**Klustron 多IDC使用手册**

# 需求背景

Klustron的fullsync 强同步复制技术和fullsync HA 高可用技术，确保kunlun-storage 节点组成的存储集群(storage shard) 的主节点发生故障时，Klustron可以自动及时发现故障并自动选举出新的主节点持续对外提供数据读写服务，并且不丢失、损坏用户数据。对于大多数应用场景所需的高可靠性要求以及能够提供的资源条件，这样的可靠性级别已经足够了。

然而面对在金融级高可靠性的极致要求，这仍然是不够的，因为如果一个数据中心（IDC）整体失效（比如断电，地震，火灾，水灾等）的情况下，如果一个Klustron 集群所有节点部署在这样一个IDC机房，它们就会同时消失，那么用户数据仍然会丢失，数据库服务也仍然会停止。

为了达到IDC 级别的高可用，Klustron 团队开发了数据中心(IDC) 级高可用技术。Klustron 从 1.2 版本开始，支持数据中心（IDC）高可用功能。

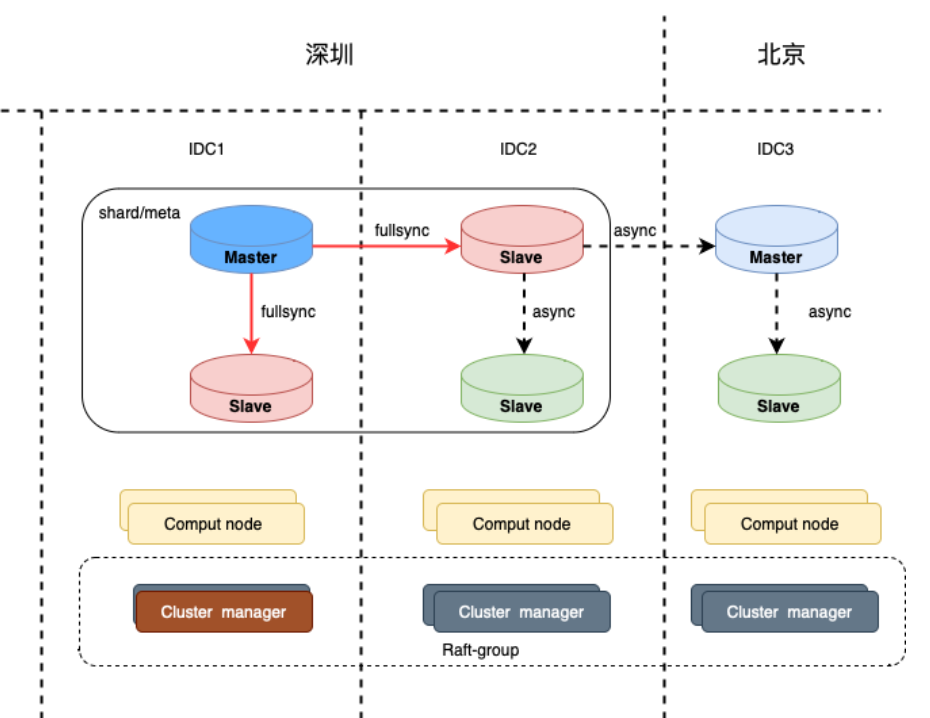
# IDC容灾架构拓扑和工作原理

## 架构拓扑

### 两地三中心架构

两地三中心是指通过同城两个数据中心（IDC），备城一个数据中心部署实现服务的跨地域 IDC级 高可用容灾。一个 Klustron 集群的每一个 shard 的拓扑如下图所示。用户可以根据需要在每一个IDC部署任意数量的计算节点(kunlun-server) 和clusterManager节点。所有计算节点都持续从元数据集群同步用户的元数据更新。

两地三中心架构实现了同城任意一个IDC灾难高可用自动恢复以及主城灾难下，备城IDC的手动恢复，Klustron 提供工具辅助用户切换。考虑到备城距离较远，网络延时较大，因此选择异步复制(async)到备城的备节点。

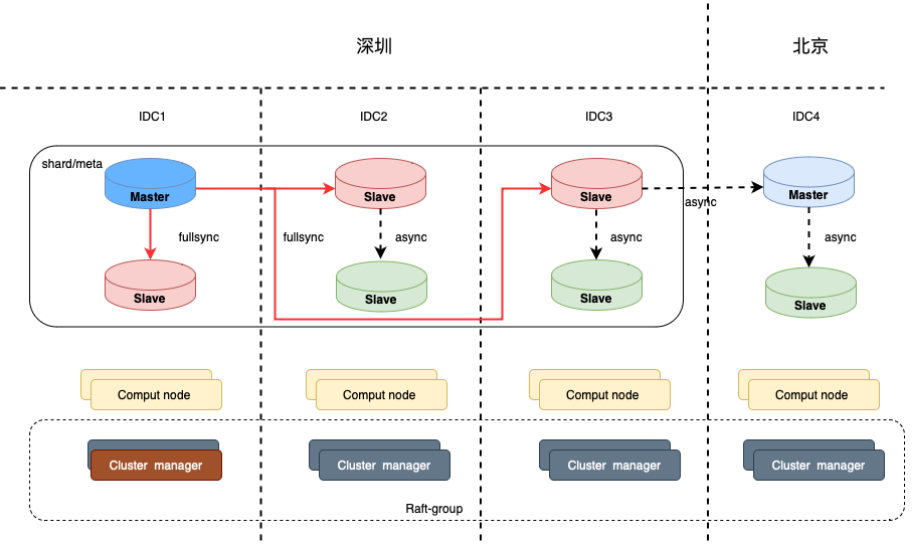


### 两地四中心架构

在两地三中心、两地四中心等容灾部署方案中，如果主城的主IDC故障，则Klustron需要自动切换到主城的备IDC；如果主城整体故障，则用户DBA需要手动切换到备城的IDC。这都意味着需要把一个Klustron集群的所有storage shard以及元数据集群的主节点都整体切换到 另一个数据中心。

应用软件平时连接主城的主IDC的计算节点，为了性能最优，不应该使用任何 主城备IDC或者备城IDC 中运行的计算节点。如果切换到备城IDC，则主城主IDC的计算节点与 新的存储集群主节点延时过大不可以继续使用。因此应用软件需要负责切换到连接与新主节点在同一个IDC中的那些计算节点。

计算节点的读写分离功能会根据网络延时信息自动选择合适的备节点，所以不会选择到备IDC的备节点。



## 工作原理

在两地三中心、两地四中心等容灾部署方案中，如果主城的主IDC故障，则Klustron需要自动切换到主城的备IDC；如果主城整体故障，则用户DBA需要手动切换到备城的IDC。这都意味着需要把一个Klustron集群的所有storage shard以及元数据集群的主节点都整体切换到 另一个数据中心。每个shard 在主IDC中需要有一主一备两个节点并且它们做fullsync复制，这样可以确保如果某个shard的主节点故障但是主IDC正常的情况下，Klustron会选举该shard在主IDC的fullsync备节点成为新的主节点，这样应用软件不需要切换计算节点就可以持续高效工作

每个shard 在主城的每个备IDC都有至少两个kunlun-storage 备节点，并且其中一个从主城主IDC 的主节点做强同步复制（下文称之为 fullsync 备机或者候选主节点） ，以确保任何一个备IDC都可以切换为主IDC 这要求每个shard在这个备IDC都有候选节点可以立刻成为新的主节点；同时，备IDC中每个shard的另一个备节点从 本shard在同IDC 的 候选主节点 做异步复制。

如果主IDC整体故障，Klustron能够自动发现这种故障情况，并为每个shard 自动选举 该shard在主城某个备IDC的 候选主节点成为该shard新的主节点，这个备IDC也就升级为新的主IDC。新的主IDC中每个shard的候选主节点升级为主节点后，它与同IDC内本shard的另一个备机节点的异步复制也会升级为fullsync复制，并且该shard在其他（主城或者备城）备IDC的候选主节点都改为从这个新的主节点复制数据更新，原来是fullsync 复制的依然是fullsync复制，原来是异步复制(async) 的依然是异步复制。这样的IDC切换可以保持数据一致性，不会丢失、损坏数据。

如果主城所有IDC全部故障，则用户可以使用Klustron 的功能接口手动切换到了备城的IDC。由于物理距离遥远，网络包传输时耗太大，每个shard在备城IDC的 候选主节点只能从该shard在主IDC的主节点做异步复制，所以这样的IDC切换还意味着用户有可能丢失灾难发生之前的少量最新的数据更新 和/或 元数据更新（如果故障发生前刚好在执行DDL的话），也就是意味着切换后集群可能因为元数据和数据不匹配而无法正常工作，此时就需要DBA人工介入修复数据。

IDC切换发生后，应用软件务必也切换到使用新的主 IDC内的计算节点，因为老的主IDC的计算节点也都不可用了。这个切换需要用户自己实现，比如使用负载均衡等工具

可以看出两地三中心和两地四中心唯一的区别就是增加了一个主城的备IDC。用户可以按照需要，增加更多的备IDC在主城 和/或 备城，只要为每个shard 复制相同的备IDC 拓扑结构即可。并且用户还可以增加更多的备城，在其中设置一个或者多个IDC。当然，只要是升级备城的IDC 成为 主IDC，那么都需要手动操作。

