提供相应的服务或心跳；
- 除了连接公网的网卡之外，每台计算机至少另外配置一块网卡作为专用于集群内部的通信，如果是双机配置，私网之间可以通过交叉网线来连接；
- 公网与私网建议使用不同的 TCP/IP 子网地址；
- 集群内节点间的网络需要能够正常通信。

网络环境配置示例如下图所示，其中 10.1.30.137 为虚拟 IP，虚拟 ip 在环境部署过程中，并不与实际网卡绑定。在高可用集群软件中，使用 IPaddr_6 脚本创建虚拟 ip 资源（具体操作参见 3.2.1.5.1 添加 ip 资源）。由运行此资源的节点提供此 ip。当资源发生切换时，虚拟 ip 随之切换。

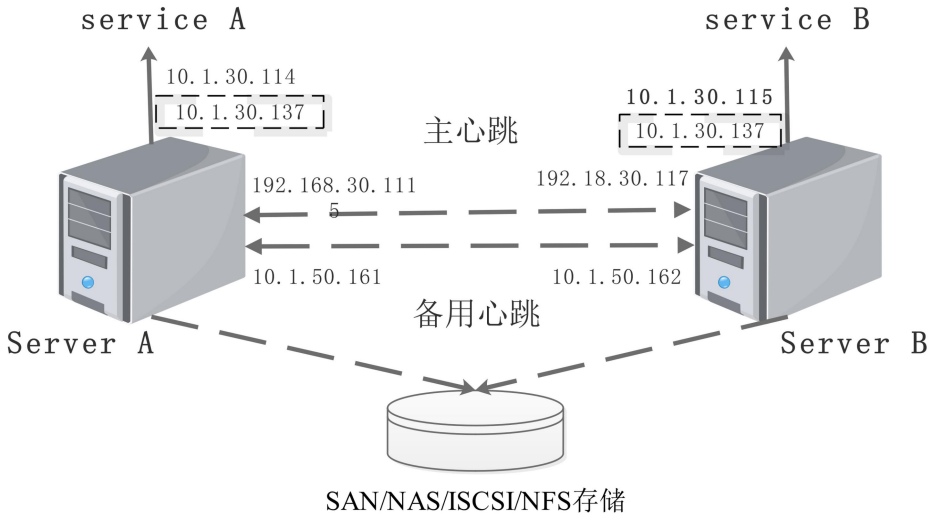


图 1-1 配置示意图

1.4 操作系统环境

- 正确配置服务器系统的网络环境，包括 IP、NetMask、GateWay、DNS 等；
- 两台服务器之间的所有网络地址可以相互通信；
- 正确配置服务器系统的安全防护策略；
- 在/etc/hosts 文件中正确设置主机的名称；
- 正确配置节点间的时钟，确保基本保持一致。

2 安装流程

2.1 安装 KylinHA

2.1.1 /etc/hosts 文件修改

在/etc/hosts 文件中正确设置集群内各节点的名称和节点 IP 的对应关系，其步骤如下：

（1）在每个节点上执行以下命令设置节点名称，其中 host1 为设置的节点名称，可根据用户实际情况设置，每个节点需设置不同的节点名称：

hostnamectl set-hostname *host1* --static

（2）编辑/etc/hosts 文件，添加所有节点 ip 和节点名称的对应关系，如下图所示：

```
127.0.0.1    localhost localhost.localdomain localhost4 localhost4.localdomain4
::1          localhost localhost.localdomain localhost6 localhost6.localdomain6
10.1.30.114 server1
10.1.30.115 server2
```

图 2-1 /etc/hosts 文件示意图

2.1.2 ha.repo 文件修改

将光盘或镜像挂载到一个目录文件下（以/mnt 为例），

光盘可以自动挂载，当进行手动挂载光盘时，执行命令：

mount /dev/cdrom /mnt

使用镜像文件进行手动挂载时，使用命令：

mount -o loop KylinHA-V10.iso /mnt

注：此处 iso 名为代称，部署时需替换成实际 iso 名。

新建/etc/corosync 目录，命令如下：

mkdir /etc/corosync

进入挂载的目录中，将 LICENSE 和.kyinfo 文件复制到/etc/corosync 目录下，如下图所示：

```
[root@host1 mnt]# ls -a /mnt
.  ..  .kyinfo  KylinHA_install.bin  LICENSE  Manual  Packages  scripts  Scripts  TRANS.TBL
[root@host1 mnt]# cp LICENSE .kyinfo  /etc/corosync
```

注记：该步骤必须在安装麒麟高可用集群软件前完成，否则会导致软件授权错误无法使用。

将.productinfo 文件同样复制到/etc/corosync 目录下。

将 scripts 目录下的 ha.repo 文件复制到/etc/yum.repos.d 目录下。如下图所示（以/mnt 为例）：

```
[root@host1 yum.repos.d]# cd /mnt/scripts/
[root@host1 scripts]# ls
corosync.conf.bak  pinghost.sh  scp.expect  ssh.expect  unins_list
ha.repo            rpm_list     ssh_dir.sh  TRANS.TBL
[root@host1 scripts]# cp ha.repo /etc/yum.repos.d
```

编辑 ha.repo 文件，将/tmp/Packags 目录修改为挂载目录中 Packages 文件夹所在路径，如下图所示（以/mnt 为例）：

```
[ks10-ha]
name = Kylin Linux Advanced Server 10 - ha
baseurl = file:///mnt/Packages
```

gpgcheck = 0

gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-kylin

enabled = 1

2.1.3 软件包安装

当使用图形化环境时，运行以下命令安装麒麟高可用集群软件：

```
yum --repo=ks10-ha install ha-api kylinha-scripts
```

当不使用图形化环境时，运行以下命令安装麒麟高可用集群软件：

```
yum --repo=ks10-ha install corosync pacemaker pcs resource-agents pacemaker-remote python3-xmltodict kylinha-scripts
```

安装完成后，运行以下命令将用户手册拷贝到下载目录下，其中/mnt 指 ha 光盘挂载的目录：

```
tar -czf /usr/share/heartbeat-gui/ha-api/static/银河麒麟高可用个集群软件 V10 手册.tar.gz /mnt/Manual
```

注记：图形化界面端口修改：将/usr/share/heartbeat-gui/ha-api/port.ini 文件 haapi_port = 8088 端口修改为空闲端口。

2.1.4 用户密码设置

麒麟高可用集群软件安装后会默认生成 hacluster 用户，使用以下命令为 hacluster 用户设置密码：

```
passwd hacluster
```

回车后输入设置密码，密码要求长度大于等于 8 个字符，至少包含 3 种字符类型。

2.2 操作系统配置

建议关闭系统防火墙、关闭系统 SELinux 安全防护措施，如果开启请确认以下端口可以正常访问：

系统端口： 22、8088、2224、5405、5406。

第二部分 使用

3 使用说明

在使用麒麟高可用集群软件前，需要确认将所有主机名写入/etc/hosts 中。

在启动管理界面前，分别在节点上启动集群的两个主服务，如下所示：

```
[root@server1 ~]# systemctl start pcsd
[root@server1 ~]# systemctl start ha-api
```

在任一节点执行认证命令，如下所示：

```
[root@server1 ~]# pcs host auth 主机名称 1 主机名称 2.....
```

若 pcs 认证显示 unable to communicate with *主机名称*，考虑将防火墙关闭后重新尝试认证，关闭防火墙命令：

```
systemctl stop firewalld
```

3.1 集群

3.1.1 创建集群

3.1.1.1 单心跳网络创建集群

在任一节点上运行以下命令进行集群创建：

```
pcs cluster setup hacluster server1 addr=10.1.30.114 …… servern addr=10.1.30.115
```

注：其中 hacluster 为集群名称，server1…servern 为节点名称，10.1.30.114 和 10.1.30.115 是心跳 IP，均可根据用户现场环境自行设置。

命令打印如下内容，集群创建成功：

```
[root@host1 mnt]# pcs cluster setup hacluster host1 addr=172.30.201.126 host2 addr=172.30.201.130
Destroying cluster on hosts: 'host1', 'host2'...
host1: Successfully destroyed cluster
host2: Successfully destroyed cluster
Requesting remove 'pcsd settings' from 'host1', 'host2'
host1: successful removal of the file 'pcsd settings'
host2: successful removal of the file 'pcsd settings'
Sending 'corosync authkey', 'pacemaker authkey' to 'host1', 'host2'
host1: successful distribution of the file 'corosync authkey'
host1: successful distribution of the file 'pacemaker authkey'
host2: successful distribution of the file 'corosync authkey'
host2: successful distribution of the file 'pacemaker authkey'
Sending 'corosync.conf' to 'host1', 'host2'
host2: successful distribution of the file 'corosync.conf'
host1: successful distribution of the file 'corosync.conf'
Cluster has been successfully set up.
[root@host1 mnt]#
```

注：此处以两节点集群为例，多节点与此类似。

3.1.1.2 冗余心跳网络创建集群

在任一节点上运行以下命令进行集群创建：

pcs cluster setup hacluster server1 addr=10.1.30.114 addr=192.168.30.111 …… servern addr=10.1.30.115 addr=192.168.30.117

注：其中 hacluster 为集群名称，server1…servern 为节点名称，10.1.30.114、192.168.30.111、10.1.30.115 和 192.168.30.117 是心跳 IP，均可根据用户现场环境自行设置。

```
Destroying cluster on hosts: 'host1', 'host2'...
host2: Successfully destroyed cluster
host1: Successfully destroyed cluster
Requesting remove 'pcsd settings' from 'host1', 'host2'
host2: successful removal of the file 'pcsd settings'
host1: successful removal of the file 'pcsd settings'
Sending 'corosync authkey', 'pacemaker authkey' to 'host1', 'host2'
host1: successful distribution of the file 'corosync authkey'
host1: successful distribution of the file 'pacemaker authkey'
host2: successful distribution of the file 'corosync authkey'
host2: successful distribution of the file 'pacemaker authkey'
Sending 'corosync.conf' to 'host1', 'host2'
host1: successful distribution of the file 'corosync.conf'
host2: successful distribution of the file 'corosync.conf'
Cluster has been successfully set up.
[root@localhost package]# _
```

注：此处以两节点集群为例，多节点与此类似。

3.1.2 启动集群

在任一节点上运行以下命令启动集群，运行结果如下图所示：

pcs cluster start --all

```
[root@host1 package]# pcs cluster start --all
host2: Starting Cluster...
host1: Starting Cluster...
[root@host1 package]#
```

3.1.3 查看集群状态

在任一节点上运行以下命令查看集群状态，运行结果如下图所示：

pcs status

```
[root@host1 mnt]# pcs status
Cluster name: hacluster

WARNINGS:
No stonith devices and stonith-enabled is not false

Stack: corosync
Current DC: host1 (version 2.0.2-3.ky10.2.02.ky10-744a30d655) - partition with quorum
Last updated: Tue Jul 13 16:43:27 2021
Last change: Tue Jul 13 16:43:09 2021 by hacluster via crmd on host1

2 nodes configured
0 resources configured

Online: [ host1 host2 ]

No resources


Daemon Status:
  corosync: active/disabled
  pacemaker: active/disabled
  pcsd: active/enabled
[root@host1 mnt]#
```

3.1.4 停止集群

使用以下命令停止集群：

pcs cluster stop [--all | <node>...] [--request-timeout=<seconds>] --force

<node>为节点名称，如果未指定节点，将停止本地节点。如果--all 被指定，则停止集群所有节点。如果集群在运行的资源需要较长时间才能停止，此时在集群实际停止前，资源的停止请求可能会超时。这种情况下，可以设置--request-timeout 为一个合适的值。

pcs cluster kill 命令则强制本地节点的 corosync 和 pacemaker 进程停止。此时系统可检测到集群不在运行然后重新启动。

3.1.5 删除集群

使用如下命令永久删除集群。并且 kill 所有集群进程和删除所有集群配置文件。使用--all 参数， 将删除集群内所有节点的集群信息。

建议在 destroy 集群前运行 pcs cluster stop。

pcs cluster destroy [--all]

3.1.6 集群配置

3.1.6.1 查看集群配置项

在银河麒麟高可用集群软件中所有集群配置项都有默认值，可以通过以下命令查看集群配置项及默认值，运行结果如下图所示：

pcs property --all

```
[root@ha1 ~]# pcs property --all
Cluster Properties:
 batch-limit: 0
 cluster-delay: 60s
 cluster-infrastructure: corosync
 cluster-ipc-limit: 500
 cluster-name: hacluster
 cluster-recheck-interval: 15min
 concurrent-fencing: true
 dc-deadtime: 20s
 dc-version: 2.1.6-9.p06.ky10-6fdc9deea29
 election-timeout: 2min
 enable-acl: false
 enable-startup-probes: true
 fence-reaction: stop
 have-watchdog: false
 join-finalization-timeout: 30min
 join-integration-timeout: 3min
 load-threshold: 80%
 maintenance-mode: False
 migration-limit: -1
 no-quorum-policy: stop
 node-action-limit: 0
 node-health-base: 0
 node-health-green: 0
 node-health-red: 0
 node-health-strategy: none
 node-health-yellow: 0
 pe-error-series-max: -1
 pe-input-series-max: 4000
 pe-warn-series-max: 5000
 placement-strategy: utilization
 priority-fencing-delay: 0
 remove-after-stop: false
 shutdown-escalation: 20min
 shutdown-lock: false
 shutdown-lock-limit: 0
 start-failure-is-fatal: True
 startup-fencing: true
 stonith-action: reboot
 stonith-enabled: False
 stonith-max-attempts: 10
 stonith-timeout: 60s
 stonith-watchdog-timeout: 0
 stop-all-resources: False
 stop-orphan-actions: true
 stop-orphan-resources: true
 symmetric-cluster: True
 transition-delay: 0s
```

