第四十六章：Redis sentinel哨兵集群

**一、Redis sentinel 概述；**

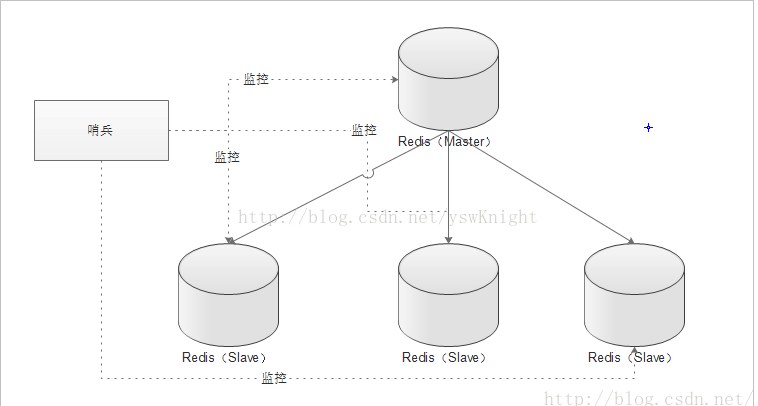
**二、Redis sentinel的工作机制；**

**三、案例：构建Redis sentinel集群；**

**四、案例：Keepalived+Haproxy+Redis实现高可用秒级切换集群；**

**一、Redis sentinel 概述；**

**概述：**Sentinel：哨兵模式，是一个分布式系统，该进程是用于监控redis集群中Master主服务器工作的状态，在Master主服务器发生故障的时候，可以实现Master和Slave服务器的切换，保证系统的高可用，其已经被集成在redis 2.6 +的版本中，Redis的哨兵模式到了2.8版本之后就得到了稳定；

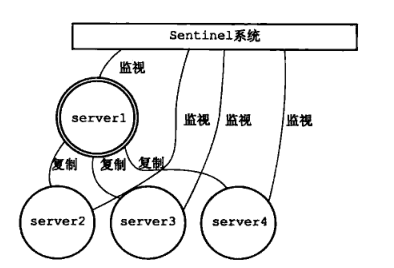


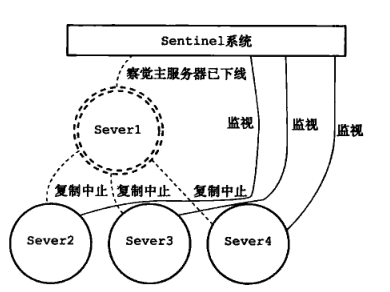
**二、Redis sentinel的工作机制；**

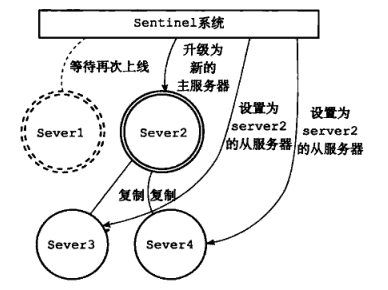
**工作进程：**

* 监控（Monitoring）: 哨兵（sentinel）通过流言协议（gossip protocols）会不断地检查你的Master和Slave是否运作正常；
* 提醒（Notification）：当被监控的某个Redis节点出现问题时, 哨兵（sentinel）可以通过 API 向管理员或者其他应用程序发送通知；
* 自动故障迁移（Automatic failover）：当一个Master不能正常工作时，哨兵通过投票协议（Agreement Protocols）会开始一次自动故障迁移操作，它会将其他一个Slave升级为新的Master，当客户端试图连接失效的Master时，集群也会向客户端返回新Master的地址，使得集群可以使用现在的Master替换失效Master。Master和Slave服务器切换后，Master的redis.conf、Slave的redis.conf和sentinel.conf的配置文件的内容都会发生相应的改变，即，Master主服务器的redis.conf配置文件中会多一行slaveof的配置，sentinel.conf的监控目标会随之调换；

**图解：**







**三、案例：构建Redis sentinel集群；**

**案例环境：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 系统类型 | IP地址 | 主机名 | 所需软件 | 端口 |
| Centos7.4 1708 64bit | 192.168.100.101 | master.linuxfan.cn | redis-4.0.9.tar.gz | redis：6379  redis sentinel：26379 |
| Centos7.4 1708 64bit | 192.168.100.102 | slave1.linuxfan.cn | redis-4.0.9.tar.gz | redis：6379 |
| Centos7.4 1708 64bit | 192.168.100.103 | slave2.linuxfan.cn | redis-4.0.9.tar.gz | redis：6379 |