# Klustron版本&集群机器配置

|  |  |
| --- | --- |
| Klustron版本 | 1.2.1 |

|  |  |  |
| --- | --- | --- |
| 机器IP | 机器配置 | 所属IDC |
| 192.168.0.174 | CentOS7.9 32C 128G 1T\*2nvme ssd | 同城主：sz-dc1 |
| 192.168.0.176 | CentOS7.9 32C 128G 1T\*2nvme ssd |
| 192.168.0.177 | CentOS7.9 32C 128G 1T\*2nvme ssd |
| 192.168.0.184 | CentOS7.9 32C 128G 1T\*2nvme ssd | 同城备：sz-dc2 |
| 192.168.0.185 | CentOS7.9 32C 128G 1T\*2nvme ssd |
| 192.168.0.186 | CentOS7.9 32C 128G 1T\*2nvme ssd |
| 172.16.128.15 | CentOS7.9 32C 128G 1T\*2nvme ssd | 备城：bj-dc1 |
| 172.16.128.16 | CentOS7.9 32C 128G 1T\*2nvme ssd |
| 172.16.128.17 | CentOS7.9 32C 128G 1T\*2nvme ssd |

# 第四章 集群安装部署

## 集群配置样例

配置多IDC和单IDC稍微有点区别，生成安装脚本必须使用参数—multipledc

以下是命令和配置文件详情，生产和测试环境可以直接使用，具体环境按照请阅读官方文档：http://doc.klustron.com/zh/user\_notes\_Installation\_Guide.html

python setup\_cluster\_manager.py --autostart --config=config.json --multipledc --product\_version=1.2.1 --action=install

{

"machines":[

{

"ip":"192.168.0.174",

"basedir":"/home/tanyueyun/mdcbase",

"user":"tanyueyun"

},

{

"ip":"192.168.0.176",

"basedir":"/home/tanyueyun/mdcbase",

"user":"tanyueyun"

},

{

"ip":"192.168.0.177",

"basedir":"/home/tanyueyun/mdcbase",

"user":"tanyueyun"

},

{

"ip":"192.168.0.184",

"basedir":"/home/tanyueyun/mdcbase",

"user":"tanyueyun"

},

{

"ip":"192.168.0.185",

"basedir":"/home/tanyueyun/mdcbase",

"user":"tanyueyun"

},

{

"ip":"192.168.0.186",

"basedir":"/home/tanyueyun/mdcbase",

"user":"tanyueyun"

},

{

"ip":"172.16.128.15",

"basedir":"/home/tanyueyun/mdcbase",

"user":"tanyueyun"

},

{

"ip":"172.16.128.16",

"basedir":"/home/tanyueyun/mdcbase",

"user":"tanyueyun"

},

{

"ip":"172.16.128.17",

"basedir":"/home/tanyueyun/mdcbase",

"user":"tanyueyun"

}

],

"datacenters": [

{

"name": "sz-dc1",

"province": "guangdong",

"city": "shenzhen",

"is\_primary": true

},

{

"name": "sz-dc2",

"province": "guangdong",

"city": "shenzhen"

},

{

"name": "bj-dc1",

"province": "beijing",

"city": "beijing"

}

],

"meta":{

"ha\_mode": "rbr",

"nodes":[

{

"ip":"192.168.0.174",

"innodb\_buffer\_pool\_size":"1024MB",

"port":6001

},

{

"ip":"192.168.0.176",

"innodb\_buffer\_pool\_size":"1024MB",

"port":6001

},

{

"ip":"192.168.0.184",

"innodb\_buffer\_pool\_size":"1024MB",

"port":6001

},

{

"ip":"192.168.0.185",

"innodb\_buffer\_pool\_size":"1024MB",

"port":6001

},

{

"ip":"172.16.128.15",

"innodb\_buffer\_pool\_size":"1024MB",

"port":6001

},

{

"ip":"172.16.128.16",

"innodb\_buffer\_pool\_size":"1024MB",

"port":6001

},

{

"ip":"172.16.128.17",

"innodb\_buffer\_pool\_size":"1024MB",

"port":6001

}

]

},

"cluster\_manager": {

"nodes": [

{

"ip": "192.168.0.174",

"brpc\_http\_port":58000,

"brpc\_raft\_port":58001,

"prometheus\_port\_start":59010

},

{

"ip": "192.168.0.176",

"brpc\_http\_port":58000,

"brpc\_raft\_port":58001,

"prometheus\_port\_start":59010

},

{

"ip": "192.168.0.184",

"brpc\_http\_port":58000,

"brpc\_raft\_port":58001,

"prometheus\_port\_start":59010

},

{

"ip": "192.168.0.185",

"brpc\_http\_port":58000,

"brpc\_raft\_port":58001,

"prometheus\_port\_start":59010

},

{

"ip": "172.16.128.15",

"brpc\_http\_port":58000,

"brpc\_raft\_port":58001,

"prometheus\_port\_start":59010

},

{

"ip": "172.16.128.16",

"brpc\_http\_port":58000,

"brpc\_raft\_port":58001,

"prometheus\_port\_start":59010

},

{

"ip": "172.16.128.17",

"brpc\_http\_port":58000,

"brpc\_raft\_port":58001,

"prometheus\_port\_start":59010

}

]

},

"node\_manager": {

"nodes": [

{

"ip": "192.168.0.174",

"brpc\_http\_port":58002,

"tcp\_port":58003,

"prometheus\_port\_start":58010,

"storage\_portrange": "57000-58000",

"server\_portrange": "47000-48000",

"dc": "sz-dc1"

},

{

"ip": "192.168.0.176",

"brpc\_http\_port":58002,

"tcp\_port":58003,

"prometheus\_port\_start":58010,

"storage\_portrange": "57000-58000",

"server\_portrange": "47000-48000",

"dc": "sz-dc1"

},

{

"ip": "192.168.0.177",

"brpc\_http\_port":58002,

"tcp\_port":58003,

"prometheus\_port\_start":58010,

"storage\_portrange": "57000-58000",

"server\_portrange": "47000-48000",

"dc": "sz-dc1"

},

{

"ip": "192.168.0.184",

"brpc\_http\_port":58002,

"tcp\_port":58003,

"prometheus\_port\_start":58010,

"storage\_portrange": "57000-58000",

"server\_portrange": "47000-48000",

"dc": "sz-dc2"

},

{

"ip": "192.168.0.185",

"brpc\_http\_port":58002,

"tcp\_port":58003,

"prometheus\_port\_start":58010,

"storage\_portrange": "57000-58000",

"server\_portrange": "47000-48000",

"dc": "sz-dc2"

},

{

"ip": "192.168.0.186",

"brpc\_http\_port":58002,

"tcp\_port":58003,

"prometheus\_port\_start":58010,

"storage\_portrange": "57000-58000",

"server\_portrange": "47000-48000",

"dc": "sz-dc2"

},

{

"ip": "172.16.128.15",

"brpc\_http\_port":58002,

"tcp\_port":58003,

"prometheus\_port\_start":58010,

"storage\_portrange": "57000-58000",

"server\_portrange": "47000-48000",

"dc": "bj-dc1"

},

{

"ip": "172.16.128.16",

"brpc\_http\_port":58002,

"tcp\_port":58003,

"prometheus\_port\_start":58010,

"storage\_portrange": "57000-58000",

"server\_portrange": "47000-48000",

"dc": "bj-dc1"

},

{

"ip": "172.16.128.17",

"brpc\_http\_port":58002,

"tcp\_port":58003,

"prometheus\_port\_start":58010,

"storage\_portrange": "57000-58000",

"server\_portrange": "47000-48000",

"dc": "bj-dc1"

}

]

},

"xpanel": {

"image": "registry.cn-hangzhou.aliyuncs.com/kunlundb/kunlun-xpanel:1.2.1",

"nodes": [

{

"ip": "192.168.0.125",

"port":18081

},

{

"ip": "192.168.0.127",

"port":18081

},

{

"ip": "192.168.0.128",

"port":18081

}

]

}

}

## 集群拓扑结构说明

* 在每个IDC中为每个KunlunBase集群都需要安装至少一个计算节点。用户指定的计算节点的安装个数，应该视为每个IDC要安装的个数。因为一旦做了IDC切换，那么应用软件也会改为使用新的主IDC中的计算节点。
* 在每个IDC中为每个shard至少安装两个存储节点，并且主IDC 每个shard 要有至少 1个备机做fullsync复制，主IDC中每个shard的其余备机（如有）做异步复制；
* 主城所有备IDC的 每个shard的候选主节点(master candidate) 从主IDC 中该shard的主节点做fullsync复制
* 备城IDC的每个shard的候选主节点从主IDC 对应shard的主节点做异步复制
* 所有备IDC（无论主城备城）每个shard的其余备节点从本shard在本IDC的候选主节点做异步复制

## 用户输入的参数及其处理

1. 用户输入每个IDC 的如下信息：

* 名称或者ID（在data\_centers表中的id）
* 类型（主城主IDC，主城备IDC or 备城IDC）
* 主备节点个数(>= 2) 和计算节点个数(> 0)

1. cluster\_mgr负责为每个IDC 按照拓扑结构的要求安装节点和建立主备关系

* 在本IDC内找到合适的计算机服务器来安装指定数量的存储节点和计算节点，注意安装存储节点的计算机服务器不应该在同一个机架
* 对于主城备IDC，建立 每个shard 的候选主节点到备节点的主备异步复制关系，以及候选主节点与本shard 主节点的fullsync复制关系
* 对于主IDC，建立主备fullsync复制关系
* 对于备城IDC，建立主备异步复制关系以及候选主节点到本shard主节点的异步复制关系

## 需要注意的事项

* 元数据集群和cluster\_mgr 集群，应该在每个IDC都能够有至少2个节点，否则无法单IDC运行。
* XPanel作为一个docker镜像，它如果运行在的IDC故障了，则XPanel整体不可访问了，所以需要在每个IDC都有一个XPanel docker镜像部署上去并且运行起来，它们连接同一个元数据集群。

