

**易云redis服务安装配置手册**

**V1.0**

**管理部门：\_\_\_**技术中心**\_\_\_**

**撰 写 人：\_\_\_**\_\_\_\_\_\_**\_\_\_\_\_\_**

**审 核 人：\_\_\_\_**\_\_\_\_\_\_**\_\_\_\_\_**

**日 期：\_\_\_**\_\_\_\_\_\_**\_\_\_\_\_\_**

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **内容** | **修订时间** | **制定人** |
| **V 1.0** | **最初版本制定** | **2016-8-2** |  |
| **V 2.0** |  |  |  |
|  |  |  |  |

目录

[1. 环境准备 1](#_Toc457907706)

[1.1. 所需软件 1](#_Toc457907707)

[1.2. 目录规划 1](#_Toc457907708)

[1.3. 安装准备 1](#_Toc457907709)

[1. 安装 redis-3.2.0 2](#_Toc457907710)

[2. 配置redis 2](#_Toc457907711)

[3. 手动启动/停止redis服务 5](#_Toc457907712)

[4. redis启动脚本 6](#_Toc457907713)

[5. 启动/停止redis服务 9](#_Toc457907714)

[6. 环境变量配置 9](#_Toc457907715)

[7. 测试redis 9](#_Toc457907716)

[7.1. 带密码登录 9](#_Toc457907717)

[7.2. 增删改查测试 10](#_Toc457907718)

[8. 启动多个redis实例 10](#_Toc457907719)

[附录：redis.conf 配置文件说明 10](#_Toc457907720)

# 环境准备

操作系统及版本：CentOS6.7\_x64

## 所需软件

* redis-3.2.0.tar.gz
* tcl-8.5.7-6.el6.x86\_64.rpm

## 目录规划

|  |  |
| --- | --- |
| **目录路径及名称** | **目录说明** |
| /usr/local/src | 软件存放目录 |
| /usr/local/redis-3.2.0 | redis安装目录 |
| /usr/local/redis | redis工作目录（安装目录的软连接） |
| /etc/redis/redis-6379.conf | redis配置文件存放，配置文件命名：redis-6379.conf |
| /usr/local/redis/6379/data | redis持久化数据存储目录 |
| /usr/local/redis/redis-6379.log | redis日志 |
| /var/run/redis\_6379.pid | redis PID目录（默认） |

## 安装准备

* 编译环境及工具安装：

yum -y install gcc gcc-\* automake autoconf libtool make zlib zlib-devel glibc-\* tcl tree

* 创建安装目录：

mkdir /usr/local/redis-3.2.0

* 创建软连接：

ln -s /usr/local/redis-3.2.0 /usr/local/redis

* 软件存放目录：

mkdir /usr/local/src

wget http://download.redis.io/releases/redis-3.2.0.tar.gz -P /usr/local/src

1. 安装 redis-3.2.0

cd /usr/local/src

tar xzf redis-3.2.0.tar.gz

cd redis-3.2.0

make test

make MALLOC=libc

make PREFIX=/usr/local/redis install

1. 配置redis

创建配置文件存放目录、持久化文件存储目录：

mkdir /etc/redis /usr/local/redis/6379/data

配置文件创建：

cp /usr/local/src/redis-3.2.0/redis.conf /etc/redis/redis-6379.conf

配置文件修改：

vim /etc/redis/redis-6379.conf

修改如下参数值：

bind 0.0.0.0

timeout 30

daemonize yes

pidfile /var/run/redis\_6379.pid

logfile "/usr/local/redis/redis-6379.log"

dir /usr/local/redis/6379/data

requirepass 123456

maxclients 10000

maxmemory 500mb

修改后配置如下：

[root@demo1 redis]# grep -v "^#" /etc/redis/redis-6379.conf |grep -v "^$"

bind 0.0.0.0

protected-mode yes

port 6379

tcp-backlog 511

timeout 30

tcp-keepalive 0

daemonize yes

supervised no

pidfile /var/run/redis\_6379.pid

loglevel notice

logfile "/usr/local/redis/redis-6379.log"

databases 16

save 900 1

save 300 10

save 60 10000

stop-writes-on-bgsave-error yes

rdbcompression yes

rdbchecksum yes

dbfilename dump.rdb

dir /usr/local/redis/6379/data

slave-serve-stale-data yes

slave-read-only yes

repl-diskless-sync no

repl-diskless-sync-delay 5

repl-disable-tcp-nodelay no

slave-priority 100

requirepass 123456

maxclients 10000

maxmemory 500mb

appendonly no

appendfilename "appendonly.aof"