**案例步骤：**

* 安装所有节点的redis服务（所有节点配置相同，在此列举master节点配置）；
* 修改master节点的配置文件，实现主从复制；
* 修改slave节点的配置文件，实现主从复制（两个slave节点配置相同，在此列举slave1节点配置）；
* 验证主从节点的主从同步；
* 在master节点上配置Redis sentinel；
* 在master节点上启动Redis sentinel；
* 测试Redis sentinel，关闭master节点后，测试集群切换；
* 验证新的master节点与slave节点之间的主从同步；
* 恢复master节点，验证集群状态；
* 扩展：如若实现master节点切换后，保证IP地址的统一性，可以选择使用Keepalived及shell脚本实现；
* **安装所有节点的redis服务（所有节点配置相同，在此列举master节点配置）；**

[root@master ~]# wget http://download.redis.io/releases/redis-4.0.9.tar.gz

[root@master ~]# tar zxvf redis-4.0.9.tar.gz

[root@master ~]# cd redis-4.0.9

[root@master redis-4.0.9]# make

[root@master redis-4.0.9]# echo $?

[root@master redis-4.0.9]# cd

[root@master ~]# mkdir -p /usr/local/redis

[root@master ~]# cp /root/redis-4.0.9/src/redis-server /usr/local/redis/

[root@master ~]# cp /root/redis-4.0.9/src/redis-cli /usr/local/redis/

[root@master ~]# cp /root/redis-4.0.9/redis.conf /usr/local/redis/

[root@master ~]# ls /usr/local/redis/

redis-cli redis.conf redis-server

[root@master ~]# sed -i '/^bind 127.0.0.1$/s/127.0.0.1/192.168.100.101/g' /usr/local/redis/redis.conf

[root@master ~]# sed -i '/protected-mode/s/yes/no/g' /usr/local/redis/redis.conf ##关闭redis的保护模式

注：配置哨兵模式时，一定需要将以上保护模式关闭，不然会出现哨兵进程无法选举切换master节点的故障问题；

[root@master ~]# sed -i '/daemonize/s/no/yes/g' /usr/local/redis/redis.conf ##开启redis的后台守护进程模式

[root@master ~]# sed -i '/requirepass/s/foobared/123123/g' /usr/local/redis/redis.conf ##设置redis的密码为123123

[root@master ~]# sed -i '/requirepass 123123/s/^#//g' /usr/local/redis/redis.conf ##开启redis的密码

[root@master ~]# ln -s /usr/local/redis/redis-cli /usr/local/bin/redis

[root@master ~]# cat <<END >>/etc/init.d/redis

#!/bin/sh

# chkconfig: 2345 80 90

# description: Start and Stop redis

#PATH=/usr/local/bin:/sbin:/usr/bin:/bin

REDISPORT=6379

EXEC=/usr/local/redis/redis-server

REDIS\_CLI=/usr/local/redis/redis-cli

PIDFILE=/var/run/redis\_6379.pid

CONF="/usr/local/redis/redis.conf"

AUTH="123123"

LISTEN\_IP=\$(netstat -utpln |grep redis-server |awk '{print \$4}'|awk -F':' '{print \$1}')

case "\$1" in

start)

if [ -f \$PIDFILE ]

then

echo "\$PIDFILE exists, process is already running or crashed"

else

echo "Starting Redis server..."

\$EXEC \$CONF

fi

if [ "\$?"="0" ]

then

echo "Redis is running..."

fi

;;

stop)

if [ ! -f \$PIDFILE ]

then

echo "\$PIDFILE does not exist, process is not running"

else

PID=\$(cat \$PIDFILE)

echo "Stopping ..."

