**Redis搭建（ubuntu）**

1. 软件准备

Redis-3.2.9

Redis-3.3.1.gem

Ruby-2.3

Ubuntu-16.04.1

二、单机环境

1、下载源码包redis-3.2.9.tar.gz，并上传到服务器目录/opt

tar xzf redis-3.2.9.tar.gz

cd redis-3.2.9

make

2、刚安装好，redis处于保护模式，只能本地链接，我们需要修改配置文件../redis.conf

2.1、如果测试的话，可以先关闭测试服务器防火墙，如果正式线环境，必须开启防火墙，放开redis端口

2.2、打开配置文redis.conf，修改bind项

bind 本机ip

2.3、Redis默认不是以守护进程的方式运行，可以通过该配置项修改，使用yes启用守护进程，设置为no不是以守护进程的方式运行

daemonize no

2.3、去掉保护模式

protected-mode no

2.4、设置密码登录

#requirepass foobared去掉注释，foobared改为自己的密码

requirepass 123456

3、二进制文件是编译完成后在src目录下，通过下面的命令启动Redis服务：

./redis-server redis.conf

如果启动失败，观察端口是否被占用，如果被占用，杀死占用端口进程，并重新启动：

ps -ef | grep redis | awk ‘{print $2}’ | xargs kill -9

4、你可以使用内置的客户端命令redis-cli进行使用

bin/redis-cli -h 192.168.20.199 -p 6379

使用密码登录

bin/redis-cli -h 192.168.20.199 -p 6379 –a 123456

关闭服务

./src/redis-cli -h 192.168.20.199 –p 6379 -a 123456（带密码）

./src/redis-cli -h 192.168.20.199 –p 6379

三、集群环境

集群规划：由于没有那么多机器，就以单机多实例方式搭建redis集群，与多机的搭建基本一致。

表-1 集群网络规划

|  |  |  |
| --- | --- | --- |
| 机器IP | 端口 | 承担角色 |
| 192.168.20.199 | 9001 | Master对应的slave为9004 |
| 192.168.20.199 | 9002 | Master对应的slave为9005 |
| 192.168.20.199 | 9003 | Master对应的slave为9006 |
| 192.168.20.199 | 9004 | Salve对应的master为9001 |
| 192.168.20.199 | 9005 | Salve对应的master为9002 |
| 192.168.20.199 | 9006 | Salve对应的master为9003 |

搭建过程：

1、创建文件夹

我们计划集群中 Redis 节点的端口号为 9001-9006 ，端口号即集群下各实例文件夹。数据存放在 端口号/data 文件夹中

mkdir /opt/redis-cluster

cd redis-cluster/

mkdir -p 9001/data 9002/data 9003/data 9004/data 9005/data 9006/data

2、复制执行脚本

mkdir redis-cluster/bin

cp ../redis-3.2.9/src/mkreleasehdr.sh ../redis-3.2.9/src/redis-benchmark ../redis-3.2.9/src/redis-check-aof ../redis-3.2.9/src/redis-cli ../redis-3.2.9/src/redis-server ../redis-3.2.9/src/redis-trib.rb /opt/redis-cluster/bin

添加环境变量

sudo vi /etc/profile

export PATH=$PATH:/opt/redis-cluster/bin

立即生效：source /etc/profile

3、复制一个新 Redis 实例

我们现在从已安装好的 Redis 中复制一个新的实例到 9001 文件夹，并修改 redis.conf 配置

cp /opt/redis-3.2.9/\* /opt/redis-cluster/9001

注意，修改 redis.conf 配置：

port 9001（每个节点的端口号）

daemonize yes

bind 192.168.119.131（绑定当前机器 IP）

dir /optl/redis-cluster/9001/data/（数据文件存放位置）

pidfile /var/run/redis\_9001.pid（pid 9001和port要对应）

cluster-enabled yes（启动集群模式）

cluster-config-file nodes9001.conf（9001和port要对应）

cluster-node-timeout 15000

appendonly yes

4、再复制5个redis实例

我们已经完成了一个节点了，其实接下来就是机械化的再完成另外五个节点，其实可以这么做：把 9001 实例 复制到另外五个文件夹中，唯一要修改的就是 redis.conf 中的所有和端口的相关的信息即可，其实就那么四个位置。

