## 1.1是什么

就是我们所说的主从复制，主机数据更新后根据配置和策略，自动同步到备机的master/slaver机制，Master以写为主，Slave以读为主。

## 1.2能干嘛

读写分离

容灾恢复

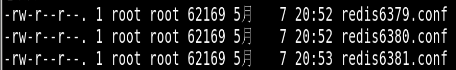
## 1.3怎么玩

1. 配从（库） 不配主（库）
2. 从库配置：slaveof 主库IP 主库端口

每次与master断开之后，都需要重新连接，除非你配置进redis.conf文件。

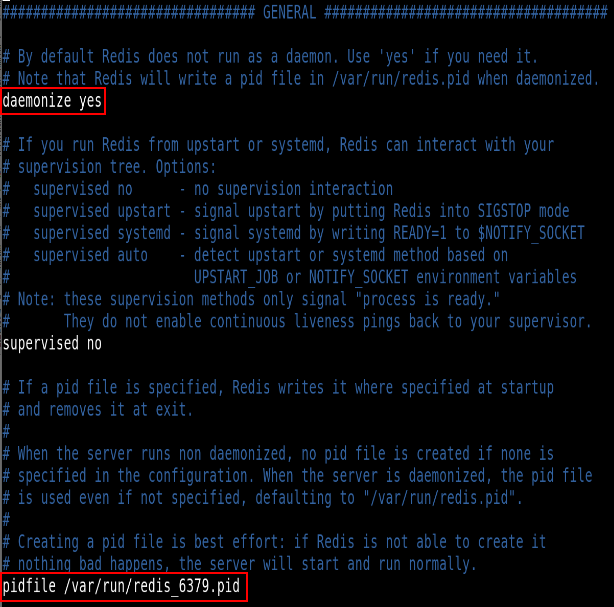
Info replication

1. 修改配置文件细节操作：
   1. 拷贝多个redis.conf文件



* 1. 开启daemonize yes，Pid文件名字以及指定端口





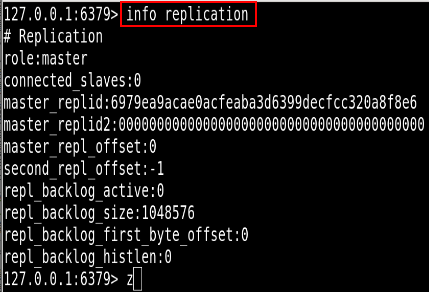
* 1. Log文件名字

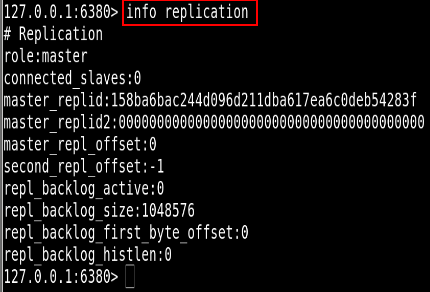


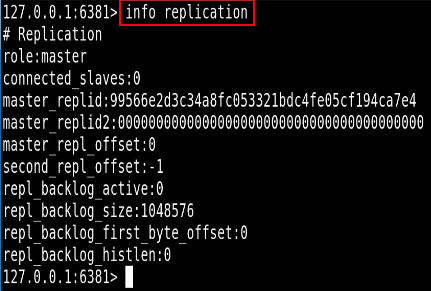
* 1. dump.rdb名字



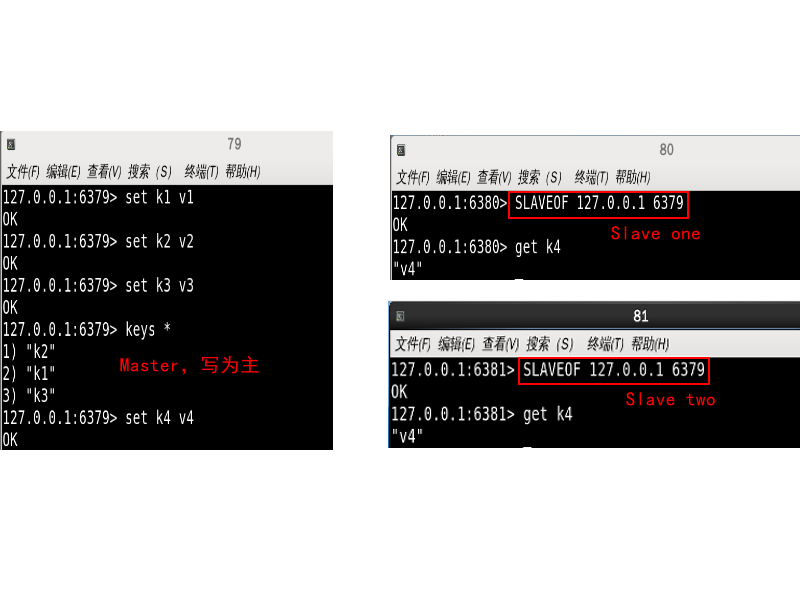
1. 常用3招
   1. 一主二仆
      1. Init







* + 1. 一个Master两个Slave



* + 1. 日志查看
    2. 主从问题演示
       1. 切入点问题？slave1、slave2是从头开始复制还是从切入点开始复制？比如从k4进来，那之前的123是否也可以复制。