3.1.6.2 首选项定义

银河麒麟高可用集群软件中首选项定义如下表所示：

集群配置项	默认值	解释
no-quorum-policy	stop	quorum 策略：当集群中存活的节点存在节点总数半数以上时，集群有效（即有 quorum），如果集群等于或者低于总节点数半数时，集群失效（no-quorum）。 no-quorum-policy 是当集群中 no-quorum 时的策略，选项如下： ignore：继续运行所有资源； freeze：集群分区将会冻结，继续运行资源，正在运行的资源不会停止（但可能重启动以响应监视事件），但不会启动受影响分区中的任何其他资源； stop：受影响集群分区中的所有资源都将以一种有序的方式停止。 suicide：受影响集群分区中的所有节点都将被屏蔽。

symmetric-cluster	TRUE	默认情况下是否所有资源都可以在任何节点上运行。
maintenance-mode	FALSE	是否开启维护模式
start-failture-is-fatal	TRUE	在特定节点上启动资源失败是否会阻止在该节点上进一步启动尝试？如果为 FALSE，集群将根据资源的当前故障计数和迁移阈值决定同一个节点是否仍然符合条件。
stonith-enabled	TRUE	此配置项定义是否允许 STONITH 设备关闭发生故障的节点以及无法停止其资源的节点。默认情况下，此全局选项设置为 true ，因为对于常规的集群操作，有必要使用 STONITH 设备。根据默认值，如果未定义 STONITH 资源，则集群将拒绝启动任何资源。 如果出于任何原因而不使用 STONITH 设备，请将 stonith-enabled 设置为 false，但请注意，这会影响产品的支持状态。此外，在 stonith-enabled="false" 的情况下，分布式锁管理器(DLM) 等资源以及依赖于 DLM 的所有服务（例如 LVM2、GFS2 和 OCFS2）都将无法启动。
node-health-strategy	none	节点健康策略，默认值为 none，其它可选值有 migrate-on-red，only-green，progressive，custom，此项需要与 HealthCPU 等资源配合使用，HealthCPU 资源有 yellow_limit 设置 CPU 空闲率多高时为 yellow 值，默认值为 50%；red_limit 设置 CPU 空闲率多高时为 red 值，默认值为 10%。这些监控系统健康状况

		的资源获取系统状态，然后设置为红、黄、绿。根据节点健康策略确定资源运行在哪个节点上。 none：不追踪节点健康属性。 migrate-on-red：健康属性为红色时节点分值为负无穷，黄色和绿色分值为 0，当某节点的健康属性为红色时，则该节点上的资源将会切换至其他节点。 only-green：健康属性为红色和黄色是节点分值为负无穷，绿色分值为 0，当某节点的健康属性为红色或黄色时，则该节点上的资源将会切换至其他节点。 progressive：将 node-health-red 集群选项的值指定为 red，将 node-health-yellow 的值指定为黄色，将 node- health-green 的值指定为绿色。每个节点额外分配一个 node-health-base 分数（当节点属性为黄色时，该节点可以运行资源）。此策略使管理员可以更好地控制每个值的重要性。 custom：管理员自行定义节点健康属性规则。
node-health-green	0	当节点健康策略为 custom 或者 progressive 时需要设置该项，健康属性为绿色时节点分值。
node-health-yellow	0	当节点健康策略为 custom 或者 progressive 时需要设置该项，健康属性为黄色时节点分值。
node-health-red	0	当节点健康策略为 custom 或者 progressive 时需要设置该项，健康属性为红色时节点分

		值。
stop-all-resources	false	是否应禁止所有资源运行（在维护期间可能很有用）
priority-fencing-delay	0	属性默认处于禁用状态。通过配置延迟值，如果一个节点丢失且其具有更高的总资源优先级，则针对该节点的 Fence 动作将延迟指定的时间。(在脑裂的情况下，在其上运行的具有所有资源的最高组合优先级的节点将更有可能保持正常运行，这在双节点集群的场景下尤其有意义。资源优先级计算中，如果为克隆资源配置了优先级，运行克隆资源的状态为 master 的节点优先级+1。)在配置多种类型的隔离延迟场景下，由 pcmk_delay_base/max 为相应隔离资源配置引入的任何静态或随机延迟，都将被累加到这个延迟中。(priority-fencing-delay 属性应该设置为一个明显大于 pcmk_delay_base 和 pcmk_delay_max 中的最大延迟的值，以确保优先级的节点是首选的。将此属性设置为此值的两倍通常是安全的。)
placement-strategy	default	集群应该如何将资源分配给节点。允许的值有 default、utilization、balanced 和 minimal。 1）default 不考虑利用率值。资源是根据分配分数分配的。如果分数相等，资源在节点之间均匀分布。 2）utilization 在决定节点是否合格（即是否有足够的可用容量来满足资源要求）时，仅考虑利用率值。负载平衡仍然是基于分配给节点的资

		源数量来完成的。 3）balanced 在决定节点是否有资格为资源提供服务以及进行负载平衡时，会考虑利用率值，因此会尝试以优化资源性能的方式分配资源。 4）minimal 在决定节点是否有资格为资源提供服务时，仅考虑利用率值。对于负载平衡，尝试将资源集中在尽可能少的节点上，从而在剩余节点上实现可能的功率节省。

3.1.6.3 首选项配置

通过以下命令进行首选项配置：

```
pcs property set <property>=[<value>]
```

例如将 no-quorum-policy 设置为 ignore 选项命令如下：

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

通过以下命令查看上述命令是否设置成功：

```
pcs property show no-quorum-policy
```

将 stonith-enabled 设置为 false 命令如下：

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

通过以下命令查看上述命令是否设置成功：

```
pcs property show stonith-enabled
```

3.1.6.4 报警配置

通过集群报警配置将集群的故障信息（资源或节点故障）以邮件的形式发送给需要接收信息的管理者。

3.1.6.4.1 查看报警配置

通过以下命令可以查看报警配置：

```
pcs alert show
```

3.1.6.4.2 创建报警邮件发件人

通过以下命令创建报警邮件发件人：

pcs alert create path=<path> [id=<alert-id>] [options [<option>=<value>]...]

在银河麒麟高可用集群中实际创建命令示例如下：

pcs alert create id=alert_Kylin path=/usr/share/pacemaker/alerts/kylin_alert.sh options email_sender=test@kylinos.cn email_server=mailgw.kylinos.cn password=EgMAUEMOUFtYQENHUCcYDg== port=25 switch=on

其中：

id 为给告警分配的 id，如果不写会自动生成；

path 为银河麒麟高可用集群自带的邮件发送插件路径，直接使用上例中的值即可；

options 中的参数为使用邮件发送插件需要的参数，email_sender 为发送邮件的邮箱地址，email_server 为发送邮件的服务器地址，password 为发送邮箱的密码；port 为发送端口号；switch 为发送开关，off 则不发送邮件。

注意，password 字段需要加密存储以保证密码安全，加密方式为使用如下命令得到新的加密密码：

/usr/bin/pwd_encode <'原始密码'>

3.1.6.4.3 修改报警发件人

通过以下命令修改已创建的报警发件人：

pcs alert update <alert-id> [path=<path>] [options [<option>=<value>]...]

其中，<alert-id>是已存在的报警发件人的 id 值，其它参数值参见创建报警发件人。

3.1.6.4.4 删除报警发件人

删除报警发件人命令如下：

pcs alert delete <alert-id>

其中，<alert-id>为报警发件人的 id 值。

注：删除报警发件人，则对应的报警收件人信息也会被同时删除。

3.1.6.4.5 创建报警收件人

报警收件人创建命令如下：

pcs alert recipient add <alert-id> value=<recipient-value> [id=<recipient-id>]

其中，<alert-id>是报警发件人的 id，即接收那个报警发件人的信息；<recipient-value>为收件人的邮箱地址；<recipient-id>为收件人的 id，如果不指定会分配默认值。

如果有多个收件人可以多次执行此命令。

3.1.6.4.6 修改报警收件人

修改报警收件人命令如下：

pcs alert recipient update <recipient-id> [value=<recipient-value>]

其中，<recipient-id>为该报警收件人的 id 值。

3.1.6.4.7 删除报警收件人

删除报警收件人命令如下：

recipient delete <recipient-id>

其中，<recipient-id>为该报警收件人的 id 值。

3.1.7 智能迁移配置

3. 1. 7. 1 配置智能迁移 cpu 指标

3.1.7.1.1 创建 health_cpu 的克隆资源

pcs resource create health_cpu ocf:pacemaker:HealthCPU yellow_limit=cpu_yellow red_limit=cpu_red dampening=30s clone

其中 cpu_yellow 为 cpu 黄色阈值，区间为 0-100 的整数；cpu_red 为 cpu 红色阈值；区间为 0-100 的整数；dampening 为等待进一步变化发生的时间，默认值为 30s。例如，当颜色持续变化时长达到设定时间，颜色才会发生改变，用来消除瞬时变化的影响。此参数同 HealthMEM 资源的 dampening、SysInfo 资源的 delay。

3.1.7.1.2 添加 allow-unhealthy-nodes 属性

pcs resource meta health_cpu-clone allow-unhealthy-nodes=true

3.1.7.1.3 更新 health_cpu 的克隆资源

如需更新资源，则执行以下命令：

pcs resource update health_cpu ocf:pacemaker:HealthCPU yellow_limit=cpu_yellow red_limit=cpu_red dampening=time1 op monitor interval=time2 clone

其中 cpu_yellow 为黄色阈值；cpu_red 为红色阈值；time1 为 dampening 时间，默认为 30s；time2 为 cpu 监测间隔时长，默认为 10s。time1 和 time2 用户可根据生产场景自定义。

3.1.7.2 配置智能迁移内存指标

3.1.7.2.1 创建 health_mem 的克隆资源

pcs resource create health_mem ocf:pacemaker:HealthMEM yellow_mem= mem_yellow red_mem= mem_red dampening=0s clone

其中 mem_yellow 为内存黄色阈值；mem_red 为内存红色阈值；dampening 默认为 0s。

3.1.7.2.2 添加 allow-unhealthy-nodes 属性

pcs resource meta health_mem-clone allow-unhealthy-nodes=true

3.1.7.2.3 更新 health_mem 的克隆资源

pcs resource update health_mem ocf:pacemaker:HealthMEM yellow_mem= mem_yellow red_mem=mem_red dampening=time1 op monitor interval=time2 clone

其中 mem_yellow 为黄色阈值；mem_red 为红色阈值；time1 为 dampening 时间，默认为 0s；time2 为内存监测间隔时长，默认为 20s，用户可根据生产场景自定义。

3.1.7.3 配置智能迁移磁盘指标

3.1.7.3.1 创建 sysinfo 的克隆资源

pcs resource create sysinfo ocf:pacemaker:SysInfo disks=disks min_disk_free= disk_red yellow_disk_free=disk_yellow delay=0s clone

其中 disks 为监控的磁盘，disk_yellow 为磁盘黄色阈值，disk_red 为磁盘红色阈值，delay 默认为 0s。

3.1.7.3.2 添加 allow-unhealthy-nodes 属性

pcs resource meta sysinfo-clone allow-unhealthy-nodes=true

3.1.7.3.3 更新 sysinfo 的克隆资源

pcs resource update sysinfo ocf:pacemaker:SysInfo disks=disk min_disk_free=disk_red yellow_disk_free=disk_yellow delay=time1 op monitor interval=time2 clone

其中 disk 为监控的磁盘设备；disk_red 为红色阈值；disk_yellow 为黄色阈值；time1 为 delay 时间，默认为 0s；time2 为磁盘监测间隔时长，默认为 60s，用户可根据生产场景自定义。

3.1.7.4 配置智能迁移策略

pcs property set node-health-strategy=migrate-on-red

注：node-health-strategy 可以取 migrate-on-red、only-green

3.1.7.5 配置报警

具体方法见章节 3.1.6.4 报警配置。

3.1.8 集群利用率

集群利用率用来配置节点提供的容量或资源所需的容量。您可以根据自己的偏好命名利用率属性，并根据配置需要定义尽可能多的名称/值对。但是，属性的值必须是整数。如果一个节点有足够的空闲容量来满足资源的要求，那么该节点被认为有资格运行这一资源。

配置了节点提供的容量和资源所需的容量后，需要在全局集群选项中设置放置策略（placement-strategy），否则容量配置将无效。可以使用下列命令进行集群放置策略的设置：

```
pcs property set placement-strategy=utilization
```

3.1.8.1 节点利用率

使用以下命令来设置节点利用率：

```
pcs node utilization [[<node>] [--name <name>] | <node> <name>=<value> ...]
```

注：该命令将指定的利用率选项添加到指定的节点。如果未指定节点，则显示所有节点的利用率。如果指定了--name，则显示所有节点中指定的利用率值。如果未指定利用率选项，则显示指定节点的利用率。使用率选项的格式应为 name=value，值必须为整数。可以通过设置不带值的选项来删除选项。

例如：

- 显示集群中设置的所有节点利用率：

  ```
  pcs node utilization
  ```

- 显示集群中 ha1 节点设置的所有利用率：

  ```
  pcs node utilization ha1
  ```

- 显示集群设置的利用率中所有 cpu 的条项：

  ```
  pcs node utilization --name=cpu
  ```

- 删除集群中 ha1 节点 cpu 的利用率：

  ```
  pcs node utilization ha1 cpu=
  ```

3.1.8.2 资源利用率

使用以下命令来设置资源利用率：

pcs resource utilization [<resource id> [<name>=<value> ...]]

