1. **Reids的5种数据类型：**

字符串(String)：

Value最多可以容纳的数据长度为**512MB**

列表(list)：

允许用户从序列的两端推入或者弹出元素，链表结构，List中可以包含的最大元素数量是42,9496,7295=**232**



散列哈希(hash)：

可以看成具有String key和String value的map容器，可以将多个key-value存储到一个key中。每一个Hash可以存储42,9496,7295=**232**个键值对。



集合(set)：

无序不可重复的，和列表一样，在执行插入和删除和判断是否存在某元素时，效率是很高的。集合最大的优势在于可以进行交集并集差集操作。Set可包含的最大元素数量是42,9496,7295=**232**



有序集合(sorted set)：

和set很像，都是字符串的集合，都不允许重复的成员出现在一个set中。他们之间差别在于有序集合中每一个成员都会有一个分数(score)与之关联，Redis正是通过分数来为集合中的成员进行从小到大的排序。尽管有序集合中的成员必须是卫衣的，但是分数(score)却可以重复。  
应用场景：可以用于一个大型在线游戏的积分排行榜，每当玩家的分数发生变化时，可以执行zadd更新玩家分数(score)，此后在通过zrange获取几分top ten的用户信息。



**对key的通用操作，所有的数据类型都可以使用的:**



1. **安装redis并启动服务**
2. 下载Redis安装包，网址<https://github.com/MicrosoftArchive/redis/releases/tag/win-3.2.100>，

下载zip文件。

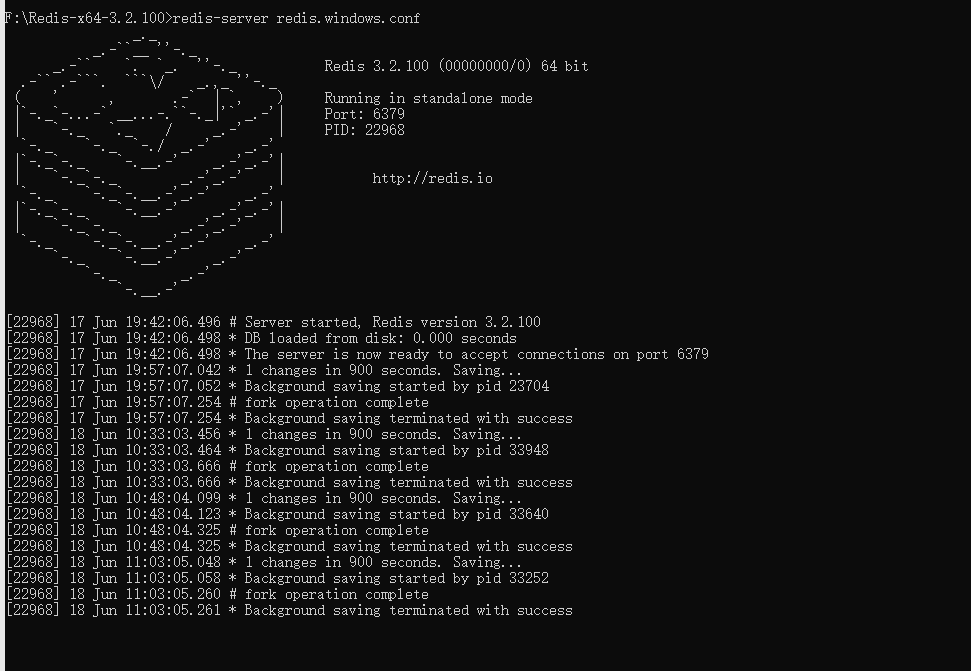
1. 进入redis安装目录，使用文本编辑器打开redis.windows.conf文件，在443行修改Redis密码，

