# 安装

## 下载

[github](https://so.csdn.net/so/search?q=github&spm=1001.2101.3001.7020)上可以下载Windows Redis5.0终版，有安装包形式和zip两种，下载地址：

<https://github.com/tporadowski/redis/releases>

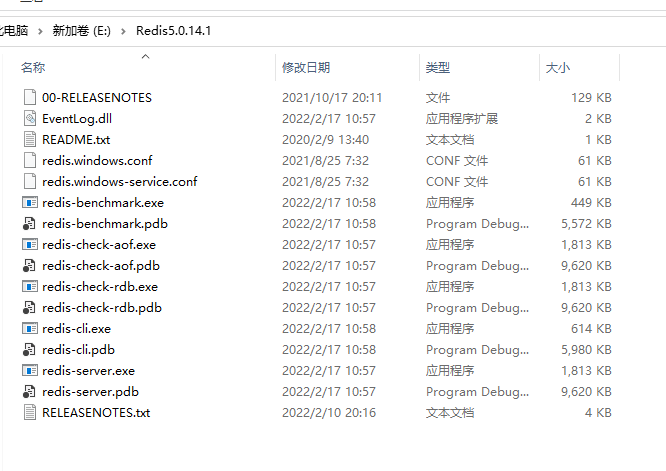


下载下来的文件如下：



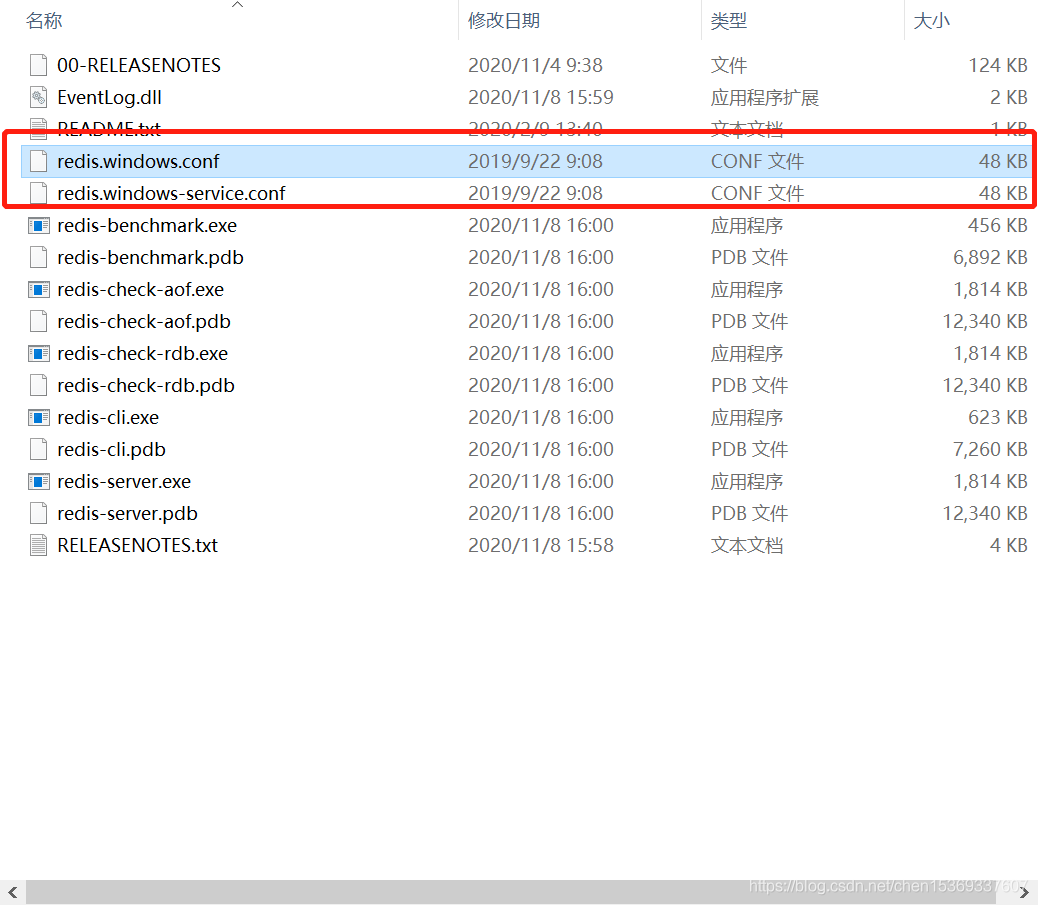
## 解压

先新建一个文件夹用于存放Redis压缩包解压后的文件，新建一个文件夹为：redis-5.0.10,然后将下载好的zip包解压到新建的redis-5.0.10文件夹中。



## 重要文件介绍

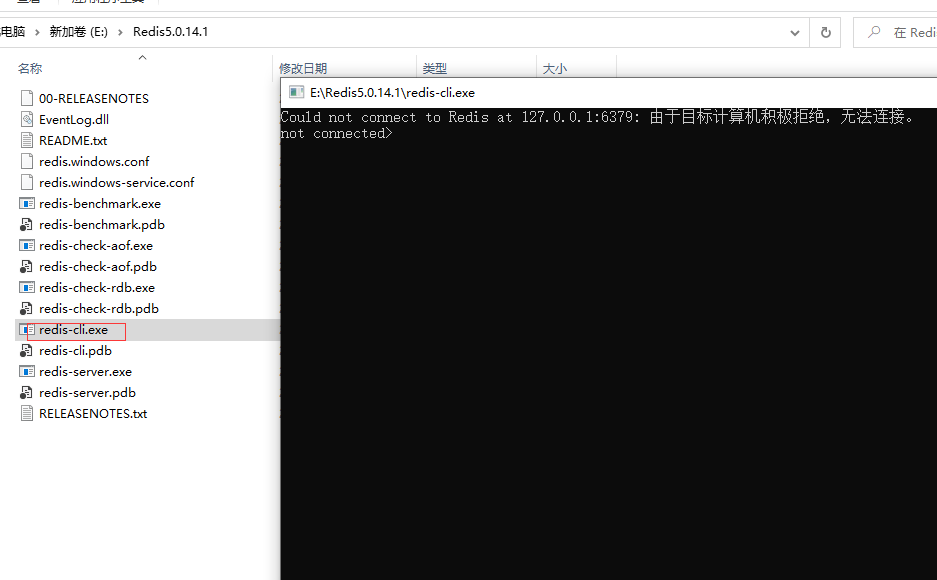
redis配置文件为：redis.windows.conf、redis.windows-service.conf  
redis绑定地址以及默认端口配置都可在这两个文件中找到



# 启动

## 启动命令行

redis命令行工具 redis-cli，可以启动redis命令行，执行redis命令；双击redis-cli.exe

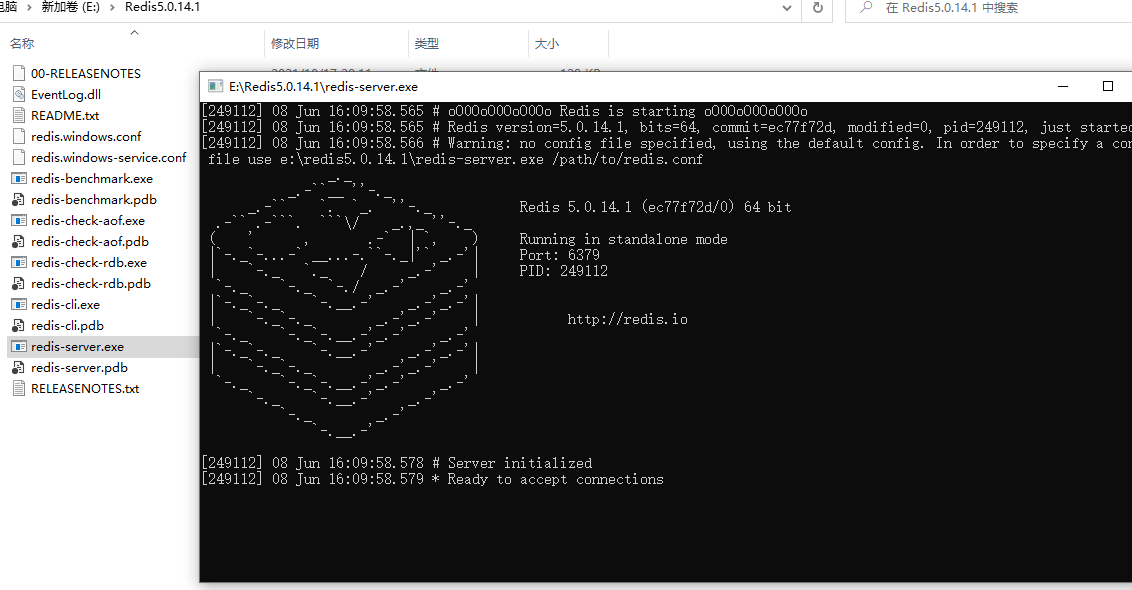


## 启动服务

redis 服务，用于启动redis服务，双击redis-server.exe

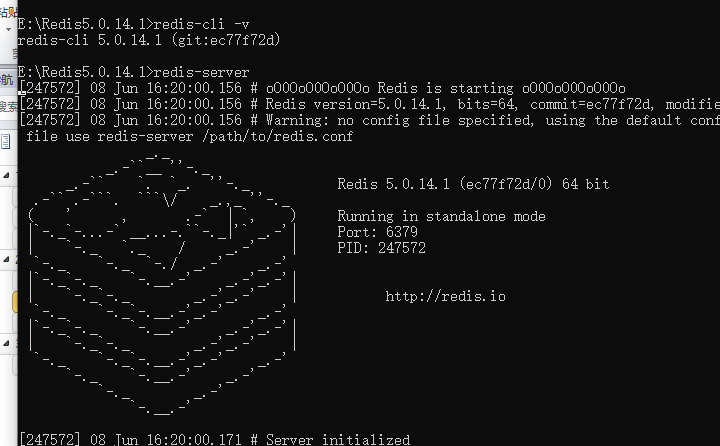
或者

**redis-server.exe redis.windows.conf （建议）**

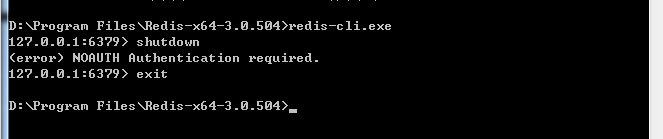


或者在cmd下，进入redis安装目录，执行如下命令（配置了环境变量后cmd下直接执行）

redis-server



如果启动失败，继续输入**redis-cli.exe**，输入**shutdown**结束，输入**exit**退出；

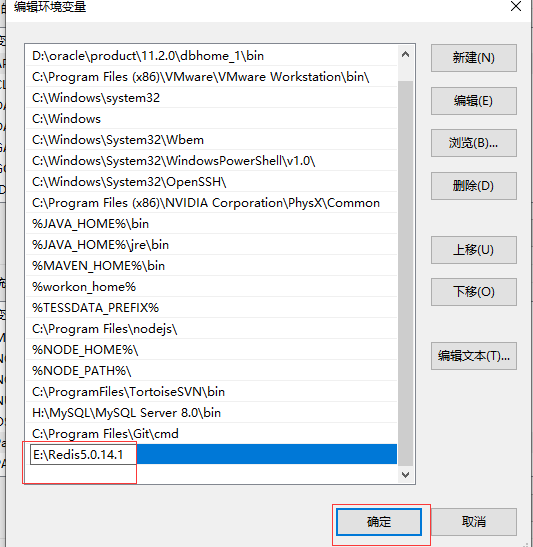


## 环境变量配置

通过配置环境的方式，让我们可以通过命令行的方式操作redis,而不用将redis在系统服务中进行操作。  
（1）复制redis的解压路径，例如我的为：E:\Redis5.0.14.1  
（2）点击【此电脑】-【属性】，找到【高级系统设置】，点击【环境变量】



（3）找到【path】环境变量，新建一个环境变量值为redis的解压路径



（4）确定，保存退出

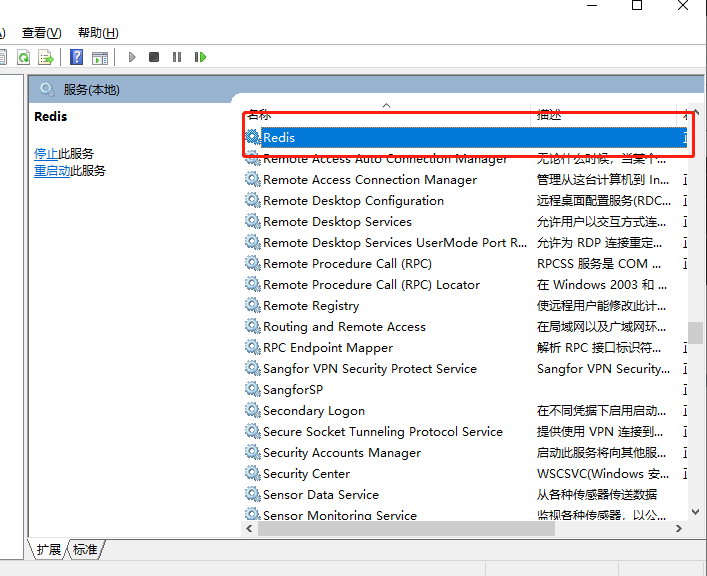
## 添加到服务

在redis安装目录下执行：**redis-server.exe --service-install redis.windows.conf --loglevel verbose**

**redis-server.exe --service-install redis.windows.conf --loglevel verbose**



完成上述操作后，就会出现服务列表中的Redis标识



## 添加到服务后操作

redis服务添加后，以下为常用操作命令（**redis安装目录下执行**）

**卸载服务**：redis-server.exe --service-uninstall

**开启服务**：redis-server.exe --service-start

**停止服务**：redis-server.exe --service-stop

# 验证与连接redis

## 验证

打开cmd，进入redis安装目录，执行

redis-cli -v



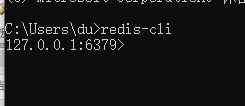
出现上图redis-cli命令行版本信息就证明环境变量配置好了

如果已经配置了环境变量，可以打开cmd直接执行

## 连接

在上述redis服务启动成功且没有关闭的情况下，启动一个新的cmd窗口。  
输入 ：

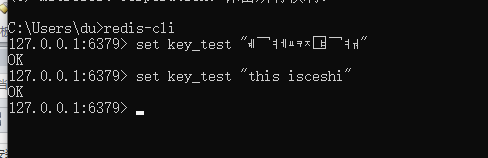
redis-cli



## 测试

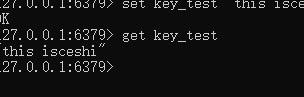
设置一个key为key-test ,value 为test-value

set key\_test “this isceshi”



然后get值

get key\_test



# 安装客户端RedisDesktop

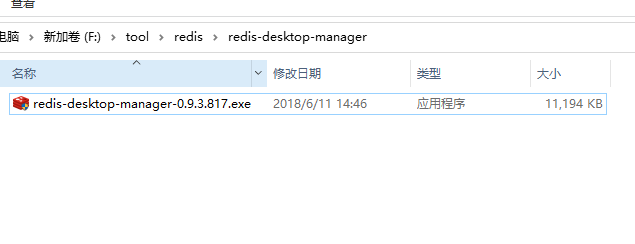
## 下载

RedisDesktop官方下载地址： https://redisdesktop.com/download 但是需要付费

这里提供一个可以免费下载的地址：https://pan.baidu.com/s/1ExBlD79pfypnqAeP1sXybg

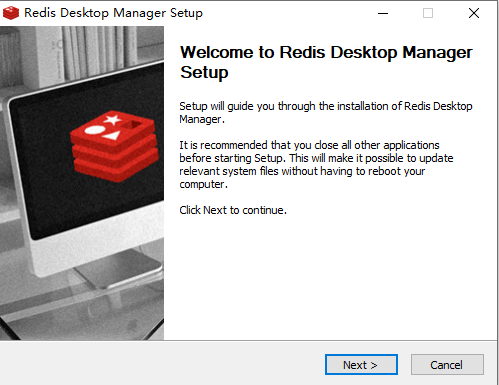
下载后解压如下：

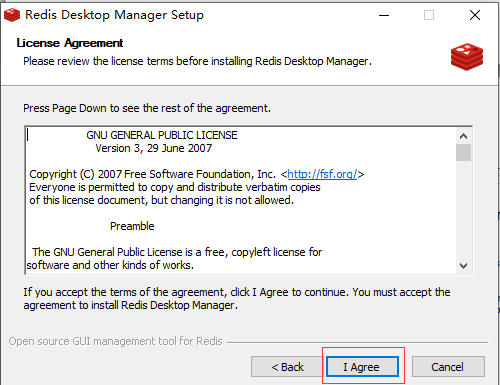
https://img-blog.csdnimg.cn/2019011810373537.png

