# Redis 配置文件示例。

#

# 注意，为了读取配置文件，Redis 必须

# 以文件路径作为第一个参数开始：

#

# ./redis-server /path/to/redis.conf

# 关于单位的注意事项：当需要内存大小时，可以指定

# 它 以1k 5GB 4M等通常的形式出现：

#

# 1k => 1000 字节

# 1kb => 1024 字节

# 1m => 1000000 字节

# 1mb => 1024\*1024 字节

# 1g => 1000000000 字节

# 1gb => 1024\*1024\*1024 字节

#

# 单位 不区分大小写，因此 1GB 1Gb 1gB 都是相同的。

###########################################

# 在此处包含一个或多个其他配置文件。 如果您

# 有 一个标准模板，可以转到所有Redis服务器，但也需要

# 来 自定义一些每个服务器的设置。 包含文件可以包含

# 其他 文件，所以明智地使用它。

#

# 注意选项"包含"不会被命令"配置重写"重写

# 来自 管理员或 Redis Sentinel。由于 Redis 始终使用上次处理的

# 行 作为配置指令的值，你最好把包含

# 在此文件的开头，以避免在运行时覆盖配置更改。

#

# 如果您有兴趣使用包含来覆盖配置

# 选项，最好使用包含作为最后一行。

#

# include /path/to/local.conf

# include /path/to/other.conf

######

# 默认情况下 ，Redis 不作为守护程序运行。如果需要，请使用"是"。

# 请注意，当守护程序化时，Redis 会在 /var/run/redis.pid 中写入一个 pid 文件。

守护程序 否

# 当运行守护程序化时，Redis 在 /var/run/redis.pid 中写入一个 pid 文件

# 默认值。您可以在此处指定自定义 pid 文件位置。

pidfile /var/run/redis.pid

# 接受指定端口上的连接，默认为6379。

# 如果指定了端口 0，Redis 将不会侦听 TCP 套接字。

端口 6379

# TCP listen（） backlog.

#

# 在每秒请求数高的环境中，您需要高积压工作才能完成

# 以避免客户端连接速度慢的问题。请注意，Linux 内核

# 将 静默地将其截断为 /proc/sys/net/core/somaxconn 的值， 因此

# 确保 同时提高 somaxconn 的值和 tcp\_max\_syn\_backlog

#以获得所需的效果。

tcp-backlog 511

# 默认情况下 ，Redis 会侦听来自所有网络接口的连接

# 在服务器上可用。可以只听一个或多个

# 接口使用 "bind" 配置指令，后跟一个或

# 更多 IP 地址。

#

# 示例：

#

# bind 192.168.1.100 10.0.0.1

# 绑定 127.0.0.1

# 指定将用于侦听的 Unix 套接字的路径

# 传入 连接。没有默认值，因此 Redis 不会侦听

# 未指定时在 unix 套接字上。

#

# unixsocket /tmp/redis.sock

# unixsocketperm 700

# 客户端空闲 N 秒后关闭连接（0 表示禁用）

超时 0

# TCP 保持活动状态。

#

# 如果为非零，请使用SO\_KEEPALIVE在没有 TCP ACK 的情况下向客户端发送 TCP ACK

通信 的数量 。这很有用，原因有两个：

#

# 1）检测失效的对等体。

#2）从网络的角度来看，使连接活着

# 设备 在中间。

#

# 在 Linux 上，指定的值（以秒为单位）是用于发送 ACK 的时间段。

#请注意，要关闭连接，需要双倍的时间。

# 在其他内核上，时间段取决于内核配置。

#

# 此选项的合理值为 60 秒。

tcp-keepalive 0

# 指定服务器详细级别。

# 这可以是以下之一：

# 调试 （大量信息，对开发/测试有用）

# 详细 （许多很少有用的信息，但不是像调试级别那样的混乱）

# 注意 （适度详细，你可能在生产中想要什么）

# 警告 （仅记录非常重要/关键的消息）

日志级别 通知

# 指定日志文件名。 空 字符串也可用于强制

# Redis 登录标准输出。请注意，如果您使用标准

# 输出 用于日志记录，但 守护程序，日志将发送到 /dev/null

日志文件 ""