假设密码为admin123，同杨修改redis.windows-service.conf文件。

1. 打开cmd，进入redis安装目录，执行redis-server.exe --service-install redis.windows.conf

--loglevel verbose回车

然后执行redis-server redis.windows.conf

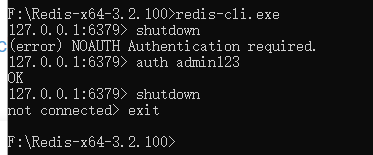


关闭该对话框之后再次启动秩序执行redis-server redis.windows.conf即可。

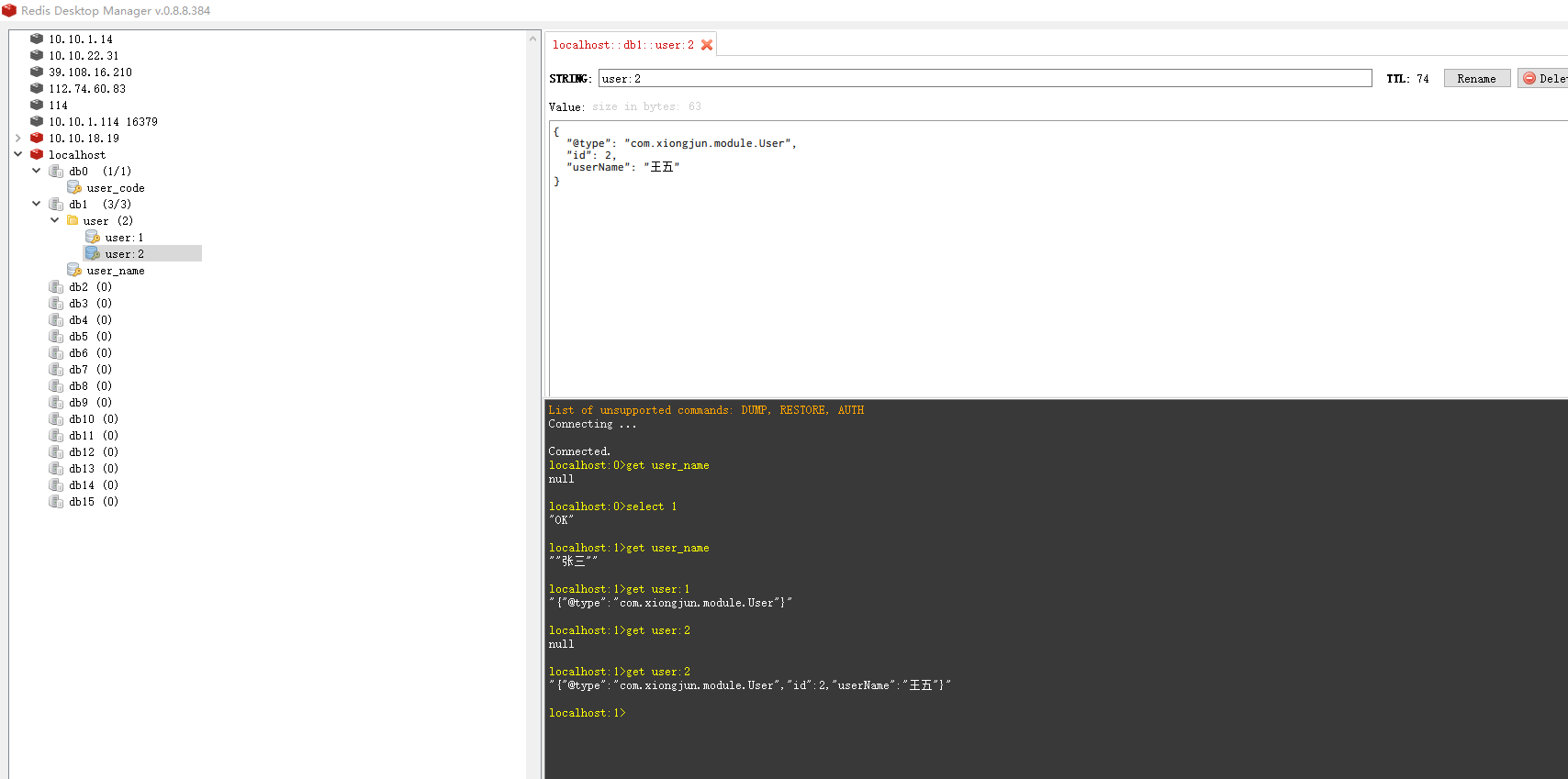
进入客户端:



或者使用auth password进行登录操作:

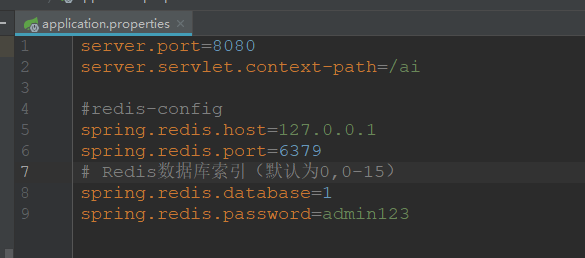


Redis Desktop Manager:



切换数据库,比如要切换到db1:select 1即可。

如果是在代码中使用，则只需要配置database即可：



如果执行命令报错:

creating server tcp listening socket 127.0.0.1:6379: bind No error

解决方案如下：

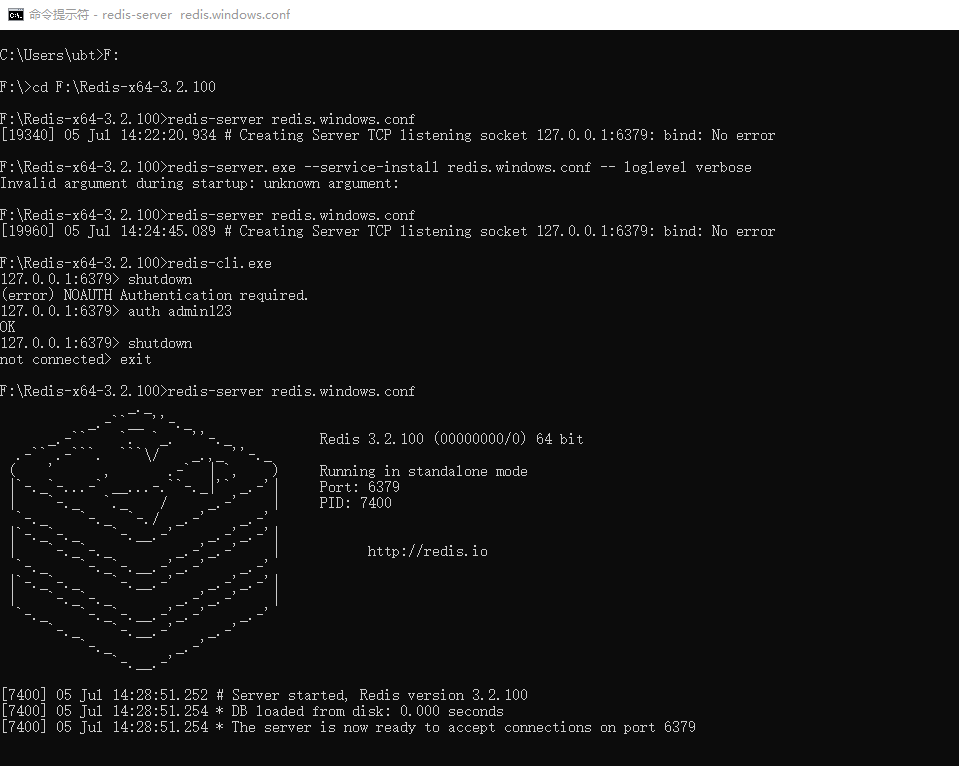
按顺序执行以下命令:

redis-cli.exe

shutdown

exit

redis-server.exe redis.windows.conf



1. **redis持久化**

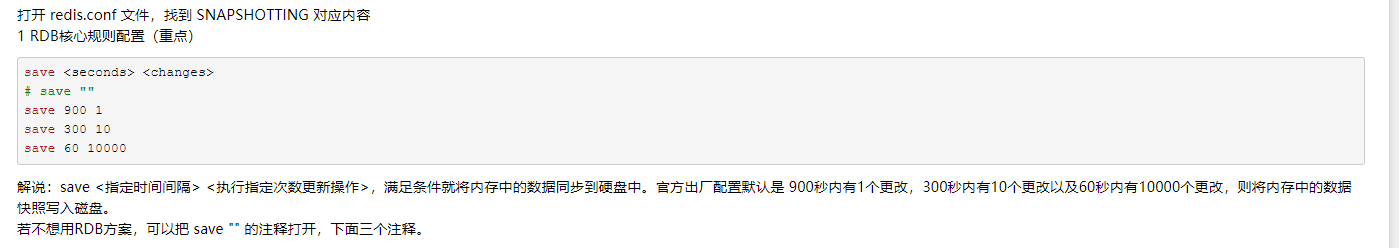
redis的数据都存放在内存中，如果不进行持久化，redis重启后数据就会全部丢失。