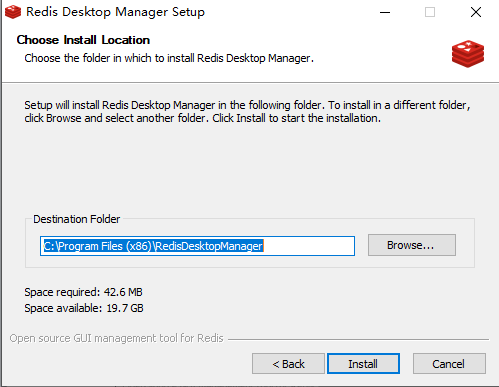


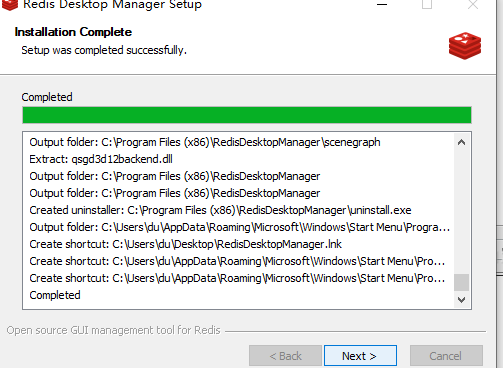
## 安装

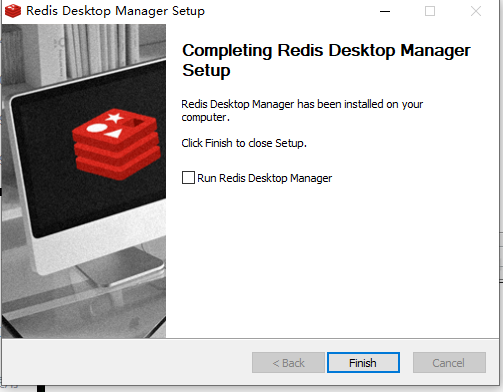
双击redis-desktop-manager-0.9.3.817.exe





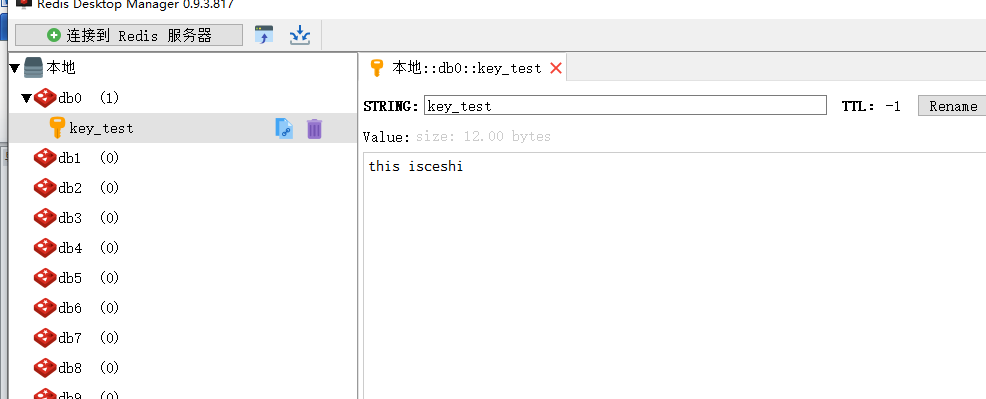






安装完成后运行，就可以选择连接到Redis服务器。

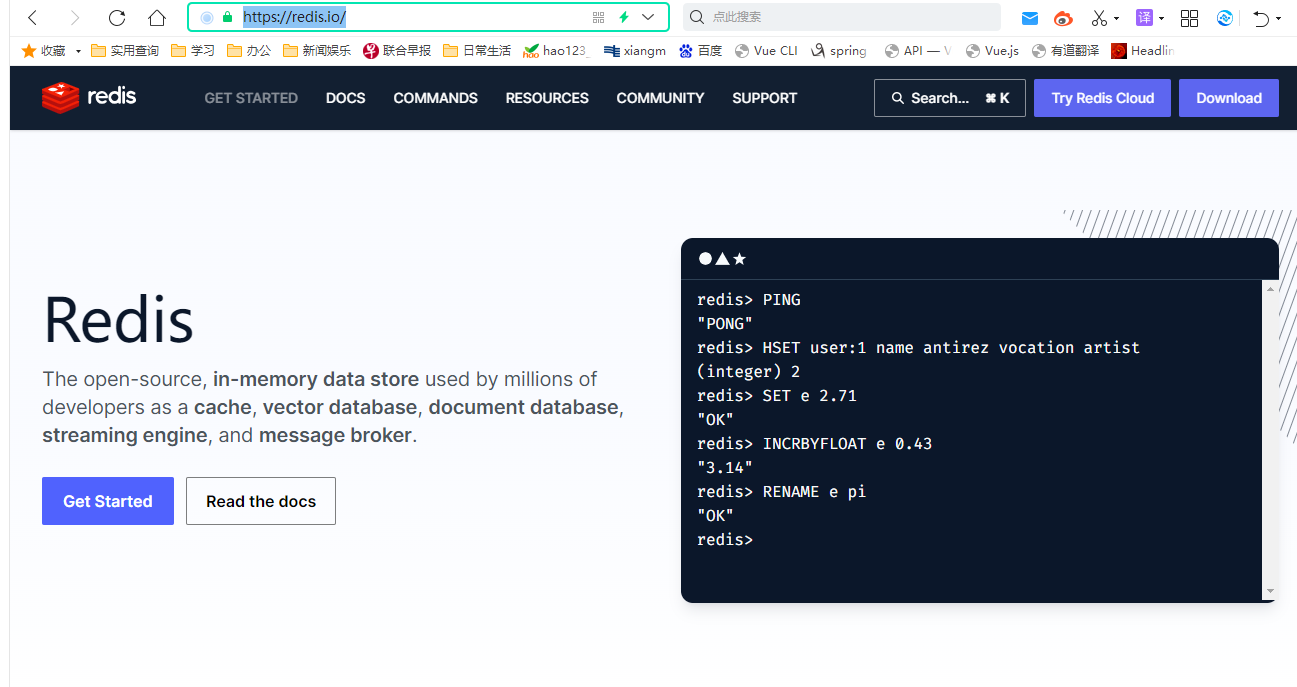


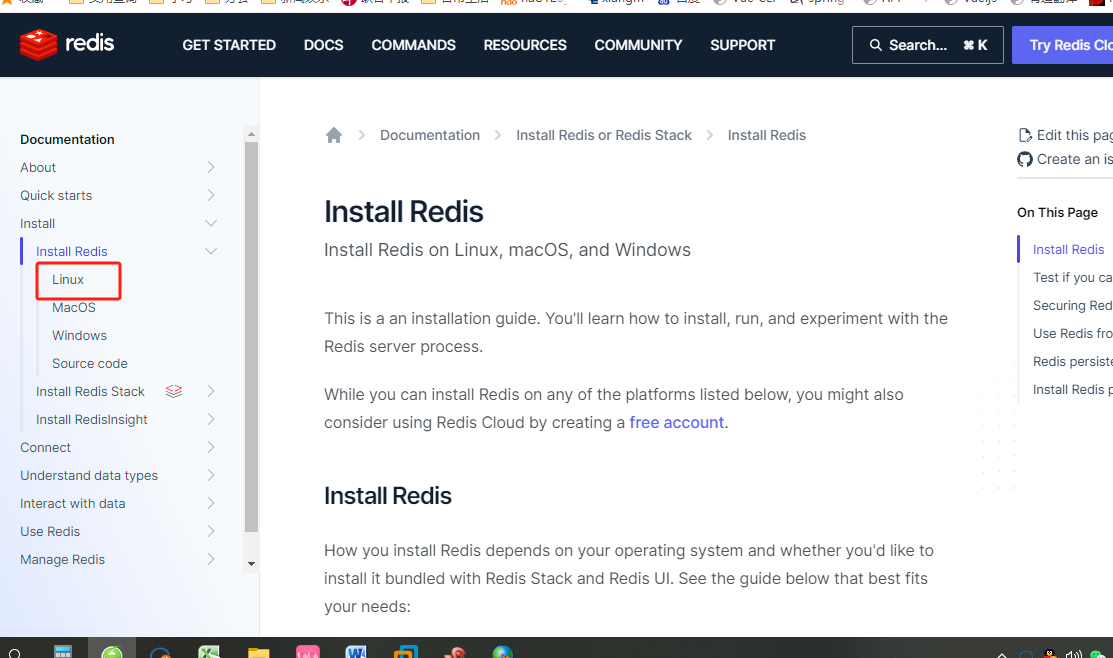


# Linux下安装Redis

## apt下载安装（直接命令安装）

首先，访问[Redis官网](https://redis.io/) https://redis.io/，点击首页的【Get Started】，然后点击[Install Redis on Linux](https://redis.io/docs/getting-started/installation/install-redis-on-linux)





依次输入如下脚本安装

sudo apt install lsb-release curl gpg

curl -fsSL https://packages.redis.io/gpg | sudo gpg --dearmor -o /usr/share/keyrings/redis-archive-keyring.gpg

echo "deb [signed-by=/usr/share/keyrings/redis-archive-keyring.gpg] https://packages.redis.io/deb $(lsb\_release -cs) main" | sudo tee /etc/apt/sources.list.d/redis.list

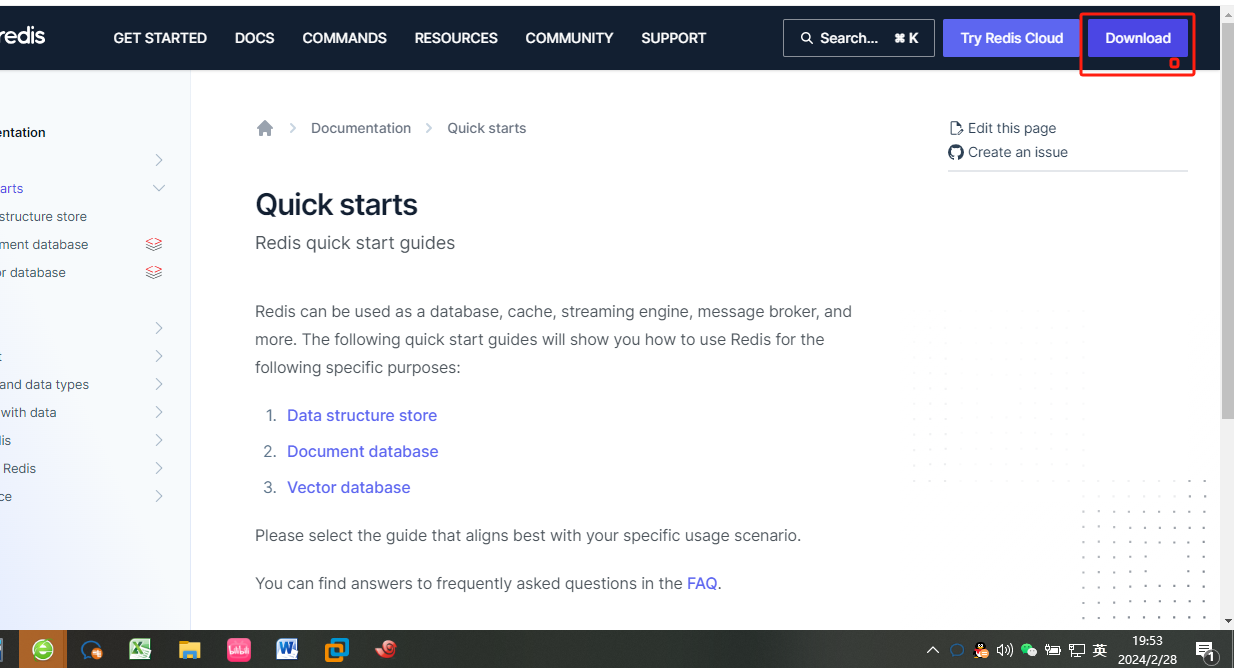
sudo apt-get update

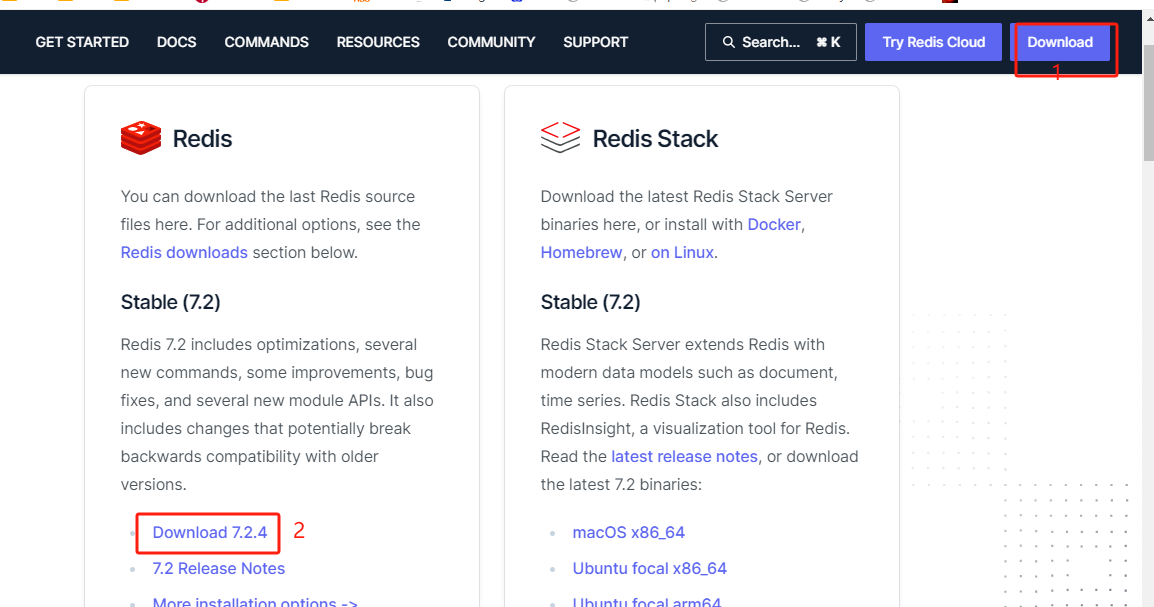
sudo apt-get install redis

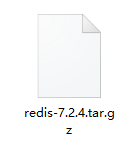
## Tar下载安装

### 5.2.1下载

首先，访问[Redis官网](https://redis.io/) https://redis.io/，点击首页的【DownLoad】，然后点击[Install Redis on Linux](https://redis.io/docs/getting-started/installation/install-redis-on-linux)





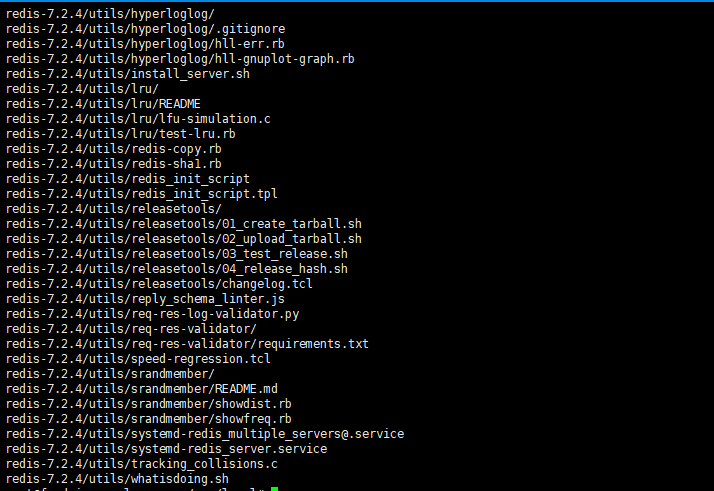
下载后如下：  


### 5.2.2安装

上传下载的安装文件到linux系统/usr/local目录下

**1.解压**

tar -zxvf redis-7.2.4.tar.gz



安装成功后，系统会有一个文件夹redis-7.2.4/



**2、编译**

**如果没有安装gcc 编译器，需要安装，命令如下：**

**yum –y install gcc**

进入redis解压之后的目录中，执行【make】命令，对redis的代码进行编译。

因为redis是使用C语言编写的，我们下载的redis压缩包中就是redis的源码，也就是一些【.c】、【.h】的头文件、源代码，所以需要在linux平台下，使用make命令将这些C语言的源代码进行编译，执行结束之后，看到下面情况就表示编译成功了。

