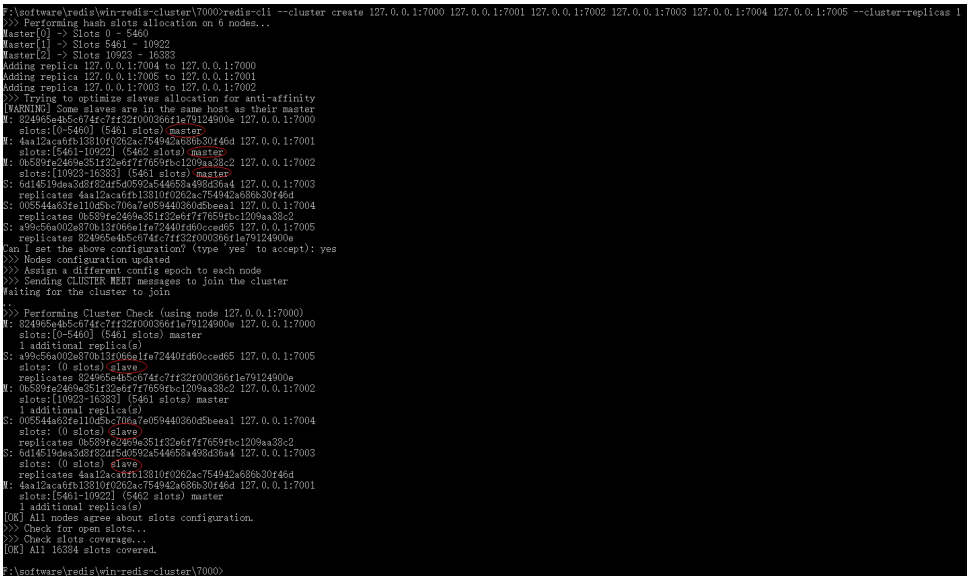
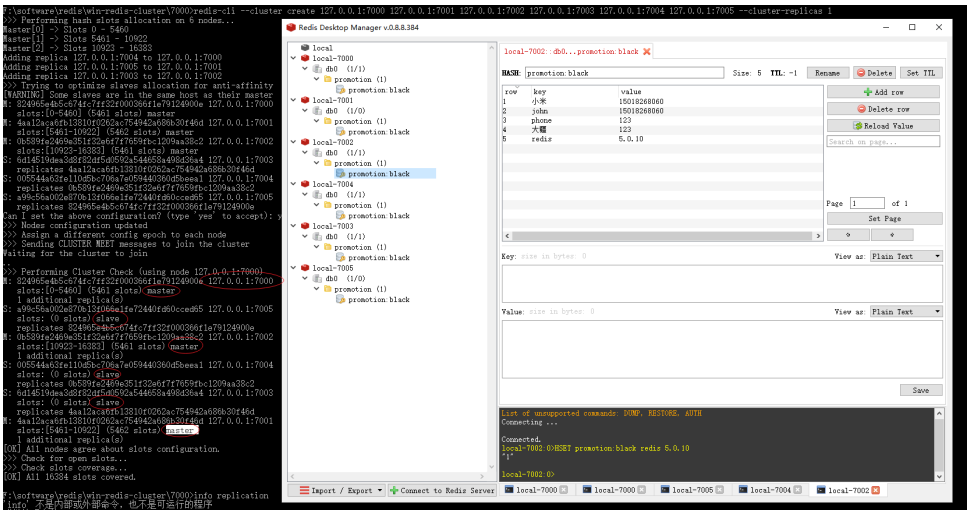
windows redis 集群搭建

一. 准备软件环境

1. windows-redis 5.0.10版本
2. windows10
3. ruby 安装包(针对Redis 版本 3 或 4)
4. 官网地址: https://redis.io/topics/cluster-tutorial#creating-the-cluster
5. redis 集群分布式特性: https://redis.io/topics/cluster-spec
6. 请注意，按预期工作的**最小集群**需要包含至少三个主节点。对于您的第一次测试，强烈建议启动一个具有三个主节点和三个从节点的六节点集群。

二. 创建三主三从集群

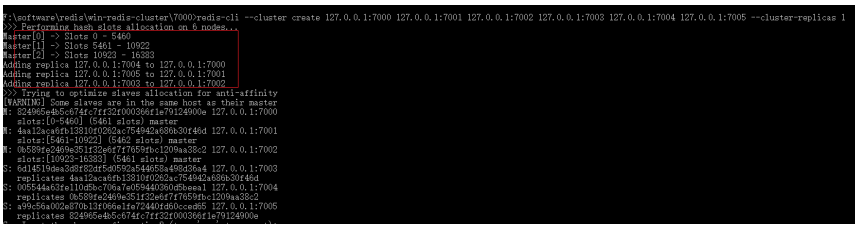
1. 摘取官网的版本区别
    - 如果您使用 Redis 5 或更高版本，这很容易实现，因为我们在 Redis Cluster 命令行实用程序的帮助下嵌入到 中`redis-cli`，可用于创建新集群、检查或重新分片现有集群等。

