# redis安装

|  |
| --- |
| wget http://download.redis.io/releases/redis-3.2.4.tar.gz  tar -zxvf redis-3.2.4.tar.gz  cd redis-3.2.4  make  make install PREFIX=/usr/local/redis |

# 单机部署

## 一、服务端配置

直接启动即可：

|  |
| --- |
| ln -s /usr/local/redis/src/redis-server /usr/bin/redis-server  redis-server redis.conf |

## 二、客户端调用

|  |
| --- |
| *//单机模式* **public static void** testStandAlone() {  *// 连接redis服务器* Jedis jedis = **new** Jedis(**"172.16.16.238"**, 6379);  System.***out***.println(**"Connection to server sucessfully"**);   *// 查看服务是否运行* System.***out***.println(**"Server is running: "** + jedis.ping());*//输出PONG则连接成功* System.***out***.println(**"--------------------------------------------"**);   *// 增加一个Key检测是否连接成功* jedis.incr(**"foo11"**);  String value = jedis.get(**"foo11"**);  System.***out***.println(**"foo-->"**+value); } |

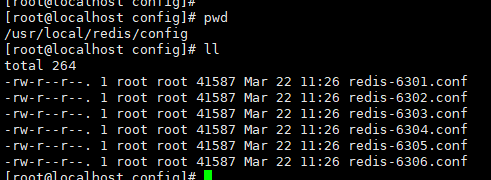
# 集群部署

采用3主3从方案

## 一、服务端配置

|  |
| --- |
| cd /usr/local/redis  mkdir config |

在config目录下创建6个redis.conf配置文件：



分别修改配置，以redis-6301（其它只需要修改端口号即可）为例：

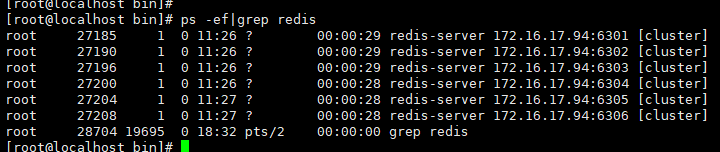
|  |
| --- |
| # 后台运行  daemonize yes  pidfile /var/run/redis-6301.pid  port 6301  bind 172.16.17.94  # 日志文件  logfile "/usr/local/redis/6301.log"  dbfilename dump.rdb  # 开启集群 把注释#去掉  cluster-enabled yes  # 集群的配置  cluster-config-file nodes\_7000.conf  # 请求超时 默认15秒，可自行设置  cluster-node-timeout 15000  appendonly yes  appendfilename "appendonly.aof" |

启动服务：

|  |
| --- |
| cd /usr/local/redis/bin  redis-server ../config/redis-6301.conf  redis-server ../config/redis-6302.conf  redis-server ../config/redis-6303.conf  redis-server ../config/redis-6304.conf  redis-server ../config/redis-6305.conf  redis-server ../config/redis-6306.conf |

查看下服务:

|  |
| --- |
| ps -ef|grep redis |



## 二、关联组建集群

依赖ruby，ruby环境安装：

|  |
| --- |
| # 先安装ruby环境  apt-get install ruby  # 再安装redis需要的插件  gem install redis |

可能会安装失败，具体安装ruby环境，请参考：

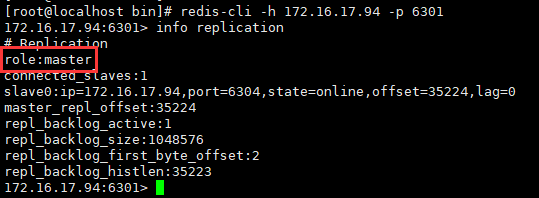
https://www.cnblogs.com/smileyes/p/7489484.html

执行如下命令：

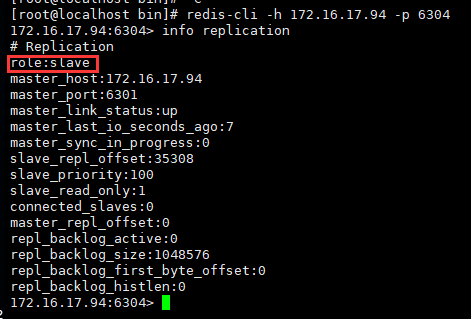
|  |
| --- |
| redis-trib.rb create --replicas 1 172.16.17.94:6301 172.16.17.94:6302 172.16.17.94:6303 172.16.17.94:6304 172.16.17.94:6305 172.16.17.94:6306 |

验证集群：

Master：



Slave：



## 三、客户端调用

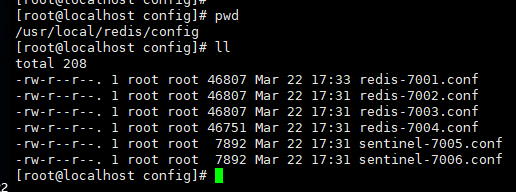
|  |
| --- |
| *//主从集群模式（官方推荐）* **public static void** testMasterSlave() {  JedisPoolConfig poolConfig = **new** JedisPoolConfig();  *// 最大连接数* poolConfig.setMaxTotal(1);  *// 最大空闲数* poolConfig.setMaxIdle(1);  *// 最大允许等待时间，如果超过这个时间还未获取到连接，则会报JedisException异常：  // Could not get a resource from the pool* poolConfig.setMaxWaitMillis(1000);  Set<HostAndPort> nodes = **new** LinkedHashSet<HostAndPort>();  nodes.add(**new** HostAndPort(**"172.16.17.94"**, 6301));  nodes.add(**new** HostAndPort(**"172.16.17.94"**, 6302));  nodes.add(**new** HostAndPort(**"172.16.17.94"**, 6303));  nodes.add(**new** HostAndPort(**"172.16.17.94"**, 6304));  nodes.add(**new** HostAndPort(**"172.16.17.94"**, 6305));  nodes.add(**new** HostAndPort(**"172.16.17.94"**, 6306));   JedisCluster cluster = **new** JedisCluster(nodes, poolConfig);   *//cluster.set("data", new Date().toString());* System.***out***.println(cluster.get(**"name"**));  System.***out***.println(cluster.get(**"age"**));  System.***out***.println(cluster.get(**"data"**));  **try** {  cluster.close();  } **catch** (IOException e) {  e.printStackTrace();  } } |