在目录/usr/local/redis-7.2.4下执行make

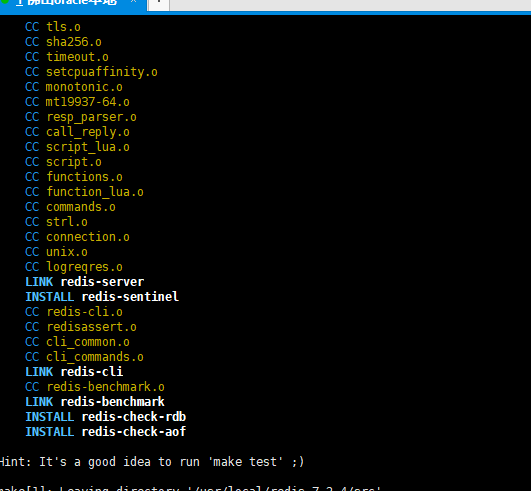
make

或者 （指定编译库）

make MALLOC=libc



执行如下：

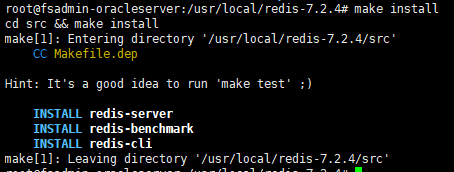


**3、安装**

在/usr/local/redis-7.2.4目录下

make install

安装完成如下：



或者指定安装目录

make PREFIX=/home/redis/redis-7.2.4 install

执行命令【make PREFIX=/home/redis/redis-7.2.4 install】安装redis，这个命令的作用是生成可执行的redis代码，PREFIX参数用于设置生成的redis可执行命令保存在哪个目录下面，不设置PREFIX参数，Linux会默认将redis相关的安装文件保存在下面几个目录里面：

/usr/local/bin：保存redis的可执行文件。

/usr/local/lib：保存redis依赖文件。

/usr/local/share：保存redis其他文件。

/usr/local/etc：保存redis配置文件。

默认的安装目录太乱了，我们最好还是指定统一的目录保存，这样便于管理。

**4、启动**

在/usr/local/redis-7.2.4目录下

执行：

redis-server



CRTAL+C 退出

安装成功后，在任何目录下都可以启动

redis-server

### 5.2.3启动和关闭

**1、启动服务**

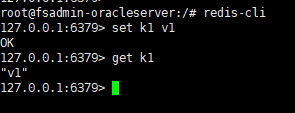
redis-server

**2、启动服务**

redis-server &

**3、启动客户端（命令行客户端）**

redis-cli



CRTAL+C 退出

**4、查看redis 进程**

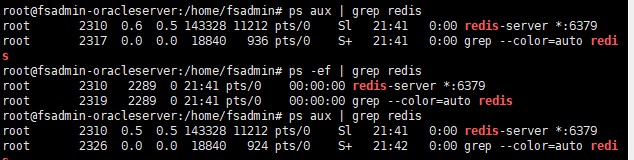
**ps aux | grep redis**

**或者**

**ps ef| grep redis**

或者

lsof –I:6379 需要安装lsof



2310 为进程号

kill -9 2310

### 5.2.4 多实例启动

1.通过redis.conf启动，必须在安装目录下/usr/local/redis-7.2.4

redis-server redis.conf &

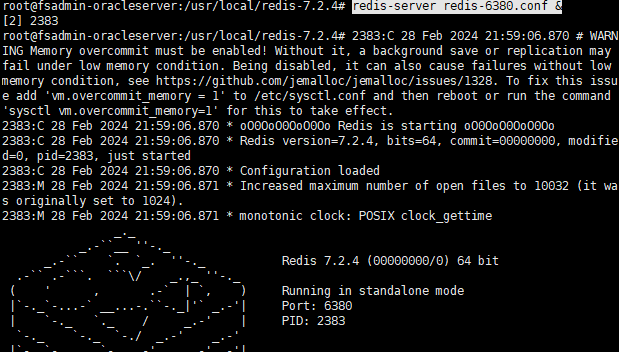
2.在创建一个conf文件，在conf中修改端口为6380，进行多实例启动

cp redis.conf redis-6380.conf

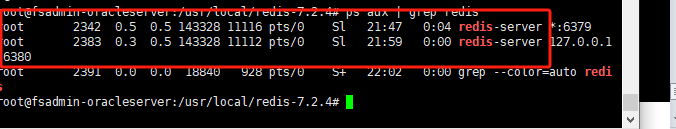
vi 修改端口

启动

redis-server redis-6380.conf &



启动后查看进程



这样可以启动多个进程

### 5.2.5 Redis 配置

1. Redis默认不是以守护进程的方式运行，可以通过该配置项修改，使用yes启用守护进程，即后台启动

**daemonize no**

2. 当Redis以守护进程方式运行时，Redis默认会把pid写入/var/run/redis.pid文件，可以通过pidfile指定

**pidfile /var/run/redis.pid**

3. 指定Redis监听端口，默认端口为6379，作者在自己的一篇博文中解释了为什么选用6379作为默认端口，因为6379在手机按键上MERZ对应的号码，而MERZ取自意大利歌女Alessia Merz的名字

**port 6379**

4. 绑定的主机地址

**bind 127.0.0.1**

5.当 客户端闲置多长时间后关闭连接，如果指定为0，表示关闭该功能

**timeout 300**

6. 指定日志记录级别，Redis总共支持四个级别：debug、verbose、notice、warning，默认为verbose

**loglevel verbose**

7. 日志记录方式，默认为标准输出，如果配置Redis为守护进程方式运行，而这里又配置为日志记录方式为标准输出，则日志将会发送给/dev/null

**logfile stdout**

8. 设置数据库的数量，默认数据库为0，可以使用SELECT <dbid>命令在连接上指定数据库id

**databases 16**

9. 指定在多长时间内，有多少次更新操作，就将数据同步到数据文件，可以多个条件配合

**save <seconds> <changes>**

    Redis默认配置文件中提供了三个条件：

**save 900 1**

**save 300 10**

**save 60 10000**

    分别表示900秒（15分钟）内有1个更改，300秒（5分钟）内有10个更改以及60秒内有10000个更改。

10. 指定存储至本地数据库时是否压缩数据，默认为yes，Redis采用LZF压缩，如果为了节省CPU时间，可以关闭该选项，但会导致数据库文件变的巨大

**rdbcompression yes**

11. 指定本地数据库文件名，默认值为dump.rdb

**dbfilename dump.rdb**

12. 指定本地数据库存放目录

**dir ./**

13. 设置当本机为slav服务时，设置master服务的IP地址及端口，在Redis启动时，它会自动从master进行数据同步

**slaveof <masterip> <masterport>**

14. 当master服务设置了密码保护时，slav服务连接master的密码

**masterauth <master-password>**

15. 设置Redis连接密码，如果配置了连接密码，客户端在连接Redis时需要通过AUTH <password>命令提供密码，默认关闭

**requirepass foobared**

16. 设置同一时间最大客户端连接数，默认无限制，Redis可以同时打开的客户端连接数为Redis进程可以打开的最大文件描述符数，如果设置 maxclients 0，表示不作限制。当客户端连接数到达限制时，Redis会关闭新的连接并向客户端返回max number of clients reached错误信息

**maxclients 128**

17. 指定Redis最大内存限制，Redis在启动时会把数据加载到内存中，达到最大内存后，Redis会先尝试清除已到期或即将到期的Key，当此方法处理 后，仍然到达最大内存设置，将无法再进行写入操作，但仍然可以进行读取操作。Redis新的vm机制，会把Key存放内存，Value会存放在swap区

**maxmemory <bytes>**

18. 指定是否在每次更新操作后进行日志记录，Redis在默认情况下是异步的把数据写入磁盘，如果不开启，可能会在断电时导致一段时间内的数据丢失。因为 redis本身同步数据文件是按上面save条件来同步的，所以有的数据会在一段时间内只存在于内存中。默认为no

**appendonly no**

19. 指定更新日志文件名，默认为appendonly.aof

**appendfilename appendonly.aof**

20. 指定更新日志条件，共有3个可选值：   
    **no**：表示等操作系统进行数据缓存同步到磁盘（快）   
    **always**：表示每次更新操作后手动调用fsync()将数据写到磁盘（慢，安全）   
    **everysec**：表示每秒同步一次（折衷，默认值）

**appendfsync everysec**

21. 指定是否启用虚拟内存机制，默认值为no，简单的介绍一下，VM机制将数据分页存放，由Redis将访问量较少的页即冷数据swap到磁盘上，访问多的页面由磁盘自动换出到内存中（在后面的文章我会仔细分析Redis的VM机制）

**vm-enabled no**

22. 虚拟内存文件路径，默认值为/tmp/redis.swap，不可多个Redis实例共享

**vm-swap-file /tmp/redis.swap**

23. 将所有大于vm-max-memory的数据存入虚拟内存,无论vm-max-memory设置多小,所有索引数据都是内存存储的(Redis的索引数据 就是keys),也就是说,当vm-max-memory设置为0的时候,其实是所有value都存在于磁盘。默认值为0

**vm-max-memory 0**

24. Redis swap文件分成了很多的page，一个对象可以保存在多个page上面，但一个page上不能被多个对象共享，vm-page-size是要根据存储的 数据大小来设定的，作者建议如果存储很多小对象，page大小最好设置为32或者64bytes；如果存储很大大对象，则可以使用更大的page，如果不 确定，就使用默认值

**vm-page-size 32**

25. 设置swap文件中的page数量，由于页表（一种表示页面空闲或使用的bitmap）是在放在内存中的，，在磁盘上每8个pages将消耗1byte的内存。

**vm-pages 134217728**

26. 设置访问swap文件的线程数,最好不要超过机器的核数,如果设置为0,那么所有对swap文件的操作都是串行的，可能会造成比较长时间的延迟。默认值为4

**vm-max-threads 4**

27. 设置在向客户端应答时，是否把较小的包合并为一个包发送，默认为开启

**glueoutputbuf yes**

28. 指定在超过一定的数量或者最大的元素超过某一临界值时，采用一种特殊的哈希算法

**hash-max-zipmap-entries 64**

**hash-max-zipmap-value 512**

29. 指定是否激活重置哈希，默认为开启（后面在介绍Redis的哈希算法时具体介绍）

**activerehashing yes**

30. 指定包含其它的配置文件，可以在同一主机上多个Redis实例之间使用同一份配置文件，而同时各个实例又拥有自己的特定配置文件

**include /path/to/local.conf**

## Redis数据类型



## 常用命令

1. redis-benchmark  性能测试工具
2. redis-check-aof  检查AOF日志
3. redis-check-dump 检查RDB日志
4. redis-cli        启动命令行客户端
5. redis-cli –p 6380 启动端口为6380的客户端
6. redis-sentinel
7. redis-server     启动Redis服务
8. string类型命令

set key value #设置值

get key #获取值

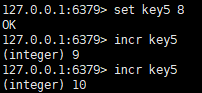
mset key1 value1 key2 value2 #批量设置值



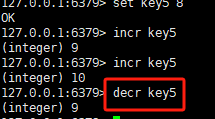
mget key1 key2



Incr key #key值递增



decr key #key值递减

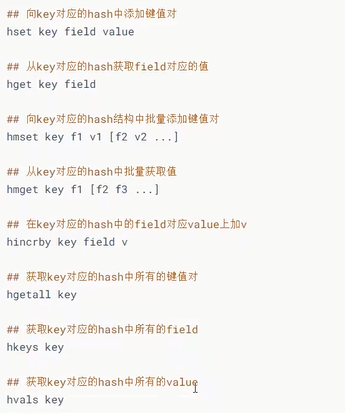




setnx key value #只设置不修改



1. hash 常用指令



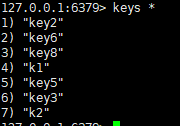


1. List 常用指令

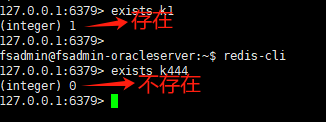


1. 查询命令：

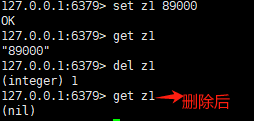
keys \* #查找所有key



exists key # 判断key是否存在

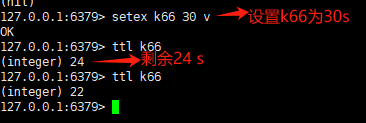


del key # 删除key



setex k66 30 v #设置key时长为30s

ttl k66 #查看key剩余时长

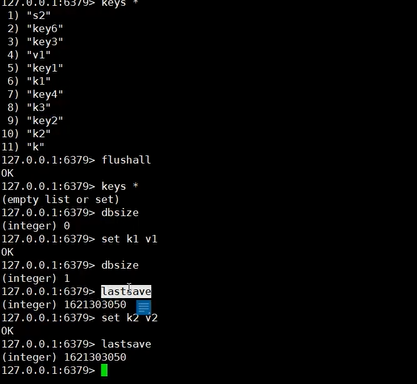


expire k66 30 #重置过期时间

1. Redis库操作







# Redis的持久化

Redis是基于内存操作，但作为一个数据库也具备数据的持久化能力;但是为了实现高效的读写操作，并不会即时进行数据的持久化，而是按照一定的规则进行持久化操作的--持久化策略

Redis提供了2中持久化策略:

* RDB (Redis DataBase)
* AQF(Append only File)

## RDB

### 6.1.1原理介绍

Redis 默认的持久化方式是 RDB ，并且默认是打开的。RDB 的保存有方式分为主动保存与被动保存。主动保存可以在 redis-cli 中输入 save 即可；被动保存需要满足配置文件中设定的触发条件，目前官方默认的触发条件可以在 redis.conf 中看到：

save 900 1

save 300 10

save 60 10000

其含义为：

服务器在900秒之内，对数据库进行了至少1次修改

服务器在300秒之内，对数据库进行了至少10次修改。

服务器在60秒之内，对数据库进行了至少10000次修改。

在满足特定的redis触发条件时，将内存中的数据以数据快照的形式存储到rdb文件中

### 6.1.2关闭RDB

打开：rdbcompression yes

关闭：rdbcompression no

或者

在配置文件redis.conf中，将这 3 个配置注释掉，并新增 save ""即可：

save ""

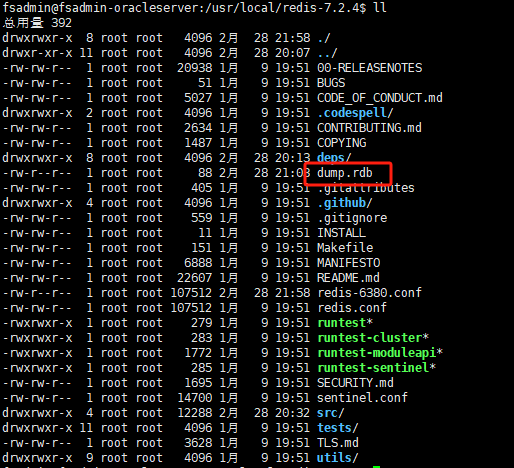
# save 900 1

# save 300 10

# save 60 10000

### 6.1.3配置RDB 数据存储文件

dbfilename dump.rdb



* 特点（缺点）

1）如果Rides是服务器出现故障，可能导致数据丢失。

2）RDB采用的是数据快照形式进行持久化，不适合实时性持久化；

3）如果数据量巨大，在RDB持久化过程中生成数据快照的子进程执行时间过长，会导致redis卡顿，因此saveo时间周期设置不宜过短。

* 优点

1）但在数据量较小的情况下，执行速度较快。

2）由于RDB是以数据快照的形式进行保存的，我们可以通过拷贝rdb文件轻松实现redis数据移植

## AOF

### 6.2.1原理介绍

AOF 方式默认每秒钟备份1次，频率很高，它的操作方式是以追加的方式记录日志而不是数据，并且它的重写过程是按顺序进行追加，所以它的文件内容非常容易读懂。可以在某些需要的时候打开 AOF 文件对其编辑，增加或删除某些记录，最后再执行恢复操作。

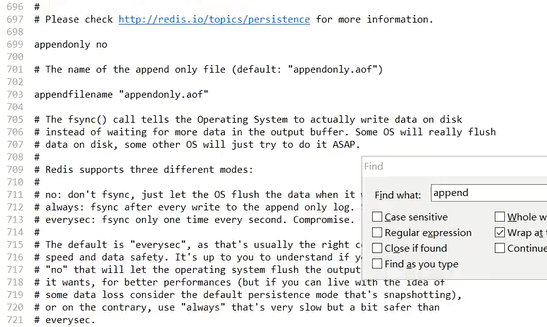
Redis将每一个成功的写操作写入到aor文件中，当redis重启的时候就执行aor文件中的指令以恢复数据