## 同城

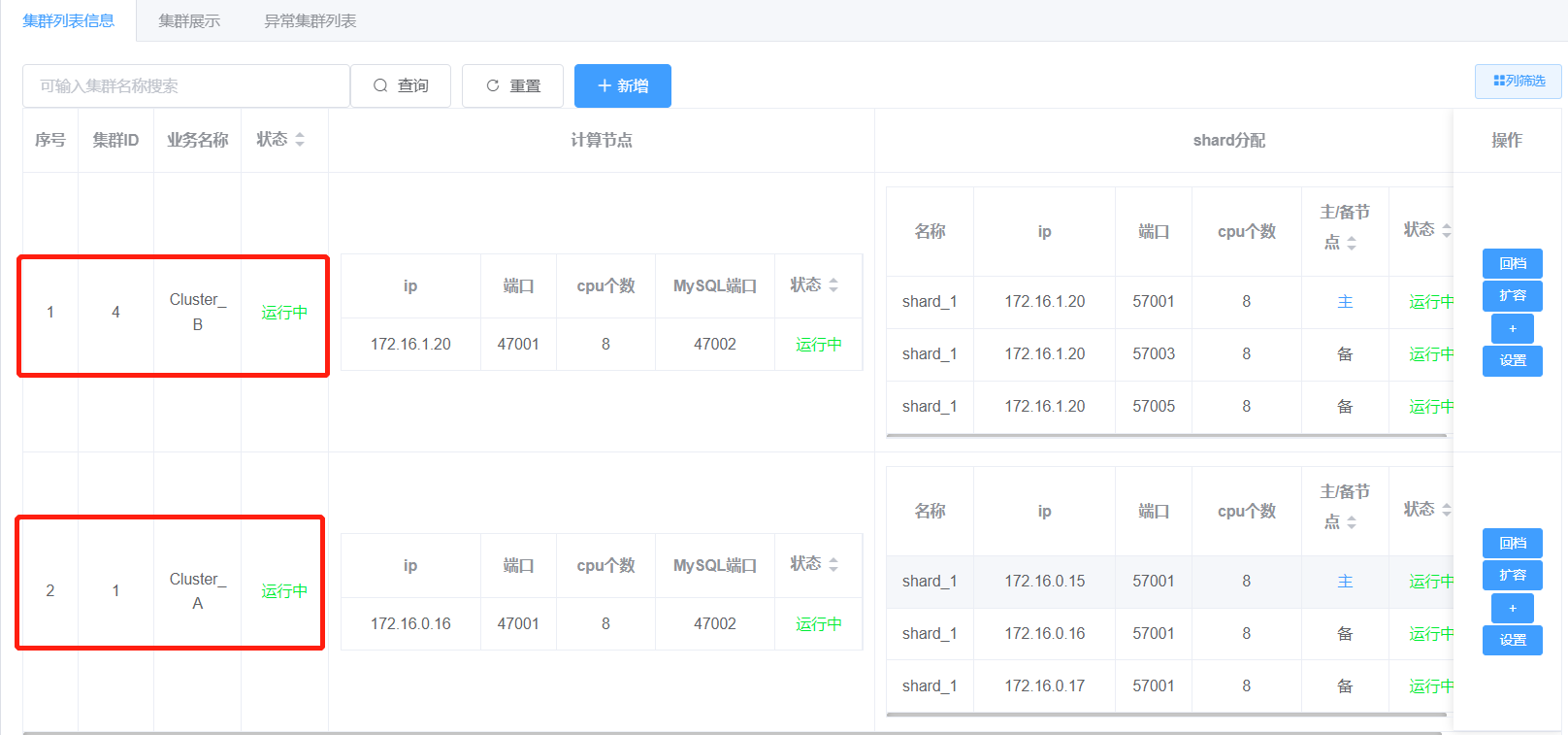
### 新增RCR

首先创建两个cluser集群分别为cluser\_A和cluser\_B,如下图所示：





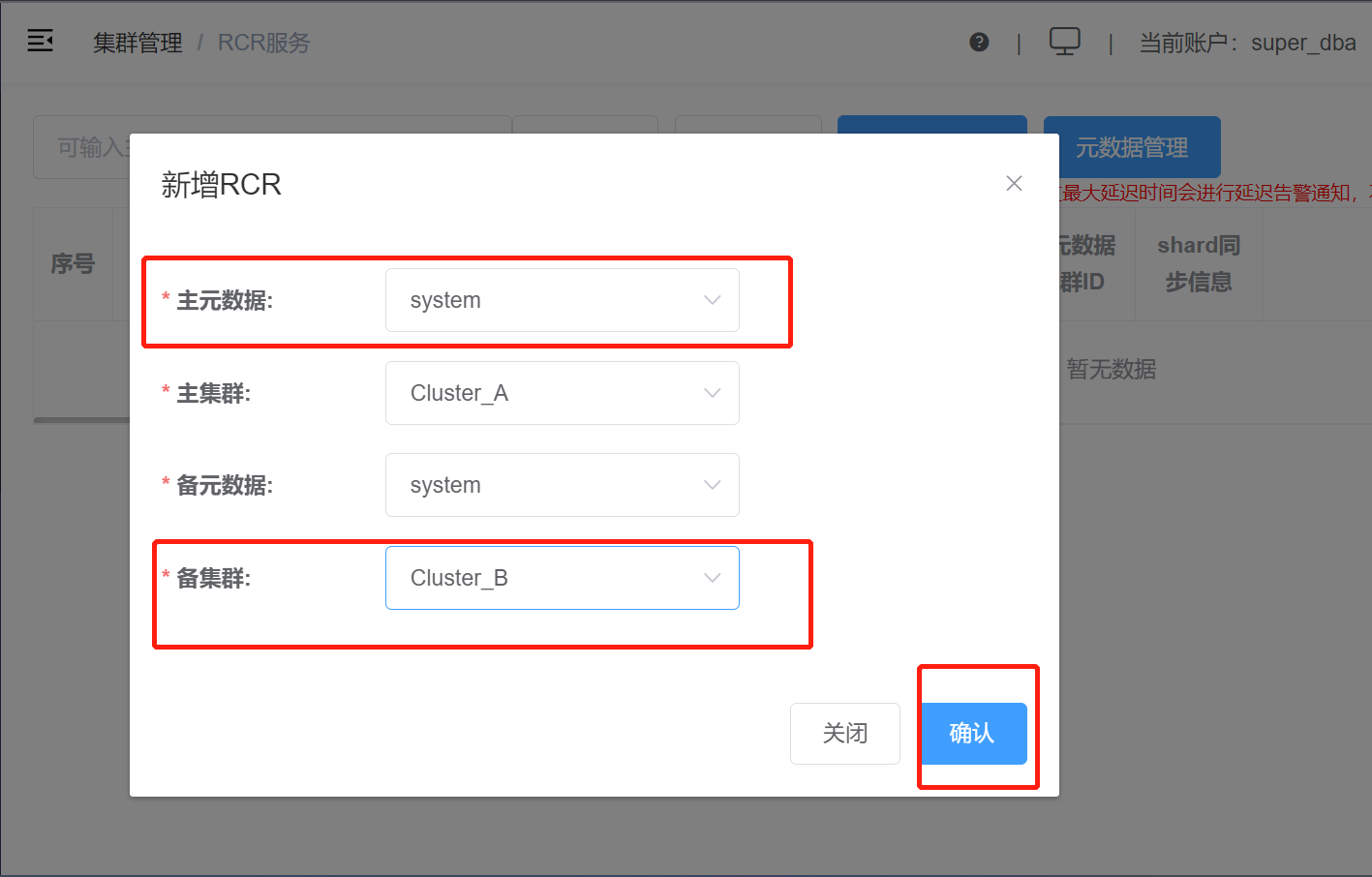
成功创建集后在集群列表中信息中显示两个集群信息如下图所示：



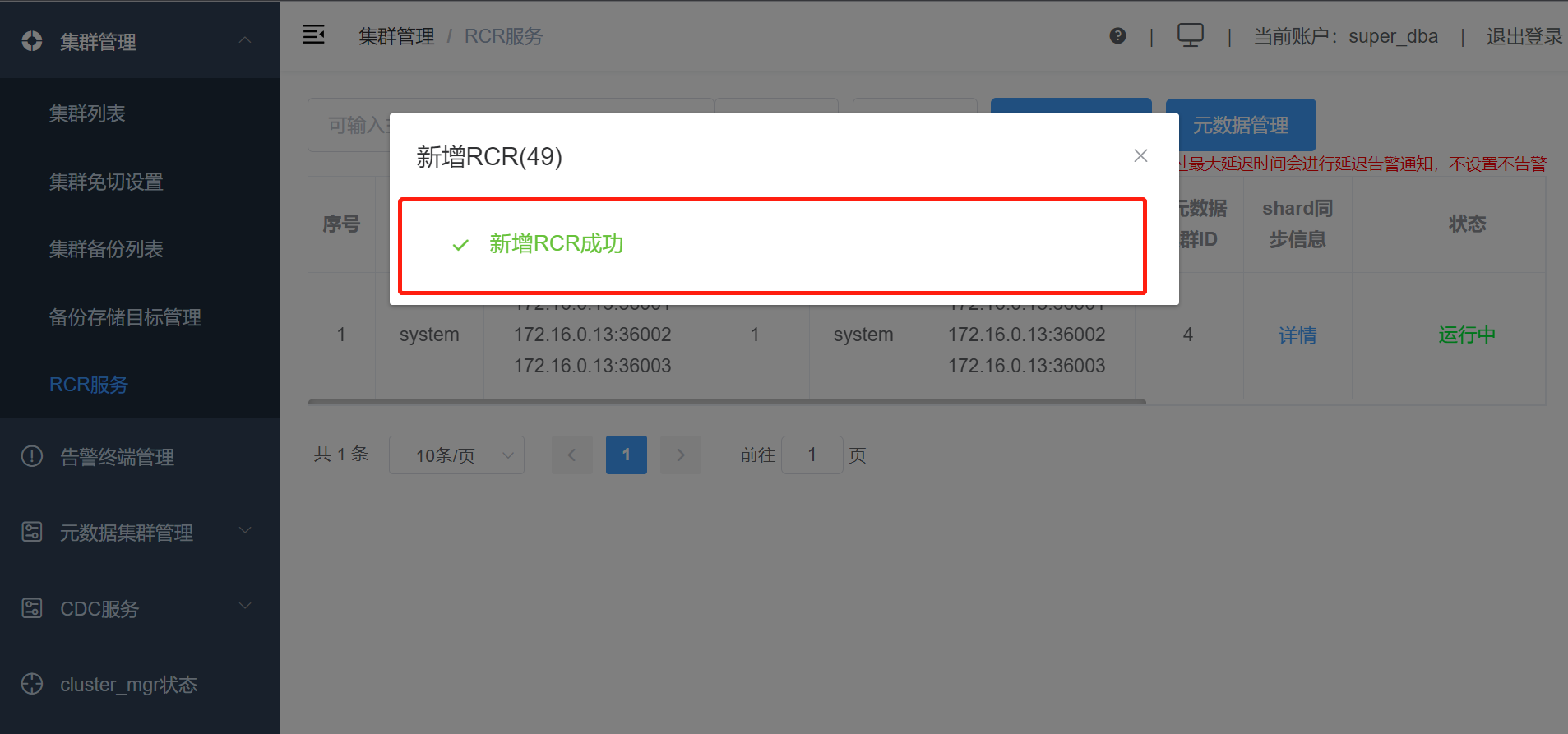
在集群管理列表中点击“RCR服务”，然后点击“+新增RCR“按钮如图所示：



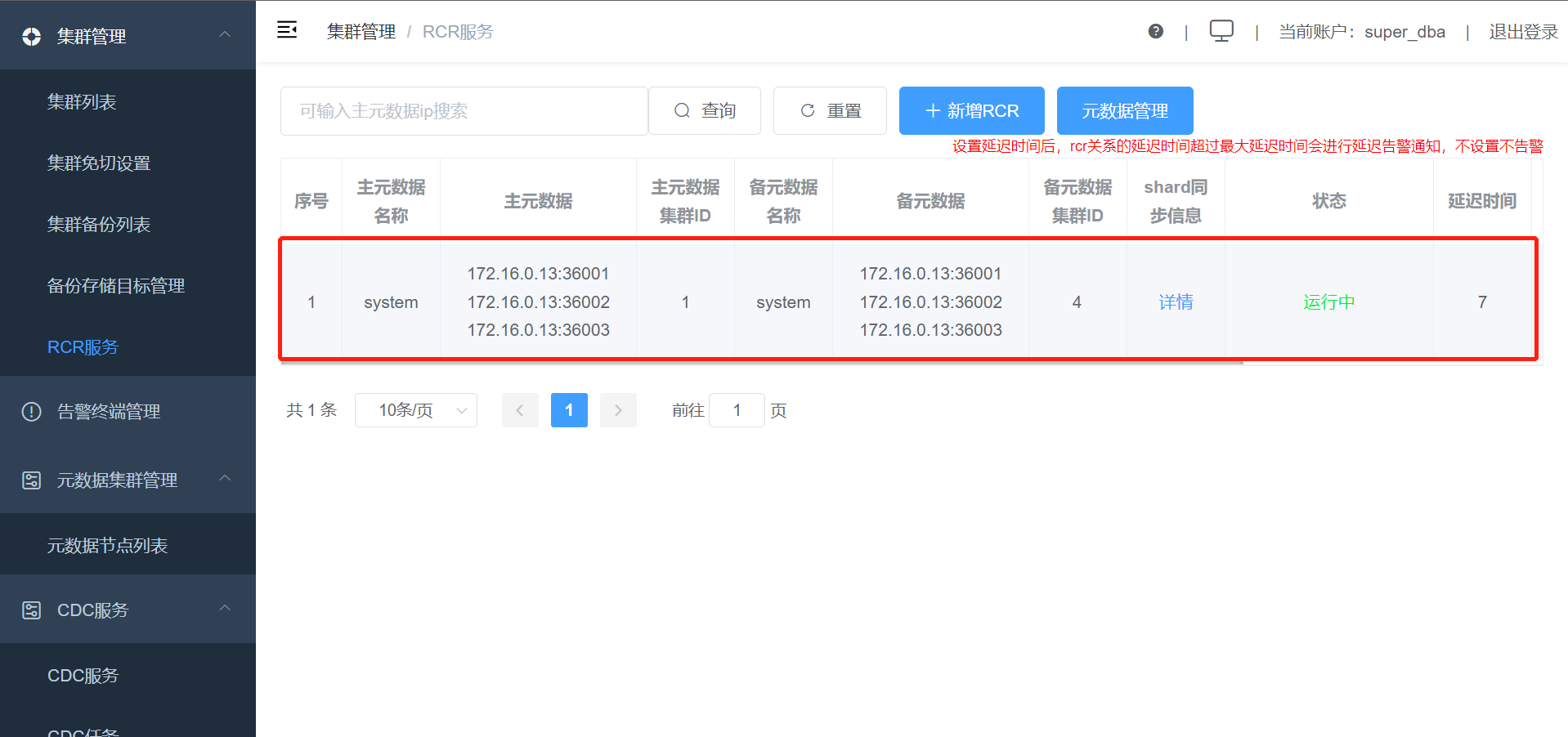
分别对应添加Cluseter\_A和Cluseter\_B元数据节点信息



元数据信息添加成功



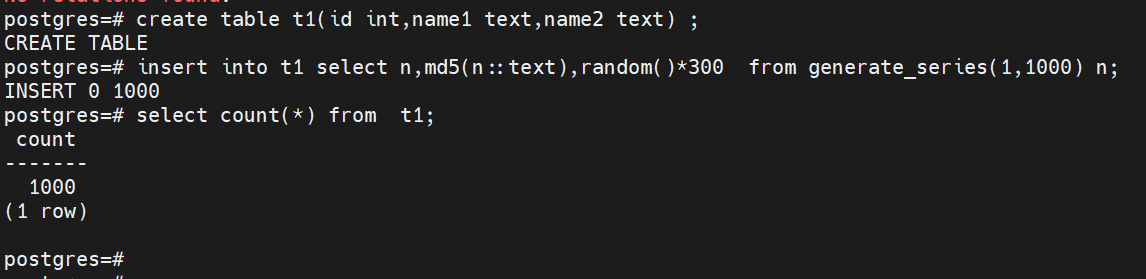
