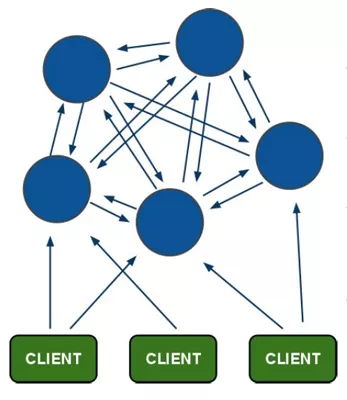
# <https://www.cnblogs.com/esofar/p/10486621.html>

# [分布式缓存 Redis 集群搭建](https://www.cnblogs.com/esofar/p/10486621.html)

## **Redis 集群简介**

Redis Cluster 即 Redis 集群，是 Redis 官方在 3.0 版本推出的一套分布式存储方案。完全去中心化，由多个节点组成，所有节点彼此互联。Redis 客户端可以直接连接任何一节点获取集群中的键值对，不需要中间代理，如果该节点不存在用户所指定的键值，其内部会自动把客户端重定向到键值所在的节点。

Redis 集群是一个网状结构，每个节点都通过 TCP 连接跟其他每个节点连接。在一个有 N 个节点的集群中，每个节点都有 N-1 个流出的 TCP 连接，和 N-1 个流入的连接，这些 TCP 连接会永久保持。



Redis Cluster 同其他分布式存储系统一样，主要具备以下两个功能：

### 数据分区

Redis 集群会将用户数据分散保存至各个节点中，突破单机 Redis 内存最大存储容量。集群引入了 哈希槽slot的概念，其搭建完成后会生 16384 个哈希槽slot，同时会根据节点的数量大致均等的将 16384 个哈希槽映射到不同的节点上。当用户存储key-value时，集群会先对key进行 CRC16 校验然后对 16384 取模来决定key-value放置哪个槽，从而实现自动分割数据到不同的节点上。

### 数据冗余

Redis 集群支持主从复制和故障恢复。集群使用了主从复制模型，每个主节点master应至少有一个从节点slave。假设某个主节点故障，其所有子节点会广播一个数据包给集群里的其他主节点来请求选票，一旦某个从节点收到了大多数主节点的回应，那么它就赢得了选举，被推选为主节点，负责处理之前旧的主节点负责的哈希槽。

## **搭建 Redis 集群**

进入正题。

依据 Redis Cluster 内部故障转移实现原理，Redis 集群至少需要 3 个主节点，而每个主节点至少有 1 从节点，因此搭建一个集群至少包含 6 个节点，三主三从，并且分别部署在不同机器上。

条件有限，测试环境下我们只能在一台机器上创建一个伪集群，通过不同的 TCP 端口启动多个 Redis 实例，组成集群。

目前 Redis Cluster 的搭建有两种方式：

1. 手动方式搭建，即手动执行 cluster 命令，一步步完成搭建流程。
2. 自动方式搭建，即使用官方提供的集群管理工具快速搭建。

两种方式原理一样，自动搭建方式只是将手动搭建方式中需要执行的 Redis 命令封装到了可执行程序。生产环境下推荐使用第二种方式，简单快捷，不易出错。不过本文实战演示两种方式都会提及。

