Redis是基于开源的BSD协议使用ANSI C编写，基于内存的且支持持久化，高性能的key-value的NoSQL数据库，支持多种数据结构类型例如字符串（string）、散列（hashes）、列表（lists）、集合（sets）、有序集合（sorted sets）与范围查询、bitmaps、hyperloglogs和地理空间（geospatial）索引半径查询，支持多种语言例如C、C++、Python、Erlang、R、C#、Java、PHP、Objective-C、Perl、Scala、Ruby、Go、JavaScript，Redis用途：缓存（StackOverFlow）、数据库（微博）、消息中间件（微博），官方网站<https://redis.io/>

NoSQL分类：

Key-value存储类型：Redis、Memcached，使用key快速查到其value

文档存储：MongoDB、CouchDB，使用JSON或类JSON的BSON数据结构存储内容为文档型，能实现部分关系型数据库的功能

列存储：HBase、Cassandra，按照列进行数据存储，便于存储结构化和半结构化数据，方便做数据压缩和针对某一列和某几列的数据查询

图存储：Neo4J、FlockDB，图形关系的存储，能够很好弥补关系型数据库在图形存储的不足

对象存储：Db4o、Verant，通过类似面向对象语言的方式操作数据库，通过对象的方式存取数据

XML数据库：Berkeley DB XML、BaseX，高效存储XML数据，支持XML的内部查询语法，如XQuery、XPath

MariaDB与MySQL是同一作者开发，内核操作基本一致，由于MySQL被Oracle收购，部分大型互联网公司以防Oracle收取开源产品MySQL数据库的相关费用开始转用MariaDB

Redis版本号最后一位数字为奇数是实验测试版本，偶数是稳定版本，Redis可视化客户端RedisDesktopManager，Redis的作者是西西里岛的意大利人antirez，现在由Pivotal（EMC和VMware合资的公司）公司出资支持开发

Redis最初是为了解决读写数据受硬盘I/O速度的限制问题开发的内存数据库

Redis附加功能：

持久化功能：将内存数据库中的数据保存到硬盘，保证数据安全，方便进行数据备份和恢复

发布与订阅功能：将消息同时分发给多个客户端，用于构建广播系统

过期键功能：为键设置一个过期时间，指定时间过后自动删除

事务功能：原子的执行多个操作，并提供乐观锁的功能，保证处理数据时的安全性

Lua脚本功能：在服务器端原子的执行多个操作，完成复杂的功能，并减少客户端与服务器之间的通信往返次数

复制：为指定的Redis服务器创建一个或多个复制品，用于提升数据安全性，并分担读请求的负载

Sentinel哨兵：监控Redis服务器的状态，并在服务器发生故障时，进行自动故障转移

集群：创建分布式数据库，每个服务器分别执行一部分写操作和读操作

Redis安装：

安装编译环境yum install gcc tcl -y

解压Redis安装包，在Redis解压包目录下执行make和make PREFIX=安装目录 install进行编译和安装，在指定的安装目录中会产生bin目录，为了方便使用可以将bin目录添加到系统环境变量中或将bin目录下文件复制到/usr/local/bin/目录下

Redis安装目录/bin/redis-server --help 查看帮助

redis-server为二进制执行文件非Shell脚本，不能直接打开查看，可以使用file redis-server查看文件相关信息

Redis安装目录/utils/install\_server.sh 可以将编译的Redis添加到Linux服务中并配置默认文件

Redis安装目录/bin/redis-server Redis的配置文件默认redis.conf 以配置文件方式启动Redis

Redis安装目录/bin/redis-cli -h/c/p/a/n 指定IP地址/集群模式/端口号/密码/数据库号登录Redis，默认登录本地无密码0号数据库Redis

help 命令/@命令/@类型 查看指定命令/相关命令/相关类型帮助

set key value [ex 超时时间单位秒] [px 超时时间单位毫秒] [xx/nx] 设置指定键值，到达指定超时时间键值自动删除，Redis中存在相应键才能设置键值/不存在相应值才能设置，默认键值存在会被覆盖

get key 获取key对应的value值

mset/msetnx key value [key value] 批量设置键值/不存在相应值才能批量设置（原子性不可拆分，批量键值中有一个键已经存在其他键值也不能设置）

