**centos8更换国内源（阿里源）**

1.备份旧的配置文件

sudo mv /etc/yum.repos.d/CentOS-Base.repo /etc/yum.repos.d/CentOS-Base.repo.backup

2.安装wget下载工具

sudo yum install -y wget

下载新的 CentOS-Base.repo 到 /etc/yum.repos.d/

sudo wget -O /etc/yum.repos.d/CentOS-Base.repo <https://mirrors.aliyun.com/repo/Centos-8.repo>

并替换部分字段（非阿里云机器需要做）

sudo sed -i -e '/mirrors.cloud.aliyuncs.com/d' -e '/mirrors.aliyuncs.com/d' /etc/yum.repos.d/CentOS-Base.repo

3.生成缓存

yum makecache

# Redis 安装

**下载地址：**<http://redis.io/download>

1. 可以在物理机上下载然后用ftp上传到虚拟机上
2. 也可以直接执行wget [http://download.redis.io/releases/redis-6.2.1.tar.gz](http://download.redis.io/releases/redis-6.0.8.tar.gz)直接下载

# tar xzf redis-6.2.1.tar.gz

# cd redis-6.2.1

安装gcc等必备程序包

# yum install -y gcc gcc-c++ make automake

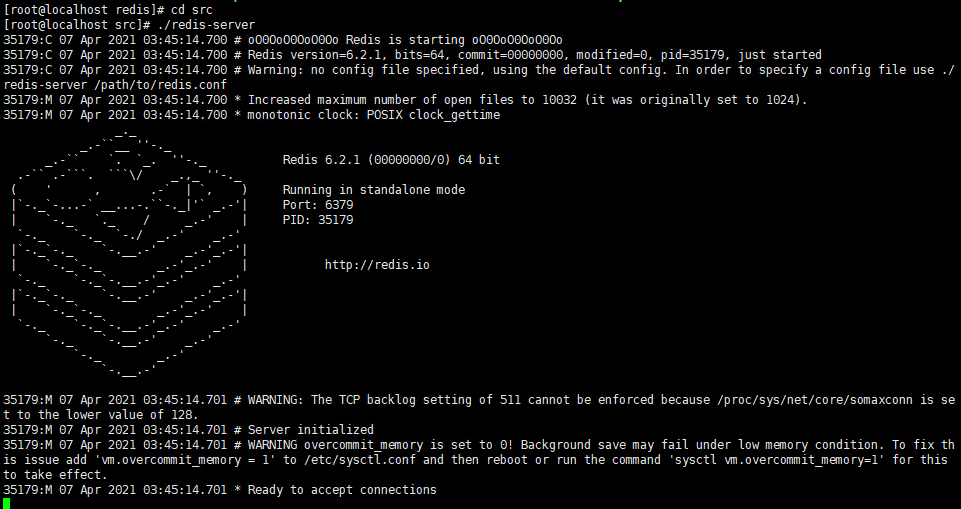
# make

执行完 make 命令后，redis 的 src 目录下会出现编译后的 redis 服务程序 redis-server，还有用于测试的客户端程序 redis-cli：

下面启动 redis 服务：

# cd src

# ./redis-server



注意这种方式启动 redis 使用的是默认配置。也可以通过启动参数告诉 redis 使用指定配置文件使用下面命令启动。

# cd src

# ./redis-server ../redis.conf

**redis.conf** 是一个默认的配置文件。我们可以根据需要使用自己的配置文件。

启动 redis 服务进程后，就可以使用测试客户端程序 redis-cli 和 redis 服务交互了。 比如：

# cd src

# ./redis-cli

redis> set foo bar

OK

redis> get foo

"bar"