### **手动方式搭建**

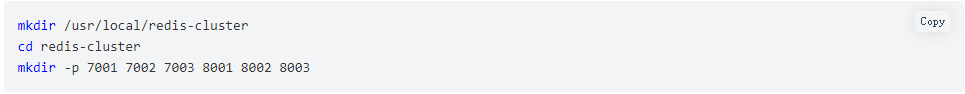
#### **启动节点**

搭建集群的第一步就是要先把参与搭建集群的每个节点启动起来。

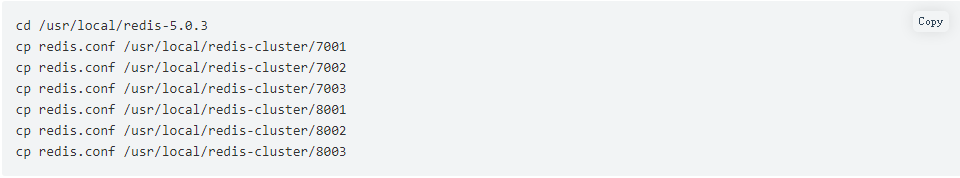
由于我们这是在一台机器上模拟多个节点，可以预先规划下各个节点的属性：

| **节点编号** | **IP 地址** | **TCP 端口** | **节点类型** | **从节点** | **启动配置** |
| --- | --- | --- | --- | --- | --- |
| A | 127.0.0.1 | 7001 | 主 | D | /usr/local/redis-cluster/7001/redis.conf |
| B | 127.0.0.1 | 7002 | 主 | E | /usr/local/redis-cluster/7002/redis.conf |
| C | 127.0.0.1 | 7003 | 主 | F | /usr/local/redis-cluster/7003/redis.conf |
| D | 127.0.0.1 | 8001 | 从 | / | /usr/local/redis-cluster/8001/redis.conf |
| E | 127.0.0.1 | 8002 | 从 | / | /usr/local/redis-cluster/8002/redis.conf |
| F | 127.0.0.1 | 8003 | 从 | / | /usr/local/redis-cluster/8003/redis.conf |

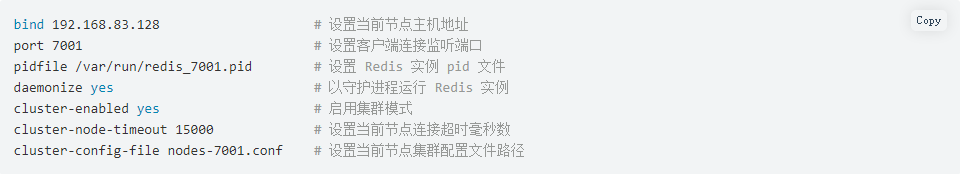
根据上述规划，可以先通过如下命令创建各个节点启动配置文件的存放目录。



顺序执行如下行命令，进入 Redis 源码包目录并将默认配置文件redis.conf分别复制到六个节点配置存放目录中，作为各自节点启动配置文件。



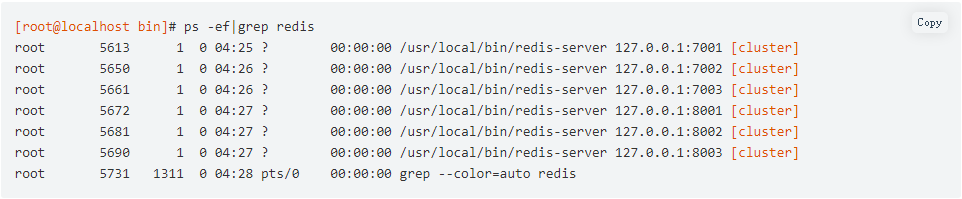
接下来需要分别修改每个节点的配置文件。下面贴的是节点 A 的配置文件/usr/local/redis-cluster/7001/redis.conf中启用或修改的一些必要参数。其他节点 B、C、D、E、F 参照修改，注意把涉及端口的地方修改成各自节点预先规划的即可。



完成上述工作就可以通过如下几组命令启动待搭建集群中的 6 个节点了。



最后通过ps -ef|grep redis命令确认各个节点服务是否已经正常运行。



如上输出可以看出上面规划的 6 个节点都成功启动。

#### **节点握手[#](https://www.cnblogs.com/esofar/p/10486621.html" \l "1404515827)**

虽然上面 6 个节点都启用了群集支持，但默认情况下它们是不相互信任或者说没有联系的。节点握手就是在各个节点之间创建链接（每个节点与其他节点相连），形成一个完整的网格，即集群。

