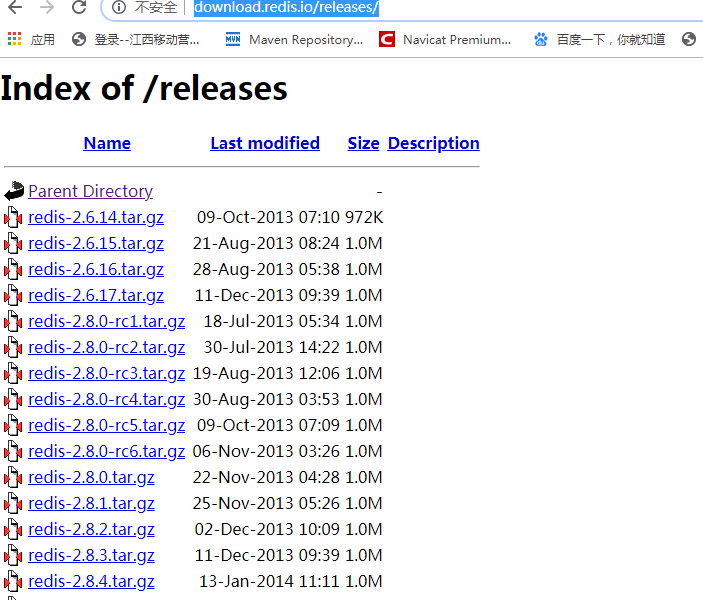
Redis

下载及安装：

下载所需版本：<http://download.redis.io/releases/>



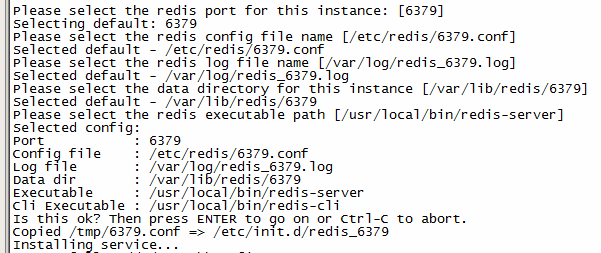
 在官网下载redis-4.0.9.tar.gz，用tar -xvf redis-4.0.9.tar.gz解压

安装Redis需要将下载的源码进行编译，编译依赖gc++,如果没有gc++环境需要安装gcc，**执行命令：**yum install gcc-c++

编译安装：make

make （PREFIX=/usr/local/redis） install 该步骤已经安装成功

./utils/install\_server.sh（初始化，生成配置文件和启动文件）



默认的配置文件：

端口：6379

配置文件路径：/etc/redis/6379.conf

日志文件：/var/log/redis\_6379.log

数据保存文件:/var/lib/redis/6379

Redis启动：redis-server /etc/redis/6379.conf（配置文件路径）

或者 /etc/init.d/redis\_6379

redis启动关闭：/etc/init.d/redis\_6379 start/stop/restart（若更改了端口或者密码，需要在配置文件改回来）

或者 redis-server /etc/redis/redis.6379.conf （配置文件）

redis-cli -h IP地址 -p 6379 shutdown

保存后，启动和开启服务  
systemctl daemon-reload 重新加载systemctl  
systemctl enable redis.service 设置开机启动  
systemctl start redis.service 启动redis服务

如果上述方式都没有成功停止redis，则可以使用终极武器 kill -9

修改配置文件：

Vim /etc/redis/redis.conf

bind参数:即允许指定ip访问redis服务器,默认本机才能访问(127.0.0.1),在此改成如下(允许任意ip访问)

Bind 0.0.0.0

### daemonize参数:守护线程，用来指定redis是否要用守护线程的方式启动。默认是no

daemonize:yes  
代表开启守护进程模式。redis采用的是单进程多线程的模式。在该模式下，redis会在后台运行，并将进程pid号写入至redis.conf选项pidfile设置的文件中，此时redis将一直运行，除非手动kill该进程。

daemonize:no  
当前界面将进入redis的命令行界面，exit强制退出或者关闭连接工具(putty,xshell等)都会导致redis进程退出。

Requirepass password，这里我已经将密码设置成“123456”，注意“#”是注释符号。

修改保护模式

还是在redis.conf文件中，查找“protected-mode”，把“yes”改为“no”

Port：端口号，默认6379

redis.conf 配置项说明如下：

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

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

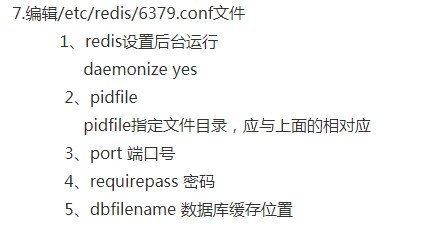
hash-max-zipmap-value 512

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

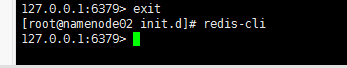
30. 指定包含其它的配置文件，可以在同一主机上多个Redis实例之间使用同一份配置文件，而同时各个实例又拥有自己的特定配置文件

include /path/to/local.conf



登录redis：redis\_cli

redis\_cli -h 主机IP -p 端口 -a 密码



Php-redis扩展

Redis命令使用：  
config get \* （获取所有配置信息，\*号可输入特定的的配置项，如config get loglevel）

Keys \* 查询所有

keys home 精确查找

keys h\* 模糊查找

DEL key

该命令用于在 key 存在时删除 key。

DUMP key

序列化给定 key ，并返回被序列化的值。

EXISTS key

检查给定 key 是否存在。

EXPIRE key seconds

为给定 key 设置过期时间。

EXPIREAT key timestamp

EXPIREAT 的作用和 EXPIRE 类似，都用于为 key 设置过期时间。 不同在于 EXPIREAT 命令接受的时间参数是 UNIX 时间戳(unix timestamp)。