# Redis 配置

Redis 的配置文件位于 Redis 安装目录下，文件名为 **redis.conf**

你可以通过 **CONFIG** 命令查看或设置配置项。

**Redis CONFIG 命令格式如下：（复制绿色底纹是命令部分）**

**redis 127.0.0.1:6379> CONFIG GET CONFIG\_SETTING\_NAME**

### 实例

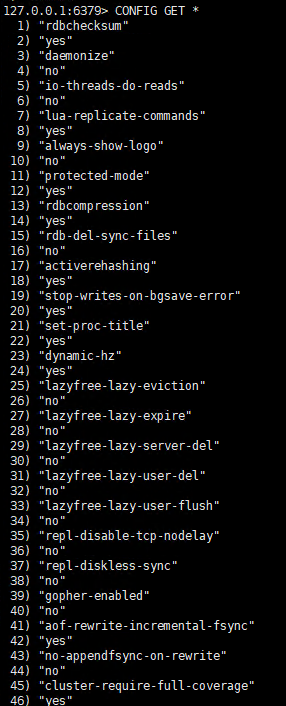
redis 127.0.0.1:6379> CONFIG GET loglevel



使用 **\*** 号获取所有配置项：

### 实例

redis 127.0.0.1:6379> CONFIG GET \*



## 编辑配置

你可以通过修改 redis.conf 文件或使用 **CONFIG set** 命令来修改配置。

### 语法

**CONFIG SET** 命令基本语法：

redis 127.0.0.1:6379> CONFIG SET CONFIG\_SETTING\_NAME NEW\_CONFIG\_VALUE

### 实例

redis 127.0.0.1:6379> CONFIG SET loglevel "notice"

OK

redis 127.0.0.1:6379> CONFIG GET loglevel

1) "loglevel"

2) "notice"

## 参数说明

redis.conf 配置项说明如下：

|  |  |  |
| --- | --- | --- |
| **序号** | **配置项** | **说明** |
| 1 | daemonize no | 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（Windows 不支持守护线程的配置为 no ） |
| 2 | pidfile /var/run/redis.pid | 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 |
| 3 | port 6379 | 指定 Redis 监听端口，默认端口为 6379，作者在自己的一篇博文中解释了为什么选用 6379 作为默认端口，因为 6379 在手机按键上 MERZ 对应的号码，而 MERZ 取自意大利歌女 Alessia Merz 的名字 |
| 4 | bind 127.0.0.1 | 绑定的主机地址 |
| 5 | timeout 300 | 当客户端闲置多长秒后关闭连接，如果指定为 0 ，表示关闭该功能 |
| 6 | 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 |
| 7 | logfile stdout | 日志记录方式，默认为标准输出，如果配置 Redis 为守护进程方式运行，而这里又配置为日志记录方式为标准输出，则日志将会发送给 /dev/null |
| 8 | databases 16 | 设置数据库的数量，默认数据库为0，可以使用SELECT 命令在连接上指定数据库id |
| 9 | save <seconds> <changes>  Redis 默认配置文件中提供了三个条件：  **save 900 1**  **save 300 10**  **save 60 10000**  分别表示 900 秒（15 分钟）内有 1 个更改，300 秒（5 分钟）内有 10 个更改以及 60 秒内有 10000 个更改。 | 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 |
| 10 | rdbcompression yes | 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 |
| 11 | dbfilename dump.rdb | 指定本地数据库文件名，默认值为 dump.rdb |
| 12 | dir ./ | 指定本地数据库存放目录 |
| 13 | slaveof <masterip> <masterport> | 设置当本机为 slave 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 |
| 14 | masterauth <master-password> | 当 master 服务设置了密码保护时，slav 服务连接 master 的密码 |
| 15 | requirepass foobared | 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 |
| 16 | maxclients 128 | 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 |
| 17 | maxmemory <bytes> | 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 |
| 18 | appendonly no | 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 |
| 19 | appendfilename appendonly.aof | 指定更新日志文件名，默认为 appendonly.aof |
| 20 | appendfsync everysec | 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（折中，默认值） |
| 21 | vm-enabled no | 指定是否启用虚拟内存机制，默认值为 no，简单的介绍一下，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（在后面的文章我会仔细分析 Redis 的 VM 机制） |
| 22 | vm-swap-file /tmp/redis.swap | 虚拟内存文件路径，默认值为 /tmp/redis.swap，不可多个 Redis 实例共享 |
| 23 | vm-max-memory 0 | 将所有大于 vm-max-memory 的数据存入虚拟内存，无论 vm-max-memory 设置多小，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 |
| 24 | vm-page-size 32 | 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，作者建议如果存储很多小对象，page 大小最好设置为 32 或者 64bytes；如果存储很大大对象，则可以使用更大的 page，如果不确定，就使用默认值 |
| 25 | vm-pages 134217728 | 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）是在放在内存中的，，在磁盘上每 8 个 pages 将消耗 1byte 的内存。 |
| 26 | vm-max-threads 4 | 设置访问swap文件的线程数,最好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 |
| 27 | glueoutputbuf yes | 设置在向客户端应答时，是否把较小的包合并为一个包发送，默认为开启 |
| 28 | hash-max-zipmap-entries 64  hash-max-zipmap-value 512 | 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 |
| 29 | activerehashing yes | 指定是否激活重置哈希，默认为开启（后面在介绍 Redis 的哈希算法时具体介绍） |
| 30 | include /path/to/local.conf | 指定包含其它的配置文件，可以在同一主机上多个Redis实例之间使用同一份配置文件，而同时各个实例又拥有自己的特定配置文件 |