### 6.2.2开启

默认是不开启 AOF 的，如果想要启用则需要到 redis.conf 配置文件中开启，打开 redis.conf:

然后在文件中找到 appendonly 并将 no 改为 yes：

appendonly yes



即为开启了 AOF 方式的持久化。

### 6.2.3设置同步策略

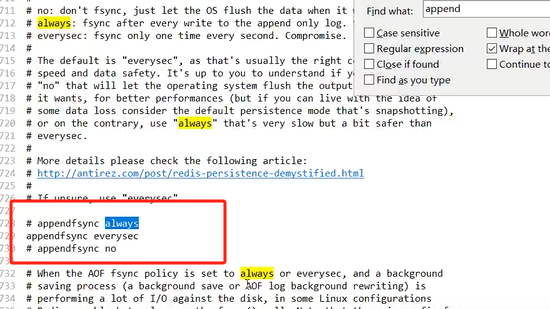
在redis.conf 配置文件中，设置如下属性：

appendfsync always # 每次有数据修改发生时都会写入AOF文件（安全但是费时）。

appendfsync everysec # 每秒钟同步一次，该策略为AOF的缺省策略。

appendfsync no # 从不同步。高效但是数据不会被持久化。

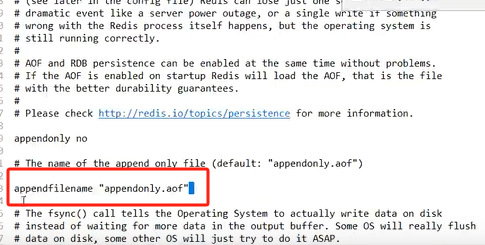
默认配置是 everysec，你可以根据需求进行调整，只能设置一个属性（三选一）



### 6.2.4自定义AOF记录文件的文件名

在redis.conf 配置文件中，设置如下属性：

appendfilename "appendonly.aof"



Redis 设置有默认的文件名，在配置中显示为：appendfilename "appendonly.aof"

可以让其保持默认名字，也可以指定其他的文件名，比如：

appendfilename "RNGLetme.aof"

将appendonly、appendfsync和appendfilename设置好并保存。重新启动 Redis 服务

### 6.2.5 AOF优缺点

1）也可以通过拷贝a0f文件进行redis数据移植中

2）AOf存储的指令，而且会对指令进行整理;而RDB直接生成数据快照，在数据量不大时RDB比较快

3）AOf是对指令文件进行增量更新，更适合实时性持久化

4）redis官方建议同时开启2中持久化策略，如果同时存在AOF文件和RDB文件的情况下、AOF优先

# Java 连接Redis配置

## 设置配置文件

Java应用连接Redis，首先要将我们的Redis设置允许远程连接，需要修改 redis.conf配置文件，步骤如下：

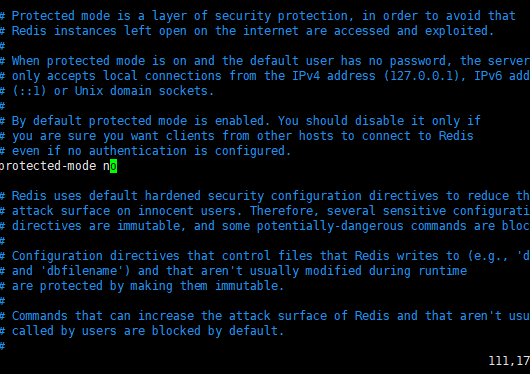
1. 建议创建新的conf文件

cp redis.conf redis-6379.conf

1. 关闭保护模式

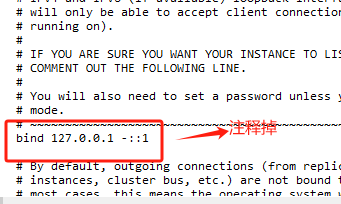
在配置文件中，修改protected-mode属性，默认为 yes，关闭设置为no

protected-mode no



1. 将bing注释掉，默认本地访问

# bind 127.0.0.1 ::1



1. 可以设置密码：

requirepass 密码，例如：

requirepass 123456

1. 重新启动

redis-server redis-6379.conf &

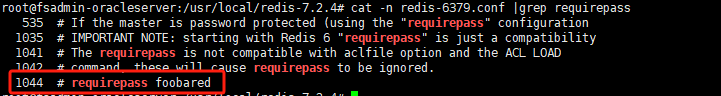
## 设置配置文件技巧

### 查找配置文件修改位置

命令：cat -n 文件名|grep 关键字

例如

cat -n redis-6379.conf |grep requirepass



### 根据查找修改位置修改

命令：vim +行号 文件名称

例如：

vim +1044 redis-6379.conf

# Redis主从复制配置

如果一个服务器只有一个节点（只用一个物理服务器部署整个程序），那么出现以下问题：

可用性：如果这个机器挂了，意味着服务就中断了.

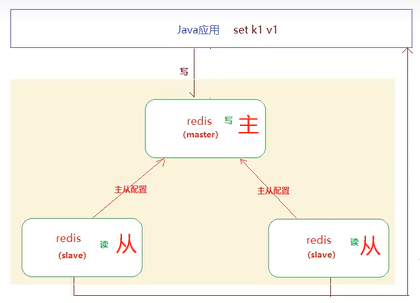
性能：支持的并发量也是比较有限的.

因此引入分布式系统，主要也是为了解决上述的单点问题。 分布式系统中，往往需要多个服务器来部署 redis 服务，从而构成 redis 集群，此时就可以让这个集群给整个分布式系统中的其他服务，提供更稳定 / 更高效的数据存储功能.

本节介绍redis集群模式之一主从模式

在 redis 主从模式下，由一个主节点和多个从节点构成，从节点通过复制主节点得到，并且后续主节点数据发生变化，从节点也必须 “听” 主节点的（从节点的数据跟随主节点一起变化，保持一致） .

如果我修改了从节点的数据呢？redis 主从模式中，从节点上的数据不允许修改，只能读取数据，而主节点既可以进行 “写操作” 也可以进行 “读操作” ~ 为什么这么搞呢？实际上更准确的说，主从模式，主要是针对 “读操作” 进行 并发量 & 可用性 的提高，因为实际的业务场景中，读操作往往比写操作更频繁~



## 创建主从实例配置文件

配置 redis 主从结构，需要启动多个 redis 服务器，分配在一个个单独的主机上（分布式），但是考虑到我们目前每个人，大概只有一个云服务器，因此就来实现一下，在一个服务器主机上，运行多个 redis-server 进程。

我们配置一个主实例和两个从实例

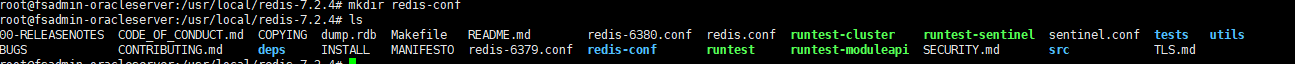
### 8.1.1建立配置文件目录

在redis安装目录下创建一个目录redi-conf，用于存放redis主从配置文件

mkdir redis-conf

例如：

root@fsadmin-oracleserver:/usr/local/redis-7.2.4# mkdir redis-conf



### 8.1.2建立主服务配置文件

我们找到一个原始的redis.conf文件，去掉注释和空白内容，复制到刚才创建的配置文件目录

**过滤掉注释和空白内容的命令：**

cat redis.conf |grep -v "#" |grep -v "^$"

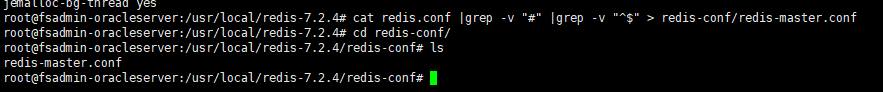
然后复制到 redis-conf下

cat redis.conf |grep -v "#" |grep -v "^$" > redis-conf/redis-master.conf

**整体命令如下：**

**其中：**grep -v "^$"是忽略掉空白行，grep -v "#" 忽略注释

cat redis.conf |grep -v "#" |grep -v "^$" > redis-conf/redis-master.conf



文件创建完成后，修改端口为6380 ，保护模式为no,允许远程当问，进程文件也改为6380，数据文件名称也要改动

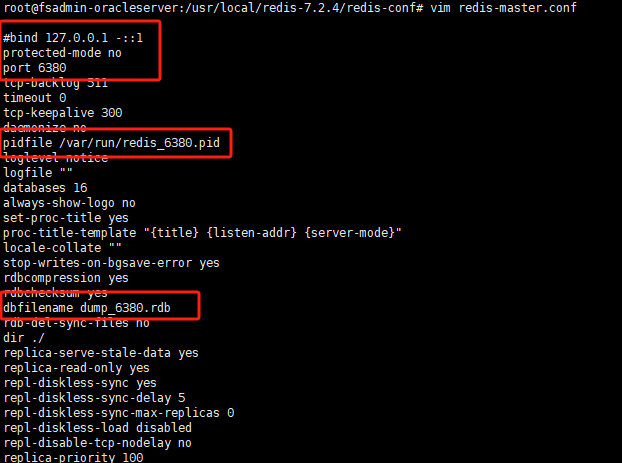
protected-mode no

port 6380

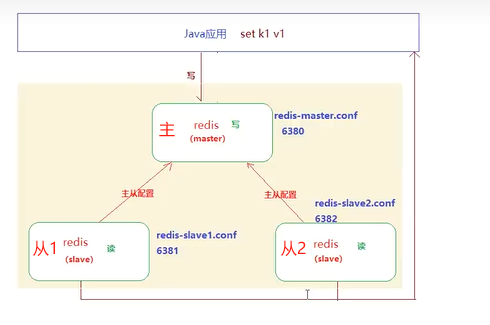
pidfile /var/run/redis\_6380.pid

dbfilename dump\_6380.rdb

appendfilename "appendonly\_6380.aof"



### 8.1.3建立两个从服务配置文件



1. 从redis-master.conf 拷贝一个文件redis-slave1.conf，作为从服务1的配置文件

cp redis-master.conf redis-slave1.conf



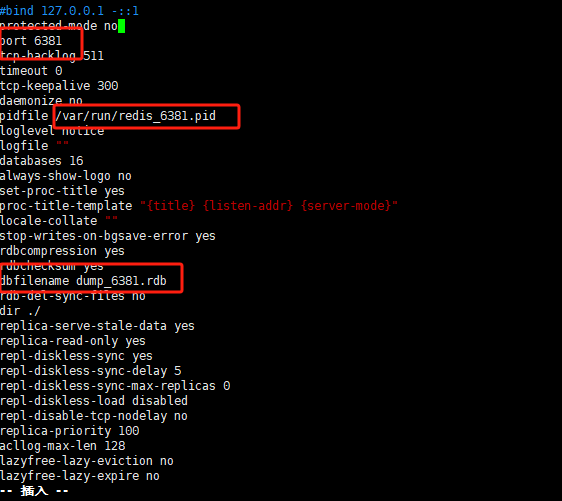
2.修改redis-slave1.conf的端口为6381和数据文件

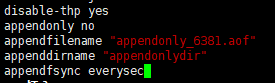
port 6381

pidfile /var/run/redis\_6381.pid

dbfilename dump\_6381.rdb

appendfilename "appendonly\_6381.aof"





3. 从redis-master.conf 拷贝一个文件redis-slave2.conf，作为从服务2的配置文件

cp redis-master.conf redis-slave2.conf

4. 修改redis-slave2.conf的端口为6382和数据文件

port 6382

pidfile /var/run/redis\_6382.pid

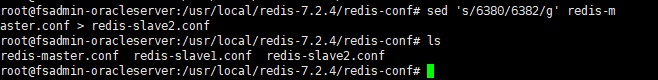
dbfilename dump\_6382.rdb

appendfilename "appendonly\_6382.aof"

