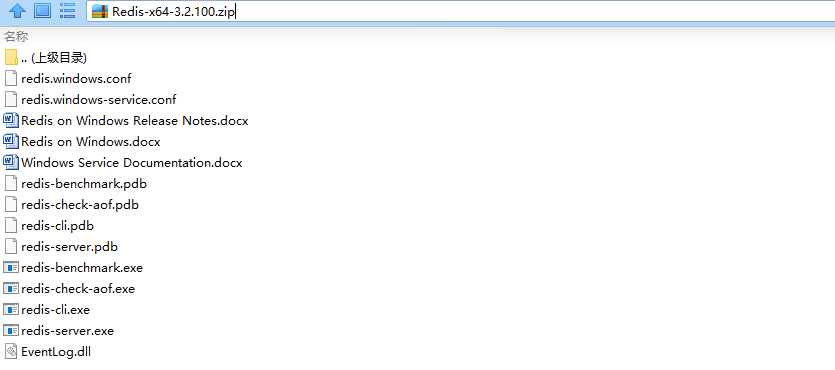
# Redis集群+sentinel

## 目录结构



Redis-x64-3.2.100



每一个文件夹拷贝一份

## Master配置

将redis.windows.conf修改为redis6379.conf（改这个文件名只是为了好区分它用那个端口,不改也可以）

### 修改配置

只需要修改一点，Port端口不需要修改，就让master用默认的6379端口

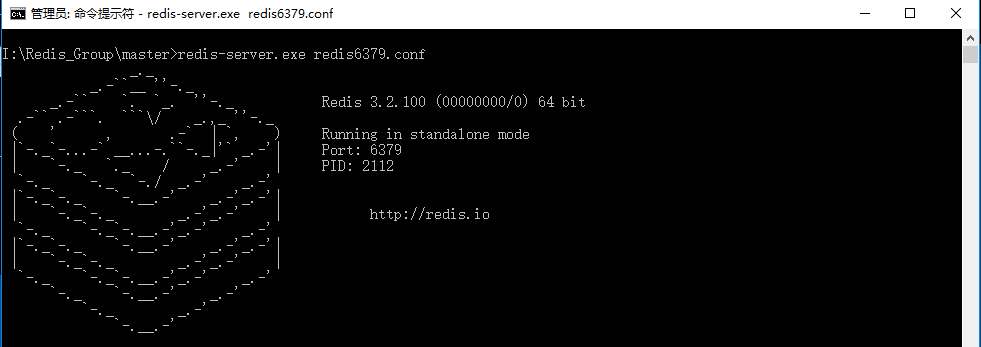
Bind 127.0.0.1

修改为（192.168.2.121是本机IP）

bind 192.168.2.121

启动Master

redis-server.exe redis6379.conf



Slave1

将redis.windows.conf修改为redis.windows6380.conf（改这个文件名只是为了好区分它用那个端口,不改也可以）

### 修改配置

#### 修改bind

Bind 127.0.0.1

修改为（192.168.2.121是本机IP）

bind 192.168.2.121

#### 修改port

Prot 6379

修改为 (6379端口 已经被master用了)

