Redis config

#CONNECTION NETWORK

bind 0.0.0.0

port 6400

tcp-backlog 511 #用于控制 Redis 服务器能够同时处理的未完成 TCP 连接数。其实就是等待队列，可以参照内核参数somaxconn 但不大于此参数，默认值为511

timeout 0 #⼀个空闲的客户端维持多少秒会关闭，0表示关闭该功能。即永不关闭

tcp-keepalive 60 #客户端连接心跳检测

#general

daemonize yes

loglevel notice #指定⽇志记录级别。Redis总共⽀持四个级别：debug、verbose、notice、warning，默认为notice。⽣产环境选择notice或者warning

databases 16

dir "/data/redis/6400"

logfile "/data/redis/6400/redis\_6400.log"

pidfile /data/redis/6400/redis\_6400.pid #存放pid⽂件的位置，每个实例会产⽣⼀个不同的pid⽂件,在多实例⽬录下

#security

protected-mode yes

requirepass "123456"

rename-command #⽣产上我们⼀般都会禁⽌使⽤，我们通过rename-command FLUSHALL "" #将清空db进⾏屏蔽 rename-command KEYS "" #将keys命令屏蔽

如果需要保留命令，我们可以重命名成⼀个只有我们知道的命令

#limit

maxclients 4064

maxmemory 100mb

maxmemory-policy volatile-lru  
- volatile-lru：使⽤LRU算法移除key，只对设置了过期时间的键；（最近最少使⽤）

- allkeys-lru：在所有集合key中，使⽤LRU算法移除key

- volatile-random：在过期集合中移除随机的key，只对设置了过期时间的键

- allkeys-random：在所有集合key中，移除随机的key

- volatile-ttl：移除那些TTL值最⼩的key，即那些最近要过期的key

- noeviction：不进⾏移除。针对写操作，只是返回错误信息

maxmemory-samples

- 设置样本数量，LRU算法和最⼩TTL算法都并⾮是精确的算法，⽽是估算值，所以你可以设置样本的⼤⼩，redis默认会检查这么多个key并选择其中LRU的那个。

- ⼀般设置3到7的数字，数值越⼩样本越不准确，但性能消耗越⼩。

# rdb 持久化相关

1.save

触发redis持久化的条件，将内存中的数据保存到磁盘中，默认配置如下：

- save 900 1：当900秒内，⾄少有1个key的值发⽣变化，则保存到磁盘中

- save 300 10：当300秒内，⾄少有10个key的值发⽣变化，则保存到磁盘中

- save 60 10000：当60秒内，⾄少有10000个key的值发⽣变化，则保存到磁盘中

如果不需要做持久化，仅使⽤缓存功能，则可以将持久化条件全部注释，且设置为save ""

2.rdbcompression #是否开启RDB⽂件压缩，默认为yes，表示开启。如果为yes，则redis采⽤LZF算法进⾏压

缩；如果不想消耗cpu来进⾏压缩，可以设置为no，但存储在磁盘中的快照会⽐较⼤

3.dbfilename #RDB⽂件名，默认为dump.rdb。

4.dir #RDB⽂件的保存路径，默认为Redis的安装⽬录。

# aof相关

1.appendonly

是否开启AOF持久化，默认为no。

redis默认使⽤RDB⽅式持久化。如果设置为yes，redis会把每次写⼊的数据接收后都写⼊appendonly.aof⽂件中，每次启动redis会先从appendonly.aof⽂件中将数据读取到内存中，忽略RDB⽂件