**也可以用sed命令进行字符替换，输出另外一个文件，sed具体如下：**

sed 's/6380/6382/g' redis-master.conf > redis-slave2.conf

意思是 替换redis-master.conf 文件中6380 为6382 ，输出到redis-slave2.conf



## 设置主从关系配置

在从服务的配置文件加 slaveof 配置主从结构（绑定父节点为 6380 端口）.，让从服务跟随主服务；

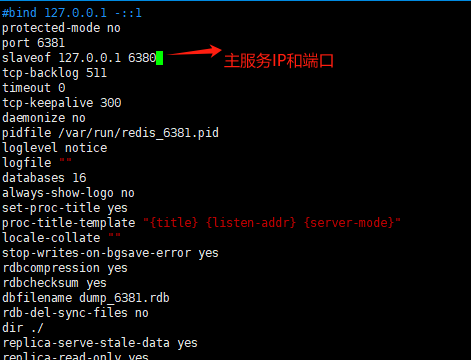
先给从服务1配置文件redis-slave1.conf添加

vim redis-slave1.conf

在配置文件任何一行添加如下代码：

slaveof 127.0.0.1 6380

注意：127.0.01 为redis主服务IP地址；6380为主服务端口



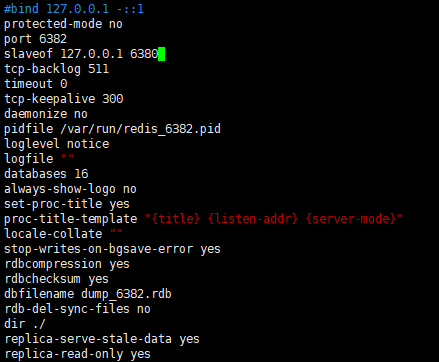
保存退出

同时给服务2配置文件redis-slave2.conf添加

vim redis-slave1.conf

在配置文件任何一行添加如下代码：

slaveof 127.0.0.1 6380

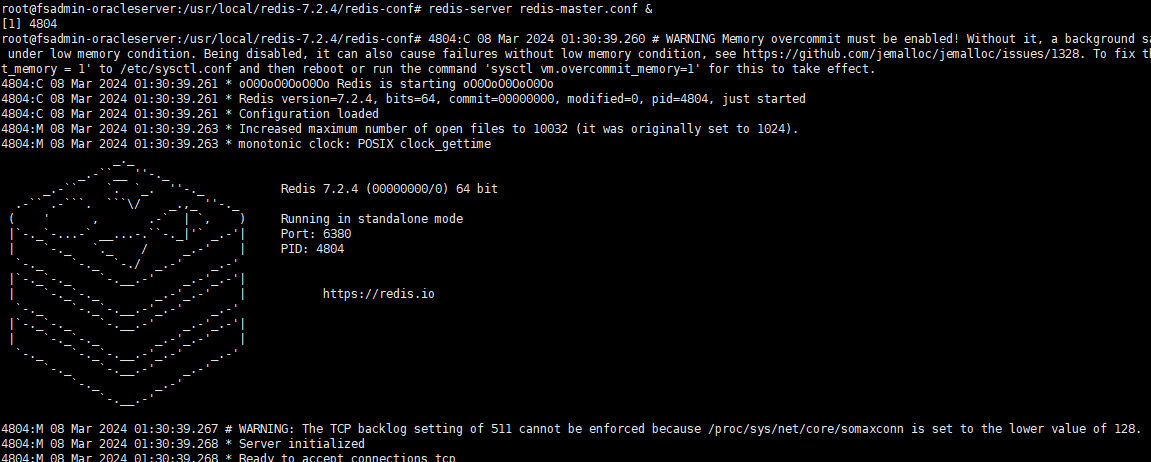


## 启动redis实例

在redis-conf目录下，依次启动redis主实例和2个从实例；

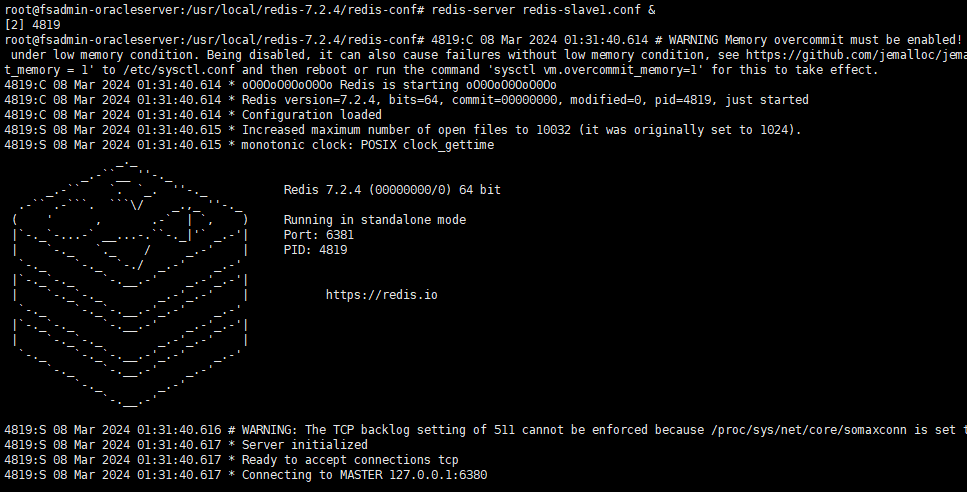
**启动主服务**

redis-server redis-master.conf &



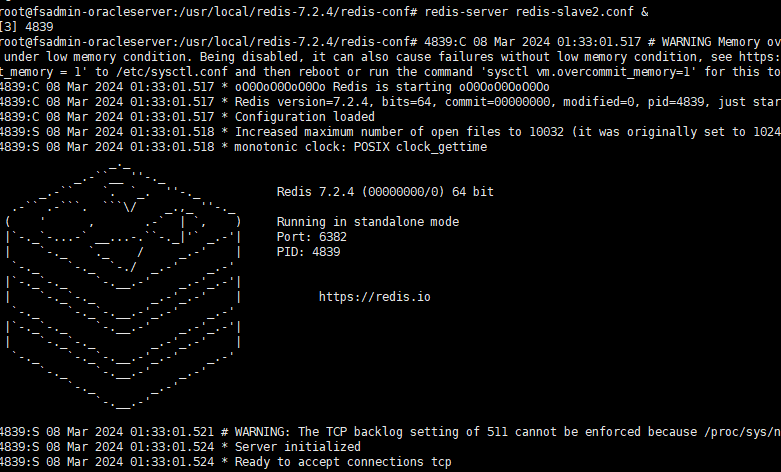
**启动服务1**

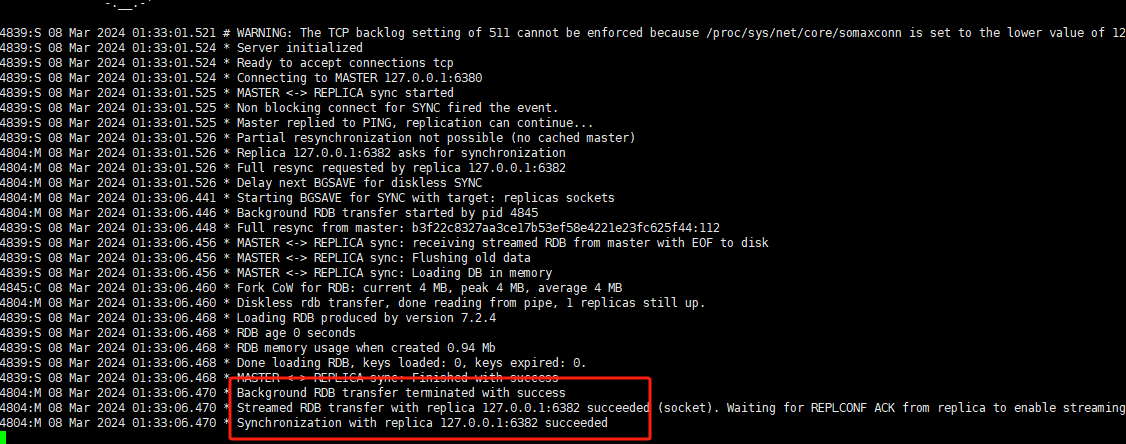
redis-server redis-slave1.conf &



**启动服务2**

redis-server redis-slave2.conf &



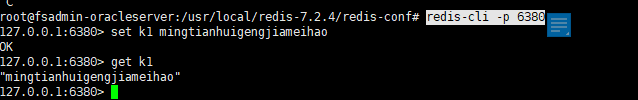


## 测试

连接主机客户端

redis-cli -p 6380

set k1 mingtianhuigengjiameihao



然后退出主服务，登录从服务，看是否能get k1值

redis-cli -p 6381

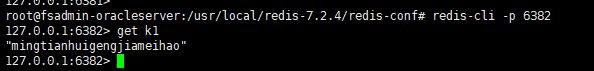
get k1



同样测试从服务2

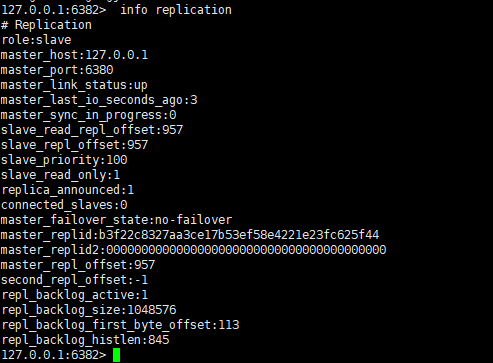
redis-cli -p 6382

get k1



在 redis 客户端，通过 info replication 就可以查看当前节点的复制状态了.

info replication



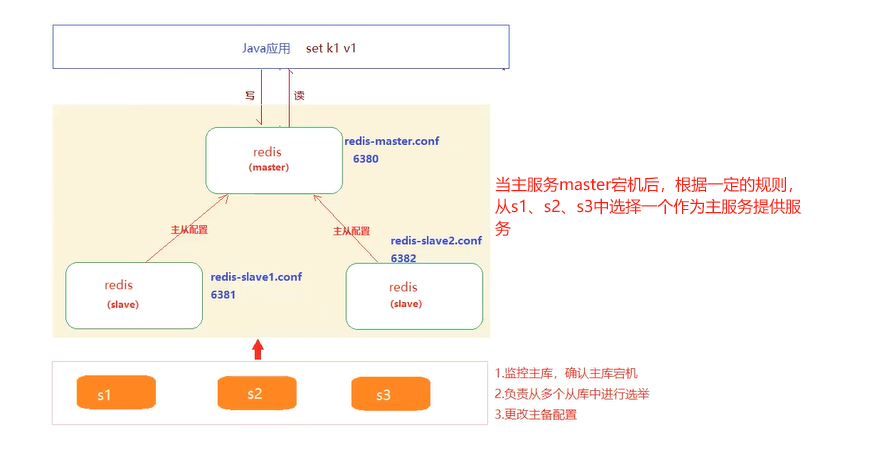
# Redis哨兵模式配置

主从复制模式，它是属于 Redis 多机运行的基础，但这种模式本身存在一个致命的问题，当主节点奔溃之后，需要人工干预才能恢复 Redis 的正常使用。

我们需要一个自动的工具——Redis Sentinel（哨兵模式）来把手动的过程变成自动的，让 Redis 拥有自动容灾恢复（failover）的能力。

哨兵就相当于对主从服务器做一个监视的任务。一旦发现主服务器宕机了，就迅速启动相应的规则将某一台从服务器升级为主服务器，无需人工干预，更稳定更快。

**哨兵模式需要依赖主从模式**



## 哨兵模式配置

配置哨兵模式的前提是实现了主从复制模式的配置，即一个主服务，2个从服务。

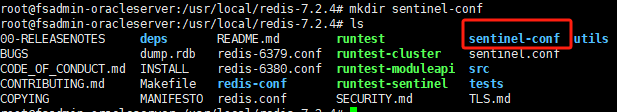
### 9.1.1创建哨兵配置文件

最少创建三个哨兵，注意哨兵的数量是奇数.

1. **创建哨兵配置文件目录**

**在redis 目录（/usr/local/redis-7.2.4）下，创建一个哨兵配置文件目录 sentinel-conf**

**mkdir sentinel-conf**



1. **创建第一个哨兵配置文件并修改配置**

在redis目录下，拷贝sentinel.conf到sentinel-conf目录下，命名为sentinel-26381.conf

cp sentinel.conf sentinel-conf/sentinel-26381.conf

**或者为了方便把配置文件中的注释和空行去掉后，生成**sentinel-26381.conf，命令如下：

cat sentinel.conf | grep -v "#" | grep -v "^$" > sentinel-conf/sentinel-26381.conf

修改配置文件sentinel-26381.conf

vim sentinel-26381.conf

修改内容如下：

#修改port

port 26381

#开启守护线程

daemonize yes

#进程文件

pidfile /var/run/redis-sentinel-26381.pid

#sentinel monitor <master-group-name> <ip> <port> <quorum>

#master-group-name是集群名称，可以自定义，

#ip：*master主机ip地址*

#quorum哨兵数量，是需要同意主节点不可用的Sentinel的数量，一般是哨兵数量减1

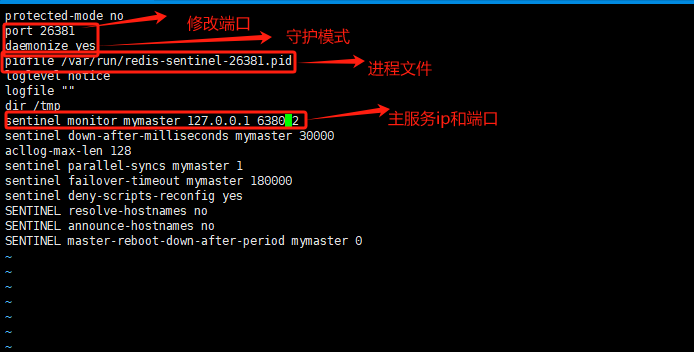
#比如2表示，当至少有2个哨兵发现master的redis挂了，

# 那么就将此master标记为宕机节点。

# 这个时候就会进行故障的转移，将其中的一个从节点变为master

sentinel monitor mymaster 127.0.0.1 6380 2

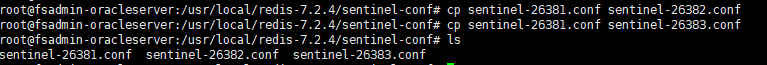
如下图：



1. **创建第二个和第三个哨兵配置文件并修改配置并修改端口**

**cp sentinel-26381.conf sentinel-26382.conf**

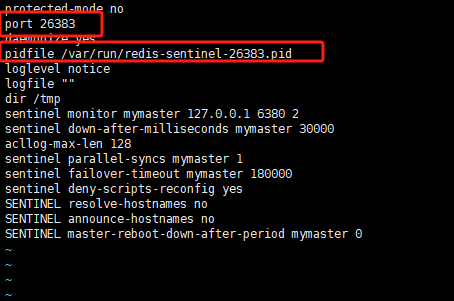
**cp sentinel-26381.conf sentinel-26383.conf**



**vim sentinel-26382.conf**



**vim sentinel-26383.conf**



### 9.1.2配置哨兵监听

分别修改vim sentinel-26381.conf、

vim sentinel-26382.conf、

vim sentinel-26383.conf

主要改如下四个配置：

# 第三个参数：哨兵名字，可自行修改。（若修改了，那后面涉及到的都得同步）

# 第四个参数：master主机ip地址

# 第五个参数：redis端口号

# 第六个参数：哨兵的数量。比如2表示，当至少有2个哨兵发现master的redis挂了，

# 那么就将此master标记为宕机节点。

# 这个时候就会进行故障的转移，将其中的一个从节点变为master

sentinel monitor mymaster 192.168.101.123 7001 2

# master中redis的密码

sentinel auth-pass mymaster 123456

# 哨兵从master节点宕机后，等待多少时间（毫秒），认定master不可用。

# 默认30s，这里为了测试，改成10s

sentinel down-after-milliseconds mymaster 10000

# 当替换主节点后，剩余从节点重新和新master做同步的并行数量，默认为 1

sentinel parallel-syncs mymaster 1

# 主备切换的时间，若在3分钟内没有切换成功，换另一个从节点切换

sentinel failover-timeout mymaster 180000

**修改如下：**

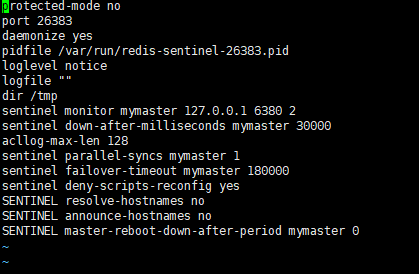
sentinel-26381.conf



sentinel-26382.conf

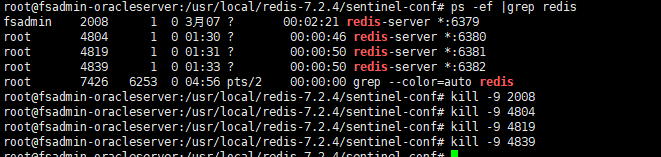


sentinel-26383.conf



## 哨兵模式测试

先关闭redis所有服务

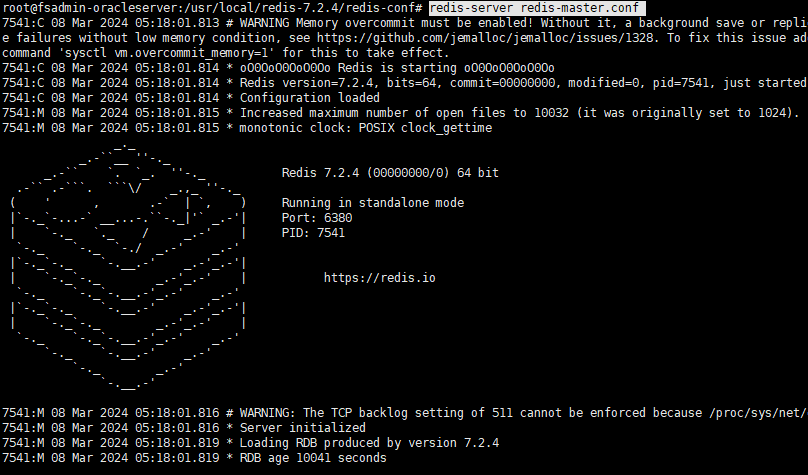


### 9.2.1启动三个主从服务

为了在一台主机上模拟启动三个服务，我们在shell开三个窗口，分别启动主从服务 6380、从服务6381、6382，不在后台启动

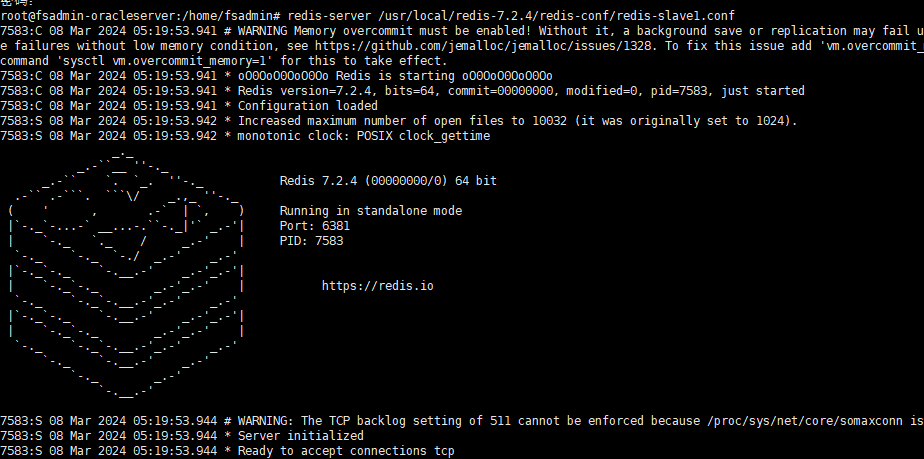
主服务 6380

redis-server redis-master.conf



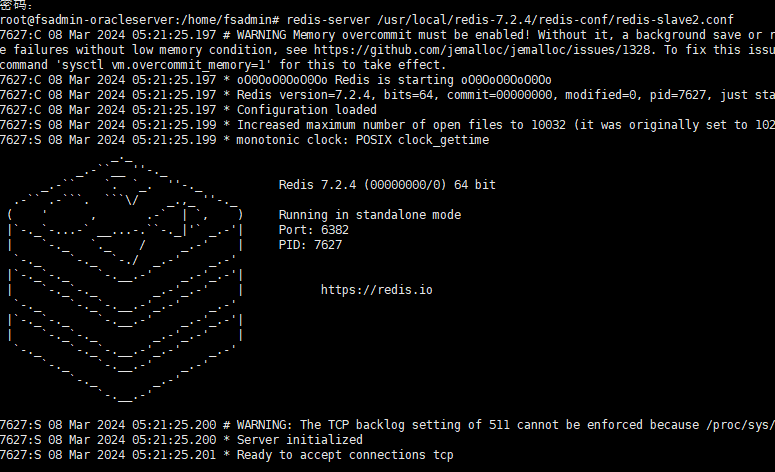
从服务1 6381

redis-server /usr/local/redis-7.2.4/redis-conf/redis-slave1.conf



从服务2 6382

redis-server /usr/local/redis-7.2.4/redis-conf/redis-slave2.conf



### 9.2.2启动哨兵服务

也是为了模拟测试，启动三个不同的shell

**启动哨兵一 26381**

redis-sentinel /usr/local/redis-7.2.4/sentinel-conf/sentinel-26381.conf

如果报如下错误

7790:X 08 Mar 2024 05:35:43.959 # WARNING Memory overcommit must be enabled! Without it, a background save or replication may fail under low memory condition. Being disabled, it can also cause failures without low memory condition, see https://github.com/jemalloc/jemalloc/issues/1328. To fix this issue add 'vm.overcommit\_memory = 1' to /etc/sysctl.conf and then reboot or run the command 'sysctl vm.overcommit\_memory=1' for this to take effect.



