redis.conf

> 翻译自 Redis 5.0.8 默认配置文件 redis.conf。
> 个人英语水平有限，应以原文档为准。

Redis 配置文件范例。

需要注意的是为了能顺利读取配置文件，Redis 启动时要将配置文件路径作为第一个参数：
./redis-server /path/to/redis.conf

INCLUDES （包含）

在这配置包含一个或多个配置文件。这个配置项适用于那些对大部分 Redis 实例有标准的配置模板，但对小部分 Redis 实例有定制化需求的场景。 包括文件可以包含其他文件，所以请明智使用。

请注意 ”include” 配置不会被 admin 或者 Redis 哨兵 ”CONFIG REWRITE” 命令重写。由于 Redis 总是使用最后处理的行作为配置值，所以最好将 includes 配置放在该文件的最开始以此避免配置在运行的时候被重写。

相反的你想要用 includes 配置来重写配置项，那 include 应该放在最后一行会更好。

#include /path/to/local.conf

#include /path/to/other.conf

MODULES（模块）

启动时（at startup）加载模块。如果 server 加载模块失败服务器会终止（abort）。

#loadmodule /path/to/my_module.so

#loadmodule /path/to/other_moudle.so

NETWORK（网络）

如果没有使用 bind 进行配置，Redis 则默认监听所有 Server 上可以访问的网络接口的连接。如果配置了 bind 指向具体的值，Redis 则只监听配置的那些连接的网络接口。可以是一个 IP 或者紧接着多个 IP 地址。

示例：

#bind 192.167.2.34 10.0.0.1
#bind 127.0.0.1 ::1

~~~ WARNING ~~~ 如果跑 Redis 的机器直接暴露在网络中，binding（指定，绑定）所有的网络接口有潜在的危险，且会让实例暴露给网络上的所有人。因此，我们取消注释了下面的 bind 指令，这会让 Redis 只监听 IPv4 的环回地址（意味着 Redis 只接受跑在和 Redis 实例一台机器上的客户端连接）。

如果你确认你的 Redis 实例可以接受来自所有地址的请求，把下面的指令注释掉即可。

bind 127.0.0.1

保护模式是安全防护的其中一层，保护模式的存在是为了避免暴露在网络中的 Redis 实例被不当的连接滥用（Redis instances left open on the internet are accessed and exploited）。

当保护模式打开且：
1）Redis 服务没有使用“bind”去绑定明确的 ip 地址集合。
2）没有配置密码。
那么，Redis 服务只接受来自 IPv4 和 IPv6 的环回地址 127.0.0.1 和 ::1并且是来自 Unix 域的套接字。

保护模式默认开启。除非你确定你的 Redis 实例在没有配置连接认证或者使用 bind 命令限制特定的 ip 连接的情况下还可以被连接。不然最好保持该模式开启。

protected-mode yes

通过特定端口进行连接，默认端口是 6379（IANA #815344）。如果端口配置成 0，Redis 就不会监听 TCP 套接字。

port 6379

TCP listen() 积压（backlog）。

在高频请求场景下的 Redis，为了避免慢的客户端连接，你需要配置较高的 backlog。提醒事项：Linux 内核会默默的将其截断成 /proc/sys/net/core/somaxconn 的值，所以保证同时提高 somaxconn 和 tcp_max_syn_backlog 的值以求预期的效果。

tcp-backlog 511

Unix 套接字

自己指定特定的 Unix 套接字路径来监听可能来的连接。Redis 没有为此配置默认值，如果你也没有手动去配置指定的话，那 Redis 不会监听一个 unix 套接字。

#unixsocket /tmp/redis.sock
#unixsocketperm 700

N 秒后（0 表示此配置无效），客户端和服务端之间是空闲的，则断开连接。
~~~

timeout 0

TCP keepalive

如果配置了非零的值，使用 SO_KEEPALIVE 发送 TCP 的 ACKs 给那些可能断连的客户端。这很管用，原因有：

1）检测死掉的同伴链接（Detect dead peers）。
2）从中间网络设备的视角来看，连接持续保存。

在 Linux，配置特定的值（单位为 秒）为周期来发送 ACKs。注意事项：需要两倍的该时间来关闭连接。不同的内核中该周期取决于内核的配置。

300 秒是一个比较合理的选择，这也是 Redis 从 3.2.1 版本开始配置的默认值。

tcp-keepalive 300

GENERAL

Redis 运行默认不是守护进程。需要的话将该项配置成 yes。
注意事项：该配置开启后，Redis 会默认在 /var/run/redis.pid 文件中写相关信息。

daemonize no

如果你是以 upstart 或者 systemd 方式跑 Redis，Redis 可以与你的监督数（supervision tree）交互。具体的选项：

- supervised no　– 不进行监督树的交互。
- supervised upstart　– 通过将 Redis 置为 SIGSTOP 模式进行 upstart 信号通知。
- supervised systemd　– 通过将 READY=1 写入 $NOTIFY_SOCKET 进行 systemd 的信号通知。
- supervised auto　– 基于 UPSTART_JOB 或者 NOTIFY_SOCKET 环境变量来检测是 upstart 还是 systemd 方式。