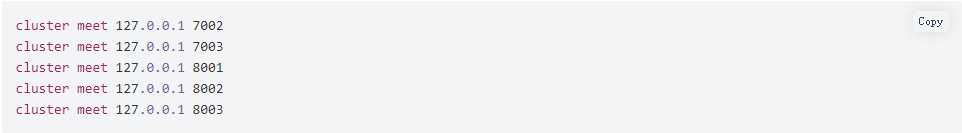
节点握手的命令如下：



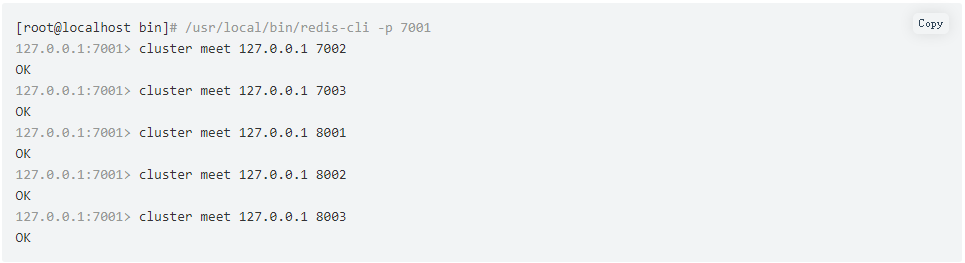
但为了创建集群，不需要发送形成完整网格所需的所有 [cluster meet](https://redis.io/commands/cluster-meet) 命令。只要能发送足够的cluster meet消息，可以让每个节点都可以通过一系列已知节点到达每个其他节点，缺失的链接将被自动创建。

例如，如果我们通过cluster meet将节点 A 与节点 B 连接起来，并将 B 与 C 连接起来，则 A 和 C 会自己找到握手方式并创建链接。

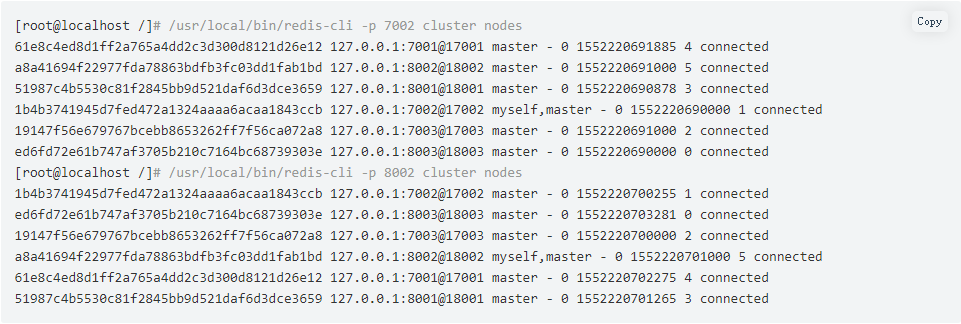
我们的创建的 6 个节点可以通过 redis-cli 连接到 A 节点执行如下五组命令完成握手，生产环境需要将 IP 127.0.0.1替换成外网 IP。



如上述命令正常执行输出结果如下。



接下来可以通过 [cluster nodes](https://redis.io/commands/cluster-nodes) 命令查看节点之间 的链接状态。我随机找了两个节点 B 和 F 测试，输出结果如下所示。

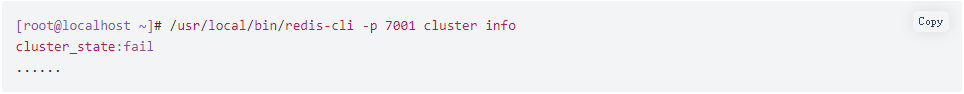


可以看到，节点 B 和节点 F 都已经分别和其他 5 个节点建立链接。

至此，节点握手完成。

#### **分配槽位[#](https://www.cnblogs.com/esofar/p/10486621.html" \l "2368706452)**

此时 Redis 集群还并没有处于上线状态，可以在任意一节点上执行 [cluster info](https://redis.io/commands/cluster-info) 命令来查看目前集群的运行状态。

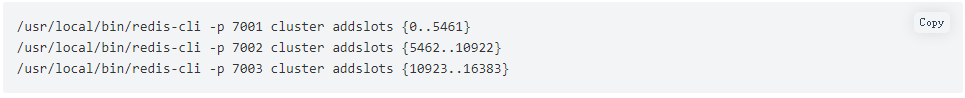


上面输出cluster\_state:fail表示当前集群处于下线状态。因为只有给集群中所有**主节点**分配好槽位（即哈希槽slot，本文第一小节有提及）集群才能上线。