元数据添加成功后，检查RCR服务列表中各属性的值，其中状态值为“运行中“表示新增一组集群之间的数据同步成功



### 验证数据同步

测试数据同步，集群Cluster\_A计算节点写人数据

PGPASSWORD=abc psql -h 127.16.0.16 -U abc -p 47001 -d postgres



在集群Cluster\_B计算节点查询，数据已经成功同步过来(注意：此时Cluster\_B作为备集群，不允许写人，可以查询相关数据)

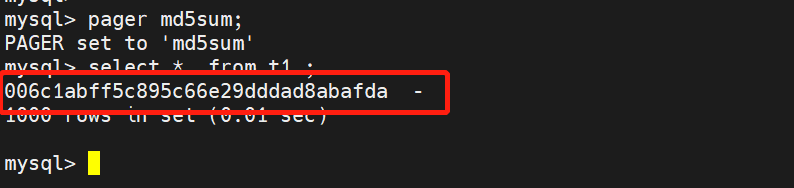
PGPASSWORD=abc psql -h 127.16.1.20 -U abc -p 47001 -d postgres



### 验证数据一致性

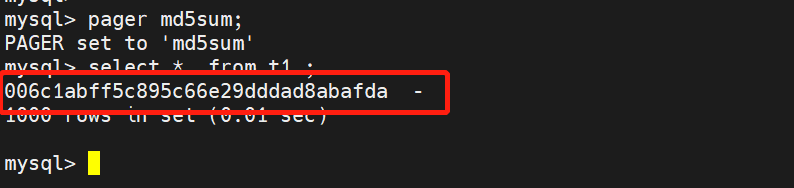
为了进一步验证数据的准确性和一致性，使用MySQL协议进入计算节点，校验表中数据中的md5值

