# Redis Cluster那些事儿

峰云就她了

xiaorui.cc

github.com/rfyiamcool

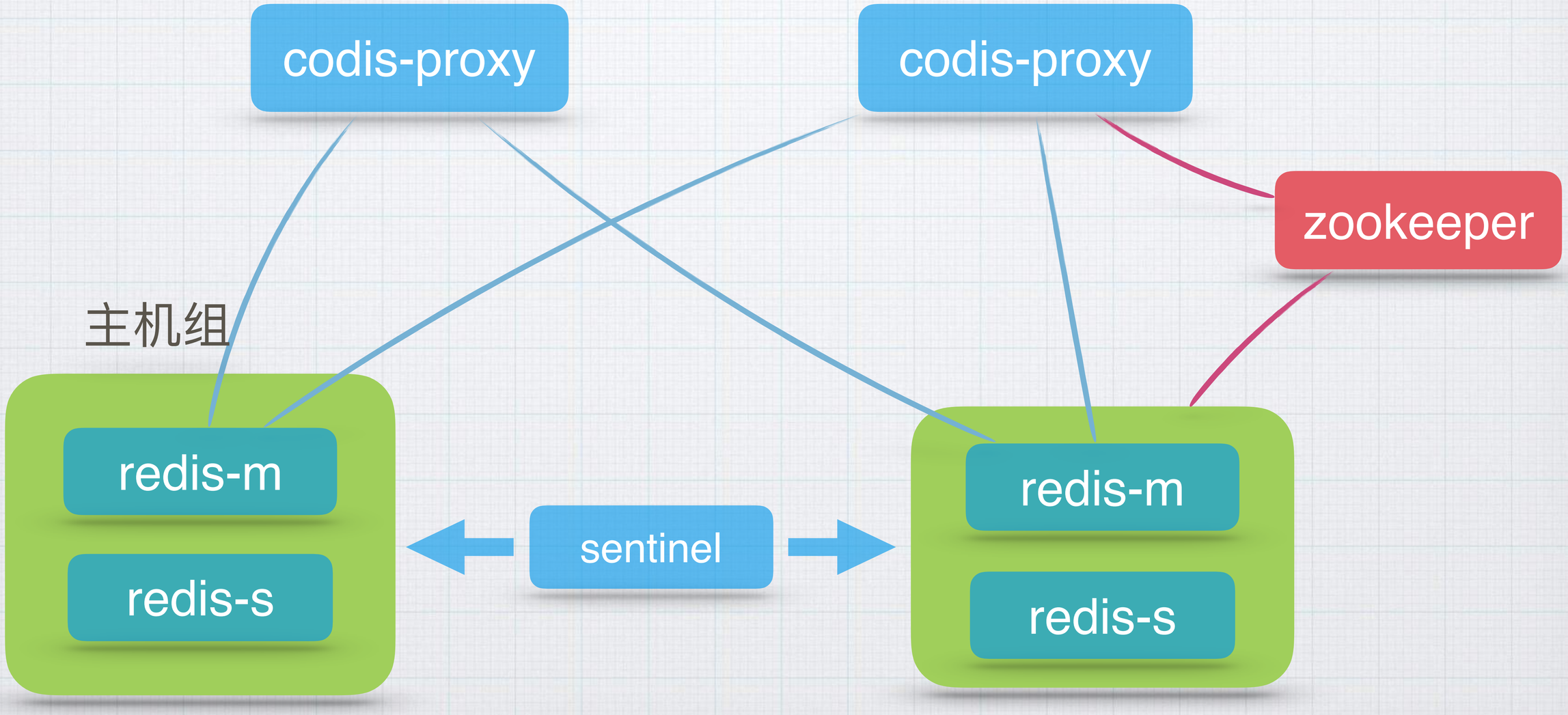
# 主流的集群方案

* vip多线程版 twemproxy
* codis
* smart_proxy + redis cluster
* redis cluster

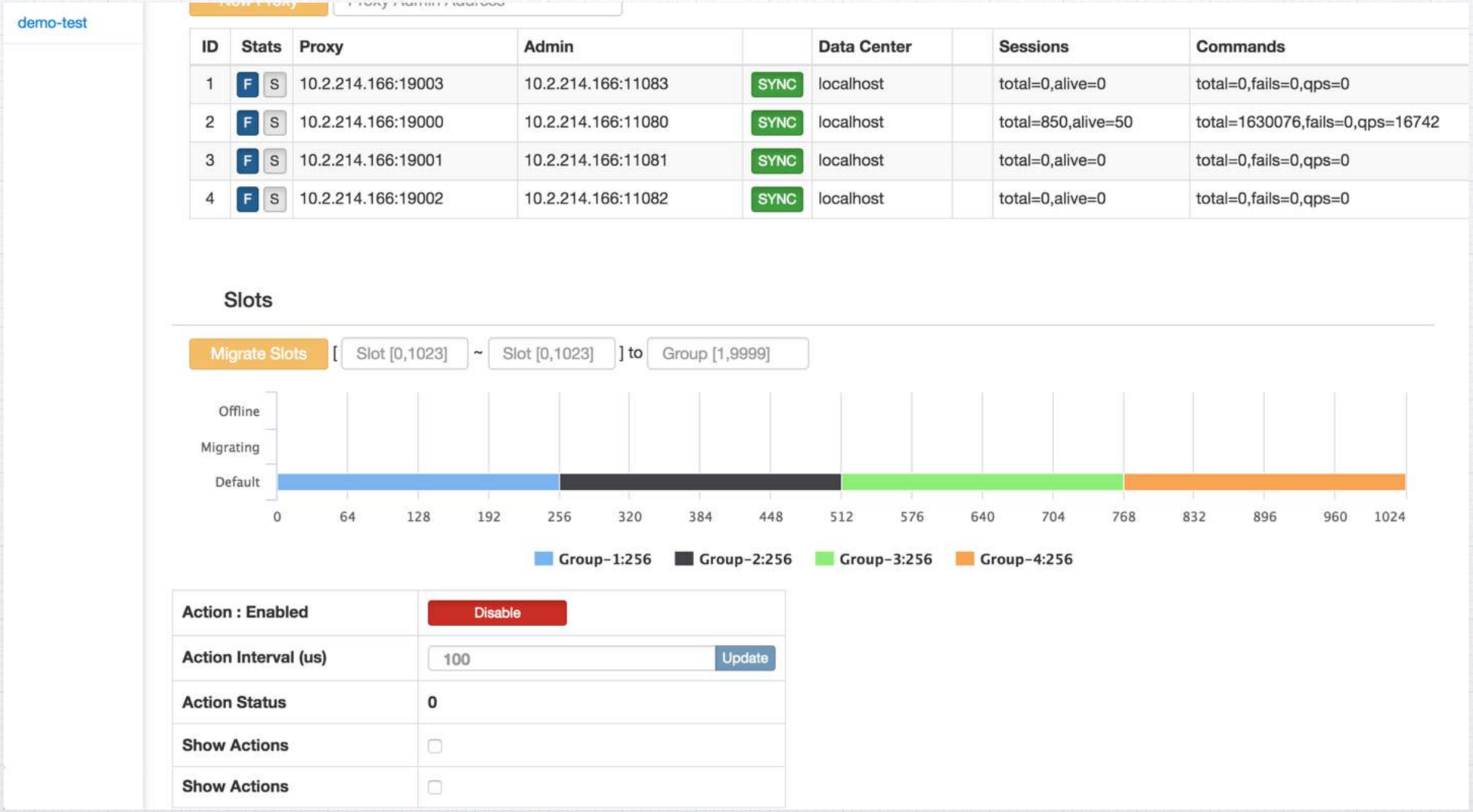
# redis cluster vs codis

| | cluster | codis |
|---|---|---|
| hash_tag | y | y |
| design | 去中心化 | 中心化 |
| pipeline | client design | 支持 |
| slot | y | y |
| 多db | n | y |
| 性能 | high | this < cluster |
| code | 相当复杂 | 简单 |
| 范围 | 广 | 也有不少大厂 |

# codis


codis-proxy
codis-proxy
zookeeper
主机组
redis-m
redis-s
sentinel
redis-m
redis-s

# codis dashboard



# codis


我个人很喜欢Codis，研究过其源码实现.

