## 数据库分为关系型数据库和非关系型数据库

数据库的目的一定都是用来做做数据的增删改查，无非是在数据存储的方式、做数据增删改查的方式不同罢了！

## 关系型数据库SQL：

**数据是结构化的（Structured）：**体现在结构化这里（Structured）：结构化简单理解就是对数据有要求、有约束！数据是二维表结构（行+列）加上数据约束（字段的约束）；约束一旦定义好，表的结构就固定了，将来插入表的所有数据都要样饿遵循这些约定，而且数据库也会对你插入的数据做校验，如果并不符合约定，自然就不报错，不允许你做数据插入，这就是结构化的含义，而且表的结构一般是不建议随便做修改的，最好是项目设计之初就定义设计好，因为一旦数据插入较多，如果你要去修改表中字段可能带来的影响是巨大的，比如导致锁表长时间不可用！而且表跟业务有关联，数据库表变了，业务也会跟着变！

**数据是关联的（Relational）：**表与表之间可以通过外键建立联系，数据库会帮我们去维护表与表的关联；

**SQL查询：**固定SQL语法查询的，所以只要是关系型数据库，都能用相同的语句去查询，不管你是MySQL还是Orical；

**强事务：**事务是ACID的；

**数据存储方式：**存储方式一般是磁盘；

**扩展性：**扩展性是垂直的，就是说关系型数据库在设计之初，可能就没有考虑到分布式数据分片的需求，所以当我们存储数据到关系型数据库的时候，都是存储在本机，影响性能的是本机服务器的性能，只能通过提升机器的性能来提升数据库的能力，虽然说MySQL支持主从，但是主从仅仅是提升机器的数量，提升读写的性能，但是你并不能提升数据存储的量，因为你主和从存储数据是一模一样的，数据存储的总量没有变，只不过是做了备份而已，虽然关系型数据库默认情况下是并不支持水平拓展的，但是你也可以基于使用第三方的组件来去实现数据库的分库，当然了，一旦引入使用了第三方的组件，肯定会对性能造成影响，并且在开发的时候需要考虑的问题就更多，开发的复杂度就增加了，不像NOSQL天然就支持水平拓展，开发难度和性能都不会有特别大的影响；

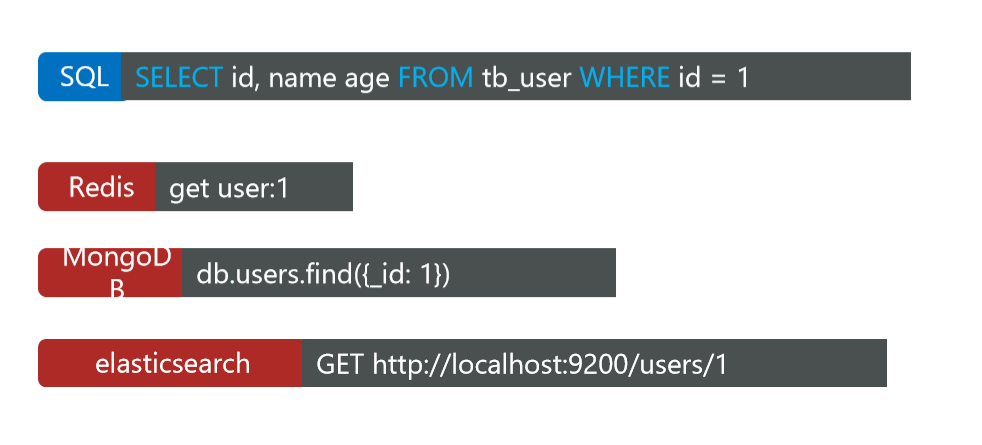
使用场景：数据结构固定；相关业务对数据安全性、一致性要求较高

## 非关系型数据库NOSQL：

**数据非结构化：**对数据没有非常严格的约束；

**数据无关联：**数据与数据之间没有直接的关联；

**非SQL：**没有固定SQL；



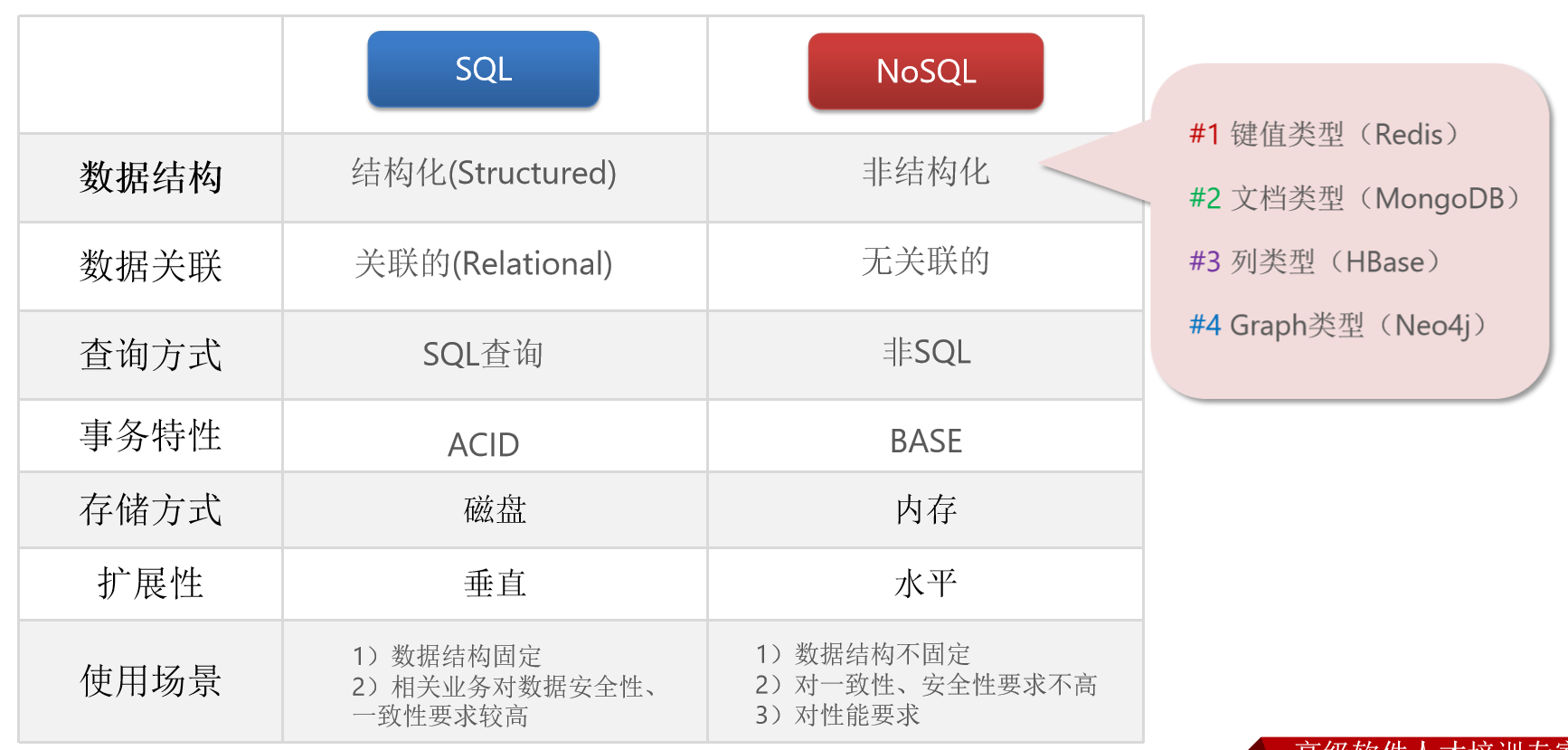
**弱事务的：**事务是BASE的；

**数据存储方式：**存储方式一般是内存；

**拓展性：**扩展性是水平的，查询性能好，NOSQL设计之初就考虑到了数据拆分的需求，因此他们插入数据的时候往往都会基于数据的ID，或者唯一的表示去对数据做HASH运算，根据HASH运算的结果去判断这个数据到底应该存储在哪一个不同的节点上，从而实现数据的拆分，天然就支持水平扩展，因此NOSQL数据库在拓展的时候是非常轻松就能实现；