\$REDIS\_CLI -h \$LISTEN\_IP -p \$REDISPORT -a \$AUTH SHUTDOWN

while [ -x \${PIDFILE} ]

do

echo "Waiting for Redis to shutdown ..."

sleep 1

done

echo "Redis stopped"

fi

;;

restart|force-reload)

\${0} stop

\${0} start

;;

\*)

echo "Usage: /etc/init.d/redis {start|stop|restart|force-reload}" >&2

exit 1

esac

END

[root@master ~]# chmod 755 /etc/init.d/redis

[root@master ~]# chkconfig --add redis

[root@master ~]# /etc/init.d/redis start

Starting Redis server...

4390:C 04 May 02:16:45.232 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo

4390:C 04 May 02:16:45.232 # Redis version=4.0.9, bits=64, commit=00000000, modified=0, pid=4390, just started

4390:C 04 May 02:16:45.232 # Configuration loaded

Redis is running...

[root@master ~]# netstat -utpln |grep redis

tcp 0 192.168.100.101:6379 0.0.0.0:\* LISTEN 4204/redis-server \*

[root@master ~]# redis -h 192.168.100.101 -a 123123 -p 6379

192.168.100.101:6379> exit

* **修改master节点的配置文件，实现主从复制；**

[root@master ~]# sed -i '450s/^\(.\).\{22\}/min-slaves-to-write 2/g' /usr/local/redis/redis.conf

##设置slave节点的数量，如果slave节点数量少于此值，那么master节点将停止客户端的一切写请求

[root@master ~]# sed -n '451s/^\(.\).\{22\}/min-slaves-max-lag 10/g' /usr/local/redis/redis.conf

##master与slave之间同步数据的超时时间，若超过此时间，master节点将停止客户端的一切写操作

[root@master ~]# /etc/init.d/redis restart

Stopping ...

Redis stopped

...

* **修改slave节点的配置文件，实现主从复制（两个slave节点配置相同，在此列举slave1节点配置）；**

[root@slave1 ~]# sed -i '281s/^\(.\).\{32\}/slaveof 192.168.100.101 6379/g' /usr/local/redis/redis.conf

##指定master的ip地址以及端口

[root@slave1 ~]# sed -i '288s/^\(.\).\{29\}/masterauth 123123/g' /usr/local/redis/redis.conf

##指定master的连接密码

[root@slave1 ~]# /etc/init.d/redis restart

/var/run/redis\_6379.pid does not exist, process is not running

Starting Redis server...

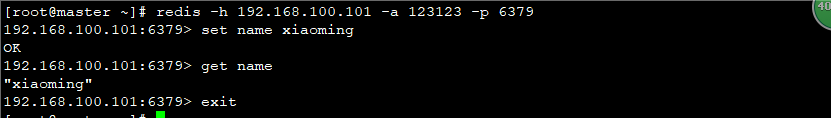
4387:C 18 May 03:24:00.027 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo

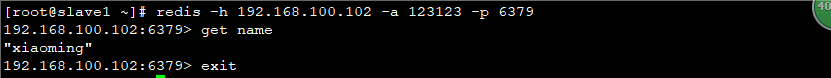
4387:C 18 May 03:24:00.027 # Redis version=4.0.9, bits=64, commit=00000000, modified=0, pid=4387, just started

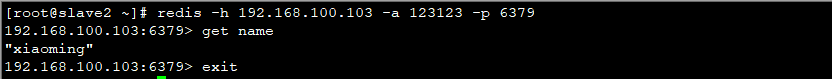
4387:C 18 May 03:24:00.027 # Configuration loaded

Redis is running...

* **验证主从节点的主从同步；**







* **在master节点上配置Redis sentinel；**

[root@master ~]# cp redis-4.0.9/src/redis-sentinel /usr/local/redis/ ## sentinel启动脚本

[root@master ~]# cp redis-4.0.9/sentinel.conf /usr/local/redis/ ## sentinel配置文件

[root@master ~]# mkdir -p /var/redis/data ##创建日志文件存放位置

[root@master ~]# vi /usr/local/redis/sentinel.conf ##修改如下

port 26379

dir "/var/redis/data"

sentinel monitor master.linuxfan.cn 192.168.100.101 6379 1 ## 1表示当一台master出现故障，则进行切换