# 集群部署之哨兵模式

## 一、服务端配置

|  |
| --- |
| cd /usr/local/redis  mkdir config |

在config目录下创建4个redis.conf配置文件，2个sentinel.conf配置文件：



修改redis.conf配置文件，以redis-7001.conf为例，其它只需要修改端口号即可：

|  |
| --- |
| daemonize yes  port 7001  pidfile "/var/run/redis-7001.pid"  #bind 127.0.0.1  logfile "/usr/local/redis/logs/ redis-7001.log"  # 关闭安全模式（3.0之后有的特性）  protected-mode no  # 节点配置文件  cluster-config-file nodes-7001.conf |

修改sentinel.conf配置文件，以sentinel-7005.conf为例，其它只需要修改端口号即可：

|  |
| --- |
| port 7005  sentinel monitor mymaster 172.16.17.96 7001 2  logfile "/usr/local/redis/logs/sentinel-7001.log"  # 关闭安全模式  protected-mode no |

注：我们稍后要启动四个redis实例，其中端口为7001的redis设为master，其他三个设为slave 。所以mymaster 后跟的是master的ip和端口，最后一个’2’代表我要启动只要有2个sentinel认为master下线，就认为该master客观下线，启动failover并选举产生新的master。通常最后一个参数不能多于启动的sentinel实例数。

启动4个redis实例：

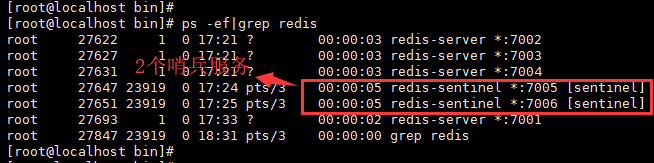
|  |
| --- |
| cd /usr/local/redis/bin  redis-server ../config/redis-6301.conf  redis-server ../config/redis-6302.conf  redis-server ../config/redis-6303.conf  redis-server ../config/redis-6304.conf |

启动2个sentinel哨兵实例：

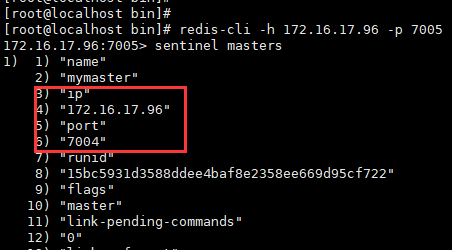
|  |
| --- |
| redis-sentinel ../config/sentinel-7005.conf &  redis-sentinel ../config/sentinel-7006.conf & |

查看下服务:

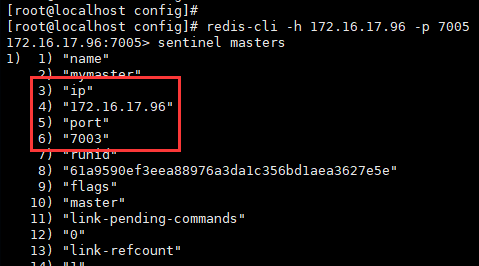
|  |
| --- |
| ps -ef|grep redis |



验证集群服务：



Kill -9 27631，这时master变成了7003：



再次将7004启动，可以看到7004变成了7003的slave。

## 二、客户端调用

|  |
| --- |
| *//哨兵集群模式* **public static void** testSentinels() {  Set<String> sentinels = **new** HashSet<String>();  sentinels.add(**new** HostAndPort(**"172.16.17.96"**, 7005).toString());  sentinels.add(**new** HostAndPort(**"172.16.17.96"**, 7006).toString());  JedisSentinelPool sentinelPool = **new** JedisSentinelPool(**"mymaster"**, sentinels);  System.***out***.println(**"Current master: "** + sentinelPool.getCurrentHostMaster().toString());   Jedis master = sentinelPool.getResource();  *//master.auth("ccc");* master.set(**"username"**,**"xfgeng"**);  String value = master.get(**"username"**);  System.***out***.println(**"username: "** + value);   master.close();  sentinelPool.close();  sentinelPool.destroy(); } |

# 总结

一、密码设置   
方式一：修改所有redis集群中的redis.conf文件加入：   
masterauth passwd123   
requirepass passwd123   
说明：这种方式需要重新启动各节点

方式二：进入各个实例进行设置：   
./redis-cli -c -p 7000   
config set masterauth passwd123   
config set requirepass passwd123   
config rewrite   
之后分别使用./redis-cli -c -p 7001，./redis-cli -c -p 7002…..命令给各节点设置上密码   
注意：各个节点密码都必须一致，否则Redirected就会失败， 推荐这种方式，这种方式会把密码写入到redis.conf里面去，且不用重启