port 6380

#### 修改slaveof

slaveof原来是注释掉的#slaveof，要将它去掉。

slaveof前面不能用空格否则启动slave时会报错

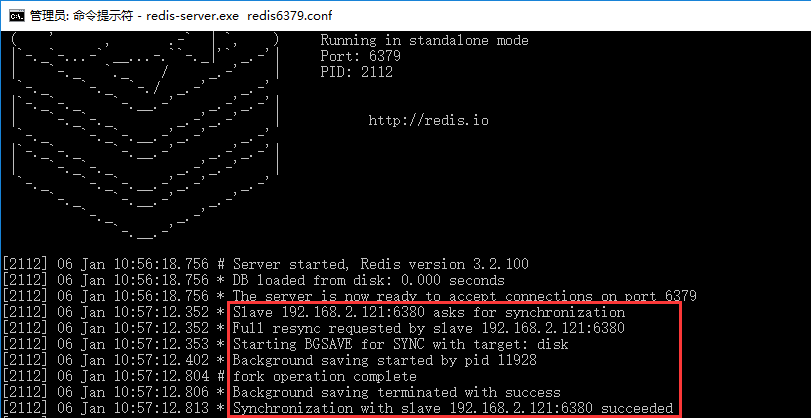
表明slave1是master的slave

slaveof 192.168.2.121 6379

### 启动salve1



启动slave1后，我们可以看到master控制台显示slave1已经连接上它了



按照配置slave1的方法分别配置slave2、slave3（配置文件里面需要修改：Bind、Port、Slaveof）

## Sentinel

## 概述

Redis-Sentinel是Redis官方推荐的高可用性(HA)解决方案，当用Redis做Master-slave的高可用方案时，假如master宕机了，Redis本身(包括它的很多客户端)都没有实现自动进行主备切换，而Redis-sentinel本身也是一个独立运行的进程，它能监控多个master-slave集群，发现master宕机后能进行自懂切换。

它的主要功能有以下几点

* 不时地监控redis是否按照预期良好地运行;
* 如果发现某个redis节点运行出现状况，能够通知另外一个进程(例如它的客户端);
* 能够进行自动切换。当一个master节点不可用时，能够选举出master的多个slave(如果有超过一个slave的话)中的一个来作为新的master,其它的slave节点会将它所追随的master的地址改为被提升为master的slave的新地址。<br/>

## Sentinel支持集群

很显然，只使用单个sentinel进程来监控redis集群是不可靠的，当sentinel进程宕掉后(sentinel本身也有单点问题，single-point-of-failure)整个集群系统将无法按照预期的方式运行。所以有必要将sentinel集群，这样有几个好处：

* 即使有一些sentinel进程宕掉了，依然可以进行redis集群的主备切换；
* 如果只有一个sentinel进程，如果这个进程运行出错，或者是网络堵塞，那么将无法实现redis集群的主备切换（单点问题）;
* 如果有多个sentinel，redis的客户端可以随意地连接任意一个sentinel来获得关于redis集群中的信息。

### 配置Sentinel

新建一个sentinel26379.conf配置文件

文件内容

bind 192.168.2.121

port 26379

sentinel monitor mymaster 192.168.2.121 6379 2

sentinel down-after-milliseconds mymaster 60000

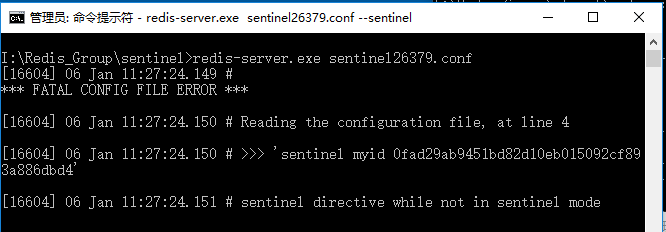
sentinel failover-timeout mymaster 180000

sentinel parallel-syncs mymaster 1

### 启动sentinel

redis-server.exe sentinel26379.conf

无法启动 sentinel



异常：

\*\*\* FATAL CONFIG FILE ERROR \*\*\*

[16604] 06 Jan 11:27:24.150 # Reading the configuration file, at line 4

[16604] 06 Jan 11:27:24.150 # >>> 'sentinel myid 0fad29ab9451bd82d10eb015092cf893a886dbd4'