      对于 Redis 版本 3 或 4，有一个`redis-trib.rb`非常相似的旧工具。您可以`src`在 Redis源代码分发目录中找到它。您需要安装`redis` gem 才能运行`redis-trib`.
    - Redis Cluster 在5.0之后取消了ruby脚本 **redis-trib.rb**的支持（手动命令行添加集群的方式不变），集合到redis-cli里，避免了再安装ruby的相关环境。直接使用redis-clit的参数--cluster 来取代
    - **主节点可读可写，从节点只读不能写**
    - 涉及命令
    - ```
      redis-cli --cluster help
      Cluster Manager Commands:
        create         host1:port1 ... hostN:portN   #创建集群
                       --cluster-replicas <arg>      #从节点个数
        check          host:port                     #检查集群
                       --cluster-search-multiple-owners #检查是否有槽同时被分配给了多个节点
        info           host:port                     #查看集群状态
        fix            host:port                     #修复集群
                       --cluster-search-multiple-owners #修复槽的重复分配问题
        reshard        host:port                     #指定集群的任意一节点进行迁移slot，重新分slots
                       --cluster-from <arg>          #需要从哪些源节点上迁移slot，可从多个源节点完成迁移，以逗号隔开，传递的是节点的node id，还可以直接传递--from all，这样源节点就是集群的所有节点，不传递该参数的话，则会在迁移过程中提示用户输入
                       --cluster-to <arg>            #slot需要迁移的目的节点的node id，目的节点只能填写一个，不传递该参数的话，则会在迁移过程中提示用户输入
                       --cluster-slots <arg>         #需要迁移的slot数量，不传递该参数的话，则会在迁移过程中提示用户输入。
                       --cluster-yes                 #指定迁移时的确认输入
                       --cluster-timeout <arg>       #设置migrate命令的超时时间
                       --cluster-pipeline <arg>      #定义cluster getkeysinslot命令一次取出的key数量，不传的话使用默认值为10
                       --cluster-replace             #是否直接replace到目标节点
        rebalance      host:port                                  #指定集群的任意一节点进行平衡集群节点slot数量
                       --cluster-weight <node1=w1...nodeN=wN>     #指定集群节点的权重
                       --cluster-use-empty-masters                #设置可以让没有分配slot的主节点参与，默认不允许
                       --cluster-timeout <arg>                    #设置migrate命令的超时时间
                       --cluster-simulate                         #模拟rebalance操作，不会真正执行迁移操作
                       --cluster-pipeline <arg>                   #定义cluster getkeysinslot命令一次取出的key数量，默认值为10
                       --cluster-threshold <arg>                  #迁移的slot阈值超过threshold，执行rebalance操作
                       --cluster-replace                          #是否直接replace到目标节点
        add-node       new_host:new_port existing_host:existing_port  #添加节点，把新节点加入到指定的集群，默认添加主节点
                       --cluster-slave                            #新节点作为从节点，默认随机一个主节点
                       --cluster-master-id <arg>                  #给新节点指定主节点
        del-node       host:port node_id                          #删除给定的一个节点，成功后关闭该节点服务
        call           host:port command arg arg .. arg           #在集群的所有节点执行相关命令
        set-timeout    host:port milliseconds                     #设置cluster-node-timeout
        import         host:port                                  #将外部redis数据导入集群
                       --cluster-from <arg>                       #将指定实例的数据导入到集群
                       --cluster-copy                             #migrate时指定copy
                       --cluster-replace                          #migrate时指定replace
        help
      For check, fix, reshard, del-node, set-timeout you can specify the host and port of any working node in the cluster.
      ```
2. 开始搭建
    1. 官网建议3个主节点，3个从节点，高可用性更稳定
      ```

2. 创建目录win-redis-cluster，再创建节点文件夹分别命名 7000, 7001, 7002，7003，7004，7005将redis安装包内容复制到6个文件夹
3. 修改各节点的配置文件redis.windows.conf (cluster-config-file 需按照端口区分文件名)
4. 设置主从节点，这里选择7000为主节点，7001,7002为从节点
    1. 不设置主从节点, 由执行搭建集群命令后自行分配

    ```
    redis-cli --cluster create 127.0.0.1:7000 127.0.0.1:7001
    127.0.0.1:7002 127.0.0.1:7003 127.0.0.1:7004
    127.0.0.1:7005 --cluster-replicas 1
    ```

    1. 打开每个节点的配置文件搜索并设置

    ```
    port 7000
    cluster-enabled yes
    cluster-config-file nodes-7000.conf
    cluster-node-timeout 5000
    appendonly yes
    ```

    其中cluster-config-file 需要安装端口区分命名文件名!!! cluster-enabled 必须是yes !!!
5. 执行各节点服务启动命令
    - 在各文件夹执行cmd
    - **redis-server.exe redis.windows.conf**
    - 生成文件log.txt，nodes-7002.conf(存储集群节点信息)
6. 注意

    在启动集群的时候redis数据库必须是空的，否则会报错提示：[ERR] Node 127.0.0.1:7001 is not empty. Either the node already knows other nodes (check with CLUSTER NODES) or contains some key in database 0。