注：该命令将指定的利用率选项添加到指定的资源。如果未指定资源，则显示所有资源的利用率。如果未指定利用率选项，则显示指定资源的利用率。使用率选项的格式应为 name=value，值必须为整数。可以通过设置不带值的选项来删除选项。

例如：

- 显示集群中设置的所有资源利用率：

  pcs resource utilization

- 设置集群资源 vip 资源的利用率 cpu 为 1：

  pcs resource utilization vip cpu=1

- 删除集群中 vip 资源的利用率 cpu：

  pcs resource utilization vip cpu=

3.1.9 添加仲裁设备

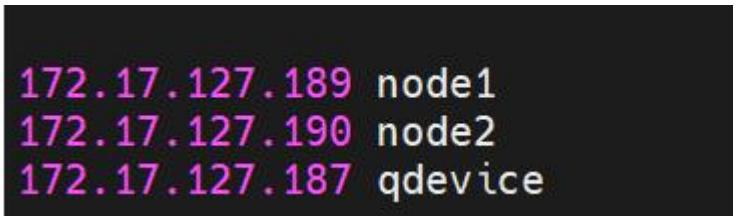
3.1.9.1 仲裁设备作用

仲裁设备通过 QDevice 和 QNet 参与仲裁投票实现。通过 corosync-qnetd，corosync-qdevice 提供一个可配置的投票数，允许集群比常规 quorum 机制下更多的失败节点。强烈建议两点集群使用 corosync-qnetd 和 corosync-qdevice，同样建议在偶数节点集群中使用。

3.1.9.2 仲裁设备环境准备

仲裁设备需要集群外机器运行 corosync-qnetd，成为 qnetd 服务器。以下将此主机命名为 qdevice。

一个集群只能连接到一个 qdevice,而一个 qdevice 可以被多个集群所使用。配置之前关闭 firewalld 和 selinux；修改/etc/hosts 将 qdevice 的主机名和 ip 地址加入进去，同时也需要在 qdevice 上安装 Ha 及 corosync-qnetd 软件包，无需认证到集群中。



分别在 node1，node2 安装 corosync-qdevice，在 HA 的产品 iso 里已包含此

软件包：

rpm -ivh corosync-qdevice-3.0.0-10.ky10.x86_64.rpm

（不同软件包不同，上述名称以 x86 平台为例）

在 qdevice 上安装 corosync-qnetd

rpm -ivh corosync-qnetd-3.0.0-10.ky10.x86_64.rpm

3.1.9.3 配置

在 qdevice 上配置 Quorum Device

直接执行：systemctl enable --now pcsd.service

pcs qdevice setup model net --enable --start

```
[root@qdevice rpms]# pcs qdevice setup model net --enable --start
Quorum device 'net' initialized
quorum device enabled
Starting quorum device...
quorum device started
[root@qdevice rpms]# 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
```

在 node1 或 node2 认证 qdevice，只用在一个节点上执行：

pcs host auth qdevice

```
[root@node2 ~]# pcs host auth qdevice
Username: hacluster
Password:
qdevice: Authorized
```

验证当前配置：pcs quorum config

pcs quorum status 底部的 Qdevice Name 中没有 Qdevice

```
[root@node1 ~]# pcs quorum config
Options:
```

添加 qdevice，执行：pcs quorum device add model net host=qdevice algorithm=ffsplit

之后再执行：pcs quorum config 和 pcs quorum status

```
[root@node1 ~]# pcs quorum config
Options:
Device:
  votes: 1
  Model: net
    algorithm: ffsplit
    host: qdevice
[root@node1 ~]# pcs quorum status
Quorum information
------------------
Date:             Wed Sep 15 17:17:33 2021
Quorum provider:  corosync_votequorum
Nodes:            2
Node ID:          1
Ring ID:          1.2f10
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
```

如上图所示则成功添加 quorum device。

执行：corosync-qdevice-tool -s

```
[root@node1 ~]# corosync-qdevice-tool -s
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:           hacluster
QNetd host:             qdevice:5403
Algorithm:              Fifty-Fifty split
Tie-breaker:            Node with lowest node ID
State:                  Connected
```

查看状态：systemctl status corosync-qdevice

```
[root@node1 ~]# systemctl status corosync-qdevice
* corosync-qdevice.service - Corosync Qdevice daemon
   Loaded: loaded (/usr/lib/systemd/system/corosync-qdevice.service; disabled;>
   Active: active (running) since Wed 2021-09-15 16:37:35 CST; 42min ago
     Docs: man:corosync-qdevice
 Main PID: 2935 (corosync-qdevic)
    Tasks: 2
   Memory: 1.7M
   CGroup: /system.slice/corosync-qdevice.service
           |-2935 /usr/sbin/corosync-qdevice -f
           `-2936 /usr/sbin/corosync-qdevice -f

Sep 15 16:37:35 node1 systemd[1]: Starting Corosync Qdevice daemon...
Sep 15 16:37:35 node1 systemd[1]: Started Corosync Qdevice daemon.
```

注：将 no-quorum-policy 设置为 stop，stonith-enabled 关掉。

执行：pcs property set no-quorum-policy=stop

pcs property set stonith-enabled=false

仲裁磁盘设置后， corosync.conf 文件内容发生变化， 如果之前是两节点集群，则集群中 two_nodes:1 将被覆盖，新的 quorum 一节内容如下：

```
quorum {
    provider: corosync_votequorum

    device {
        model: net
        votes: 1

        net {
            algorithm: ffsplit
            host: qdevice
        }
    }
}
```

将节点间网络心跳断掉，但是需要节点可访问共享设备，此时 1 个节点具有 quroum,而另一节点无 quorum。根据集群设置 no-quorum-policy 为 stop。资源仅会运行在有 quorum 的节点上，从而保证两节点集群在心跳故障时，资源既可运行，同时不会发生资源在两节点同时运行而导致脑裂的现象。

3.2 资源配置

3.2.1 添加资源

银河麒麟高可用集群软件支持三种添加资源的方式，分别为普通资源、组资源和克隆资源。

普通资源：普通资源即 primitive 资源。可以作为独立的资源在集群中运行，也可以作为组资源的子资源，或者通过 clone 成为克隆资源。作为独立资源时，仅运行在一个节点上。

组资源：多个有关联的资源形成组资源。组资源中资源启动时自上而下，关闭时自下而上。

克隆资源：克隆资源可同时运行在多个节点上。主从资源是一种特殊类型的克隆资源，命令如下：

pcs promotable <resource id | group id> [<clone id>] [clone options]... [--wait[=n]]

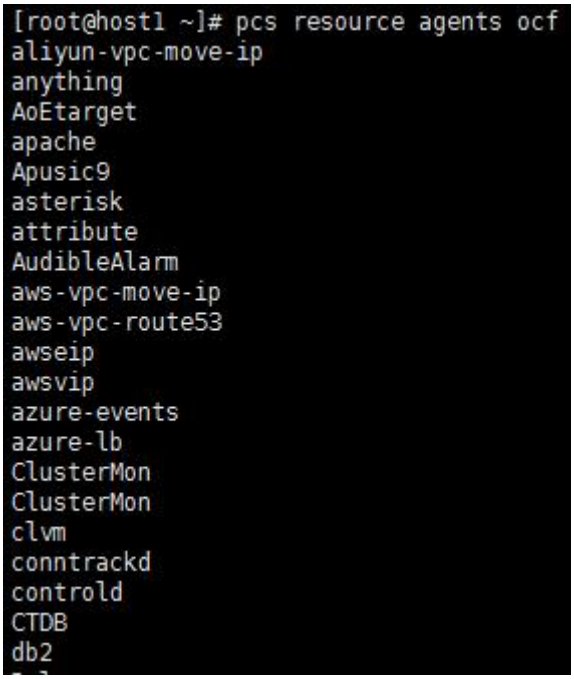
等同于如下命令：

pcs resource clone <resource id> promotable=true

3.2.1.1 添加普通资源

银河麒麟高可用集群软件资源类型分为四大类：ocf、service、systemd、stonith。通过以下命令查看各大类资源包含的资源类型（此处以 ocf 类型举例），运行结果如下图所示：

pcs resource agents ocf



查看某一资源类型参数命令如下，参数中标记为 required 的项为必填项：

pcs resource describe [<standard>:[<provider>:]]<type> [--full]

注：上述命令中[<standard>:[<provider>:]]<type>e 代表资源类型，当添加了

--full 则显示所有参数。

例如：查看 IP 资源参数命令为：pcs resource describe IPaddr_6

```
[root@host1 ~]# pcs resource describe IPaddr_6
Assumed agent name 'ocf:heartbeat:IPaddr_6' (deduced from 'IPaddr_6')
ocf:heartbeat:IPaddr_6 - Manages virtual IPv4 addresses (portable version)

This script manages IP alias IP addresses
It can add an IP alias, or remove one.

Resource options:
  ip (required) (unique): The IPv4 address to be configured in dotted quad notation, for example
                          "192.168.1.1".
  nic (required) (unique): The base network interface on which the IP address will be brought online. If
                           left empty, the script will try and determine this from the routing table. Do
                           NOT specify an alias interface in the form eth0:1 or anything here; rather,
                           specify the base interface only. Prerequisite: There must be at least one static
                           IP address, which is not managed by the cluster, assigned to the network
                           interface. If you can not assign any static IP address on the interface, modify
                           this kernel parameter: sysctl -w net.ipv4.conf.all.promote_secondaries=1 (or per
                           device)
  cidr_netmask (required) (unique): The netmask for the interface in CIDR format. (ie, 24), or in dotted
                                    quad notation 255.255.255.0). If unspecified, the script will also try
                                    to determine this from the routing table.
  broadcast: Broadcast address associated with the IP. If left empty, the script will determine this from
             the netmask.
  iflabel: You can specify an additional label for your IP address here.
  lvs_support: Enable support for LVS Direct Routing configurations. In case a IP address is stopped, only
               move it to the loopback device to allow the local node to continue to service requests, but
               no longer advertise it on the network.
  local_stop_script: Script called when the IP is released
  local_start_script: Script called when the IP is added
  ARP_INTERVAL_MS: milliseconds between ARPs
  ARP_REPEAT: How many gratuitous ARPs to send out when bringing up a new address
  ARP_BACKGROUND: run in background (no longer any reason to do this)
  ARP_NETMASK: netmask for ARP - in nonstandard hexadecimal format.

Default operations:
  start: interval=0s timeout=90
  stop: interval=0s timeout=100
  monitor: OCF_CHECK_LEVEL=10 interval=5s start-delay=1s timeout=20s
[root@host1 ~]#
```

添加某一资源命令如下（以 IP 资源为例）：

pcs resource create vip ocf:heartbeat:IPaddr_6 ip=172.30.201.131 nic=ens3 cidr_netmask=255.255.255.0

注：其中 vip 为添加资源名称，可根据用户现场实际情况填写，IPaddr_6 为资源类型，ip 和 cidr_netmask 为添加 IPadd_6 资源类型所必填的参数。

通过以下命令查看集群内资源状态，运行结果如下图所示：

pcs resource status

```
[root@host1 ~]# pcs resource status
 vip    (ocf::heartbeat:IPaddr_6):      Started host1
[root@host1 ~]#
```

3.2.1.2 添加组资源

添加组资源时，集群中需要至少存在一个普通资源。添加组资源命令如下：

pcs resource group add <group id> <resource id> [resource id] ... [resource id] [--before <resource id> | --after <resource id>] [--wait[=n]]

注：上述命令中 group id 为组资源名称，可根据实际情况自行命名。<resource

id> [resource id] ... [resource id][--before <resource id> | --after <resource id>] [--wait[=n]]为添加到组资源内包含的子资源名称。组资源的启动是按照子资源的顺序启动的，所以写子资源时需要注意按照顺序填写。

通过以下命令查看集群内的组资源状态：

pcs resource group list

例如：将 http、vip 资源添加到组资源 group1 中的命令如下，集群内组资源状态运行结果如下图所示

pcs resource group add group1 http vip

```
[root@host1 ~]# pcs resource group list
group1: vip http
[root@host1 ~]#
```

3.2.1.3 添加克隆资源

添加克隆资源命令如下：

pcs resource clone <resource id | group id> [clone options]... [--wait[=n]]

注：其中 resource id | group id 为克隆对象资源名称，[clone options]... [--wait[=n]]为元属性，可根据实际情况添加，否则将按照系统默认配置值处理。

例如：克隆组资源 group1 命令如下，克隆后集群内资源状态如下图所示：

pcs resource clone group1 globally-unique=true clone-max=2 clone-node-max=2

```
[root@host1 ~]# pcs resource status
 Clone Set: group1-clone [group1]
     Started: [ host1 host2 ]
[root@host1 ~]#
```

3.2.1.4 资源属性

实例属性：

在集群中添加一个资源时，资源使用的具体参数即为资源的实例属性。不同类型资源实例属性不同。具体命令可参见资源添加示例中的参数设置。

元属性：

元属性都是针对某一资源设置，定义如下表所示：

元属性	默认值	解释
target-role	started	集群保持资源所处的状态， 支持两种选项： stopped：停止资源

		started：允许启动资源
priority	0	如果资源无法全部工作，集群会停止低优先级的资源，以便以高优先级的资源工作。priority 用来定义资源的优先级。
is-managed	TRUE	是否允许集群启动或停止该资源
resource-stickiness	0	资源运行在节点上的附加分值
migration-threshold	INFINITY	允许资源失败的次数，当资源失败次数达到所设值后将切换到其他节点上运行
multiple-active	stop_start	设置当集群发现资源同时在多个节点上运行时的操作，有以下选项可供选择： block：将该资源标记为 unmanaged； stop_only：停止该资源 stop_start：在同时运行的节点上停止该资源，并在某一节点启动资源
failure-timeout	0	资源出现故障后等待时间设置，当达到所设时间将进行自动清理
allow-migrate	ocf:pacemaker:remote 资源默认为 TRUE，其他资源默认为 FALSE	是否实时执行迁移资源操作
allow-unhealthy-nodes	FALSE	资源是否应该能够在节点

		上运行，即使节点的运行状况分数会阻止它运行

添加元属性命令如下：

pcs resource meta <resource id | group id | clone id> <meta options> [--wait[=n]]

注：<resource id | group id | clone id>为普通资源名称/组资源名称/克隆资源名称，<meta options>为元属性设置，格式为 name=value，name 为上表中所列的元属性，当 value 设置为空时则删除该元属性。[--wait[=n]]设置该项后将等待 n 秒使配置生效。

