**Redis**

# Redis 简介



## Redis是简介

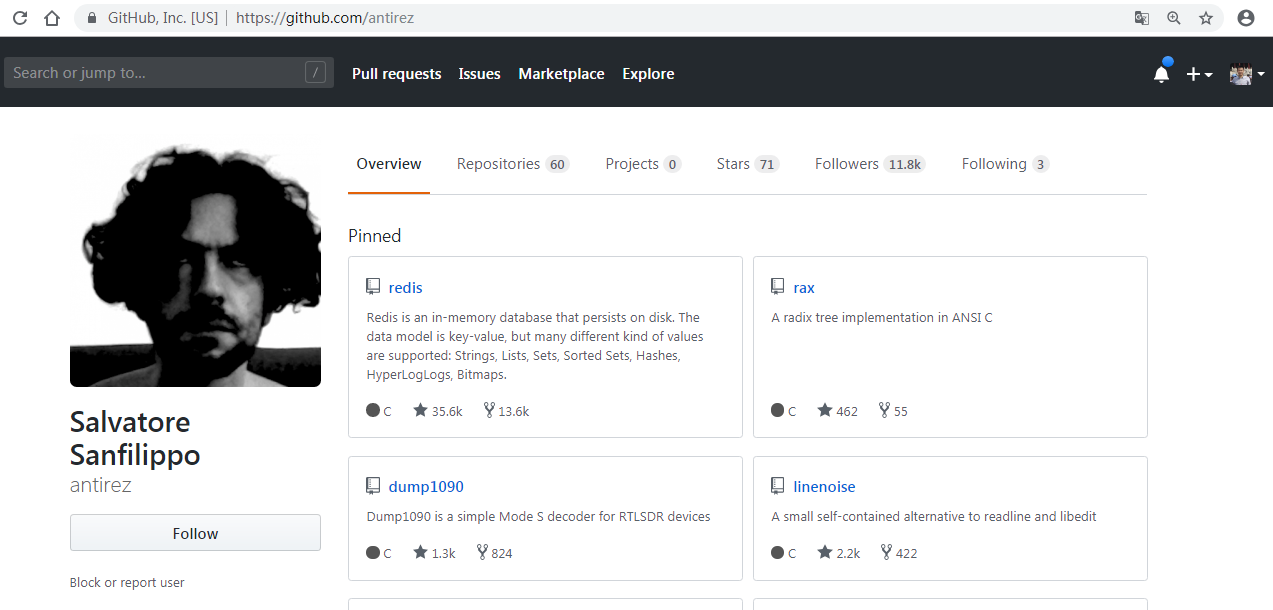
Redis 是完全开源免费的，遵守BSD协议，是一个高性能(NOSQL)的key-value数据库,Redis是一个开源的使用ANSI [C语言](https://baike.baidu.com/item/C%E8%AF%AD%E8%A8%80)编写、支持网络、可基于内存亦可持久化的日志型、Key-Value[数据库](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93)，并提供多种语言的API。从2010年3月15日起，Redis的开发工作由VMware主持。从2013年5月开始，Redis的开发由Pivotal赞助。(Vmware在资助着redis项目的开发和维护)

[BSD](https://baike.baidu.com/item/BSD)是"Berkeley Software Distribution"的缩写，[意思](https://baike.baidu.com/item/%E6%84%8F%E6%80%9D" \t "_blank)是"伯克利软件发行版"。

BSD开源协议是一个给于使用者很大自由的协议。可以自由的使用，修改源代码，也可以将修改后的代码作为开源或者专有软件再发布。BSD代码鼓励代码共享，但需要尊重代码作者的著作权。BSD由于允许使用者修改和重新发布代码，也允许使用或在BSD代码上开发商业软件发布和销售，因此是对商业集成很友好的协议。

## Redis作者

redis的作者，他叫Salvatore Sanfilippo，来自意大利的西西里岛，现在居住在卡塔尼亚。目前供职于Pivotal公司。



地址是antirez.com，当然也可以去follow他的github，地址是<http://github.com/antirez>。

## NoSQL介绍

NoSQL，泛指非关系型的数据库。随着互联网[web2.0](https://baike.baidu.com/item/web2.0)网站的兴起，传统的关系数据库在应付web2.0网站，特别是超大规模和高并发的[SNS](https://baike.baidu.com/item/SNS/10242)类型的web2.0纯[动态网](https://baike.baidu.com/item/%E5%8A%A8%E6%80%81%E7%BD%91)站已经显得力不从心，暴露了很多难以克服的问题，而非关系型的数据库则由于其本身的特点得到了非常迅速的发展。**NoSQL数据库的产生就是为了解决大规模数据集合多重数据种类带来的挑战，尤其是大数据应用难题**。