# 要启用系统记录器的日志记录，只需将"syslog-enabled"设置为 yes，

# 并 可选择更新其他 syslog 参数以满足您的需要。

# 启用了系统日志否

# 指定系统日志标识。

# syslog-ident redis

# 指定系统日志工具。必须是 USER 或介于 LOCAL0-LOCAL7 之间。

# syslog-facility local0

# 设置数据库数量。默认数据库为 DB 0，您可以选择

# 基于每个连接的不同，使用 SELECT <dbid>其中

# dbid 是介于 0 和 "databases"-1 之间的数字

数据库 16

#######

#

# 将数据库保存在磁盘上：

#

# 保存<秒> <更改>

#

# 如果给定的秒数和给定的秒数，将保存 DB

# 发生了针对数据库的写入操作数。

#

# 在下面的示例中， 行为 将是保存：

# 900 秒（15 分钟）后，如果至少更改了 1 个密钥

# 300 秒（5 分钟）后，如果至少更改了 10 个键

# 60 秒后，如果至少更改了 10000 个密钥

#

#注意：您可以通过注释掉所有"保存"行来完全禁用保存。

#

# 也可以删除所有以前配置的保存

# 通过添加具有单个空字符串参数的 save 指令来点

# 如 以下示例所示：

#

# 保存 ""

节省 900 1

节省 300 10

节省 60 10000

# 默认情况下 ，如果启用了 RDB 快照，Redis 将停止接受写入

# （至少一个 保存 点）和最新的后台保存失败。

#这将使用户意识到（以一种艰难的方式）数据没有持久化

# 在磁盘上 正确，否则很可能没有人会注意到和一些

# 灾难 将会发生。

#

# 如果后台保存过程将再次开始工作 Redis 将

# 自动 允许再次写入。

#

# 但是 ，如果您已经设置了对 Redis 服务器的正确监控

# 和 持久性，您可能希望禁用此功能，以便 Redis 将

# 即使磁盘有问题，也能照常工作，

# 权限，依此类推。

停止-写入-在 bgsave-错误 是

# 转储时使用 LZF 压缩字符串对象 。rdb 数据库？

#默认情况下，将其设置为"是"，因为它几乎总是获胜。

# 如果要在保存子级中保存一些CPU，请将其设置为"否"，但是

# 如果您有可压缩的值或键，数据集可能会更大。

rdb压缩 是

# 由于RDB的版本5，CRC64校验和被放置在文件的末尾。

#这使得格式更耐损坏，但有一个性能

# 在 保存和加载RDB文件时点击付款（约10%），因此您可以禁用它

# 实现 最佳性能。

#

# 禁用校验和创建的RDB文件的校验和为零

# 告诉 加载代码跳过检查。

rdbchecksum yes

# 转储数据库的文件名

dbfilename dump.rdb

# 工作目录。

#

# DB将写入此目录中，并指定文件名

# 使用 'dbfilename' 配置指令。

#

# "仅追加文件"也将在此目录中创建。

#

# 请注意，您必须在此处指定目录，而不是文件名。

目录 。/

############################################

# 主从复制。使用 slaveof 使 Redis 实例成为

# 另一个 Redis 服务器。尽快了解 Redis 复制的一些事项。

#

# 1） Redis 复制是异步的，但您可以将主服务器配置为

# 停止 接受写入，如果它似乎没有连接到至少

# 给定数量的从站。

# 2） Redis 从站能够与

# 如果复制链路丢失了相对较少的量，则为主节点

# 时间。您可能需要配置复制积压工作 （请参阅下一个

# 此 文件的部分），根据您的需要提供合理的值。

#3）复制是自动的，不需要用户干预。在

# 网络 分区从站自动尝试重新连接主站

# 并与 它们重新同步。

#

# slaveof <masterip> <masterport>

# 如果主站有密码保护（使用"requirepass"配置

# 指令 如下） 可以告诉从站进行身份验证之前

# 启动 复制同步过程，否则主节点将

# 拒绝 从属请求。

#

# 主< 主密码>

# 当从站失去与主站的连接时，或者当复制时

# 仍在 进行中，从站可以通过两种不同的方式行事：

#

#1）如果从属服务陈旧数据设置为"是"（默认值），则从属服务器将

# 仍然 回复客户端请求，可能包含过期的数据，或者

# 如果这是第一次同步，则数据集可能刚为空。

#

#2）如果从属服务过时数据设置为"否"，则从属服务器将回复

# 所有 命令的错误"与主节点同步"到所有类型的命令

# 但 到信息和奴隶。

#

从属服务过时数据 是

# 您可以将从属实例配置为接受或不接受写入。写反对

# 从属实例可能有助于存储一些临时数据（因为数据

# 写 在从站上，与主站重新同步后很容易删除），但是

# 如果客户端由于

# 配置错误。

#

# 由于 Redis 2.6 默认从属设备是只读的。

#

# 注意：只读从站不是为向不受信任的客户端公开而设计的

#在互联网上。它只是一个防止滥用实例的保护层。

# 默认情况下，只读从站会导出所有管理命令

# 如 配置、调试等。在有限的程度上，你可以改善

# 只读从站的安全性，使用"重命名命令"来隐藏所有

# 管理/危险命令。

从属只读是