您的 Redis 容器日志中的警告信息指出了内存超分配（overcommit）的设置问题。在 Linux 系统中，内存超分配是一种内核行为，允许进程分配超出物理内存大小的虚拟内存。对于 Redis，这是一个重要的设置，尤其是当执行如 RDB 快照或 AOF 日志重写等内存密集型操作时。

解决办法：

**临时更改**（不会在重启后保持）：

在宿主机上执行以下命令：

sudo sysctl vm.overcommit\_memory=1



**永久更改**：

编辑 /etc/sysctl.conf 文件（或在某些系统上，可能是 /etc/sysctl.d/ 目录下的文件）。

添加或修改以下行：

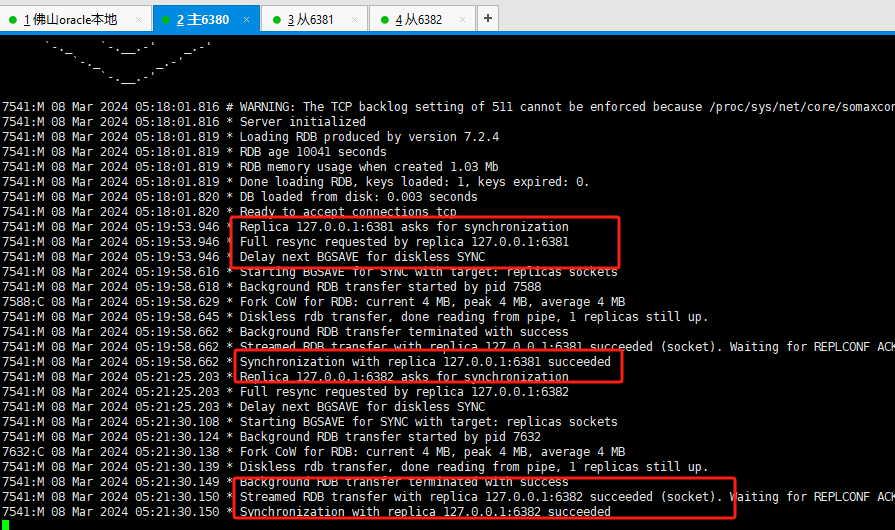
vm.overcommit\_memory = 1

保存文件并重新启动系统，或者运行 sudo sysctl -p 来立即应用更改。

启动后如下：

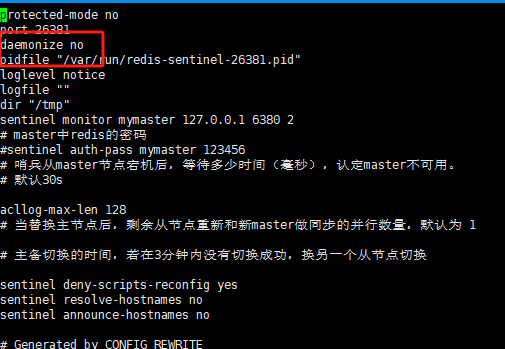


启动后，主服务6380也可以感知到



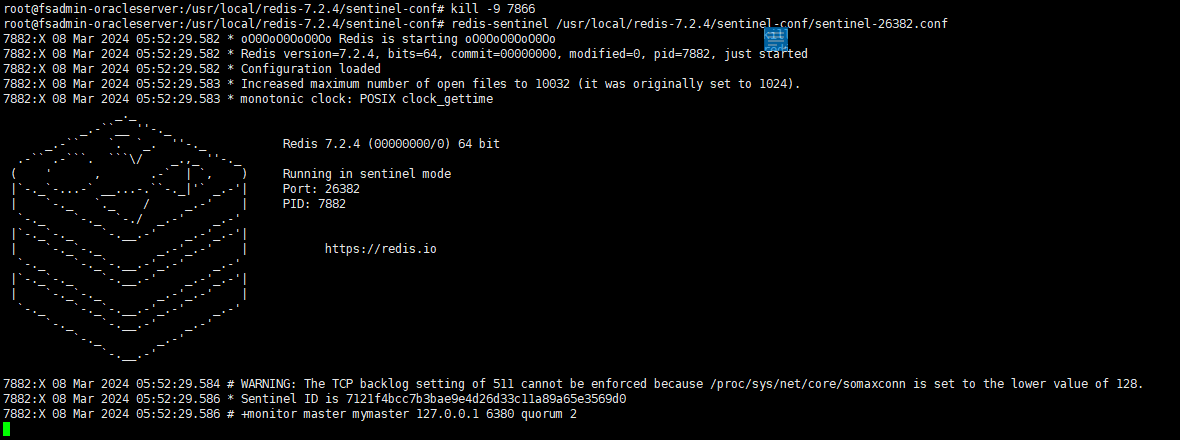
**启动哨兵二 26382**

为了测试，在配置文件中把守护模式改为 no

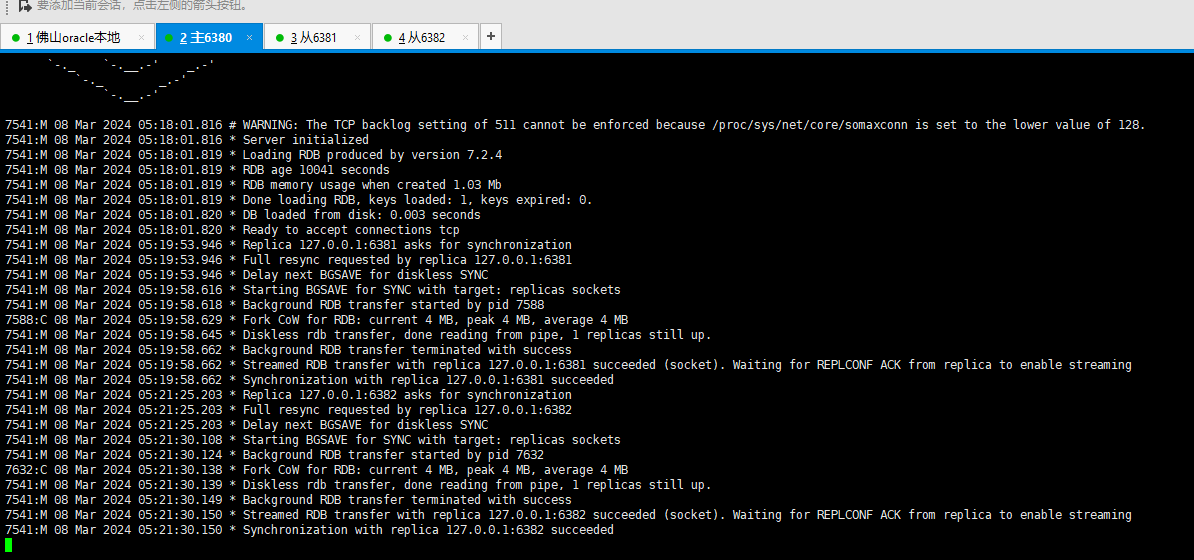


再启动

redis-sentinel /usr/local/redis-7.2.4/sentinel-conf/sentinel-26382.conf

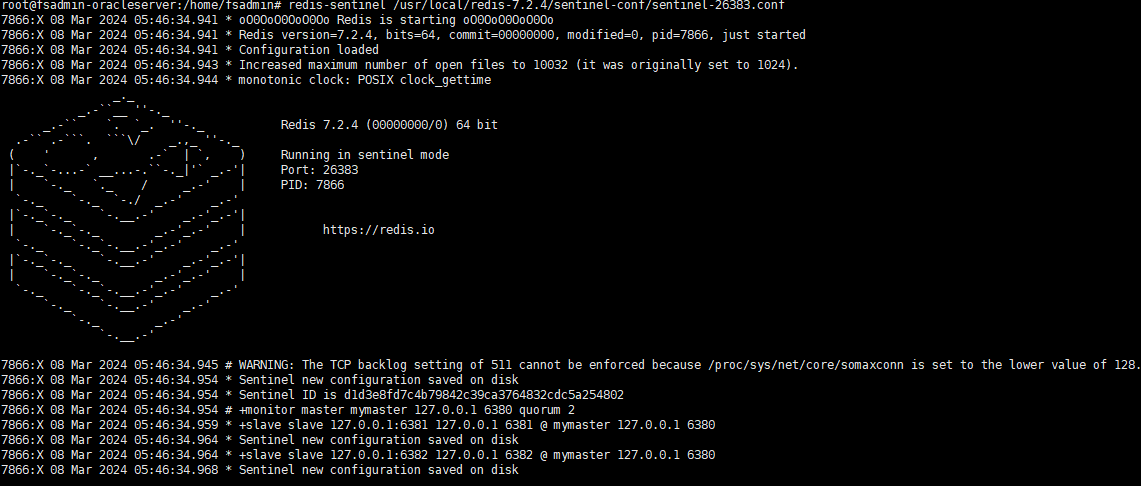


启动后后，主服务6380也有反应



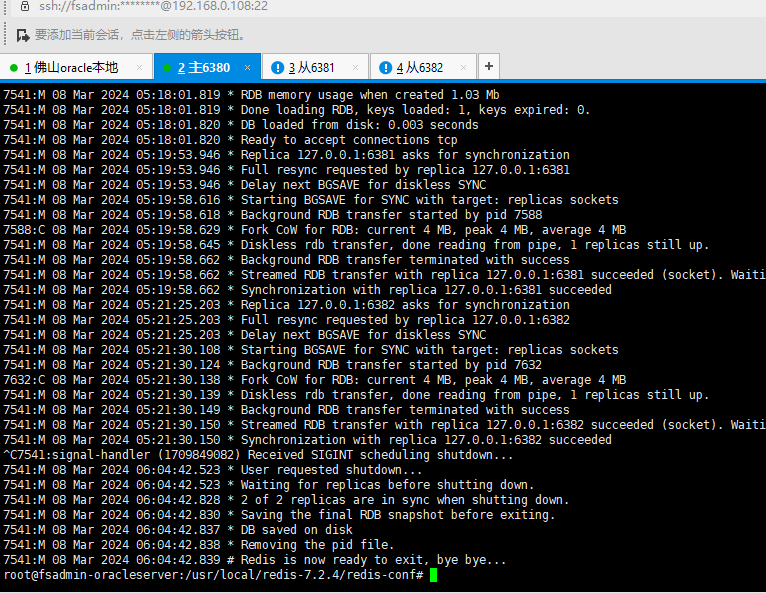
**启动哨兵三26383**

redis-sentinel /usr/local/redis-7.2.4/sentinel-conf/sentinel-26383.conf



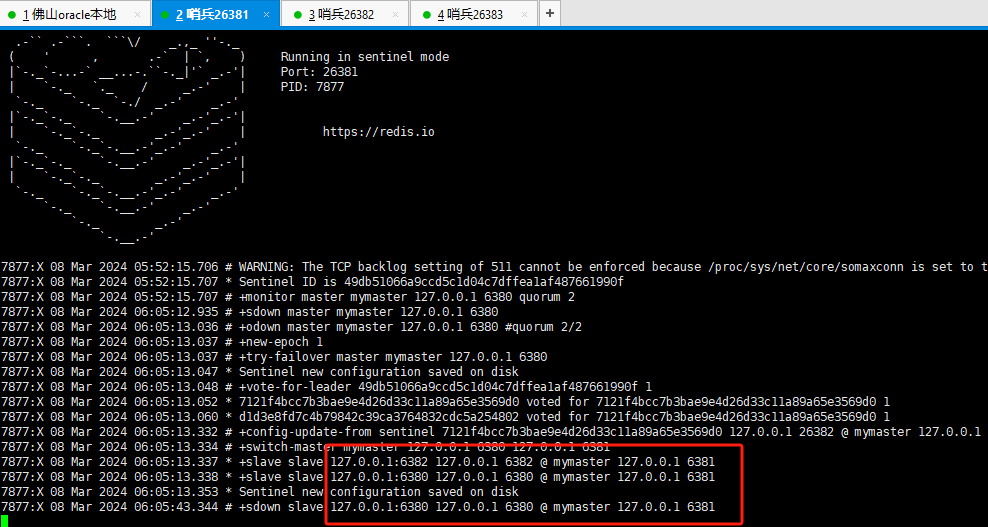
### 9.2.3 模拟测试

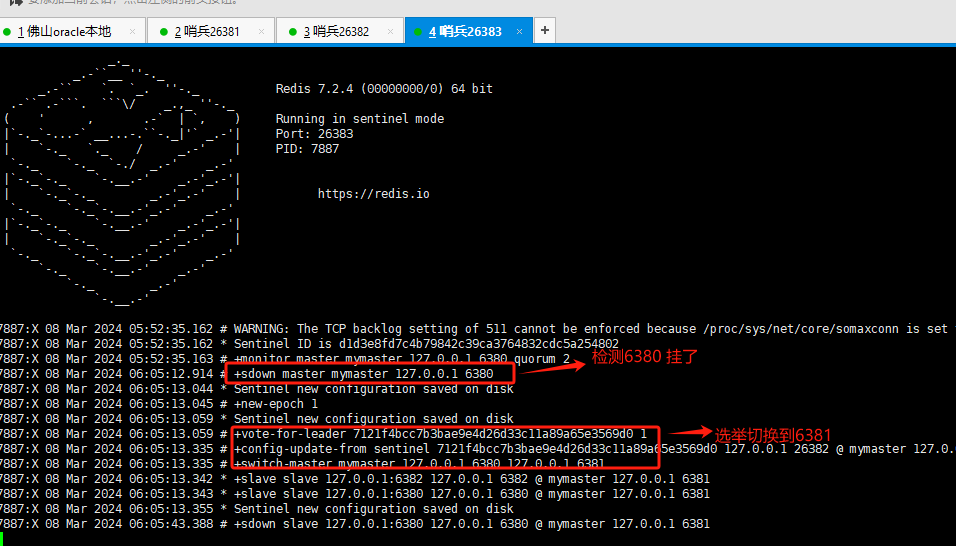
1.模拟强制关闭主服务redis-master 6380

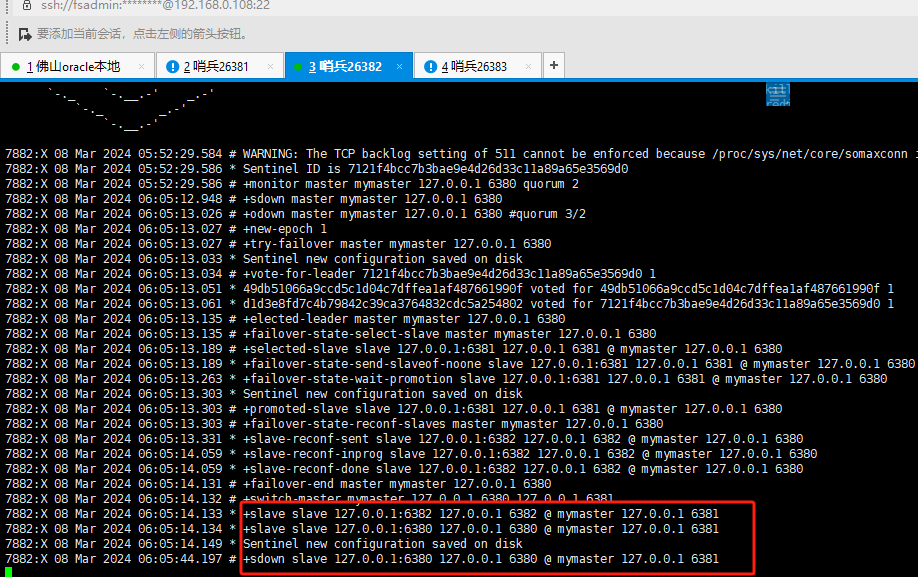


主服务redis-master 6380已经关闭

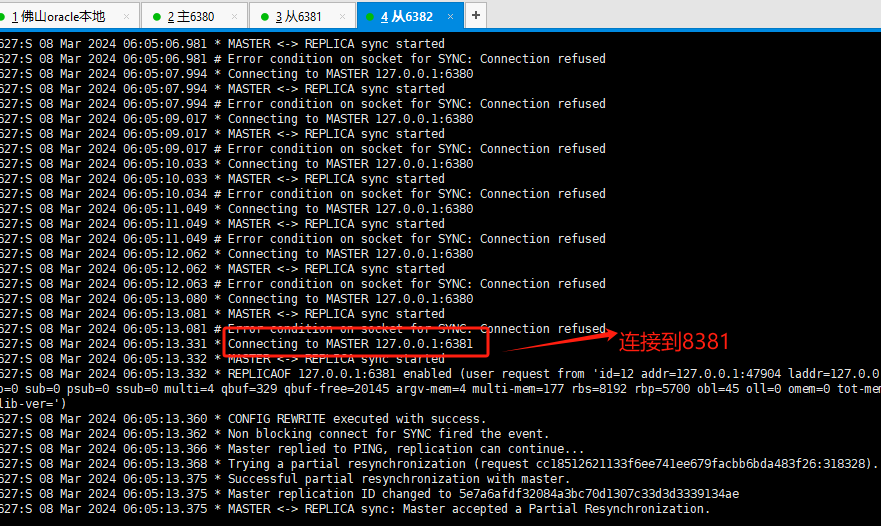
2.30秒后，查看哨兵启动窗口，发现主服务从6380 切换到6381



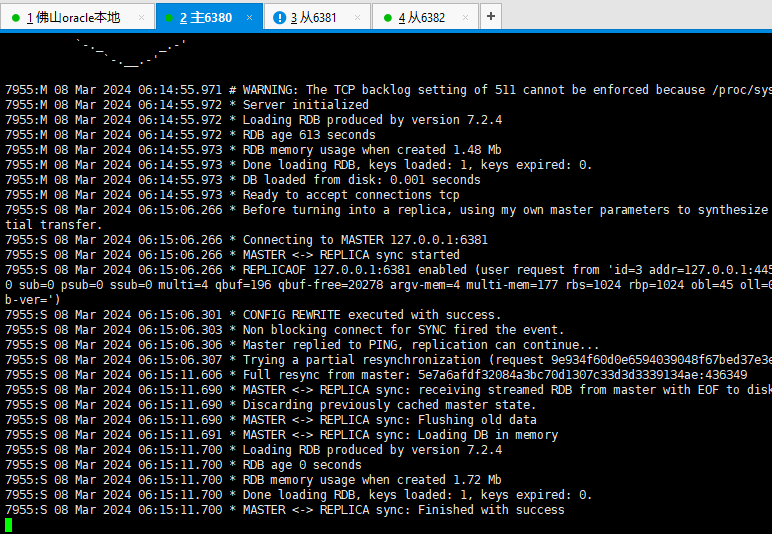




再看从服务，顺利连到6381

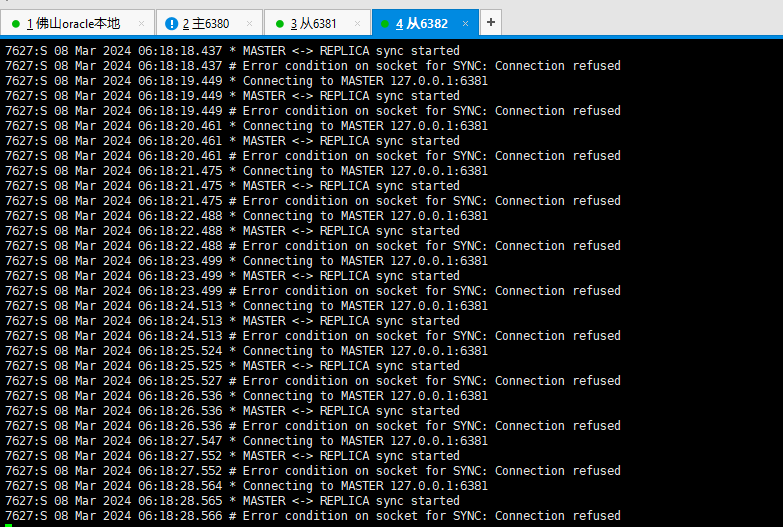


重启启动强制挂掉的6380

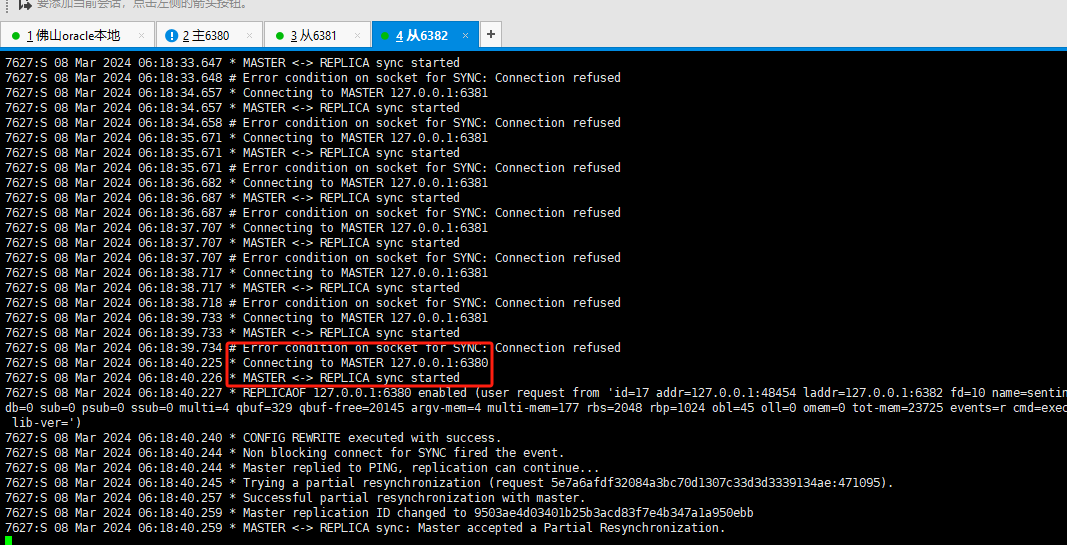


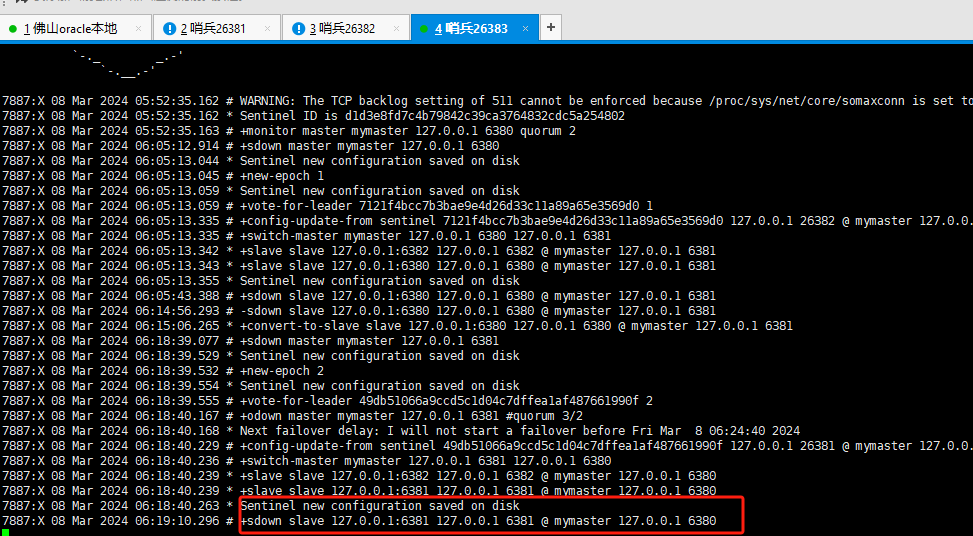
再看6381，已经检测到6380已经连接

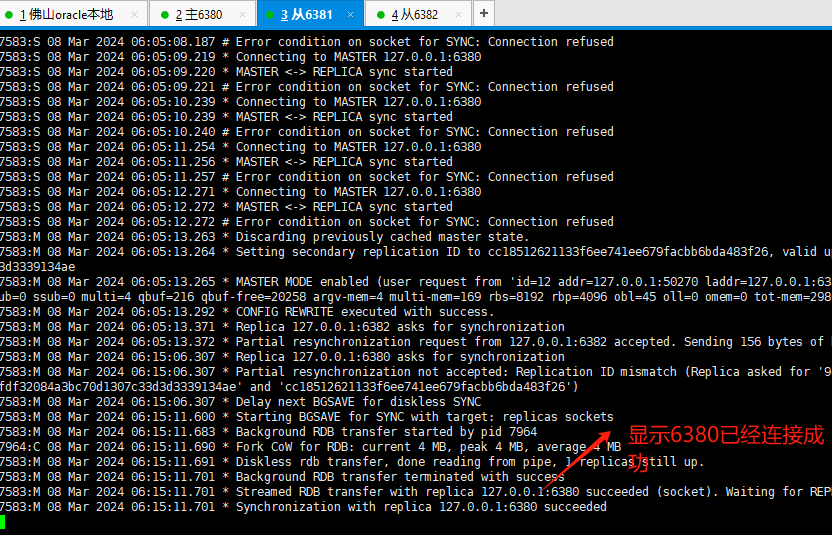
再重复一遍，如果挂了6381，则6382显示如下：



30秒后，又切换到6380







## 哨兵模式启动与关闭

### 9.3.1启动

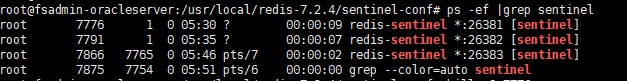