mget key [key] 批量获取键值

Redis一般键值的命名规范为key::field value

keys \*/正则表达式 查看所有键/匹配正则表达式的键

getset key value 返回指定key的值并设置新的键值value，内部实现是get和set命令的组合可以减少一次网络传输

append key value 将指定值追加到指定key对应的值的末尾

strlen key 返回键对应字符串值的长度

setrange key 数值 value 设置从指定数值开始替换key对应的值为value，数值索引的取值从左向右替换为0~\*，从右向左为-1~-\*

getrange key 数值 数值 获取指定key对应值从数值到数值范围的值

Redis存储的值为字符串类型，但可以转换为数值型操作后再以字符串类型存储（支持转换64位有符号的十进制整数或者标准64位浮点数，值不能包含字符否则转换为字符串，使用字符串命令会将可以转换为数值类型的字符串转换为字符串操作）

incrby key 数值 将指定key的值增加指定数值，key不存在默认设置key的值为0

decrby key 数值 将指定key的值减少指定数值（支持负数）

incr key 将指定key的值增加1

decr key 将指定key的值减少1

incrbyfloat key 浮点数 将指定key的值增加指定浮点数值，无decrbyfloat命令，可以通过增加负浮点数值实现减少

flushdb 清空当前库所有数据

flushall 清空所有库的所有数据

expire/pexpire key 数值/时间戳 设置已存在键key的值的过期时间为指定数值（单位秒/毫秒）/时间戳

persist key 删除指定key的值的过期时间

ttl/pttl key 查看指定key的值的剩余过期时间（单位秒/毫秒），无过期时间返回-1，key曾经存在返回-2

type key 查看指定key对应值的数据类型

exists key 查看指定key是否存在

rename/renamenx key 键 重命名指定键key为指定键/重命名不存在的key为指定键

del key [key] 删除指定键值

setbit key 数值 value 设置指定key对应的值的二进制第数值位为value，数值从右向左从0开始

getbit key 数值 获取指定key对应值的二进制第数值位的值

bitpos key 0/1 [start] [end] 获取指定key对应值的二进制从start数值开始到end数值结束0/1第一次出现的位置

bitop and/or/xor key [key] 将一个或多个key的值逻辑并/或/异或/非的结果保存到第一个key中

bitop not 键 key 将指定key的值逻辑非的结果保存到指定键中

bitcount key [start] [end] 统计指定key对应值的二进制指定位数区间上的1的个数，从左向右从0开始，从右向左从-1开始，中文字符会随编码格式不同二进制结果不同

Redis的键key为二进制序列，可以为空，key一般不能过长（消耗内存，查找键值计算成本高），也不能过短（可读性差），key的数量一般不能过多（key带有其他附加信息）

Redis的基本类型时字符串也是二进制存储，位图bitmap类型属于字符串类型最多能存储512M字节的数据

Redis的List列表结构是基于Linked List实现，元素类型为字符串，元素值可以重复，最多为2^32-1个元素，列表索引从左向右从0开始，从右向左从-1开始

命令头部B/L/R/X block块阻塞/left左/right右/exist存在

lpush/rpush key value [value] 从左/右追加指定key的List列表指定value的元素

lpop/rpop key 从左/右或者头/尾弹出元素即返回对应元素后从列表中删除

rpoplpush 键 key 从指定键的列表尾部弹出元素加到指定key的列表头部

lrange key start stop 返回指定key的列表指定范围的元素

lindex key 数值 获取指定key的列表的指定数值的元素

lset key index value 设置指定位置元素的值

llen key 获取列表长度即元素个数

lrem key count value 从列表头部开始删除值为value的元素数量count（count为正数从列表头部开始，为负数从尾部开始，数量为count的绝对值，count为0删除所有与value相等的元素）

ltrim key start stop 返回指定列表范围内的元素

linsert key before/after pivot value 在列表指定pivot值前/后插入元素value，key或pivot不存在不操作

B命令开头阻塞，弹出的列表不存在或者为空会阻塞，超时时间设置为0永久阻塞直到有数据弹出，多个阻塞在同一个列表上使用FIFS先到先服务原则

blpop/brpop key [key] timeout 从左/右或者头/尾阻塞弹出元素

brpoplpush source destination timeout 从source类别尾部阻塞弹出元素到destination列表头部

Hash散列由field和关联的value组成的map键值对，field和value为字符串类型，一个Hash散列最多存储2^32-1个键值对

hset/hsetnx key field value 设置指定key为Hash散列名的键值对/ key和field不存在才能设置

hmset key field value [field value] 设置多个Hash散列键值对

hlen key 获取指定key散列的长度

hexists key field 判断指定key散列的field是否存在，key或者field不存在返回0

hget key field 获取指定key散列的field对应的值

hmget key field [field] 获取指定key散列多个field对应的值

hgetall key 获取指定key散列的所有键值对

hkeys key 获取指定key散列的所有field

hvals key 获取指定key散列的所有value

hincrby key field 数值 在key散列field键的值上进行整数增量指定数值

hincrbyfloat key field 浮点数 在key散列field键的值上进行浮点数增量指定浮点数

hdel key field [field] 删除key散列多个field的值

Hash类型存储用途：节约空间，Redis每创建一个键key会附加一些信息（超时时间等），Hash散列的超时时间只能加到key上不能加到键值对上，Hash散列不能使用二进制位操作命令