# 复制同步策略：磁盘或套接字。

#

# -------------------------------------------------------

# 警告：无盘复制目前处于试验阶段

# -------------------------------------------------------

#

# 新的从站和重新连接无法继续复制的从站

# 处理 只是接收差异，需要做什么叫"全"

# 同步"。RDB文件从主站传输到从站。

# 传输可以通过两种不同的方式发生：

#

#1）磁盘备份：Redis主节点创建一个新进程来写入RDB

# 磁盘上的文件。稍后，文件由父级传输

# 进程 增量到从站。

#2）无盘：Redis主节点创建一个新进程，直接写入

# RDB文件到从属套接字，根本不接触磁盘。

#

# 使用磁盘备份复制，在生成RDB文件的同时，从站更多

# 可以 排队，并在当前子产品生成后立即与RDB文件一起提供

# RDB 文件完成其工作。使用无盘复制代替一次

# 转移开始，新的从站到达将排队，新的转移

# 将在 当前一个终止时启动。

#

# 使用无盘复制时，主服务器等待可配置的

# 时间 （以秒为单位）开始传输之前，希望多个从属

# 将 到达，传输可以并行化。

#

# 使用慢速磁盘和快速（大带宽）网络，无盘复制

# 效果 更好。

repl-disk-sync no

# 启用无盘复制后，可以配置延迟

# 服务器等待生成通过套接字传输 RDB 的子级

# 给 奴隶。

#

#这很重要，因为一旦转移开始，就不可能服务

# 新的 从站到达，即将排队等待下一次RDB传输，所以服务器

# 等待 延迟，以便让更多的奴隶到达。

#

# 延迟以秒为单位指定，默认为5秒。禁用

# 它 完全只是将其设置为0秒，传输将尽快开始。

无盘同步延迟 5

# 从站以预定义的时间间隔向服务器发送 PING。可以更改

# 此 间隔与 repl\_ping\_slave\_period 选项。默认值为 10

# 秒。

#

# 重复-ping-从-周期 10

# 以下选项设置以下选项的复制超时：

#

# 1）SYNC期间的批量传输I / O，从从站的角度来看。

#2）从从站（数据，ping）的角度来看主超时。

# 3）从主站的角度来看从超时（REPLCONF ACK pings）。

#

# 请务必确保此值大于该值

# 指定为 repl-ping-slave-period，否则将检测到超时

# 每次 主站和从站之间的流量较低时。

#

# 重复超时 60

# SYNC后禁用从属插槽上的TCP\_NODELAY？

#

# 如果您选择"是"，Redis将使用较少数量的TCP数据包和

# 更少的 带宽将数据发送到从站。但这可能会增加延迟

# 数据出现在从端，最长可达40毫秒

# 使用默认配置的 Linux 内核。

#

#如果您选择"否"，则数据出现在从端的延迟将

# 已 减少，但更多的带宽将用于复制。

#

# 默认情况下 ，我们针对低延迟进行优化，但在非常高的流量条件下

# 或者 当主站和从站相距许多跳跃时，将其转换为"是"可能会

# 成为 一个好主意。

repl-disable-tcp-nodelay no

# 设置复制积压工作大小。积压工作是累积的缓冲区

# 从属 数据当从站断开连接一段时间时，使从属服务器断开连接时

# 想要 再次重新连接，通常不需要完全重新同步，而是部分重新同步

# 重新同步 就足够了，只需传递从属服务器错过的数据部分，而

# 已断开连接。

#

# 复制积压越大，从属时间越长

# 断开连接 ，以后能够执行部分重新同步。

#

# 只有在至少连接了从属服务器时，才会分配积压工作。

#

# 积压工作大小 1mb

# 主站不再连接从站一段时间后，积压工作

# 将被释放。以下选项配置

# 需要 经过，从最后一个从站断开连接开始，对于

# 要释放的积压工作缓冲区。

#

# 值为 0 表示从不释放积压工作。

#

# repl-backlog-ttl 3600

# 从属优先级是 Redis 在 INFO 输出中发布的整数。

# Redis Sentinel用它来选择一个从站来提升成一个

# 如果 主站不再正常工作，则为主站。

#

# 优先级较低的从站被认为更适合升级，因此

# 例如， 如果有三个优先级为 10、100、25 的从站，则哨兵将

# 选择 优先级为10的那个，这是最低的。

#

# 但是 ，特殊优先级为 0 会将从站标记为无法执行

# 主角色，因此优先级为 0 的从站将永远不会被 选择

# Redis Sentinel for Promotion。

#

# 默认情况下 ，优先级为 100。

从属优先级 100

# 如果写入次数少于

# 连接 N 个从站，其滞后小于或等于 M 秒。

#

# N个从站需要处于"联机"状态。

#

# 以秒为单位的滞后，必须< =指定值，计算公式为

# 从从站接收的最后一个 ping，通常每秒发送一次。

#

# 此选项不保证 N 个副本将接受写入，但

# 将 限制丢失写入的暴露窗口，以防没有足够的从属

# 可用 ，达到指定的秒数。

#

# 例如 ，至少需要 3 个延迟< = 10 秒的从站使用：

#

# 最小从站写入 3

# 最小从站最大滞后 10

#

# 将其中一个或另一个设置为 0 将禁用该功能。

#

# 默认情况下 ，最小从写次数设置为 0（禁用功能）和

# 最小从站最大滞后设置为 10。

###########################################

# 要求客户端在处理任何其他内容之前发出 AUTH <PASSWORD>

# 命令。 这在您不信任的环境中可能很有用

# 其他有权访问运行 redis-server 的主机的人。

#

#这应该保持注释掉，以向后兼容，因为大多数

