# Redis 配置文件

# 当配置中需要配置内存大小时，可以使用 1k, 5GB, 4M 等类似的格式，其转换方式如下(不区分大小写)  
#  
# 1k => 1000 bytes  
# 1kb => 1024 bytes  
# 1m => 1000000 bytes  
# 1mb => 1024\*1024 bytes  
# 1g => 1000000000 bytes  
# 1gb => 1024\*1024\*1024 bytes  
#  
# 内存配置大小写是一样的.比如 1gb 1Gb 1GB 1gB

# daemonize no 默认情况下，redis不是在后台运行的，如果需要在后台运行，把该项的值更改为yes  
daemonize yes

# 当redis在后台运行的时候，Redis默认会把pid文件放在/var/run/redis.pid，你可以配置到其他地址。  
# 当运行多个redis服务时，需要指定不同的pid文件和端口  
pidfile /var/run/redis.pid

# 指定redis运行的端口，默认是6379  
port 6379

# 指定redis只接收来自于该IP地址的请求，如果不进行设置，那么将处理所有请求，  
# 在生产环境中最好设置该项  
# bind 127.0.0.1

# Specify the path for the unix socket that will be used to listen for  
# incoming connections. There is no default, so Redis will not listen  
# on a unix socket when not specified.  
#  
# unixsocket /tmp/redis.sock  
# unixsocketperm 755

# 设置客户端连接时的超时时间，单位为秒。当客户端在这段时间内没有发出任何指令，那么关闭该连接  
# 0是关闭此设置  
timeout 0

# 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  
# debug 记录很多信息，用于开发和测试  
# varbose 有用的信息，不像debug会记录那么多  
# notice 普通的verbose，常用于生产环境  
# warning 只有非常重要或者严重的信息会记录到日志  
loglevel debug

# 配置log文件地址  
# 默认值为stdout，标准输出，若后台模式会输出到/dev/null  
#logfile stdout  
logfile /var/log/redis/redis.log

# To enable logging to the system logger, just set 'syslog-enabled' to yes,  
# and optionally update the other syslog parameters to suit your needs.  
# syslog-enabled no

# Specify the syslog identity.  
# syslog-ident redis

# Specify the syslog facility. Must be USER or between LOCAL0-LOCAL7.  
# syslog-facility local0

# 可用数据库数  
# 默认值为16，默认数据库为0，数据库范围在0-（database-1）之间  
databases 16

################################ 快照 #################################  
#  
# 保存数据到磁盘，格式如下:  
#  
# save <seconds> <changes>  
#  
# 指出在多长时间内，有多少次更新操作，就将数据同步到数据文件rdb。  
# 相当于条件触发抓取快照，这个可以多个条件配合  
#   
# 比如默认配置文件中的设置，就设置了三个条件  
#  
# save 900 1 900秒内至少有1个key被改变  
# save 300 10 300秒内至少有300个key被改变  
# save 60 10000 60秒内至少有10000个key被改变

save 900 1  
save 300 10  
save 60 10000

# 存储至本地数据库时（持久化到rdb文件）是否压缩数据，默认为yes  
rdbcompression yes

# 本地持久化数据库文件名，默认值为dump.rdb  
dbfilename dump.rdb

# 工作目录  
#  
# 数据库镜像备份的文件放置的路径。  
# 这里的路径跟文件名要分开配置是因为redis在进行备份时，先会将当前数据库的状态写入到一个临时文件中，等备份完成时，  
# 再把该该临时文件替换为上面所指定的文件，而这里的临时文件和上面所配置的备份文件都会放在这个指定的路径当中。  
#   
# AOF文件也会存放在这个目录下面  
#   
# 注意这里必须制定一个目录而不是文件  
dir ./