redis-sentinel /usr/local/redis-7.2.4/sentinel-conf/sentinel-26381.conf



### 9.3.1关闭

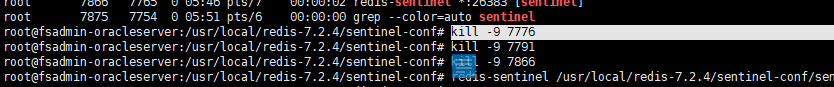
查找进程

ps -ef |grep sentinel



根据进程号kill

kill -9 7776



# Redis集群配置

## 集群介绍

### 10.1.1为什么需要集群

在之前的哨兵当中，**一旦主服务 master 挂掉了，就势必会导致一段时间内的数据写不进去了**，当并发量很大的时候这势必是一个比较大的问题。集群就说为了解决这个问题而诞生的。

### 10.1.2什么是集群

* 由于数据量过大，单个Master主从集难以承担，因此需要对多个主从集进行集群，形成水平扩展每个主从集只负责存储整个数据集的一部分，这就是Redis的集群，其作用是提供在多个Redis节点间共享数据的程序集。
* Redis集群是一个提供在多个Redis节点间共享数据的程序集
* Redis集群可以支持多个Master

### 10.1.3集群作用

* Redis集群支持多个Master，每个Master又可以挂载多个Slave
  + 读写分离
  + 支持海量数据的高可用
  + 支持海量数据的读写存储操作
* 由于Cluster自带Sentinel的故障转移机制，内置了高可用的支持，无需再去使用哨兵功能
* 客户端和Redis的节点连接，不再需要连接集群中所有节点，只需连接集群中的任意一个可用节点即可（一个有所有都有了）
* 槽位slot负责分配到各个物理服务节点，由对应的集群来负责维护节点、插槽和数据之间的关系

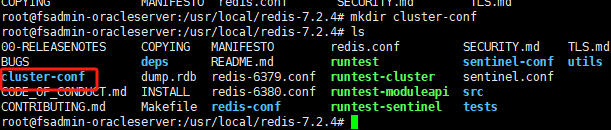
## 集群搭建

### 10.2.1创建集群配置目录

在redis安装目录下，创建文件夹cluster-conf,用于存放集群配置文件

cd /usr/local/redis-7.2.4

mkdir cluster-conf



### 10.2.2创建配置集群配置文件

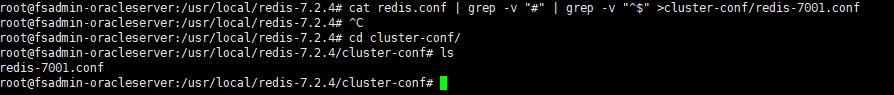
我们以redis.conf为原始文件，拷贝6个文件放在cluster-conf目录中（为了方便配置，最好cat命令取掉注释和空格空白行），作为集群的配置文件，在配置文件中修改端口和其他参数，我们定义6个集群节点端口分别为 7001、7002、7003、7004、7005、7006

**1、创建并配置第一个节点redis-7001.conf**

cp redis.conf cluster-conf/redis-7001.conf

或者

cat redis.conf | grep -v "#" | grep -v "^$" >cluster-conf/redis-7001.conf （建议）



修改配置redis-7001

cd cluster-conf/

vim redis-7001.conf

主要修改如下项

#注释，允许远程访问

#bind 127.0.0.1 -::1

#关闭保护模式

protected-mode no

#端口

port 7001

#后台运行

daemonize yes

dbfilename dump\_7001.rdb

appendfilename "appendonly\_7001.aof"