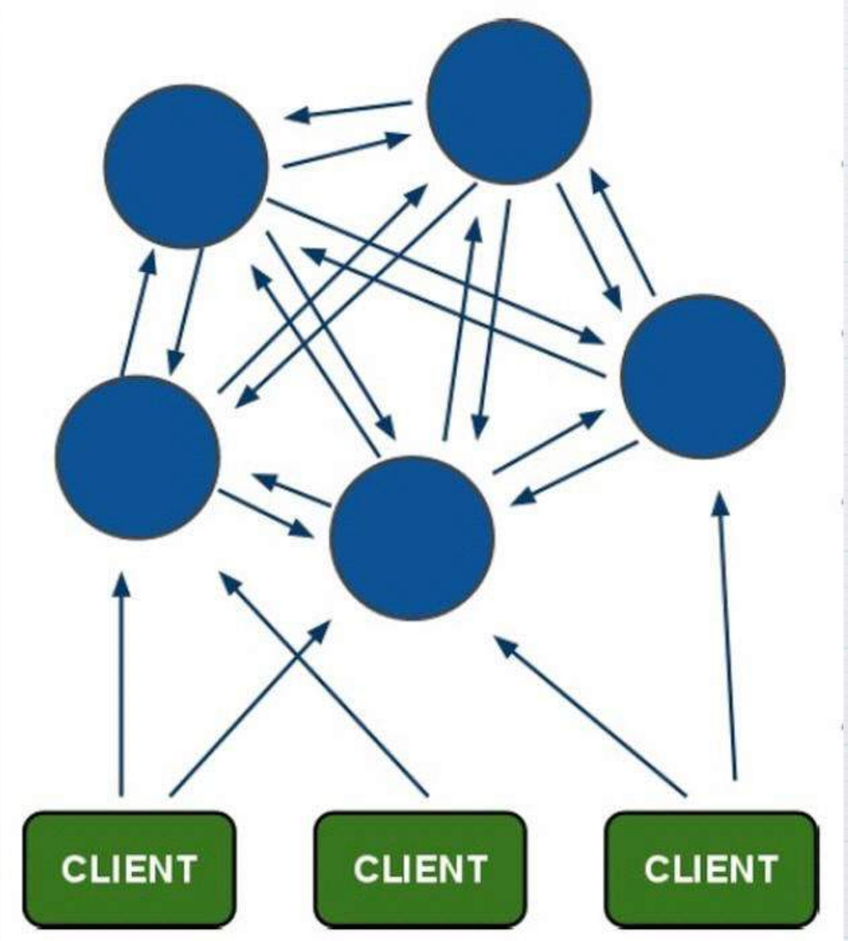
but 这次主题是Redis Cluster !!!