Set集合：无序，去重，元素为字符串类型，最多2^32-1个元素

sadd/srem key value [value] 指定key的Set集合中加入/删除多个value，value重复自动忽略

smembers key 获取key的Set集合的所有元素，元素过多会发生阻塞

sismember key value 检查key的Set集合中是否存在value元素，由于Set集合无序，返回结果可以顺序不同

srandmember key [count] 随机返回key的Set集合中count绝对值个数的集合或数组，count为正数返回元素不相同的集合，count为负数返回元素可以重复的数组，count为0返回空，count默认为1

scard key 获取key的Set集合的元素个数，命令不会遍历集合，集合的元素个数附加保存在key里

spop key 随机弹出一个元素

smove source destination member 从source的Set集合中移出member元素到destination的Set集合

sdiff/sinter/sunion key [key] 获取差集（从第一个key的Set集合中去除其他key的Set集合中与第一个key的交集部分）/交集/并集

sdiffstore/sinterstore/sunionstore destination key [key] 获取差集/交集/并集保存到destination中

有序集合SortedSet：有序，去重，元素为字符串类型，每个元素关联一个浮点数值（score，计算机只能近似表达一个浮点数），按照score从小到大排序集合元素，score可以相同，最多2^32-1个元素

zadd key score value [score value] 指定key的SortedSet集合中加入多个score value，value重复使用覆盖score

zrem key value [value] 指定key的SortedSet集合中删除多个value，value不存在自动忽略

zscore key member 获取key的有序集合元素member对应的score

zincrby key 数值 member 指定key的有序集合的member元素值增加指定数值

zrank/zrevrank key member 获取key的有序集合的member元素的排名/倒序排名

zrange/zrevrange key start stop [withscores] 获取key的有序集合的指定区间内的正排序（默认按照score从小到大）/逆排序元素[带score]

zrangebyscore/zrevrangebyscore key min max [withscores] [limit offset count] 返回key的有序集合中score在指定区间[带score][跳过offset个元素后的count个元素]的按照score升序/降序（相同按照值的字典升序/降序）元素，min和max默认闭区间使用()表示开区间，+inf和-inf表示正负无穷

zremrangebyrank key start stop 删除key的有序集合中指定排名区间的元素

zremrangebyscore key min max 删除key的有序集合中指定score区间的元素

zcard key 获取key有序集合的元素个数

zcount key min max 获取key有序集合中score区间的元素个数

zdiffstore/zinterstore/zunionstore destination 数值 key [key] [weights weight] [aggregate sum/min/max] 获取数值个有序集合差集/交集/并集[与第一个key对应的score乘以权重][聚合方式为相同元素的score的和/最小值/最大值]保存到destination中

Redis持久化方式：

RDB（Redis DB，全量备份保存整个数据库，保存时间长，数据还原速度快，一般用于数据备份）：将服务器包含的所有数据库的数据以二进制文件的形式保存到硬盘默认dump.rdb，再次启动会加载RDB文件恢复还原数据，可以通过命令save（客户端阻塞命令）和bgsave（后台命令，创建子进程消耗内存不阻塞服务速度比save慢）创建RDB文件或在配置文件中配置自动执行bgsave，RDB文件已存在会被覆盖

自动创建RDB配置文件中配置：save 秒数 次数 距离上次创建RDB文件时间在指定秒数内所有数据库超过指定修改次数进行自动创建RDB文件，默认配置为：

save 900 1

save 300 10

save 60 10000

条件中满足一个服务器就会自动持久化，持久化完成后时间和次数计数器清零不累加，持久化次数不能过于频繁会消耗资源可能会产生数据丢失

AOF（AppendOnlyFile，增量备份将Redis执行的命令持久化，保存时间间隔默认1秒，数据还原速度一般，不会阻塞服务器，适合保存数据，默认不开启），Redis每执行一次命令会将命令追加到AOF文件尾部，理论上不会丢失数据但由于操作系统执行write函数写文件时为了提高效率会先将内容写入buffer，缓冲区buffer满时或者fsync函数和fdatasync函数调用时才会将缓冲区内容写入到硬盘，Redis为AOF持久化提供了appendfsync选项：always（服务器每执行一次命令调用一次fdatasync函数，速度慢影响效率，安全性与SQL数据库一致）、everysec（服务器每一秒调用一次fdatasync函数，默认选项，速度较快）和no（服务器不主动调用fdatasync函数，速度较快）

