Linux下搭建redis服务，3个服务器，一个master，2个slave，3个sentinel

环境：三台linux运行在虚拟机中

Linux版本、平台：

[root@master ~]# uname -orsm

Linux 2.6.32-696.el6.x86\_64 x86\_64 GNU/Linux

目录准备

新建用户rediser并设置密码

[root@master ~]# useradd rediser

[root@master ~]# passwd rediser

将rediser加入sudoers

[root@master ~]# vim /etc/sudoers

切换到rediser用户并进入到其家目录

[root@master mongo]# su rediser

[mongoer@master mongo]$ cd

创建两个目录src用来存放源文件，app目录用来放置安装文件

[rediser@master ~]$ mkdir src

[rediser@master ~]$ mkdir app

进入到src目录并下载redis源代码：

[rediser@master ~]$ cd src

[rediser@master src]$ wget <http://download.redis.io/releases/redis-4.0.1.tar.gz>

将redis源代码解压到app目录

[rediser @master src]# tar -xzvf redis-4.0.1.tar.gz -C ../app/

预编译、编译、安装

进入到app目录进行操作预编译的准备

[rediser @master src]# cd ../app/

[rediser @master app]# ls

redis-4.0.1

[rediser @master app]# cd redis-4.0.1/

[rediser @master redis-4.0.1]# ls

00-RELEASENOTES CONTRIBUTING deps Makefile README.md runtest runtest-sentinel src utils

BUGS COPYING INSTALL MANIFESTO redis.conf runtest-cluster sentinel.conf tests

按照官方（https://redis.io/download）的文档，只需要在redis解压后目录执行make，就会将server和cli安装到src目录下。

[root@master redis-4.0.1]# make

为了方便使用，可以创建bin目录，将src中的redis-server、redis-cli移动到bin目录

[rediser@master redis-4.0.1]$ mv ./src/redis-server ./bin/

[rediser@master redis-4.0.1]$ mv ./src/redis-cli ./bin/

[rediser@master redis-4.0.1]$ cd bin/

[rediser@master bin]$ ls

redis-cli redis-server

配置和启动服务

进入到src目录，以默认的配置文件直接启动redis服务，

[rediser@master bin]$ ./redis-server

10736:C 10 Aug 20:53:35.712 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo

10736:C 10 Aug 20:53:35.712 # Redis version=4.0.1, bits=64, commit=00000000, modified=0, pid=10736, just started

10736:C 10 Aug 20:53:35.712 # Warning: no config file specified, using the default config. In order to specify a config file use ./redis-server /path/to/redis.conf

\_.\_

\_.-``\_\_ ''-.\_

\_.-`` `. `\_. ''-.\_ Redis 4.0.1 (00000000/0) 64 bit

.-`` .-```. ```\/ \_.,\_ ''-.\_

( ' , .-` | `, ) Running in standalone mode

|`-.\_`-...-` \_\_...-.``-.\_|'` \_.-'| Port: 6379

| `-.\_ `.\_ / \_.-' | PID: 10736

`-.\_ `-.\_ `-./ \_.-' \_.-'

|`-.\_`-.\_ `-.\_\_.-' \_.-'\_.-'|

| `-.\_`-.\_ \_.-'\_.-' | http://redis.io

`-.\_ `-.\_`-.\_\_.-'\_.-' \_.-'

|`-.\_`-.\_ `-.\_\_.-' \_.-'\_.-'|

| `-.\_`-.\_ \_.-'\_.-' |

`-.\_ `-.\_`-.\_\_.-'\_.-' \_.-'

`-.\_ `-.\_\_.-' \_.-'

`-.\_ \_.-'

`-.\_\_.-'

输出这个图表明redis启动正常

master配置编辑,配置文件在加压目录下：

[rediser@master redis-4.0.1]$ cp redis.conf redis.conf.bak #解压之前先备份

[rediser@master redis-4.0.1]$ vim redis.conf

修改ip绑定

bind 192.168.86.128

修改为后台服务启动

daemonize yes

修改pid文件路径

pidfile /home/rediser/app/redis-4.0.1/redis\_6379.pid

然后将解压目录（带安装文件）scp 到其他的两台节点（需要事先创建nginxer用户和准备好目录），并在这两台机器上启动服务查看是否成功。

这是node1（192.168.86.129）的redis-4.0.1/bin目录：

[rediser@node1 bin]$ ls

dump.rdb redis-cli redis-server

[rediser@node1 bin]$ ./redis-server #服务启动正常

在node2(192.168.86.130)上也启动正常

下面配置node1 和node2上的redis.conf 以作为master的从

与master配置不同的是指明master的ip和端口号以及bind的ip，其他配置一样

Master下redis的启动，指定修改的配置文件

[rediser@master redis-4.0.1]$ ./bin/redis-server redis.conf

10906:C 10 Aug 21:59:06.368 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo

10906:C 10 Aug 21:59:06.368 # Redis version=4.0.1, bits=64, commit=00000000, modified=0, pid=10906, just started

10906:C 10 Aug 21:59:06.368 # Configuration loaded

通过连接客户端测试：

[rediser@master redis-4.0.1]$ ./bin/redis-cli -h 192.168.86.128 #需要在指定ip(即配置文件中的bind的ip)