So，…

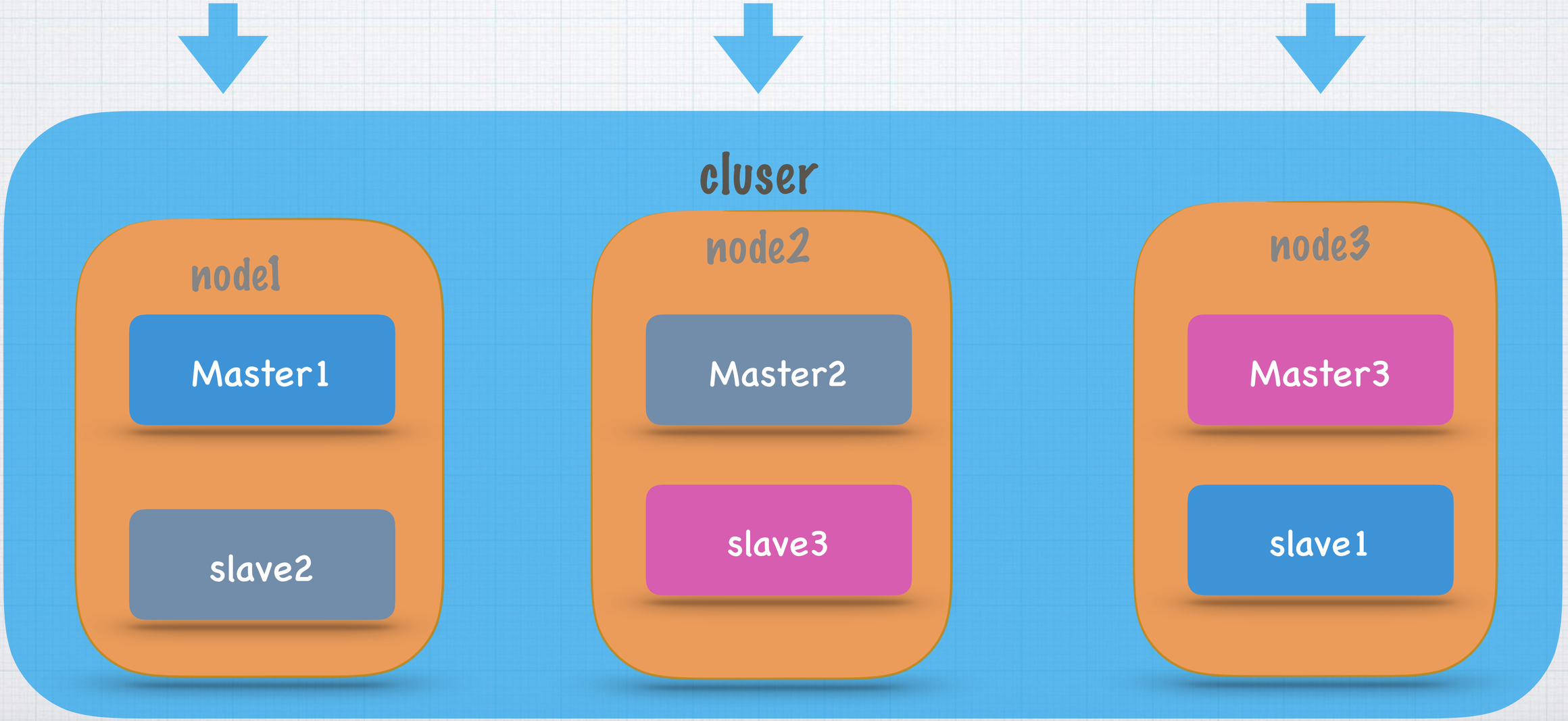
# redis cluster

去中心化

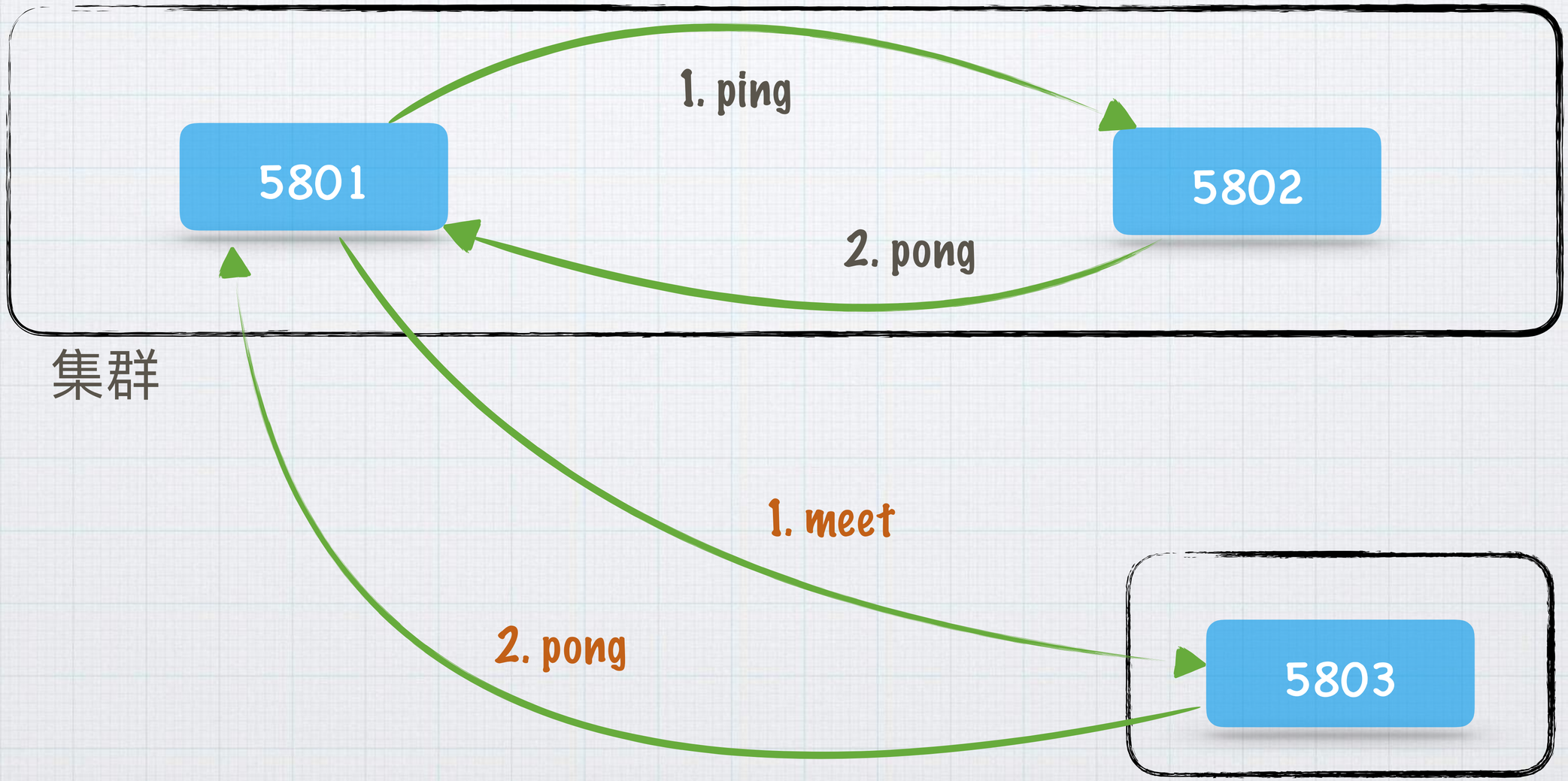
Gossip协议



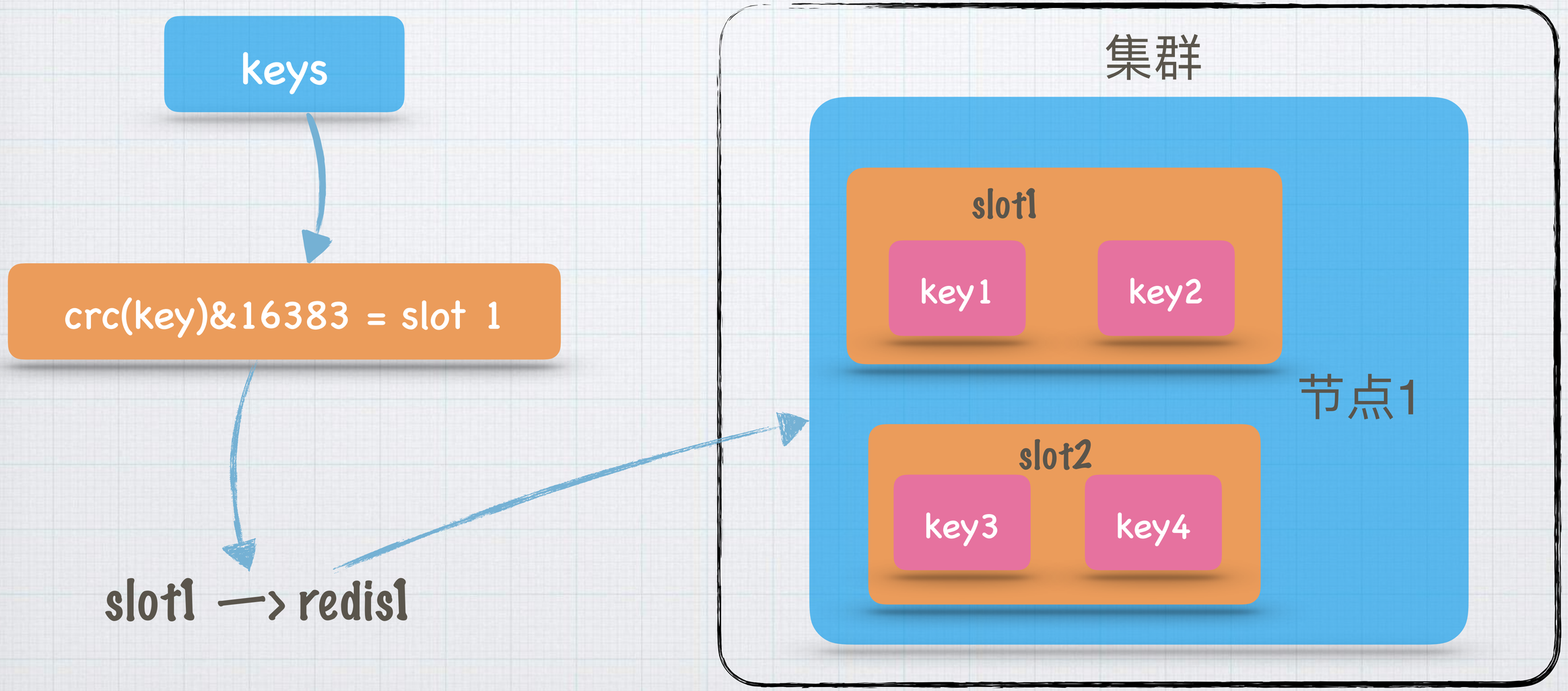
# 架构


cluser
node1
Master1
slave2
node2
Master2
slave3
node3
Master3
slave1

# gossip



# slot mapping



# redis cluster优点

* 高性能, 避免proxy代理的消耗
* 高可用, 自动故障转义
* 自带迁移功能
* 丰富的集群管理命令

# redis cluster缺点

* client实现复杂, 需要缓存slot mapping
* 迁移异常不能自修复
* 节点太多时, 节点检测占用带宽
* 不支持不同slot的批量命令
* more…