Redis为避免AOF文件冗余命令以及大小控制会进行AOF文件重写（整合冗余命令），可用通过命令bgrewriteaof命令手动重写AOF文件或配置文件中配置：

auto-aof-rewrite-min-size 数值带单位 文件到达指定数值大小进行重写

auto-aof-rewrite-percentage 100 避免文件无限接近指定大小无限重写，设置文件新增大小比例才重写

Redis主从复制Replication模式：Redis服务Master会将自己的数据一直同步更新到多个复制服务Slaves，Master可以执行写命令，Slaves只能执行读命令分担Master读数据的负载

Redis安装目录/bin/redis-server –slaveof 主IP地址 主端口 启动Redis服务作为指定主的从服务器

Redis内部命令slaveof 主IP地址 主端口 修改当前服务器转换指定主的从服务器，slaveof no one 修改当前服务器转换主服务器（已同步的数据不丢失）

配置文件中加入slaveof 主IP地址 主端口 使用配置文件启动时自动作为指定主的从服务器

Redis哨兵Sentinel高可用：分布式系统管理多个Redis服务实例，可以运行多个Sentinel

Redis安装目录/bin/redis-sentinel Sentinel配置文件或Redis安装目录/bin/redis-server Redis安装目录/sentinel.conf --sentinel 启动运行在Sentinel模式下的Redis服务实例

每一个Sentinel会不断检查多个Maser和Master的Slaves是否正常，监控同一个Master的Sentinel之间会自动通信交换监控服务器的信息（类似ZooKeeper但没ZooKeeper智能，一般Sentinel数量为奇数），Master宕机监控Master的Sentinel会将Master下的从服务器转换为主服务器其他从服务器转换为新主服务器的从服务器，宕机的Master重新启动会被Sentinel自动作为从服务器

Sentinel配置文件

sentinel monitor 名称 主IP地址 主端口 数值 监控指定主服务器，名称为指定名称（Sentinel根据名称区分监控的多个Master），Sentinel最小成功投票数量为数值（一般为Sentinel数量的一半），Sentinel默认端口为26379，Sentinel会根据主服务器配置找到从服务器，Master写请求有单点问题

Redis Twemproxy：Twitter基于分片Sharding多节点分担关系型数据库处理大表水平切分的思想开发Twemproxy代理用户的读写请求，兼容Redis和Memcached，允许将多个Redis服务器添加到一个服务器池pool中，通过用户选择的散列函数和分布函数将请求分发给服务器池中的服务器

Twemproxy配置文件一般配置格式（YML文件）：

sxt: 服务器池的名字，支持多个服务器池

listen:服务器池代理服务器IP地址:端口号（一般22121）

hash:fnvla\_64 散列算法用于将键映射为一个散列值（代理服务器将请求映射成散列值，根据区域分布到不同服务器上）

distribution:ketama 分布算法决定键分布到的服务器

auto\_eject\_hosts:true 散列值对应的服务器宕机是否拒绝请求

redis:true 代理redis命令请求，默认false代理Memcached请求

server\_retry\_timeout: 连接超时时间

server\_failure\_limit: 连接超时次数

servers:

[- Redis服务器IP地址:端口号:权重值]

Twemproxy安装：

安装并启动redis

在1台机器上安装Twemproxy

tar xf autoconf-2.69.tar.gz

cd autoconf-2.69

默认放到/usr/local目录下，目录指定到/usr会自动放在/usr/bin目录下

./configure --prefix=/usr

make install

autoconf -V

tar xf automake-1.15.tar.gz

./configure --prefix=/usr

make

make install

tar xf libtoo1-2.4.5.tar.gz

./configure --prefix=/usr

make

make install

tar xf twemproxy-0.4.1.tar.gz

cd twemproxy-0.4.1

aclocal生成aclocal.m4

autoconf

mkdir config

autoheader

libtoolize

automake -a 生成configure文件

./configure

make

make install

cd /usr/1oca1/sbin/nutcracker

新建修改Twemproxy配置文件.yml

nutcracker -d -c nutcracker.bjsxt.yml 启动Twemproxy，d代表 daemon后台，twemproxy代理了 Redis的协议，可以直接用Redis安装目录/bin/redis-cli -h Twemproxy代理IP地址 -p 端口连接