# 人们 不需要身份验证（例如，他们运行自己的服务器）。

#

#警告：由于Redis非常快，外部用户可以尝试

#每秒150k密码对一个好的盒子。这意味着您应该

# 使用 非常强的密码，否则很容易被破解。

#

# requirepass foobared

# 命令重命名。

#

# 可以更改共享中危险命令的名称

# 环境。例如 ，CONFIG命令可能被重命名为某些东西

# 很难 猜测，以便它仍然可用于内部使用的工具

# 但 不适用于一般客户端。

#

# 示例：

#

# 重命名命令 CONFIG b840fc02d524045429941cc15f59e41cb7be6c52

#

# 也可以通过将命令重命名为

# 空 字符串：

#

# 重命名命令配置""

#

# 请注意，更改登录到

# AOF文件或传输到从站可能会导致问题。

#############################################

# 设置同时连接的客户端的最大数量。默认情况下

# 此 限制设置为 10000 个客户端，但是如果 Redis 服务器不是

# 能够 配置进程文件限制以允许指定的限制

# 允许的最大 客户端数设置为当前文件限制

# 减去 32（因为 Redis 保留了一些文件描述符供内部使用）。

#

#一旦达到限制，Redis将关闭所有发送的新连接

# 错误 "达到的最大客户端数"。

#

# 最大客户端 10000

# 不要使用超过指定字节数的内存。

# 当达到内存限制时，Redis 将尝试删除密钥

# 根据 所选的逐出策略（请参见 最大内存策略）。

#

# 如果 Redis 无法根据策略删除密钥，或者如果策略为

# 设置为 "noeviction"，Redis 将开始回复命令的错误

# 将使用更多内存，如 SET、LPUSH 等，并将继续

# 回复 只读命令，如 GET。

#

# 此选项在使用 Redis 作为 LRU 缓存或设置时通常很有用

# 实例的硬内存限制（使用"noeviction"策略）。

#

# 警告：如果您有从站附加到启用了 maxmemory 的实例，

# 减去馈入从站所需的输出缓冲器的大小

# 从 已用内存计数，以便网络问题/重新同步将

# 不 触发密钥被逐出的循环，进而输出

# 从站缓冲区已满，DL 的密钥被逐出触发删除

的更多键，依此类推，直到数据库被完全清空。

#

# 简而言之...如果您附加了从站，建议您设置较低的

#最大内存限制，以便系统上有一些用于从站的可用RAM

# 输出缓冲区（但如果策略为"noeviction"，则不需要这样做）。

#

# 最大内存 <字节>

# MAXMEMORY 政策：Redis 将如何选择在最大内存时删除的内容

已到达 # 。您可以从以下五种行为中进行选择：

#

# volatile-lru ->使用 LRU 算法删除具有过期集的密钥

# allkeys-lru ->根据 LRU 算法删除任何密钥

# volatile-random ->删除具有过期集的随机密钥

# allkeys-random ->删除一个随机密钥，任何密钥

# volatile-ttl ->删除过期时间最接近的密钥（次要 TTL）

# noeviction ->根本不过期，只需在写入操作时返回错误

#

# 注意：使用上述任何策略，Redis 都会在写入时返回错误

# 操作，当没有合适的键进行逐出时。

#

# 在编写这些命令之日，这些命令是： setnx setex append

# incr decr rpush lpush rpushx lpushx linsert lset rpoplpush sadd

# 烧结 烧结店 sunion sunionstore sdiff sdiffstore zadd zincrby

# zunionstore zinterstore hset hsetnx hmset hincrby incrby decrby

# getset mset msetnx exec 出来了

#

# 默认值为：

#

# 最大内存策略 noeviction

# LRU 和最小 TTL 算法不是精确的算法，而是近似算法

# 算法 （为了节省内存），因此您可以调整它的速度或

# 准确性。对于默认，Redis将检查五个键，并选择一个

# 最近 使用 较少，您可以使用以下方式更改样本数量

# 配置 指令。

#

# 默认值 5 会产生足够好的结果。10 非常接近的近似值

# 真正的 LRU，但 CPU 成本更高。3 非常快，但不是很准确。

#

# 最大内存样本 5

######################################

# 默认情况下 ，Redis 会将数据集异步转储到磁盘上。此模式是

# 在许多应用程序中足够好，但 Redis 进程存在问题或

# 断电可能导致几分钟的写入丢失（取决于

# 配置的保存点）。

#

# "仅追加文件"是一种替代持久性模式，提供

# 更好的耐用性。例如 ，使用默认数据同步策略

# （请参阅稍后的配置文件）Redis 在

# 戏剧性 的事件，如服务器断电，或单次写入，如果有什么

# Redis进程本身发生错误，但操作系统是

# 仍在 正常运行。

#

# AOF和RDB持久化可以同时启用，没有问题。

# 如果在启动时启用了 AOF Redis 将加载 AOF，即文件

# 具有 更好的耐用性保证。

#

# 请查看 http://redis.io/topics/persistence 了解更多信息。

附录 否

# 仅追加文件的名称（默认："appendonly.aof"）

appendfilename "appendonly.aof"

