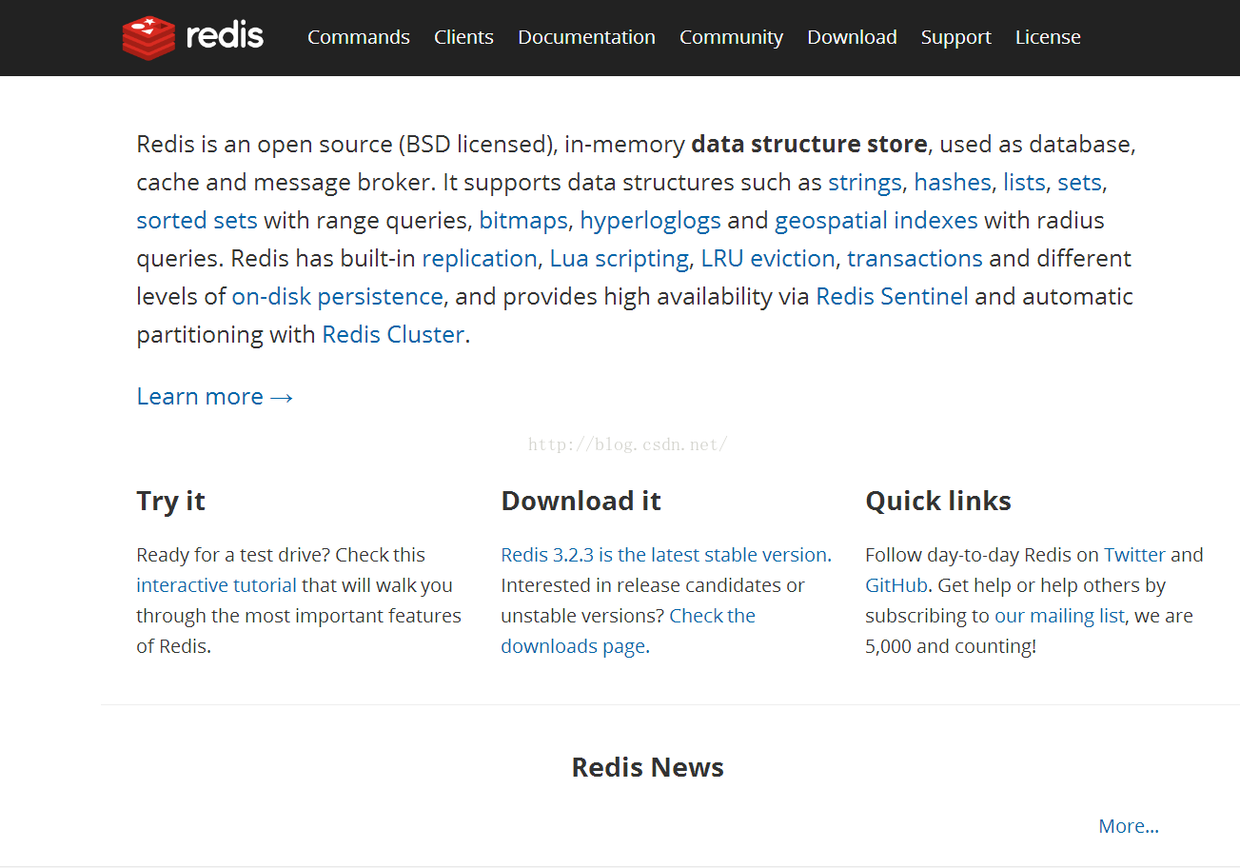
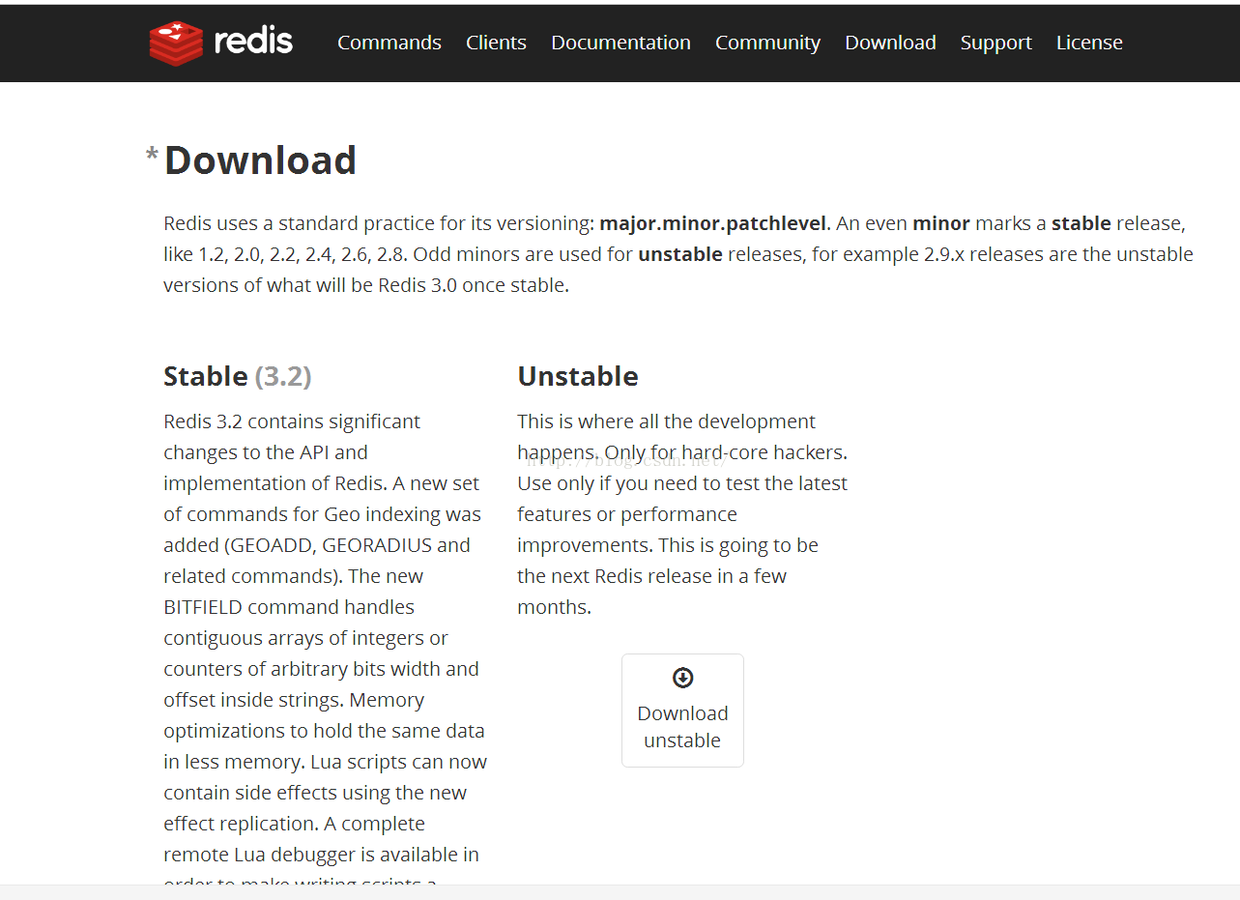
**Linux下redis安装链接 ：**[**转到**](http://blog.csdn.net/jinwufeiyang/article/details/70307611)