PEXPIRE key milliseconds

设置 key 的过期时间以毫秒计。

PEXPIREAT key milliseconds-timestamp

设置 key 过期时间的时间戳(unix timestamp) 以毫秒计

KEYS pattern

查找所有符合给定模式( pattern)的 key 。

MOVE key db

将当前数据库的 key 移动到给定的数据库 db 当中。

PERSIST key

移除 key 的过期时间，key 将持久保持。

PTTL key

以毫秒为单位返回 key 的剩余的过期时间。

TTL key

以秒为单位，返回给定 key 的剩余生存时间(TTL, time to live)。

RANDOMKEY

从当前数据库中随机返回一个 key 。

RENAME key newkey

修改 key 的名称

RENAMENX key newkey

仅当 newkey 不存在时，将 key 改名为 newkey 。

TYPE key

返回 key 所储存的值的类型。

## **Redis字符串命令**

|  |  |
| --- | --- |
| 1 | [SET key value](http://www.runoob.com/redis/strings-set.html) 设置指定 key 的值 |
| 2 | [GET key](http://www.runoob.com/redis/strings-get.html) 获取指定 key 的值。 |
| 3 | [GETRANGE key start end](http://www.runoob.com/redis/strings-getrange.html) 返回 key 中字符串值的子字符 |
| 4 | [GETSET key value](http://www.runoob.com/redis/strings-getset.html) 将给定 key 的值设为 value ，并返回 key 的旧值(old value)。 |
| 5 | [GETBIT key offset](http://www.runoob.com/redis/strings-getbit.html) 对 key 所储存的字符串值，获取指定偏移量上的位(bit)。 |
| 6 | [MGET key1 [key2..]](http://www.runoob.com/redis/strings-mget.html) 获取所有(一个或多个)给定 key 的值。 |
| 7 | [SETBIT key offset value](http://www.runoob.com/redis/strings-setbit.html) 对 key 所储存的字符串值，设置或清除指定偏移量上的位(bit)。 |
| 8 | [SETEX key seconds value](http://www.runoob.com/redis/strings-setex.html) 将值 value 关联到 key ，并将 key 的过期时间设为 seconds (以秒为单位)。 |
| 9 | [SETNX key value](http://www.runoob.com/redis/strings-setnx.html) 只有在 key 不存在时设置 key 的值。 |
| 10 | [SETRANGE key offset value](http://www.runoob.com/redis/strings-setrange.html) 用 value 参数覆写给定 key 所储存的字符串值，从偏移量 offset 开始。 |
| 11 | [STRLEN key](http://www.runoob.com/redis/strings-strlen.html) 返回 key 所储存的字符串值的长度。 |
| 12 | [MSET key value [key value ...]](http://www.runoob.com/redis/strings-mset.html) 同时设置一个或多个 key-value 对。 |
| 13 | [MSETNX key value [key value ...]](http://www.runoob.com/redis/strings-msetnx.html) 同时设置一个或多个 key-value 对，当且仅当所有给定 key 都不存在。 |
| 14 | [PSETEX key milliseconds value](http://www.runoob.com/redis/strings-psetex.html) 这个命令和 SETEX 命令相似，但它以毫秒为单位设置 key 的生存时间，而不是像 SETEX 命令那样，以秒为单位。 |
| 15 | [INCR key](http://www.runoob.com/redis/strings-incr.html) 将 key 中储存的数字值增一。 |
| 16 | [INCRBY key increment](http://www.runoob.com/redis/strings-incrby.html) 将 key 所储存的值加上给定的增量值（increment） 。 |
| 17 | [INCRBYFLOAT key increment](http://www.runoob.com/redis/strings-incrbyfloat.html) 将 key 所储存的值加上给定的浮点增量值（increment） 。 |
| 18 | [DECR key](http://www.runoob.com/redis/strings-decr.html) 将 key 中储存的数字值减一。 |
| 19 | [DECRBY key decrement](http://www.runoob.com/redis/strings-decrby.html) key 所储存的值减去给定的减量值（decrement） 。 |
| 20 | [APPEND key value](http://www.runoob.com/redis/strings-append.html) 如果 key 已经存在并且是一个字符串， APPEND 命令将 value 追加到 key 原来的值的末尾。 |

config get requirepass //获取当前密码

config set requirepass "yourpassword"//设置当前临时密码,服务重新启动后又会置为默认，即无密码；不建议此种方式

select 1 //选择数据库，配置中会设置数据库数量

info //查看info信息，包括占用内存，及一些配置信息

info memory //查看占用内存情况

dbsize //查看数据库有多少个key

keys \*banner\* //模糊搜索带banner的key值，因为redis会给key加上前缀，所以在不知道前缀的情况下，先模糊搜索一下

get keys //指定key获取value值

说明：  
redis集群至少需要3个主节点，每个主节点有一个从节点总共6个节点  
replicas指定为1表示每个主节点有一个从节点

3.1  ruby环境

redis集群管理工具redis-trib.rb依赖ruby环境，首先需要安装ruby环境：

安装ruby

yum install ruby

yum install rubygems

安装ruby和redis的接口程序

拷贝redis-3.0.0.gem至/usr/local下

执行：

gem install /usr/local/redis-3.0.0.gem