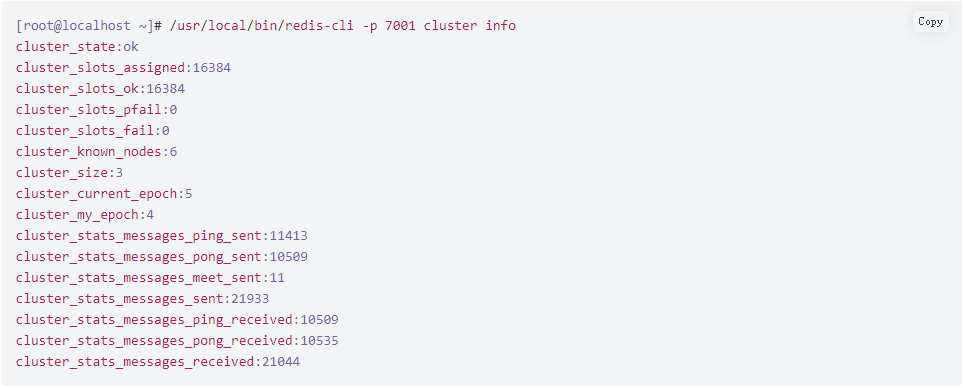
分配槽位的命令如下：



根据预先规划，这一步需要使用 [cluster addslots](http://www.redis.cn/commands/cluster-addslots.html) 命令手动将 16384 个哈希槽大致均等分配给**主节点** A、B、C。



上面三组命令执行完毕，可以再次查看目前集群的一些运行参数。



如上输出cluster\_state:ok证明 Redis 集群成功上线。

#### **主从复制[#](https://www.cnblogs.com/esofar/p/10486621.html" \l "3379914285)**

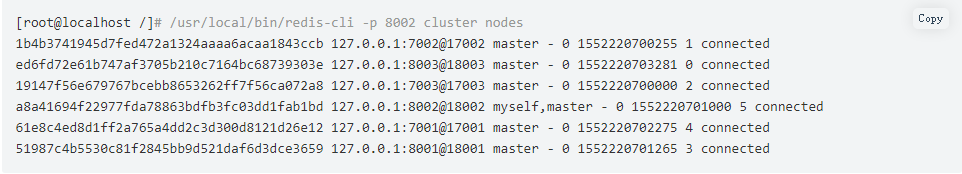
Redis 集群成功上线，不过还没有给主节点指定从节点，此时如果有一个节点故障，那么整个集群也就挂了，也就无法实现高可用。

集群中需要使用 [cluster replicate](https://redis.io/commands/cluster-replicate) 命令手动给从节点配置主节点。

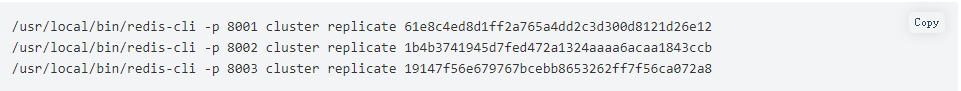
集群复制命令如下：



集群中各个节点的node-id可以用cluster nodes命令查看，如下输出1b4b3741945d7fed472a1324aaaa6acaa1843ccb即是主节点 B 的node-id。

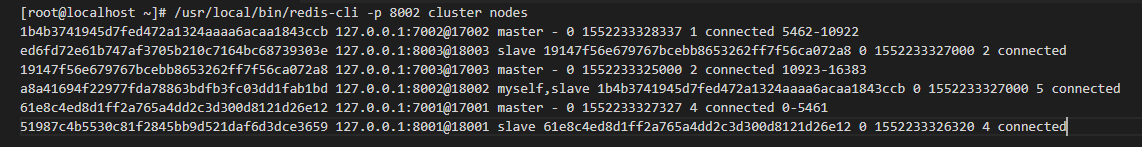


根据预先规划，A主D从；B主E从；C主F从。执行如下三组命令分别为从节点 D、E、F 指定其主节点，使群集可以自动完成主从复制。



命令执行成功后，我们便算以手动方式成功搭建了一个 Redis 集群。

最后，再来查看一下集群中的节点信息。



### **自动方式搭建[#](https://www.cnblogs.com/esofar/p/10486621.html" \l "1455434105)**

Redis 3.0 版本之后官方发布了一个集群管理工具 redis-trib.rb，集成在 Redis 源码包的src目录下。其封装了 Redis 提供的集群命令，使用简单、便捷。