mysql -uabc -pabc -h172.16.128.16 -P47002



同样使用MySQL协议登录计算节点校验表中数据中的md5值

mysql -uabc -pabc -h172.16.1.20 -P47002

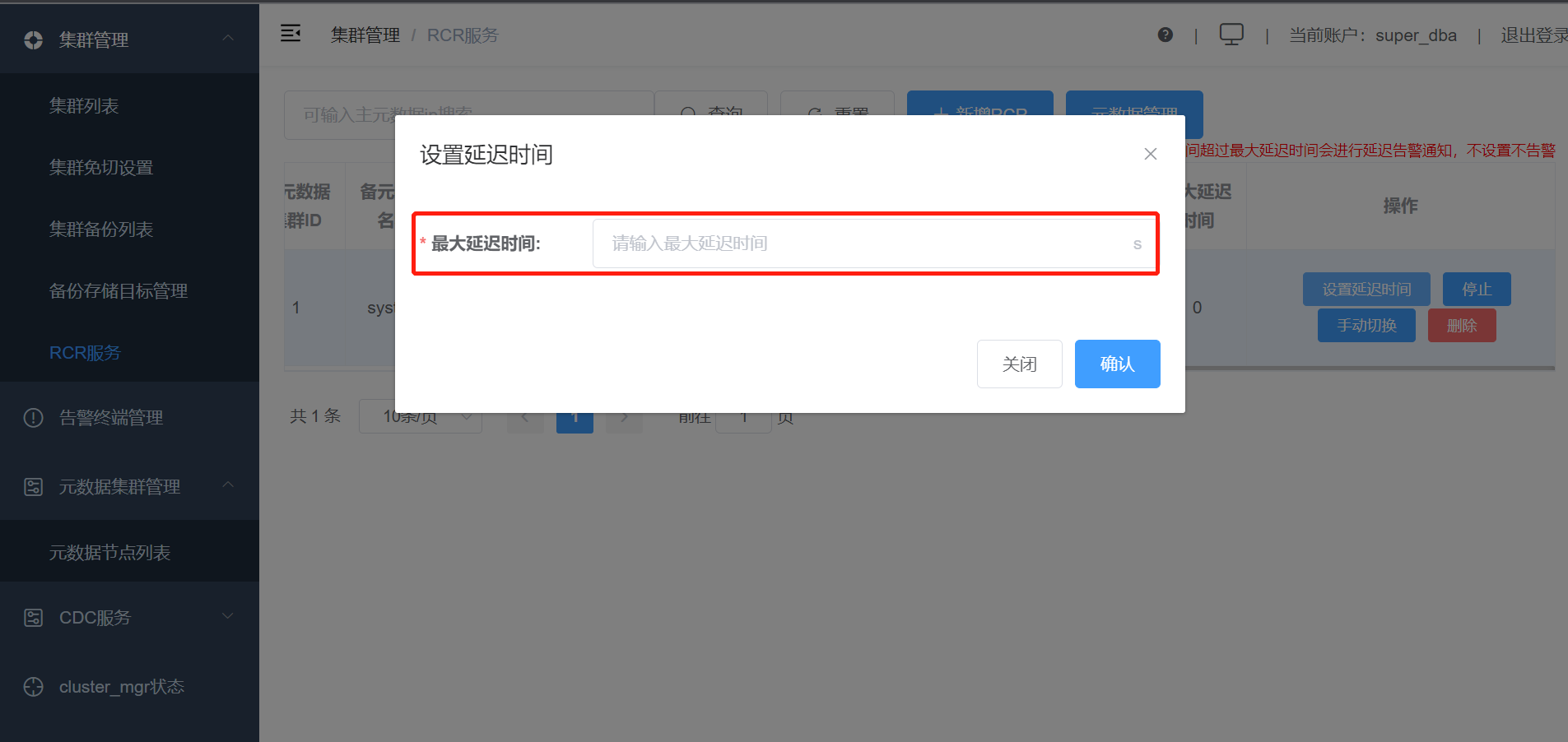


其结果与主Cluster\_A集群对应数据的md5值完全一样

### 延迟复制设置

延迟复制设置，此功能类似mysql延迟复制，Cluster\_A等待多少秒后数据同步到Cluster\_B中

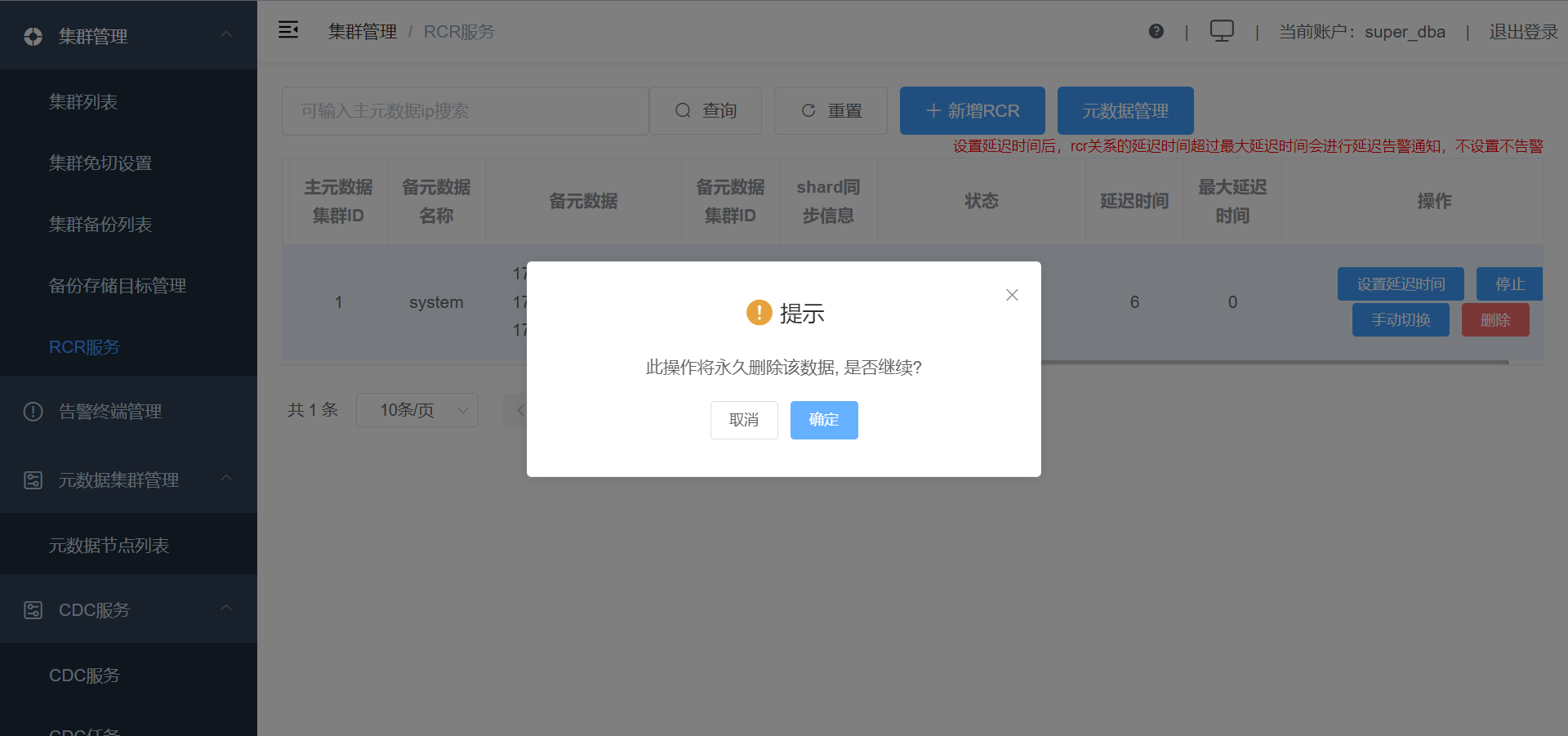


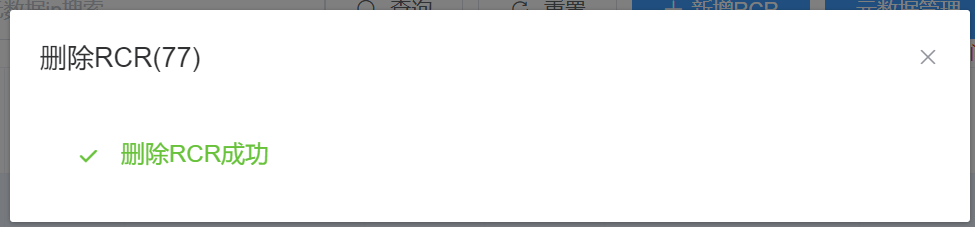