**Redis**是一个key-value存储系统。和Memcached类似，它支持存储的value类型相对更多，包括string(字符串)、list(链表)、set(集合)、zset(sorted set --有序集合)和hash（哈希类型）。这些数据类型都支持push/pop、add/remove及取交集并集和差集及更丰富的操作，而且这些操作都是原子性的。在此基础上，redis支持各种不同方式的排序。与memcached一样，为了保证效率，数据都是缓存在内存中。区别的是redis会周期性的把更新的数据写入磁盘或者把修改操作写入追加的记录文件，并且在此基础上实现了master-slave。

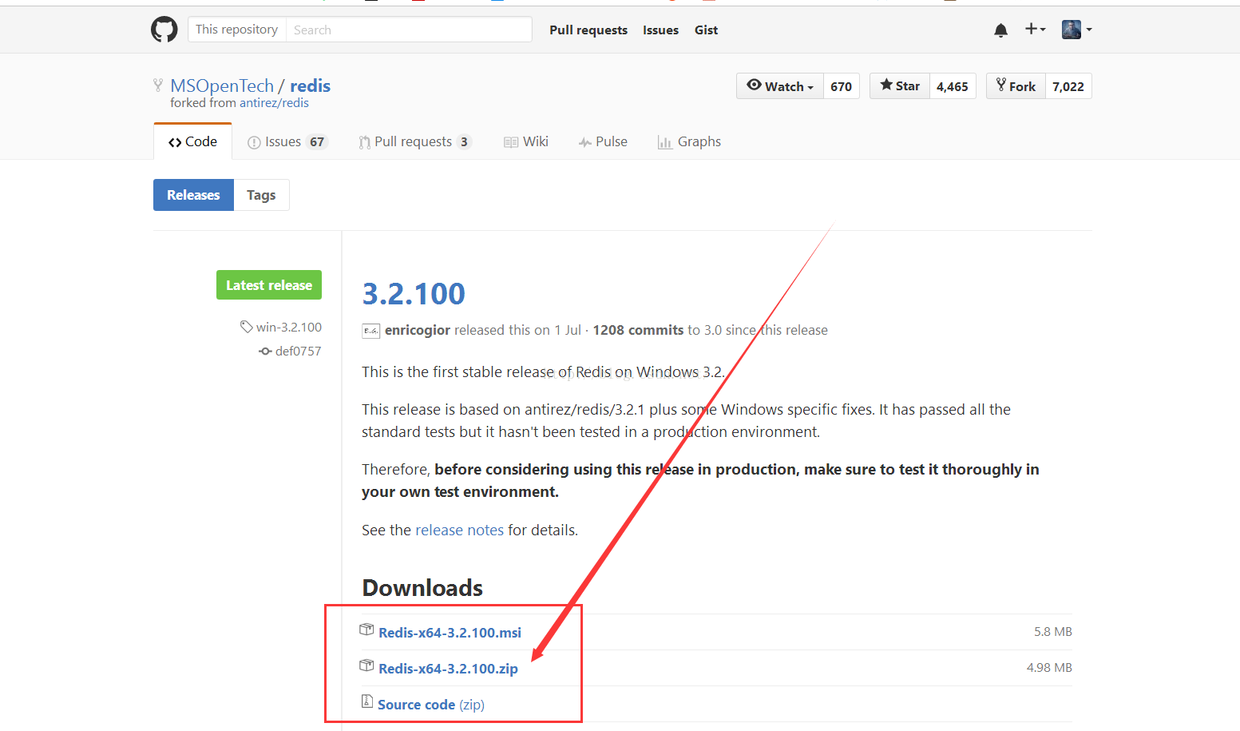
**1、安装Redis**  
官方网站：<http://redis.io/>



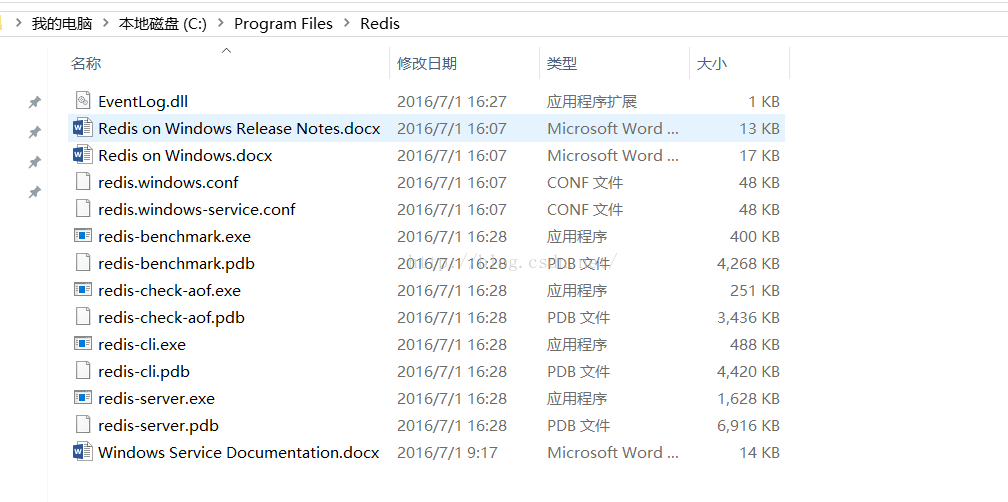
官方下载：<http://redis.io/download> 可以根据需要下载不同版本



windows版：<https://github.com/MSOpenTech/redis/releases>



我选择的是第三种，在github上下载的zip压缩文件。  
下载完成后，在C盘【Program Files】目录下新建文件夹【Redis】，右键解压Redis ZIP包，把所有文件解压到redis文件夹中。