\cp -rf /opt/redis-cluster/9001/\* /opt/redis-cluster/9002

\cp -rf /opt/redis-cluster/9001/\* /opt/redis-cluster/9003

\cp -rf /opt/redis-cluster/9001/\* /opt/redis-cluster/9004

\cp -rf /opt/redis-cluster/9001/\* /opt/redis-cluster/9005

\cp -rf /opt/redis-cluster/9001/\* /opt/redis-cluster/9006

\cp -rf不使用别名复制，不用确认。

1. 修改 9002-9006 的 redis.conf 文件

vim redis.conf

:%s/9001/9002g

其实我们也就是替换了下面这四行：

port 9002

dir /opt/redis-cluster/9002/data/

cluster-config-file nodes-9002.conf

pidfile /var/run/redis\_9002.pid

6、启动 9001-9006 六个节点

redis-server 9001/redis.conf

redis-server 9002/redis.conf

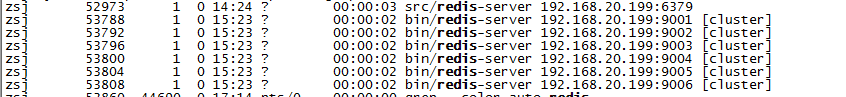
redis-server 9003/redis.conf

redis-server 9004/redis.conf

redis-server 9005/redis.conf

redis-server 9006/redis.conf

可以检查一下是否启动成功：ps -el | grep redis



出现图中的情况，证明都启动成功了。

7、找一个节点测试试

redis-cli -h 192.168.119.131 -p 9001

set name mafly

这时会报错，(error) CLUSTERDOWN Hash slot not served（不提供集群的散列槽），

这是因为虽然我们配置并启动了 Redis 集群服务，但是他们暂时还并不在一个集群中，互相直接发现不了，而且还没有可存储的位置，就是所谓的slot（槽）。

1. 安装集群所需软件

由于 Redis 集群需要使用Redis 集群命令行工具 redis-trib ，所以我们需要安装 ruby 和相关接口。

sudo apt-get install ruby

安装redis的gem，上传redis-3.3.1.gem到/opt/目录

cd /opt/

sudo gem install redis-3.3.1.gem

然后启动集群

redis-trib.rb create --replicas 1 192.168.20.199:9001 192.168.20.199:9002 192.168.20.199:9003 192.168.20.199:9004 192.168.20.199:9005 192.168.20.199:9006

备注：调用 ruby 命令来进行创建集群，--replicas 1 表示主从复制比例为 1:1，即一个主节点对应一个从节点；然后，默认给我们分配好了每个主节点和对应从节点服务，以及 solt 的大小，因为在 Redis 集群中有且仅有 16383 个 solt ，默认情况会给我们平均分配，当然你可以指定，后续的增减节点也可以重新分配。