[16604] 06 Jan 11:27:24.151 # sentinel directive while not in sentinel mode

加上参数-- sentinel 启动正常

**redis-server.exe sentinel26379.conf --sentinel**



Sentinel也可以像redis配置集聚，当其中一个挂掉之后，其它sentinel仍然可以继续工作。

只需要多拷贝一份，修改一下端口即可。

## 测试

一：在master中设置值，看slave是否能够同步

1：启动一个client连接master，然后，设置值。

**查看一下基本信息**

I:\Redis\_Group\master>redis-cli.exe -h 192.168.2.121 -p 6379

192.168.2.121:6379> info Replication

# Replication

**role:master**

**connected\_slaves:3**

slave0:ip=192.168.2.121,port=6380,state=online,offset=189652,lag=1

slave1:ip=192.168.2.121,port=6382,state=online,offset=189793,lag=0

slave2:ip=192.168.2.121,port=6381,state=online,offset=189793,lag=1

master\_repl\_offset:189793

repl\_backlog\_active:1

repl\_backlog\_size:1048576

repl\_backlog\_first\_byte\_offset:2

repl\_backlog\_histlen:189792

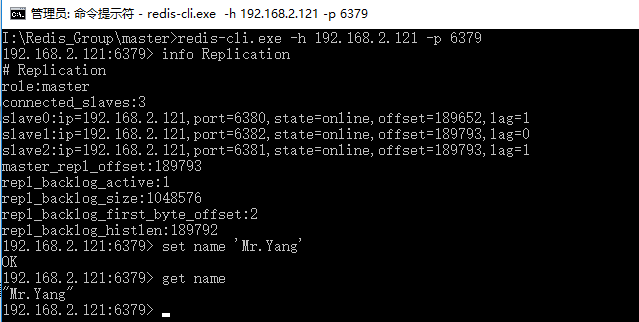
**设置值|获取值**

192.168.2.121:6379> set name 'Mr.Yang'

OK

192.168.2.121:6379> get name

"Mr.Yang"



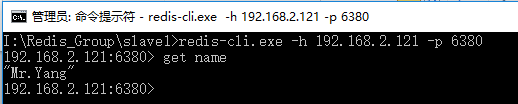
另起一个client连接slave看slave是否能够获取到master设置的值

I:\Redis\_Group\slave1>redis-cli.exe -h 192.168.2.121 -p 6380

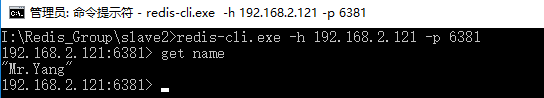
192.168.2.121:6380> get name

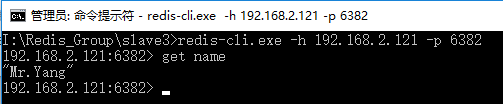
"Mr.Yang"

192.168.2.121:6380>

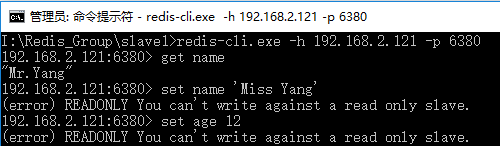


同理，我们开起client连接slave2、slave3同样也能获取到master设置的值





测试在slave1设置

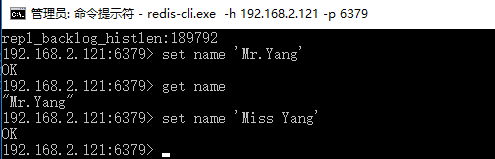


提示：

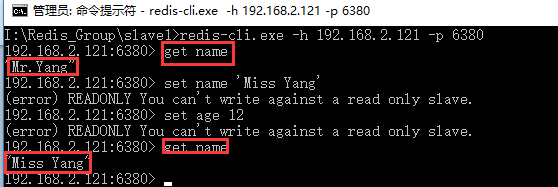
(error) READONLY You can't write against a read only slave.

Slave为只读不能够设置值

**Master修改值，看slave是否能同步**

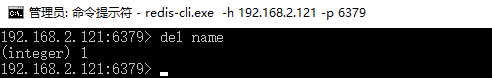


Slave1

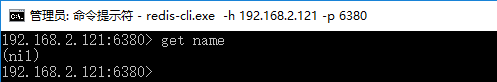


其他slave也一样.