文件介绍：

redis-benchmark.exe #基准测试

redis-check-aof.exe # aof

redischeck-dump.exe # dump

redis-cli.exe # 客户端

redis-server.exe # 服务器

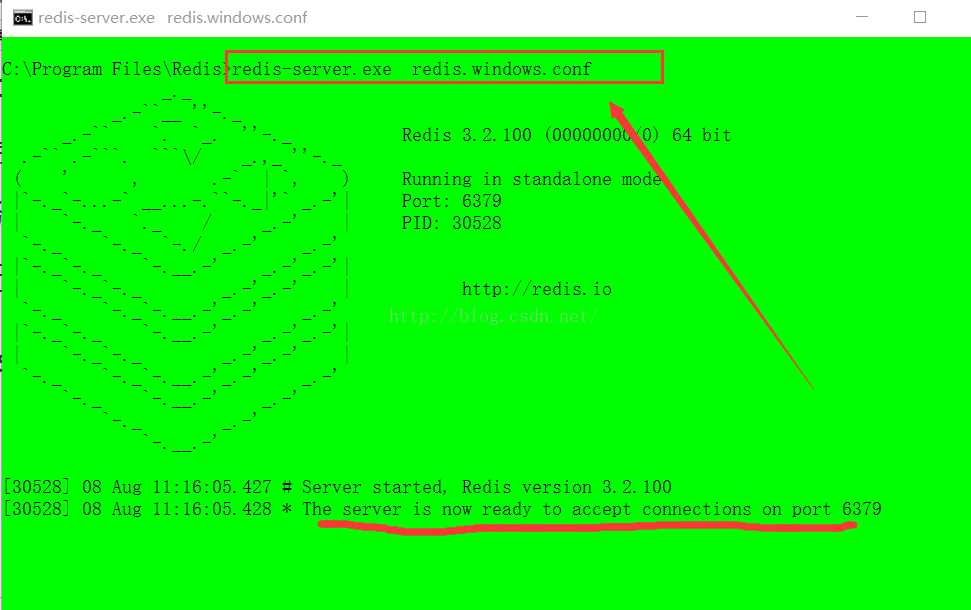
redis.windows.conf # 配置文件

**2、启动Redis**

windows 运行（快捷键：按住shift + 鼠标右键），选择在此处打开命令窗口，进入cmd操作系统窗口。

使用命令【**redis-server.exe redis.windows.conf**】，启动redis 服务。

**服务启动成功状态**



不要问我为啥颜色是绿色的，自己去设置，个人感觉护眼（心理作用，哈哈 ^\_^）。

启动redis服务的doc窗口，不用关闭，因为服务需要一直执行，关闭服务，直接关闭窗口就行。

**3、测试使用**

新打开一个doc窗口，用自带的客户端工具进行测试 命令【**redis-\*\***cli.exe\*\*】,详细操作如下。。



**4.Java开发包Jedis**  
Jedis ：<http://www.oschina.net/p/jedis> （Redis的官方首选Java开发包）

<!--Redis -->

<dependency>

<groupId>redis.clients</groupId>

<artifactId>jedis</artifactId>

<version>2.0.0</version>

<type>jar</type>

<scope>compile</scope>

</dependency>

测试例子原帖：<http://flychao88.iteye.com/blog/1527163>

package com.lujianing.utils;

import org.junit.Before;

import org.junit.Test;

import redis.clients.jedis.Jedis;

import redis.clients.jedis.JedisPool;

import redis.clients.jedis.JedisPoolConfig;

import java.util.HashMap;

import java.util.Iterator;

import java.util.List;

import java.util.Map;

/\*\*

\* Created by lujianing on 14-2-28.