使用场景：数据结构不固定；相关业务对数据安全性、一致性要求不高；有性能要求

总结：关系型数据库是结构化的、关联的；非关系型数据库是非结构化的、无关联的



其实非关系型数据库是对关系型数据库的补充，并不是、也不可能取代关系型数据库，关系型数据库永远是主流，根基，非关系型数据库只是在特定场景下比关系型数据库更适合！

## Redis：键值型的非关系型数据库

基于内存的键值型非关系型数据库，根据key找value

1：键值（key-value）型：键是字符串类型，值value支持不同的数据结构

2：单线程：redis是单线程执行命令的，所以所有的命令是串行执行的，不会存在一个命令执行到一半的时候，其他命令插入进来的情况，即每个命令具备原子性、安全的，有人会说，Redis从6.x开始不是说已经变成多线程了吗？这是因为Redis6.x的所线程仅仅是在对于网络请求处理之块，而核心的命令的执行这部分依旧是单线程的，所以我们说redis6.x是单线程依然是对的！

3：低延迟、速度快（基于内存、IO多路复用、良好的编码）：为什么Redis单线程性能速度依旧优秀，这是因为Redis基于内存的，就是说数据都是在内存存储，不想MySQL存储在磁盘，内存的读写速度相比磁盘可是快多了，速度是指数级的提升；第二尽管Redis是单线程，但却基于IO多路复用的方式，大大提高了服务的吞吐能力；第三个原因就是Redis良好的编码，Redis是基于C语言编写的，编码风格、习惯是标杆级别的好。基于以上三个原因，Redis尽管单线程，但性能依旧非常地优异！当然最重要的原因还是基于内存存储。

4：支持数据持久化，确保数据安全性：基于内存存储数据查询性能确实高，但是存在不安全的情况，比如一旦断电，内存数据就不存在了，所以为了解决这个问题，Redis做了数据持久化的功能，会定期地将数据从内存持久化到磁盘，从而确保数据的安全性！

5：支持主从集群、分片集群：主从集群在MySQL中也有，就是指从节点可以去备份主节点的数据，也是一种数据安全性的考虑嘛，这样一来一旦有节点宕机，我们的数据在其他从节点上就能找到，这就是一种安全策略；同时主从可以做读写分离，从而大大提高查询、读写的效率。分片就是对数据做拆分，也就是说把数据拆成N份存到不同的节点中去，这样一来由很多机器一起来存数据，存储数据的上限就提高了，也就是一种水平的拓展。

6：支持多语言客户端，JAVA、Python、C各种各样的语言都支持，能操作Redis

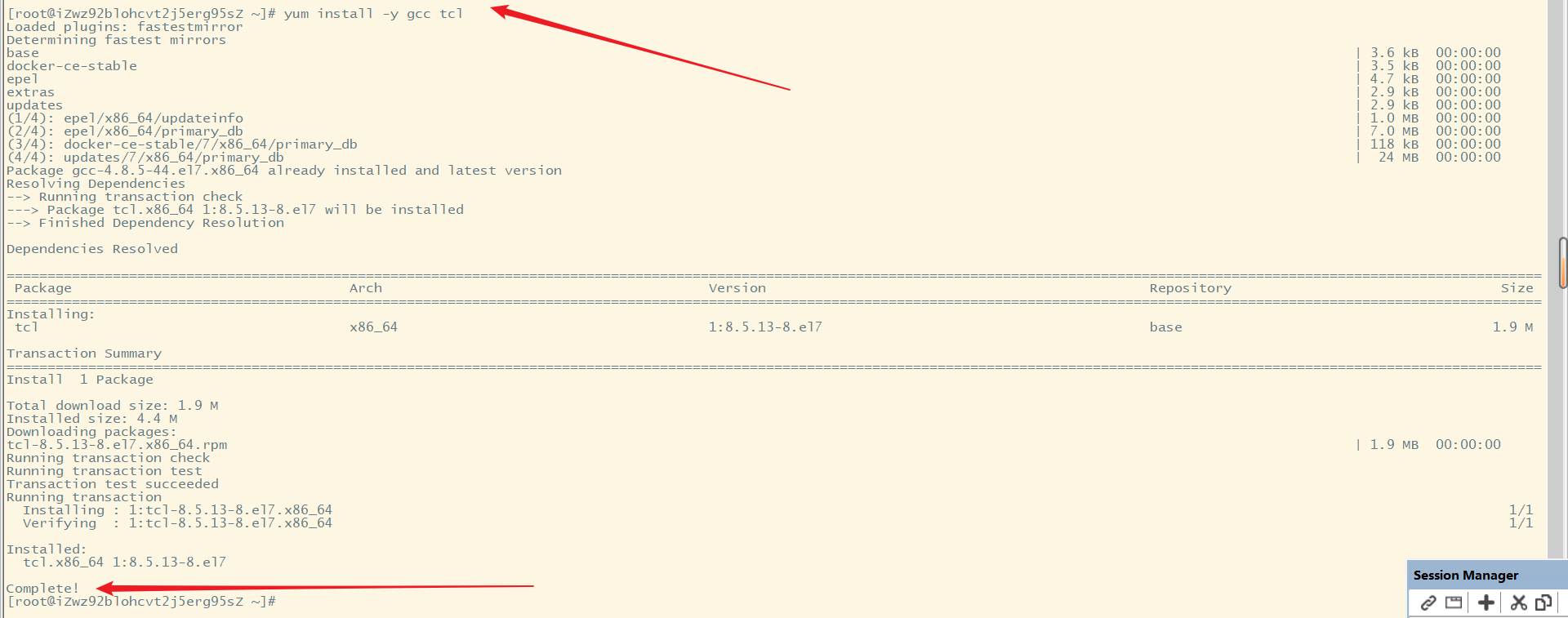
你看看，功能丰富；性能还好；但支持数据持久化、安全；还能做各种集群，轻易地做水平拓展；并且任何语言都能用，企业不用Redis用什么！Redis活该火！

## 安装Redis

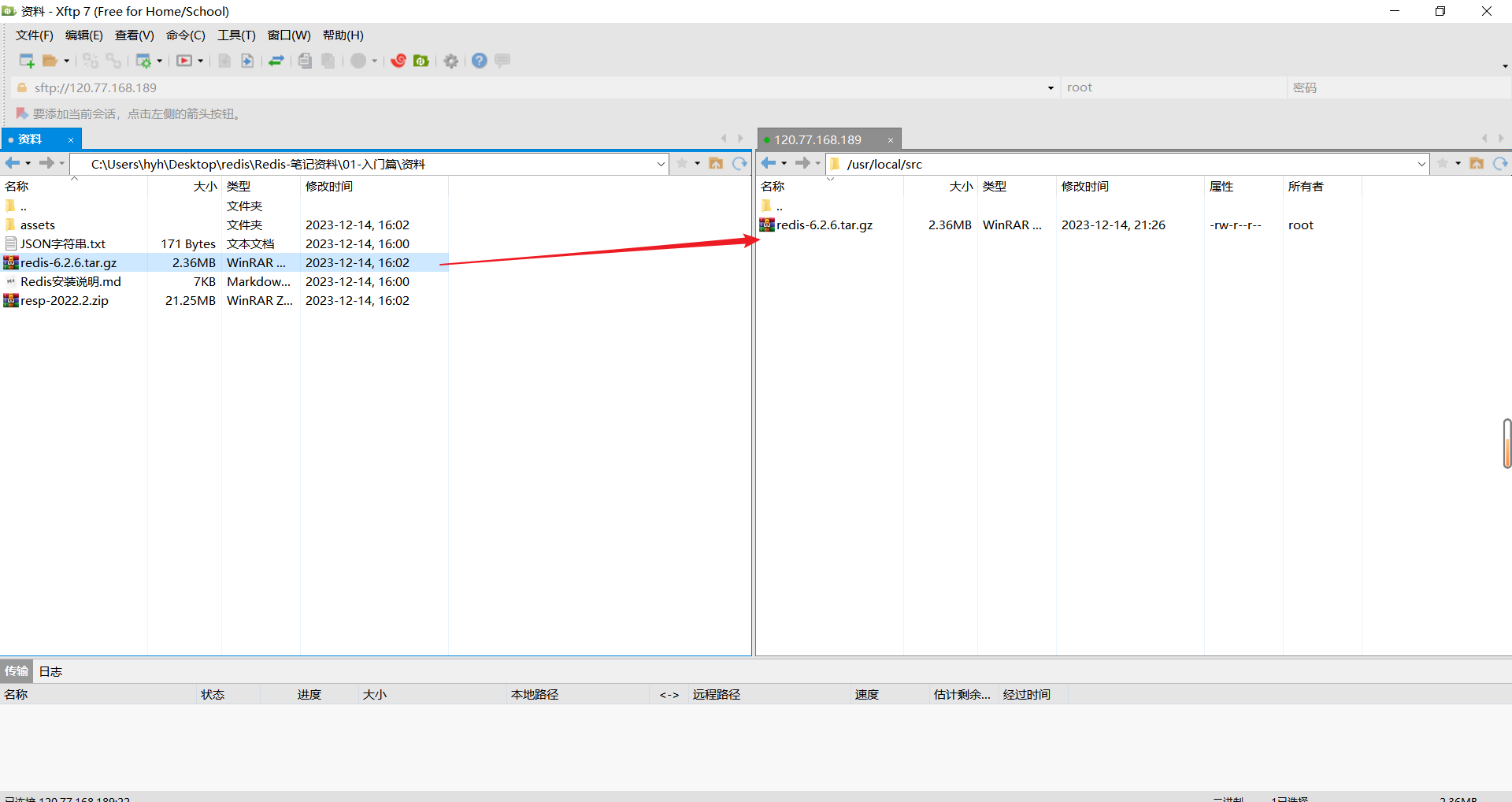
大多数企业都是基于Linux服务器来部署项目，而且Redis官方也没有提供Windows版本的安装包，你在网上找到的Windows版本的Redis其实并不是官方提供的，而是由微软自己编译出来的。因此课程中我们会基于Linux系统来安装Redis。选择的Linux版本为CentOS 7。

