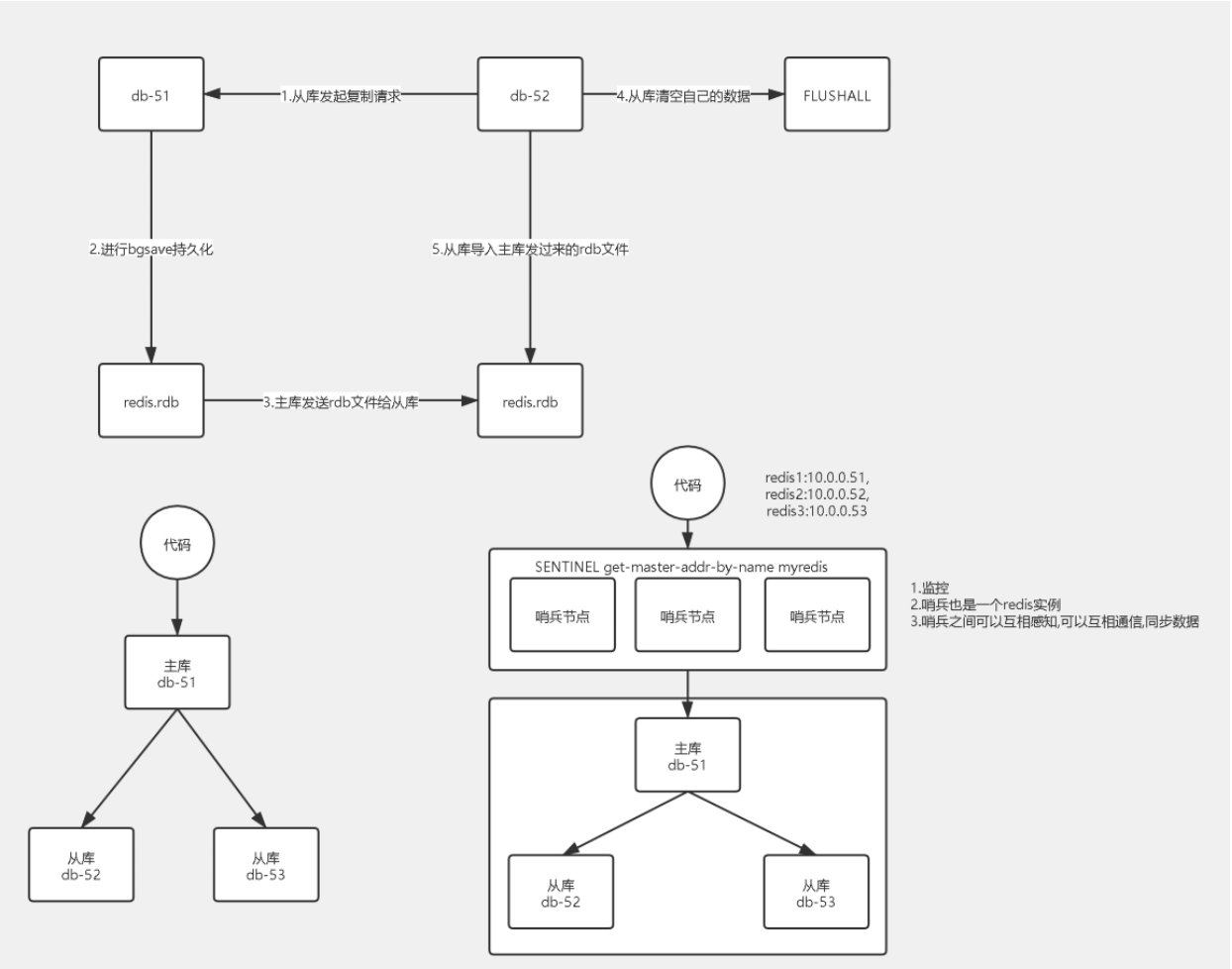


第12章 redis用户验证

1.配置密码认证功能 配置文件里加

```
requirepass 123456
```

2.使用密码

第一种:

```
[root@db-51 ~]# systemctl restart redis.service
[root@db-51 ~]# redis-cli
127.0.0.1:6379> set k3 v3
(error) NOAUTH Authentication required.
127.0.0.1:6379> AUTH 123456
OK
```