\*/

public class JedisUtilTest {

JedisPool pool;

Jedis jedis;

@Before

public void setUp() {

pool = new JedisPool(new JedisPoolConfig(), "192.168.10.61");

jedis = pool.getResource();

// jedis.auth("password");

}

@Test

public void testGet(){

System.out.println(jedis.get("lu"));

}

/\*\*

\* Redis存储初级的字符串

\* CRUD

\*/

@Test

public void testBasicString(){

//-----添加数据----------

jedis.set("name","minxr");//向key-->name中放入了value-->minxr

System.out.println(jedis.get("name"));//执行结果：minxr

//-----修改数据-----------

//1、在原来基础上修改

jedis.append("name","jarorwar"); //很直观，类似map 将jarorwar append到已经有的value之后

System.out.println(jedis.get("name"));//执行结果:minxrjarorwar

//2、直接覆盖原来的数据

jedis.set("name","闵晓荣");

System.out.println(jedis.get("name"));//执行结果：闵晓荣

//删除key对应的记录

jedis.del("name");

System.out.println(jedis.get("name"));//执行结果：null

/\*\*

\* mset相当于

\* jedis.set("name","minxr");

\* jedis.set("jarorwar","闵晓荣");

\*/

jedis.mset("name","minxr","jarorwar","闵晓荣");

System.out.println(jedis.mget("name","jarorwar"));

}

/\*\*

\* jedis操作Map

\*/

@Test

public void testMap(){

Map<String,String> user=new HashMap<String,String>();

user.put("name","minxr");

user.put("pwd","password");

jedis.hmset("user",user);

//取出user中的name，执行结果:[minxr]-->注意结果是一个泛型的List

//第一个参数是存入redis中map对象的key，后面跟的是放入map中的对象的key，后面的key可以跟多个，是可变参数

List<String> rsmap = jedis.hmget("user", "name");

System.out.println(rsmap);

//删除map中的某个键值

// jedis.hdel("user","pwd");

System.out.println(jedis.hmget("user", "pwd")); //因为删除了，所以返回的是null

System.out.println(jedis.hlen("user")); //返回key为user的键中存放的值的个数1

System.out.println(jedis.exists("user"));//是否存在key为user的记录 返回true

System.out.println(jedis.hkeys("user"));//返回map对象中的所有key [pwd, name]

System.out.println(jedis.hvals("user"));//返回map对象中的所有value [minxr, password]

Iterator<String> iter=jedis.hkeys("user").iterator();

while (iter.hasNext()){

String key = iter.next(); System.out.println(key+":"+jedis.hmget("user",key));

}

}

/\*\*

\* jedis操作List列表允许重复元素

\*对空列表 mylist 执行命令 LPUSH mylist a b c ，列表的值是 c b a ，

\*这等同于原子性地执

\*行 LPUSH mylist a 、 LPUSH mylist b 和 LPUSH mylist c 三个命令

\*/

@Test

public void testList(){

//开始前，先移除所有的内容

jedis.del("java framework");

System.out.println(jedis.lrange("java framework",0,-1));

//先向key java framework中存放三条数据

jedis.lpush("java framework","spring");

jedis.lpush("java framework","struts");

jedis.lpush("java framework","hibernate");

//再取出所有数据jedis.lrange是按范围取出，

// 第一个是key，第二个是起始位置，第三个是结束位置，jedis.llen获取长度 -1表示取得所有

System.out.println(jedis.lrange("java framework",0,-1));

}

/\*\*

\* jedis操作Set

\*/

@Test

public void testSet(){

//添加

jedis.sadd("sname","minxr");

jedis.sadd("sname","jarorwar");

jedis.sadd("sname","闵晓荣");

jedis.sadd("sanme","noname");

//移除noname

jedis.srem("sname","noname");

System.out.println(jedis.smembers("sname"));//获取所有加入的value

System.out.println(jedis.sismember("sname", "minxr"));//判断 minxr 是否是sname集合的元素

System.out.println(jedis.srandmember("sname"));

System.out.println(jedis.scard("sname"));//返回集合的元素个数

}

@Test

public void test() throws InterruptedException {

//keys中传入的可以用通配符

System.out.println(jedis.keys("\*")); //返回当前库中所有的key [sose, sanme, name, jarorwar, foo, sname, java framework, user, braand]

System.out.println(jedis.keys("\*name"));//返回的sname [sname, name]

System.out.println(jedis.del("sanmdde"));//删除key为sanmdde的对象 删除成功返回1 删除失败（或者不存在）返回 0