Master删除值，看slave是否能同步



Slave



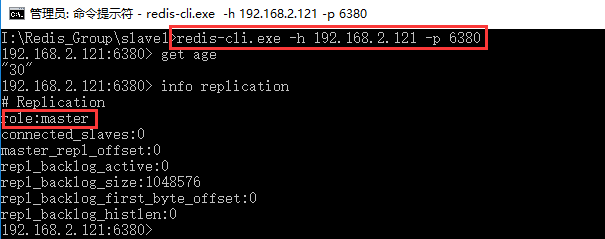
二：当master挂掉之后，看sentinel是否能够将其中一台slave由slave提升成master从而保证系统的稳定性

我们直接将启动master的窗口关闭，再看setinel将slave1、slave2、slave3当中那一个slave设置成master了。

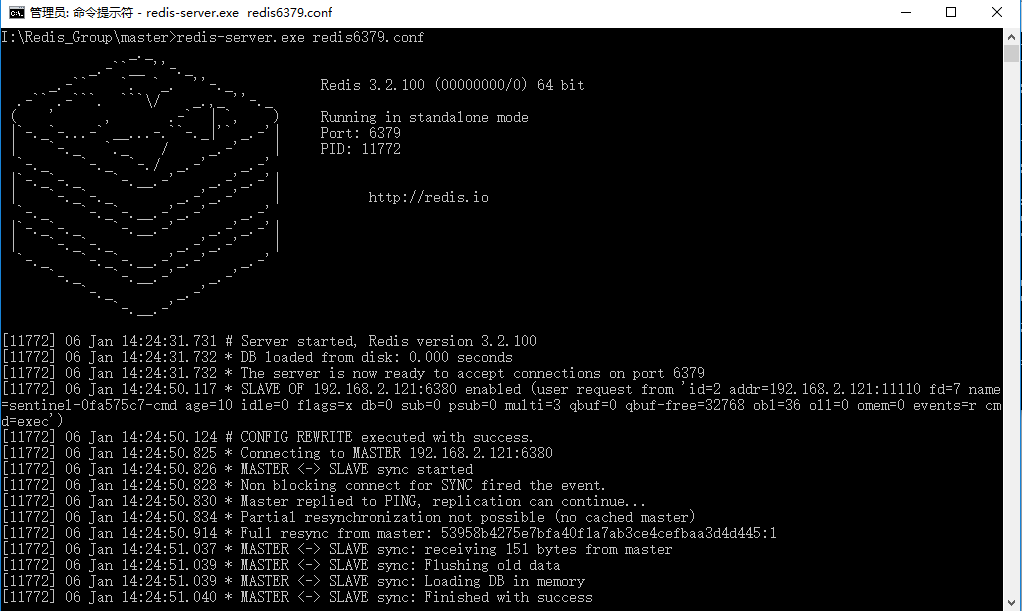
Sentinel将 slave1从slave变成master了

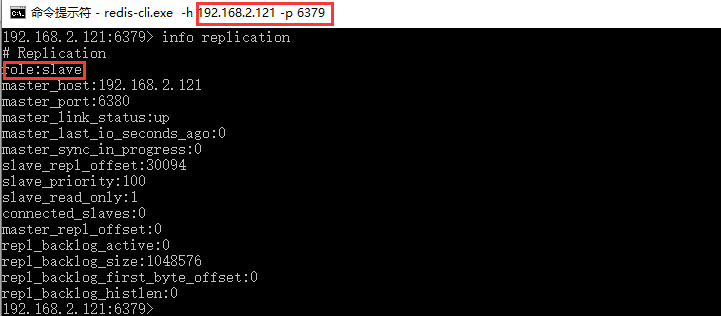
Slave1





重新开启原来已经关闭掉的master(6379端口)





可以看到它已经由master变成slave了

同时我们在原来的slave1(现在它已经变成master了)的连接信息中可以看到6379这个slave(原来的master)已经连接到它了。