注意：以上的 supervision 方法只通知“处理准备就绪”的信号。他们不会持续的响应你配置的 supervisor。

supervised no

如果配置指定了 pid 文件，Redis 就用该配置的 pid 文件写入，退出的时候移除对应的 pid 文件。

如果 Redis 是以非守护进程模式的运行，又没有配置指定的 pid 文件，那么不会创建 pid 文件。如果 Redis 是守护进程的模式，即使没有配置指定的 pid 文件，会默认使用“/var/run/redis.pid”文件。

最好创建一个 pid 文件（Creating a pid file is best effort）：没有创建 pid 文件不会有任何影响，Server 还是会正常运行。

pidfile /var/run/redis_6379.pid

指定 Server 的日志级别（Specify the server **verbosity** level）。
有以下四种级别：

- debug（包含许多具体信息，开发/测试 环境下很方便）
- verbose（包含许多不常用的信息，但没有 debug 级别那么混乱）
- notice（moderately verbose，不多不少，很适合生产环境）
- warning（只记录重要或者非常的信息）

loglevel notice

指定 log 文件名。配置成空串的话可以强制 Redis 在标准输出记录日志。注意事项：如果你使用标准输出进行日志记录且是以 守护进程 的模式运行，日志会在 /dev/null 中。

logfile ""

想让日志记录到系统日志，设置 'syslog-enabled' 成 yes，使用 syslog 带有的其他配置选项来满足你的需求。

#syslog-enabled no

指定 syslog 的身份。

#syslog-ident redis

指定 syslog 工具（facility）。一定要是 USER 或者在 LOCAL0-LOCAL7 之间。

#syslog-facility local0

设置数据库的号码。默认的数据库号是 DB 0，你在每个连接中，通过 SELECT <dbid>，选择一个 0~databases-1 的数来配置特定的数据库号。

databases 16

Redis 会在启动的时候，如果标准输出日志是 TTY，则会在开始记录标准输出日志的时候展示一个 ASCII 字符组成的 Redis logo。也就是说，通常只在交互的会话中会展示该 logo。

always-show-logo yes

SNAPSHOTTING（快照）

在硬盘保存数据库：

#save <seconds> <changes>，如果 seconds 和 写操作都配置了，那么一旦达到了配置条件 Redis 会将 DB 保存到硬盘。

以本配置文件的默认配置举例，达到了以下条件会触发写磁盘：
900 秒内（15 分钟）且数据库中至少有一个 key 被改变。
300 秒内（5 分钟）且数据库中至少有10 个 key 被改变。
60 秒内 且数据库中只有一个 10000 个 key 被改变。

提醒：你可以通过注释以下所有的 save 配置行以取消该功能。

也可以通过添加一个带空串的 save 指令来让配置的 save 选择失效。比如：
save ""

save 900 1
save 300 10

save 60 10000

在开启了 RDB 快照后，如果最近的一次 RDB 快照在后台生成失败的话，Redis 默认会拒绝所有的写请求。这么做的目的是为了让用户注意到后台持久化可能出现了问题。否则用户可能一直无法注意到问题，进而可能导致灾难级别的事情发生。

如果后台存储（bgsave）能继续顺利工作，Redis 会自动的继续处理写请求。

但是，如果你已经为你的 Redis 实例和持久化配置了合适的监控手段，且希望 Redis 在非理想情况下（比如硬盘问题，权限问题等等）仍继续提供服务，可以将此项配置为 no。

stop-writes-on-bgsave-error yes

想要在生成 rdb 文件的时候使用 LZF 压缩 String 对象?
将该配置保持默认为 'yes' 几乎不会出现意外状况。（it's almost alwats a win）
可以将该配置设置为“no”来节省 CPU 开销。但是那些原本可以被压缩的 key 和 value 会让数据集更大。

rdbcompression yes

从 5.0 版本开始 RDB 文件的末尾会默认放置一个 CRC64 的校验码。
这会让文件的格式更加容易检验验证，代价是生成和加载 RDB 文件的性能会损失 10% 左右。你可以把该配置关闭以求更佳的性能。