**并添加集群配置**

#开启集群

cluster-enabled yes

#集群的配置文件名称，不需要我们创建，redis自行维护

cluster-config-file nodes-7001.conf

#节点心跳失效的超时时间

cluster-node-timeout 15000

修改完成如下：



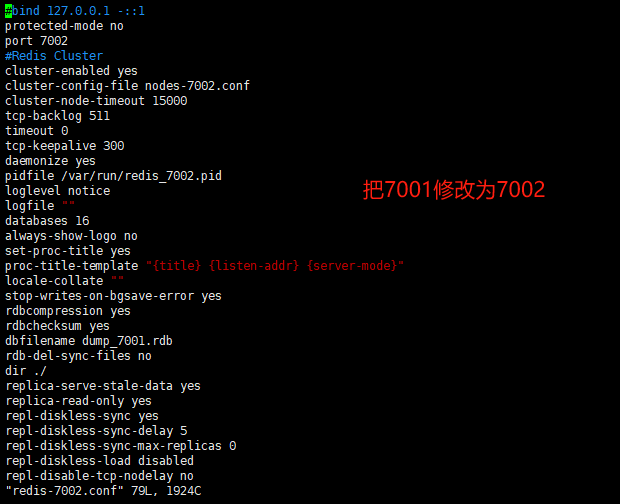
**2、创建并配置第二个节点redis-7002.conf**

拷贝redis-7001.conf ，修改端口就可以

cp redis-7001.conf redis-7002.conf



vim redis-7002.conf



**3、创建并配置第三个节点redis-7003.conf**

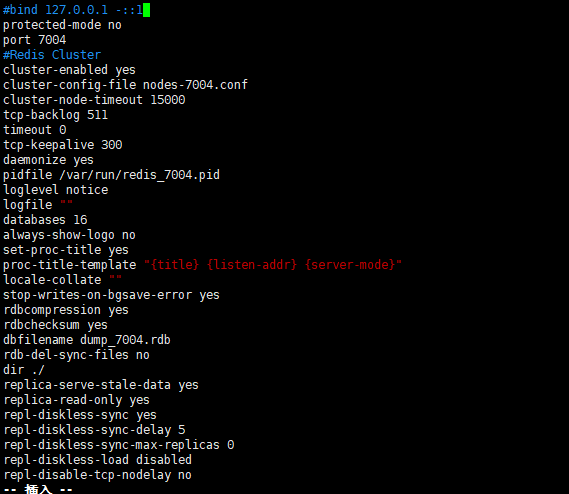
cp redis-7001.conf redis-7003.conf

修改端口

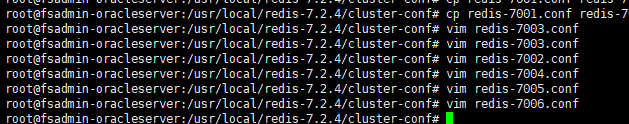
vim redis-7003.conf



**4、同理创建并配置第四，第五和第六个节点redis-7004.conf、redis-7005.conf、redis-7006.conf**



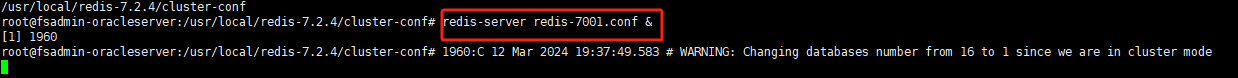
全部修改完毕



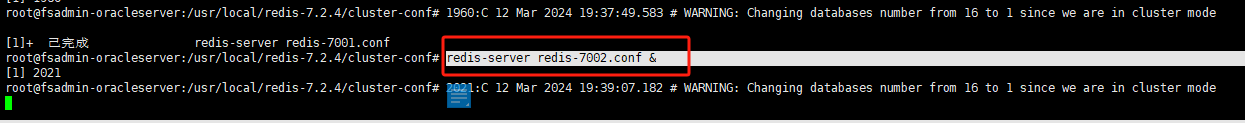
### 10.2.3启动创建的6个集群节点实例

在/usr/local/redis-7.2.4/cluster-conf下，分别启动6个服务

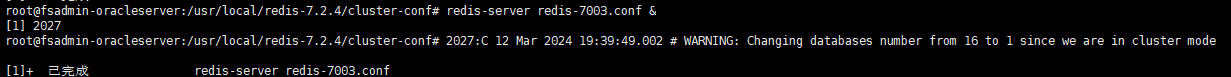
redis-server redis-7001.conf &



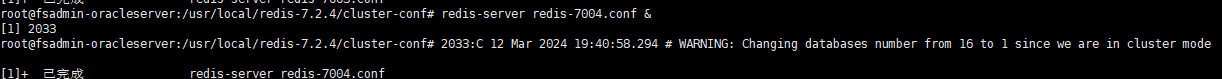
redis-server redis-7002.conf &



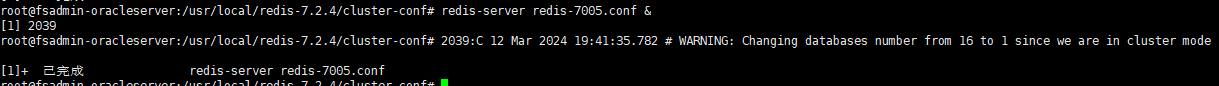
redis-server redis-7003.conf &



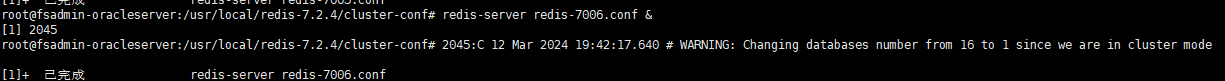
redis-server redis-7004.conf &



redis-server redis-7005.conf

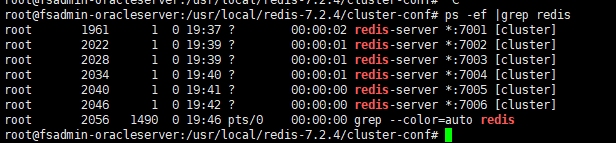


redis-server redis-7006.conf &



查看6个实例是否启动

ps -ef |grep redis



### 10.2.4创建并启动集群

现在只是开启了redis服务，并没有形成分片集群，要形成分片集群，就需要进行一下操作。

我们使用的redis是6.2.6版本的redis，对应redis5.0之后，集群管理已经集成到了redis-cli命令里面去了。

启动集群命令如下：

redis-cli --cluster create --cluster-replicas 1 192.168.0.108:7001 192.168.0.108:7002 192.168.0.108:7003 192.168.0.108:7004 192.168.0.108:7005 192.168.0.108:7006

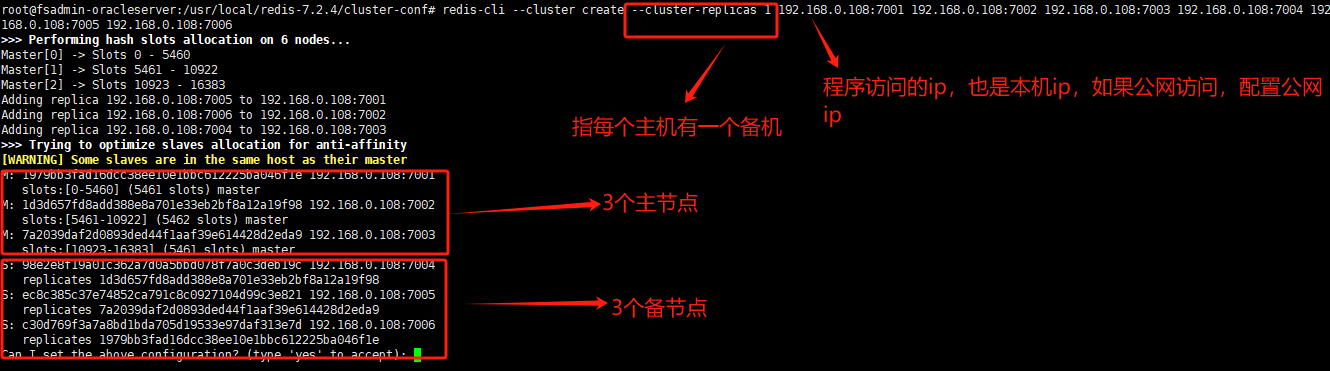
**--cluster：代表使用集群操作命令。**

**create：创建集群。**

**192.168.0.106：客户端或者程序访问的IP地址**

**--replicas 1：也就是设置replicas=1指一个主机提供一个备机**

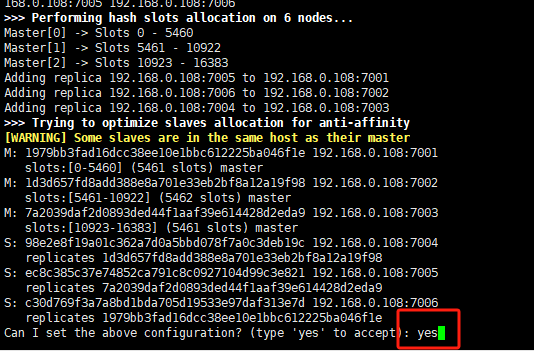
**提供的端口的节点总数的基础上/（replicas+1），此时有6的接口，除以2对应就是有3个主节点，每个对应一个备机。**



我们可以通过这个图，可以看到**slots**就是redis把主节点分配的插槽位置

这里输入yes

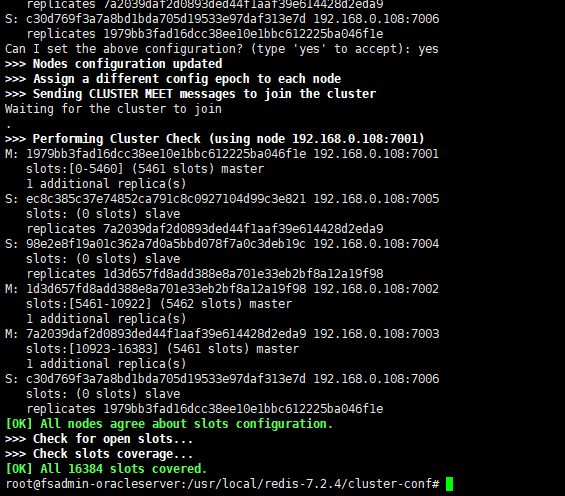
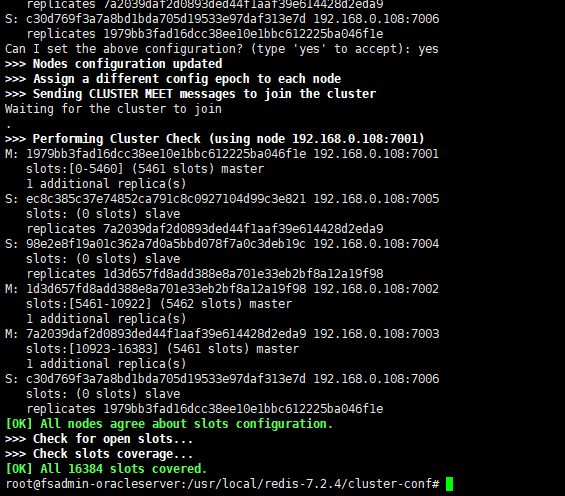
Can I set the above configuration? (type 'yes' to accept): yes



则会出现

[OK] All 16384 slots covered.

则说明集群组建完毕



## 集群测试

可以连接任何一个节点，输入如下命令，进入客户端

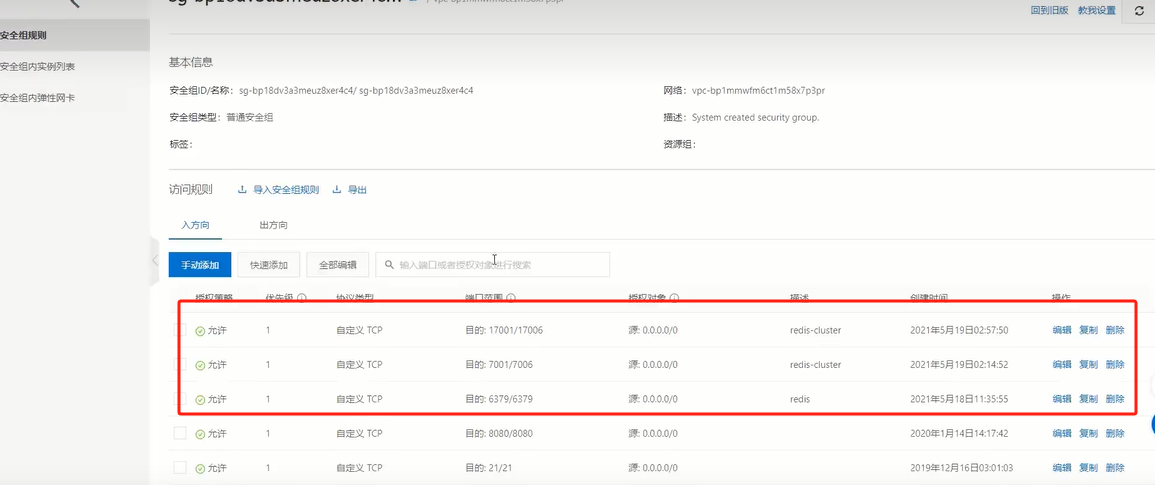
redis-cli -p 7001 –c



在7001节点set一个值，集群会存在一个节点上；当get获的时候，又从存的节点取出。



需要注意的是，在linux下，开放端口策略的时候，如果开通了7001、7002、7003、7004、7005、7006后还无法访问的话，需要开通17001到17006端口，即端口+10000

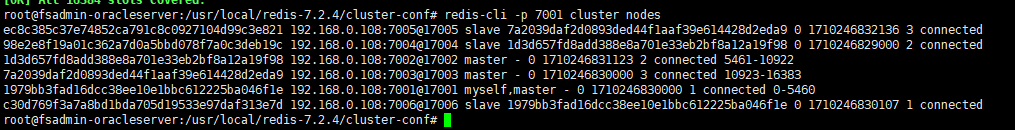


## 集群管理

### 10.4.1查看集群状态

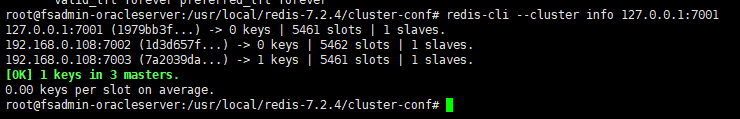
查看集群状态

redis-cli -p 7001 cluster nodes



或者

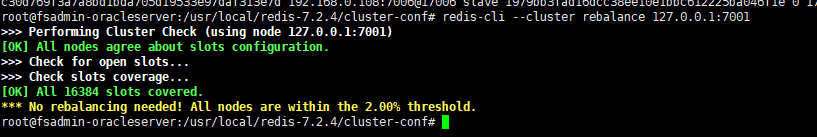
redis-cli --cluster info 127.0.0.1:7001



### 10.4.2平衡节点的数据槽数

当槽的节点不平衡时，需要平衡槽的数量

redis-cli --cluster rebalance 127.0.0.1:7001



### 10.4.3增加集群

首先在cluster-conf中增加集群配置文件，然后启动集群节点，通过如下命令加入到集群

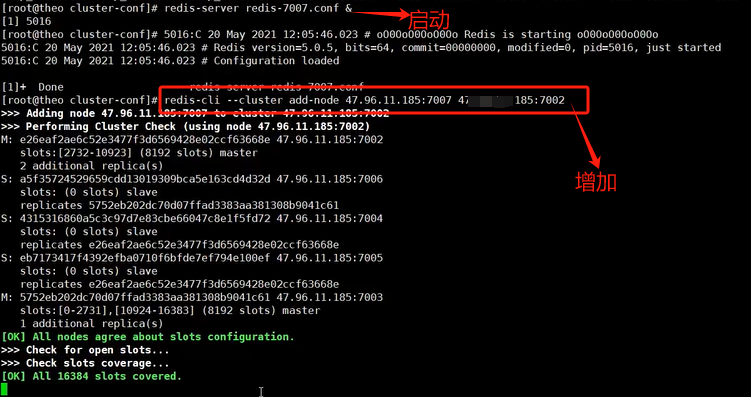
先要启动节点

redis-server redis-7007.conf &

再增加节点

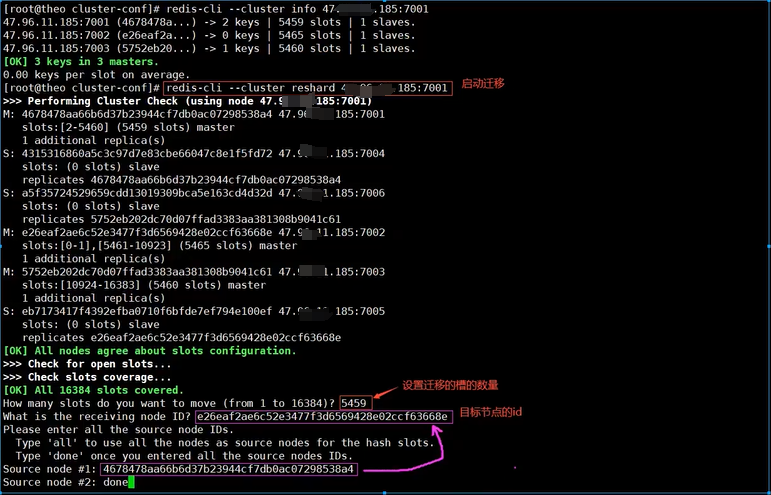
redis-cli --cluster add-node 192.168.0.108:7007 192.168.0.108:7001

添加7007节点到集群，7001表示存在的节点，任何一个存在的节点都可以。



### 10.4.4迁移节点槽

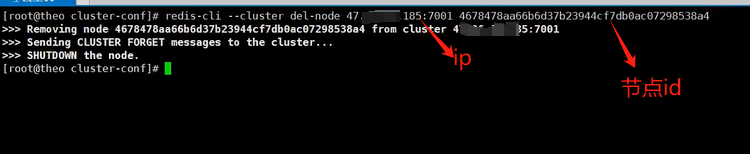
redis-cli --cluster reshard 192.168.0.108:7001



### 10.4.5删除节点

删除前需要迁移节点槽数据，有数据的节点不能删除

redis-cli --cluster del-node 192.168.0.108:7001 节点id



## Springboot 连接redis集群

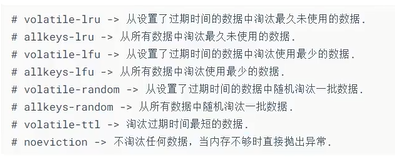
Springboot 连接redis集群和单机基本一致1，唯一不同的就是配置文件



# Redis淘汰策略

Redis是基于内存结构进行数据缓存的，当内存资源消耗完毕，想要有新的数据缓存进来，必然要从Redis的内存结构中释放一些数据。如何进行数据的释放呢?…Redis的淘汰策略

## Redis八种淘汰策略介绍



## Redis淘汰策略

**1.针对设置过期时间的key**

volatile-ttl：设置了过期时间的键值对中，根据过期时间的先后顺序进行淘汰，ttl值越小越先被淘汰。没有设置过过期时间的不淘汰。

volatile-lru：设置了过期时间的键值对中，最少使用的key被优先淘汰。没有设置过过期时间的不淘汰。

volatile-lfu(redis4.0版本新增)：设置了过期时间的键值对中，最近一段时间被访问次数最少的数据进行删除。

volatile-random：设置了过期时间的键值对中，根据随机数淘汰。

**2.针对所有key**

allkeys-lru：根据lru算法(最近最少使用)进行淘汰。淘汰的key的范围是全部的key的集合，没有设置过期时间的也会被淘汰。

allkeys-lfu：使用 LFU 算法进行淘汰，也就是淘汰最近一段时间被访问次数最少的数据。

allkeys-random：在虽有key中随机选择key进行淘汰。

**3.不处理**

noeviction：默认策略，拒绝所有的写入操作并返回客户端错误信息error 。del命令和读请求可以继续进行。不会删除数据。

## Redis淘汰策略配置

在redis.config 配置文件中，选择那种策略就ok