# cluster cmd

```
//集群(cluster)
CLUSTER INFO 打印集群的信息
CLUSTER NODES 列出集群当前已知的所有节点（node），以及这些节点的相关信息。

//节点(node)
CLUSTER MEET <ip> <port> 将 ip 和 port 所指定的节点添加到集群当中，让它成为集群的一份子。
CLUSTER FORGET <node_id> 从集群中移除 node_id 指定的节点。
CLUSTER REPLICATE <node_id> 将当前节点设置为 node_id 指定的节点的从节点。
CLUSTER SAVECONFIG 将节点的配置文件保存到硬盘里面。

//槽(slot)
CLUSTER ADDSLOTS <slot> [slot ...] 将一个或多个槽（slot）指派（assign）给当前节点。
CLUSTER DELSLOTS <slot> [slot ...] 移除一个或多个槽对当前节点的指派。
CLUSTER FLUSHSLOTS 移除指派给当前节点的所有槽，让当前节点变成一个没有指派任何槽的节点。
CLUSTER SETSLOT <slot> NODE <node_id> 将槽 slot 指派给 node_id 指定的节点，如果槽已经指派给另一个节点
CLUSTER SETSLOT <slot> MIGRATING <node_id> 将本节点的槽 slot 迁移到 node_id 指定的节点中。
CLUSTER SETSLOT <slot> IMPORTING <node_id> 从 node_id 指定的节点中导入槽 slot 到本节点。
CLUSTER SETSLOT <slot> STABLE 取消对槽 slot 的导入（import）或者迁移（migrate）。

//键 (key)
CLUSTER KEYSLOT <key> 计算键 key 应该被放置在哪个槽上。
CLUSTER COUNTKEYSINSLOT <slot> 返回槽 slot 目前包含的键值对数量。
CLUSTER GETKEYSINSLOT <slot> <count> 返回 count 个 slot 槽中的键。
```

# redis-trib

官方集群管理工具

* create：创建集群
* check：检查集群
* info：查看集群信息
* fix：修复集群
* add-node：将新节点加入集群
* del-node：从集群中删除节点
* reshard：在线迁移slot
* rebalance：平衡集群节点slot数量
* set-timeout：设置集群节点间心跳连接的超时时间
* call：在集群全部节点上执行命令
* import：将外部redis数据导入集群

# quick start multi redis

```
/etc/redis
├── 85010.conf
├── 85011.conf
├── 85012.conf
├── 8501.conf
├── 8502.conf
├── 8503.conf
├── 8504.conf
├── 8505.conf
├── 8506.conf
├── 8507.conf
├── 8508.conf
└── 8509.conf
```

```
daemonize yes
port 8501
dir /data/redis/8501
cluster-enabled yes
cluster-config-file 8501_nodes.conf
cluster-node-timeout 10000
repl-backlog-siz 64m
```

```
/data/redis/
├── 8501
│   ├── 8501_nodes.conf
│   ├── appendonly.aof
│   └── dump.rdb
├── 8502
│   ├── 8502_nodes.conf
│   ├── appendonly.aof
│   └── dump.rdb
├── 8503
│   ├── 8503_nodes.conf
│   ├── appendonly.aof
│   └── dump.rdb
├── 8504
│   ├── 8504_nodes.conf
│   ├── appendonly.aof
│   └── dump.rdb
├── 8505
│   ├── 8505_nodes.conf
│   ├── appendonly.aof
│   └── dump.rdb
└── 8506
    ├── 8506_nodes.conf
    ├── appendonly.aof
    └── dump.rdb
```

```
redis-ser 32265 root    5u   REG      0,9        0     3853 [eventpoll]
redis-ser 32265 root    6u  IPv4 23235759      0t0      TCP *:8502 (LISTEN)
redis-ser 32265 root    7w   REG    252,17      451   688138 /data/redis/8502/appendonly.aof
redis-ser 32265 root    8wW  REG    252,17      765   688139 /data/redis/8502/8502_nodes.conf
redis-ser 32265 root    9u  IPv4 23235776      0t0      TCP *:18502 (LISTEN)
redis-ser 32265 root   10u  IPv4 23344352      0t0      TCP localhost:8502->localhost:51615 (ESTABLISHED)
redis-ser 32265 root   11u  IPv4 23344118      0t0      TCP localhost:57912->localhost:18501 (ESTABLISHED)
redis-ser 32265 root   12u  IPv4 23344139      0t0      TCP localhost:18502->localhost:44333 (ESTABLISHED)
redis-ser 32265 root   13u  IPv4 23344170      0t0      TCP localhost:18502->localhost:44381 (ESTABLISHED)
redis-ser 32265 root   14u  IPv4 23344172      0t0      TCP localhost:39035->localhost:18506 (ESTABLISHED)
redis-ser 32265 root   16u  IPv4 23344179      0t0      TCP localhost:40157->localhost:18504 (ESTABLISHED)
redis-ser 32265 root   19u  IPv4 23344224      0t0      TCP localhost:18502->localhost:44419 (ESTABLISHED)
```

# create redis cluster

* 第一种

./redis-trib.rb create --replicas 1 127.0.0.1:8501 127.0.0.1:8502 127.0.0.1:8503 127.0.0.1:8504 127.0.0.1:8505 127.0.0.1:8506

* 第二种

> meet & cluster add-node &cluster setslot

# add redis cluster

* 第一种
    * 127.0.0.1:8507 > cluster meet 127.0.0.1 8501
* 第二种
    * > redis-trib.rb add-node 127.0.0.1:8507 127.0.0.1:8501

# extend cluster

* 准备新节点
* 加入集群
* 迁移槽位和数据
* 通告

* 第一种
  * > redis-trib.rb reshard 127.0.0.1:8501
* 第二种
  1. 目标 cluster setslot 6818 importing
  2. 源 cluster setslot 6818 migrating
  3. cluster getkeysinslot 6818 5
  4. migrate 127.0.0.1 6818 “” 0 1000 keys k1 k2 k3