没有开启校验码配置的 RDB 文件会将校验码设置为 0，加载该文件的程序就会跳过校验过程。

rdbchecksum yes

配置 rdb 文件的名称。

dbfilename dump.rdb

存储 rdb 文件的目录。

数据库会使用该配置放置 rdb 文件，文件的名字使用上面的 'dbfilename' 指定的文件名。

AOF 文件的存储位置也会使用这个配置项。

注意：配置一个目录而不是文件名。

dir ./

REPLICATION（复制）

主从复制。使用 replicaof 来让一个 Redis 实例复制另一个 Redis 实例。接来下是关于 Redis 复制需要了解的一些事情。

1）Redis 复制时异步进行的，但是可以通过配置让 Redis 主节点拒绝写请求：配置会给定一个值，主节点至少需要和大于该值的从节点个数成功连接。
2）如果 Redis 从节点和主节点意外断连了很少的一段时间，从节点可以向主节点进行**增量复制**。你可以根据你的需要配置复制的备份日志文件大小（在下一部分可以看到相关的配置）
3）复制会自动进行且不需要人为介入（intervention）。在网络划分后复制会自动与主节点重连且同步数据。

#replicaof <masterip> <masterport>

如果主节点配置了密码（使用了 ”requirepass” 配置项），从节点需要进行密码认证才能进行复制同步的过程，否则主节点会直接拒绝从节点的复制请求。

#masterauth <master-password>

当复制过程与主节点失去连接，或者当复制正在进行时，复制可以有两种行为模式：

1）如果 replica-serve-stale-data 设置为 ’yes’（默认设置），从节点仍可以处理客户端请求，但该从节点的数据很可能和主节点不同步，从节点的数据也可能是空数据集，如果这是与主节点进行的第一次同步。

2）如果 replica-serve-stale-data 设置成 ’no’，从节点会对除了 INFO，replicaOF，AUTH，PING，SHUTDOWN，REPLCONF，ROLE，CONFIG，SUBSCRIBE，UNSUBSCRIBE，PSUBSCRIBE，PUNSUBSCRIBE，PUBLISH，PUBSUB，COMMAND，POST，HOST： and LATENCY 这些命令之外的请求均返回 ”SYNC with master in process”。

replica-serve-stale-data yes

可以配置从节点是否可以处理写请求。针对从节点开启写权限来存储时效低的（ephemeral）数据可能是一种有效的方式（因为写入到从节点的数据很可能随着重新同步而被删除），但是开启该配置也会导致一些问题。

从 Redis 2.6 开始从节点默认是仅可读的。

提示：可读的从节点一般不会暴露给网络中不信任的客户端。这仅是针对不正确使用实例的一层保护。从节点默认仍会响应管理层级的命令，比如 CONFIG，DEBUG 等等。在一定程度上可以使用 ’rename-command’ 避免那些 管理/危险 的命令，提高安全性（To a limited

extent you can improve security of read only replicas using 'rename-command' to shadow all the administrative / dangerous commands）。

replica-read-only yes

同步复制策略：硬盘或者套接字。

警告：不使用硬盘的复制策略目前还在实验阶段

新建立连接和重连的副本不会根据数据情况进行恢复传输，只会进行全量复制。主节点会传输在从节点之间传输 RDB 文件。传输行为有两种方式：

1）硬盘备份：Redis 主节点创建一个子进程来向硬盘写 RDB 文件。之后由父进程持续的文件传给副本。
2）不使用硬盘：Redis 主节点建立一个进程直接向副本的网络套接字写 RDB 文件，不涉及到硬盘。

对于方式 1，在生成 RDB 文件时，多个副本会进行入队并在当前子进程完成 RDB 文件时立即为副本进行 RDB 传输。
对于方式 2，一旦传输开始，新来的副本传输请求会入队且只在当前的传输断开后才建立新的传输连接。

如果使用方式 2，主节点会等待一段时间，根据具体的配置，等待是为了可以在开始传输前可以有期望的副本同步请求到达，这样可以使用并行传输提高效率。

对于配置是比较慢的硬盘，而网络很快（带宽大）的情况下，使用方式 2 进行副本同步会更适合。

repl-diskless-sync no

如果 diskless sync 是开启的话，就需要配置一个延迟的秒数，这样可以服务更多通过 socket 传输 RDB 文件的副本。

这个配置很主要，因为一旦传输开始，就不能为新来的副本传输服务，只能入队等待下一次 RDB 传输，所以该配置一个延迟的值就是为了让更多的副本请求到达。

延迟配置的单位是秒，默认是 5 秒。不想要该延迟的话可以配置为 0 秒，传输就会立即开始。

repl-diskless-sync-delay 5

副本会根据配置好的时间间隔（interval）想主节点发送 PING 命令。可以通过 repl_ping_replica_period 配置修改时间间隔。默认为 10 秒。

#repl-ping-replica-period 10

下面的配置会将副本进行超时处理，为了：

1）在副本的角度，在同步过程中批量进行 I/O 传输。

2）从副本s的角度，主节点超时了。
3）从主节点的角度，副本超时了。

需要重视的一点是确保该选项的配置比 repl-ping-replica-period 配置的值更高，否则每次主从之间的网络比较拥挤时就容易被判定为超时。

#repl-timeout 60