    解决方法：删除生成的配置文件nodes.conf，如果不行则说明现在创建的结点包括了旧集群的结点信息，需要删除redis的持久化文件后再重启redis，比如：appendonly.aof、dump.rdb。
7. 修改日志输入到控制台
    - Redis默认的设置为verbose，开发测试阶段可以用debug，生产模式一般选用notice

      \1. debug：会打印出很多信息，适用于开发和测试阶段

      \2. verbose（冗长的）：包含很多不太有用的信息，但比debug要清爽一些

      \3. notice：适用于生产模式

      \4. warning : 警告信息
    - 打开配置文件，loglevel notice 改为 loglevel debug
    - logfile log.txt 改为 logfile ""
    - 重启启动后打印日志在控制台

8. 创建集群主从节点,设置主节点1个副本， 如果需要集群需要认证，则在最后加入 **-a xx** 即可

```
redis-cli --cluster create 127.0.0.1:7000 127.0.0.1:7001
127.0.0.1:7002 127.0.0.1:7003 127.0.0.1:7004 127.0.0.1:7005 --
cluster-replicas 1
```



9. 使用客户端连接6个节点测试

   key 命名遵循阿里redis使用规范使其产生文件夹区分其他vapp



```
local-7000:0>INFO REPLICATION
"# Replication
role:master
connected_slaves:1
slave0:ip=127.0.0.1,port=7003,state=online,offset=8812,lag=1
master_replid:76f8ef6fd21c666b0407bb9ec005897141da5ef4
master_replid2:0000000000000000000000000000000000000000
master_repl_offset:8812
second_repl_offset:-1
repl_backlog_active:1
repl_backlog_size:1048576
repl_backlog_first_byte_offset:1
repl_backlog_histlen:8812
"
```

1. 发现只有一个0号数据库，默认是16个数据库的
2. 在主节点或者从节点 进入控制台，设置 HSET promotion:black phone 123 后 都可以在其他节点同步到数据
3. 这里显示有问题，在其他节点控制台设置索引，就可以看到起表被加载出来了
4. 新增7007作为master节点
    - 赋值文件夹7007
    - ```
      ./redis-cli --cluster add-node 127.0.0.1:7007 127.0.0.1:7001
      ```
    - 检查节点信息发现没有分配slots

      ```
      ./redis-cli --cluster check 127.0.0.1:7001
      ```
    - 重新分配slots

      ```
      ./redis-cli --cluster reshard 127.0.0.1:7007
      ```
5. 新增7008从节点到7007
    - ```
      ./redis-cli --cluster add-node 127.0.0.1:7008 127.0.0.1:7007 --cluster-slave --cluster-master-id e3ed175cd38c9ea5b7a0827f2be7b8bfa9385ba2
      ```
6. 删除从节点7008
    - ```
      ./redis-cli --cluster del-node 127.0.0.1:7008 609e99ae01ce067323f8c44207f512b5cd3546e2
      ```
7. 删除主节点7007
    1. 先转移slots

       ```
       ./redis-cli --cluster reshard 127.0.0.1:7001
       ```
    2. 分配均匀使用

       ```
       ./redis-cli --cluster rebalance --cluster-threshold 127.0.0.1:7001
       ```
       ```

3. 删除7007主节点

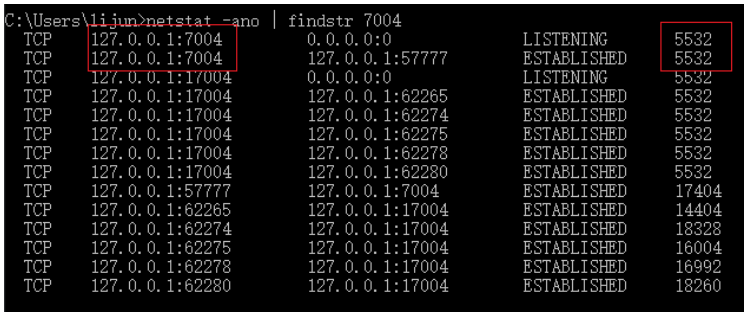
```
./redis-cli --cluster del-node 127.0.0.1:7007
`73f19b384906113507b25f256a781ce184777162
```