appendfsync everysec

no-appendfsync-on-rewrite no

auto-aof-rewrite-percentage 100

auto-aof-rewrite-min-size 64mb

aof-load-truncated yes

lua-time-limit 5000

slowlog-log-slower-than 10000

slowlog-max-len 128

latency-monitor-threshold 0

notify-keyspace-events ""

hash-max-ziplist-entries 512

hash-max-ziplist-value 64

list-max-ziplist-size -2

list-compress-depth 0

set-max-intset-entries 512

zset-max-ziplist-entries 128

zset-max-ziplist-value 64

hll-sparse-max-bytes 3000

activerehashing yes

client-output-buffer-limit normal 0 0 0

client-output-buffer-limit slave 256mb 64mb 60

client-output-buffer-limit pubsub 32mb 8mb 60

hz 10

aof-rewrite-incremental-fsync yes

redis3.2 版本新增默认参数：

protected-mode yes

supervised no

list-max-ziplist-size -2

list-compress-depth 0

aof-rewrite-incremental-fsync yes

1. 手动启动/停止redis服务

cd /usr/local/redis/bin

./redis-server /etc/redis/redis-6379.conf

./redis-server --port 6379

**控制台出现如下提示信息：**

[31726] 16 Jan 15:30:32.946 # Unable to set the max number of files limit to 10032 (Operation not permitted), setting the max clients configuration to 992.

\_.\_

\_.-``\_\_ ''-.\_

\_.-`` `. `\_. ''-.\_ Redis 3.2.0 (00000000/0) 64 bit

.-`` .-```. ```\/ \_.,\_ ''-.\_

( ' , .-` | `, ) Running in stand alone mode

|`-.\_`-...-` \_\_...-.``-.\_|'` \_.-'| Port: 6379

| `-.\_ `.\_ / \_.-' | PID: 31726

`-.\_ `-.\_ `-./ \_.-' \_.-'

|`-.\_`-.\_ `-.\_\_.-' \_.-'\_.-'|

| `-.\_`-.\_ \_.-'\_.-' | http://redis.io

`-.\_ `-.\_`-.\_\_.-'\_.-' \_.-'

|`-.\_`-.\_ `-.\_\_.-' \_.-'\_.-'|

| `-.\_`-.\_ \_.-'\_.-' |

`-.\_ `-.\_`-.\_\_.-'\_.-' \_.-'

`-.\_ `-.\_\_.-' \_.-'

`-.\_ \_.-'

`-.\_\_.-'

[31726] 16 Jan 15:30:32.955 # Server started, Redis version 2.8.4

[31726] 16 Jan 15:30:32.955 \* The server is now ready to accept connections on port 6379

**停止redis服务**

redis-cli -p 7030 shutdown

**如果设置了密码123456，需用下面的命令停止：**

redis-cli -p 6379 -a 123456 shutdown

1. redis启动脚本

复制脚本到目录： /etc/rc.d/init.d/

cp redis-3.2.0/utils/redis\_init\_script /etc/init.d/redis-6379

（redis-3.2.0/utils/redis\_init\_script，在redis源码目录）

将redis\_init\_script复制到/etc/init.d/redis-6379，同时命名为redis-6379。

添加注册服务：

chkconfig --add redis-6379

注意：注册系统服务时报错及解决：

redis 服务不支持 chkconfig

解决方法：需要更改redis脚本：

vim /etc/init.d/redis-6379

#!/bin/sh

# chkconfig: 2345 80 90

# Simple Redis init.d script conceived to work on Linux systems

# as it does use of the /proc filesystem.

REDISPORT=6379

EXEC=/usr/local/redis/bin/redis-server

CLIEXEC=/usr/local/redis/bin/redis-cli

PIDFILE=/var/run/redis\_${REDISPORT}.pid

CONF="/etc/redis/redis-${REDISPORT}.conf"

case "$1" in

start)

if [ -f $PIDFILE ]

then

echo "$PIDFILE exists, process is already running or crashed"

else

echo "Starting Redis server..."

$EXEC $CONF &

fi

if [ "$?" = "0" ]

then

echo "Redis is running..."

fi

;;

stop)

if [ ! -f $PIDFILE ]

then

echo "$PIDFILE does not exist, process is not running"

else

PID=$(cat $PIDFILE)

echo "Stopping ..."