不过 redis-trib.rb 是 Redis 作者使用 Ruby 语言开发的，故使用该工具之前还需要先在机器上安装 Ruby 环境。后面作者可能意识到这个问题，Redis 5.0 版本开始便把这个工具集成到 redis-cli 中，以--cluster参数提供使用，其中create命令可以用来创建集群。

#### **启动节点[#](https://www.cnblogs.com/esofar/p/10486621.html" \l "2468730903)**

使用集群管理工具搭建集群之前，也是需要先把各个节点启动起来的。节点的启动方式请参见本文「手动方式创建」-「启动节点」一节，此处不再赘述。

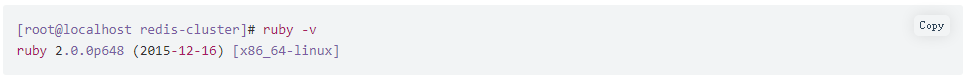
#### **集群管理工具搭建[#](https://www.cnblogs.com/esofar/p/10486621.html" \l "3310245629)**

如果您安装的 Redis 是 3.x 和 4.x 的版本可以使用 redis-trib.rb 搭建，不过之前需要安装 Ruby 环境。

先使用 yum 安装 Ruby 环境以及其他依赖项。



确认安装版本。



再使用 redis-trib.rb 脚本搭建集群，具体命令如下所示。

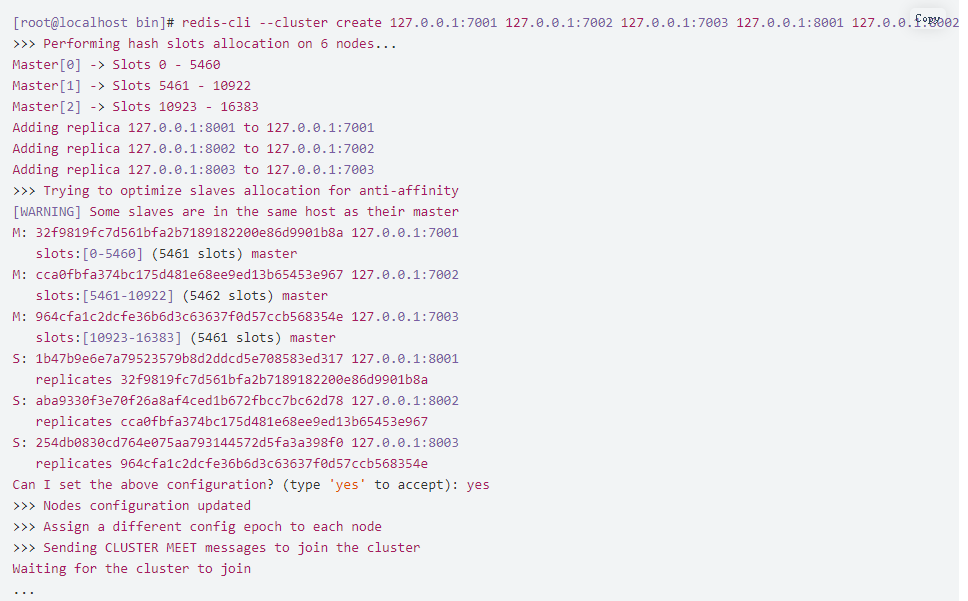
/usr/local/redis-5.0.3/src/redis-trib.rb create --replicas 1 127.0.0.1:7001 127.0.0.1:7002 127.0.0.1:7003 127.0.0.1:8001 127.0.0.1:8002 127.0.0.1:8003

不过，本文实验环境使用的 Redis 版本是 5.0.3，所以我可以直接使用redis-cli --cluster create命令搭建，具体命令如下所示。

/usr/local/bin/redis-cli --cluster create 127.0.0.1:7001 127.0.0.1:7002 127.0.0.1:7003 127.0.0.1:8001 127.0.0.1:8002 127.0.0.1:8003 --cluster-replicas 1

主节点在前，从节点在后。其中--cluster-replicas参数用来指定一个主节点带有的从节点个数，如上--cluster-replicas 1即表示 1 个主节点有 1 个从节点。

命令执行成功会有类似如下输出。





OK，搭建完成！一条命令搞定。