8. 测试从节点删除后从节点作为主节点



- 获取7004端口的进程

    1. 执行命令: netstat -ano | findstr 7004



```
C:\Users\lijun>netstat -ano | findstr 7004
  TCP    127.0.0.1:7004         0.0.0.0:0              LISTENING       5532
  TCP    127.0.0.1:7004         127.0.0.1:57777        ESTABLISHED     5532
  TCP    127.0.0.1:17004        0.0.0.0:0              LISTENING       5532
  TCP    127.0.0.1:17004        127.0.0.1:62265        ESTABLISHED     5532
  TCP    127.0.0.1:17004        127.0.0.1:62274        ESTABLISHED     5532
  TCP    127.0.0.1:17004        127.0.0.1:62275        ESTABLISHED     5532
  TCP    127.0.0.1:17004        127.0.0.1:62278        ESTABLISHED     5532
  TCP    127.0.0.1:17004        127.0.0.1:62280        ESTABLISHED     5532
  TCP    127.0.0.1:57777        127.0.0.1:7004         ESTABLISHED     17404
  TCP    127.0.0.1:62265        127.0.0.1:17004        ESTABLISHED     14404
  TCP    127.0.0.1:62274        127.0.0.1:17004        ESTABLISHED     18328
  TCP    127.0.0.1:62275        127.0.0.1:17004        ESTABLISHED     16004
  TCP    127.0.0.1:62278        127.0.0.1:17004        ESTABLISHED     16992
  TCP    127.0.0.1:62280        127.0.0.1:17004        ESTABLISHED     18260
```

- 获取5532进程的进程任务

    1. 执行命令: tasklist|findstr 5532

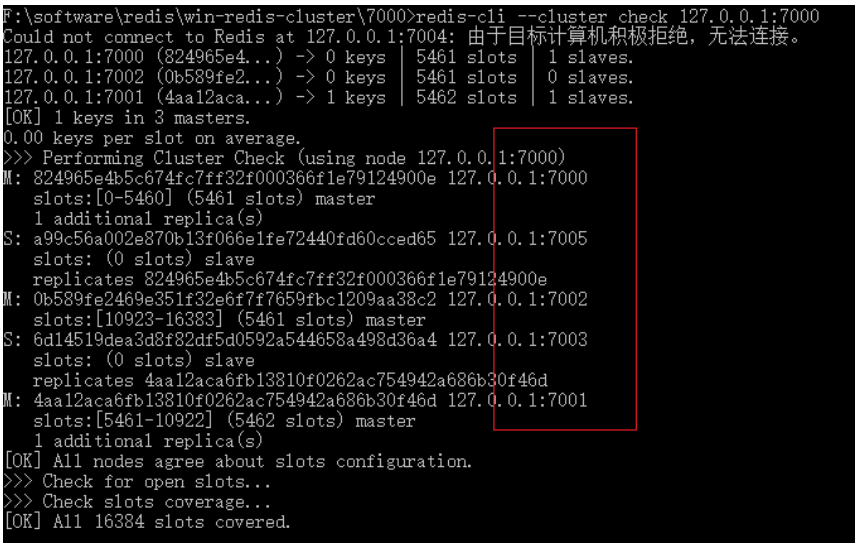
```
C:\Users\lijun>tasklist|findstr 5532
redis-server.exe              5532 Console                    1     11,508 K
```

- 删除5532进程- 结束进程： TASKKILL /F /IM 进程名称 /T 或 TASKKILL /PID 进程ID /T

    1. 执行命令: TASKKILL /F /PID 5532

```
C:\Users\lijun>TASKKILL /F /PID 5532
成功: 已终止 PID 为 5532 的进程。
```

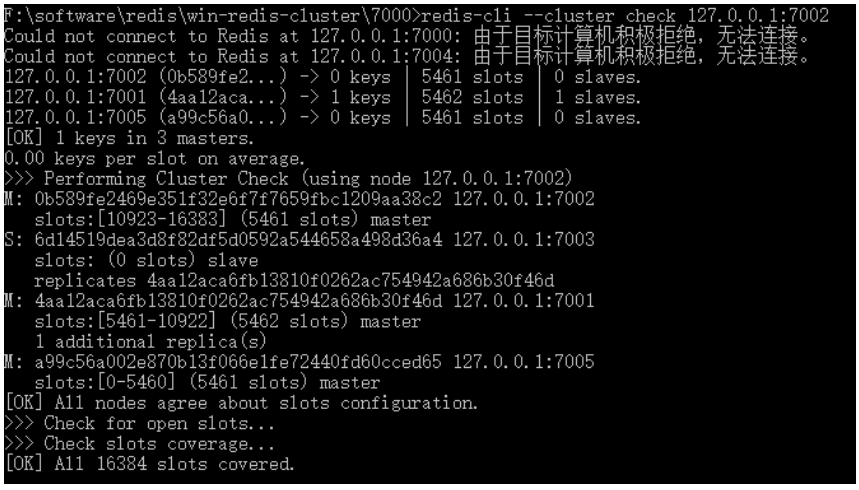
9. 查看节点情况发现7004节点不见了，而7000还是作为主节点



```
F:\software\redis\win-redis-cluster\7000>redis-cli --cluster check 127.0.0.1:7000
Could not connect to Redis at 127.0.0.1:7004: 由于目标计算机积极拒绝，无法连接。
127.0.0.1:7000 (824965e4...) -> 0 keys | 5461 slots | 1 slaves.
127.0.0.1:7002 (0b589fe2...) -> 0 keys | 5461 slots | 0 slaves.
127.0.0.1:7001 (4aa12aca...) -> 1 keys | 5462 slots | 1 slaves.
[OK] 1 keys in 3 masters.
0.00 keys per slot on average.
>>> Performing Cluster Check (using node 127.0.0.1:7000)
M: 824965e4b5c674fc7ff32f000366f1e79124900e 127.0.0.1:7000
   slots:[0-5460] (5461 slots) master
   1 additional replica(s)
S: a99c56a002e870b13f066e1fe72440fd60cced65 127.0.0.1:7005
   slots: (0 slots) slave
   replicates 824965e4b5c674fc7ff32f000366f1e79124900e
M: 0b589fe2469e351f32e6f7f7659fbc1209aa38c2 127.0.0.1:7002
   slots:[10923-16383] (5461 slots) master
S: 6d14519dea3d8f82df5d0592a544658a498d36a4 127.0.0.1:7003
   slots: (0 slots) slave
   replicates 4aa12aca6fb13810f0262ac754942a686b30f46d
M: 4aa12aca6fb13810f0262ac754942a686b30f46d 127.0.0.1:7001
   slots:[5461-10922] (5462 slots) master
   1 additional replica(s)
[OK] All nodes agree about slots configuration.
>>> Check for open slots...
>>> Check slots coverage...
[OK] All 16384 slots covered.
```