例如：设置某资源 target-role 属性为 stopped 命令如下：

pcs resource meta testreource target-role=stopped

操作属性：

操作属性都是针对某一资源设置，操作是指集群对资源执行的操作，例如启动、停止、状态、监控等，定义如下表所示：

操作属性	解释
name	需要执行的操作，包括以下几个选项： start：启动 stop：停止 monitor：监控 注：所有资源都必须添加监控操作
role	角色：只在所选角色的节点上运行该操作，角色包括 stopped（已停止）、started（已启动）、master（仅克隆资源可选）、slave（仅克隆资源可选）。
on-fail	资源故障时采取的操作，包括以下几个选项：、 ingore：忽略资源故障 block：不对资源执行其他操作 stop：停止资源并在其他节点上不启动 restart：重启资源（重启后该资源可能在其他节点运行） fence：执行 fence 操作，如果集群已配置 fence 资源，stop 操作的默认值为 fence

	standby：将该资源发生故障的节点上所有资源迁移到其他节点。
interval	执行该操作的频率，0 代表不执行该操作，正值定义重复性操作，该选项通常在 name 为 monitor 时设置。
timeout	宣布资源失败前等待时间
OCF_CHECK_LEVEL	脚本附加监控

添加操作属性命令如下：

pcs resource op add <resource id> <operation action> [operation properties]

注：<operation action>为上表中的 name 值[operation properties]为上表中其他值设置。

例如：设置某资源启动超时时间为 90 秒命令如下：

pcs resource op add testresource start timeout=90s

3.2.1.5 添加资源示例

3.2.1.5.1 添加 IP 资源

添加 IP 资源命令如下：

pcs resource create ipp ocf:heartbeat:IPaddr_6 ip=192.168.1.1 nic=ens192 cidr_netmask=255.255.255.0

注：其中 ipp 为添加资源名称，可根据用户现场实际情况填写，IPaddr_6 为资源类型，ip，nic 和 cidr_netmask 为添加 IPadd_6 资源类型所必填的参数。

通过以下命令查看集群内资源状态。运行结果如下图所示：

pcs resource status

```
[root@sp2-04 ~]# pcs resource status
  * ipp (ocf::heartbeat:IPaddr_6):       Started sp2-03
```

3.2.1.5.2 添加文件系统资源

添加 Filesystem 资源命令如下：

pcs resource create fs ocf:heartbeat:Filesystem device=/dev/sdb3 directory=/data fstype=ext4 op monitor OCF_CHECK_LEVEL=10

注：其中 fs 为添加资源名称，可根据用户现场实际情况填写，Filesystem 为资源类型，device，directory 和 fstype 为添加 Filesystem 资源类型所必填的参数。op monitor OCF_CHECK_LEVEL 为操作属性，设置该属性后在拔存储线时文件

系统资源会进行切换。

device：文件系统的块设备名称，或用于挂载的-U、-L 选项，或 NFS 挂载规范。

directory：文件系统的挂载点。

fstype：要挂载的文件系统类型。

通过以下命令查看集群内资源状态。运行结果如下图所示：

pcs resource status

```
[root@sp2-1 ~]# pcs resource status
  * fs  (ocf::heartbeat:Filesystem):     Started sp2-1
```

3.2.1.5.3 添加数据库资源

添加 DMDB8 资源示例命令如下：

pcs resource create dm ocf:pacemaker:DMDB8 datadir=/opt/dmdbms instancedir=/opt/dmdbms/DAMENG

注：其中 dm 为添加资源名称，可根据用户现场实际情况填写，DMDB8 为资源类型，datadir 和 instancedir 为添加 DMDB8 资源类型所必填的参数。

datadir：包含数据库目录。

instancedir：数据库实例路径。参数可根据自己的环境配置。

通过以下命令查看集群内资源状态。运行结果如下图所示：

pcs resource status

```
  * dm  (ocf::pacemaker:DMDB8):  Started sp2-04
```

3.2.1.5.4 添加中间件资源

添加 InforSuite 资源命令如下

pcs resource create in ocf:pacemaker:InforSuite InforSuite_HOME=/InforSuite-AS-StE-9.1B050100 port=8060 sleeptime=15

注：其中 in 为添加资源名称，可根据用户现场实际情况填写，InforSuite 为资源类型，InforSuite_HOME port 和 sleeptime 为添加 InforSuite 资源类型所必填的参数。

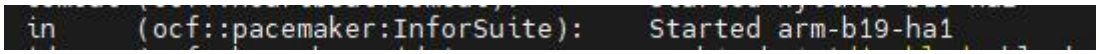
InforSuite_HOME :包含 InforSuite 的目录。

port ： InforSuite 的端口。

sleeptime ：启动进程需要休眠的时间，可能需要更改。

通过以下命令查看集群内资源状态。运行结果如下图所示：

pcs resource status



3.2.1.5.5 添加 fence 资源

在使用 fence 资源前，需要保证 ipmitool 正常工作：

这里列举一个 ipmitool 的使用案例：

ipmitool -H 10.1.123.40 -U ADMIN -P ADMIN power status -I lan

-H 为电源管理口地址

-U 为用户名

-P 为密码

-I: lan 为默认值，可以不写。如果是 lanplus，需要-I lanplus

返回值为：Chassis Power is on 说明主机运行正常。

当需使用 fence 资源时，使用以下命令安装软件包：

yum --repo=ks10-ha install fence-agents-ipmilan

在首选项的 stonith-enabled 设置为 true 的情况下，必须添加 Fencing 资源之后，集群中的普通资源才能运行。实例属性添加 ipaddr、lanplus、login、passwd、method。

pcmk_host_list 为被控制的主机列表

ipaddr：fence 设备的 ip 地址，这里使用 bmc 的地址。

lanplus：是否使用 lanplus 提高连接的安全性。

login：fence 设备（bmc）的登录名。

passwd：fence 设备（bmc）的登录密码。

method：fence 方式。

添加 fence 资源的命令如下（bmc 每一个 ip 对应一个 host）：

pcs stonith create fence fence_ipmilan pcmk_host_list=host1 delay=10 ipaddr=host1 的 bmc 地址 login=ADMIN passwd=**** lanplus=TRUE

pcs stonith create fence fence_ipmilan pcmk_host_list=host2 ipaddr=host2 的 bmc 地址 login=ADMIN passwd=**** lanplus=TRUE

当资源启动后，bmc 控制的主机会重启。由于集群在 bmc 控制的主机上，因此主机会重启，HA 集群会关闭。

3.2.1.5.6 添加 SBD 资源

当需使用 SBD 资源时，使用以下命令安装软件包：

```
yum --repo=ks10-ha install fence-agents-sbd
```

假设 SBD 设备的名称为/dev/sda1，以下示例中都使用该名称的设备，应用时替换为实际设备名称即可。

在集群某一个使用该共享设备的节点上，执行如下命令：

```
sbd -d /dev/sda1 create
```

命令会在设备上写一个 header，并为多达 255 个共享该设备的节点创建 slot。

目前支持最多三个设备，以/dev/sda1、/dev/sda2、/dev/sda3 为例，应用时替换为实际设备名称即可。

如果将多个设备应用于 sbd，请多次指定-d 选项，命令如下：

```
sbd -d /dev/sda1 -d /dev/sda2 -d /dev/sda3 create
```

可以使用如下命令查看写入的内容：

```
sbd -d /dev/sda1 dump
```

其执行结果如下：

```
==Dumping header on disk /dev/sda1
Header version         : 2.1
UUID                   : 04a365c3-f12a-45d4-bfcc-7f6096a18f36
Number of slots        : 255
Sector size            : 512
Timeout (watchdog) : 5
Timeout (allocate) : 2
Timeout (loop)         : 1
Timeout (msgwait)    : 10
==Header on disk /dev/sda1 is dumped
```

可以看到，超时也写在 header 中，以确保所有参与的节点都同意。

安装 watchdog

使用适合你硬件的 watchdog 驱动，如 HP 服务器使用 hpwdt。如果没有与你硬件匹配的 watchdog 则可以使用 softdog。其配置命令如下：

```
/sbin/modprobe    softdog
```

需要在集群的所有节点上执行该命令。

如果需要系统开机默认加载该功能，可以使用如下命令：

```
echo softdog > /etc/modules-load.d/softdog.conf
```

注记：如果系统中已存在 watchdog（查看/dev 下是否有 watchdog），则新增加的 softdog 在 dev 中可能显示为 watchdog+数字的形式，如 watchdog1。这种情况下 softdog 不是默认的/dev/watchdog，需要修改 sbd 的配置文件/etc/sysconfig/sbd，将变量 SBD_WATCHDOG_DEV 的值修改为此时 softdog 的值，如 SBD_WATCHDOG_DEV=/dev/watchdog1。

启动 SBD Daemon

sbd daemon 是集群至关重要的一部分。当集群启动甚至崩溃的时候它都必须运行。

以下操作需要在集群的每个节点上都执行。

- 编辑 sbd 配置文件

  将下面代码加入到/etc/sysconfig/sbd 中：

  ```
  SBD_DEVICE="/dev/sda1"
  SBD_OPTS="-W"
  ```

  其中， -W 是支持 watchdog，是强烈建议做的。如果需要指定多设备，使用分号来分隔（顺序并不重要）：

  ```
  SBD_DEVICE=”/dev/sda1;/dev/sda2;/dev/sda3”
  ```

  如果 sbd 设备不可访问，Daemon 将启动失败。

- 启动 sbd Daemon

使用 `systemctl enable sbd` 命令，会创建启动链接，sbd Daemon 会随着集群的启动而启动，如果集群已经启动，需要重启集群使 sbd 服务启动。

如果 sbd 服务启动失败则集群无法正常启动。

取消 sbd 服务启动，使用 `systemctl disable sbd` 命令。

如果没有共享磁盘设备，默认是监控 pacemaker；如果要使用共享磁盘，需要将设备名称写入到/etc/sysconfig/sbd 中。

添加 SBD 资源的命令如下：

```
pcs stonith create fence_sbd fence_sbd pcmk_host_list=host1,host2 devices=/dev/sdb pcmk_delay_base=”host1:1s,host2:5s”
```

其中 host1,host2 为需要重启的主机名，/dev/sdb 为 sbd 设备的路径名称，存

在多个磁盘设备时，使用逗号分隔。

通过 pcs stonith 查看 fence_sbd 资源的状态：

```
[root@ha1 ~]# pcs stonith create FenceSBD fence_sbd devices=/dev/mapper/vg-sbd1 pcmk_host_list=ha1,ha2 pcmk_delay_base="ha1:1s,ha2:5s"
[root@ha1 ~]# pcs stonith enable FenceSBD
[root@ha1 ~]# pcs stonith
  * FenceSBD    (stonith:fence_sbd):     Started ha1
```

3.2.1.5.7 添加 ping 资源

添加 ping 资源的命令如下：

pcs resource create ping ocf:pacemaker:ping host_list=172.30.201.254, 172.30.202.254 dampen=5 multiplier=1000 name=pingd

其中，host_list 为想要 ping 的 ip 地址，一般为业务网关，可以是多个，中间用逗号隔开；dampen 为等待进一步变化发生的时间；multiplier 为 ping 通一个主机列表中的主机可以获得的分值，注意要和限制条件里的分值要匹配；pingd 为 ping 资源参数 name 的默认值，可以修改，注意保持一致。

在解决脑裂问题时将 ping 资源设置为克隆资源 ping-clone

pcs constraint location 资源名 rule score=-INFINITY not_defined pingd

pcs constraint location 资源名 rule score=-INFINITY pingd lt 1000

集群的 ping 资源状态如下所示：

```
[root@ha1 ~]# pcs resource create ping ocf:pacemaker:ping host_list=ha1,ha2 dampen=5 multiplier=1000
[root@ha1 ~]# pcs resource
  * ping        (ocf::pacemaker:ping):   Started ha2
```

3.2.1.5.8 添加 bundle 资源

bundle 用于管理容器映像的多个实例，以及容器所需的网络和存储。

在使用集群的 bundle 功能时，集群节点需要有容器相关软件包及容器中的镜像。Bundle 支持三种容器：podman , docker, rkt。 以下例子以 docker 为例，镜像名称为： pcmktest:http2。

在下面例子中，bundle 绑定普通资源 httpd，资源类型为 ocf:heartbeat:apache。Bundle 资源中的端口及存储映射均为此资源运行而设置。在 pcmktest:http2 镜像中需要具备的软件包入下：

httpd bind-utils lsof wget resource-agents openssh-clients pacemaker pacemaker-remote

bundle 资源操作如下：

1.创建 bundle 资源

创建 bundle 资源命令如下：

pcs resource bundle create <bundle id> container <container type> [<container options>] [network <network options>] [port-map <port options>]... [storage-map <storage options>]... [meta <meta options>] [--disabled] [--wait[=n]]

注：其中 bundle id 为需要创建的 bundle 资源名称，container type 为所选的容器类型；[container options]为容器中的配置，包括启动的镜像、数量和命令；[network <network options>]为网络配置，包括 ip 和网卡以及掩码。

一般来说测试集群必须连接到有多个顺序的、可用的未使用的 IP 地址，实际数量由 container 参数 replicas 指定，但是这些 ip 地址必须是连续的。

添加 bundle 资源命令如下：

pcs resource bundle create httpd-bundle container docker image="pcmktest:http2" replicas="2" run-command="/usr/sbin/pacemaker_remoted" network ip-range-start="172.17.127.185" host-interface="ens33" 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" storage-map id="httpd-logs" source-dir-root="/var/log/pacemaker/bundles" target-dir="/etc/httpd/logs" options="rw"

replicas 表示的是需要启动的容器数量，ip-range-start 即表示有序 ip 的起始位置，例如本次设置中将会占用 172.17.127.185-186 两个 ip，容器、镜像名、网卡掩码请按照实际情况进行设置。