System.out.println(jedis.ttl("sname"));//返回给定key的有效时间，如果是-1则表示永远有效

jedis.setex("timekey", 10, "min");//通过此方法，可以指定key的存活（有效时间） 时间为秒

Thread.sleep(5000);//睡眠5秒后，剩余时间将为<=5

System.out.println(jedis.ttl("timekey")); //输出结果为5

jedis.setex("timekey", 1, "min"); //设为1后，下面再看剩余时间就是1了

System.out.println(jedis.ttl("timekey")); //输出结果为1

System.out.println(jedis.exists("key"));//检查key是否存在 System.out.println(jedis.rename("timekey","time"));

System.out.println(jedis.get("timekey"));//因为移除，返回为null

System.out.println(jedis.get("time")); //因为将timekey 重命名为time 所以可以取得值 min

//jedis 排序

//注意，此处的rpush和lpush是List的操作。是一个双向链表（但从表现来看的）

jedis.del("a");//先清除数据，再加入数据进行测试

jedis.rpush("a", "1");

jedis.lpush("a","6");

jedis.lpush("a","3");

jedis.lpush("a","9");

System.out.println(jedis.lrange("a",0,-1));// [9, 3, 6, 1]

System.out.println(jedis.sort("a")); //[1, 3, 6, 9] //输入排序后结果

System.out.println(jedis.lrange("a",0,-1));

}

}

Redis会定时 保存数据到硬盘上。

**$**redis-cli -h 127.0.0.1 -p 6379

windows下 ：redis-cli.exe -h 127.0.0.1 -p 6379  
   
redis 127.0.0.1:6379>

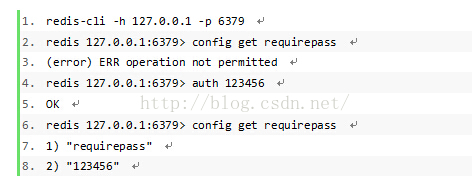
远程服务停止：

**$**redis-cli -h 172.168.10.254 -p6379 shutdown

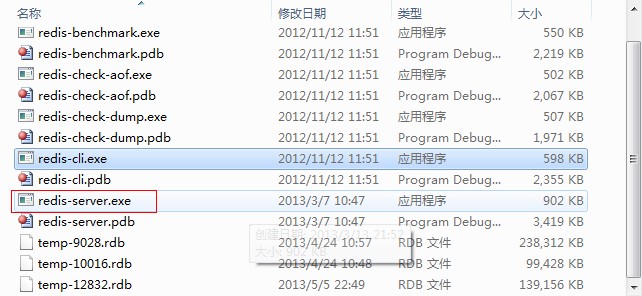
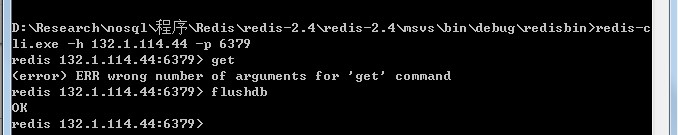
2) 有权限控制时(加上-a 密码)：

redis-cli -h 127.0.0.1 -p 6379 -a 123456

除了在登录时通过 -a 参数制定密码外，还可以登录时不指定密码，而在执行操作前进行认证。



Redis默认端口号为127.0.0.1，端口号默认为：6379。

此处本机访问远程IP为132.1.114.44的计算机，则首先要在已经安装了Redis的远程计算机上打开其服务器，redis.server.exe[](http://photo.blog.sina.com.cn/showpic.html#blogid=4a552a740101kdfz&url=http://s15.sinaimg.cn/orignal/4a552a74gdc2bda7800ee)  
接下来在本机运行redis.cli.exe，也可以通过命令行实现：输入-h 远程计算机IP -p 6379即可连接：  
[](http://photo.blog.sina.com.cn/showpic.html#blogid=4a552a740101kdfz&url=http://s13.sinaimg.cn/orignal/4a552a74g7c6ac9bc173c)

OK了，接下来如果想用自己写的客户端什么的连接远程Redis数据库也只需要输入远程计算机的IP就可以了~

以下内容转自:http://blog.csdn.net/chen88358323/article/details/47318303

# Redis系列-远程连接redis

转载 2015年08月06日 16:01:39

* 标签：
* [redis](http://so.csdn.net/so/search/s.do?q=redis&t=blog)
* 56105

转自:http://www.linuxidc.com/Linux/2012-12/75693.htm

假设两台redis服务器，ip分别为：192.168.1.101和192.168.1.103，如何在101上通过redis-cli访问103上的redis呢？在远程连接103之前，先讲下redis-cli的几个关键参数：

用法：redis-cli [OPTIONS] [cmd [arg [arg ...]]]