10. 重新运行7004节点后新增7004节点到主节点7000作为从节点

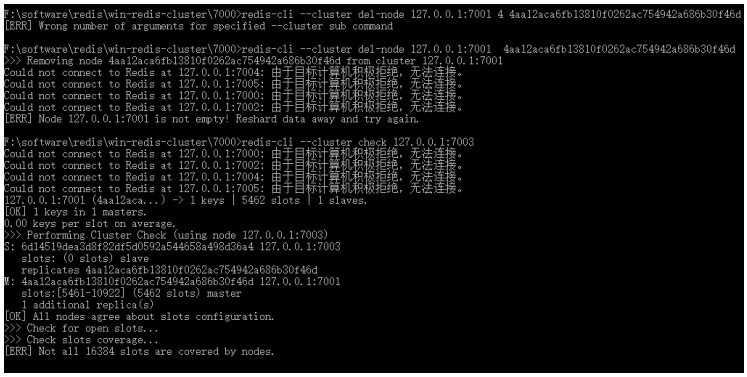
11. 删除7000，7004主节点，发现其从节点会作为主节点继续值守



12. 再删除7002，7005从节点，发现只剩下一对主从节点了

13. 删除最后一个主节点

    1. 执行命令: redis-cli --cluster del-node 127.0.0.1:7001 4 4aa12aca6fb13810f0262ac754942a686b30f46d
    2. 发现删除失败



8. 其他集群命令参考

1. https://www.cnblogs.com/zhoujinyi/p/11606935.html

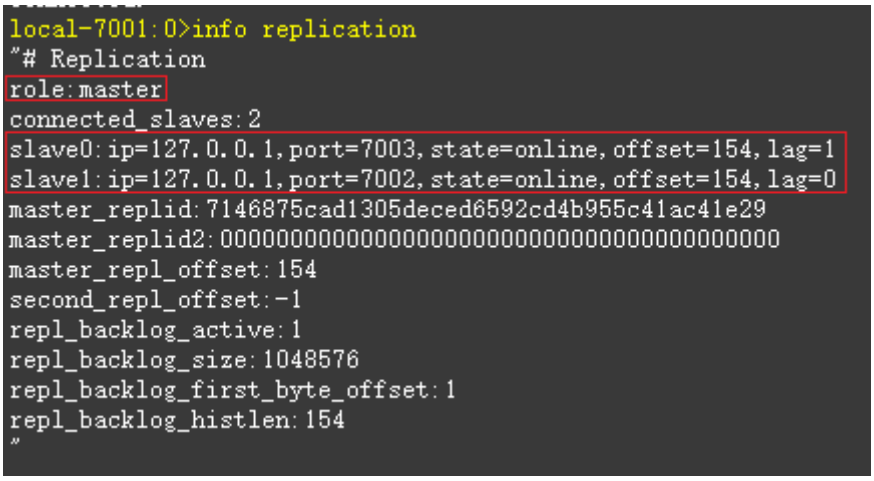
2. https://blog.csdn.net/qq_28289405/article/details/84063921?utm_medium=distribute.pc_relevant.none-task-blog-baidujs_title-1&spm=1001.2101.3001.4242
3. https://www.choupangxia.com/2019/11/07/redis-node-is-not-empty/
4. https://www.cnblogs.com/tanghaorong/p/14339880.html