$CLIEXEC -p $REDISPORT -a "123456" shutdown

while [ -x /proc/${PID} ]

do

echo "Waiting for Redis to shutdown ..."

sleep 1

done

echo "Redis stopped"

fi

;;

restart|force-reload)

${0} stop

${0} start

;;

\*)

echo "Usage: /etc/init.d/redis-${REDISPORT} {start|stop|restart|force-reload}" >&2

;;

esac

**说明：**

1. 配置文件中也要指定redis运行的端口号为6379，以免启动多个redis实例时，产生混乱。
2. 每一个启动/重启脚本对应一个redis配置文件；
3. 每个redis.conf中的配置（端口，pid，日志路径/名称，数据保存路径等）都不能一样，以免产生数据覆盖，导致数据丢失或混乱。

**与原文件相比，修改的部分如下：**

1. **在第二行增加了：**

#chkconfig: 2345 80 90

1. **原文件EXEC、CLIEXEC参数**

EXEC=/usr/local/redis/bin/redis-server

CLIEXEC=/usr/local/redis/bin/redis-cli

1. **启动redis在后台运行。（若已配置了daemonize yes，则该步骤可以不配置）**

$EXEC $CONF &

ps:注意后面的那个"&"，即是将服务转到后面运行的意思，否则启动服务时，Redis服务将占据在前台，占用了主用户界面，造成其它的命令执行不了。

1. **将redis配置文件拷贝到/etc/redis/redis-${REDISPORT}.conf**
2. **启用认证**

默认情况下，Redis未启用认证，可以通过开启 redis-6379.conf 的 requirepass 指定一个验证密码，如123456（实际情况建议设置稍微复杂点）。

以上操作完成后，即可注册redis服务：

chkconfig --add redis

1. 启动/停止redis服务

service redis-6379 start

service redis-6379 stop

1. 环境变量配置

将redis的命令所在目录添加到系统参数PATH中

修改profile文件：

vi /etc/profile

在最后行追加:

export PATH="$PATH:/usr/local/redis/bin"

应用这个文件：

. /etc/profile

这样就可以直接调用redis-cli的命令了，如下所示：

$ redis-cli

redis 127.0.0.1:6379> auth superman

OK

redis 127.0.0.1:6379> ping

PONG

redis 127.0.0.1:6379>

1. 测试redis
   1. 带密码登录

redis-cli -h 192.168.71.128 -p 6379 -a 123456

或者：进入redis安装目录的bin目录下执行如下命令：

./redis-cli -h 127.0.0.1 -p 6379 -a 123456

127.0.0.1:6379> ping

PONG

127.0.0.1:6379> set name miao

OK

127.0.0.1:6379> get name

"miao "

* 1. 增删改查测试

# /usr/redis-2.8.17/src/redis-cli

新增

redis> set foo bar

获取

redis> get foo"bar"

redis>keys f\*

redis>keys f?o?

删除

redis>del foo

1. 启动多个redis实例

拷贝默认的redis.conf 改为redis-6380.conf，打开redis-6380.conf配置文件，找到 port 6379 这行，把 6379 改为 6380，将 pid、logfile、dumpdir 等配置参考上述配置完毕后，用如下命令启动多个服务。

# redis-server /etc/redis/redis-6379.conf

# redis-server /etc/redis/redis-6380.conf

附录：redis.conf 配置文件说明

# By default Redis does not run as a daemon. Use 'yes' if you need it.

# Note that Redis will write a pid file in /var/run/redis.pid when daemonized.

# 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

daemonize yes

# When running daemonized, Redis writes a pid file in /var/run/redis.pid by

# default. You can specify a custom pid file location here.

# 当 Redis 以守护进程的方式运行的时候,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\_6379.pid

# Accept connections on the specified port, default is 6379.

# If port 0 is specified Redis will not listen on a TCP socket.

# 端口没什么好说的

port 6379

# If you want you can bind a single interface, if the bind option is not

# specified all the interfaces will listen for incoming connections.

# 指定Redis可接收请求的IP地址,不设置将处理所有请求,建议生产环境中设置

# bind 127.0.0.1

# Close the connection after a client is idle for N seconds (0 to disable)

# 客户端连接的超时时间,单位为秒,超时后会关闭连接

timeout 30

# Specify the log file name. Also 'stdout' can be used to force

# Redis to log on the standard output. Note that if you use standard

# output for logging but daemonize, logs will be sent to /dev/null

# 配置 log 文件地址,默认打印在命令行终端的窗口上

# logfile stdout

logfile "/usr/local/redis/redis-6379.log"