-h <主机ip>，默认是127.0.0.1

-p <端口>，默认是6379

-a <密码>，如果redis加锁，需要传递密码

--help，显示帮助信息

通过对rendis-cli用法介绍，在101上连接103应该很简单：

[root@linuxidc001 ~]# redis-cli -h 192.168.1.103 -p 6379

redis 192.168.1.103:6379>

在101上对103设置个个string值 user.1.name=zhangsan

redis 192.168.1.103:6379> set user.1.name zhangsan

OK

看到ok，表明设置成功了。然后直接在103上登陆，看能不能获取到这个值。

redis 192.168.1.103:6379>  keys \*

redis 192.168.1.103:6379>  select 1

### 1、连接操作相关的命令

* quit：关闭连接（connection）
* auth：简单密码认证

### 2、对value操作的命令

* exists(key)：确认一个key是否存在
* del(key)：删除一个key
* type(key)：返回值的类型
* keys(pattern)：返回满足给定pattern的所有key
* randomkey：随机返回key空间的一个key
* rename(oldname, newname)：将key由oldname重命名为newname，若newname存在则删除newname表示的key
* dbsize：返回当前数据库中key的数目
* expire：设定一个key的活动时间（s）
* ttl：获得一个key的活动时间
* select(index)：按索引查询
* move(key, dbindex)：将当前数据库中的key转移到有dbindex索引的数据库
* flushdb：删除当前选择数据库中的所有key
* flushall：删除所有数据库中的所有key

### 3、对String操作的命令

* set(key, value)：给数据库中名称为key的string赋予值value
* get(key)：返回数据库中名称为key的string的value
* getset(key, value)：给名称为key的string赋予上一次的value
* mget(key1, key2,…, key N)：返回库中多个string（它们的名称为key1，key2…）的value
* setnx(key, value)：如果不存在名称为key的string，则向库中添加string，名称为key，值为value
* setex(key, time, value)：向库中添加string（名称为key，值为value）同时，设定过期时间time
* mset(key1, value1, key2, value2,…key N, value N)：同时给多个string赋值，名称为key i的string赋值value i
* msetnx(key1, value1, key2, value2,…key N, value N)：如果所有名称为key i的string都不存在，则向库中添加string，名称key i赋值为value i
* incr(key)：名称为key的string增1操作
* incrby(key, integer)：名称为key的string增加integer
* decr(key)：名称为key的string减1操作
* decrby(key, integer)：名称为key的string减少integer
* append(key, value)：名称为key的string的值附加value
* substr(key, start, end)：返回名称为key的string的value的子串

### 4、对List操作的命令

* rpush(key, value)：在名称为key的list尾添加一个值为value的元素
* lpush(key, value)：在名称为key的list头添加一个值为value的 元素
* llen(key)：返回名称为key的list的长度
* lrange(key, start, end)：返回名称为key的list中start至end之间的元素（下标从0开始，下同）
* ltrim(key, start, end)：截取名称为key的list，保留start至end之间的元素
* lindex(key, index)：返回名称为key的list中index位置的元素
* lset(key, index, value)：给名称为key的list中index位置的元素赋值为value
* lrem(key, count, value)：删除count个名称为key的list中值为value的元素。count为0，删除所有值为value的元素，count>0从 头至尾删除count个值为value的元素，count<0从尾到头删除|count|个值为value的元素。 lpop(key)：返回并删除名称为key的list中的首元素 rpop(key)：返回并删除名称为key的list中的尾元素 blpop(key1, key2,… key N, timeout)：lpop命令的block版本。即当timeout为0时，若遇到名称为key i的list不存在或该list为空，则命令结束。如果timeout>0，则遇到上述情况时，等待timeout秒，如果问题没有解决，则对 keyi+1开始的list执行pop操作。
* brpop(key1, key2,… key N, timeout)：rpop的block版本。参考上一命令。
* rpoplpush(srckey, dstkey)：返回并删除名称为srckey的list的尾元素，并将该元素添加到名称为dstkey的list的头部