### 1：安装Redis依赖：yum install -y gcc tcl

因为Redis是基于C语言编写的，因此首先需要安装Redis所需要的gcc依赖

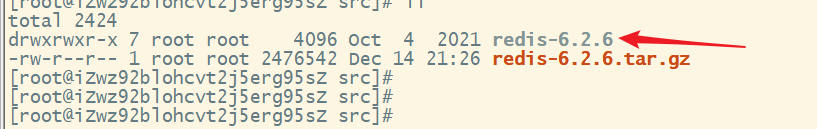


### 2：上传Redis安装包到目录：/usr/local/src

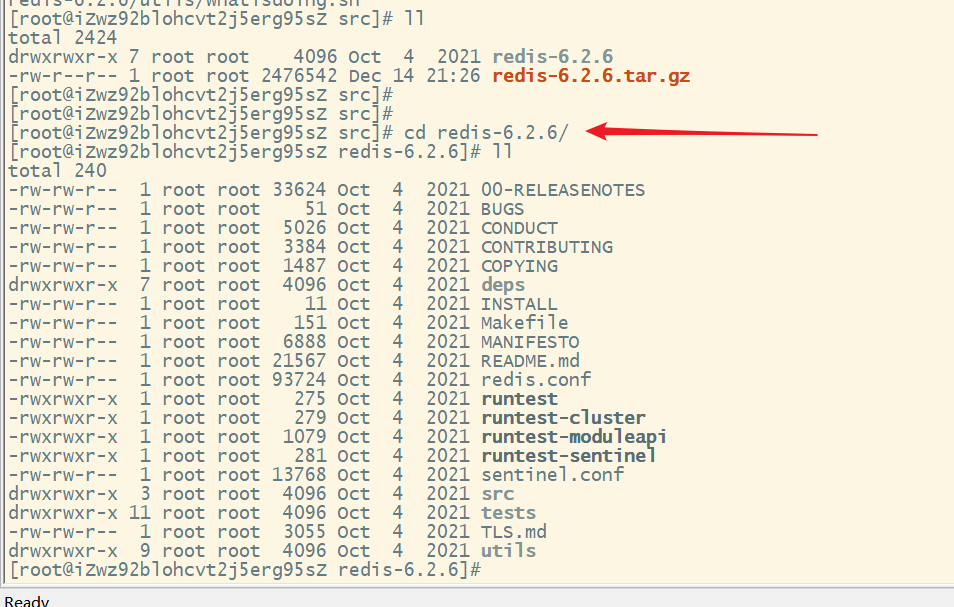


### 3：解压安装包：tar -zxvf redis-6.2.6

当然如果你觉得解压后的Redis目录太长，你可以修改解压之后的Redis目录名，这里就不修改了

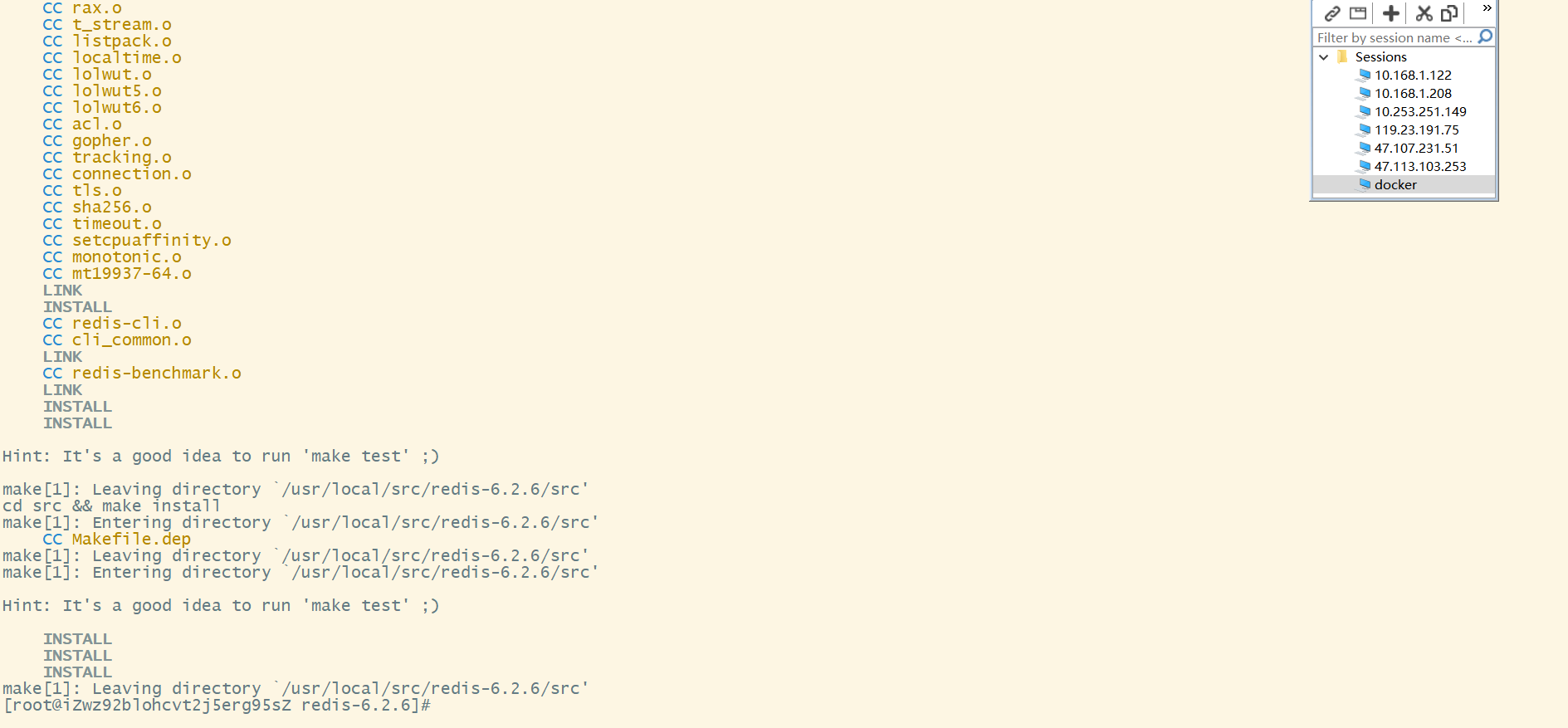


### 4：进入解压后的Redis目录



### 5：运行编译命令：make && make install

make是编译，make install是安装的意思



### 6：Redis官方文档地址：<https://redis.io/commands/>

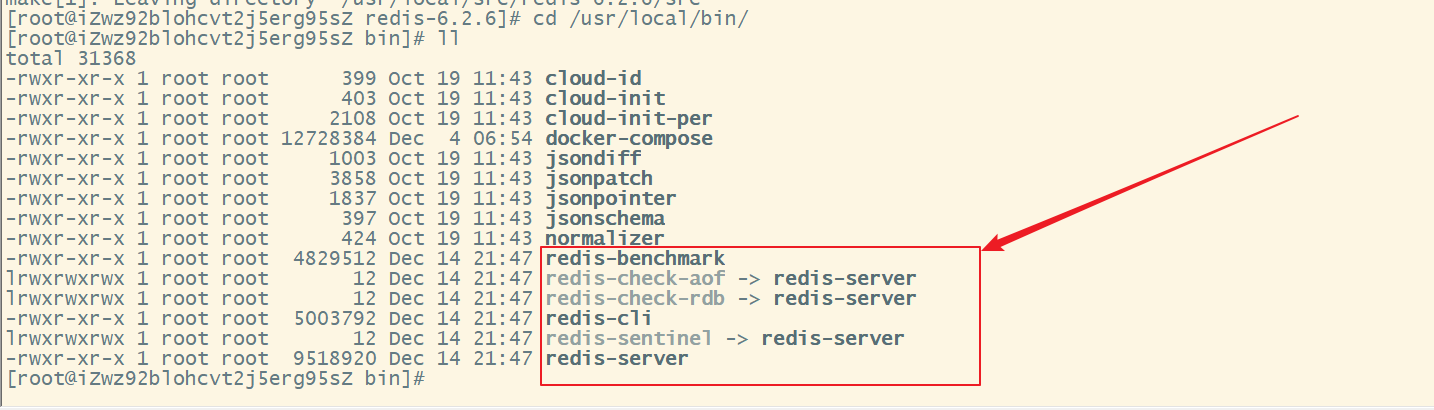
### 7：安装完成后，默认Redis的安装目录在：/usr/local/bin