# reduce cluster

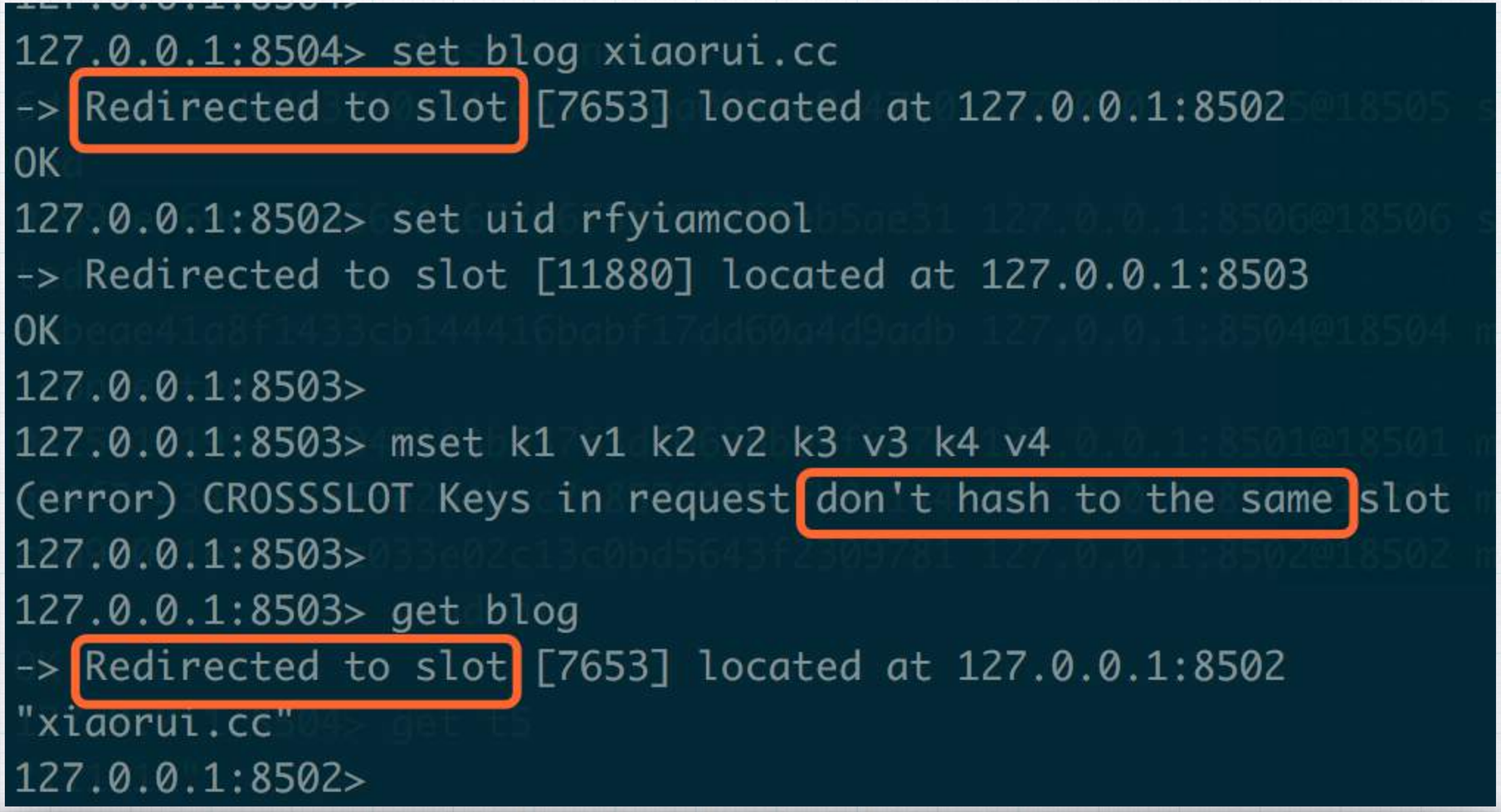
* 迁移槽
  * > redis-trib.rb reshard 127.0.0.1:8501
* 删除主机
  * > redis-trib.rb del-node 127.0.0.1:8501

# cluster status

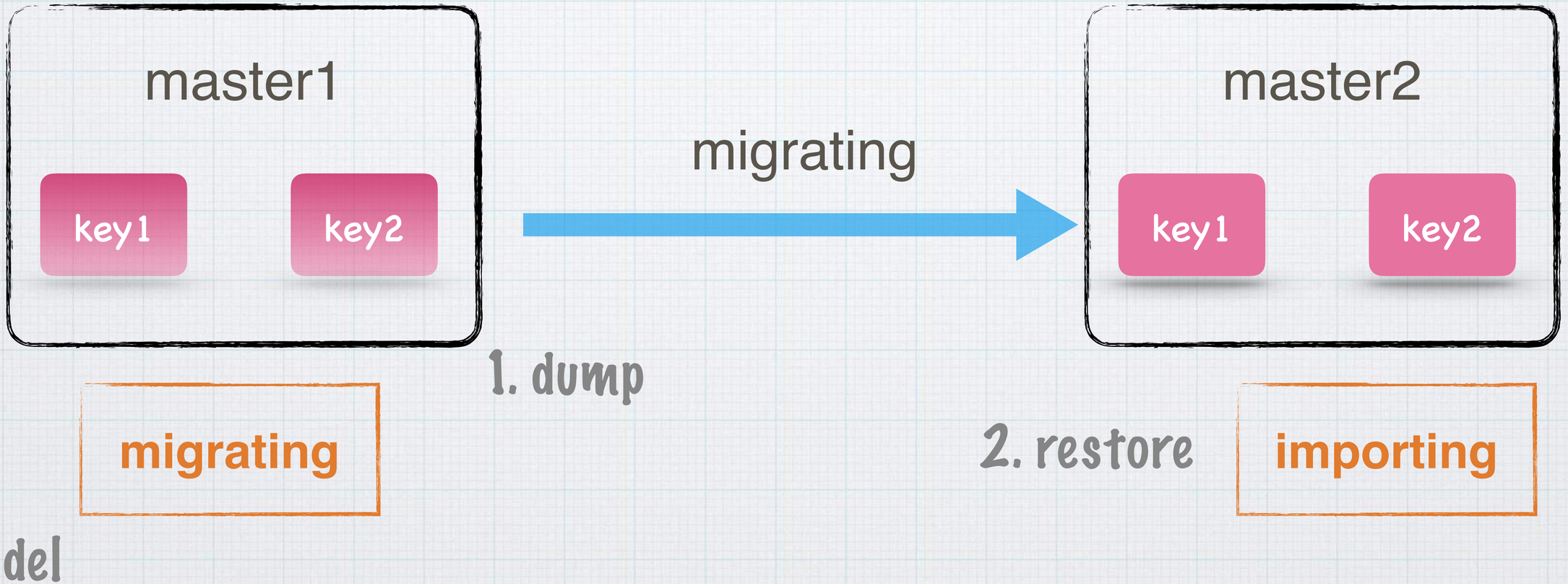
```
127.0.0.1:8504> cluster nodes
6d88a412ed0403740a84cd535ba6a095ca0a47b0 127.0.0.1:8505@18505 slave a34f7f434d6a7afee2d4bcc2e8576975cfb8e144 0 1527604079000 5 connected
07994e369c731a56f1a67206758057262eb5ae31 127.0.0.1:8506@18506 slave 0425e1919319c1045163b53763df069bb99f647a 0 1527604079692 6 connected
48beae41a8f1433cb144416babf17dd60a4d9adb 127.0.0.1:8504@18504 myself,slave c9d9d031e73c6dc033e02c13c0bd5643f2309781 0 1527604075000 4 connec
0425e1919319c1045163b53763df069bb99f647a 127.0.0.1:8501@18501 master - 0 1527604079592 1 connected 0-5460
a34f7f434d6a7afee2d4bcc2e8576975cfb8e144 127.0.0.1:8503@18503 master - 0 1527604079000 3 connected 10923-16383
c9d9d031e73c6dc033e02c13c0bd5643f2309781 127.0.0.1:8502@18502 master - 0 1527604079000 2 connected 5461-10922
```

```
127.0.0.1:8504> cluster info
cluster_state:ok
cluster_slots_assigned:16384
cluster_slots_ok:16384
cluster_slots_pfail:0
cluster_slots_fail:0
cluster_known_nodes:6
cluster_size:3
cluster_current_epoch:6
cluster_my_epoch:2
cluster_stats_messages_ping_sent:347606
cluster_stats_messages_pong_sent:349257
cluster_stats_messages_meet_sent:3
cluster_stats_messages_sent:696866
cluster_stats_messages_ping_received:349254
cluster_stats_messages_pong_received:347609
cluster_stats_messages_meet_received:3
cluster_stats_messages_received:696866
```

