# Redis配置文件样例

# Note on units: when memory size is needed, it is possible to specifiy

# it in the usual form of 1k 5GB 4M and so forth:

#

# 1k => 1000 bytes

# 1kb => 1024 bytes

# 1m => 1000000 bytes

# 1mb => 1024\*1024 bytes

# 1g => 1000000000 bytes

# 1gb => 1024\*1024\*1024 bytes

#

# units are case insensitive so 1GB 1Gb 1gB are all the same.

# 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

# 启用守护进程后，Redis会把pid写到一个pidfile中，在/var/run/redis.pid

daemonize no

# 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

# 指定Redis监听端口，默认端口为6379

# 如果指定0端口，表示Redis不监听TCP连接

port 6379

# 绑定的主机地址

# 你可以绑定单一接口，如果没有绑定，所有接口都会监听到来的连接

# 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

# 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 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 (很多信息, 对开发／测试比较有用)

# verbose (many rarely useful info, but not a mess like the debug level)

# notice (moderately verbose, what you want in production probably)

# warning (only very important / critical messages are logged)

loglevel verbose

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

logfile stdout

# 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

# 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

# dbid是从0到‘databases’-1的数目

databases 16

################################ SNAPSHOTTING #################################

# 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 the DB on disk:

#

# save <seconds> <changes>

#

# Will save the DB if both the given number of seconds and the given

# number of write operations against the DB occurred.

#

# 满足以下条件将会同步数据:

# 900秒（15分钟）内有1个更改

# 300秒（5分钟）内有10个更改

# 60秒内有10000个更改

# Note: 可以把所有“save”行注释掉，这样就取消同步操作了

save 900 1

save 300 10

save 60 10000

# 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

# 指定本地数据库文件名，默认值为dump.rdb

dbfilename dump.rdb

# 工作目录.

# 指定本地数据库存放目录，文件名由上一个dbfilename配置项指定

#

# Also the Append Only File will be created inside this directory.

#

# 注意，这里只能指定一个目录，不能指定文件名

dir ./

################################# REPLICATION #################################

# 主从复制。使用slaveof从 Redis服务器复制一个Redis实例。注意，该配置仅限于当前slave有效

# so for example it is possible to configure the slave to save the DB with a

# different interval, or to listen to another port, and so on.

# 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，slav服务连接master的密码

# 下文的“requirepass”配置项可以指定密码

# masterauth <master-password>

# When a slave lost the connection with the master, or when the replication

# is still in progress, the slave can act in two different ways:

#

# 1) if slave-serve-stale-data is set to 'yes' (the default) the slave will

# still reply to client requests, possibly with out of data data, or the

# data set may just be empty if this is the first synchronization.

#

# 2) if slave-serve-stale data is set to 'no' the slave will reply with

# an error "SYNC with master in progress" to all the kind of commands

# but to INFO and SLAVEOF.

#

slave-serve-stale-data yes

# Slaves send PINGs to server in a predefined interval. It's possible to change

# this interval with the repl\_ping\_slave\_period option. The default value is 10

# seconds.

#

# repl-ping-slave-period 10

# The following option sets a timeout for both Bulk transfer I/O timeout and

# master data or ping response timeout. The default value is 60 seconds.

#

# It is important to make sure that this value is greater than the value

# specified for repl-ping-slave-period otherwise a timeout will be detected

# every time there is low traffic between the master and the slave.

#

# repl-timeout 60

################################## SECURITY ###################################

# Warning: since Redis is pretty fast an outside user can try up to

# 150k passwords per second against a good box. This means that you should

# use a very strong password otherwise it will be very easy to break.

# 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

# Command renaming.

#

# It is possilbe to change the name of dangerous commands in a shared

# environment. For instance the CONFIG command may be renamed into something

# of hard to guess so that it will be still available for internal-use

# tools but not available for general clients.

#

# Example:

#

# rename-command CONFIG b840fc02d524045429941cc15f59e41cb7be6c52

#

# It is also possilbe to completely kill a command renaming it into

# an empty string:

#

# rename-command CONFIG ""

################################### LIMITS ####################################

# 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

# Don't use more memory than the specified amount of bytes.

# When the memory limit is reached Redis will try to remove keys with an

# EXPIRE set. It will try to start freeing keys that are going to expire

# in little time and preserve keys with a longer time to live.

# Redis will also try to remove objects from free lists if possible.

#

# If all this fails, Redis will start to reply with errors to commands

# that will use more memory, like SET, LPUSH, and so on, and will continue

# to reply to most read-only commands like GET.

#

# WARNING: maxmemory can be a good idea mainly if you want to use Redis as a

# 'state' server or cache, not as a real DB. When Redis is used as a real

# database the memory usage will grow over the weeks, it will be obvious if

# it is going to use too much memory in the long run, and you'll have the time

# to upgrade. With maxmemory after the limit is reached you'll start to get

# errors for write operations, and this may even lead to DB inconsistency.