2.bundle 资源参数修改

查看 httpd-bundle 资源参数：pcs resource config httpd-bundle

```
[root@node1 kylinha-scripts]# pcs resource config httpd-bundle
 Bundle: httpd-bundle
  Docker: image=pcmktest:http2 replicas=2 run-command=/usr/sbin/pacemaker_remoted
  Network: host-interface=ens33 host-netmask=24 ip-range-start=172.17.127.185
  Port Mapping:
   port=80 (httpd-port)
  Storage Mapping:
   options=rw source-dir-root=/var/local/containers target-dir=/var/www/html (httpd-root)
   options=rw source-dir-root=/var/log/pacemaker/bundles target-dir=/etc/httpd/logs (httpd-logs)
```

如果需要修改某一项参数：

pcs resource bundle update <bundle id> [container <container options>] [network <network options>] [port-map (add <port options>) | (delete | remove <id>...)]... [storage-map (add <storage options>) | (delete | remove <id>...)]... [meta <meta options>][--wait[=n]]

例如修改容器数量：pcs resource bundle update httpd-bundle container

replicas="1"（修改容器配置和网络配置方法相同）

添加或者删除端口 port-map 和存储映射 storage-map 注意在关键词后加入 add 或者 delete，例如之前的配置可以拆分成两步：

pcs resource bundle create httpd-bundle container docker image="pcmktest:http2" replicas="2" run-command="/usr/sbin/pacemaker_remoted" network ip-range-start="172.17.127.185" host-interface="ens33" host-netmask="24"

pcs resource bundle update httpd-bundle port-map add id="httpd-port" port="80" storage-map add id="httpd-root" source-dir-root="/var/local/containers" target-dir="/var/www/html" options="rw" storage-map add id="httpd-logs" source-dir-root="/var/log/pacemaker/bundles" target-dir="/etc/httpd/logs" options="rw"

reset 指令修改现有的 bundle 资源需要带上镜像名称，可以更改也可以不更改，如下所示：

pcs resource bundle reset <bundle id> [container <container options>] [network <network options>] [port-map <port options>]... [storage-map <storage options>]... [meta <meta options>] [--disabled] [--wait[=n]]

修改镜像名称：

pcs resource bundle reset httpd-bundle container image="pcmktest:http" replicas="2" run-command="/usr/sbin/pacemaker_remoted"

修改其他参数，与 update 类似：

pcs resource bundle reset httpd-bundle container image="pcmktest:http2" network ip-range-start="172.17.127.186"

注：必须带上镜像名称

3.绑定普通资源

绑定 bundle 和普通资源命令如下：

pcs resource create httpd ocf:heartbeat:apache bundle httpd-bundle

再次运行 pcs resource config httpd-bundle：

```
[root@node1 ~]# pcs resource config httpd-bundle
 Bundle: httpd-bundle
  Docker: image=pcmktest:http2 replicas=2 run-command=/usr/sbin/pacemaker_remoted
  Network: host-interface=ens33 host-netmask=24 ip-range-start=172.17.127.185
  Port Mapping:
   port=80 (httpd-port)
  Storage Mapping:
   options=rw source-dir-root=/var/local/containers target-dir=/var/www/html (httpd-root)
   options=rw source-dir-root=/var/log/pacemaker/bundles target-dir=/etc/httpd/logs (httpd-logs)
  Resource: httpd (class=ocf provider=heartbeat type=apache)
   Operations: monitor interval=10s timeout=20s (httpd-monitor-interval-10s)
               start interval=0s timeout=40s (httpd-start-interval-0s)
               stop interval=0s timeout=60s (httpd-stop-interval-0s)
```

4.查看运行结果

执行 crm_mon，可查看运行结果如下：

```
Node List:
  * Online: [ node1 node2 ]
  * GuestOnline: [ httpd-bundle-0@node2 httpd-bundle-1@node1 ]

Active Resources:
  * Container bundle set: httpd-bundle [pcmktest:http2]:
    * httpd-bundle-0 (172.17.127.185)   (ocf::heartbeat:apache):         Started node2
    * httpd-bundle-1 (172.17.127.186)   (ocf::heartbeat:apache):         Started node1
```

在 node1 节点上执行 ip a 命令，可以看到 172.17.127.186 的 ip 地址。

在浏览器中可以对 http://172.17.127.186:80 进行访问。此服务即为运行在 node1 上的容器提供的 httpd 服务。

3.2.1.5.9 添加 fence_virsh 资源

此脚本是高可用节点为虚拟机时执行 fence 动作的资源。在使用 fence_virsh 资源前，需要安装 fence-agents-virsh 软件包。

这里列举一个 fence_virsh 的使用案例：

```
fence_virsh --ip=** --username=root --password=** -x -n 虚拟域的名称
```

此条命令执行时，需要虚拟机能访问到物理机，才可以让指定的虚拟域重启，用来验证 fence_virsh 是否可用。

其中 ip 为被 fence 虚拟机所在物理机的 ip 地址，username 为物理机的登录用户，password 为物理机的登录密码，两侧使用单引号，避免!+数字方式的干扰，!+数字为历史第 n 条执行命令，x 为使用 ssh 连接方式，n 为虚拟域的名称或者 uuid。

在首选项的 stonith-enabled 设置为 true 的情况下，必须添加 Fence 资源后，集群中的普通资源才能运行。基本属性添加 ip、ipaddr、login、passwd、pcmk_host_list、username，实例属性添加 plug，其中 ip、ipaddr 为被 fence 虚拟机所在物理机的 ip 地址，login、username 为物理机的登录用户，passwd 为物理机的登录密码，pcmk_host_list 为被控制的主机列表，plug 为虚拟域的名称。

添加 fence_virsh 资源命令如下：

pcs stonith create fence_virsh ip=** ipaddr=** login=root passwd=*** pcmk_host_list=** plug=** username=root

当资源启动并触发 fence 后，被控制的虚拟机会重启。

3.2.1.5.10 添加虚拟机资源

虚拟机是一种特殊的资源，以 VirtualDomain 资源为例。此脚本 VirtualDomain 是由 libvirtd 管理的虚拟域(也称为 domU，虚拟机，虚拟环境等，取决于上下文)的资源代理。可控制虚拟机的开机、关机、迁移。

此脚本运行需要满足以下条件：

1.服务器作为高可用集群的节点，配置节点间免密登录，配置方法如下：

a.生成公钥和私钥对

ssh-keygen -t rsa

b. 将公钥复制到其他的节点上

ssh-copy-id -i /root/.ssh/id_rsa.pub root@ip

2.所有节点开启 libvirtd 服务

3.所有节点的 cpu 型号需要相同

4.所有节点配置共享存储，保证每一个节点都能访问到共享存储

5.在其中一个服务器上安装虚拟机，存储位置选择共享存储

6.使用 virsh dumpxml file > /etc/libvirt/qemu/file.xml (此处 file 为虚拟机的名称，需替换为实际的虚拟机名称)生成虚拟机的配置文件，并在所有节点的相同位置放置一份。

注：以上为虚拟机支持热迁移所需条件。如果虚拟机无需热迁移，则需要条件为：

1.服务器作为高可用集群的节点，并配置节点间免密登录

2.所有节点开启 libvirtd 服务

3.所有节点配置共享存储，保证每一个节点都能访问到共享存储

4.在其中一个节点上安装虚拟机，存储位置选择共享存储

5.将虚拟机的 xml 文件拷贝到其它节点（注意此处使用 dump 会包含 CPU 信息，可能会导致在其它节点上启动失败）

添加 VirtualDomain 资源命令如下：

pcs resource create VirtualDomain ocf:heartbeat:VirtualDomain config=/etc/libvirt/qemu/file.xml migrate_options=--persistent migration_transport=ssh

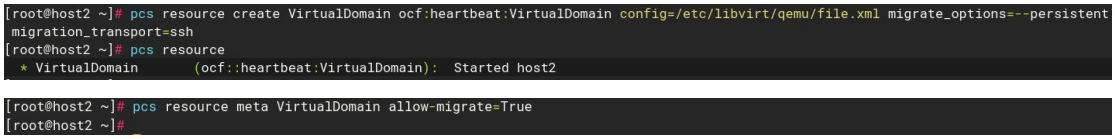
其中/etc/libvirt/qemu/file.xml 为虚拟机的 xml 文件位置。

添加 allow-migrate 元属性命令如下：

pcs resource meta VirtualDomain allow-migrate=True

如果虚拟机无需热迁移，allow-migrate 设置为 False；反之，设置为 True。

虚拟机资源创建如下所示：



3.2.2 修改资源

3.2.2.1 修改普通资源参数

编辑资源命令如下：

pcs resource update <resource id> [resource options] [op [<operation action> <operation options>]...] [meta <meta operations>...] [--wait[=n]]

注：其中 resource id 为需要编辑的资源的资源名称， [resource options] [op [<operation action> <operation options>]...] [meta <meta operations>...] [--wait[=n]] 为需要编辑的资源的参数，根据用户现场实际填写相应内容。

通过以下命令查看修改后该资源的参数：

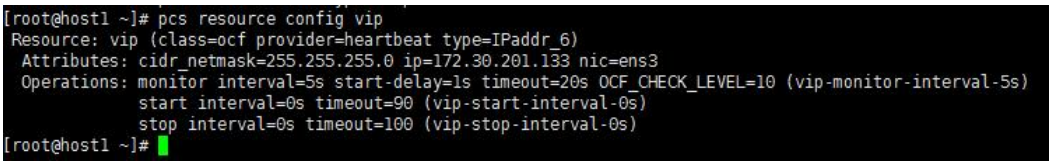
pcs resource config <resource id>

例如：将 vip 资源的 ip 地址修改为 172.30.201.133 的命令如下：

pcs resource update vip ip=172.30.201.133

查看修改后的 vip 资源参数：pcs resource config vip

运行结果如下图：



3.2.2.2 修改组资源

在已有组资源中增加新的子资源命令：

pcs resource group add <group id> <resource id> [resource id] ... [resource id] [--before <resource id> | --after <resource id>] [--wait[=n]]

例如：在已有组资源中 group1 中添加 dummy 资源，查看集群内组资源状态如下图所示：

pcs resource group add group1 dummy

```
[root@host1 ~]# pcs resource group list
group1: vip http dummy
[root@host1 ~]#
```

在已有组资源中移除某一子资源命令：

pcs resource group remove <group id> <resource id> [resource id] ... [resource id] [--wait[=n]]

例如：将 group1 组资源中的 dummy 子资源移除命令如下，查看集群内组资源状态如下图所示：

pcs resource group remove group1 dummy

```
[root@host1 ~]# pcs resource group remove group1 dummy
[root@host1 ~]# pcs resource group list
group1: vip http
[root@host1 ~]#
```

3.2.3 启动资源

启动资源命令如下：

pcs resource enable <resource id>... [--wait[=n]]

注：其中<resource id>... [--wait[=n]]为需启动的资源名称。

例如：启动 http 和 dummy 资源命令如下，集群内资源状态如下图所示：

pcs resource enable http dummy

```
[root@host1 ~]# pcs resource enable http dummy
[root@host1 ~]# pcs resource status
 http   (service:httpd):        Started host1
 dummy  (ocf::heartbeat:Dummy): Started host2
[root@host1 ~]#
```

3.2.4 停止资源

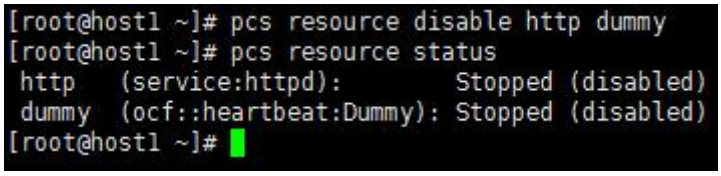
停止资源命令如下：

pcs resource disable <resource id>... [--wait[=n]]

注：其中<resource id>... [--wait[=n]]为需停止的资源名称。

例如：停止 http 和 dummy 资源命令如下，集群内资源状态如下图所示：

pcs resource disable http dummy



3.2.5 清理资源

当资源在某一节点上运行失败后，将会切换到其他节点上继续运行，如果用户进行了报警配置，在邮箱中会收到资源失败的报警邮件。此时需要人为介入对运行失败的资源进行修复，当资源修复成功后，必须执行清理资源的操作，资源才可在此节点上重新运行。

清理资源的命令如下：

pcs resource cleanup [<resource id>] [node=<node>]

注：其中<resource id>为资源名称，如果不指定将会清理所有资源状态；[node=<node>]为节点主机名，若不指定将对所有节点上的该资源状态进行清理。

例如：

- 清理所有节点上的 vip 资源：pcs resource cleanup vip
- 清理所有节点上的所有资源：pcs resource cleanup

3.2.6 迁移资源

银河麒麟高可用集群软件支持将处于运行状态的资源迁移至其他节点上运行。支持选择是否为强制迁移

迁移资源命令：

pcs resource move <resource id> [destination node] [lifetime=<lifetime>] [--wait[=n]]

注：

- <resource id>为要迁移的资源名称；
- [destination node]为迁移到的节点名称；
- [lifetime=<lifetime>]为迁移有效期，即资源迁移到目标主机后，资源限制在目标主机上运行持续时间。例如：资源 R 从主机 A 迁移到主机 B，有效时间是 10 小时，资源 R 将在 B 主机上持续运行 10 小时，10 小时后，

资源的限制条件会自动消失，资源 R 可能会从 B 主机回迁到 A 主机。

例如：将资源 vip 迁移到 host2 节点命令如下：

```
pcs resource move vip host2
```

```
[root@host1 ~]# pcs resource status
 vip    (ocf::heartbeat:IPaddr_6):      Started host1
 dummy  (ocf::heartbeat:Dummy): Started host2
[root@host1 ~]# pcs resource move vip host2
[root@host1 ~]# pcs resource status
 vip    (ocf::heartbeat:IPaddr_6):      Started host2
 dummy  (ocf::heartbeat:Dummy): Started host1
[root@host1 ~]#
```