Twemproxy代理可以做到写请求的负载均衡，但集群扩展性差，容错性差，可以使用Sentinel与Twemproxy组合实现一定的业务需求，Sentinel与Twemproxy的整合方案redis-mgr，开源网站：<https://github.com/changyibiao/redis-mgr>，可参考一般使用Redis集群方案

Redis集群：3.0版本之后支持，集群由多个Redis服务器组成分布式网络服务集群，每个Redis服务器成为节点Node，节点之间两两互相通信但增加了网络IO，无中心节点，Redis集群的每个节点有两种角色：主节点MasterNode和从节点SlaveNode，主节点用于存储数据，从节点用于某个主节点的数据复制，添加从节点可以扩展集群系统的读性能，Redis集群的主节点内置了类似Redis Sentinel的节点故障检测和自动故障转移功能，某个主节点下线时会被集群其他在线主节点检测到并对下线的主节点进行故障转移，将下线主节点对应的数据复制从节点上线为主节点，集群故障转移的方法与Redis Sentinel故障转移方法基本一致，集群中故障转移交由其他在线主节点负责，所有集群不需要使用Redis Sentinel

Redis集群将数据库分为16384个槽位slot，每个key对应一个slot，key的槽位计算公式slot\_number=crc16(key)%16384，crc16位16位的循环冗余校验和函数，集群中每个节点都可以处理0到16384个槽，集群模式时所有主节点平均分配16384个槽位（可以手动重新分配以及指定每个主节点的槽位范围和迁移槽位），当16384个槽位都由某个节点负责处理时，集群进入上线状态并处理客户端发送的数据命令请求，由于Redis集群无中心节点，客户端请求可以发给任意主节点，主节点只会处理自己负责槽位的命令请求，其他槽位的命令请求，该主节点会返回客户端转向错误，客户端根据错误中包含的地址和端口重新向正确的主节点发送命令请求

Redis集群搭建：

安装编译环境yum install gcc tcl -y

解压Redis安装包，在Redis解压包目录下执行make和make PREFIX=安装目录 install进行编译和安装，在指定的安装目录中会产生bin目录，为了方便使用可以将bin目录添加到系统环境变量中或将bin目录下文件复制到/usr/local/bin/目录下

解压Redis安装包进入根目录

make 编译

make PREFIX=安装目录 install 安装

mkdir cluster 由于Redis可以根据配置文件在一台机器上启动多个实例，但有可能会产生相同文件名覆盖等情况，一般为每个实例按照端口创建文件夹避免文件覆盖情况

配置Redis的配置文件redis.conf，一般格式为：

redis配置文件

#是否后台运行

daemonize yes

#redis的进程文件

pidfile /var/run/redis\_8000.pid

#指定redis接收指定IP地址的请求

bind 10.124.142.138

#redis监听的端口号

port 8000

#是否开启集群模式

cluster-enabled yes

#集群配置文件的名称

cluster-config-file nodes\_8000.conf

#集群节点超时时间，单位毫秒

cluster-node-timeout 5000

#持久化方式

appendonly yes

持久化文件名

appendfilename ""