192.168.86.128:6379>

同样的，启动node1和node2上的redis服务

node1上：

[rediser@node1 redis-4.0.1]$ ./bin/redis-server redis.conf

4484:C 10 Aug 22:01:30.909 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo

4484:C 10 Aug 22:01:30.909 # Redis version=4.0.1, bits=64, commit=00000000, modified=0, pid=4484, just started

4484:C 10 Aug 22:01:30.909 # Configuration loaded

[rediser@node1 redis-4.0.1]$ ./bin/redis-cli -h 192.168.86.129

192.168.86.129:6379>

node2上：

[rediser@node2 redis-4.0.1]$ ./bin/redis-server ./redis.conf

4882:C 10 Aug 22:03:28.726 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo

4882:C 10 Aug 22:03:28.726 # Redis version=4.0.1, bits=64, commit=00000000, modified=0, pid=4882, just started

4882:C 10 Aug 22:03:28.726 # Configuration loaded

[rediser@node2 redis-4.0.1]$ ./bin/redis-cli -h 192.168.86.130

192.168.86.130:6379>

查看主从状态，使用info [SELECTION]命令查看

192.168.86.128:6379> info Replication

# Replication

role:master #这里maser表明连接的服务器是maser

connected\_slaves:2 #这里指明了两个连接的slave的信息

slave0:ip=192.168.86.129,port=6379,state=online,offset=406,lag=1

slave1:ip=192.168.86.130,port=6379,state=online,offset=420,lag=1

master\_replid:db33a43be1d3d1d7f5448ca1235c49ade832fcde

master\_replid2:0000000000000000000000000000000000000000

master\_repl\_offset:420

second\_repl\_offset:-1

repl\_backlog\_active:1

repl\_backlog\_size:1048576

repl\_backlog\_first\_byte\_offset:1

repl\_backlog\_histlen:420

由此可见，redis的slave可以动态的添加和删除

可以连上master 存储数据并在从服务器上获取验证

**接下来配置sentinel**

master上：

在解压目录中已经存在了sentinel.conf文件，这就是sentinel的配置文件，修改该文件，指定监控的

修改sentinel.conf

bind 127.0.0.1 192.168.86.128 #设置绑定ip

port 26379 #设置sentinel的端口

sentinel monitor mymaster 192.168.86.128 6379 2

sentinel monitor #为sentinel的监控配置项

myredismaster #为监控的master的名字（可以任意取名）

192.168.86.128 #为master的ip

6379 #为端口

2 #2表示当master出现故障后可以决策（选取新的master并执行同步操作）的最小sentinel个数

daemonize yes #如果没有则增加此命令

node1和node2上做相似的处理

在master、node1和node2上分别启动sentinel

我们可以将安装的redis-sentinel移动到bin，也可以使用server以sentinel模式来启动sentinel，这里采用第二种

master上：

[rediser@master redis-4.0.1]$ ./bin/redis-server ./sentinel.conf --sentinel

连接测试

[rediser@master redis-4.0.1]$ ./bin/redis-cli -p 26379

127.0.0.1:26379>

node1

[rediser@node1 redis-4.0.1]$ ./bin/redis-server sentinel.conf --sentinel

4559:X 10 Aug 22:32:32.161 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo

4559:X 10 Aug 22:32:32.161 # Redis version=4.0.1, bits=64, commit=00000000, modified=0, pid=4559, just started

4559:X 10 Aug 22:32:32.161 # Configuration loaded

连接测试

node2

[rediser@node2 redis-4.0.1]$ ./bin/redis-server sentinel.conf --sentinel

4969:X 10 Aug 22:31:40.820 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo

4969:X 10 Aug 22:31:40.820 # Redis version=4.0.1, bits=64, commit=00000000, modified=0, pid=4969, just started

4969:X 10 Aug 22:31:40.820 # Configuration loaded

连接测试

[rediser@node2 redis-4.0.1]$ ./bin/redis-cli -p 26379

127.0.0.1:26379>

在master的sentinel上：

127.0.0.1:26379> sentinel masters

1) 1) "name"

2) "mymaster" #配置中指定的master名（实际每个sentinel可以监控多个master）

3) "ip"

4) "192.168.86.128" #master的ip

5) "port"

6) "6379"

7) "runid"

8) "336e524fb115fbd4395d8f26e79edc62fe776dfb"

9) "flags"

10) "master"

11) "link-pending-commands"

12) "0"

13) "link-refcount"

14) "1"

15) "last-ping-sent"

16) "0"

17) "last-ok-ping-reply"

18) "759"

19) "last-ping-reply"

20) "759"

21) "down-after-milliseconds"

22) "30000"

23) "info-refresh"

24) "6847"

25) "role-reported"

26) "master"

27) "role-reported-time"

28) "97206"

29) "config-epoch"

30) "0"

31) "num-slaves"

32) "2"

33) "num-other-sentinels"

34) "0"

35) "quorum" #决策最小sentinel数量

36) "2"

37) "failover-timeout"

38) "180000"

39) "parallel-syncs"

40) "1"

故障演练