该目录已经默认配置到了环境变量，因此可以在任意目录运行Redis命令，最常用：

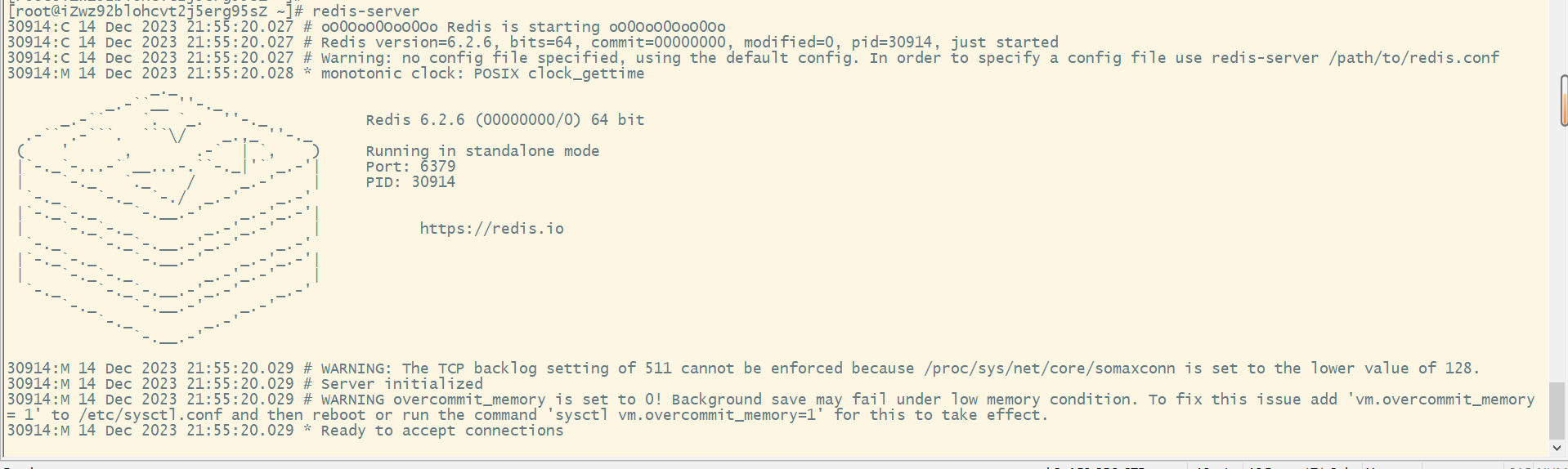
**redis-cli：是Redis提供的命令行客户端**

**redis-server：是Redis的服务端启动脚本**

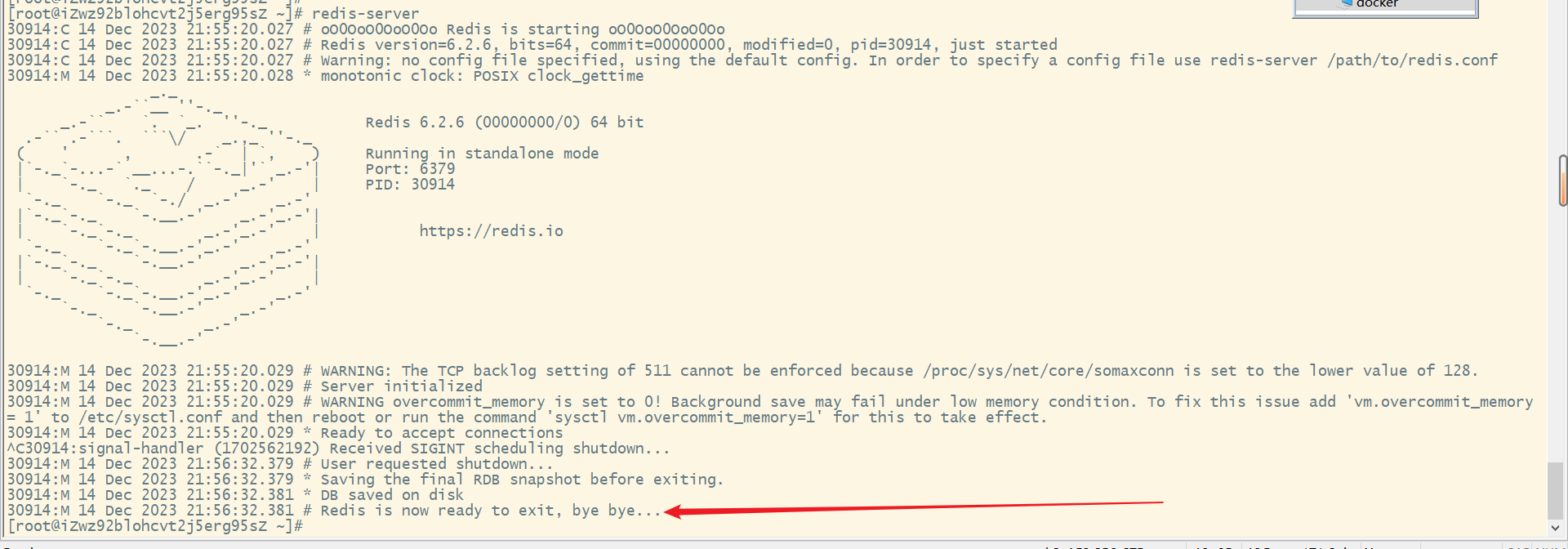
**redis-sentinel：是Redis的哨兵启动脚本**



### 8：前台运行启动Redis：redis-server



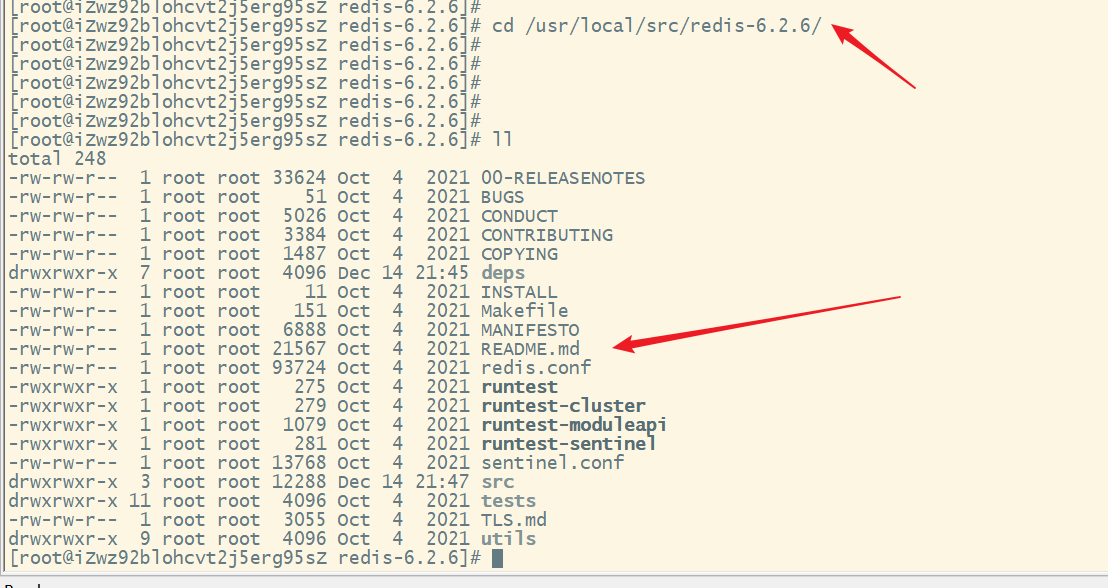
### 9：退出前台运行着的Redis，Ctrl + C



### 10：Redis后台启动

怎么样才能让Redis后台启动呢？必须要去修改Redis的配置文件，并且启动时要指定配置文件启动，配置文件就在我们之前解压的redis安装包下（`/usr/local/src/redis-6.2.6`），名字叫redis.conf。

#### 10.1：进入Redis安装目录：cd /usr/local/src/redis-6.2.6/



#### 10.2：备份配置文件：cp redis.conf redis.conf.bck

要修改配置文件，最好提前把配置文件备份一下，万一你修改配置文件错了，将来还有备份文件让你恢复



#### 10.3：修改允许访问的地址

Redis允许访问的地址默认是127.0.0.1，会导致只能在本地访问。修改为0.0.0.0则可以在任意IP访问，生产环境不要设置为0.0.0.0

**bind 0.0.0.0** 或者 **bind \* -::\***

#### 10.4：修改守护进程，修改为yes后即可后台运行

退出编辑模式下，/daemonize，再按回车键，可以快速定位到

**daemonize yes**

#### 10.5：修改密码，设置后访问Redis必须输入密码

退出编辑模式下，/requirepass foo，再按回车键，可以快速定位到

**requirepass 123456**

### 11：其他常见的配置

#### 11.1：监听的端口，默认就行，一般不修改

port 6379

#### 11.2：Redis的工作目录，默认是当前目录，意思就是你在哪里运行redis-server命令，哪里就是你的工作目录。也就是在什么目录运行redis-server命令，日志、持久化等文件会保存在这个目录。

dir .