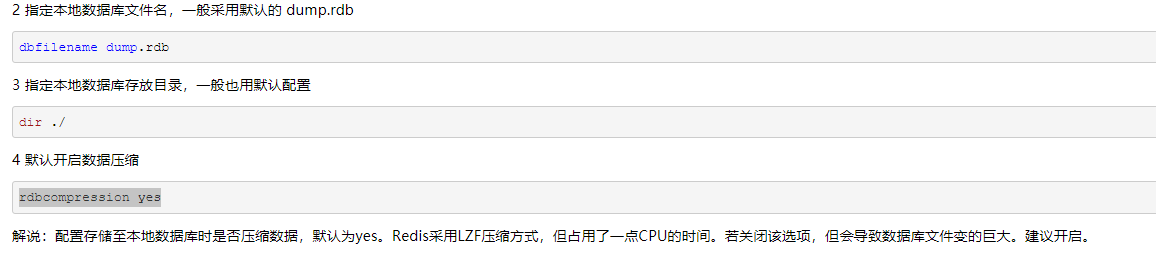
1、RDB:

在指定的时间间隔内，执行指定次数的写操作，将内存中的数据写到磁盘中。在指定目录下生成一个dump.rdb文件。redis重启会通过加载该文件恢复数据。

操作过程：Redis会fork出一个子进程，读取内存中的数据，并重新写到一个临时文件中（并没有读取已经存在的.rdb文件），写入成功后，再替换旧的.rdb文件,用二进制压缩存储。

通常情况下**1000w的字符串类型键，1GB快照文件，同步到内存中时间是20-30秒**。





执行save时，阻塞，只保存快照，其他等待。

优点:

①、适合大规模的数据恢复

缺点：

①、数据的完整性和一致性不高，因为可能会在最后一次备份时宕机了；

②、备份时暂用内存，因为Redis在备份时会独立创建一个子线程，将数据写到一个临时文件，然后再将临时文件替换之前的备份文件，这样备份的时候内存中的数据是没有备份时的2倍。

2、AOF:

采用日志的形式来记录每个写操作，并追加到文件中。Redis重启的时候会根据日志文件的内容将写指令从前到后执行一次来恢复数据。



AOF的工作原理是将写操作追加到文件中，文件的冗余内容会越来越多。所以聪明的 Redis 新增了重写机制。当AOF文件的大小超过所设定的阈值时，Redis就会对AOF文件的内容压缩。Redis 会fork出一条新进程，读取内存中的数据，并重新写到一个临时文件中（并没有读取旧文件），最后替换旧的aof文件。

### **根据AOF文件恢复数据**

正常情况下，将appendonly.aof 文件拷贝到redis的安装目录的bin目录下，重启redis服务即可。但在实际开发中，可能因为某些原因导致appendonly.aof 文件格式异常，从而导致数据还原失败，可以通过命令redis-check-aof --fix appendonly.aof 进行修复 。从下面的操作演示中体会。

优点:

数据完整性更好；

缺点：

因为AOF记录的内容多，文件会越来越大，服务在重启后数据恢复也会越来越慢。

两种持久化方式可以同时使用。这种情况下，当redis重启时，他会优先使用aof文件来还原数据集，因为aof文件保存的数据即通常比rdb文件锁保存的数据集更完整。

1. **redis的过期键的删除策略**

过期键的删除策略通常有以下三种：

1. 定时删除

在设置键的过期时间的同时，创建一个定时器(timer)，让定时器在键的过期时间来临时，立即执行对键的删除操作。

这种策略对内存时最友好的，通过使用定时器定时删除数据可以保证过期键会尽可能快被删除，并释放过期键所占用的内存。但是对CPU不友好，在过期键比较多的情况下，删除过期键这一行为可能会占用相当一部分CPU时间，在内存不紧张但是CPU时间非常紧张的情况下，将CPU时间用在删除和当前任务无关的过期键上，会对服务器的响应时间和吞吐量造成影响。

1. 惰性删除

放任键过期不管，但每次从键空间中获取键时，都检查取得的键是否过期，如果过期的话，就删除该键，如果没有过期就返回该键。

这种策略对CPU来说是友好的，程序只会在取键时才对键进行过期检查，着可以保证删除过期键的操作只会在非做不可得情况下进行。但是对内存不友好，如果一个键已经过期，而这个键仍然保存在数据库中，只要这个键不被删除，它所占用得内存就不会被释放。

1. 定期删除

每隔一段时间，程序就进行检查一次，删除过期键。至于删除多少过期键，以及要检查多少个数据库，由算法决定。

1. **内存不足时，redis的淘汰策略**

<https://blog.csdn.net/libafei/article/details/80311372>

<https://blog.csdn.net/hehmxy/article/details/102615953>

<https://www.cnblogs.com/lukexwang/p/4694094.html>