第二种:

```
[root@db-51 ~]# redis-cli -a '123456' get k1
Warning: Using a password with '-a' or '-u' option on the comman
d line interface may not be safe.
```

```
"v1"
```

3.为什么redis的密码认证这么简单?

```
1.redis一般都部署在内网环境，默认是相对比较安全的
2.有同学担心密码写在配置文件里，不用担心，因为开发不允许SSH登陆到Linux服务器，但是可以远程连接Redis，所以设置密码还是有作用的
3.使用的是普通用户运行的，而不是root，即使被利用漏洞，影响也有限
4.防火墙策略，登录redis服务器只能通过堡垒机登录
```

第13章 禁用或重命名危险命令

1.禁用危险命令

```
rename-command KEYS ""
rename-command SHUTDOWN ""
rename-command CONFIG ""
rename-command FLUSHALL ""
```

2.重命名危险命令

```
rename-command KEYS "QQ526195417"
rename-command SHUTDOWN ""
rename-command CONFIG ""
rename-command FLUSHALL ""
```

可以把重命名的命令改在""里

第14章 主从复制

1.快速部署第二台Redis服务器　db-52

```
ssh-keygen
ssh-copy-id 10.0.0.51
rsync -avz 10.0.0.51:/opt/redis_6379 /opt/
rsync -avz 10.0.0.51:/usr/local/bin/redis* /usr/local/bin/
rsync -avz 10.0.0.51:/usr/lib/systemd/system/redis.service /usr/lib/systemd/system/
sed -i 's#51#52#g' /opt/redis_6379/conf/redis_6379.conf
mkdir -p /data/redis_6379
```

```
groupadd redis -g 2000
useradd redis -u 2000 -g 2000 -M -s /sbin/nologin
 chown -R redis:redis /opt/redis*
 chown -R redis:redis /data/redis*
```

db-52配置文件

```
[root@db-52 /opt/redis_6379/conf]# cat redis_6379.conf
daemonize yes
bind 127.0.0.1 10.0.0.52
port 6379
pidfile /opt/redis_6379/pid/redis_6379.pid
logfile /opt/redis_6379/logs/redis_6379.log

save 900 1
save 300 10
 save 60 10000
 dbfilename redis.rdb
 dir /data/redis_6379/

 appendonly yes
 appendfilename "redis.aof"
 appendfsync everysec
```

```

systemctl daemon-reload
systemctl start redis
```

主从复制db-52

```
1.db-52安装配置redis
#复制命令
scp 10.0.0.51:/usr/local/bin/redis* /usr/local/bin/
#创建目录和用户
mkdir -p /opt/redis_6379/{conf,logs,pid}
mkdir -p /data/redis_6379
useradd redis -M -s /sbin/nologin
```

```
chown -R redis:redis /opt/redis*
chown -R redis:redis /data/redis*
#编写配置文件
cat >/opt/redis_6379/conf/redis_6379.conf<<EOF
daemonize yes
bind 127.0.0.1 10.0.0.52
port 6379
pidfile /opt/redis_6379/pid/redis_6379.pid
logfile /opt/redis_6379/logs/redis_6379.log
save 900 1
save 300 10
save 60 10000
dbfilename redis.rdb
dir /data/redis_6379/
EOF
#编写启动文件
cat >/usr/lib/systemd/system/redis.service<<EOF
[Unit]
Description=Redis persistent key-value database
After=network.target
After=network-online.target
Wants=network-online.target

[Service]
ExecStart=/usr/local/bin/redis-server /opt/redis_6379/conf/redis_6379.conf --supervised systemd
ExecStop=/usr/local/bin/redis-cli shutdown
Type=notify
User=redis
Group=redis
RuntimeDirectory=redis
RuntimeDirectoryMode=0755

[Install]
WantedBy=multi-user.target
```

```
EOF
#启动服务
systemctl daemon-reload
systemctl start redis
```