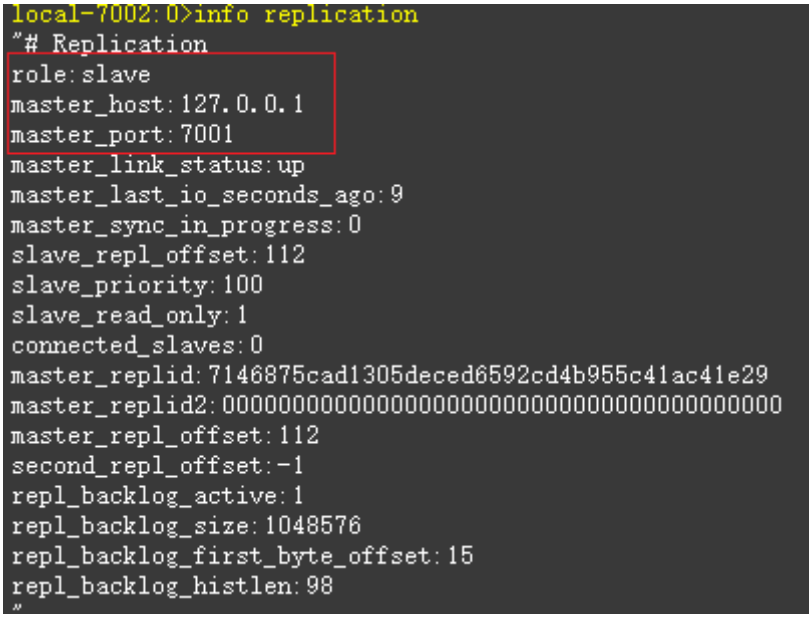
二. 创建一主二从集群

1. 将单机redis程序包复制成3分: 7001,7002,7003 （其中7001作为主节点，7002，7003 作为从节点）
2. 修改配置文件redis.windows.conf

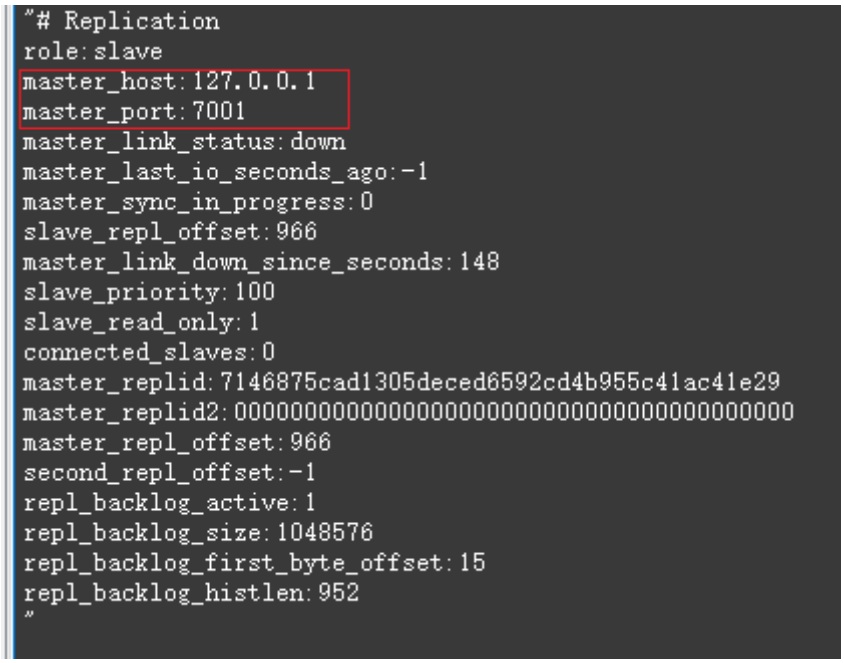
	port	# slaveof	cluster-enabled
Master主节点 7001	7001	不设置	yes
Slaver1 从节点 7002	7002	slaveof 127.0.0.1 7001	no
Slaver1 从节点7003	7003	slaveof 127.0.0.1 7001	no

3. 先启动主节点7001，后启动2个从节点
4. 主节点支持读写，从节点只支持读取，主节点宕掉，从节点还是只是可读，不可写，也不会自动升级为主节点
5. 连接主节点后查看集群关系





6. 发现主节7001点挂了后，从节点7002或7003没有选举新的主节点上任，由此需要加上哨兵机制



三. 搭建一主二从三哨兵机制

0. 基于搭建好一主二从集群

1. 常用命令

1.
```
1. 根据配置文件启动redis：./src/redis-server redis.conf
2. 启动哨兵在src下 ./redis-sentnel ../sentnel.conf          要是启动失败后边加上两个横杠 **--**
3. 启动redis客户端：redis-cli -p port
4. Slaver连接Master：slaveof host：ip
5. 关闭Redis：shutdown
6. 查看主从信息：info Replication
```

2. 在主目录下分别创建sentinel.conf文件
    1. 主节点sentinel.conf

```
port 27001
sentinel monitor zlj-master 127.0.0.1 7001 2
sentinel down-after-milliseconds zlj-master 5000
sentinel parallel-syncs zlj-master 1
sentinel failover-timeout zlj-master 15000
```