# cli


127.0.0.1:8504> set blog xiaorui.cc
-> Redirected to slot [7653] located at 127.0.0.1:8502
OK
127.0.0.1:8502> set uid rfyiamcool
-> Redirected to slot [11880] located at 127.0.0.1:8503
OK
127.0.0.1:8503>
127.0.0.1:8503> mset k1 v1 k2 v2 k3 v3 k4 v4
(error) CROSSSLOT Keys in request don't hash to the same slot
127.0.0.1:8503>
127.0.0.1:8503> get blog
-> Redirected to slot [7653] located at 127.0.0.1:8502
"xiaorui.cc"
127.0.0.1:8502>

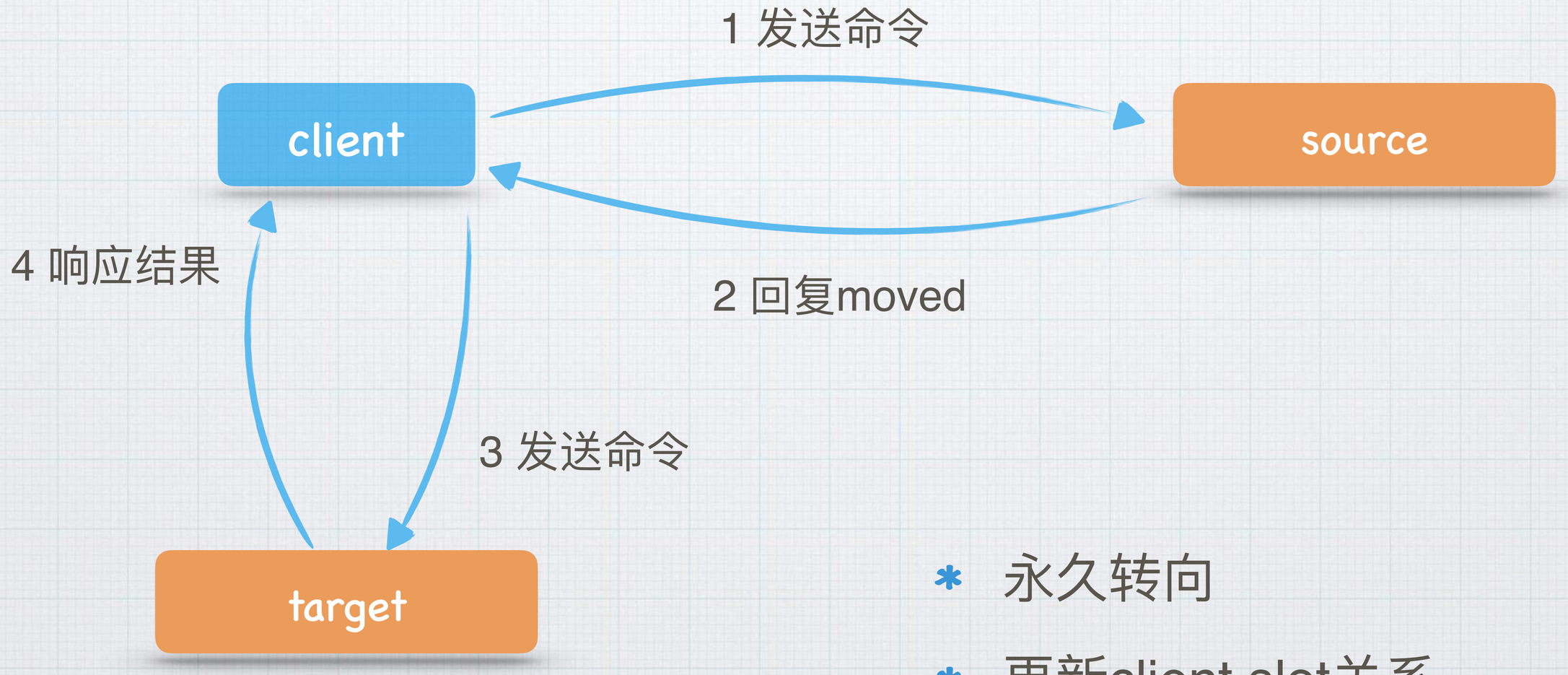
# migrate


master1
key1
key2
migrating
master2
key1
key2
1. dump
migrating
2. restore
importing
3. del