# Redis 数据类型

Redis支持五种数据类型：string（字符串），hash（哈希），list（列表），set（集合）及zset(sorted set：有序集合)。

## String（字符串）

string 是 redis 最基本的类型，你可以理解成与 Memcached 一模一样的类型，一个 key 对应一个 value。

string 类型是二进制安全的。意思是 redis 的 string 可以包含任何数据。比如jpg图片或者序列化的对象。

string 类型是 Redis 最基本的数据类型，string 类型的值最大能存储 512MB。

### 实例

redis 127.0.0.1:6379> SET runoob "菜鸟教程"

OK

redis 127.0.0.1:6379> GET runoob

"菜鸟教程"

在以上实例中我们使用了 Redis 的 **SET** 和 **GET** 命令。键为 runoob，对应的值为 **菜鸟教程**。

**注意：**一个键最大能存储 512MB。

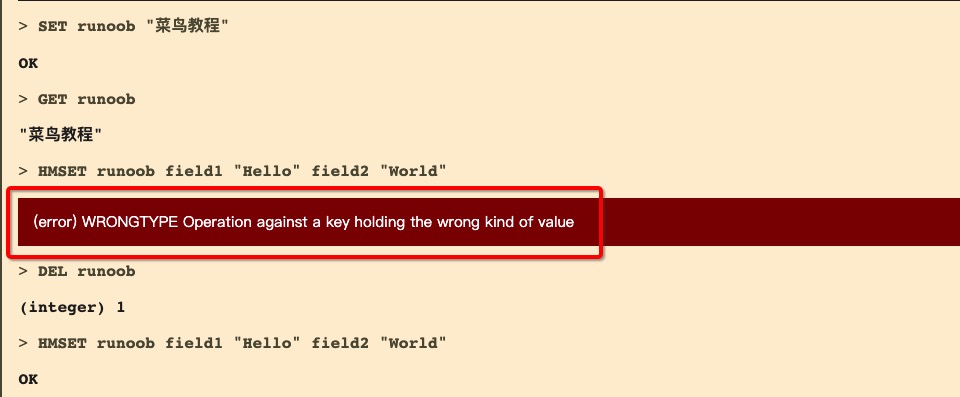
## Hash（哈希）

Redis hash 是一个键值(key=>value)对集合。

Redis hash 是一个 string 类型的 field 和 value 的映射表，hash 特别适合用于存储对象。

### 实例

**DEL runoob** 用于删除前面测试用过的 key，不然会报错：**(error) WRONGTYPE Operation against a key holding the wrong kind of value**



redis 127.0.0.1:6379> DEL runoob

redis 127.0.0.1:6379> HMSET runoob field1 "Hello" field2 "World"

"OK"

redis 127.0.0.1:6379> HGET runoob field1

"Hello"

redis 127.0.0.1:6379> HGET runoob field2

"World"

实例中我们使用了 Redis **HMSET, HGET** 命令，**HMSET** 设置了两个 **field=>value** 对, HGET 获取对应 **field** 对应的 **value**。

