redis运行

服务端：redis-server

客户端：redis-cli [-h ip -p port -a password]

redis配置

配置文件：redis.windows.conf

命令格式：CONFIG GET 配置名

1. 查看所有配置项：CONFIG GET *

设置配置：CONFIG SET 配置名 配置值

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

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 keys 命令

语法：COMMAND KEY_NAME

设置：SET keyname key_value

删除：DEL keyname，执行成功返回1，否则0

命令：https://www.runoob.com/redis/redis-keys.html

序号	命令	用法
1	DEL key	key存在时删除key
2	DUMP key	序列化给定 key ，并返回被序列化的值
3	EXISTS key	检查给定 key 是否存在
4	EXPIRE key sedonds	为给定 key 设置过期时间，以秒计
5	EXPIREAT key timestamp	EXPIREAT 的作用和 EXPIRE 类似，都用于为 key 设置过期时间。不同在于 EXPIREAT 命令接受的时间参数是 UNIX 时间戳(unix timestamp)
6	EXPIREAT key milliseconds	设置 key 过期时间的时间戳(unix timestamp) 以毫秒计
7	PEXPIREAT key milliseconds-timestamp	设置 key 过期时间的时间戳(unix timestamp) 以毫秒计
8	KEYS pattern	查找所有符合给定模式(pattern)的 key
9	MOVE key db	将当前数据库的 key 移动到给定的数据库 db 当中
10	PERSIST key	移除 key 的过期时间，key 将持久保持
11	PTTL key	以毫秒为单位返回 key 的剩余的过期时间
12	TTL key	以秒为单位，返回给定 key 的剩余生存时间(TTL, time to live)
13	RANDOMKEY	从当前数据库中随机返回一个 key
14	RENAME key newkey	修改 key 的名称
15	RENAMEX key newkey	仅当 newkey 不存在时，将 key 改名为 newkey
16	SCAN cursor [MATCH pattern] [COUNT count]	迭代数据库中的数据库键
17	TYPE key	返回 key 所储存的值的类型

redis数据类型

String（字符串）

string类型是二进制安全的，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。

```
SET testkey "例子字符"
```

```
GET testkey
```

```
"例子字符"
```

String命令

序号	命令及描述
1	SET key value 设置指定 key 的值
2	GET key 获取指定 key 的值。
3	GETRANGE key start end 返回 key 中字符串值的子字符
4	GETSET key value 将给定 key 的值设为 value ， 并返回 key 的旧值(old value)。
5	GETBIT key offset 对 key 所储存的字符串值，获取指定偏移量上的位(bit)。
6	MGET key1 [key2..] 获取所有(一个或多个)给定 key 的值。
7	SETBIT key offset value 对 key 所储存的字符串值，设置或清除指定偏移量上的位(bit)。
8	SETEX key seconds value 将值 value 关联到 key ， 并将 key 的过期时间设为 seconds (以秒为单位)。
9	SETNX key value 只有在 key 不存在时设置 key 的值。
10	SETRANGE key offset value 用 value 参数覆写给定 key 所储存的字符串值，从偏移量 offset 开始。
11	STRLEN key 返回 key 所储存的字符串值的长度。

12	MSET key value [key value ...] 同时设置一个或多个 key-value 对。
13	MSETNX key value [key value ...] 同时设置一个或多个 key-value 对，当且仅当所有给定 key 都不存在。
14	PSETEX key milliseconds value 这个命令和 SETEX 命令相似，但它以毫秒为单位设置 key 的生存时间，而不是像 SETEX 命令那样，以秒为单位。
15	INCR key 将 key 中储存的数字值增一。
16	INCRBY key increment 将 key 所储存的值加上给定的增量值（increment） 。
17	INCRBYFLOAT key increment 将 key 所储存的值加上给定的浮点增量值（increment） 。
18	DECR key 将 key 中储存的数字值减一。
19	DECRBY key decrement key 所储存的值减去给定的减量值（decrement） 。
20	APPEND key value 如果 key 已经存在并且是一个字符串， APPEND 命令将指定的 value 追加到该 key 原来值（value）的末尾。