# fsync（） 调用告诉操作系统在磁盘上实际写入数据

# 而不是 等待输出缓冲区中的更多数据。某些操作系统将真正刷新

# 磁盘上的数据 ，其他一些操作系统只会尝试尽快完成。

#

# Redis 支持三种不同的模式：

#

# 不：不要 同步，只是让操作系统在需要时刷新数据。更快。

# always： 每次 写入仅追加日志后 fsync。最慢，最安全。

# 每秒： 每秒只异步 一次。妥协。

#

# 默认值为"每一秒"，因为这通常是正确的折衷方案。

# 速度 和数据安全。这取决于你，看看你是否可以放松这一点

# "no"，当

# 它 想要，为了更好的表现（但如果你能忍受这个想法

# 一些 数据丢失考虑快照的默认持久性模式），

# 或者 相反，使用"总是"，它非常慢，但比更安全

# 每秒。

#

# 更多详情请查看以下文章：

# http://antirez.com/post/redis-persistence-demystified.html

#

# 如果不确定，请使用"每一秒"。

# 始终追加同步

appendfsync everysec

# appendfsync no

# 当 AOF 异步 策略设置为"始终"或 "每秒"时，以及后台

# 保存 过程（后台保存或AOF日志后台重写）是

# 在某些 Linux 配置中，对磁盘执行大量 I/O

# Redis 可能会在 fsync（） 调用上阻塞太长时间。请注意，没有修复程序

# 目前，由于即使在 不同的线程中执行 fsync 也会阻塞

# 我们的同步 写入（2）调用。

#

# 为了缓解 这个问题 ，可以使用以下选项

# 将阻止在主进程中调用 fsync（）， 而

# BGSAVE 或 BGREWRITEAOF 正在进行中。

#

# 这意味着，当另一个孩子在储蓄时，Redis 的耐用性是

# 与"appendfsync none"相同。实际上，这意味着它是

# 在最坏的情况下，可能会丢失长达 30 秒的日志（使用

# 默认 的 Linux 设置）。

#

#如果您有延迟问题，请将其设置为"是"。否则，请将其保留为

#从耐用性的角度来看，"不"是最安全的选择。

no-appendfsync-on-rewrite no

# 自动重写仅追加文件。

# Redis 能够自动重写日志文件隐式调用

# 当 AOF 日志大小按指定百分比增长时，BGREWRITEAOF。

#

# 这是它的工作原理：Redis 会在

# 最新 重写（如果自重新启动以来未发生重写，则大小

# 使用启动时的 AOF）。

#

# 此基本大小与当前大小进行比较。如果当前大小为

# 大于 指定的百分比，将触发重写。也

# 您需要 为要重写的 AOF 文件指定最小大小，如下所示

# 有助于 避免重写 AOF 文件，即使百分比增加

#已到达，但它仍然很小。

#

# 指定零的百分比以禁用自动 AOF

# 重写 功能。

自动重写百分比 100

自动重写最小大小 64mb

# 在 Redis 期间，AOF 文件可能会在末尾被截断

# 启动 过程，当 AOF 数据加载回内存时。

# 当运行 Redis 的系统时，可能会发生这种情况

# 崩溃，特别是当 ext4 文件系统在没有

# data=ordered option（但是当 Redis 本身时，这不会发生

# 崩溃 或中止，但操作系统仍能正常工作）。

#

# Redis 可以在发生这种情况时退出并显示错误，也可以加载尽可能多的内容

# 尽可能的数据 （现在的默认值），如果找到 AOF 文件，则启动

# 在末尾被截断。以下选项控制此行为。

#

# 如果 aof-load-truncated 被设置为 yes，则会加载一个截断的 AOF 文件，并且

# Redis 服务器开始发出日志，通知用户该事件。

# 否则 ，如果该选项设置为 no，服务器将中止并显示错误

# 并 拒绝启动。当该选项设置为 no 时，用户需要

# 在重新启动之前使用"redis-check-aof"实用程序修复AOF文件

# 服务器。

#

# 注意如果AOF文件在中间会被发现损坏

# 服务器仍将退出并显示错误。此选项仅适用于以下情况

# Redis 会尝试从 AOF 文件中读取更多数据，但字节数不足

将找到 # 。

aof-load 截断是

########