################################# 复制 #################################

# 主从复制. 设置该数据库为其他数据库的从数据库.   
# 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>

# 当master服务设置了密码保护时(用requirepass制定的密码)  
# slav服务连接master的密码  
#   
# masterauth <master-password>

# 当从库同主机失去连接或者复制正在进行，从机库有两种运行方式：  
#  
# 1) 如果slave-serve-stale-data设置为yes(默认设置)，从库会继续相应客户端的请求  
#   
# 2) 如果slave-serve-stale-data是指为no，出去INFO和SLAVOF命令之外的任何请求都会返回一个  
# 错误"SYNC with master in progress"  
#  
slave-serve-stale-data yes

# 从库会按照一个时间间隔向主库发送PINGs.可以通过repl-ping-slave-period设置这个时间间隔，默认是10秒  
#  
# repl-ping-slave-period 10

# repl-timeout 设置主库批量数据传输时间或者ping回复时间间隔，默认值是60秒  
# 一定要确保repl-timeout大于repl-ping-slave-period  
# repl-timeout 60

################################## 安全 ###################################

# 设置客户端连接后进行任何其他指定前需要使用的密码。  
# 警告：因为redis速度相当快，所以在一台比较好的服务器下，一个外部的用户可以在一秒钟进行150K次的密码尝试，这意味着你需要指定非常非常强大的密码来防止暴力破解  
#  
# requirepass foobared

# 命令重命名.  
#  
# 在一个共享环境下可以重命名相对危险的命令。比如把CONFIG重名为一个不容易猜测的字符。  
#  
# 举例:  
#  
# rename-command CONFIG b840fc02d524045429941cc15f59e41cb7be6c52  
#  
# 如果想删除一个命令，直接把它重命名为一个空字符""即可，如下：  
#  
# rename-command CONFIG ""

################################### 约束 ####################################

# 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

# 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  
# Redis同时也会移除空的list对象  
#  
# 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  
#   
# 注意：Redis新的vm机制，会把Key存放内存，Value会存放在swap区  
#  
# maxmemory的设置比较适合于把redis当作于类似memcached的缓存来使用，而不适合当做一个真实的DB。  
# 当把Redis当做一个真实的数据库使用的时候，内存使用将是一个很大的开销  
# maxmemory <bytes>

# 当内存达到最大值的时候Redis会选择删除哪些数据？有五种方式可供选择  
#   
# volatile-lru -> 利用LRU算法移除设置过过期时间的key (LRU:最近使用 Least Recently Used )  
# allkeys-lru -> 利用LRU算法移除任何key  
# volatile-random -> 移除设置过过期时间的随机key  
# allkeys->random -> remove a random key, any key   
# volatile-ttl -> 移除即将过期的key(minor TTL)  
# noeviction -> 不移除任何可以，只是返回一个写错误  
#   
# 注意：对于上面的策略，如果没有合适的key可以移除，当写的时候Redis会返回一个错误  
#  
# 写命令包括: set setnx setex append  
# incr decr rpush lpush rpushx lpushx linsert lset rpoplpush sadd  
# sinter sinterstore sunion sunionstore sdiff sdiffstore zadd zincrby  
# zunionstore zinterstore hset hsetnx hmset hincrby incrby decrby  
# getset mset msetnx exec sort  
#  
# 默认是:  
#  
# maxmemory-policy volatile-lru

# LRU 和 minimal TTL 算法都不是精准的算法，但是相对精确的算法(为了节省内存)，随意你可以选择样本大小进行检测。  
# Redis默认的灰选择3个样本进行检测，你可以通过maxmemory-samples进行设置  
#  
# maxmemory-samples 3

############################## AOF ###############################

# 默认情况下，redis会在后台异步的把数据库镜像备份到磁盘，但是该备份是非常耗时的，而且备份也不能很频繁，如果发生诸如拉闸限电、拔插头等状况，那么将造成比较大范围的数据丢失。  
# 所以redis提供了另外一种更加高效的数据库备份及灾难恢复方式。  
# 开启append only模式之后，redis会把所接收到的每一次写操作请求都追加到appendonly.aof文件中，当redis重新启动时，会从该文件恢复出之前的状态。  
# 但是这样会造成appendonly.aof文件过大，所以redis还支持了BGREWRITEAOF指令，对appendonly.aof 进行重新整理。  
# 你可以同时开启asynchronous dumps 和 AOF