从头开始复制。

* + - 1. 从机是否可以写？set可否？

从机只负责读，不可以写。



* + - 1. 主机shutdown后情况如何？从机是上位还是原地待命？

如果没有配置哨兵模式，则是原地待命。

* + - 1. 主机又回来后，主机新增记录，从机还能否顺利恢复值？

能

* + - 1. 其中一台从机down后情况如何？依照原有它能跟上大部队吗？

需要重新连接。

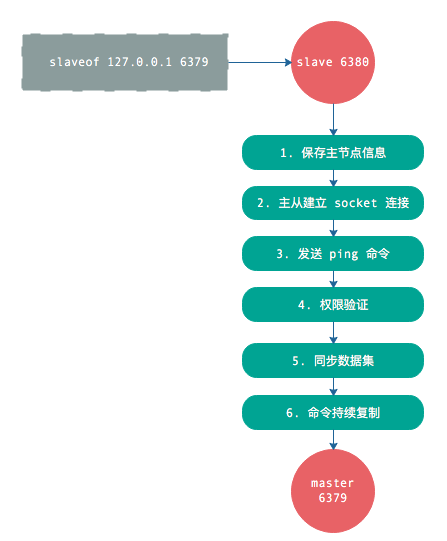
* 1. 薪火相传
     1. 上一个Slave可以是下一个Slave的Master，Slave同样可以接受其他Slaves的连接和同步请求，那么该Slave作为了链条中下一个的Master，可以有效减轻Master的写压力。
     2. 中途变更转向：会清除之前的数据，重新建立拷贝最新的
     3. slaveof 新主库IP 新主库端口
  2. 反客为主

主机挂了，从机变成主机

slaveof no one

使当前数据库停止与其他数据库的同步，转成主数据库

## 1.4复制原理



1. Slave启动成功连接到Master后会发送一个sync命令。
2. Master接到命令启动后台的存盘程进程，同时收集所有接收到的用于修改数据集命令，在后台进程执行完毕之后，Master将传送整个数据文件到Slave，以完成一次完全同步。
3. 全量复制：而Slave服务在接收到数据库文件数据后，将其存盘并加载到内存中。



1. 增量复制：Master继续将新的所有收集到的修改命令依次传给Slave，完成同步。
2. 但是只要是重新连接Master，一次完全同步（全量复制）将被自动执行。

## 1.5哨兵模式（sentinel）

1. 是什么

反客为主的自动版，能够后台监控主机是否故障，如果故障了根据投票数自动将从库转换为主库。

1. 怎么玩（使用步骤）
   1. 调整结构，6379带着80、81
   2. 自定义的/myredis目录下新建sentinel.conf文件，名字决不能错。
   3. 配置哨兵，填写内容：

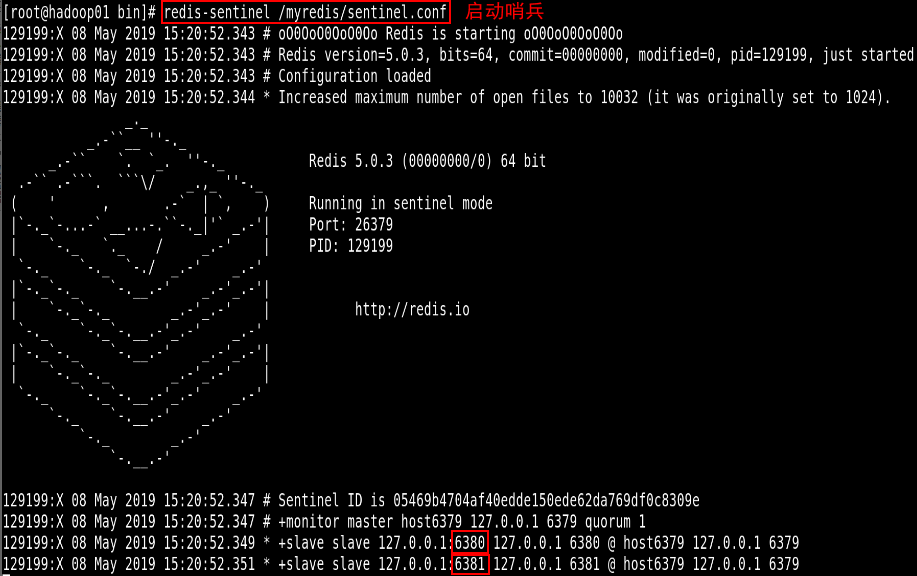


sentinel monitor 被监控数据库名字(自己起名字) 127.0.0.1 6379 1

上面最后一个数字1，表示主机挂掉后Slave投票看让谁接替成为主机，得票数多少后成为主机。

* 1. 启动哨兵

Redis-sentinel /myredis/centinel.conf



* 1. 正常主从演示
  2. 原有的Master挂了
  3. 投票新选
  4. 重新主从继续开工，info replication查查看
  5. 问题：如果之前的Master重启回来，会不会双Master冲突？

不会

1. 一组sentinel能同时监控多个Master

配置哨兵时可以监控多个Master

## 1.6复制的缺点

延时复制

由于所有的写操作都是先在Master上操作，然后同步更新到Slave上，所以从Master同步到Slave机器有一定的延迟，当系统很繁忙的时候，延迟问题会更加严重，Slave机器数量的增加也会使这个问题更加严重。