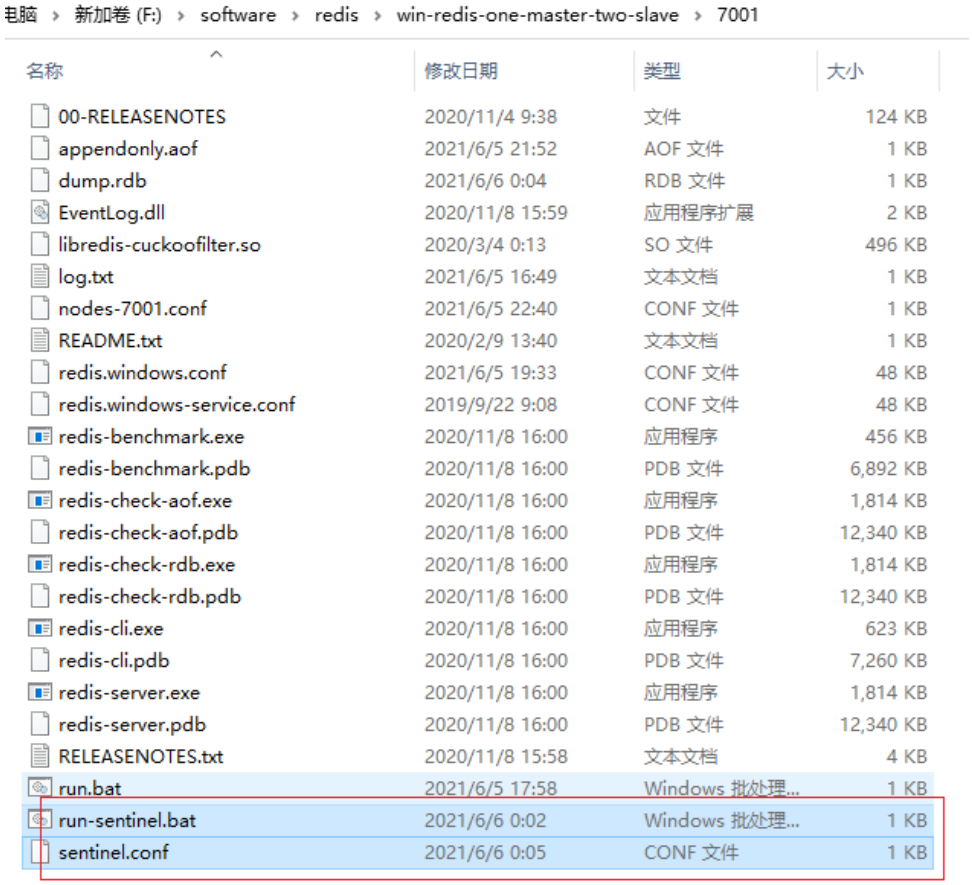
    2. 从节点sentinel7002.conf

```
port 27002
sentinel monitor zlj-master 127.0.0.1 7002 2
sentinel down-after-milliseconds zlj-master 5000
sentinel parallel-syncs zlj-master 1
sentinel failover-timeout zlj-master 15000
```

3. 从节点sentinel7003.conf

```
port 27003
sentinel monitor zlj-master 127.0.0.1 7003 2
sentinel down-after-milliseconds zlj-master 5000
sentinel parallel-syncs zlj-master 1
sentinel failover-timeout zlj-master 15000
```

4. 文件效果

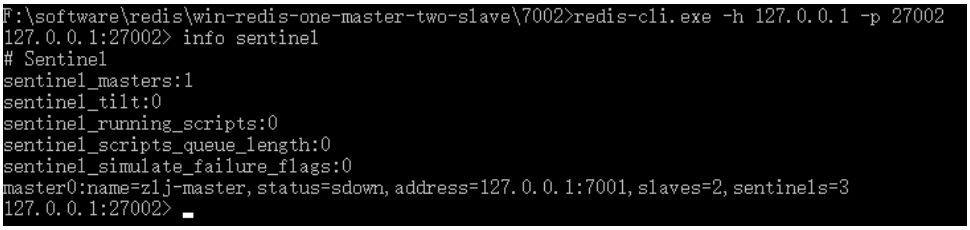


5. 参数解释

```
1. port :当前Sentinel服务运行的端口(和其redis tcp接口一一映射)
2.sentinel monitor mymaster 127.0.0.1 6379 2:Sentinel去监视一个名为mymaster的主redis实例，这个主实例的IP地址为本机地址127.0.0.1，端口号为6379，而将这个主实例判断为失效至少需要2个 Sentinel进程的同意，只要同意Sentinel的数量不达标，自动failover就不会执行
3.sentinel down-after-milliseconds mymaster 5000:指定了Sentinel认为Redis实例已经失效所需的毫秒数。当 实例超过该时间没有返回PING，或者直接返回错误，那么Sentinel将这个实例标记为主观下线。只有一个 Sentinel进程将实例标记为主观下线并不一定会引起实例的自动故障迁移：只有在足够数量的Sentinel都将一个实例标记为主观下线之后，实例才会被标记为客观下线，这时自动故障迁移才会执行
4.sentinel parallel-syncs mymaster 1：指定了在执行故障转移时，最多可以有多少个从Redis实例在同步新的主实例，在从Redis实例较多的情况下这个数字越小，同步的时间越长，完成故障转移所需的时间就越长
5.sentinel failover-timeout mymaster 15000：如果在该时间（ms）内未能完成failover操作，则认为该failover失败
```