# ask in migrate

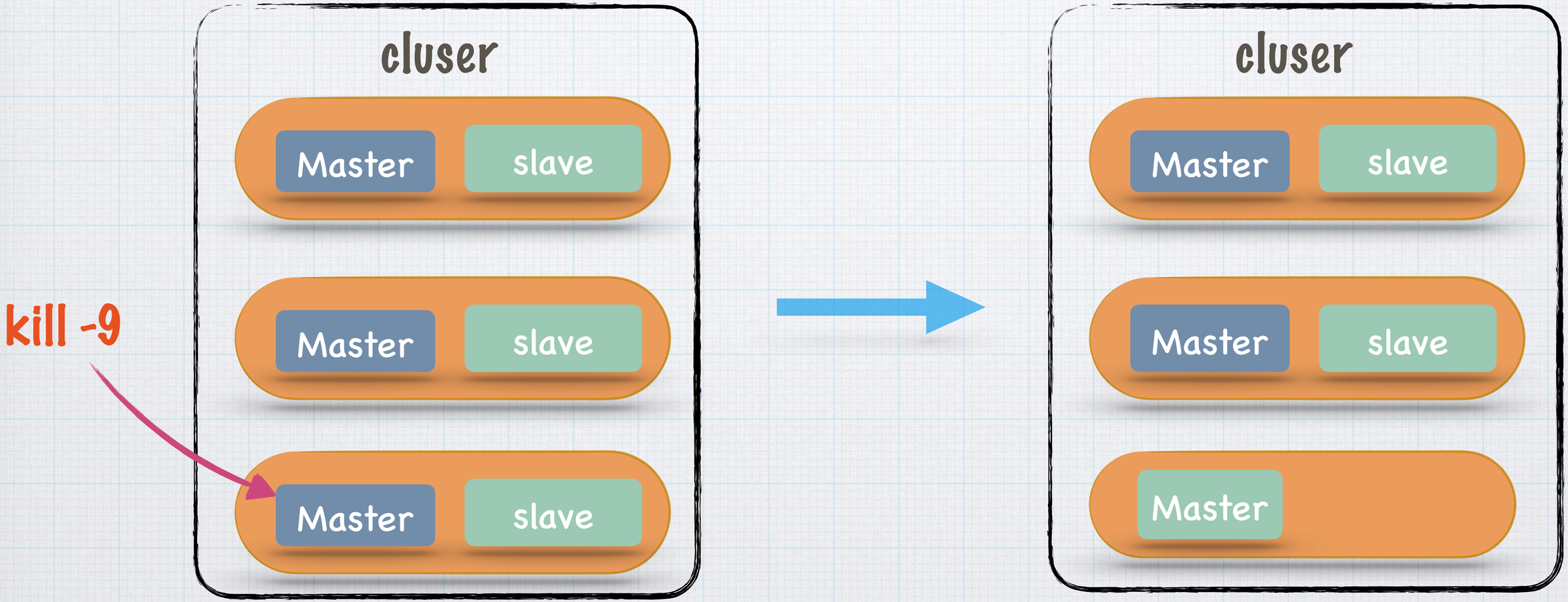

* ask临时转向
* ask只用在迁移中
* 对端需要asking，不然对端会拒绝

# moved




* 永久转向
* 更新client slot关系

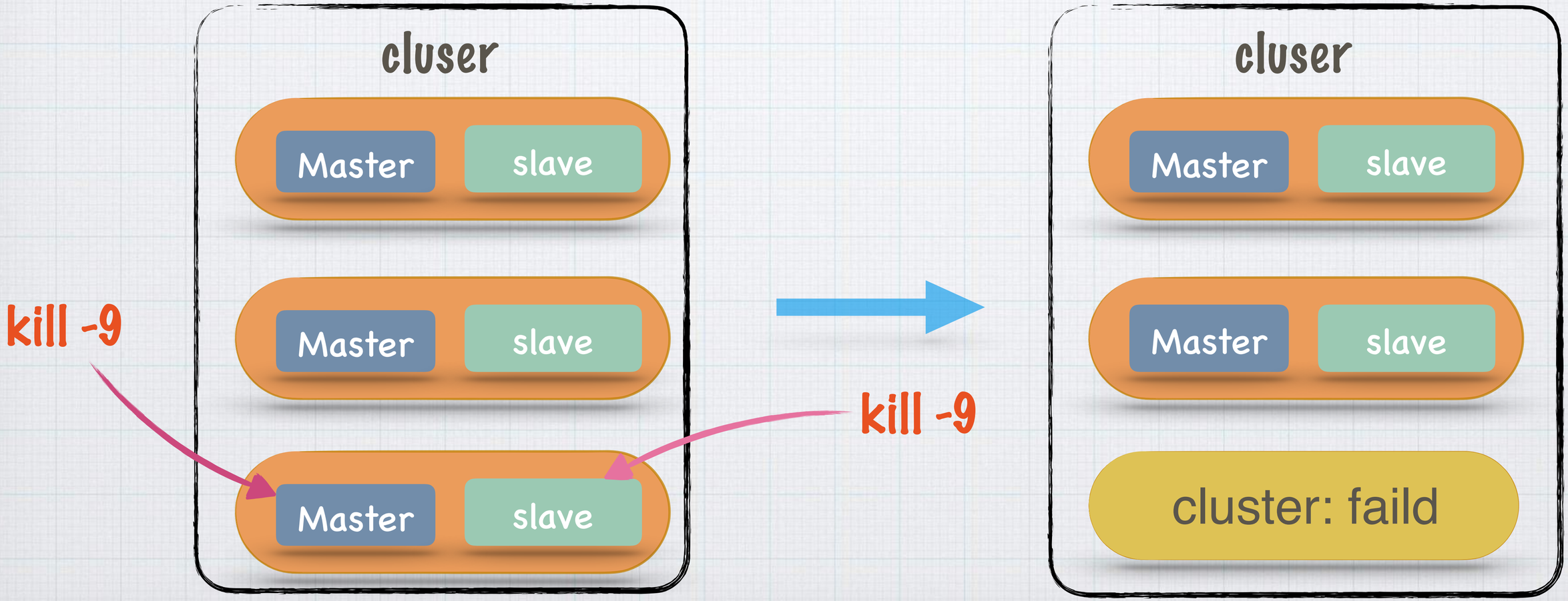
# 高可用性


cluser
Master
slave
Master
slave
Master
slave
kill -9
cluser
Master
slave
Master
slave
Master

# 高可用性？

cluster-require-full-coverage ?

```
127.0.0.1:8504> get t9
(error) CLUSTERDOWN The cluster is down
127.0.0.1:8504> set t1 val
(error) CLUSTERDOWN The cluster is down
```



# cluster fail原因

* 至少有一个hash slot不可用
* 集群中大部份Master都进入了PFAIL状态(可能失已失效)

# Fail探测

* 节点Fail探测超过timeout会标记为PFAIL
* PFAIL标记会随着gossip传播
* 每次收到心跳包会检测其中对其他节点的PFAIL标记，当做对该节点FAIL的投票维护在本机对
* 某个节点的PFAIL标记达到大多数时，将其变为FAIL标记并广播FAIL消息

# 故障恢复

* slave发现自己的master变为FAIL
* 将自己记录的集群currentEpoch加1，并广播Failover Request信息
* 其他节点收到该信息，只有各个master响应，判断请求者的合法性，并发送FAILOVER_AUTH_ACK，对每一个epoch只发送一次ack
* 尝试failover的slave收集FAILOVER_AUTH_ACK
* 超过半数后变成新Master
* 广播Pong通知其他集群节点