报错总结:

1.在db51上执行了命令

2.配置文件里的密码没删掉

3.配置文件里的重命名参数没删掉

4.用户id和组id冲突

5.没有rsync

6.拷贝过来的配置文件没有修改IP地址

2.db51插入测试命令

```
for i in {1..5000};do redis-cli set key_${i} v_${i} && echo "${i} is ok";done
```

3.配置主从复制

方法1:临时生效

```
redis-cli -h 10.0.0.52 SLAVEOF 10.0.0.51 6379
```

方法2:写进配置文件永久生效

```
[root@db-52 /data/redis_6379]# redis-cli -h 10.0.0.52
10.0.0.52:6379> SLAVEOF 10.0.0.51 6379
```

4.检查复制进度

```
redis-cli
INFO replication
ROLE
```

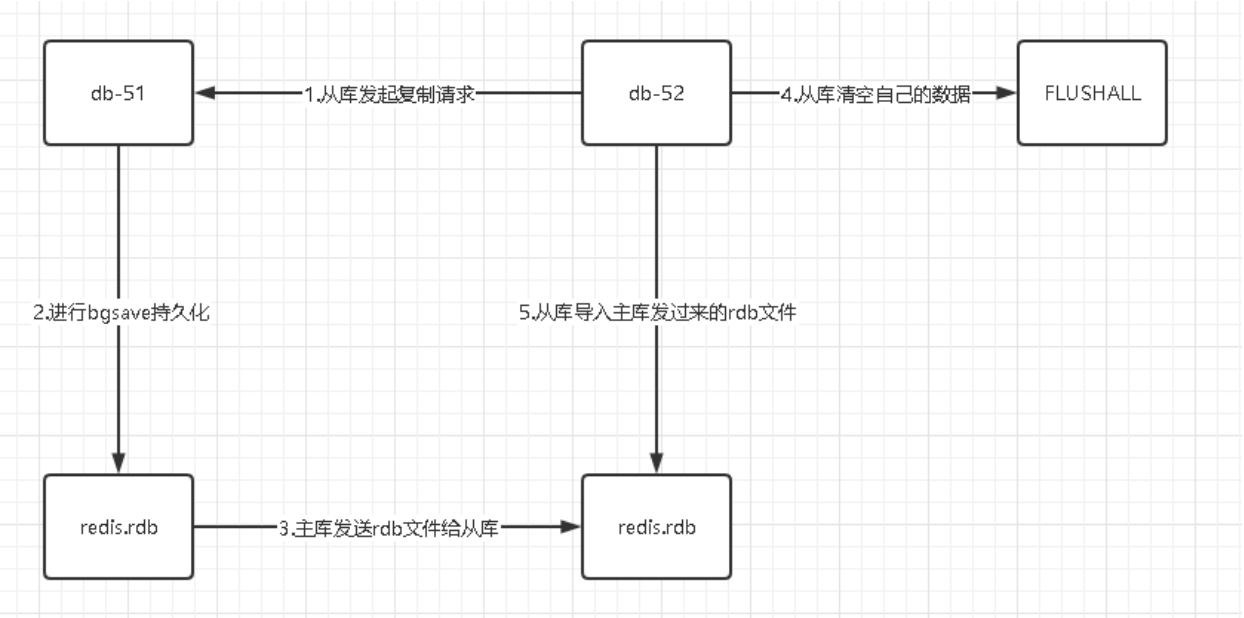
5.主从复制流程

1.简单流程

```
1.从节点发送同步请求到主节点
2.主节点接收到从节点的请求之后,做了如下操作
  - 立即执行bgsave将当前内存里的数据持久化到磁盘上
  - 持久化完成之后,将rdb文件发送给从节点
3.从节点从主节点接收到rdb文件之后,做了如下操作
  - 清空自己的数据
  - 载入从主节点接收的rdb文件到自己的内存里
```

```
4.后面的操作就是和主节点实时的了