# Set the number of databases. The default database is DB 0, you can select

# a different one on a per-connection basis using SELECT <dbid> where

# dbid is a number between 0 and 'databases'-1

# 设置数据库的个数,可以使用 SELECT <dbid>命令来切换数据库。默认使用的数据库是 0

databases 16

#

# 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.

#

# In the example below the behaviour will be to save:

# after 900 sec (15 min) if at least 1 key changed

# after 300 sec (5 min) if at least 10 keys changed

# after 60 sec if at least 10000 keys changed

#

# Note: you can disable saving at all commenting all the "save" lines.

#设置 Redis 进行数据库镜像的频率。

#900秒之内有1个keys发生变化时

#30秒之内有10个keys发生变化时

#60秒之内有10000个keys发生变化时

save 900 1

save 300 10

save 60 10000

# Compress string objects using LZF when dump .rdb databases?

# For default that's set to 'yes' as it's almost always a win.

# If you want to save some CPU in the saving child set it to 'no' but

# the dataset will likely be bigger if you have compressible values or keys.

#在进行镜像备份时,是否进行压缩

rdbcompression yes

# The filename where to dump the DB

#镜像备份文件的文件名

dbfilename dump.rdb

# The working directory.

#

# The DB will be written inside this directory, with the filename specified

# above using the 'dbfilename' configuration directive.

#

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

#

# Note that you must specify a directory here, not a file name.

#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

dir /usr/local/redis/6379/data

# Master-Slave replication. Use slaveof to make a Redis instance a copy of

# another Redis server. Note that the configuration is local to the slave

# 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.

# 设置该数据库为其他数据库的从数据库，当本机为从服务时，设置主服务的IP及端口

# slaveof <masterip> <masterport>

# If the master is password protected (using the "requirepass" configuration

# directive below) it is possible to tell the slave to authenticate before

# starting the replication synchronization process, otherwise the master will

# refuse the slave request.

#指定与主数据库连接时需要的密码验证，当本机为从服务时，设置主服务的连接密码

# masterauth <master-password>

# Require clients to issue AUTH <PASSWORD> before processing any other

# commands. This might be useful in environments in which you do not trust

# others with access to the host running redis-server.

#

# This should stay commented out for backward compatibility and because most

# people do not need auth (e.g. they run their own servers).

#

# 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.

#设置客户端连接后进行任何其他指定前需要使用的密码。

警告:redis速度相当快,一个外部的用户可以在一秒钟进行150K次的密码尝试,你需要指定非常非常强大的密码来防止暴力破解。

# requirepass foobared

requirepass 123456

# Set the max number of connected clients at the same time. By default there

# is no limit, and it's up to the number of file descriptors the Redis process

# is able to open. The special value '0' means no limits.

# Once the limit is reached Redis will close all the new connections sending

# an error 'max number of clients reached'.

#限制同时连接的客户数量。当连接数超过这个值时,redis 将不再接收其他连接请求,客户端尝试连接时将收到 error 信息

# maxclients 128

maxclients 10000

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

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

# accordingly to the eviction policy selected (see maxmemmory-policy).

#

# If Redis can't remove keys according to the policy, or if the policy is

# set to 'noeviction', Redis will start to reply with errors to commands

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

# to reply to read-only commands like GET.

#

# This option is usually useful when using Redis as an LRU cache, or to set

# an hard memory limit for an instance (using the 'noeviction' policy).

#

# WARNING: If you have slaves attached to an instance with maxmemory on,

# the size of the output buffers needed to feed the slaves are subtracted

# from the used memory count, so that network problems / resyncs will

# not trigger a loop where keys are evicted, and in turn the output

# buffer of slaves is full with DELs of keys evicted triggering the deletion

# of more keys, and so forth until the database is completely emptied.

#

# In short… if you have slaves attached it is suggested that you set a lower

# limit for maxmemory so that there is some free RAM on the system for slave

# output buffers (but this is not needed if the policy is 'noeviction').

# 设置redis能够使用的最大内存。当内存满了的时候,如果还接收到set命令,redis将先尝试剔除设置过expire信息的key,而不管该key的过期时间还没有到达。

# 在删除时,将按照过期时间进行删除,最早将要被过期的key将最先被删除。如果带有expire信息的key都删光了,那么将返回错误。

# 这样,redis将不再接收写请求,只接收get请求。maxmemory的设置比较适合于把redis当作于类似memcached 的缓存来使用

# 最大内存使用设置，达到最大内存设置后，Redis会先尝试清除已到期或即将到期的Key，当此方法处理后，任到达最大内存设置，将无法再进行写入操作。

# maxmemory <bytes>

# By default Redis asynchronously dumps the dataset on disk. If you can live

# with the idea that the latest records will be lost if something like a crash

# happens this is the preferred way to run Redis. If instead you care a lot

# about your data and don't want to that a single record can get lost you should

# enable the append only mode: when this mode is enabled Redis will append

# every write operation received in the file appendonly.aof. This file will

# be read on startup in order to rebuild the full dataset in memory.

#

# 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.

#

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