# 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>

# MAXMEMORY POLICY: how Redis will select what to remove when maxmemory

# is reached? You can select among five behavior:

#

# volatile-lru -> remove the key with an expire set using an LRU algorithm

# allkeys-lru -> remove any key accordingly to the LRU algorithm

# volatile-random -> remove a random key with an expire set

# allkeys->random -> remove a random key, any key

# volatile-ttl -> remove the key with the nearest expire time (minor TTL)

# noeviction -> don't expire at all, just return an error on write operations

#

# Note: with all the kind of policies, Redis will return an error on write

# operations, when there are not suitable keys for eviction.

#

# At the date of writing this commands are: 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

#

# The default is:

#

# maxmemory-policy volatile-lru

# LRU and minimal TTL algorithms are not precise algorithms but approximated

# algorithms (in order to save memory), so you can select as well the sample

# size to check. For instance for default Redis will check three keys and

# pick the one that was used less recently, you can change the sample size

# using the following configuration directive.

#

# maxmemory-samples 3

############################## APPEND ONLY MODE ###############################

#

# Note that you can have both the async dumps and the append only file if you

# like (you have to comment the "save" statements above to disable the dumps).

# Still if append only mode is enabled Redis will load the data from the

# log file at startup ignoring the dump.rdb file.

# 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

# IMPORTANT: Check the BGREWRITEAOF to check how to rewrite the append

# log file in background when it gets too big.

appendonly no

# 指定更新日志文件名，默认为appendonly.aof

# appendfilename appendonly.aof

# The fsync() call tells the Operating System to actually write data on disk

# instead to wait for more data in the output buffer. Some OS will really flush

# data on disk, some other OS will just try to do it ASAP.

# 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

# appendfsync no

# When the AOF fsync policy is set to always or everysec, and a background

# saving process (a background save or AOF log background rewriting) is

# performing a lot of I/O against the disk, in some Linux configurations

# Redis may block too long on the fsync() call. Note that there is no fix for

# this currently, as even performing fsync in a different thread will block

# our synchronous write(2) call.

#

# In order to mitigate this problem it's possible to use the following option

# that will prevent fsync() from being called in the main process while a

# BGSAVE or BGREWRITEAOF is in progress.

#

# 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.

# Redis is able to automatically rewrite the log file implicitly calling

# BGREWRITEAOF when the AOF log size will growth by the specified percentage.

#

# This is how it works: Redis remembers the size of the AOF file after the

# latest rewrite (or if no rewrite happened since the restart, the size of

# the AOF at startup is used).

#

# This base size is compared to the current size. If the current size is

# bigger than the specified percentage, the rewrite is triggered. Also

# you need to specify a minimal size for the AOF file to be rewritten, this

# is useful to avoid rewriting the AOF file even if the percentage increase

# is reached but it is still pretty small.

#

# Specify a precentage of zero in order to disable the automatic AOF

# rewrite feature.

auto-aof-rewrite-percentage 100

auto-aof-rewrite-min-size 64mb

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

# The Redis Slow Log is a system to log queries that exceeded a specified

# execution time. The execution time does not include the I/O operations

# like talking with the client, sending the reply and so forth,

# but just the time needed to actually execute the command (this is the only

# stage of command execution where the thread is blocked and can not serve

# other requests in the meantime).

#

# You can configure the slow log with two parameters: one tells Redis

# what is the execution time, in microseconds, to exceed in order for the

# command to get logged, and the other parameter is the length of the

# slow log. When a new command is logged the oldest one is removed from the

# queue of logged commands.

# The following time is expressed in microseconds, so 1000000 is equivalent

# to one second. Note that a negative number disables the slow log, while

# a value of zero forces the logging of every command.

slowlog-log-slower-than 10000

# There is no limit to this length. Just be aware that it will consume memory.

# You can reclaim memory used by the slow log with SLOWLOG RESET.

slowlog-max-len 1024

################################ VIRTUAL MEMORY ###############################

### WARNING! Virtual Memory is deprecated in Redis 2.4

### The use of Virtual Memory is strongly discouraged.

### 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.

# 指定是否启用虚拟内存机制，默认值为no，

# 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

# 把vm-enabled设置为yes，根据需要设置好接下来的三个VM参数，就可以启动VM了

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.