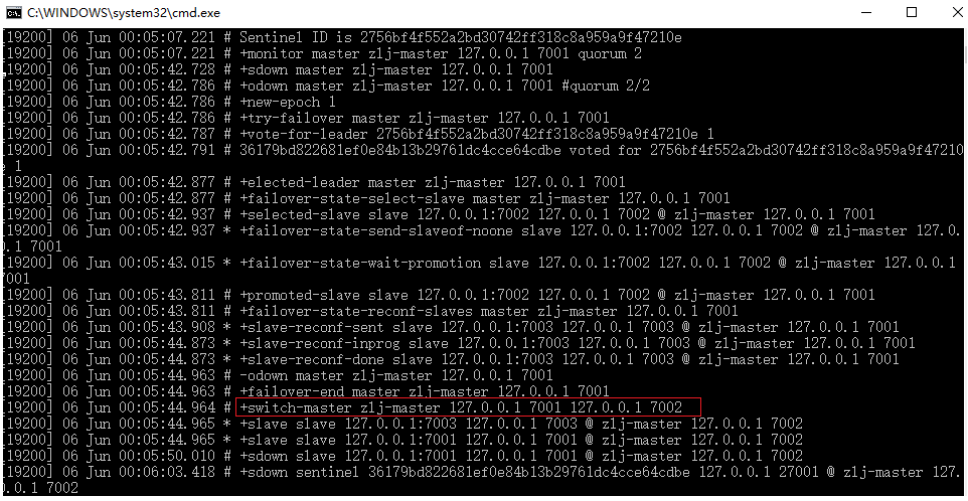
3. 先启动主节点，后从节点redis服务
    1. 分别在目录下执行: redis-server.exe redis.windows.conf
4. 先启动主节点的哨兵，后启动从节点的哨兵服务
    1. 分别在目录下执行命令
        - redis-server.exe sentinel.conf --sentinel
        - redis-server.exe sentinel7002.conf --sentinel
        - redis-server.exe sentinel7003.conf --sentinel
5. 启动效果

6. 查看哨兵状态
    1. 运行命令连接哨兵端口
        - redis-cli.exe -h 127.0.0.1 -p 27002
    2. 查看效果

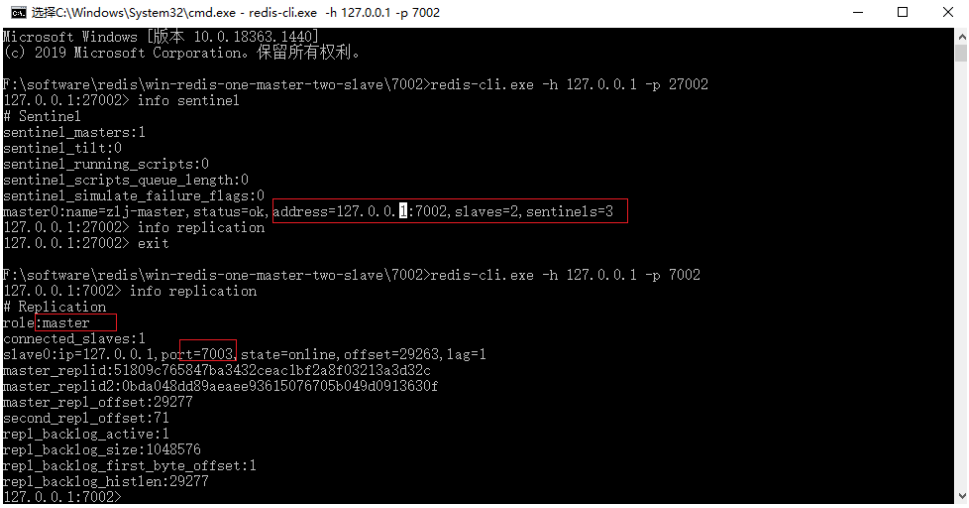


7. 验证主节点宕机后，在其中一个从节点选举出新的主节点
    - 关闭主节点7001的服务

- 可看到主节点转移到7002服务



- 可以看到7002服务以及上升到master主节点，剩下的7003是从服务，由此组成一主一从服务，实现高可用



8. 总结

    1. Master可读可写，Slaver只能读，不能写
    2. Master可以对应多个Slaver，但是数量越多压力越大，延迟就可能越严重
    3. Master写入后立即返回，几乎同时将写入异步同步到各个Slaver，所以基本上延迟可以忽略
    4. 可以通过slaveof no one命令将Slaver升级为Master（当Master挂掉时，手动将某个Slaver变为Master）
    5. 可以通过sentinel哨兵模式监控Master，当Master挂掉时自动选举Slaver变为Master，其它Slaver自动重连新的Master

9. 参考资料

    1. https://www.cnblogs.com/justdoyou/p/10253668.html
    2. https://redis.io/topics/sentinel
    3. https://blog.csdn.net/qq_36881887/article/details/81262425?utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7EBlogCommendFromMachineLearnPai2%7Edefault-2.control&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2%7Edefault%7EBlogCommendFromMachineLearnPai2%7Edefault-2.control