### NoSQL数据库的四大分类

**键值(**[*Key-Value*](https://baike.baidu.com/item/Key-Value)**)存储[数据库](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93)**  
 这一类数据库主要会使用到一个[哈希表](https://baike.baidu.com/item/%E5%93%88%E5%B8%8C%E8%A1%A8)，这个表中有一个特定的键和一个指针指向特定的数据。Key/value模型对于IT系统来说的优势在于简单、易部署。但是如果[DBA](https://baike.baidu.com/item/DBA/3349)只对部分值进行查询或更新的时候，Key/value就显得效率低下了。[3]  举例如：Tokyo Cabinet/Tyrant, **Redis**, Voldemort, Oracle BDB.  
  
**列存储数据库**  
 这部分数据库通常是用来应对分布式存储的海量数据。键仍然存在，但是它们的特点是指向了多个列。这些列是由列家族来安排的。如：Cassandra, **HBase**, Riak.  
  
**文档型数据库**  
 文档型数据库的灵感是来自于Lotus Notes办公软件的，而且它同第一种键值存储相类似。该类型的数据模型是版本化的文档，半结构化的文档以特定的格式存储，比如JSON。文档型数据库可 以看作是键值数据库的升级版，允许之间嵌套键值。而且文档型数据库比键值数据库的查询效率更高。如：CouchDB**, MongoDb**. 国内也有文档型数据库SequoiaDB，已经开源。  
  
**图形(Graph)数据库**  
 图形结构的数据库同其他行列以及刚性结构的SQL数据库不同，它是使用灵活的图形模型，并且能够扩展到多个服务器上。NoSQL数据库没有标准的查询语言(SQL)，因此进行数据库查询需要制定数据模型。许多NoSQL数据库都有REST式的数据接口或者查询API。[2]  如：Neo4J, InfoGrid, Infinite Graph.  
  
 因此，我们总结NoSQL数据库在以下的这几种情况下比较适用：1、数据模型比较简单；2、需要灵活性更强的IT系统；3、对数据库性能要求较高；4、不需要高度的数据一致性；5、对于给定key，比较容易映射复杂值的环境。

**Redis 与其他 key - value 缓存产品有以下三个特点：**

* Redis支持数据的持久化，可以将内存中的数据保存在磁盘中，重启的时候可以再次加载进行使用。
* Redis不仅仅支持简单的key-value类型的数据，同时还提供list，set，zset，hash等数据结构的存储。
* Redis支持数据的备份，集群等高可用功能。

## 特点:

* 性能极高 – Redis能读的速度是110000次/s,写的速度是81000次/s 。
* 丰富的数据类型 – Redis支持的类型 String, List, Hash, Set 及 Ordered Set 数据类型操作。
* 原子 – Redis的所有操作都是原子性的，意思就是要么成功执行要么失败完全不执行。单个操作是原子性的。多个操作也支持事务，即原子性，通过MULTI和EXEC指令包起来。
* 丰富的特性 – Redis还支持 publish/subscribe, 通知, key 过期等等特性。

**Redis是一个简单的，高效的，分布式的，基于内存的缓存工具。   
架设好服务器后，通过网络连接（类似数据库），提供Key－Value式缓存服务。**

**简单，是Redis突出的特色。   
简单可以保证核心功能的稳定和优异。**

## Redis总结

redis单个key 存入512M大小

redis支持多种类型的数据结构(string,list,hash.set.zset)

redis 是单线程 原子性

redis可以持久化 因为使用了 RDB和AOF机制

redis支持集群 而且redis 支持库(0-15) 16个库

redis 还可以做消息队列 比如聊天室 IM

企业级开发中:可以用作数据库、缓存(热点数据（经常会被查询，但是不经常被修改或者删除的数据)和消息中间件等大部分功能。

**优点：**   
1. 丰富的数据结构

2.高速读写，redis使用自己实现的分离器，代码量很短，没有使用lock（MySQL），因此效率非常高。

**缺点：**   
1. 持久化。Redis直接将数据存储到内存中，要将数据保存到磁盘上，Redis可以使用两种方式实现持久化过程。定时快照（snapshot）：每隔一段时间将整个数据库写到磁盘上，每次均是写全部数据，代价非常高。第二种方式基于语句追加（aof）：只追踪变化的数据，但是追加的log可能过大，同时所有的操作均重新执行一遍，回复速度慢。   
2. 耗内存，占用内存过高。