每个 hash 可以存储 232 -1 键值对（40多亿）。

## List（列表）

Redis 列表是简单的字符串列表，按照插入顺序排序。你可以添加一个元素到列表的头部（左边）或者尾部（右边）。

### 实例

redis 127.0.0.1:6379> DEL runoob

redis 127.0.0.1:6379> lpush runoob redis

(integer) 1

redis 127.0.0.1:6379> lpush runoob mongodb

(integer) 2

redis 127.0.0.1:6379> lpush runoob rabbitmq

(integer) 3

redis 127.0.0.1:6379> lrange runoob 0 10

1) "rabbitmq"

2) "mongodb"

3) "redis"

redis 127.0.0.1:6379>

列表最多可存储 232 - 1 元素 (4294967295, 每个列表可存储40多亿)。

## Set（集合）

Redis 的 Set 是 string 类型的无序集合。

集合是通过哈希表实现的，所以添加，删除，查找的复杂度都是 O(1)。

### sadd 命令

添加一个 string 元素到 key 对应的 set 集合中，成功返回 1，如果元素已经在集合中返回 0。

sadd key member

### 实例

redis 127.0.0.1:6379> DEL runoob

redis 127.0.0.1:6379> sadd runoob redis

(integer) 1

redis 127.0.0.1:6379> sadd runoob mongodb

(integer) 1

redis 127.0.0.1:6379> sadd runoob rabbitmq

(integer) 1

redis 127.0.0.1:6379> sadd runoob rabbitmq

(integer) 0

redis 127.0.0.1:6379> smembers runoob

1) "redis"

2) "rabbitmq"

3) "mongodb"

**注意：**以上实例中 rabbitmq 添加了两次，但根据集合内元素的唯一性，第二次插入的元素将被忽略。

集合中最大的成员数为 232 - 1(4294967295, 每个集合可存储40多亿个成员)。

## zset(sorted set：有序集合)

Redis zset 和 set 一样也是string类型元素的集合,且不允许重复的成员。

不同的是每个元素都会关联一个double类型的分数。redis正是通过分数来为集合中的成员进行从小到大的排序。

zset的成员是唯一的,但分数(score)却可以重复。

### zadd 命令

添加元素到集合，元素在集合中存在则更新对应score

zadd key score member

### 实例

redis 127.0.0.1:6379> DEL runoob

redis 127.0.0.1:6379> zadd runoob 0 redis

(integer) 1

redis 127.0.0.1:6379> zadd runoob 0 mongodb

(integer) 1

redis 127.0.0.1:6379> zadd runoob 0 rabbitmq

(integer) 1

redis 127.0.0.1:6379> zadd runoob 0 rabbitmq

(integer) 0

redis 127.0.0.1:6379> ZRANGEBYSCORE runoob 0 1000

1) "mongodb"

2) "rabbitmq"

3) "redis"

# Hbase和Redis对比

HBase和Redis的功能上比较相似。都是nosql类型的数据库。但是在适用场景上，两者还是有比较明显的区别的。

下面基于读写性能、数据类型、数据量、部署难易、数据可靠性、应用场景、两者的结合等多个维度来比较一下两者。

**读写性能：**

HBase写快读慢，HBase的读取时长通常是几毫秒，而Redis的读取时长通常是几十微秒。性能相差非常大。

**数据类型：**

HBase和Redis都支持KV类型。但是Redis支持List、Set等更丰富的类型。

**数据量：**

Redis支持的数据量通常受内存限制，而HBase没有这个限制，可以存储远超内存大小的数据。

**部署难易：**

HBase部署需要依赖hadoop、zookeeper等服务，而Redis的部署非常简单。

**数据可靠性：**

HBase采用WAL，先记录日志再写入数据，理论上不会丢失数据。而Redis采用的是异步复制数据，在failover时可能会丢失数据。

**应用场景：**

HBase适合做大数据的持久存储，而Redis比较适合做缓存。如果数据丢失是不能容忍的，那就用只能用HBase；如果需要一个高性能的环境，而且能够容忍一定的数据丢失，那完全可以考虑使用Redis。

**两者的结合：**

HBase可以用来做数据的固化，也就是数据存储，做这个他非常合适。Redis适合做cache。可以用HBase+Redis实现数据仓库加缓存数据库，速度和扩展性都兼顾。

综合上述比较，我们可以看出。这两者都有各自擅长的领域，不存在相互替代。具体选用哪个服务，要根据具体业务场景、数据量选择最合适的方案。