M: 995ecce2ae144561a6c778b044f3bd1c0ba28e3c 192.168.20.199:9001

slots:0-5460 (5461 slots) master

1 additional replica(s)为主节点Id

S:48a01820c422762cd4d52af96fdc78b399cae448 192.168.20.199:9004

slots: (0 slots) slave

replicates 995ecce2ae144561a6c778b044f3bd1c0ba28e3c 从节点下对应主节点Id

目前来看，9001-9003 为主节点，9004-9006 为从节点，并向你确认是否同意这么配置。输入 yes 后，会开始集群创建。

如果redis设置有密码，上一步会报错

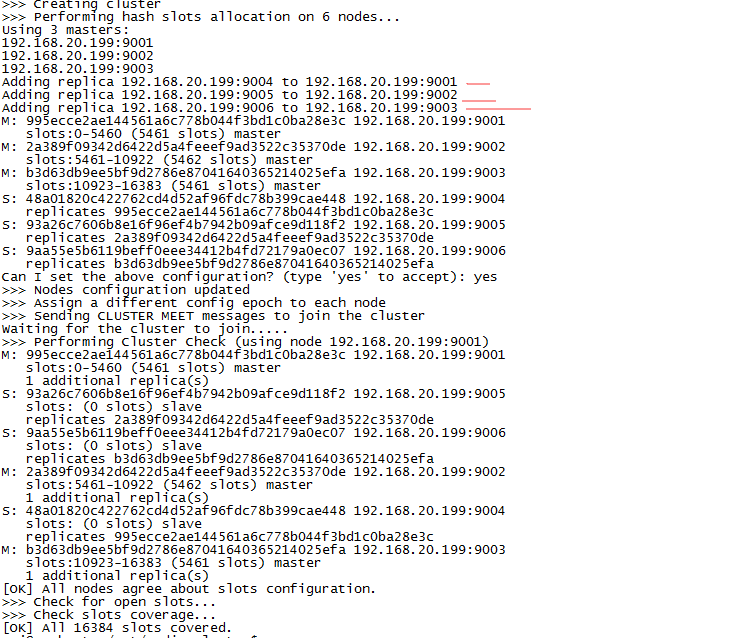


关闭上述的6个redis实例，修改redis.conf注释掉requirepass，然后重新启动6个实例。

再次运行

redis-trib.rb create --replicas 1 192.168.20.199:9001 192.168.20.199:9002 192.168.20.199:9003 192.168.20.199:9004 192.168.20.199:9005 192.168.20.199:9006

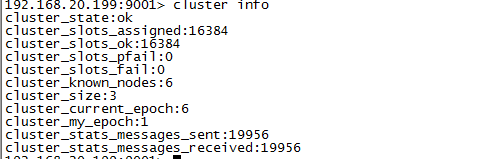
看到下图的提示，就代表集群启动成功了。



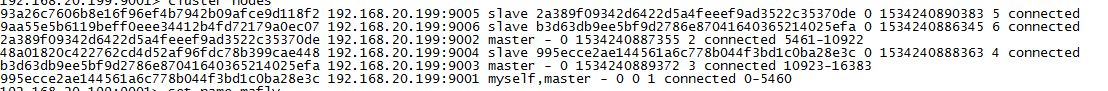
1. 测试

redis-cli -c -h 192.168.20.199 -p 9001

cluster info

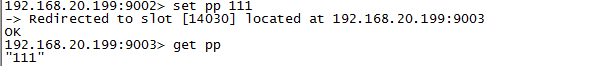


cluster nodes



通过命令，可以详细的看出集群信息和各个节点状态，主从信息以及连接数、槽信息等。

存储一些数据测试：  
你会发现，当我们 set pp 111 时，出现了 Redirected to slot 信息并自动连接到了9003节点。这也是集群的一个数据分配特性。



1. 常用集群命令

集群

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

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，那么先让另一个节点删除该槽>，然后再进行指派。

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

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。

1. 给集群设置密码

1、如果是使用redis-trib.rb工具构建集群，集群构建完成前不要配置密码，集群构建完毕再通过config set + config rewrite命令逐个机器设置密码

2、如果对集群设置密码，那么requirepass和masterauth都需要设置，否则发生主从切换时，就会遇到授权问题，可以模拟并观察日志

3、各个节点的密码都必须一致，否则Redirected就会失败

config set masterauth abc

config set requirepass abc

config rewrite

1. 监控集群

动态扩容、增加节点和减少节点，重新分配槽大小等，当然，还有最重要的就是怎么和我们程序结合起来，以及如何更好的把 Redis 缓存集群发挥出应有的效果，这些才是最重要的。