清除迁移所设置的规则：

```
pcs resource clear <resource id> [node] [--expired] [--wait[=n]]
```

注：

- 其中<resource id>为资源名称；
- [node]为节点名称
- [--expired] 清除迁移有效期

3.2.7 删除资源

3.2.7.1 删除资源

删除资源命令：

```
pcs resource delete <resource id|group id|bundle id|clone id>
```

注：<resource id|group id|bundle id|clone id>可以是普通资源名称、组资源名称和克隆资源名称，此处会将组资源全部删除（包括组资源内的子资源）。

例如：在集群中删除 vip 资源命令：`pcs resource delete vip`

3.2.7.2 解除组资源

解除组资源命令：

```
pcs resource ungroup <group id> [resource id] ... [resource id] [--wait[=n]]
```

注：如果只写组资源名称，不写资源名称，将组资源内所有资源解除组关系，如果指定资源名称，则将指定资源解除组关系。

例如：将组资源 group1 内所有组资源解除组关系命令及运行结果如下所示：

```
pcs resource ungroup group1
```

```
[root@host1 ~]# pcs resource status
 Resource Group: group1
     vip        (ocf::heartbeat:IPaddr_6):      Started host1
     dummy      (ocf::heartbeat:Dummy): Started host1
[root@host1 ~]# pcs resource ungroup group1
[root@host1 ~]# pcs resource status
 vip    (ocf::heartbeat:IPaddr_6):      Started host1
 dummy  (ocf::heartbeat:Dummy): Started host2
[root@host1 ~]#
```

将组资源 group1 中 vip 资源解除组关系命令及运行结果如下所示：

pcs resource ungroup group1 vip

```
[root@host1 ~]# pcs resource status
 Resource Group: group1
     dummy      (ocf::heartbeat:Dummy): Started host1
     vip        (ocf::heartbeat:IPaddr_6):      Started host1
[root@host1 ~]# pcs resource ungroup group1 vip
[root@host1 ~]# pcs resource status
 Resource Group: group1
     dummy      (ocf::heartbeat:Dummy): Started host1
 vip    (ocf::heartbeat:IPaddr_6):      Started host2
[root@host1 ~]#
```

3.2.7.3 取消克隆资源

取消克隆资源命令：

pcs resource unclone <resource id | group id> [--wait[=n]]

例如：取消克隆 vip 资源命令及运行结果如下：

pcs resource unclone vip

```
[root@host1 ~]# pcs resource clone vip
[root@host1 ~]# pcs resource status
 Resource Group: group1
     dummy      (ocf::heartbeat:Dummy): Started host1
 Clone Set: vip-clone [vip]
     Started: [ host1 host2 ]
[root@host1 ~]# pcs resource unclone vip
[root@host1 ~]# pcs resource status
 Resource Group: group1
     dummy      (ocf::heartbeat:Dummy): Started host1
 vip    (ocf::heartbeat:IPaddr_6):      Started host2
[root@host1 ~]#
```

3.2.8 设置资源关系

资源关系即为目标资源设定限制条件，资源的限制条件分为三种：资源位置、资源协同和资源顺序。

以下命令显示资源所有限制条件：

pcs constraint [list|show] [--full] [--all]

3.2.8.1 资源位置

资源位置是设置集群中的节点对于该资源的运行级别，由此确定启动或者切

换时资源在哪个节点上运行，运行级别按照从高到低的顺序依次为：Master Node、Slave 1、Slave 2……。

设置资源位置命令：

pcs constraint location add <id> <resource> <node> <score> [resource-discovery=<option>]

对应于图形设置，20000 为 Master node，16000 为 Slave 1,之后进行 1000 的递减。

3.2.8.2 资源协同

资源协同是设置目标资源与集群中的其他资源是否运行在同一节点上，同节点资源表示该资源与目标资源必须运行在相同节点上，互斥节点资源表示该资源与目标资源不能运行在相同的节点上。

设置资源协同命令：

pcs constraint colocation add [master|slave] <source resource id> with [master|slave] <target resource id> [score] [options] [id=constraint-id]

注：设置该资源与目标资源是否必须运行在相同节点上或不能运行在相同节点上，当 score 为正值时表示必须运行在同一节点上，负值表示不能运行在相同节点上。

3.2.8.3 资源顺序

资源顺序是设置目标资源与集群中的其他资源启动时的先后顺序，前置资源是指目标资源运行之前，该资源必须已经运行；后置资源是指目标资源运行之后，该资源才能运行。

设置资源顺序命令如下：

pcs constraint order [action] <resource id> then [action] <resource id> [options]

注： <resource id> then [action] <resource id>为某一资源必须运行在某一资源前。

例如：test2 资源运行前，test1 资源必须已经运行命令如下：

pcs constraint order test1 then test2

3.2.8.4 资源规则

对于更复杂的位置约束，使用 rule 规则来确定资源的位置。

3.2.8.4.1 查看规则

已定义的规则使用如下命令查看：

```
pcs constraint location show --full
```

此命令会列出集群所有的限制条件，查看 rule 相关内容即可。

3.2.8.4.2 创建规则

为资源创建一个新规则的命令如下：

```
pcs constraint location <resource> rule [score=score] expression
```

其中：

<resource>是添加规则的资源名称；

score 是如果满足条件则资源在节点的分值；

expression 表达式，表达式可以是以下的形式：

- defined|not_defined <node attribute>
- <node attribute> lt|gt|lte|gte|eq|ne [string|integer|number|version] <value>
- <expression> and|or <expression>

为资源 testrule 创建规则示例如下：

```
pcs constraint location testrule rule score=INFINITY defined pingd
pcs constraint location testrule rule score=10000 pingd gt 9000
```

3.2.8.4.3 删除规则

删除已存在的规则命令如下：

```
pcs constraint rule delete <rule id>
```

其中<rule id>为指定的 rule 的 id 值。

3.2.9 Tag 功能

Tag 功能可以把一系列相关的资源做统一标记处理，对拥有同一标记的资源可做一键启动，停止操作。

3.2.9.1 获取 Tag

通过以下命令可以查看集群 Tag 配置：

```
pcs tag config
```

3.2.9.2 创建 Tag

通过以下命令进行 Tag 创建

```
pcs tag create <tag id> <id> [<id>]
```

注：其中<tag id>是要创建的 tag 的 id；后续 id 为 tag 内资源名称。

3.2.9.3 删除 Tag

通过以下命令进行 Tag 删除

pcs tag delete <tag id>

注：其中<tag id>是要创建的 tag 的 id。

3.2.9.4 编辑 Tag

通过以下命令进行 Tag 编辑

pcs tag update <tag id> [add <id> [<id>]... [--before <id> | --after <id>]] [remove <id> [<id>]...]

注：其中<tag id>为指定的需更新 tag 的 id。可以在 tag 中添加、删除或移动资源 id。您可以使用--before 或--after 来指定添加的 id 相对于 tag 中已经存在的某个 id 的位置。

例如：

- 在标记 tag1 中资源 vip 后添加 Filesystem：

  pcs tag update tag1 add Filesystem --after vip

- 在标记 tag1 中删除资源 LVM

  pcs tag update tag1 remove LVM

3.2.9.5 启动 Tag 内资源

通过以下命令进行 Tag 内资源启动：

pcs resource enable <tag id>... [--wait[=n]]

注: 其中<tag id>为需停止的 Tag 的 id；如果指定了--wait，PC 将等待 Tag 启动 n 秒，如果 Tag 已启动，则返回 0；如果 Tag 尚未启动，则返 1。如果未指定“n”，则默认为 60 分钟。

3.2.9.6 停止 Tag 内资源

通过以下命令进行 Tag 内资源停止：

pcs resource disable <tag id>... [--wait[=n]]

注：其中<tag id>为需停止的 Tag 的 id；如果指定了--wait，PC 将等待 Tag 停止 n 秒，如果 Tag 已停止，则返回 0；如果 Tag 尚未停止，则返 1。如果未指定“n”，则默认为 60 分钟。

3.3 GFS 配置

当需使用 GFS 时，使用以下命令安装相关软件包：

yum --repo=ks10-ha install dlm dlm-lib gfs2-utils

GFS2 是适用于银河麒麟高可用集群软件的共享磁盘文件系统。GFS2 允许所有节点直接同时访问同一个共享存储块。集群中所有节点对等，最多支持 32 个集群节点。GFS2 通过锁管理器来控制对存储的访问。

GFS2 配置前，需要集群处于启动状态，集群中增加 fencing 资源，并启动。

软件包需求： dlm dlm-lib gfs2-utils

所有节点安装软件包，并且在系统中共享存储有统一的设备名之后，可以进行共享存储的格式化操作。以下例子假设共享存储为/dev/sda。

文件系统格式化命令：

mkfs.gfs2 -j 2 -p lock_dlm -t *clusertname:gfs /dev/sda*

-j 参数为节点个数

-t :前为集群名称

在集群内添加 GFS2 资源，需要添加一个 controld 资源和一个文件系统类型为 gfs2 的文件系统资源。将以上两个资源设置为组资源， 再将组资源设置为克隆资源。添加步骤如下：

pcs resource create dlm ocf:pacemaker:controld allow_stonith_disabled=true

pcs resource create gfs ocf:heartbeat:Filesystem device=/dev/sda directory=/gfs2 fstype=gfs2

pcs resource group add gfs2 dlm gfs

pcs resource clone gfs2

以上步骤即完成添加资源。相关资源查询结果如下：

```
Clone: gfs2-clone
 Group: gfs2
  Resource: dlm (class=ocf provider=pacemaker type=controld)
   Attributes: allow_stonith_disabled=true
   Operations: monitor interval=10s start-delay=0s timeout=20s (dlm-monitor-interval-10s)
               start interval=0s timeout=90s (dlm-start-interval-0s)
               stop interval=0s timeout=100s (dlm-stop-interval-0s)
  Resource: gfs (class=ocf provider=heartbeat type=Filesystem)
   Attributes: device=/dev/sda directory=/gfs2 fstype=gfs2
   Operations: monitor interval=20s timeout=40s (gfs-monitor-interval-20s)
               start interval=0s timeout=60s (gfs-start-interval-0s)
               stop interval=0s timeout=60s (gfs-stop-interval-0s)
```

执行 pcs resource enable gfs2-clone 启动资源。

ps -ef | grep dlm 查询是否有 dlm_controld 进程；

执行 df 命令，可以发现/dev/sda 被 mount 到/gfs2。

注记：allow_stonith_disabled=true 为无 fencing 设备的实验环境下需要添

加的参数，生成环境使用 gfs2 文件系统时，fencing 为必备资源。此参数无需添加。

3.4 DRBD 配置

当需使用 DRBD 时，使用以下命令安装相关软件包：

```
yum --repo=ks10-ha install kmod-drbd90 drbd90-utils
```

3.4.1 DRBD 环境准备

1. 在每个节点上，各自提供自己的存储设备（大小最好一致）；

2. 如果磁盘包含您已不再需要的文件系统，请使用以下命令销毁文件系统结构：

```
root # dd if=/dev/zero of=YOUR_DEVICE count=16 bs=1M
```

3. 如果集群已在使用 DRBD，请将集群置于维护模式：

```
pcs property set maintenance-mode=true
```

当集群已使用 DRBD 时，如果您跳过此步骤，当前配置中的语法错误会导致服务关闭。

3.4.2 DRBD 配置

DRBD 配置文件位于目录 /etc/ drbd.d/ 下。

1. 配置文件/etc/drbd.d/global_common.conf，它已包含一些全局预定义值。如果需要进行全局的参数设置可以在此文件中进行修改。

2. 创建文件 /etc/drbd.d/r0.res 。根据具体情况更改以下几行并保存：

```
resource r0 {                          #r0 为资源名称
  on ha1 {                             #ha1 为主机名称
    device /dev/drbd1;
    disk     /dev/vdb;                 #/dev/vdb 为磁盘
    address 172.30.30.38:7789;   #172.30.30.38:7789 为 ip 和端口
    meta-disk internal;
  }
  on ha2 {
    device /dev/drbd1;
    disk     /dev/vdb;
    address 172.30.30.39:7789;
```

```
    meta-disk internal;
  }
}
```

其中：

- resource ：为 DRBD 的资源名称，可以根据资源关联进行定义，本文使用了比较通用的的 r0；
- on：为主机名，指定了此配置语句要应用到的主机；
- device：DRBD 的设备名及其编号；
- disk：在节点间复制的原始设备，即之前准备的磁盘在每个节点上的设备名称；
- address：各个节点的 IP 地址和端口号。每个资源都需要单独的端口，资源的两个端口必须相同；

3.4.3 初始化 DRBD 资源

准备好系统并配置好 DRBD 后，请执行磁盘的首次初始化：

1. 在两个节点上，初始化元数据储存：

```
root # drbdadm create-md r0
root # drbdadm up r0
```

2. 在主节点上，启动重新同步过程：

```
root # drbdadm primary --force r0
```

3. 使用以下命令检查状态：

```
root # drbdadm status r0
```

```
r0 role:Primary
disk:UpToDate
bob role:Secondary
peer-disk:UpToDate
```

4. 在 DRBD 主节点设备上创建文件系统，例如：

```
mkfs.ext4 /dev/drbd1
```

5. 在主节点上挂载文件系统：

```
mount /dev/drbd1 /drbd
```

其中，/drbd 为挂载目录，可以行选择或创建。

3.4.4 集群中 DRBD 主备配置

DRBD 主备模式，是指两个节点不能使用同一个资源，主节点能挂载能读写设备，从节点不能挂载，也不能读写设备。配置步骤如下：

1. 创建 DRBD 资源

创建资源命令如下：

```
#pcs resource create msdrbd ocf:linbit:drbd drbd_resource=r0 op monitor interval=60s
```

其中：msdrbd 是 HA 中的资源名称，可修改；

drbd_resource 的值为 drbd 配置的资源名称。

2. 创建克隆资源

创建资源的克隆资源命令如下：

```
#pcs resource promotable msdrbd notify=true promoted-max=1 clone-max=2
```