# log file in background when it gets too big.

# 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

appendonly no

# 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.

#

# Redis supports three different modes:

#

# no: don't fsync, just let the OS flush the data when it wants. Faster.

# always: fsync after every write to the append only log . Slow, Safest.

# everysec: fsync only if one second passed since the last fsync. Compromise.

#

# The default is "everysec" that's usually the right compromise between

# speed and data safety. It's up to you to understand if you can relax this to

# "no" that will will let the operating system flush the output buffer when

# it wants, for better performances (but if you can live with the idea of

# some data loss consider the default persistence mode that's snapshotting),

# or on the contrary, use "always" that's very slow but a bit safer than

# everysec.

#

# If unsure, use "everysec".

#设置对 appendonly.aof 文件进行同步的频率。always 表示每次有写操作都进行同步,everysec 表示对写操作进行累积,每秒同步一次。

#是否在每次更新操作后进行日志记录，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。默认值为no

# appendfsync always

appendfsync everysec

# 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.

# redis 2.0 中引入了 hash 数据结构。

# hash 中包含超过指定元素个数并且最大的元素当没有超过临界时,hash 将以zipmap(又称为 small hash大大减少内存使用)来存储,这里可以设置这两个临界值

hash-max-zipmap-entries 512

hash-max-zipmap-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.

#

# use "activerehashing yes" if you don't have such hard requirements but

# want to free memory asap when possible.

# 开启之后,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

# 是否重置Hash表

activerehashing yes

# The client output buffer limits can be used to force disconnection of clients

# that are not reading data from the server fast enough for some reason (a

# common reason is that a Pub/Sub client can't consume messages as fast as the

# publisher can produce them).

#

# The limit can be set differently for the three different classes of clients:

#

# normal -> normal clients including MONITOR clients

# slave -> slave clients

# pubsub -> clients subscribed to at least one pubsub channel or pattern

#

# The syntax of every client-output-buffer-limit directive is the following:

#

# client-output-buffer-limit <class> <hard limit> <soft limit> <soft seconds>

#

# A client is immediately disconnected once the hard limit is reached, or if

# the soft limit is reached and remains reached for the specified number of

# seconds (continuously).

# So for instance if the hard limit is 32 megabytes and the soft limit is

# 16 megabytes / 10 seconds, the client will get disconnected immediately

# if the size of the output buffers reach 32 megabytes, but will also get

# disconnected if the client reaches 16 megabytes and continuously overcomes

# the limit for 10 seconds.

#

# By default normal clients are not limited because they don't receive data

# without asking (in a push way), but just after a request, so only

# asynchronous clients may create a scenario where data is requested faster

# than it can read.

#

# Instead there is a default limit for pubsub and slave clients, since

# subscribers and slaves receive data in a push fashion.

#

# Both the hard or the soft limit can be disabled by setting them to zero.

client-output-buffer-limit normal 0 0 0

client-output-buffer-limit slave 256mb 64mb 60

client-output-buffer-limit pubsub 32mb 8mb 60

# Redis calls an internal function to perform many background tasks, like

# closing connections of clients in timeout, purging expired keys that are

# never requested, and so forth.

#

# Not all tasks are performed with the same frequency, but Redis checks for

# tasks to perform according to the specified "hz" value.

#

# By default "hz" is set to 10. Raising the value will use more CPU when

# Redis is idle, but at the same time will make Redis more responsive when

# there are many keys expiring at the same time, and timeouts may be

# handled with more precision.

#

# The range is between 1 and 500, however a value over 100 is usually not

# a good idea. Most users should use the default of 10 and raise this up to

# 100 only in environments where very low latency is required.

hz 10

# When a child rewrites the AOF file, if the following option is enabled

# the file will be fsync-ed every 32 MB of data generated. This is useful

# in order to commit the file to the disk more incrementally and avoid

# big latency spikes.

aof-rewrite-incremental-fsync yes