# Redis安装

## Windows安装

<https://jingyan.baidu.com/article/0f5fb099045b056d8334ea97.html>

## Linux安装

**安装Redis**

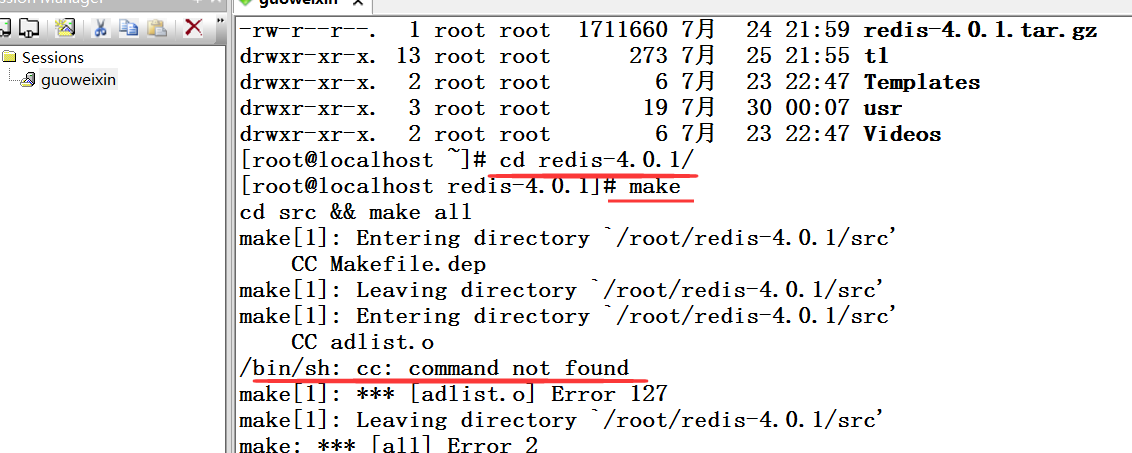
官方网站：http://redis.io/

官方下载：http://redis.io/download 可以根据需要下载不同版本

（域名后缀io属于国家域名，是british Indian Ocean territory，即英属印度洋领地）

**Redis安装**

Redis是C语言开发，安装Redis需要先将官网下载的源码进行编译，编译依赖gcc环境，如果没有gcc环境，需要安装gcc



### 安装gcc

gcc的安装很简单，首先要确保root登录，其次就是Linux要能连外网

|  |
| --- |
| yum -y install gcc automake autoconf libtool make |

**注意：**运行yum时出现/var/run/yum.pid已被锁定,PID为xxxx的另一个程序正在运行的问题解决

|  |
| --- |
| rm -f /var/run/yum.pid |

### 安装Redis

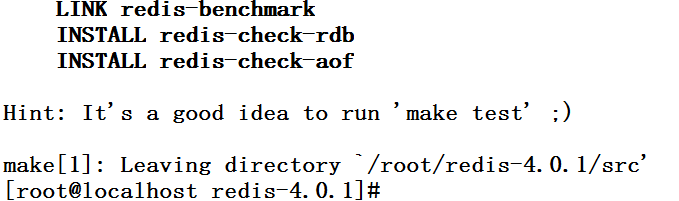
注意按照下面的步骤执行即可。

命令1： wget http://download.redis.io/releases/redis-4.0.1.tar.gz

命令2：**tar zxvf redis-4.0.1.tar.gz**

命令3： **cd redis-4.0.1**

命令4（编译）： make 或 如下图代表成功：



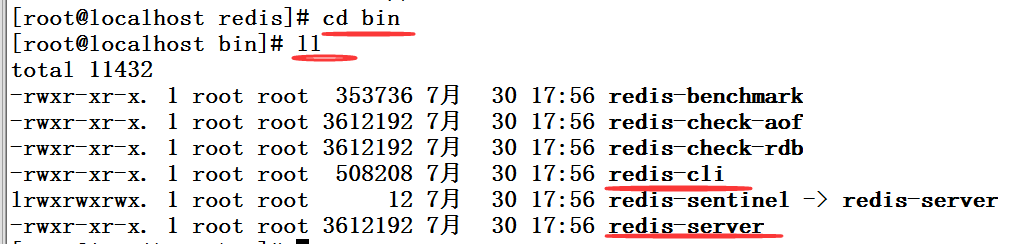
命令5：**make PREFIX=/usr/local/redis install**