#### 11.3：数据库数量，设置为1，代表只使用1个库，默认有16个库，编号0~15，这就是Redis与MySQL不同的地方，MySQL中数据库是可以随便创建的，但在Redis中数据库是固定的，默认是16个，当然数量你可以控制，可以在1到16之间做选择。

databases 1

#### 11.4：设置redis能够使用的最大内存

maxmemory 512mb

#### 11.5：日志文件，默认为空，不记录日志，可以指定日志文件名，需要注意的是，下面示例的日志文件只有些文件名，没有写路径，那么将来这个日志文件的路径就是在上面11.2的dir下

logfile "redis.log"

### 12：后台启动Redis

#### 12.1：进入redis安装目录 cd /usr/local/src/redis-6.2.6

#### 12.2：启动：

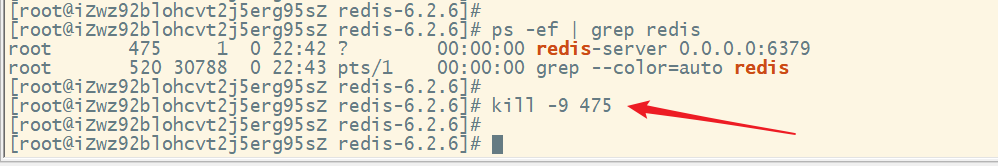
要想在启动的时候指定配置文件，只需要启动命令redis-server后面跟上配置文件就可以了，因为我们在Redis安装目录下启动Redis的，而redis.conf配置文件就在redis安装目录下，那么可以直接写redis.conf配置文件的文件名称。

#### redis-server redis.conf



### 13：停止Redis：kill -9 redis的进程id

### 14：配置Redis开启自启（先停掉运行中的Redis哦）



要想Redis开机自启，我们需要自己编写一个系统文件，即服务文件，把Redis加入到操作系统服务当中，以后就能开机自启了。

#### 13.1：新建一个系统服务文件：

vi /etc/systemd/system/redis.service



**[Unit]**

**Description=redis-server**

**After=network.target**

**[Service]**

**Type=forking**

**ExecStart=/usr/local/bin/redis-server /usr/local/src/redis-6.2.6/redis.conf**

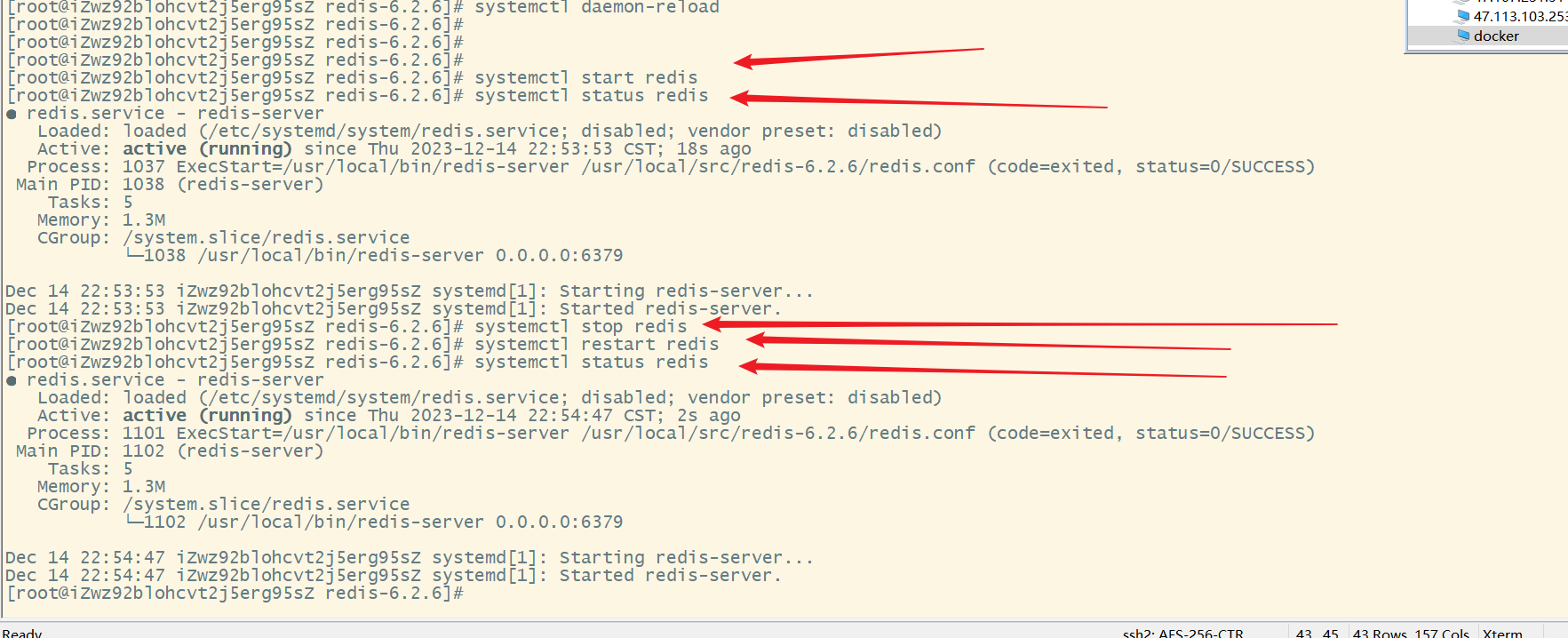
**PrivateTmp=true**

**[Install]**

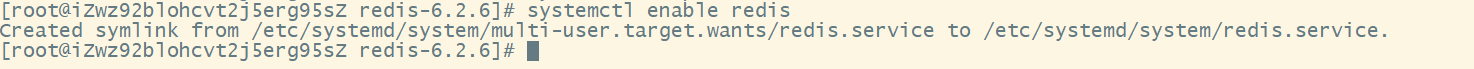
**WantedBy=multi-user.target**

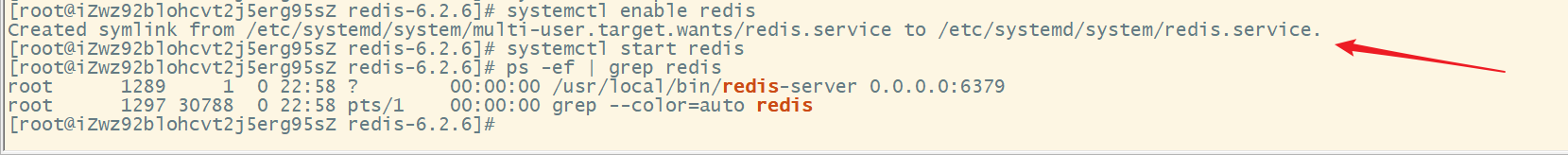
#### 14.2：让文件生效，重新加载系统服务：systemctl daemon-reload