```



6.取消复制

```
[root@db-52 ~]# redis-cli
127.0.0.1:6379> SLAVEOF no one
```

7.主从复制注意

1.从节点只读不可写

2.从节点不会自动故障转移，他会一直尝试同步主节点，并且依然不可写

3.主从复制故障转移需要介入的地方

- **修改代码指向新主的IP**
- **从节点需要执行slaveof no one**

4.从库建立同步时会清空自己的数据，如果同步对象写错了，就清空了

5.从库也可以正常的RDB持久化

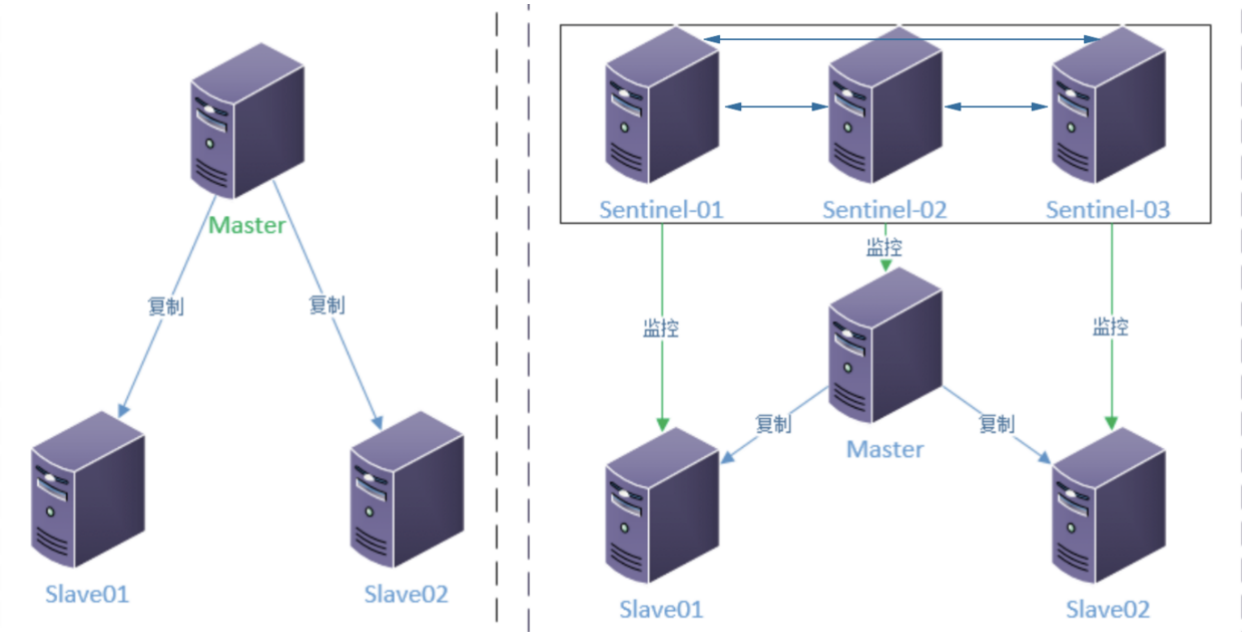
8.有密码认证的情况下设置主从复制　在配置文件加

```
masterauth 123456
```

9.安全的操作

一定要做好数据备份，无论是主节点还是从节点，操作前最好做下备份

第15章 Redis Sentinel(哨兵)