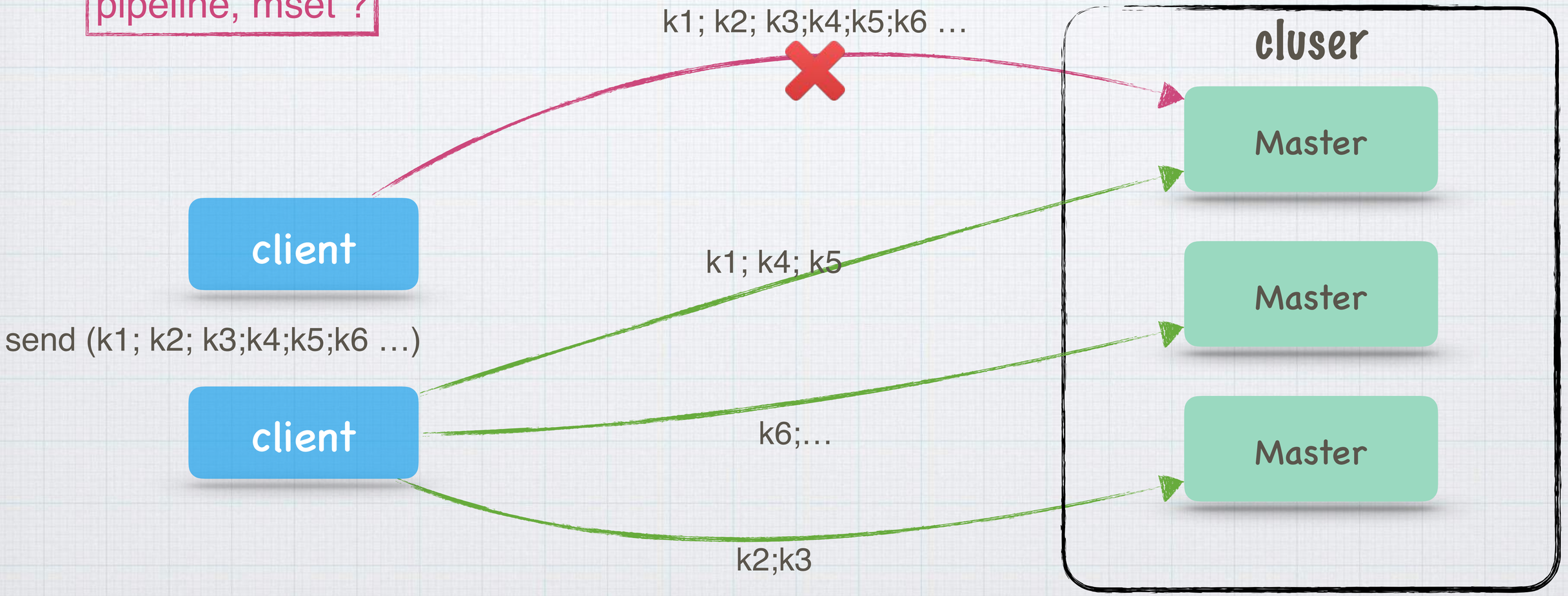
# 读写分离 ？

```
127.0.0.1:8504> cluster nodes
6d88a412ed0403740a84cd535ba6a095ca0a47b0 127.0.0.1:8505@18505 slave a34f7f434d6a7afee2d4bcc2e8576975cfb8e14
ted
07994e369c731a56f1a6720675805726 2eb5ae31 127.0.0.1:8506@18506 slave 0425e1919319c1045163b53763df069bb99f647
ted
48beae41a8f1433cb144416babf17dd60a4d9adb 127.0.0.1:8504@18504 myself,slave c9d9d031e73c6dc033e02c13c0bd5643
 connected
0425e1919319c1045163b53763df069bb99f647a 127.0.0.1:8501@18501 master - 0 1527586978069 1 connected 0-5460
a34f7f434d6a7afee2d4bcc2e8576975cfb8e144 127.0.0.1:8503@18503 master - 0 1527586979071 3 connected 10923-16
c9d9d031e73c6dc033e02c13c0bd5643f2309781 127.0.0.1:8502@18502 master - 0 1527586979572 2 connected 5461-109
127.0.0.1:8504> readonly
OK
127.0.0.1:8504> get t5
"111111"
```

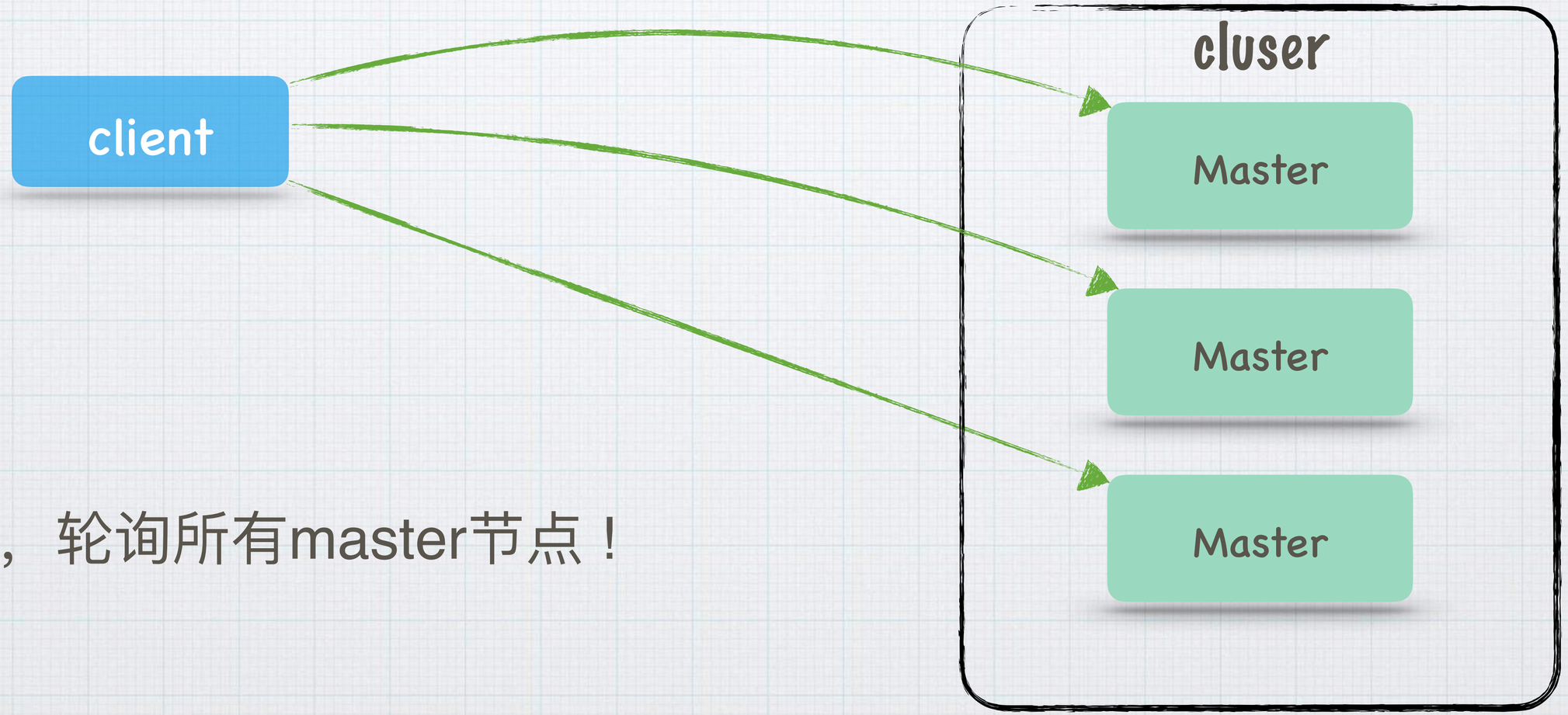

就是想读？基于连接的readonly !!!

# 批量？

官方不推荐
pipeline, mset ?



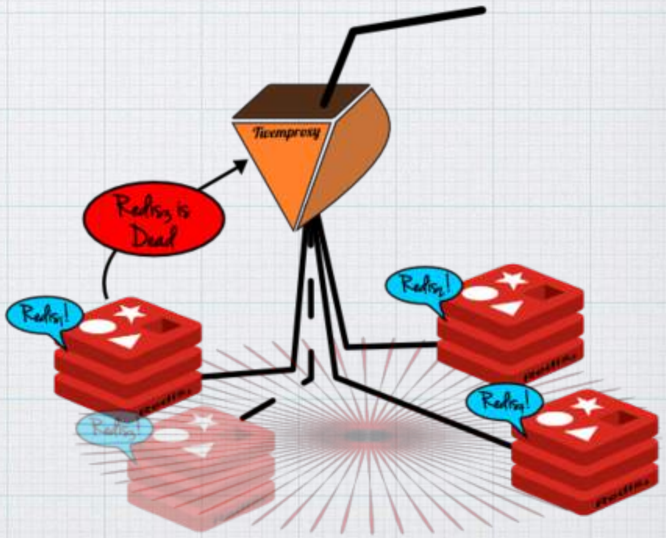
# find bigkey in cluster




对的，轮询所有master节点！

# link

* https://redis.io/topics/cluster-tutorial
* https://redis.io/topics/cluster-spec



redis

“别说话！”

–峰云就她了

redis