# Lua 脚本的最长执行时间（以毫秒为单位）。

#

# 如果达到最大执行时间 Redis 将记录脚本

# 在最大允许时间后仍在执行中，并将开始

# 回复 有错误的查询。

#

# 当长时间运行的脚本超过最大执行时间时，只有

# 脚本终止和关机 NOSAVE 命令可用。第一个可以是

# 用于 停止尚未调用写入命令的脚本。第二个

# 是在 写入命令的情况下关闭服务器的唯一方法

# 已 由脚本发出，但用户不想等待自然

# 脚本终止。

#

# 将其设置为 0 或负值，以便无限制执行而不发出警告。

lua-time-limit 5000

#####

#

# ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

# 警告实验性：Redis 集群被认为是稳定的代码，但是

#为了将其标记为"成熟"，我们需要等待一个非平凡的百分比

要在生产环境中部署它的用户数。

# ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#

# 普通的 Redis 实例不能是 Redis 集群的一部分;仅具有以下条件的节点

# 作为 群集节点可以启动。为了启动 Redis 实例作为

# 集群 节点启用集群支持 取消以下注释：

#

# 已启用群集 是

# 每个集群节点都有一个集群配置文件。此文件不是

# 旨在 手动编辑。它由 Redis 节点创建和更新。

# 每个 Redis 集群节点都需要不同的集群配置文件。

# 确保在同一系统中运行的实例没有

# 重叠的 集群配置文件名。

#

# cluster-config-file nodes-6379.conf

# 集群节点超时是指节点必须无法访问的毫秒数

# 以 将其视为处于失败状态。

# 大多数其他内部时间限制是节点超时的倍数。

#

# 集群节点超时 15000

# 如果故障主站的数据，故障主站的从站将避免启动故障转移

# 看起来 太旧了。

#

# 没有简单的方法可以让从属者真正有 一个 精确的度量

# 它的"数据年龄"，因此执行以下两个检查：

#

#1）如果有多个从站能够故障转移，它们交换消息

# 为了尽量给从业者最好的优势

# 复制 偏移量（处理的主节点中的更多数据）。

# 奴隶将尝试通过偏移量获得他们的排名，并应用于开始

# 的 故障转移延迟与其等级成正比。

#

#2）每个从站都计算最后一次与

# 它 的主人。这可以是最后一次收到 ping 或命令（如果主节点

# 仍处于"已连接"状态），或者自

# 与主服务器 断开连接 （如果复制链路当前已关闭）。

# 如果最后一次交互太旧，从站不会尝试故障转移

# 根本没有 。

#

# 第"2点"可由用户调整。 具体来说 ，从站不会执行

# 故障转移，如果自上次与主节点交互以来，时间

# 已用 大于：

#

# （node-timeout \* slave-validity-factor） + repl-ping-slave-period

#

# 例如 ，如果节点超时为 30 秒，并且从属有效性因子

# 为 10，并且假设默认的 repl-ping-slave 周期为 10 秒，则

# 如果 从站 无法与主站通信，则不会尝试故障转移

# 超过 310 秒。

#

# 较大的从属有效性因素可能允许数据过旧而无法故障转移的从属服务器

# 一个主节点，而太小的值可能会阻止集群

# 選出 一個奴隸。

#

# 为了获得最大的可用性，可以设置从属有效性因子

# 到 值 0，这意味着从属服务器将始终尝试故障转移

# 母版 ，无论他们上次与主站互动的时间如何。

#（但是 他们总是会尝试应用与他们的成比例的延迟

# 偏移 排名）。

#

# 零是唯一能够保证当所有分区愈合时的值

# 集群将始终能够继续。

#

# 集群从站有效性因子 10

# 集群从站能够迁移到孤立的主站，即主站

#没有工作的奴隶。这提高了集群能力

# 抵御故障，否则孤立的主节点无法故障转移

# 如果 失败，如果它没有工作从站。

#

# 只有当至少还有一个

# 给定 其他工作奴隶的数量为他们的旧主人。此数字

# 是 "移民障碍"。迁移势垒为 1 表示奴隶

# 仅当其主站至少有 1 个其他工作从站时，才会迁移

# 等等。它通常反映了您想要的奴隶数量

# 集群中的主节点。

#

# 默认值为 1（从站仅当主节点至少保留时才会迁移

# 一个 从站）。要禁用迁移，只需将其设置为非常大的值即可。

# 可以设置值 0，但仅用于调试和危险

# 在生产中 。

#

# 集群迁移障碍 1

# 默认情况下 ，Redis 集群节点在检测到查询时会停止接受查询

# 至少 是 一个未覆盖的哈希槽（没有可用的节点为其提供服务）。

# 如果集群部分关闭（例如一系列哈希槽），则此方式

# 不再覆盖）所有群集最终都变得不可用。

#一旦所有插槽再次被覆盖，它会自动返回可用。