sentinel down-after-milliseconds master.linuxfan.cn 3000 ##指定master的失效时间，单位毫秒，3秒

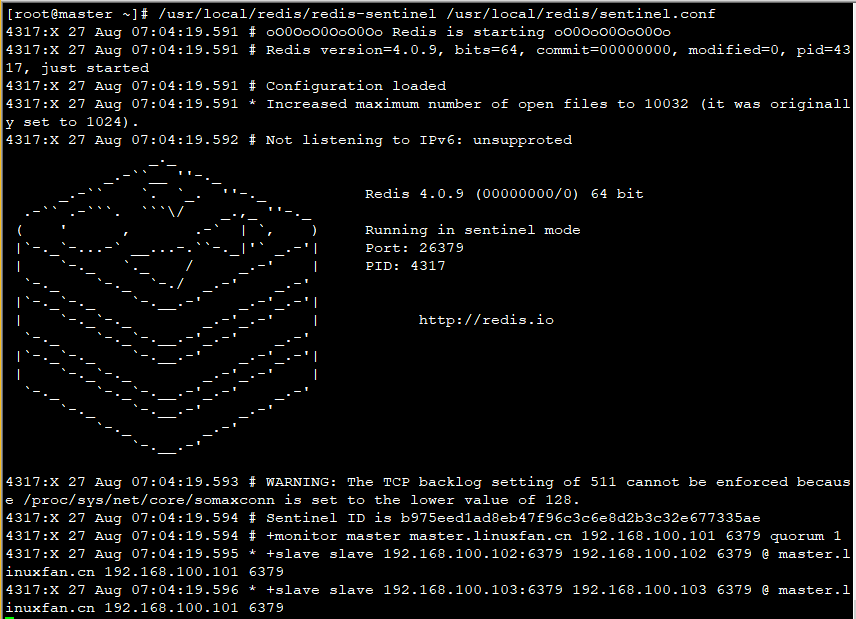
sentinel auth-pass master.linuxfan.cn 123123 ##连接master与slave节点的密码

sentinel config-epoch master.linuxfan.cn 1 ##切换后，最多有多少节点可以与新的master进行同步

sentinel failover-timeout mymaster 180000 ##切换操作完成的时间周期为180秒，逾期认为切换失败

:wq

* **在master节点上启动Redis sentinel；**



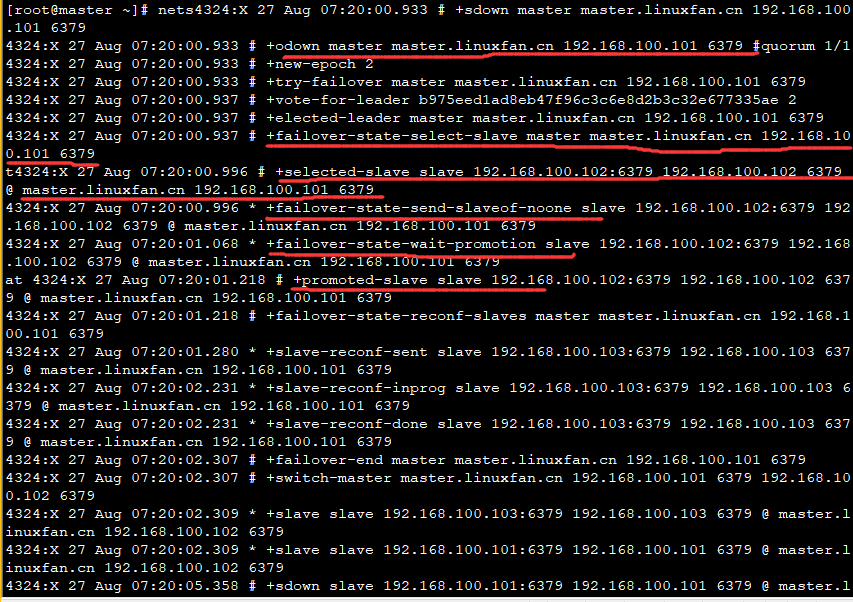


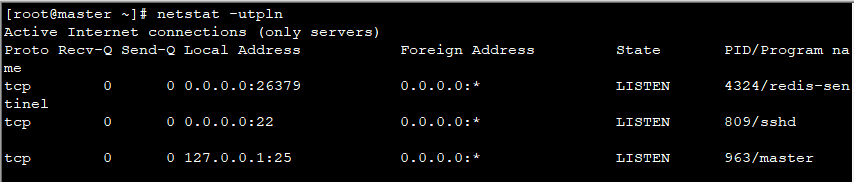
[root@master ~]# netstat -utpln |grep redis-sen

tcp 0 0 0.0.0.0:26379 0.0.0.0:\* LISTEN 4324/redis-setinel

* **测试Redis sentinel，关闭master节点后，测试集群切换；**







注解：Sentinel日志语句所表达的意义

+reset-master <instance details> -- 当master被重置时.