### 5、对Set操作的命令

* sadd(key, member)：向名称为key的set中添加元素member
* srem(key, member) ：删除名称为key的set中的元素member
* spop(key) ：随机返回并删除名称为key的set中一个元素
* smove(srckey, dstkey, member) ：将member元素从名称为srckey的集合移到名称为dstkey的集合
* scard(key) ：返回名称为key的set的基数
* sismember(key, member) ：测试member是否是名称为key的set的元素
* sinter(key1, key2,…key N) ：求交集
* sinterstore(dstkey, key1, key2,…key N) ：求交集并将交集保存到dstkey的集合
* sunion(key1, key2,…key N) ：求并集
* sunionstore(dstkey, key1, key2,…key N) ：求并集并将并集保存到dstkey的集合
* sdiff(key1, key2,…key N) ：求差集
* sdiffstore(dstkey, key1, key2,…key N) ：求差集并将差集保存到dstkey的集合
* smembers(key) ：返回名称为key的set的所有元素
* srandmember(key) ：随机返回名称为key的set的一个元素

### 6、对zset（sorted set）操作的命令

* zadd(key, score, member)：向名称为key的zset中添加元素member，score用于排序。如果该元素已经存在，则根据score更新该元素的顺序。
* zrem(key, member) ：删除名称为key的zset中的元素member
* zincrby(key, increment, member) ：如果在名称为key的zset中已经存在元素member，则该元素的score增加increment；否则向集合中添加该元素，其score的值为increment
* zrank(key, member) ：返回名称为key的zset（元素已按score从小到大排序）中member元素的rank（即index，从0开始），若没有member元素，返回“nil”
* zrevrank(key, member) ：返回名称为key的zset（元素已按score从大到小排序）中member元素的rank（即index，从0开始），若没有member元素，返回“nil”
* zrange(key, start, end)：返回名称为key的zset（元素已按score从小到大排序）中的index从start到end的所有元素
* zrevrange(key, start, end)：返回名称为key的zset（元素已按score从大到小排序）中的index从start到end的所有元素
* zrangebyscore(key, min, max)：返回名称为key的zset中score >= min且score <= max的所有元素 zcard(key)：返回名称为key的zset的基数 zscore(key, element)：返回名称为key的zset中元素element的score zremrangebyrank(key, min, max)：删除名称为key的zset中rank >= min且rank <= max的所有元素 zremrangebyscore(key, min, max) ：删除名称为key的zset中score >= min且score <= max的所有元素
* zunionstore / zinterstore(dstkeyN, key1,…,keyN, WEIGHTS w1,…wN, AGGREGATE SUM|MIN|MAX)：对N个zset求并集和交集，并将最后的集合保存在dstkeyN中。对于集合中每一个元素的score，在进行 AGGREGATE运算前，都要乘以对于的WEIGHT参数。如果没有提供WEIGHT，默认为1。默认的AGGREGATE是SUM，即结果集合中元素 的score是所有集合对应元素进行SUM运算的值，而MIN和MAX是指，结果集合中元素的score是所有集合对应元素中最小值和最大值。

### 7、对Hash操作的命令

* hset(key, field, value)：向名称为key的hash中添加元素field<—>value
* hget(key, field)：返回名称为key的hash中field对应的value
* hmget(key, field1, …,field N)：返回名称为key的hash中field i对应的value
* hmset(key, field1, value1,…,field N, value N)：向名称为key的hash中添加元素field i<—>value i
* hincrby(key, field, integer)：将名称为key的hash中field的value增加integer
* hexists(key, field)：名称为key的hash中是否存在键为field的域
* hdel(key, field)：删除名称为key的hash中键为field的域
* hlen(key)：返回名称为key的hash中元素个数
* hkeys(key)：返回名称为key的hash中所有键
* hvals(key)：返回名称为key的hash中所有键对应的value
* hgetall(key)：返回名称为key的hash中所有的键（field）及其对应的value

### 8、持久化

* save：将数据同步保存到磁盘
* bgsave：将数据异步保存到磁盘
* lastsave：返回上次成功将数据保存到磁盘的Unix时戳
* shundown：将数据同步保存到磁盘，然后关闭服务

### 9、远程服务控制

* info：提供服务器的信息和统计
* monitor：实时转储收到的请求
* slaveof：改变复制策略设置
* config：在运行时配置Redis服务器