#### 14.3：启动Redis:systemctl start redis



#### 14.4：让Redis服务开机自启（配置前先对Redis停止运行）：systemctl enable redis





## 访问Redis

安装完成Redis，我们就可以操作Redis，实现数据的CRUD了。操作Redis这需要用到Redis客户端，包括三种：

- 命令行客户端

- 图形化桌面客户端

- 编程客户端

命令行客户端在安装Redis的时候就已经安装好了，也就是redis-cli。

Redis安装完成后就自带了命令行客户端：redis-cli，使用方式如下：

redis-cli [options] [commonds]

其中常见的options有：

-h 127.0.0.1：指定要连接的redis节点的IP地址，默认是127.0.0.1，本机不用谢

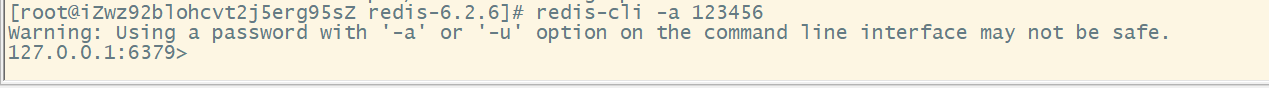
-p 6379：指定要连接的redis节点的端口，默认是6379，端口没修改过不用写

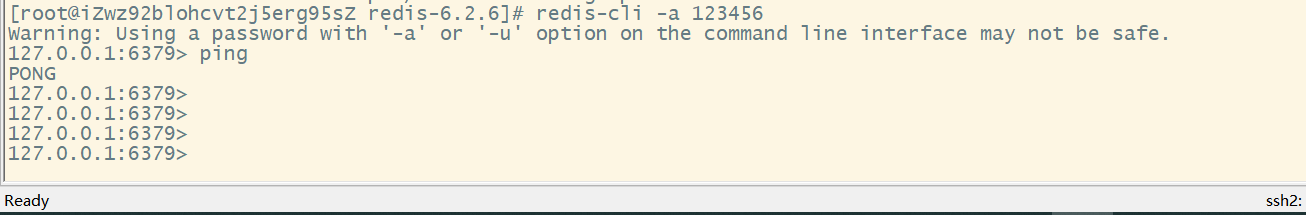
-a 123321：指定redis的访问密码

其中的commonds就是Redis的操作命令，例如：

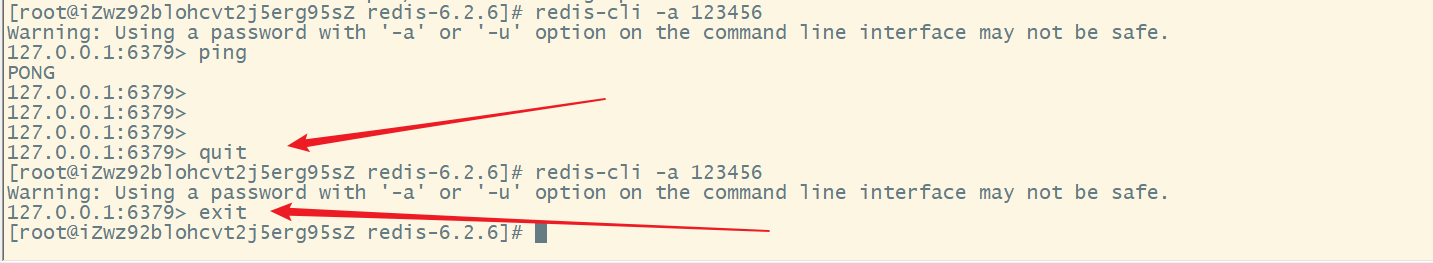
ping：与redis服务端做心跳测试，服务端正常会返回pong

一盘情况下都不会输入commonds，因为我们希望先连上Redis，当不指定commond时，会进入redis-cli的交互控制台。

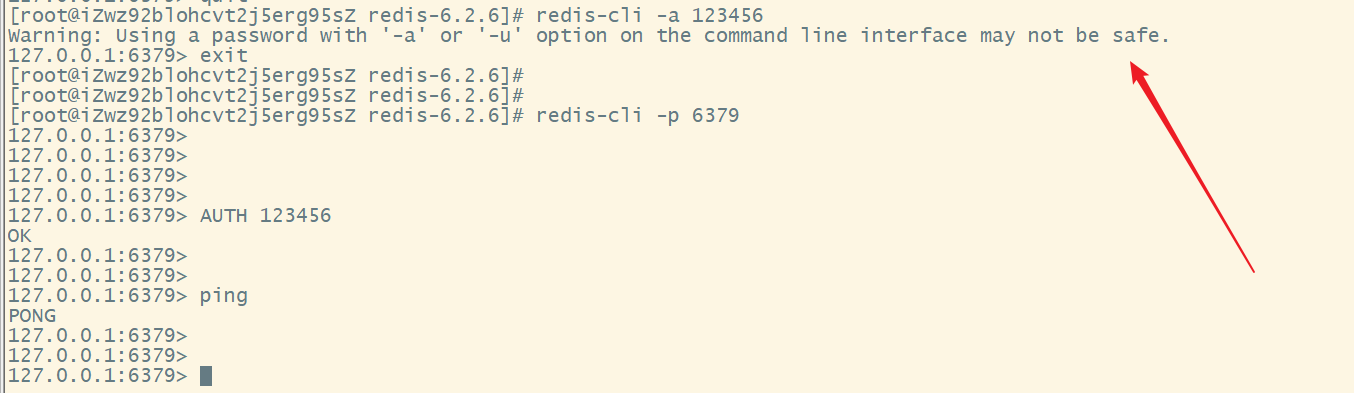




**输入exit或者quit都可以退出Redis连接**



**要想不出现警告，可以这样做：**



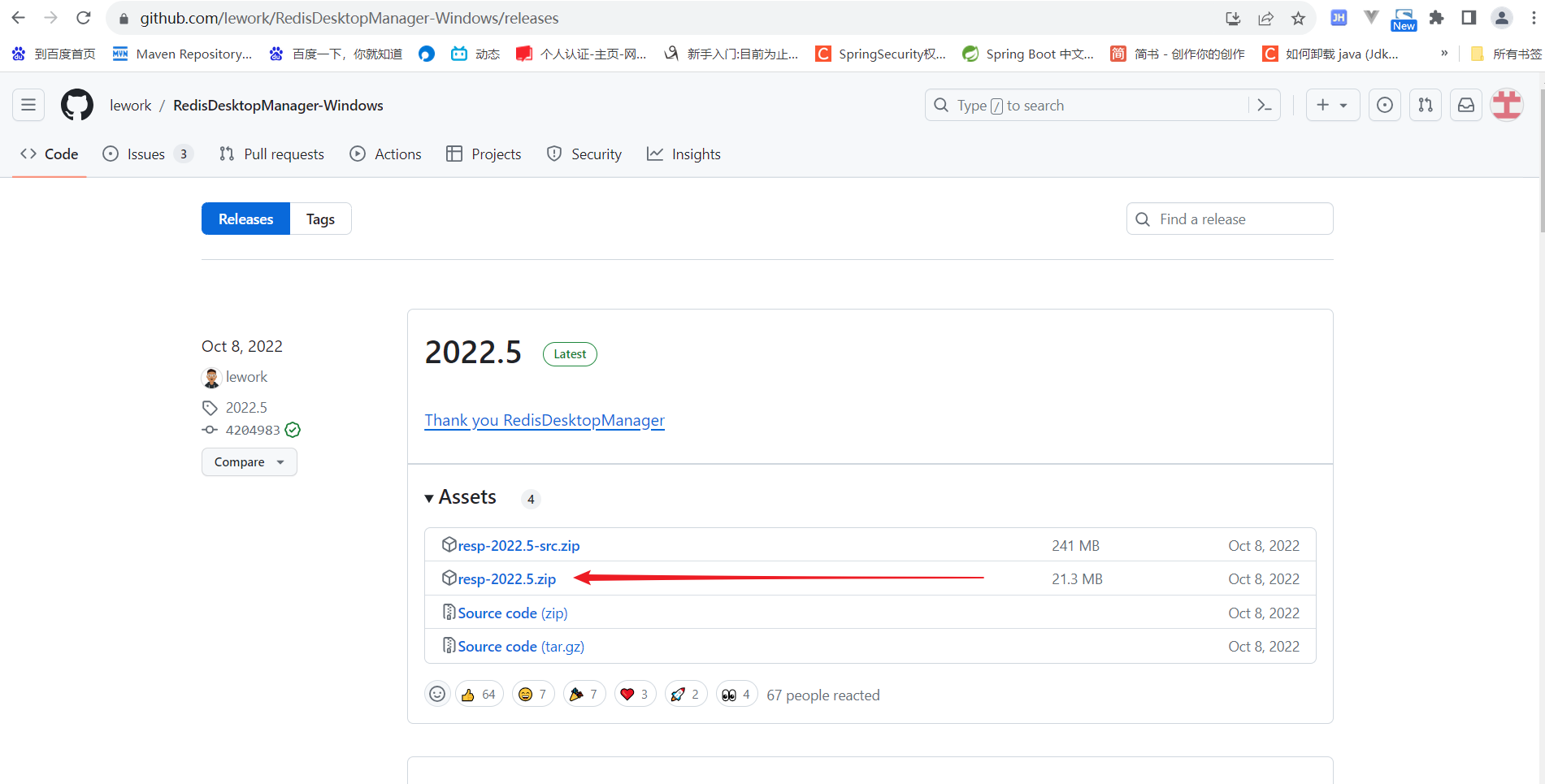
## 操作Redis

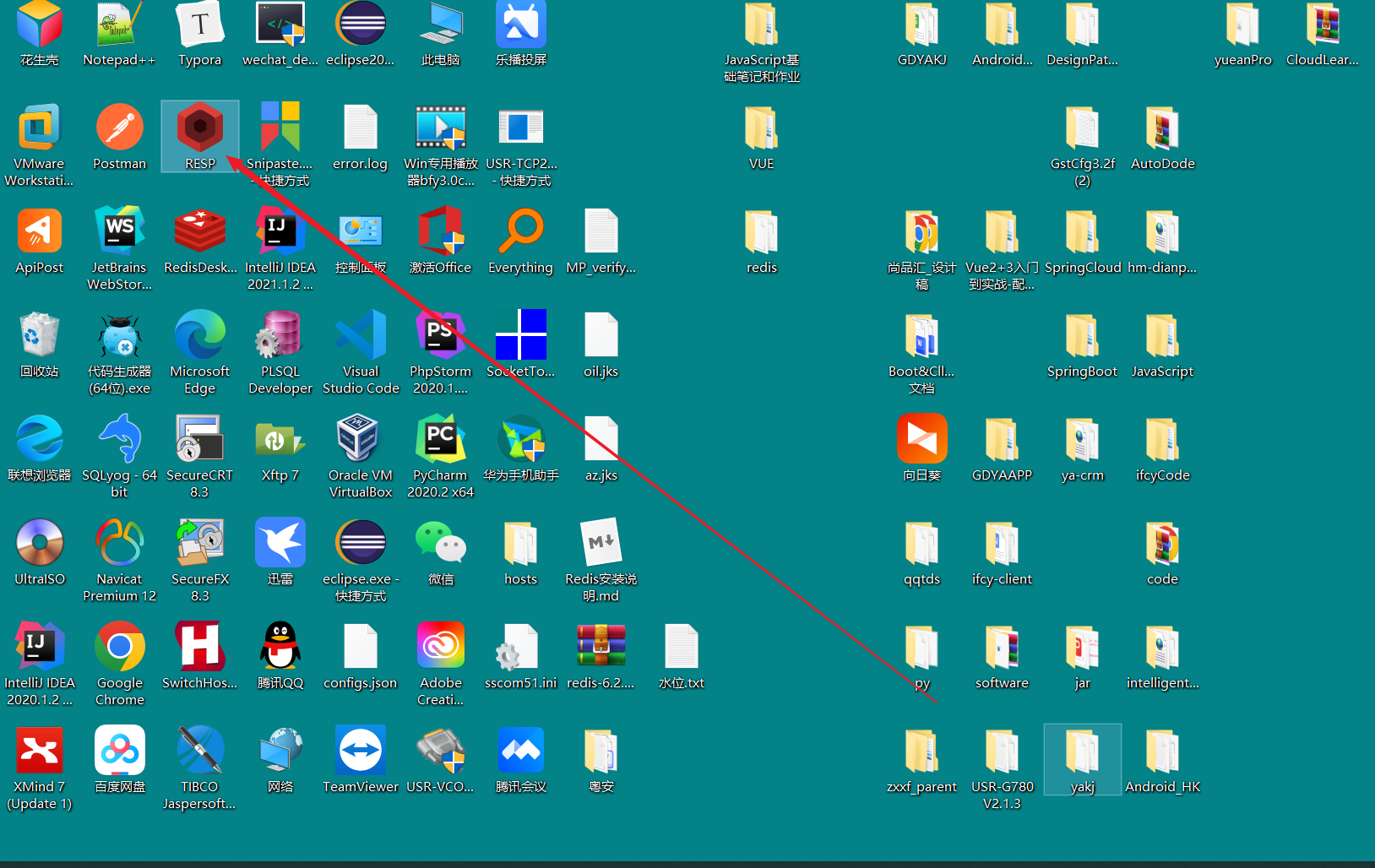
### 图形化桌面客户端RedisDesktopManager

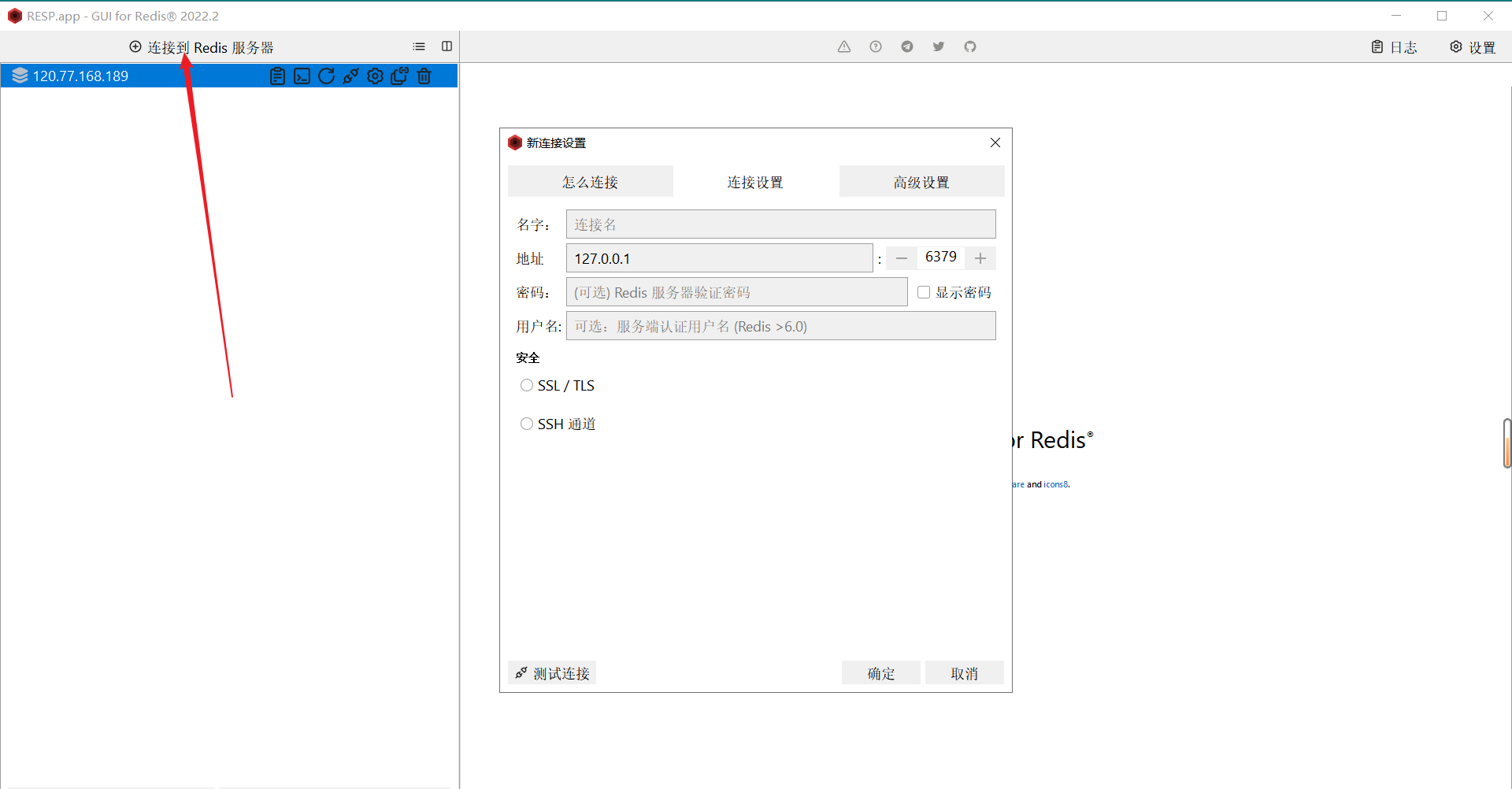
GitHub上的大神编写了Redis的图形化桌面客户端，地址：<https://github.com/uglide/RedisDesktopManager>

不过该仓库提供的是RedisDesktopManager的源码，并未提供windows安装包。在下面这个仓库可以找到安装包：

<https://github.com/lework/RedisDesktopManager-Windows/releases>







Redis通用命令：通用指令是部分数据类型的，都可以使用的指令

帮助文档：help @generic

HELP COMMAND：查看命令的帮助文档，通过help [command]可以查看一个命令的具体用法

Eg：help keys

KEYS：查看符合模板的所有key

Eg：keys \*

慎用，数据量大，耗时；Redis单线程，阻塞。

DEL：删除一个或多个指定的key，返回删除的键的数量

Eg：del key

Exists：判断key是否存在

Eg：exists key 20

EXPIRE：给key设置一个有限期，有效期到期时该key会被自动删除

Eg：expire name