2.appendfilename

AOF⽂件名，默认为appendonly.aof。

3.appendfsync

AOF持久化策略，默认为everysec，表示每秒同步⼀次。有以下选项

- no：表示不执⾏fsync命令，由操作系统保证将内存中的数据同步到磁盘中，速度最快

- always：表示每次写⼊都执⾏fsync命令，将内存中的数据同步到磁盘中

- everysec：表示每秒执⾏⼀次fsync命令，可能会导致丢失这1s数据（如果执⾏fsync命令失败），默认项

4.no-appendfsync-on-rewrite

在重写⼊aof或rdb⽂件时会造成阻塞，会执⾏⼤量IO，如果对延迟要求较⾼，则设置为yes，表示rewrite期间对新

写操作不执⾏fsync命令，暂存于内存中，等rewrite完成后再写⼊，默认值为no，建议为yes。

5.auto-aof-rewrite-percentage

默认值为100。aof⾃动重写配置，当前aof⽂件⼤⼩超过上⼀次重写的aof⽂件的百分之多少进⾏重写，redis执⾏

bgrewriteaof对⽇志⽂件进⾏重写。

6.auto-aof-rewrite-min-size

⽤于配置重写AOF⽂件所需的最⼩⽂件⼤⼩,默认为64mb。设置允许重写的最⼩aof⽂件⼤⼩，避免达到了约定100%⽂

件任然很⼩的情况还要重写

7.aof-load-truncated：如果redis宕机，在恢复启动后，可能aof⽂件尾部是不完整的。默认值为yes，aof⽇志

⽂件会继续使⽤，如果设置为no，则恢复失败，需要⽤户⼿动 redis-check-aof 修复AOF⽂件才可使⽤

# replication

1. slaveof <masterip> <masterport> #添加从节点

设置Redis作为从节点时连接的主节点的IP地址和端⼝号，格式为slaveof <masterip> <masterport>。

2.masterauth

当Redis作为从节点时，连接主节点的密码认证。

1. slave-serve-stale-data yes
2. slave-read-only yes
3. repl-diskless-sync no #⽆盘复制适⽤于主节点所在机器磁盘性 能较差但⽹络带宽较充裕的场景\*/
4. repl-diskless-sync-delay 5 #两次diskless模式的数据同步操作的时间间隔
5. repl-ping-slave-period 10 #Slave节点向Master节点发送ping指令的事件间隔,10s
6. repl-timeout 60 #Master和Slave之间的超时时间
7. repl-disable-tcp-nodelay no #主从复制时使⽤的⽹络资源优化参数
   1. 默认关闭
   2. 关闭:所有命令数据发送从节点,延迟变少,但是⽹络带宽消耗增加。适合同机房
   3. 开启:合并较⼩tcp数据包,默认40ms，节省带宽增⼤主从延时，适合跨机房部署
8. repl-backlog-size 1MB #主节点复制积压缓冲区⼤⼩
9. slave-priority #当前Slave节点的优先级权重
10. min-slaves-to-write #拒绝数据写操作的策略,用于指定在将写操作传播到从节点之前，至少需要多少个从节点成功同步数据。min-slaves-to-write 的取值范围是 0 到 N，其中 N 是从节点的数量。 默认情况下，min-slaves-to-write 的值为 0，表示只要主节点成功执行写操作，就会立即将写操作传播到从节点。
11. min-slaves-max-lag #用于指定主节点与从节点之间允许的最大延迟。min-slaves-max-lag 的取值范围是 0 到 N，其中 N 是秒数。 默认情况下，min-slaves-max-lag 的值为 10，表示主节点与从节点之间允许的最大延迟为 10 秒。min-slaves-max-lag 的作用是：防止主节点与从节点之间的数据差异过大。当主节点与从节点之间的延迟超过 min-slaves-max-lag 时，主节点会拒绝执行写操作。可以提高数据的安全性，但会降低写操作的性能。

# cluster

1. cluster-enabled

redis集群开关

2.cluster-config-file

redis集群配置⽂件，每个节点都有⼀个集群配置⽂件

3.cluster-node-timeout

集群节点互联超时毫秒数，默认为15000毫秒

4. cluster-slave-validity-factor #⽤来判断slave节点与master节点断线的时间是否过⻓

5. cluster-migration-barrier

master节点需要的最⼩slave节点数，只有达到这个数，master节点失败时，slave节点才能迁移到其他master节点上

6.cluster-require-full-coverage

默认情况下，集群全部的slot有节点负责，集群状态才为ok，才能提供服务。 设置为no，可以在slot没有全部分配

的时候提供服务。不建议打开该配置，这样会造成分区的时候，⼩分区的master⼀直在接受写请求，⽽造成很⻓时间数据不⼀致