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。

Redis 中每个 hash 可以存储 2^32 - 1 键值对（40多亿）

HMSET test_hash name 'yl' age '25'

```
127.0.0.1:6379> HMSET test_hash name 'yl' age '25'
OK
127.0.0.1:6379> HGETALL test_hash
1) "name"
2) "yl"
3) "age"
4) "25"
127.0.0.1:6379> HGET test_hash name
"yl"
127.0.0.1:6379> HMGET test_hash name
1) "yl"
127.0.0.1:6379> HMGET test_hash name age
1) "yl"
2) "25"
```

Hash命令

序号	命令及描述
1	HDEL key field1 [field2] 删除一个或多个哈希表字段
2	HEXISTS key field 查看哈希表 key 中，指定的字段是否存在。
3	HGET key field 获取存储在哈希表中指定字段的值。
4	HGETALL key 获取在哈希表中指定 key 的所有字段和值
5	HINCRBY key field increment 为哈希表 key 中的指定字段的整数值加上增量 increment 。
6	HINCRBYFLOAT key field increment 为哈希表 key 中的指定字段的浮点数值加上增量 increment 。
7	HKEYS key 获取所有哈希表中的字段
8	HLEN key 获取哈希表中字段的数量
9	HMGET key field1 [field2] 获取所有给定字段的值
10	HMSET key field1 value1 [field2 value2] 同时将多个 field-value (域-值)对设置到哈希表 key 中。
11	HSET key field value 将哈希表 key 中的字段 field 的值设为 value 。
12	HSETNX key field value 只有在字段 field 不存在时，设置哈希表字段的值。
13	HVALS key 获取哈希表中所有值。
14	HSCAN key cursor [MATCH pattern] [COUNT count] 迭代哈希表中的键值对。

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）

一个列表最多可以包含 2^32 - 1 个元素 (4294967295, 每个列表超过40亿个元素)

```
127.0.0.1:6379> LPUSH test_list 1
(integer) 1
127.0.0.1:6379> LPUSH test_list 2 3 4
(integer) 4
127.0.0.1:6379> LRANGE test_list 0 -1
1) "4"
2) "3"
3) "2"
4) "1"
```

List命令

序号	命令及描述
1	BLPOP key1 [key2] timeout 移出并获取列表的第一个元素， 如果列表没有元素会阻塞列表直到等待超时或发现可弹出元素为止。
2	BRPOP key1 [key2] timeout 移出并获取列表的最后一个元素， 如果列表没有元素会阻塞列表直到等待超时或发现可弹出元素为止。
3	BRPOPLPUSH source destination timeout 从列表中弹出一个值，将弹出的元素插入到另外一个列表中并返回它； 如果列表没有元素会阻塞列表直到等待超时或发现可弹出元素为止。
4	LINDEX key index 通过索引获取列表中的元素
5	LINSERT key BEFORE|AFTER pivot value 在列表的元素前或者后插入元素
6	LLEN key 获取列表长度
7	LPOP key 移出并获取列表的第一个元素
8	LPUSH key value1 [value2] 将一个或多个值插入到列表头部
9	LPUSHX key value 将一个值插入到已存在的列表头部
10	LRANGE key start stop 获取列表指定范围内的元素
11	LREM key count value 移除列表元素
12	LSET key index value 通过索引设置列表元素的值
13	LTRIM key start stop 对一个列表进行修剪(trim)，就是说，让列表只保留指定区间内的元素，不在指定区间之内的元素都将被删除。
14	RPOP key 移除列表的最后一个元素，返回值为移除的元素。
15	RPOPLPUSH source destination 移除列表的最后一个元素，并将该元素添加到另一个列表并返回
16	RPUSH key value1 [value2] 在列表中添加一个或多个值
17	RPUSHX key value 为已存在的列表添加值