（安装编译后的文件） 安装到指目录：

**注意：**PREFIX必须大写、同时会自动为我们创建redis目录，并将结果安装此目录

命令6： **cd /usr/local/redis** 查看

命令7：查看bin目录下，如图：

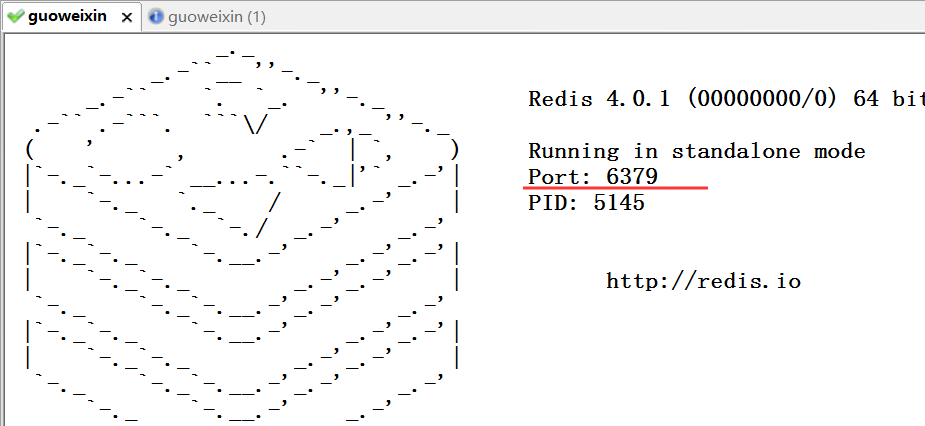


# Redis启动

## 启动Redis服务

进入对应的安装目录 **/usr/local/redis**

执行命令： **./bin/redis-server**

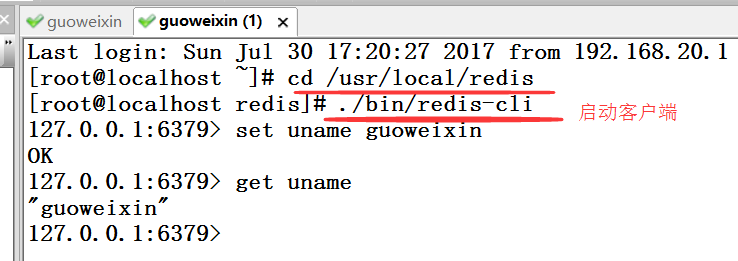
****

## 启动Redis客户端

**进入Redis客服端**（Clone Session克隆一个窗口）：

进入对应的安装目录 **cd /usr/local/redis**

执行命令： **./bin/redis-cli**

****

**启动Redis 客户端命令：**

**redis-cli –h IP地址 –p 端口**

**退出客户端命令：Ctrl+C**

检测是否服务端启动

启动 redis 客户端，打开终端并输入命令 **redis-cli**。该命令会连接本地的 redis 服务。

$redis-cli

redis 127.0.0.1:6379>

redis 127.0.0.1:6379> PING

PONG

在以上实例中我们连接到本地的 redis 服务并执行 **PING** 命令，该命令用于检测 redis 服务是否启动。

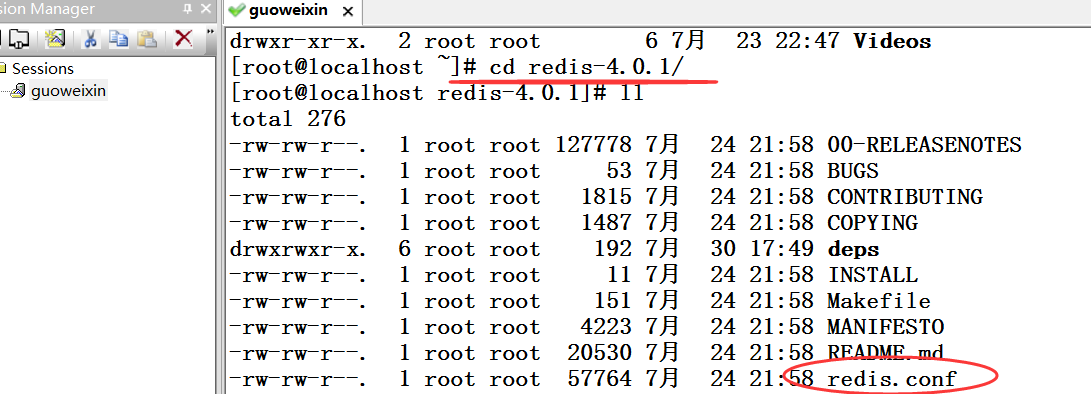
# Linux配置Redis

Redis 的配置文件位于 Redis 安装目录下，文件名为 **redis.conf**(Windows 名为 redis.windows.conf)。

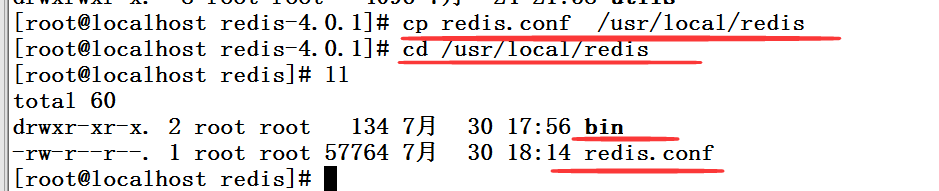
## 配置Redis

Redis端口号或启动有默认配置。但一般我们都会通过手动配置完成

回到根目录找到**解压文件中**的reids.conf



命令：**cp redis.conf /usr/local/redis** 将配置文件复制到安装文件的目录下



**Redis配置至此全部完成**

## redis.conf 配置文件详解

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，**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。持久化到硬盘上面去。

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（折中，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中的内存维护策略**

redis作为优秀的中间缓存件，时常会存储大量的数据，即使采取了集群部署来动态扩容，也应该即时的整理内存，维持系统性能。

**在redis中有两种解决方案**，

**一是为数据设置超时时间**，到期之后就会自动消除。

**二是采用LRU算法动态将不用的数据删除**。内存管理的一种页面置换算法，对于在内存中但又不用的数据块（内存块）叫做LRU，操作系统会根据哪些数据属于LRU而将其移出内存而腾出空间来加载另外的数据。

1.volatile-lru：设定超时时间的数据中,删除最不常使用的数据.