TTL：查看一个key的剩余有效期，返回秒，-1永久有效，-2失效

Eg：ttl key

Redis基本类型：是值value的类型

1：string（就是字符串类型，）

2：hash（就是一个Map）

3：list（其实是一个链表，a指向b，b指向c，c指向d）

4：set（其实就是一个无序不重复的列表）

5：zset（又叫SortedSet，其实就是一个有序不重复的列表）

Key的结构的最佳实践：层级存储

Redis的key允许有多个单词形成层级结构，多个单词之间用冒号“：”隔开，格式如：项目名:业务名:类型:id

当然这个格式并非固定，可以根据自己的需求来删除或添加词条

例如我们的项目名称叫heima，有user和product两种不同类型的模块，我们可以这样定义key：

User相关的key：heima:user:1

product相关的key：heima:product:1

如果value是一个java对象，例如一个User对象，则可以将对象序列化为JSON字符串后存储：

heima:user:1 {“id”:1, “name”:”jack”, “age”:28}

String类型：

常见命令：

单个增、单个查：

SET：添加或者修改已经存在的一个String类型的键值对

GET：根据key获取String类型的value

批量增、批量查：

MSET：批量添加多个String类型的键值对

MGET：根据多个key获取多个String类型的value

操作value数值自增、自减：

INCR：让一个整形的key自增1

INCRBY：让一个整形的key自增并指定步长，比如 incrby num 2 让num值自增2

INCRBYFLOAT：让一个浮点类型的数字自增并指定步长

组合命令：

SETNX：添加一个String类型的简直对，前提是这个key不存在，否则不执行

SETEX：添加一个String类型的简直对，并且指定有效期

Hash类型：Redis中并没有MySql中的Table的概念，String类型是将对象序列化为JSON字符串后存储，但当需要修改对象某个字段时很不方便；Hash结构可以将对象中的每个字段独立存储，可以针对单个字段做CRUD

常见命令：

单个增、单个查：

HSET key field value：添加或者修改hash类型的key的field值

HGET key field：获取一个hash类型key的field值

批量增、批量查：

HMSET key field2 value1 field2 value2：批量添加多个hash类型的key的field值

HMGET key field1 field2：根据多个hash类型key的field值

HGETALL key：获取一个hash类型的key中的所有的field和value

HKEYS key：获取一个hash类型的key中的多有的field

HVALS key：取一个hash类型的key中的多有的value

HINCRBY key filed 步长：让一个hash类型key的字段值自增并指定步长

HSETNX：添加一个hash类型的key的field值，前提是这个field不存在，否则不执行

List类型：reids中的List类型与java中的LinkedList类似，可以看作是一个双向链表结构。既可以支持正向检索，也可以支持反向检索。

特征也与LinkedList类似：

有序；

元素可以重复；

插入和删除块；