+slave <instance details> -- 当检测到一个slave并添加进slave列表时.

+failover-state-reconf-slaves <instance details> -- Failover状态变为reconf-slaves状态时

+failover-detected <instance details> -- 当failover发生时

+slave-reconf-sent <instance details> -- sentinel发送SLAVEOF命令把它重新配置时

+slave-reconf-inprog <instance details> -- slave被重新配置为另外一个master的slave，但数据复制还未发生时。

+slave-reconf-done <instance details> -- slave被重新配置为另外一个master的slave并且数据复制已经与master同步时。

-dup-sentinel <instance details> -- 删除指定master上的冗余sentinel时 (当一个sentinel重新启动时，可能会发生这个事件).

+sentinel <instance details> -- 当master增加了一个sentinel时。

+sdown <instance details> -- 进入SDOWN状态时;

-sdown <instance details> -- 离开SDOWN状态时。

+odown <instance details> -- 进入ODOWN状态时。

-odown <instance details> -- 离开ODOWN状态时。

+new-epoch <instance details> -- 当前配置版本被更新时。

+try-failover <instance details> -- 达到failover条件，正等待其他sentinel的选举。

+elected-leader <instance details> -- 被选举为去执行failover的时候。

+failover-state-select-slave <instance details> -- 开始要选择一个slave当选新master时。

no-good-slave <instance details> -- 没有合适的slave来担当新master

selected-slave <instance details> -- 找到了一个适合的slave来担当新master

failover-state-send-slaveof-noone <instance details> -- 当把选择为新master的slave的身份进行切换的时候。

failover-end-for-timeout <instance details> -- failover由于超时而失败时。

failover-end <instance details> -- failover成功完成时。

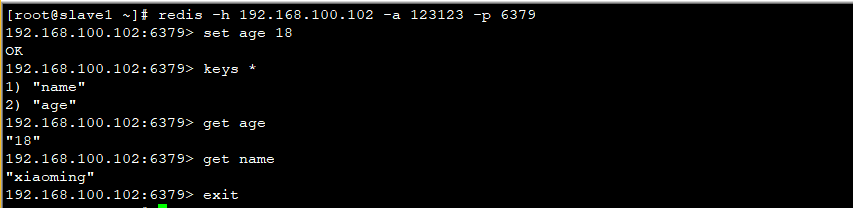
switch-master <master name> <oldip> <oldport> <newip> <newport> -- 当master的地址发生变化时。通常这是客户端最感兴趣的消息了。