持久化sync选项

appendfsync everysec

AOF重写新增大小比例

auto-aof-rewrite-percentage 100

AOF文件重写最小体积

auto-aof-rewrite-min-size

#最大内存容量

maxmemory 1gb

#达到最大内存容量后的处理策略

maxmemory-policy allkeys-lru

#是否主动清理内存碎片

activedefrag no

# 内存碎片达到多少的时候开启整理

active-defrag-ignore-bytes 100mb

# 碎片率达到百分之多少开启整理

active-defrag-threshold-lower 10

# 碎片率小余多少百分比开启整理

active-defrag-threshold-upper 100

#以CPU百分比表示的碎片整理的最小工作量

active-defrag-cycle-min 25

#以CPU百分比表示的碎片整理的最大工作量

active-defrag-cycle-max 90

#redis主节点密码

masterauth 123456

#redis密码配置

requirepass 123456

#日志文件位置

logfile /home/kaku/app/redis-4.0.12/conf/redis\_6379.log

#数据库数量

databases 16

Redis安装目录/bin/redis-server Redis的配置文件路径名 以配置文件方式启动Redis，需要集群每一个Redis实例

配置文件配置的端口为客户端的连接的端口

Redis会根据配置文件cluster-enabled yes默认会另外开启配置端口+1000作为内部沟通的端口

创建集群：

Redis槽位分配的脚步Redis安装目录/src/redis-trib.rb是Ruby脚本需要安装Ruby

yum install ruby rubygems -y 安装Ruby

gem install redis 网络下载安装Ruby与Redis连接枢纽

gem文件下载地址：https://rubygems.org/gems/redis/versions

也可以通过下载文件后安装gem install --local redis-3.3.0.gem文件路径名

Redis安装目录/src/redis-trib.rb create --replicas Master的Slaves个数 Redis主服务器IP地址:端口号[Redis主服务器IP地址:端口号] Redis从服务器IP地址:端口号[Redis从服务器IP地址:端口号]（Redis主从服务器个数需要相对应） 将指定已经启动的Redis实例创建Redis集群分配槽位slot

Redis安装目录/bin/redis-cli -h/c/p/a/n 指定IP地址/集群模式/端口号/密码/数据库号登录Redis，默认登录本地无密码0号数据库Redis

Redis安装目录/src/redis-trib.rb也可以对Redis集群的槽位进行重新分配

redis命令

debug populate 7000000 asdf 150 构造大量数据产生内存碎片

info keyspace 查看内存使用情况

info memory 查看内存碎片率(mem\_fragmentation\_ratio)

memory malloc-stats 查看内存分配情况

config set activedefrag yes 手动配置启动自动整理内存碎片

memory purge 手动清理内存碎片

cluster meet IP地址 端口号 增加redis节点

cluster nodes 查看redis集群节点信息

cluster info 查看redis集群信息

cluster replicate 主节点ID号 设置登录节点作为制定主节点的从节点

config set masterauth "" 设置主服务器的暂时密码，重启失效

config set requirepass "" 设置请求的暂时密码，重启失效

config rewrite 设置Redis配置的命令生效

设置密码之后如果需要使用redis-trib.rb的各种命令

如：./redis-trib.rb check 127.0.0.1:7000，则会报错ERR] Sorry, can’t connect to node 127.0.0.1:7000

解决办法：vim /usr/local/rvm/gems/ruby-2.3.3/gems/redis-4.0.0/lib/redis/client.rb，然后修改passord

Redis集群的开源管理平台CacheCloud安装：

下载解压安装包

创建CacheCloud需要的数据库

使用SQL脚本CacheCloud解压目录/script/cachecloud.sql初始化CacheCloud需要的数据库的表结构

修改CacheCloud解压目录/cachecloud-open-web/src/main/swap/online.properties文件，一般格式：

cachecloud.db.url = jdbc:mysql://127.0.0.1:3306/cache-cloud 配置CacheCloud的数据库连接地址

cachecloud.db.user = cachecloud CacheCloud数据库用户名

cachecloud.db.password = CacheCloud的数据库密码

cachecloud.maxPoolSize = 20

isClustered = true

isDebug = true

spring-file = classpath:spring/spring-local.xml

log\_base = /opt/cachecloud-web/logs 日志目录

web.port = 9999 CacheCloud的Web访问端口号

log.level = INFO

进入CacheCloud解压目录即根目录使用命令mvn clean compile install -Ponline进行Maven编译和构建

mkdir -p /opt/cachecloud-web/ 创建工作目录，因为一些脚本大部分目录是固定的

cp -f /opt/soft/cachecloud/cachecloud-open-web/target/cachecloud-open-web-1.0-SNAPSHOT.war /opt/cachecloud-web/ 将Maven编译和构建的war包复制到工作目录

cp -f /opt/soft/cachecloud/cachecloud-open-web/src/main/resources/cachecloud-web.conf /opt/cachecloud-web/ 将配置复制到工作目录

ln -s /opt/cachecloud-web/cachecloud-open-web-1.0-SNAPSHOT.war /etc/init.d/cachecloud-web 创建war包软链接加入到系统服务目录下

/etc/init.d/cachecloud-web start 启动CacheCloud

CacheCloud的Web访问地址：启动CacheCloud主机IP地址: CacheCloud的Web访问端口号 用户名密码默认为admin/admin

CacheCloud的GitHub地址：<https://github.com/sohutv/cachecloud>

Redis的java客户端一般用Jedis：https://github.com/xetorthio/jedis