1.哨兵的作用

1.解决主从复制需要人为干预的问题

2.提供了自动的高可用方案

2.目录和端口规划

redis节点　6379

哨兵节点　26379

3.部署3台redis单节点主从关系

主从节点部署：

```
1.db-51和db-52的操作
systemctl stop redis
rm -rf /data/redis_6379/*
cat >/opt/redis_6379/conf/redis_6379.conf << EOF
daemonize yes
bind 127.0.0.1 $(ifconfig eth0|awk 'NR==2{print $2}')
port 6379
pidfile "/opt/redis_6379/pid/redis_6379.pid"
logfile "/opt/redis_6379/logs/redis_6379.log"
 dbfilename "redis.rdb"
 dir "/data/redis_6379"
 appendonly yes
```

```
appendfilename "redis.aof"
appendfsync everysec
EOF
systemctl start redis
redis-cli

2.db-53的操作
mkdir -p /opt/redis_6379/{conf,logs,pid}
mkdir -p /data/redis_6379
useradd redis -M -s /sbin/nologin
chown -R redis:redis /opt/redis*
chown -R redis:redis /data/redis*
scp 10.0.0.51:/usr/local/bin/redis* /usr/local/bin/
scp 10.0.0.51:/opt/redis_6379/conf/redis_6379.conf /opt/redis_6379/conf
scp 10.0.0.51:/usr/lib/systemd/system/redis.service /usr/lib/systemd/system/
sed -i 's#51#53#g' /opt/redis_6379/conf/redis_6379.conf
systemctl daemon-reload
systemctl start redis

3.配置复制关系
redis-cli -h 10.0.0.52 slaveof 10.0.0.51 6379
redis-cli -h 10.0.0.53 slaveof 10.0.0.51 6379

4.在51上创k 在52和53上看是否同步成功
[root@db-51 ~]# redis-cli
127.0.0.1:6379> set k2323 v13123
OK
[root@db-52 /data/redis_6379]# redis-cli
127.0.0.1:6379> keys *
1) "k2323"
[root@db-53 ~]# redis-cli
127.0.0.1:6379> keys *
1) "k2323"
```

5.部署哨兵节点-3台机器都操作

==========部署哨兵节点-3台机器都操作==========

#1.创建目录

```
mkdir -p /data/redis_26379
mkdir -p /opt/redis_26379/{conf,pid,logs}
```

#编写配置文件

```
cat >/opt/redis_26379/conf/redis_26379.conf << EOF
bind $(ifconfig eth0|awk 'NR==2{print $2}')
port 26379
daemonize yes
logfile /opt/redis_26379/logs/redis_26379.log
dir /data/redis_26379
sentinel monitor myredis 10.0.0.51 6379 2
sentinel down-after-milliseconds myredis 300
sentinel parallel-syncs myredis 1
 sentinel failover-timeout myredis 18000
 EOF
```

#2.更改目录授权

```
chown -R redis:redis /data/redis*
chown -R redis:redis /opt/redis*
```

#3.配置启动文件

```
cat >/usr/lib/systemd/system/redis-sentinel.service<<EOF
[Unit]
Description=Redis persistent key-value database
After=network.target
After=network-online.target
Wants=network-online.target

[Service]
```

```
ExecStart=/usr/local/bin/redis-sentinel /opt/redis_26379/conf/redis_26379.conf --supervised systemd
ExecStop=/usr/local/bin/redis-cli -h $(ifconfig eth0|awk 'NR==2{print $2}') -p 26379 shutdown
Type=notify
User=redis
Group=redis
RuntimeDirectory=redis
RuntimeDirectoryMode=0755

[Install]
WantedBy=multi-user.target
EOF
```

#4.启动服务

```
systemctl daemon-reload
systemctl start redis-sentinel
```

#5.连接测试

```
redis-cli -h $(ifconfig eth0|awk 'NR==2{print $2}') -p 26379
netstat -lntup|grep redis
tcp 0 0 10.0.0.51:26379 0.0.0.0:* LISTEN 4844/redis-sentinel
```