appendonly no

# AOF文件名称 (默认: "appendonly.aof")  
# appendfilename appendonly.aof

# Redis支持三种同步AOF文件的策略:  
#  
# no: 不进行同步，系统去操作 . Faster.  
# always: always表示每次有写操作都进行同步. Slow, Safest.  
# everysec: 表示对写操作进行累积，每秒同步一次. Compromise.  
#  
# 默认是"everysec"，按照速度和安全折中这是最好的。  
# 如果想让Redis能更高效的运行，你也可以设置为"no"，让操作系统决定什么时候去执行  
# 或者相反想让数据更安全你也可以设置为"always"  
#  
# 如果不确定就用 "everysec".

# appendfsync always  
appendfsync everysec  
# appendfsync no

# AOF策略设置为always或者everysec时，后台处理进程(后台保存或者AOF日志重写)会执行大量的I/O操作  
# 在某些Linux配置中会阻止过长的fsync()请求。注意现在没有任何修复，即使fsync在另外一个线程进行处理  
#  
# 为了减缓这个问题，可以设置下面这个参数no-appendfsync-on-rewrite  
#  
# This means that while another child is saving the durability of Redis is  
# the same as "appendfsync none", that in pratical terms means that it is  
# possible to lost up to 30 seconds of log in the worst scenario (with the  
# default Linux settings).  
#   
# If you have latency problems turn this to "yes". Otherwise leave it as  
# "no" that is the safest pick from the point of view of durability.  
no-appendfsync-on-rewrite no

# Automatic rewrite of the append only file.  
# AOF 自动重写  
# 当AOF文件增长到一定大小的时候Redis能够调用 BGREWRITEAOF 对日志文件进行重写   
#   
# 它是这样工作的：Redis会记住上次进行些日志后文件的大小(如果从开机以来还没进行过重写，那日子大小在开机的时候确定)  
#  
# 基础大小会同现在的大小进行比较。如果现在的大小比基础大小大制定的百分比，重写功能将启动  
# 同时需要指定一个最小大小用于AOF重写，这个用于阻止即使文件很小但是增长幅度很大也去重写AOF文件的情况  
# 设置 percentage 为0就关闭这个特性

auto-aof-rewrite-percentage 100  
auto-aof-rewrite-min-size 64mb

################################## SLOW LOG ###################################

# Redis Slow Log 记录超过特定执行时间的命令。执行时间不包括I/O计算比如连接客户端，返回结果等，只是命令执行时间  
#   
# 可以通过两个参数设置slow log：一个是告诉Redis执行超过多少时间被记录的参数slowlog-log-slower-than(微妙)，  
# 另一个是slow log 的长度。当一个新命令被记录的时候最早的命令将被从队列中移除

# 下面的时间以微妙微单位，因此1000000代表一分钟。  
# 注意制定一个负数将关闭慢日志，而设置为0将强制每个命令都会记录  
slowlog-log-slower-than 10000

# 对日志长度没有限制，只是要注意它会消耗内存  
# 可以通过 SLOWLOG RESET 回收被慢日志消耗的内存  
slowlog-max-len 1024

################################ VM ###############################

### WARNING! Virtual Memory is deprecated in Redis 2.4  
### The use of Virtual Memory is strongly discouraged.

# Virtual Memory allows Redis to work with datasets bigger than the actual  
# amount of RAM needed to hold the whole dataset in memory.  
# In order to do so very used keys are taken in memory while the other keys  
# are swapped into a swap file, similarly to what operating systems do  
# with memory pages.  
#  
# To enable VM just set 'vm-enabled' to yes, and set the following three  
# VM parameters accordingly to your needs.

vm-enabled no  
# vm-enabled yes

# This is the path of the Redis swap file. As you can guess, swap files  
# can't be shared by different Redis instances, so make sure to use a swap  
# file for every redis process you are running. Redis will complain if the  
# swap file is already in use.  
#  
# The best kind of storage for the Redis swap file (that's accessed at random)   
# is a Solid State Disk (SSD).  
#  
# \*\*\* WARNING \*\*\* if you are using a shared hosting the default of putting  
# the swap file under /tmp is not secure. Create a dir with access granted  
# only to Redis user and configure Redis to create the swap file there.  
vm-swap-file /tmp/redis.swap