#

# 但是，有时您需要正在工作的集群子集，

# 继续 接受对仍然处于键空间部分的查询

# 已覆盖。为此，只需设置群集需要全覆盖

# 选项 到否。

#

# 集群需要全覆盖 是

# 为了设置您的集群，请务必阅读文档

# 可在 http://redis.io 网站上找到。

#########################################

# Redis 慢日志是一个系统，用于记录超过指定查询的查询

# 执行 时间。执行时间不包括 I/O 操作

# 比如 与客户交谈、发送回复等，

# 但 只是实际执行命令所需的时间（这是唯一的

# 线程被阻塞且无法服务的命令执行阶段

# 在此期间 的其他 请求）。

#

# 您可以使用两个参数配置慢日志：一个告诉 Redis

# 执行时间是多少，以微秒为单位，为了

# 命令 来获取日志，而另一个参数是

# 慢日志 。记录新命令时，将从

# 记录的命令队列。

# 以下时间以微秒表示，所以1000000等效

# 到 一秒钟。请注意，负数会禁用慢日志，而

# 值为零将强制记录每个命令。

慢日志-日志-慢-小于 10000

# 此长度没有限制。请注意，它会消耗内存。

# 您可以使用 SLOWLOG RESET 回收慢日志使用的内存。

slowlog-max-len 128

########################################

# Redis 延迟监控子系统对不同的操作进行采样

# 在 运行时，以便收集与可能的来源相关的数据

# Redis 实例的延迟。

#

# 通过 LATENCY 命令，此信息可供用户使用，该用户可以

# 打印 图表并获取报告。

#

# 系统只记录在等于或

# 大于 通过

# 延迟监视器阈值配置指令。设置其值时

# 为 零，延迟监视器已关闭。

#

# 默认情况下 ，延迟监控处于禁用状态，因为它几乎不需要

# 如果您 没有延迟问题，并且收集数据具有性能

# 影响，虽然很小，但在大负载下可以测量。延迟

# 监控 可以在运行时使用命令轻松启用

# 如果需要，"配置设置延迟监视器阈值<毫秒>"。

延迟监视器阈值 0

#####################################

# Redis 可以通知发布/订阅客户端密钥空间中发生的事件。

# 此功能记录在 http://redis.io/topics/notifications

#

# 例如，如果启用了密钥空间事件通知，并且客户端

# 对存储在数据库 0 中的密钥"foo"执行 DEL 操作，2

# 消息 将通过发布/订阅发布：

#

# 发布 \_\_keyspace@0\_\_：foo del

# PUBLISH \_\_keyevent@0\_\_：del foo

#

# 可以在一组事件中选择 Redis 将通知的事件

类 的数量 。每个类都由一个字符标识：

#

# K Keyspace 事件，以 \_\_keyspace@<db>\_\_ 前缀发布。

# E Keyevent events，以 \_\_keyevent@<db>\_\_ 前缀发布。

# g 通用命令（非特定于类型），如 DEL、EXPIRE、RENAME、...

# $ 字符串命令

# l 列出命令

# s 设置命令

# h 哈希命令

# z 排序集命令

# x 过期事件（每次密钥过期时生成的事件）

# e 已逐出的事件（在最大内存期间逐出密钥时生成的事件）

# g$lshzxe 的别名，因此"AKE"字符串表示所有事件。

#

# "notify-keyspace-events"将一个组合的字符串作为参数

# 个 零个或多个字符。空字符串表示通知

# 已 禁用。

#

# 示例：启用列表和泛型事件，从

# 事件 名称，使用：

#

# notify-keyspace-events Elg

#

# 示例 2：获取订阅通道的过期密钥流

# 名称 \_\_keyevent@0\_\_：过期 使用：

#

# 通知密钥空间事件 Ex

#

# 默认情况下 ，所有通知都处于禁用状态，因为大多数用户不需要

# 此功能 和该功能有一些开销。请注意，如果您不这样做

# 指定 K 或 E 中的至少一个，不会传递任何事件。

通知密钥空间事件 ""