Set（集合）

Redis 的 Set 是 String 类型的无序集合。集合成员是唯一的，这就意味着集合中不能出现重复的数据。

Redis 中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)。

集合中最大的成员数为 2^32 - 1 (4294967295, 每个集合可存储40多亿个成员)。

```
127.0.0.1:6379> SADD test_set 1 2 3 1
(integer) 3
127.0.0.1:6379> SMEMBERS test_set
1) "1"
2) "2"
3) "3"
```

Set命令

序号	命令及描述
1	SADD key member1 [member2] 向集合添加一个或多个成员
2	SCARD key 获取集合的成员数
3	SDIFF key1 [key2] 返回第一个集合与其他集合之间的差异。
4	SDIFFSTORE destination key1 [key2] 返回给定所有集合的差集并存储在 destination 中
5	SINTER key1 [key2] 返回给定所有集合的交集
6	SINTERSTORE destination key1 [key2] 返回给定所有集合的交集并存储在 destination 中
7	SISMEMBER key member 判断 member 元素是否是集合 key 的成员
8	SMEMBERS key 返回集合中的所有成员
9	SMOVE source destination member 将 member 元素从 source 集合移动到 destination 集合
10	SPOP key 移除并返回集合中的一个随机元素

zset（sorted set：有序集合）

Redis 有序集合和集合一样也是 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。

有序集合的成员是唯一的,但**分数(score)却可以重复**。

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)。 集合中最大的成员数为 2**32 - 1 (4294967295, 每个集合可存储40多亿个成员)。

语法：COMMAND setkeyname score member

```
127.0.0.1:6379> ZADD test_zset 1 yl 2 dl 1 dll
(integer) 3
127.0.0.1:6379> ZRANGE test_zset 0 -1
1) "dll"
2) "yl"
3) "dl"
127.0.0.1:6379> ZRANGE test_zset 0 -1 WITHSCORES
1) "dll"
2) "1"
3) "yl"
4) "1"
5) "dl"
6) "2"
```

zset命令

序号	命令及描述
1	ZADD key score1 member1 [score2 member2] 向有序集合添加一个或多个成员，或者更新已存在成员的分数
2	ZCARD key 获取有序集合的成员数
3	ZCOUNT key min max 计算在有序集合中指定区间分数的成员数
4	ZINCRBY key increment member 有序集合中对指定成员的分数加上增量 increment
5	ZINTERSTORE destination numkeys key [key ...] 计算给定的一个或多个有序集的交集并将结果集存储在新的有序集合 destination 中
6	ZLEXCOUNT key min max 在有序集合中计算指定字典区间内成员数量
7	ZRANGE key start stop [WITHSCORES] 通过索引区间返回有序集合指定区间内的成员
8	ZRANGEBYLEX key min max [LIMIT offset count] 通过字典区间返回有序集合的成员
9	ZRANGEBYSCORE key min max [WITHSCORES] [LIMIT] 通过分数返回有序集合指定区间内的成员
10	ZRANK key member 返回有序集合中指定成员的索引

11	ZREM key member [member ...] 移除有序集合中的一个或多个成员
12	ZREMRANGEBYLEX key min max 移除有序集合中给定的字典区间的所有成员
13	ZREMRANGEBYRANK key start stop 移除有序集合中给定的排名区间的所有成员
14	ZREMRANGEBYSCORE key min max 移除有序集合中给定的分数区间的所有成员
15	ZREVRANGE key start stop [WITHSCORES] 返回有序集中指定区间内的成员，通过索引，分数从高到低
16	ZREVRANGEBYSCORE key max min [WITHSCORES] 返回有序集中指定分数区间内的成员，分数从高到低排序
17	ZREVRANK key member 返回有序集合中指定成员的排名，有序集成员按分数值递减(从大到小)排序
18	ZSCORE key member 返回有序集中，成员的分数值
19	ZUNIONSTORE destination numkeys key [key ...] 计算给定的一个或多个有序集的并集，并存储在新的 key 中
20	ZSCAN key cursor [MATCH pattern] [COUNT count] 迭代有序集合中的元素（包括元素成员和元素分值）