其中，msdrbd 为第一步中定义的资源名称。

3. 启动 DRBD 资源

启动第二步创建的 DRBD 克隆资源，命令如下：

```
#pcs resource enable msdrbd-clone
```

其中，msdrbd-clone 为克隆资源的名称

4. 创建文件系统资源

为 DRBD 设置创建文件系统资源，命令如下：

```
#pcs resource create fs-drbd Filesystem device=/dev/drbd1 directory=/data fstype=ext4
```

其中：fs-drbd 为文件系统的资源名称，可以修改；

device 的值为 DRBD 配置中定义的设备名称；

directory 的值为文件系统挂载的目录，需保证该目录存在且未被使用；

fstype 的值为 DRBD 初始化时格式化文件系统的类型。

5. 设置限制条件

为了使文件系统在 DRBD 主节点上挂载，并且先启动 DRBD 资源再挂载文件系统，需设置如下限制条件：

```
#pcs constraint colocation add fs-drbd with msdrbd-clone INFINITY with-rsc-role=Master
```

#pcs constraint order promote msdrbd-clone then start fs-drbd

其中：fs-drbd 为定义的文件系统资源名称；

Msdrbd-clone 为定义的 drbd 资源的克隆资源。

3.4.5 集群中 DRBD 主主配置

DRBD 主主模式，是指两个节点能同时使用同一个资源，能同时挂载能读写设备。配置步骤如下：

1. 修改 DRBD 配置文件

在 drbd 的资源配置文件需要加入类似如下内容：

```
net {
    protocol C;
    allow-two-primaries yes;
    fencing resource-and-stonith;
    after-sb-0pri discard-zero-changes;
    after-sb-1pri discard-secondary;
    after-sb-2pri disconnect;
}
handlers {
        # Make sure the other node is confirmed
        # dead after this!
        outdate-peer "/sbin/kill-other-node.sh";
        split-brain "/usr/lib/drbd/notify-split-brain.sh root";
}
```

其中，最主要的是 allow-two-primaries yes，即允许两个节点都是主节点；其余配置项主要在出现脑裂状况时的处理方式，可以自行定义。

2. 创建 DRBD 资源

创建资源命令如下：

#pcs resource create msdrbd ocf:linbit:drbd drbd_resource=r0 op monitor interval=60s

其中：msdrbd 是 HA 中的资源名称，可修改；

drbd_resource 的值为 drbd 配置的资源名称。

3. 创建克隆资源

创建资源的克隆资源命令如下：

#pcs resource promotable msdrbd notify=true promoted-max=2 clone-max=2

其中，msdrbd 为第一步中定义的资源名称。这里 promoted-max 设置成 2 就会启用双主模式。

4. 启动 DRBD 资源

启动第二步创建的 DRBD 克隆资源，命令如下：

#pcs resource enable msdrbd-clone

5. 创建 controld 的克隆资源

#pcs resource create dlm ocf:pacemaker:controld allow_stonith_disabled=true clone

如果集群首选项 stonith-enable 为 false，则需要设置 allow_stonith_disabled=true，否则不需要设置。

6. 格式化 gfs 文件系统

mkfs.gfs2 -p lock_dlm -t hacluster:gfs2 -j 2 /dev/drbd1

7. 创建 GFS 文件系统资源

#pcs resource create gfs ocf:heartbeat:Filesystem device=/dev/drbd1 directory=/drbd fstype=gfs2 clone

注：关于 dlm 和 gfs 资源相关具体配置请参见 GFS 配置。

8. 设置限制条件

GFS 文件系统资源需要在 dlm 和 drbd 资源之后启动，命令如下：

pcs constraint colocation add gfs-clone with msdrbd-clone INFINITY

pcs constraint order promote msdrbd-clone then start gfs-clone

pcs constraint colocation add gfs-clone with dlm-clone INFINITY

pcs constraint order dlm-clone then gfs-clone

3.5 节点管理

集群创建时可以加入集群节点。在已经运行的集群中，可以进行节点的添加、删除操作。

显示集群中节点状态命令：

crm_node -l

3.5.1 添加节点

在已运行的集群中添加新的节点需要指定节点名称和地址。节点名称为’pcs

host auth’命令提供的节点名称。

在添加节点之前，首先执行认证命令：

[root@server1 ~]# pcs host auth 节点名称

添加节点命令:

pcs cluster node add <node name> [addr=<node address>]... [watchdog=<watchdog path>] [device=<SBD device path>]... [--start [--wait[=<n>]]] [--enable] [--no-watchdog-validation]

在执行上述命令时， addr 参数需要和集群中网络心跳保持相同的个数和次序。

注：此小节仅针对向已创建集群中添加节点。

3.5.2 删除节点

关闭指定节点，并将它们移出集群。

pcs cluster node delete <node name> [<node name>]...

pcs cluster node remove <node name> [<node name>]...

例如：将集群内 host1 节点关闭并移出集群命令及运行结果如下所示：

pcs cluster delete host1

```
[root@host2 ~]# pcs cluster node delete host1
Destroying cluster on hosts: 'host1'...
host1: Successfully destroyed cluster
Sending updated corosync.conf to nodes...
host2: Succeeded
host2: Corosync configuration reloaded
[root@host2 ~]#
```

3.5.3 节点启动/停止

集群里节点启动或停止是指启动或停止节点上的高可用服务。

启动命令：

pcs cluster start

节点启动后，可使用 pcs status 查询节点上的服务状态。

停止命令

pcs cluster stop

如果此节点为集群中运行的最后一个节点，或停止操作会导致集群丢失 quorum，在这种情况下若仍想要进行节点停止操作，需要在命令最后添加--force 参数。命令格式为：

pcs cluster stop 节点名称 --force

3.5.4 节点启用/备用

集群可通过节点备用的方法， 使得资源不在此节点上运行。节点备用时，依然参与心跳的投票。

节点备用命令：

pcs node standby [--all | <node>...] [--wait[=n]]

以上命令将指定节点设置为 standby 状态。如果未指定节点，把运行此命令的节点设置为 standby 状态。

节点启用命令：

pcs node unstandby [--all | <node>...] [--wait[=n]]

以上命令取消节点的 standby 状态。如果未指定节点，则取消运行此命令的节点的 standby 状态。取消之后，节点可运行资源。

如果指定--wait 参数，pcs 将等待 n 秒执行 standby 或者 unstandby 操作。如果 n 未指定，则默认等待 60 秒。

3.6 心跳管理

集群创建时可以对心跳进行设置，在已经创建的集群中，可以进行心跳的添加、删除和编辑操作。

3.6.1 向现有集群中添加心跳

向现有集群中添加一路心跳，命令如下：

pcs cluster link add <node_name>=<node_address>... [options <link options>]

<node_name>为集群中节点的名称，<node_address>要添加的节点的心跳 ip，<link options> 为添加心跳时相关的一些心跳配置信息。在添加心跳的时候，集群中所有节点的集群名称和 ip 信息都需要填写，不可以只填写部分心跳节点的心跳信息。在添加集群心跳的时候，也可以配置一些心跳的配置选项，常见的<link options>有 link_priority, linknumber, mcastport, ping_interval, ping_precision, ping_timeout, pong_count, transport。

添加一路心跳示例命令如下：

pcs cluster link add node1=10.0.5.11 node2=10.0.5.12 node3=10.0.5.31 options linknumber=5

在本例中，添加了一路编号为 5 的网络心跳，其中 node1 节点上心跳 ip 为

10.0.5.11，node2 节点上心跳 ip 为 10.0.5.12，node3 节点上心跳 ip 为 10.0.5.31。

3.6.2 删除集群心跳

删除心跳命令如下：

```
pcs cluster link delete <linknumber> [<linknumber>]
```

或

```
pcs cluster link remove <linknumber> [<linknumber>]
```

<linknumber> 是心跳的编号 ID，具体可以通过 corosync-cfgtool -s 命令看到要删除的心跳的编号 ID。删除心跳的时候可以指定多个要删除的心跳的编号 ID。需要注意，删除心跳需要至少保留至少一路心跳，不可以将所有的心跳删除。

删除编号为 5 的一路网络心跳示例命令如下：

```
pcs cluster link delete 5
```

在添加和删除心跳时有如下注意点：

- 在添加心跳时，必须为每个节点指定一个心跳 ip 地址；
- 只有在 knet 传输协议时，才能添加和删除心跳；
- 需要保证在任何时候，集群中至少都需要保留有一个网络心跳被定义；
- 集群中的最多添加的心跳数量为 8 个，编号为 0-7。当添加心跳而不指定链接号时，将使用可用的最低心跳编号。

3.6.3 编辑心跳

编辑心跳的命令如下：

pcs cluster link update <linknumber> [<node_name>=<node_address>...] [options <link options>]

<linknumber> 是心跳的编号 ID，<node_name>为集群中节点的名称，<node_address>要添加的节点的心跳 ip。心跳协议为 transport 时，可修改的<link options>有 link_priority, mcastport, ping_interval, ping_precision,ping_timeout, pong_count, transport (udp or sctp)。心跳协议为 udp 或 udpu 时，可修改的选项有 bindnetaddr, broadcast, mcastaddr, mcastport, ttl。这一命令不要求指定所有节点地址和选项。只需要指定要更改的心跳 ip 和心跳选项即可。同时在一般情况下，心跳选项按照默认即可，不建议做特殊修改。

在集群启动的时候是无法执行编辑心跳命令，需要先将集群关闭才可以编辑心跳。示例步骤如下：

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

pcs cluster link update 1 node1=10.0.5.11 node3=10.0.5.31 options link_priority=11

要删除某个选项，可以使用选项【option】=null 实现。

pcs cluster start --all

在本例中，将集群心跳链路 1 中 node1 和 node3 节点的心跳 ip 进行更新，同时将心跳选项 link_priority 更新为 11。

3.6.4 磁盘心跳

3.6.4.1 磁盘心跳功能

在高可用集群中，将磁盘心跳技术作为一种维护集群通信的手段，通过定期检测磁盘的心跳信号来确认磁盘是否正常工作；通过磁盘心跳技术，在网络故障导致网络心跳中断或异常时，维持集群间通信，无需立即杀死主机，保持系统的正常运行从而减少系统故障和停机时间，极大提高存储系统的可靠性。

3.6.4.2 磁盘心跳环境准备

配置磁盘心跳，在各个节点之间需要准备一块共享磁盘。共享磁盘用于提供磁盘心跳通信的介质，以维持集群之间正常的通信。这里共享磁盘也可以使用一块共享盘的一个分区或者是上面的一个逻辑卷。

3.6.4.3 磁盘心跳配置

（1）修改配置文件内容

修改集群中各个节点的/etc/corosync/corosync.conf 文件中的内容，增加磁盘心跳配置信息。具体修改内容如下：

```
[root@host224 ~]# cat /etc/corosync/corosync.conf
totem {
    version: 2
    cluster_name: test1
    transport: knet
    crypto_cipher: aes256
    crypto_hash: sha256
    cluster_uuid: 850d3169ab804230b4032c26d1d8481b
}

nodelist {
    node {
        ring0_addr: 192.168.122.224
        ring7_addr: disk
        name: host224
        nodeid: 1
    }

    node {
        ring0_addr: 192.168.122.225
        ring7_addr: disk
        name: host225
        nodeid: 2
    }
}

quorum {
    provider: corosync_votequorum
    two_node: 1
}

logging {
    to_logfile: yes
    logfile: /var/log/cluster/corosync.log
    to_syslog: yes
    timestamp: on
}

disk {
    path: /dev/sdd
    label: qwer123
}
```

在 nodelist 部分，针对每个 node，需要添加一个字段 ring7_addr: disk。同时需要新增 disk 节，其中 path 字段指的是具体的共享磁盘的路径，label 为为共享磁盘设置的特定的标志字段信息，label 字段内容只能包含字母和数字。

如果共享磁盘在集群中各个节点的盘符信息是一致的，则也可以在修改一个节点以后，通过 pcs cluster sync 命令来将当前配置修改同步到其他的节点。

若是初次配置需要执行命令 #echo "qwer123" > /dev/sdd 将 label 写入磁盘开头。

（2）重载 corosync 配置信息

磁盘心跳配置信息修改以后，通过如下命令使配置生效：

corosync-cfgtool -R

通过 corosync-cfgtool -s 命令可以看到新增了一路 link 7，即为磁盘心跳一路

的心跳信息。

```
[root@host224 ~]# corosync-cfgtool -s
Local node ID 1, transport knet
LINK ID 0 udp
        addr    = 192.168.122.224
        status:
                nodeid:          1:     localhost
                nodeid:          2:     connected
LINK ID 7 udp
        addr    = disk
        status:
                nodeid:          1:     localhost
                nodeid:          2:     connected
```

3.6.4.4 磁盘心跳删除

磁盘心跳的 link ID 为 7，因此删除磁盘心跳命令如下：

pcs cluster link remove 7

3.6.4.5 磁盘心跳编辑

在需要编辑修改磁盘心跳配置时，目前并不支持直接编辑生效，需要先使用如下命令删除磁盘心跳一路的信息。

pcs cluster link remove 7

然后依次将配置的修改内容添加到/etc/corosync/corosync.conf 文件中。内容修改以后，再执行如下命令进行配置同步和生效：

pcs cluster sync

corosync-cfgtool -R

最后执行 corosync-cfgtool -s 查看磁盘心跳是否配置生效

3.7 日志分析

通过执行命令对日志进行分析，日志分析命令：

kylin-ha-log-analyzer

该命令可输出未被分割的 pacemaker.log 错误日志到当前路径；需进一步对日志进行分析，可参照下表。

命令	解释
kylin-ha-log-analyzer	默认输出未被分割的 pacemaker.log 中的错误日志 ha_analyzer.log 到当前路径；多个参数可结合使用
kylin-ha-log-analyzer -f	-f：日志起始时间