关键配置解释:

```
sentinel monitor myredis 10.0.0.51 6379 2
# myredis主节点别名  主节点IP 端口  需要2个哨兵节点同意
sentinel down-after-milliseconds myredis 3000
# 认定服务器已经断线所需要的毫秒数
sentinel parallel-syncs myredis 1
# 向主节点发给复制操作的从节点个数，1表示轮训发起复制
sentinel failover-timeout myredis 18000
# 故障转移超时时间
```

6.哨兵注意

1.哨兵发起故障转移的条件是master节点失去联系，从节点挂掉不会发起故障转移

2.哨兵会自己维护配置文件，不需要手动修改

3.如果主从的结构发生变化，哨兵之间会自动同步最新的消息并且自动更新配置文件

4.哨兵启动完成之后，以后不要再自己去设置主从关系

7.验证主机点

```
redis-cli -h 10.0.0.51 -p 26379 SENTINEL get-master-addr-by-name myredis
```

8.哨兵的常用命令

```
redis-cli -h 10.0.0.51 -p 26379 SENTINEL get-master-addr-by-name myredis
redis-cli -h 10.0.0.51 -p 26379 SENTINEL master myredis
redis-cli -h 10.0.0.51 -p 26379 SENTINEL slaves myredis
redis-cli -h 10.0.0.51 -p 26379 SENTINEL ckquorum myredis
```

9.模拟故障转移

模拟方法:

1.模拟故障-关闭db-51上的所有redis节点

关闭redis当前的主节点

观察其他2个节点会不会发生选举

查看哨兵配置文件会不会更新

查看从节点配置文件会不会更新

查看主节点能不能写入

查看从节点是否同步正常

2.哨兵注意事项

1.哨兵发起故障转移的条件是master节点失去联系，从节点挂掉不会发起故障转移

2.哨兵会自己维护配置文件，不需要手动修改

3.如果主从的结构发生变化，哨兵之间会自动同步最新的消息并且自动更新配置文件

4.哨兵启动完成之后，以后不要再自己去设置主从关系

流程:

1）在从节点列表中选出一个节点作为新的主节点，选择方法如下:

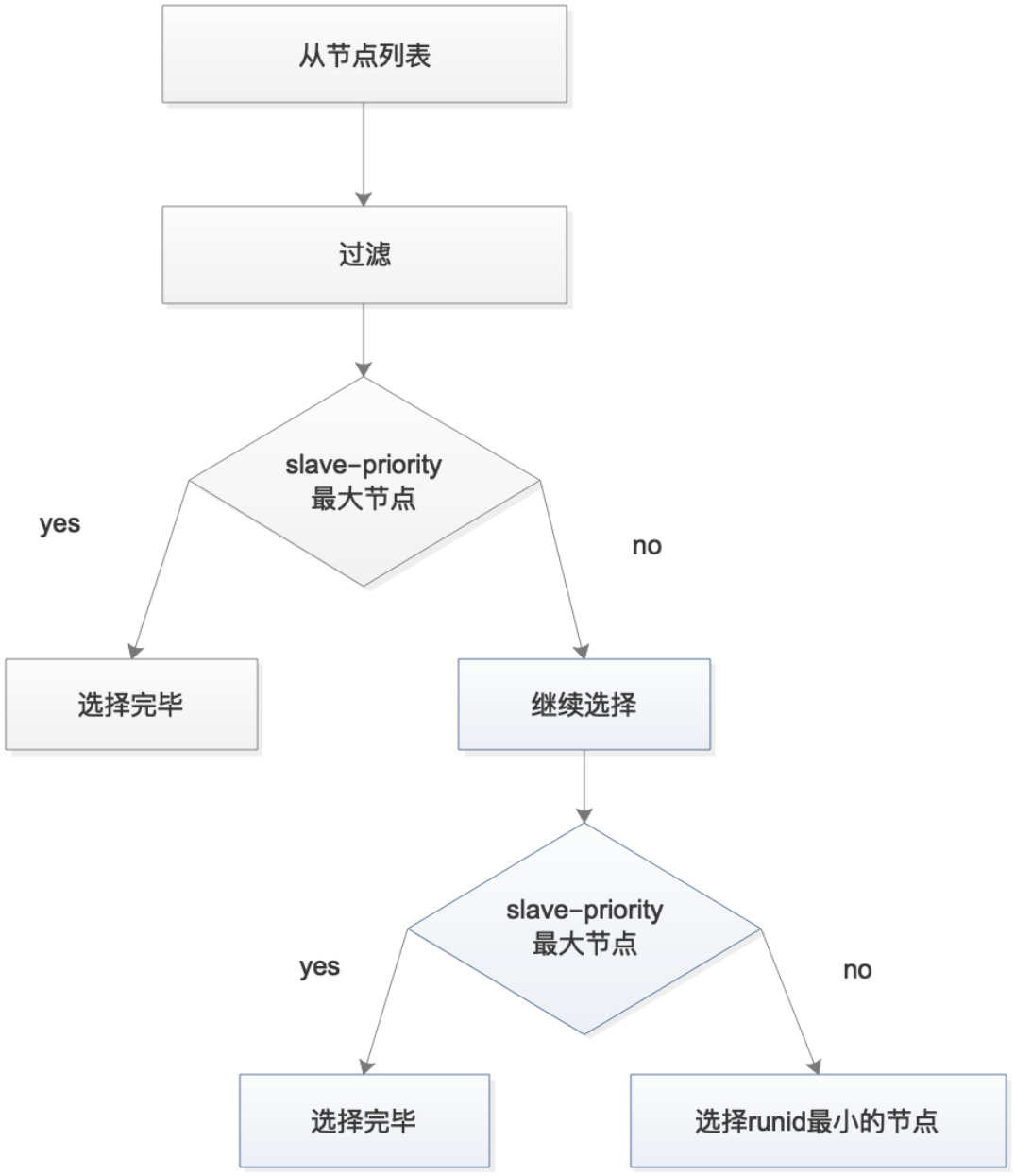
a）过滤："不健康"（主观下线、断线）、5秒内没有回复过Sentinel节点ping响应、与主节点失联超过down-after-milliseconds*10秒。

b）选择slave-priority（从节点优先级）最高的从节点列表，如果存在则返回，不存在则继续。

c）选择复制偏移量最大的从节点（复制的最完整），如果存在则返回，不存在则继续。

d）选择runid最小的从节点

流程图：



结论：

```
主节点挂掉，哨兵会选举新的主节点
在新主节点上执行slaveof no one
在从节点执行slave of 新主节点
自动更新哨兵配置
自动更新从节点配置
```

10.故障修复重新上线

启动单节点

检查是否变成从库

11.哨兵加权选举

查看权重

```
redis-cli -h 10.0.0.51 -p 6379 CONFIG GET slave-priority
redis-cli -h 10.0.0.52 -p 6379 CONFIG GET slave-priority
redis-cli -h 10.0.0.53 -p 6379 CONFIG GET slave-priority
```

暗箱操作：假如选中53作为最新的master

```
redis-cli -h 10.0.0.51 -p 6379 CONFIG SET slave-priority 0
redis-cli -h 10.0.0.52 -p 6379 CONFIG SET slave-priority 0
```

检查：

```
redis-cli -h 10.0.0.51 -p 6379 CONFIG GET slave-priority
redis-cli -h 10.0.0.52 -p 6379 CONFIG GET slave-priority
redis-cli -h 10.0.0.51 -p 26379 sentinel get-master-addr-by-name
myredis
```

主动发生选举

```
redis-cli -h 10.0.0.51 -p 26379 sentinel failover myredis
redis-cli -h 10.0.0.51 -p 26379 sentinel get-master-addr-by-name
myredi
```

恢复原状

```
redis-cli -h 10.0.0.51 -p 6379 CONFIG SET slave-priority 100
redis-cli -h 10.0.0.52 -p 6379 CONFIG SET slave-priority 100
```