查询速度一般

适用场景：对顺序有要求的场景，比如朋友圈点赞、朋友圈的评论，有谁先谁后的

常见命令：

LPUSH key element...：向列表左侧插入一个或多个元素

LPOP key：移除并返回列表左侧的第一个元素，没有则返回nil

RPUSH key element...：向列表右侧插入一个或多个元素

RPOP key：移除并返回列表右侧的第一个元素，没有则返回nil

LRANGE key star end：返回一段角标范围内的所有元素

BLPOP key和BRPOP key：与LPOP和RPOP类似，只不过在没有元素时会等待指定时间，而不是直接返回nil

Set类型：reids中的List类型与java中的HashSet类似，可以看作是一个value为null的HashMap。因为也是一个hash表，因此具备与HashSet类型的特征：

无序；

元素不可以重复；

查询速度快；

支持交集、并集、差集等功能

适用场景：好友列表、共同好友、关注列表等，社交型的应用中使用比较广泛。

常见命令：

单个集合的操作：

SADD key member...：向set中添加一个或多个元素

SREM key member...：移除set中的指定元素

SCARD key：返回set中元素的个数

SISMEMBER key member：判断一个元素是否存在于set中

SISMEMBERS：获取set中的多有元素

多个集合的操作：

SINTER key1 key2....：求key1与key2的交集

SDIFF key1 key2....：求key1与key2的差集：就是key1有 key2没有

SUNION key1 key2....：求key1与key2的并集

SortedSet类型：redis中的SortedSet是一个可排序的set集合，与java中的TreeSet有些类似。但底层数据结构却区别很大。SortedSet中的每一个元素都带有一个score属性，可以基于score属性对元素排序，底层的实现是一个跳表（SkipList）加hash表。

SortedSet具备的特性：

可排序；

元素不重复；

查询速度快

因为SortedSet的可排序性，经常被用来实现排行榜这样的功能。

常见命令：（排名默认都是按照升序的排名，如果要降序则在命令的Z后面添加REV（反转的意思）即可）

单个集合的操作：

ZADD key score member：添加一个或者多个元素到sorted set，如果已经存在则更新其score值中

ZREM key member：移除sorted set中的一个指定元素

ZSCORE key member：获取sorted set中的指定元素的score值

ZRANK key member：获取sorted set中的指定元素的排名

ZCARD key：返回sorted set中元素的个数

ZCOUNT key min max：统计sorted set中score值在给定范围内的所有元素的个数

ZINCRBY key increment meMber：让sorted set的指定元素自增，步长为指定的increment值

ZRANGE key min max：按照score排序后，获取指定排名范围内的元素

ZRANGEBYSCORE key min max：按照score排序后，获取指定score范围内的元素

多个集合的操作：

ZDIFF、ZINTER、ZUNION：求差集、交集、并集