年-月-日 时:分:秒 -t 年-月-日 时:分:秒	-t：日志结束时间，-t 可以省略，省略后输出-f 设定时间至今的日志；
kylin-ha-log-analyzer --level event/error/fence	默认输出错误日志 1.kylin-ha-log-analyzer -level event：输出全量操作日志 2.kylin-ha-log-analyzer -level error：输出错误日志 3.kylin-ha-log-analyzer -level fence：输出 fence 相关日志
kylin-ha-log-analyzer -o 目录	日志输出到指定目录

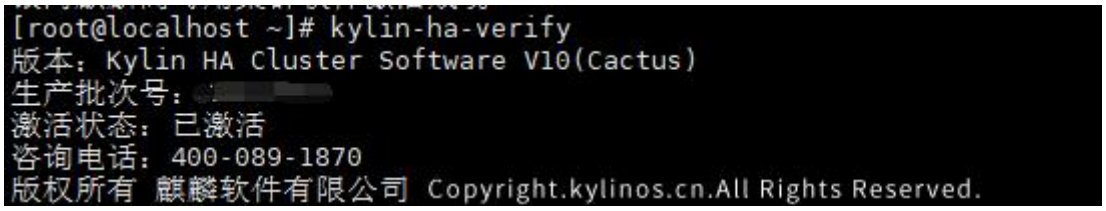
4 激活

4.1 查看激活状态

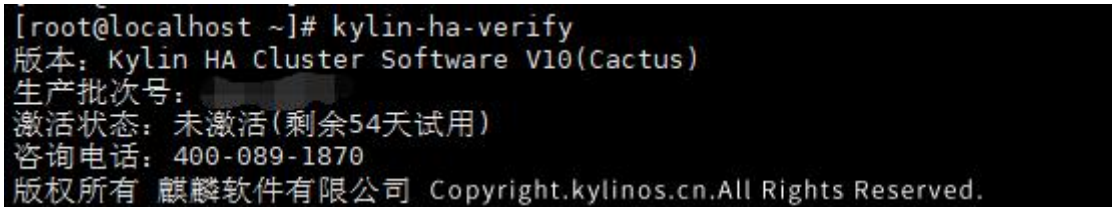
查看激活状态命令：

kylin-ha-verify

已激活状态如下图所示：



未激活状态如下图所示：

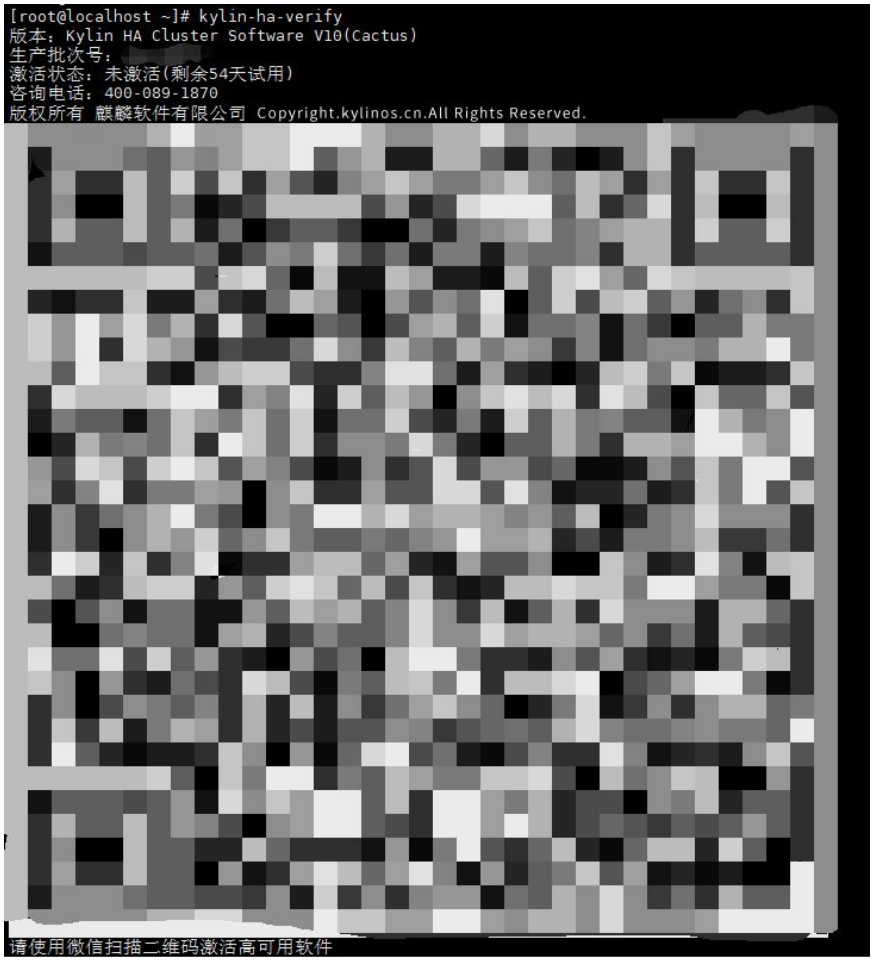


4.2 二维码激活

使用二维码激活命令如下：

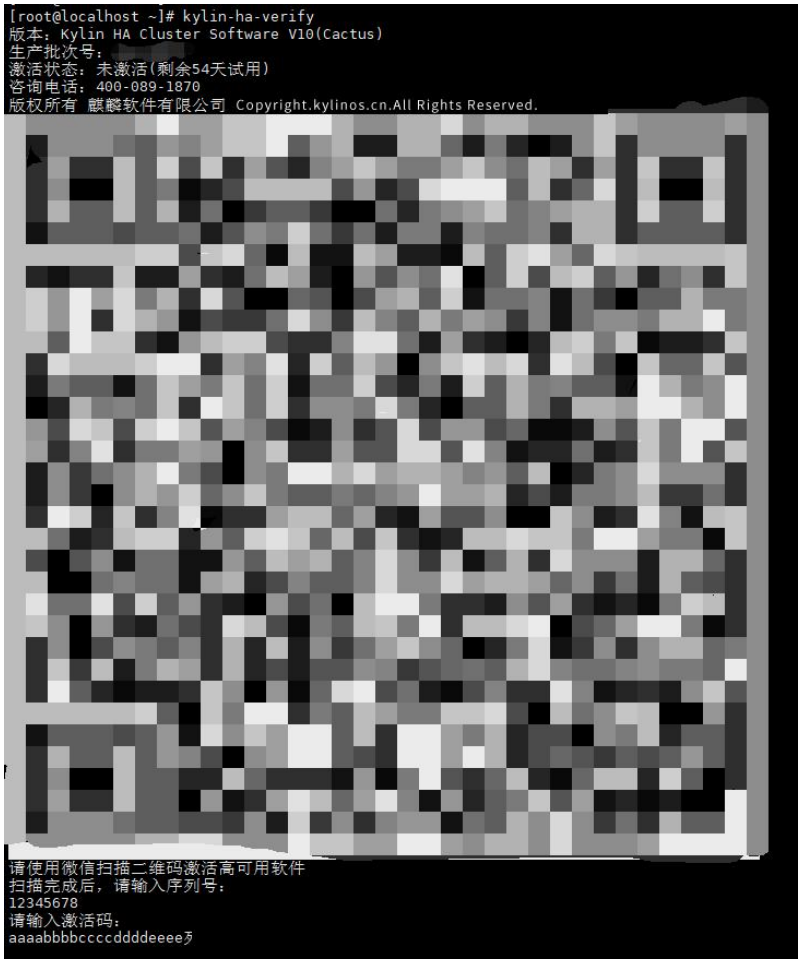
kylin-ha-verify

当服务器处于联网状态时，二维码激活显示如下图所示：



使用微信扫描上方所显示二维码，执行激活操作。在微信上激活后，在上图所示界面中输入回车激活麒麟高可用集群软件。

当服务器处于断网状态时，二维码激活显示如下图所示：



使用微信扫描上方所显示二维码，执行激活操作，激活后会生成激活码。在微信上激活后，在上图所示界面中输入服务序列号后回车，输入生成的激活码回车完成麒麟高可用集群软件激活。

4.3 url 激活

使用 url 激活命令如下：

```
kylin-ha-verify -u
```

当服务器处于联网状态时，在手机浏览器的地址栏输入命令输出的 url，执行激活操作。在微信上激活后，输入回车激活麒麟高可用集群软件。

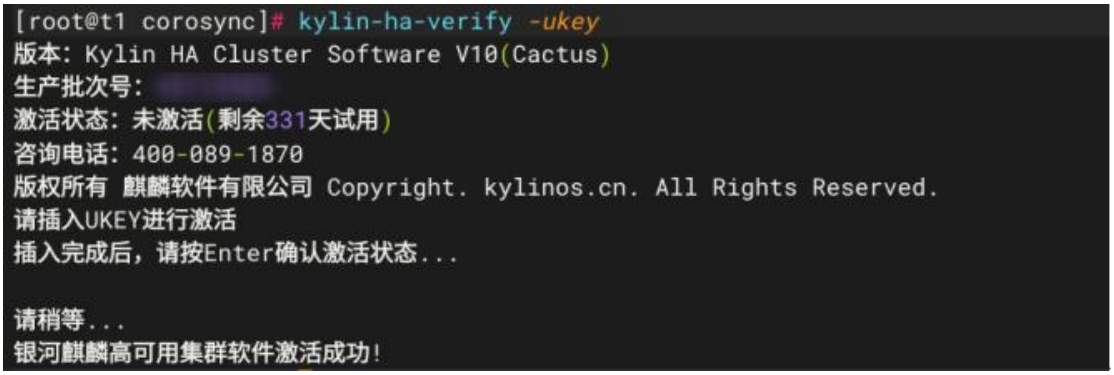
当服务器处于断网状态时，在手机浏览器的地址栏输入命令输出的 url，执行激活操作。激活后会生成激活码，在微信上激活后，输入服务序列号后回车，输入生成的激活码回车完成麒麟高可用集群软件激活。

4.4 ukey 激活

系统插入 ukey，然后使用如下命令进行 ukey 激活：

```
kylin-ha-verify -ukey
```

按照提示，“Enter”确认，等待激活结果，显示“银河麒麟高可用集群软件激活成功”则完成 ukey 激活。



4.5 场地授权

场地授权命令：

kylin-ha-verify *PATH*/.kyinfo *PATH*/LICENSE

其中 PATH 为.kyinfo 和 LICENSE 文件所在目录。

4.6 二维码续保

前期激活使用二维码、url 或者 ukey 方式的情况下，可使用二维码续保命令，如下：

kylin-ha-verify

当服务器处于联网状态时，使用微信扫描命令输出的二维码，执行续保操作，在微信上续保后，输入回车续保麒麟高可用集群软件。

当服务器处于断网状态时，使用微信扫描命令输出的二维码，执行续保操作，续保后会生成激活码。在微信上续保后，在上图所示界面中输入服务序列号后回车，输入生成的激活码回车完成麒麟高可用集群软件续保。

4.7 url 续保

前期激活使用二维码或 url 方式的情况下，可使用 url 续保命令如下：

kylin-ha-verify -u

当服务器处于联网状态时，在手机浏览器的地址栏输入 url，执行续保操作，在微信上续保后，在上图所示界面中输入回车续保麒麟高可用集群软件。

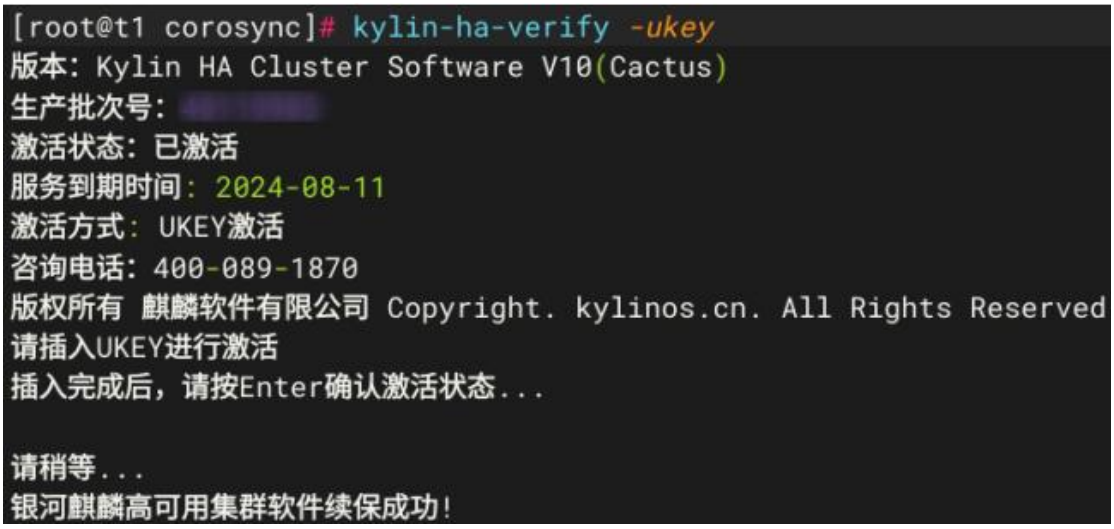
当服务器处于断网状态时，在手机浏览器的地址栏输入 url，执行续保操作，

续保后会生成激活码。在微信上续保后，在上图所示界面中输入服务序列号后回车，输入生成的激活码回车完成麒麟高可用集群软件续保。

4.8 ukey 续保

前期使用二维码、url 或者 ukey 进行激活的情况下，可以使用 ukey 进行续保，命令如下：

```
kylin-ha-verify -ukey
```



“Enter”确定，等待激活结果，显示“银河麒麟高可用集群软件续保成功”则完成续保。

4.9 场地续保

前期激活使用场地授权方式的情况下，仅接受使用场地续保命令进行续保，命令如下：

```
kylin-ha-verify    PATH/.kyinfo PATH/LICENSE
```

其中 PATH 为.kyinfo 和 LICENSE 文件所在目录。