#########################################

# 哈希值使用内存高效数据结构进行编码，当它们具有

# 条目数量少 ，且最大条目不超过给定

# 阈值。可以使用以下指令配置这些阈值。

哈希-最大-ziplist-entries 512

哈希-最大-ziplist-值 64

# 与 哈希类似，小列表也以特殊的方式按顺序编码

# 节省 大量空间。特殊表示仅在以下情况下使用

# 您 处于以下限制之下：

列表-最大-拉链列表-条目 512

列表最大拉链值 64

# 集合仅在一种情况下具有特殊编码：当集合组成时

恰好是范围内基数 10 中的整数的字符串的数量

# 的 64 位有符号整数。

# 以下配置设置设置了大小限制

# 设置 为了使用这种特殊的内存节省编码。

set-max-intset-entries 512

# 与 哈希和列表类似，排序的集合也特别编码在

# 订单 节省大量空间。此编码仅在长度和

# 排序集的元素低于以下限制：

zset-max-ziplist-entries 128

zset-max-ziplist-value 64

# HyperLogLog 稀疏表示字节限制。限制包括

# 16 字节标头。当使用稀疏表示的 HyperLogLog 交叉时

# 此 限制，它被转换为密集表示。

#

# 大于 16000 的值是完全无用的，因为此时

# 密集 表示更节省内存。

#

# 建议值为~3000，以便有

# 节省空间的编码，无需减慢太多的PFADD，

# 这是 O（N），采用稀疏编码。该值可以提高到

# ~ 10000 当 CPU 不是问题，但空间是，并且数据集是

# 由许多 基数在 0 - 15000 范围内的 HyperLogLog 组成。

hll-稀疏-最大字节 3000

# 主动重哈希每 100 毫秒 CPU 时间使用 1 毫秒

# 为了帮助重新哈希主 Redis 哈希表（映射顶级的那个）

# 值的键 ）。Redis 使用的哈希表实现（请参阅 字典.c）

# 执行 惰性重哈希：您在哈希表中运行的操作越多

# 正在 重述，执行的重哈希"步骤"越多，因此如果

# 服务器 处于空闲状态，重新哈希永远不会完成，并且使用了更多内存

# 通过 哈希表。

#

# 默认为每秒使用10次此毫秒，以便

# 主动 重新处理主词典，尽可能释放内存。

#

# 如果不确定：

# 使用"主动哈希 否"，如果你有硬延迟要求，它是

# 在您的环境中 ，Redis 可以不时回复不是一件好事

# 到 延迟为 2 毫秒的查询。

#

#使用"主动哈希是 "，如果你没有这样的硬要求，但

# 希望 尽可能释放内存。

主动 哈希是

# 客户端输出缓冲区限制可用于强制客户端断开连接

# 由于某种原因（a

# 常见 原因是发布/订阅客户端无法像

# 出版商 可以制作它们）。

#

# 对于三种不同类别的客户端，限制可以有不同的设置：

#

# 正常 -> 普通客户端，包括 MONITOR 客户端

# 从属 -> 从属客户端

# pubsub -> 客户端订阅了至少一个 pubsub 频道或模式

#

# 每个客户端输出缓冲区限制指令的语法如下：

#

# 客户端-输出-缓冲区-限制<类> <硬限制> <软件限制> <软件秒数>

#

# 一旦达到硬限制，客户端将立即断开连接，或者如果

# 已达到软限制，并且仍达到指定数量的软限制

# 秒 （连续）。

# 例如 ，如果硬限制为 32 MB，而软限制为

#16兆字节/ 10秒，客户端将立即断开连接

# 如果 输出缓冲区的大小达到 32 MB，但也会得到

# 如果客户端达到 16 MB 并持续克服，则断开连接

# 10 秒的限制。

#

# 默认情况下 ，普通客户端不受限制，因为它们不接收数据

# 没有 询问（以推送方式），但只是在请求之后，所以只有

# 异步 客户端可能会创建一个请求数据速度更快的场景

# 比 它能读的。

#

# 相反，pubsub 和从属客户端有一个默认限制，因为

# 订户 和从站以推送方式接收数据。

#

# 硬限制或软限制都可以通过设置为零来禁用。

客户端-输出-缓冲区-限制正常 0 0 0

客户端输出缓冲区限制从站 256mb 64mb 60

客户端-输出-缓冲区-限制 pubsub 32mb 8mb 60

# Redis 调用一个内部函数来执行许多后台任务，例如

# 在 timeot 中关闭客户端的连接，清除过期的密钥

# 从未 请求过，依此类推。

#

# 并非所有任务都以相同的频率执行，但 Redis 会检查

# 根据指定的"hz"值执行的任务。

#

# 默认情况下 ，"hz"设置为 10。在以下情况下，提高该值将使用更多 CPU

# Redis处于空闲状态，但同时会让 Redis 在以下情况下响应速度更快

# 有许多 密钥同时到期，超时可能是

# 更精确地处理。

#

# 范围介于 1 和 500 之间，但超过 100 的值通常不是

# 一个好主意。大多数用户应使用默认值 10，并将其提高到

# 100 仅在需要非常低延迟的环境中使用。

赫兹 10

# 当子级重写 AOF 文件时（如果启用了以下选项）

#文件将每生成32 MB数据进行同步。这很有用

# 为了将文件更增量地提交到磁盘并避免

# 大 延迟峰值。

aof-rewrite-incremental-fsync yes

###########################################

# 在此处包含一个或多个其他配置文件。 如果您

# 有 一个标准模板，可以转到所有Redis服务器，但也需要

# 来 自定义一些每个服务器的设置。 包含文件可以包含

# 其他 文件，所以明智地使用它。

#

# include /path/to/local.conf

# include /path/to/other.conf