### 删除RCR

集群之间建立RCR关系，集群不允许删除，解绑关系之后才能删除，注意：生产环境中慎重删除集群操作

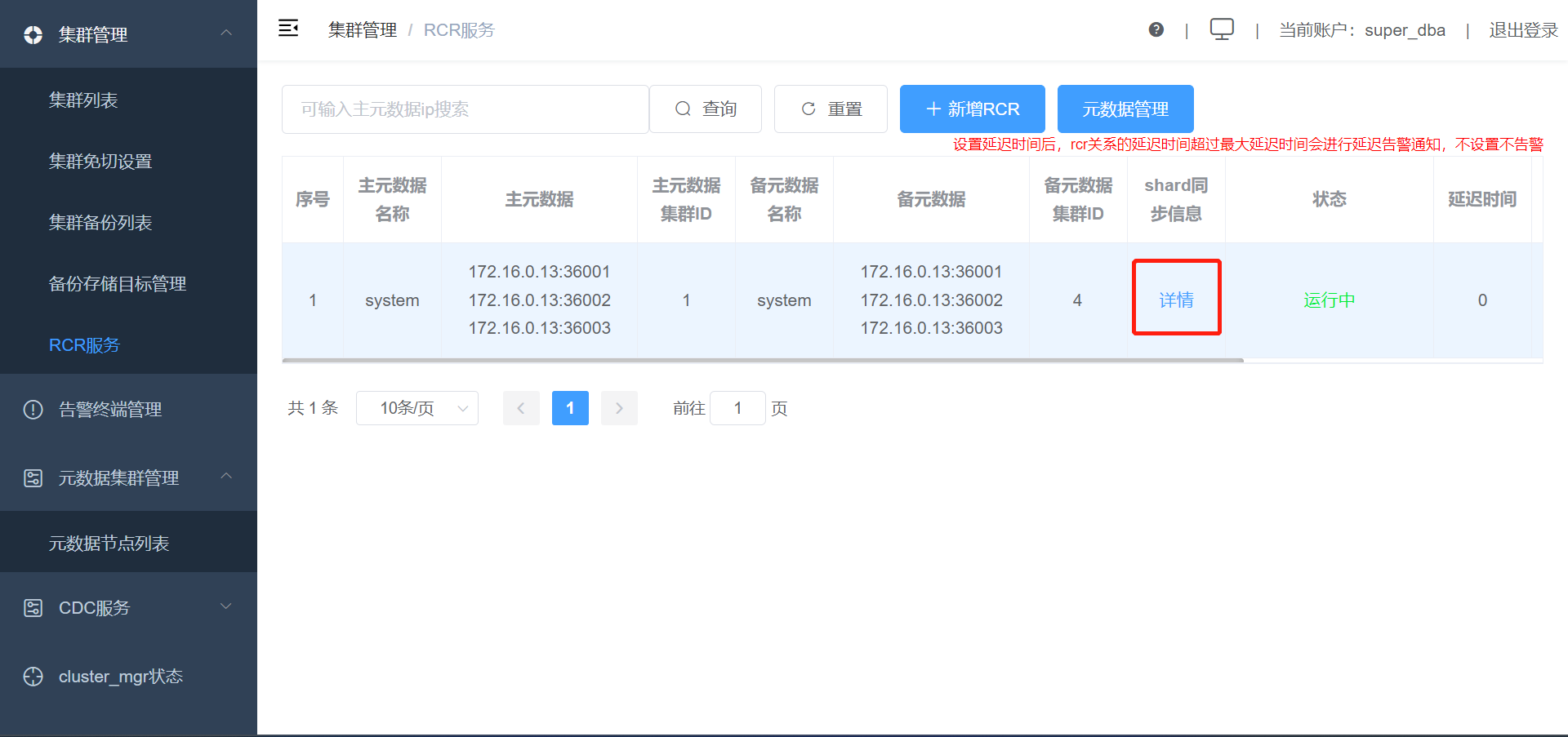






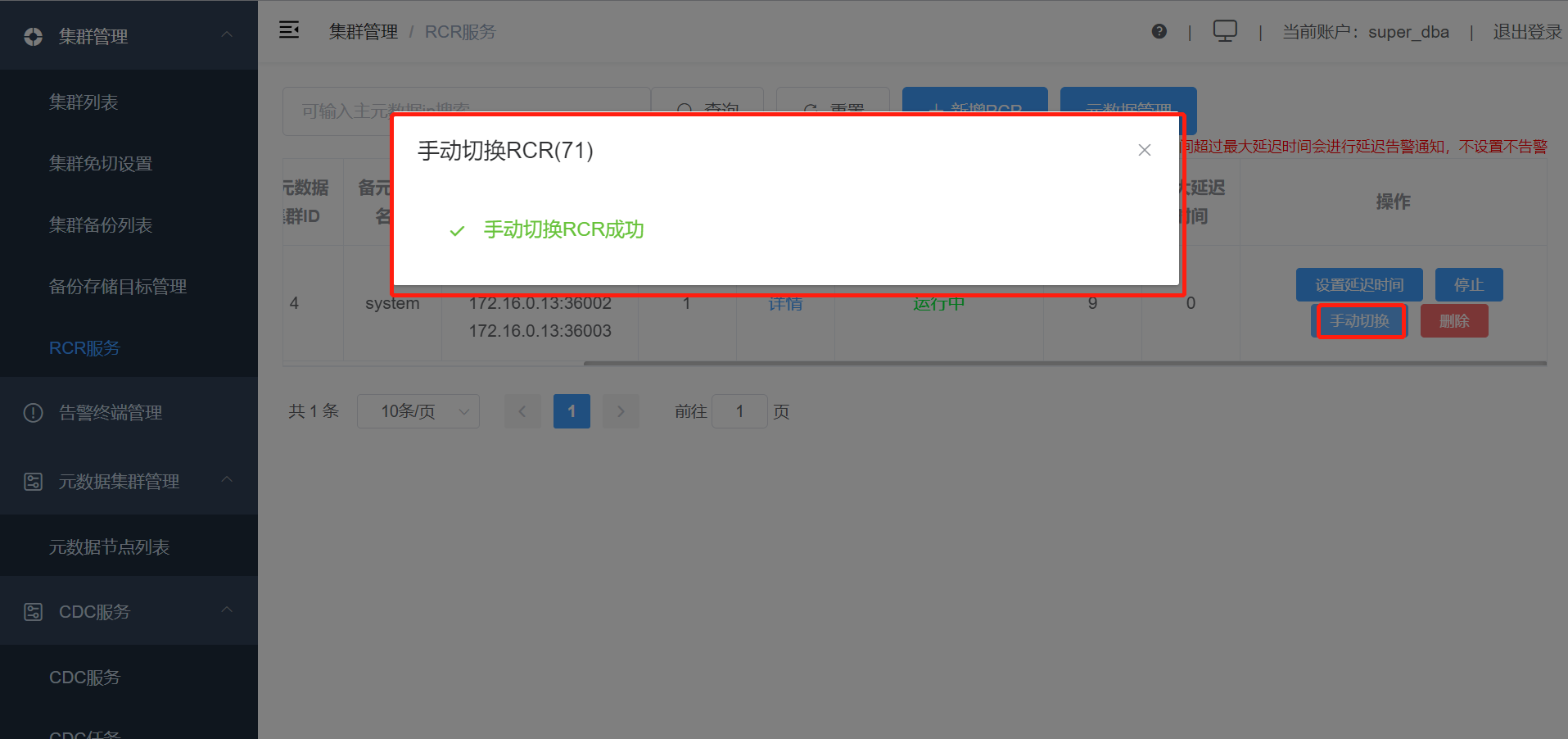
### 切换RCR

集群手工切换



切换之前点击“详情”查询一下主备对应的复制信息



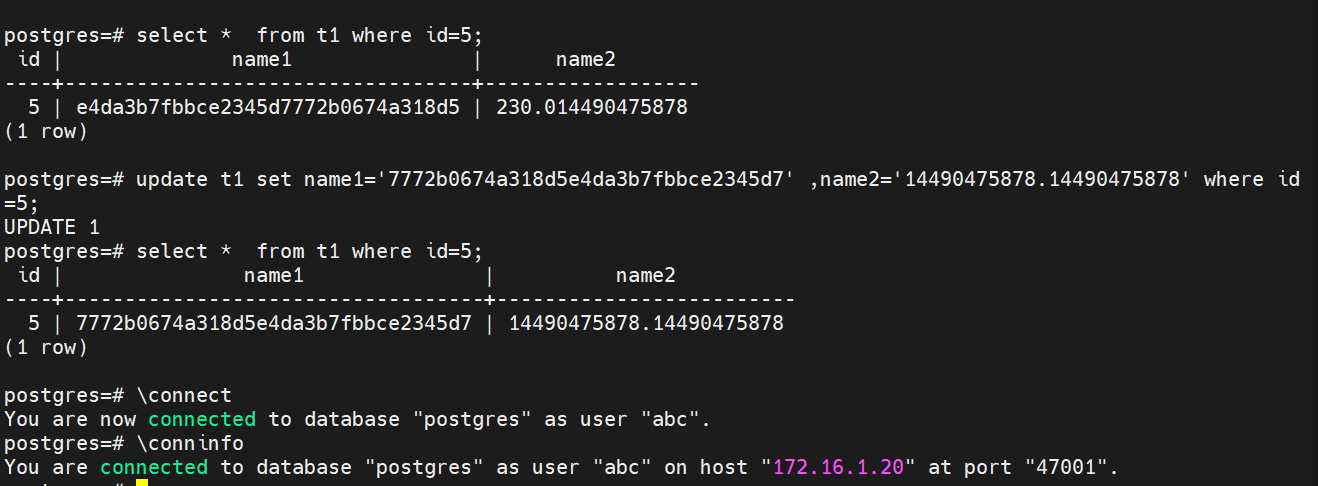




切换后验证数据准确性

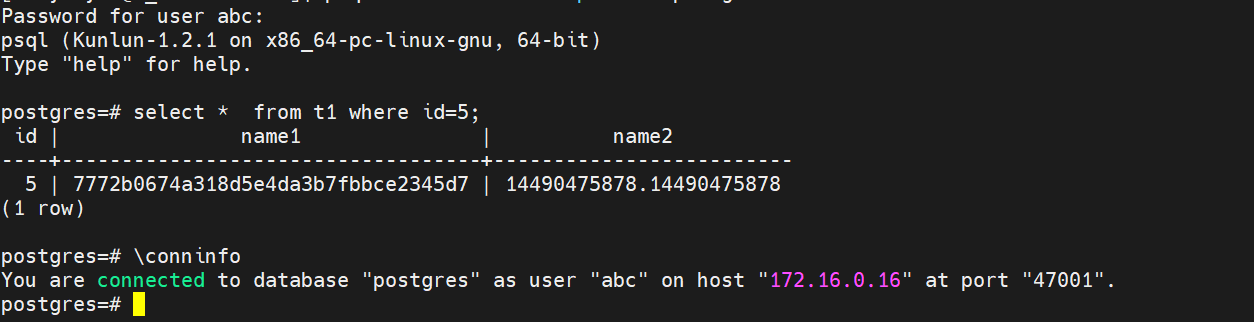
主集群上面修改信息Cluster\_B(没有切换之前为备，现在为主)，登录主集群计算节点