**2.allkeys-lru：查询所有的key中最近最不常使用的数据进行删除，这是应用最广泛的策略.**

3.volatile-random：在已经设定了超时的数据中随机删除.

**4.allkeys-random：查询所有的key,之后随机删除**.

5.volatile-ttl：查询全部设定超时时间的数据,之后排序,将马上将要过期的数据进行删除操作.

6.noeviction：如果设置为该属性,则不会进行删除操作,如果内存溢出则报错返回.

* volatile-lfu：从所有配置了过期时间的键中驱逐使用频率最少的键
* allkeys-lfu：从所有键中驱逐使用频率最少的键

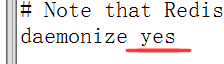
<https://www.jianshu.com/p/c8aeb3eee6bc>

## 自定义配置Redis

**1、进入对应的安装目录 /usr/local/redis**

修改 **redis.conf** 配置文件 **vim redis.conf** (进入命令模式 通过/内容 查找相应字符串）

**2、Redis配置默认必须修改：**

**daemonize no 修改为 daemonize yes** 

**bind 127.0.01 注释掉** 

**requirepass 设置密码**  

Redis采用的是单进程多线程的模式。当redis.conf中选项daemonize设置成yes时，代表开启守护进程模式。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。但当daemonize选项设置成no时，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。  
**服务端开发的大部分应用都是采用后台运行的模式**

# Redis启动：

## 服务端启动：

**./bin/redis-server ./redis.conf**

## 客户端启动：

### 本地客户端登录

**用redis-cli 密码登陆（redis-cli -a  password）**

### 远程服务上执行命令

如果需要在远程 redis 服务上执行命令，同样我们使用的也是 **redis-cli** 命令。

语法：在下面的前面需要加上： ./bin/

redis-cli -h host -p port -a password

**redis-cli –h IP地址 –p 端口 –a 密码**

# Redis关闭

**第一种关闭方式：（断电、非正常关闭。容易数据丢失）**

**查询PID ps -ef | grep -i redis //其中的i：表示搜索的时候忽略大小写。**

**kill -9 PID ：**<https://blog.csdn.net/a1010256340/article/details/75253353>

**第二种关闭方式（正常关闭、数据保存）**

**./bin/redis-cli shutdown**  关闭redis服务，通过客户端进行shutdown

如果redis设置了密码，需要先在客户端通过密码登录，再进行shutdown即可关闭服务端



**通过 ps –ef | grep –i redis 查看当前进程：**