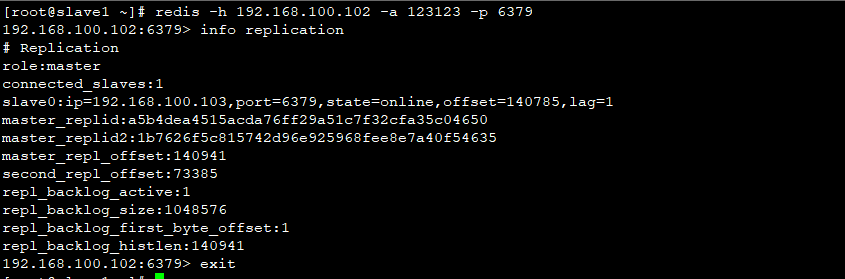
+tilt -- 进入Tilt模式。

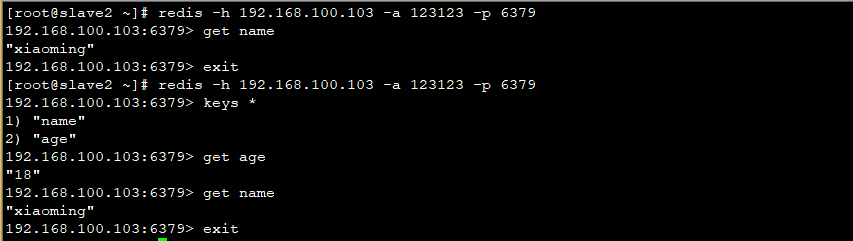
-tilt -- 退出Tilt模式。

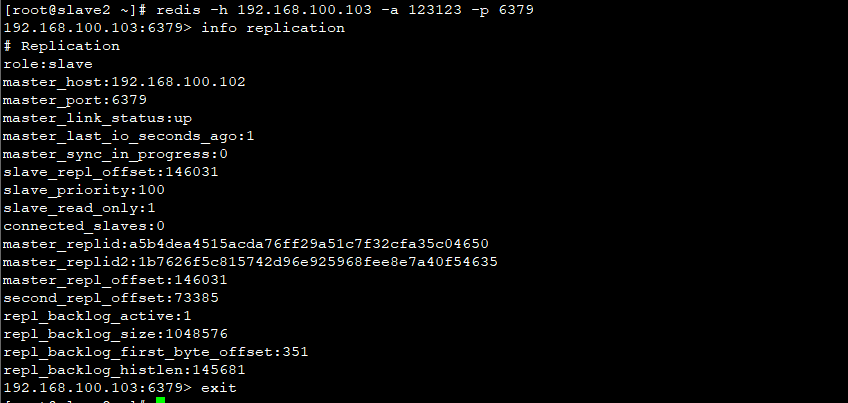


* **验证新的master节点与slave节点之间的主从同步；**

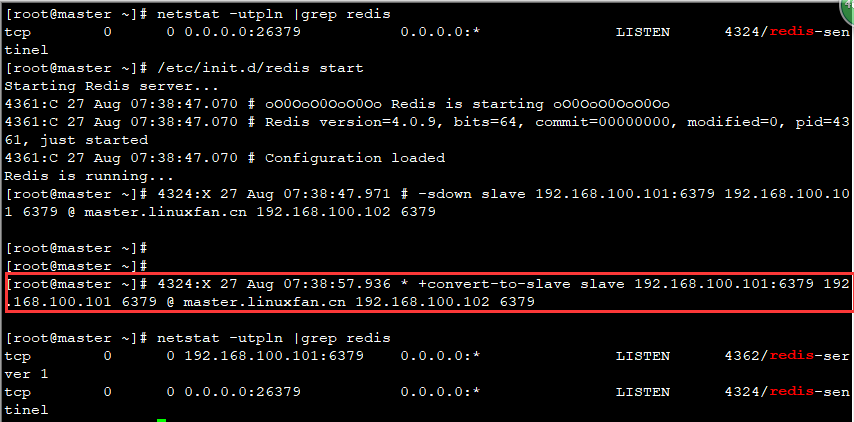


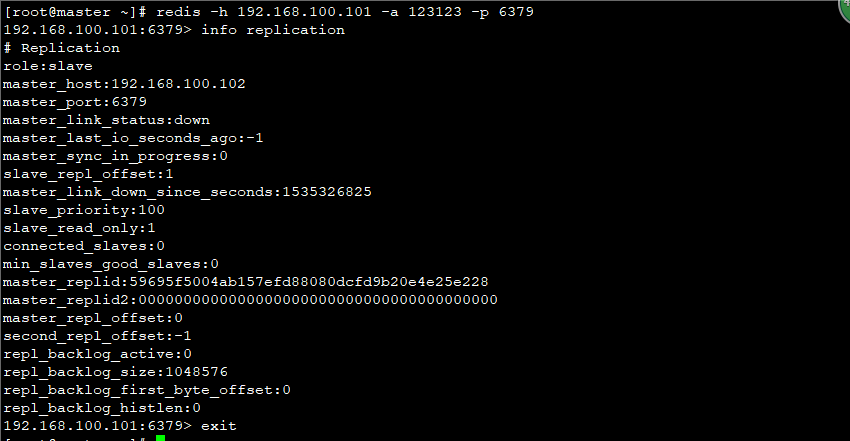






* **恢复master节点，验证集群状态；**





* **扩展：如若实现master节点切换后，保证IP地址的统一性，可以选择使用Keepalived及shell脚本实现；**

**四、案例：Keepalived+Haproxy+Redis实现高可用秒级切换集群；**

http://blog.51cto.com/yw666/1940272