PGPASSWORD=abc psql -h 127.16.1.20 -U abc -p 47001 -d postgres



备集群上面修改信息Cluster\_A(没有切换之前为主，现在为备)，登录备集群计算节点

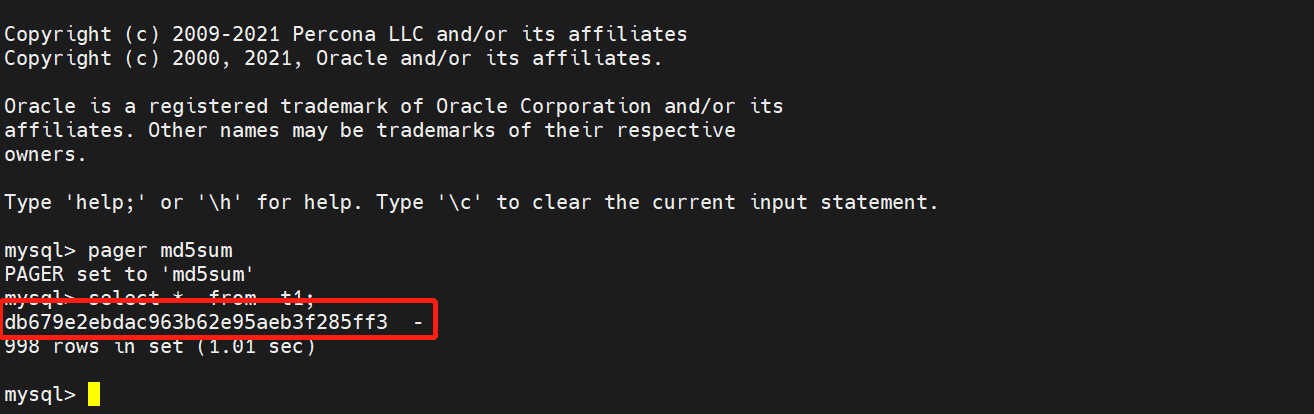
PGPASSWORD=abc psql -h 127.16.0.15 -U abc -p 47001 -d postgres



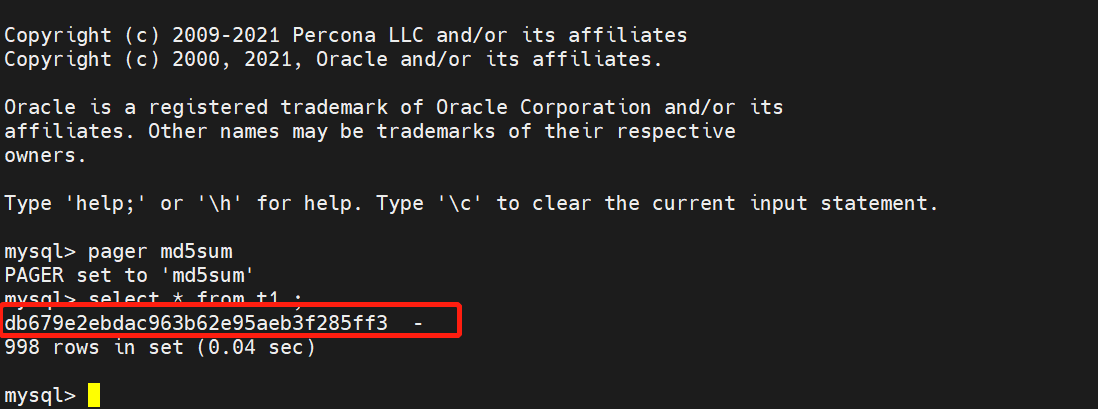
数据查询为一致

进一步验证数据一致性，使用MySQL协议分别登入主备集群计算节点查询

mysql -uabc -pabc -h172.16.1.20 -P47002



mysql -uabc -pabc -h172.16.128.15 -P47002



结果显示，查询数据的MD5值完全一致

### 启动/停止RCR

停止RCR



启动RCR



停止和启动属性列表状态会有相应的变化

### CRC添加/删除shard

创建CRC关系的时候，主备cluster之间shard数量必须一致，如果在建立RCR关系cluster中任何一个cluster添加/删除shard，系统自动根据RCR关系到对端cluster中添加/删除shard