# vm-max-memory configures the VM to use at max the specified amount of  
# RAM. Everything that deos not fit will be swapped on disk \*if\* possible, that  
# is, if there is still enough contiguous space in the swap file.  
#  
# With vm-max-memory 0 the system will swap everything it can. Not a good  
# default, just specify the max amount of RAM you can in bytes, but it's  
# better to leave some margin. For instance specify an amount of RAM  
# that's more or less between 60 and 80% of your free RAM.  
vm-max-memory 0

# Redis swap files is split into pages. An object can be saved using multiple  
# contiguous pages, but pages can't be shared between different objects.  
# So if your page is too big, small objects swapped out on disk will waste  
# a lot of space. If you page is too small, there is less space in the swap  
# file (assuming you configured the same number of total swap file pages).  
#  
# If you use a lot of small objects, use a page size of 64 or 32 bytes.  
# If you use a lot of big objects, use a bigger page size.  
# If unsure, use the default :)  
vm-page-size 32

# Number of total memory pages in the swap file.  
# Given that the page table (a bitmap of free/used pages) is taken in memory,  
# every 8 pages on disk will consume 1 byte of RAM.  
#  
# The total swap size is vm-page-size \* vm-pages  
#  
# With the default of 32-bytes memory pages and 134217728 pages Redis will  
# use a 4 GB swap file, that will use 16 MB of RAM for the page table.  
#  
# It's better to use the smallest acceptable value for your application,  
# but the default is large in order to work in most conditions.  
vm-pages 134217728

# Max number of VM I/O threads running at the same time.  
# This threads are used to read/write data from/to swap file, since they  
# also encode and decode objects from disk to memory or the reverse, a bigger  
# number of threads can help with big objects even if they can't help with  
# I/O itself as the physical device may not be able to couple with many  
# reads/writes operations at the same time.  
#  
# The special value of 0 turn off threaded I/O and enables the blocking  
# Virtual Memory implementation.  
vm-max-threads 4

############################### ADVANCED CONFIG ###############################

# 当hash中包含超过指定元素个数并且最大的元素没有超过临界时，  
# hash将以一种特殊的编码方式（大大减少内存使用）来存储，这里可以设置这两个临界值  
# Redis Hash对应Value内部实际就是一个HashMap，实际这里会有2种不同实现，  
# 这个Hash的成员比较少时Redis为了节省内存会采用类似一维数组的方式来紧凑存储，而不会采用真正的HashMap结构，对应的value redisObject的encoding为zipmap,  
# 当成员数量增大时会自动转成真正的HashMap,此时encoding为ht。  
hash-max-zipmap-entries 512  
hash-max-zipmap-value 64

# list数据类型多少节点以下会采用去指针的紧凑存储格式。  
# list数据类型节点值大小小于多少字节会采用紧凑存储格式。  
list-max-ziplist-entries 512  
list-max-ziplist-value 64

# set数据类型内部数据如果全部是数值型，且包含多少节点以下会采用紧凑格式存储。  
set-max-intset-entries 512

# zsort数据类型多少节点以下会采用去指针的紧凑存储格式。  
# zsort数据类型节点值大小小于多少字节会采用紧凑存储格式。  
zset-max-ziplist-entries 128  
zset-max-ziplist-value 64

# Redis将在每100毫秒时使用1毫秒的CPU时间来对redis的hash表进行重新hash，可以降低内存的使用  
#   
# 当你的使用场景中，有非常严格的实时性需要，不能够接受Redis时不时的对请求有2毫秒的延迟的话，把这项配置为no。  
#  
# 如果没有这么严格的实时性要求，可以设置为yes，以便能够尽可能快的释放内存  
activerehashing yes

################################## INCLUDES ###################################

# 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  
# include /path/to/other.conf