#

# Redis交换文件最好的存储是SSD（固态硬盘）

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

# \*\*\* 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

# 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.

# 将所有大于vm-max-memory的数据存入虚拟内存，无论vm-max-memory设置多少，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

# 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。

# 建议如果存储很多小对象，page大小最后设置为32或64bytes；如果存储很大的对象，则可以使用更大的page，如果不确定，就使用默认值

vm-page-size 32

# 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）是存放在内存中的，在磁盘上每8个pages将消耗1byte的内存

# swap空间总容量为 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.

# 设置访问swap文件的I/O线程数，最后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

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

# Hashes are encoded in a special way (much more memory efficient) when they

# have at max a given numer of elements, and the biggest element does not

# exceed a given threshold. You can configure this limits with the following

# configuration directives.

# 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 512

hash-max-zipmap-value 64

# Similarly to hashes, small lists are also encoded in a special way in order

# to save a lot of space. The special representation is only used when

# you are under the following limits:

list-max-ziplist-entries 512

list-max-ziplist-value 64

# Sets have a special encoding in just one case: when a set is composed

# of just strings that happens to be integers in radix 10 in the range

# of 64 bit signed integers.

# The following configuration setting sets the limit in the size of the

# set in order to use this special memory saving encoding.

set-max-intset-entries 512

# Similarly to hashes and lists, sorted sets are also specially encoded in

# order to save a lot of space. This encoding is only used when the length and

# elements of a sorted set are below the following limits:

zset-max-ziplist-entries 128

zset-max-ziplist-value 64

# Active rehashing uses 1 millisecond every 100 milliseconds of CPU time in

# order to help rehashing the main Redis hash table (the one mapping top-level

# keys to values). The hash table implementation redis uses (see dict.c)

# performs a lazy rehashing: the more operation you run into an hash table

# that is rhashing, the more rehashing "steps" are performed, so if the

# server is idle the rehashing is never complete and some more memory is used

# by the hash table.

#

# The default is to use this millisecond 10 times every second in order to

# active rehashing the main dictionaries, freeing memory when possible.

#

# If unsure:

# use "activerehashing no" if you have hard latency requirements and it is

# not a good thing in your environment that Redis can reply form time to time

# to queries with 2 milliseconds delay.

# 指定是否激活重置哈希，默认为开启

activerehashing yes

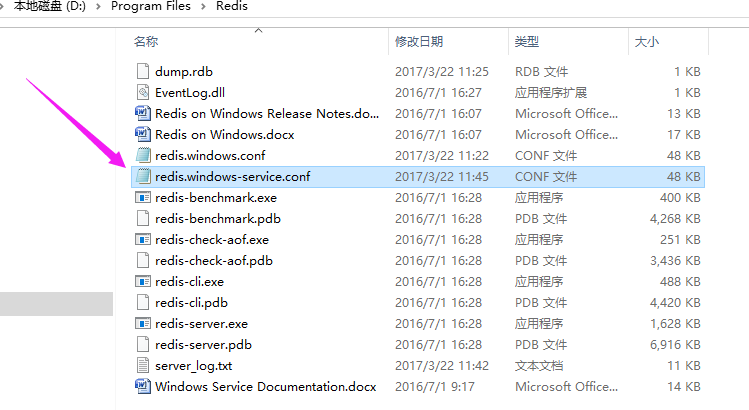
################################## INCLUDES ###################################

# 指定包含其他的配置文件，可以在同一主机上多个Redis实例之间使用同一份配置文件，而同时各实例又拥有自己的特定配置文件

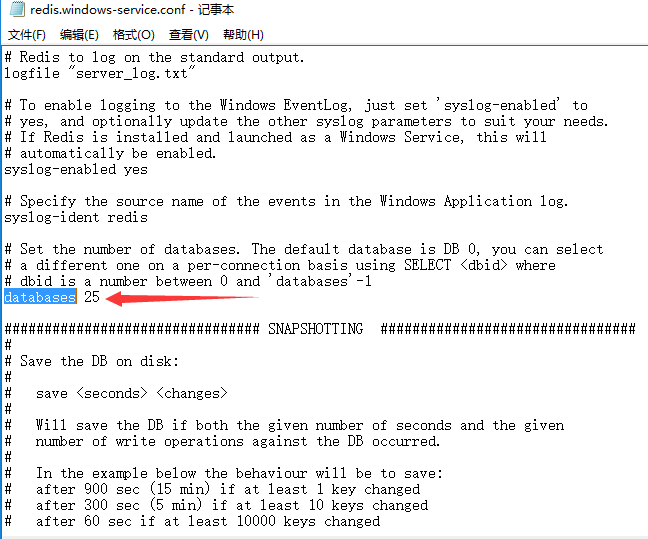
# include /path/to/local.conf

# include /path/to/other.conf

Redis默认16个数据库，并以数字为索引，可以手工修改这个配置，增加数量。



找到redis配置文件，打开：



用记事本编辑，并保存后，在windows服务中，找到redis服务，重启，即可完成配置读取。