在主cluster上增加shard，如下图所示：





主添加shard成功后，备集群也对应添加了相应的shard节点





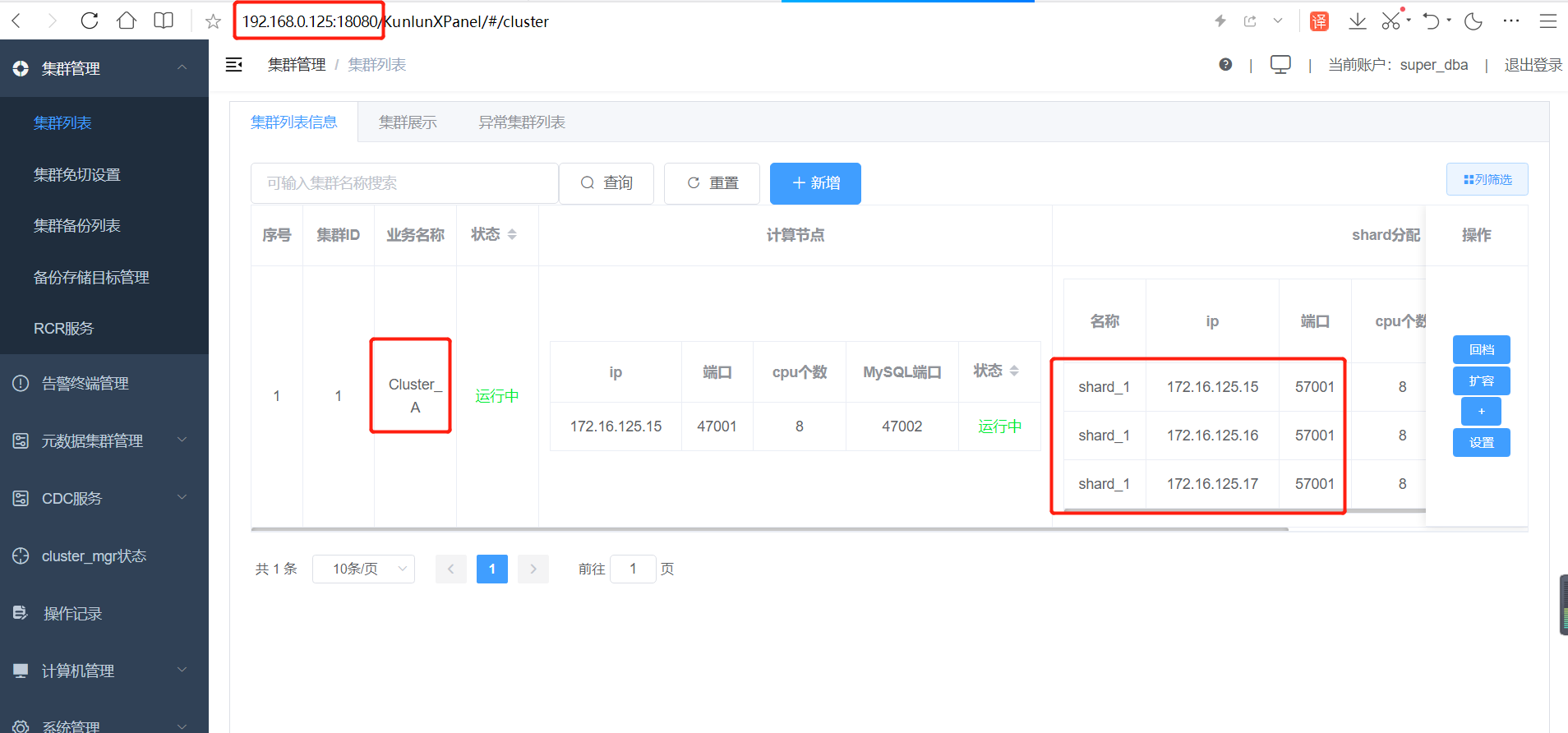
删除操作也同样如此，此处不再演示

## 非本地xpanel下管理的集群之间的数据同步

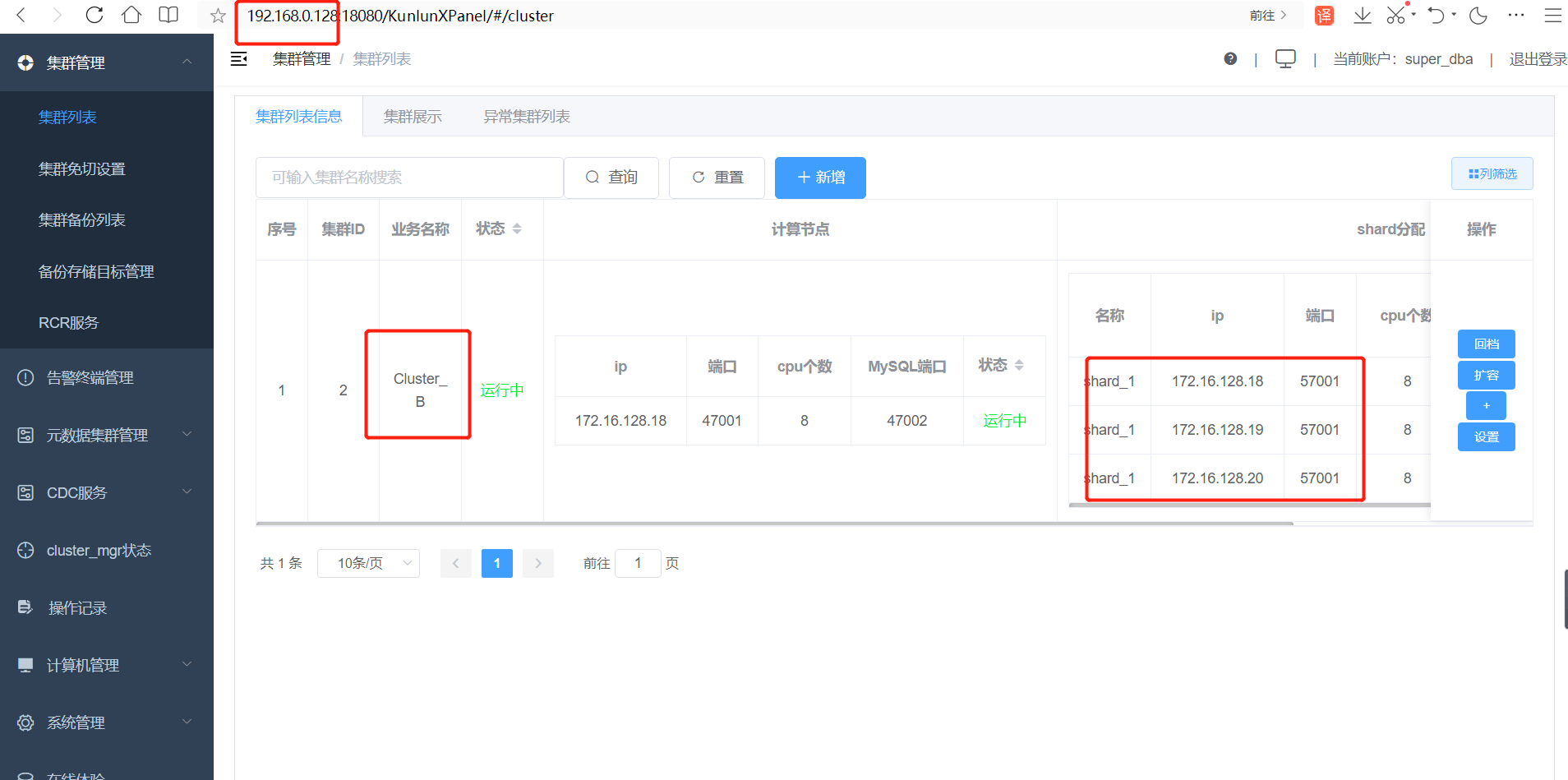
前面介绍了本地xpanel管理集群之间的数据同步即（所有集群共享同一个元数据集群），下面介绍不同xpanel管理下集群数据同步即（所有集群不在同一个元数据集群）在同步数据之前，逻辑上没有任何关系。

### 数据准备

[http://192.168.0.125:18080/KunlunXPanel/](http://192.168.0.125:18080/KunlunXPanel/2) 管理1个集群，为Cluster\_A



<http://192.168.0.128:18080/KunlunXPanel/> 管理1个集群，为Cluster\_B

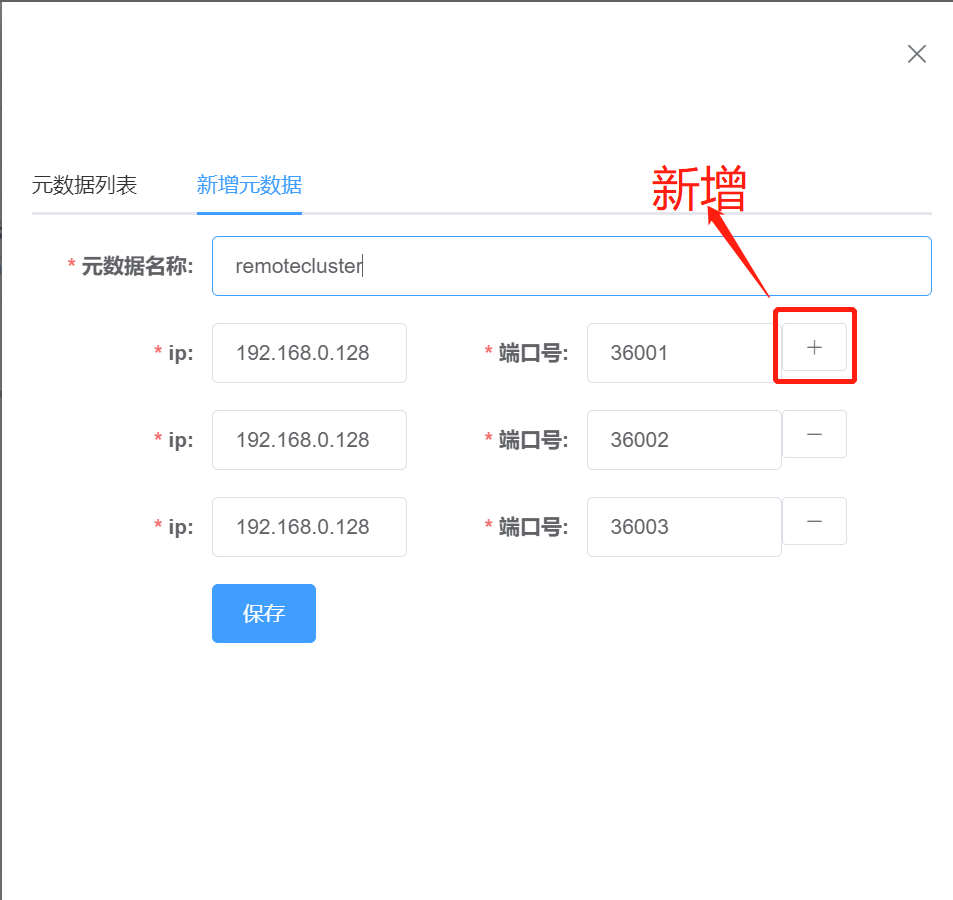


### 新增RCR

在集群管理列表中点击“RCR服务”，然后点击“元数据管理“按钮如图所示：



在新增元数据列表中添加备集群的元数据信息



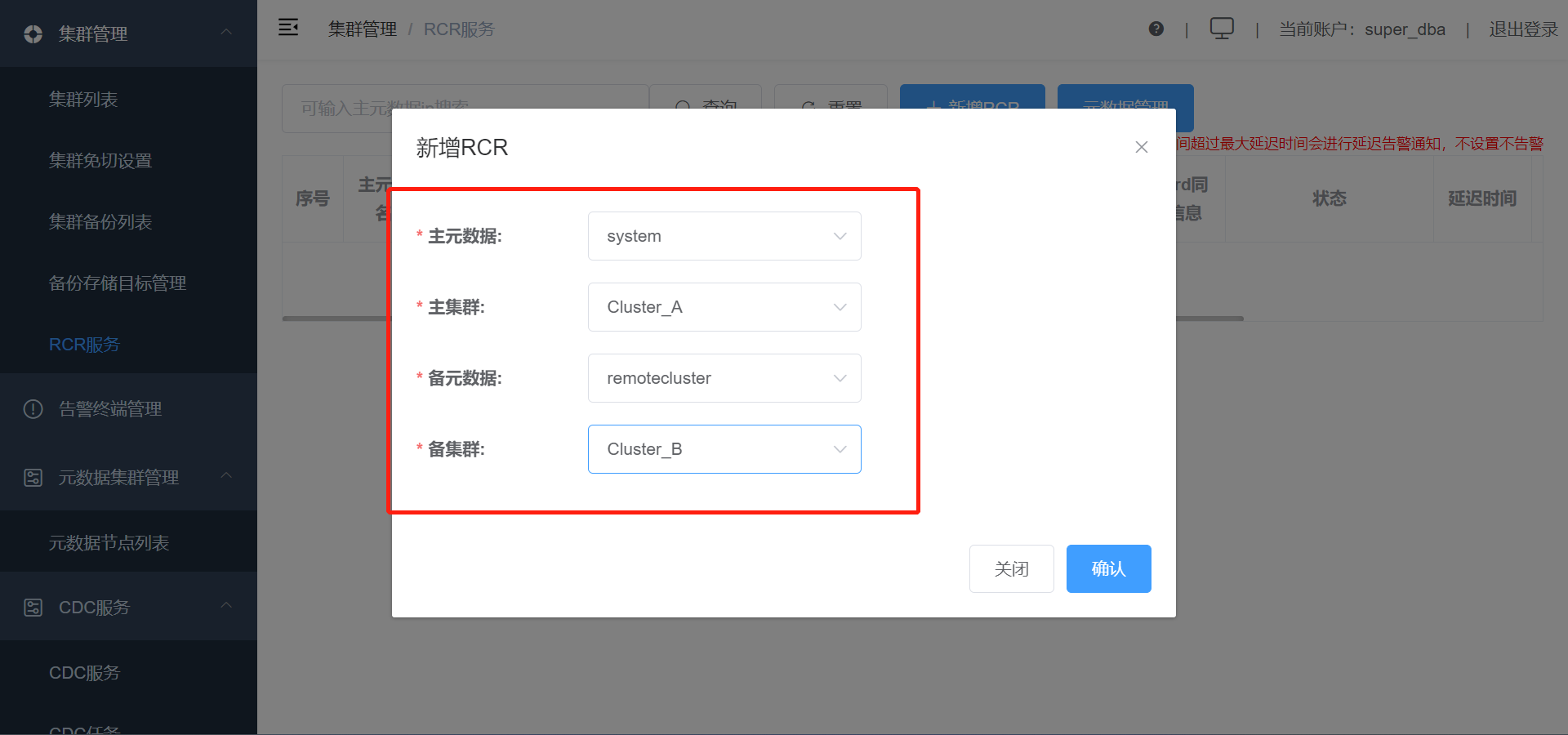
新增成功后,在元数据列表中显示刚才添加的相应的信息如下图所示:



在集群管理列表中点击“RCR服务”，然后点击“+新增RCR“按钮如图所示：



分别对应添加Cluseter\_A和Cluseter\_B元数据节点信息



新增RCR成功后，列表信息显示如下：



点击“详情”可以查询到相关的同步信息：





其它功能和本地xpanel集群同步功能一样，不再累述！