《Zookeeper分布式与Dubbo微服务学习笔记》

作者：吴里庆庆

*如果说大脑是我们的一级缓存，那么学习笔记就是我们的二级缓存。好记性不如烂笔头，因此本文主要记录了我实战Zookeeper与Dubbo入门学习过程。如有不足之处，欢迎指正！*

|  |
| --- |
| 联系邮箱：[wuqingvika@qq.com](mailto:wuqingvika@qq.com).  微博：@吴里庆庆  个人博客：<https://wuqingvika.github.io/>  或<https://wuqingvikaleo.github.io/> |

*欢迎一起学习交流！*

目录

[一、Jdk8安装 1](#_Toc513384299)

[1.1、官网下载Linux安装包 1](#_Toc513384300)

[1.2、启动ftp传输工具 2](#_Toc513384301)

[1.3、解压Jdk8压缩包 4](#_Toc513384302)

[1.4、添加jdk配置信息 5](#_Toc513384303)

[二、单机安装Zookeeper: 7](#_Toc513384304)

[2.1、去官网下载安装包 7](#_Toc513384305)

[2.2、启动ftp传输工具 9](#_Toc513384306)

[2.3、解压安装zookeeper 10](#_Toc513384307)

[2.4、添加Zookeeper配置信息 11](#_Toc513384308)

[三、操作Zookeeper 14](#_Toc513384309)

[3.1、启动zkServer 14](#_Toc513384310)

[3.2、查看状态 14](#_Toc513384311)

[3.3、启动zkCli 14](#_Toc513384312)

[3.4、查看帮助 15](#_Toc513384313)

[3.5、 ls&ls2命令 15](#_Toc513384314)

[3.6、get&stat命令 16](#_Toc513384315)

[3.7、create创建持久/临时/有序节点 17](#_Toc513384316)

[3.7.1 create持久 17](#_Toc513384317)

[3.7.2 Create –e 临时 17](#_Toc513384318)

[3.7.3 create –s 有序 18](#_Toc513384319)

[3.8、set&delete命令 19](#_Toc513384320)

[3.8.1 set操作 19](#_Toc513384321)

[3.8.2 set乐观锁操作 19](#_Toc513384322)

[3.8.3 delete操作 20](#_Toc513384323)

[3.9、watcher命令行操作 20](#_Toc513384324)

[3.9.1 父节点操作 20](#_Toc513384325)

[3.9.2 子节点操作 22](#_Toc513384326)

[四、zk特性 24](#_Toc513384327)

[4.1、session的基本原理 24](#_Toc513384328)

[4.1.1 原理 24](#_Toc513384329)

[4.2、理解watcher机制 24](#_Toc513384330)

[4.2.1 机制 24](#_Toc513384331)

[4.2.2 事件类型 25](#_Toc513384332)

[五、ACL(access control lists)权限控制 26](#_Toc513384333)

[5.1、ACL构成 26](#_Toc513384334)

[5.2、Acl命令行 27](#_Toc513384335)

[5.2.1 getAcl 27](#_Toc513384336)

[5.2.2 world 27](#_Toc513384337)

[5.2.3 Auth 29](#_Toc513384338)

[5.2.4 digest 31](#_Toc513384339)

[5.2.5 使用Ip 31](#_Toc513384340)

[5.2.6 使用Super 32](#_Toc513384341)

[5.3Acl使用场景 33](#_Toc513384342)

[六、ZK四字命令 34](#_Toc513384343)

[6.1 Four Letter words 34](#_Toc513384344)

[6.2常用四字命令 34](#_Toc513384345)

[6.2.1 stat 34](#_Toc513384346)

[6.2.2 ruok 35](#_Toc513384347)

[6.2.3 dump 35](#_Toc513384348)

[6.2.4 conf 35](#_Toc513384349)

[6.2.5 cons 36](#_Toc513384350)

[6.2.6 envi 36](#_Toc513384351)

[6.2.7 mntr 36](#_Toc513384352)

[6.2.8 wchs 37](#_Toc513384353)

[6.2.9 wchc&wchp 38](#_Toc513384354)

[七、搭建集群 40](#_Toc513384355)

[7.1、单机伪分布式安装zookeeper集群 40](#_Toc513384356)

[7.2、测试集群角色以及选举 44](#_Toc513384357)

[八、ZooKeeper原生Java API使用 46](#_Toc513384358)

[8.1、建立客户端与zk服务端的连接 46](#_Toc513384359)

[8.2、Zk会话重连机制 47](#_Toc513384360)

[8.3、创建Zk节点 48](#_Toc513384361)

[8.3.1 同步创建 48](#_Toc513384362)

[8.3.2 异步创建 48](#_Toc513384363)

[8.4、删除节点 49](#_Toc513384364)

[8.4.1 同步删除 49](#_Toc513384365)

[8.4.2 异步删除 49](#_Toc513384366)

[8.5、修改节点 50](#_Toc513384367)

[8.5.1 同步修改 50](#_Toc513384368)

[8.6、CountDownLatch代码示例 51](#_Toc513384369)

[8.7、获取节点数据 51](#_Toc513384370)

[8.8、获取子节点列表 51](#_Toc513384371)

[8.8.1 同步获取 51](#_Toc513384372)

[8.8.2 异步获取 52](#_Toc513384373)

[8.9、判断zk节点是否存在 54](#_Toc513384374)

[8.10、ACL权限列表 55](#_Toc513384375)

[8.10.1 默认匿名权限 55](#_Toc513384376)

[8.10.2 自定义用户权限 56](#_Toc513384377)

[8.10.3 ip权限 57](#_Toc513384378)

[8.11 总结 58](#_Toc513384379)

[九、Apache curator 59](#_Toc513384380)

[9.1、建立Curator与zkServer的连接 59](#_Toc513384381)

[9.2、zk命名空间以及创建节点 60](#_Toc513384382)

[9.3、修改节点数据及删除节点 60](#_Toc513384383)

[9.4、获取节点数据 62](#_Toc513384384)

[9.5、获取子节点数据 63](#_Toc513384385)

[9.6、判断节点是否存在 63](#_Toc513384386)

[9.7、UsingWatcher 64](#_Toc513384387)

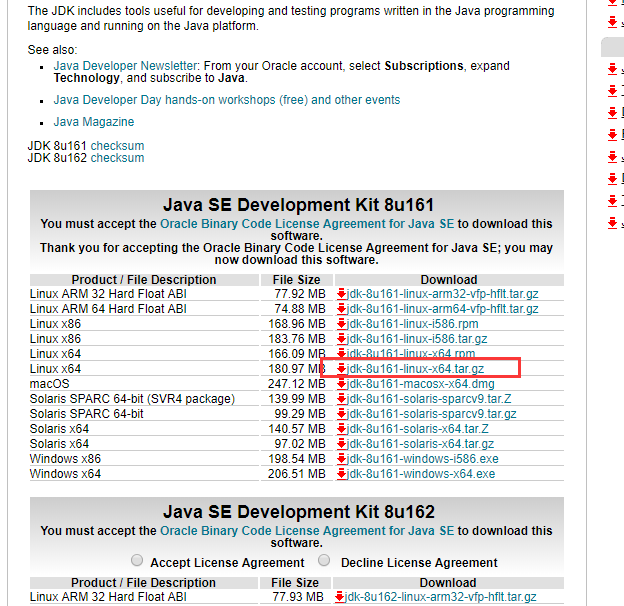
[9.8、nodeCache一次注册N次监听 64](#_Toc513384388)

[9.9、pathChildrenCache子节点监听 65](#_Toc513384389)

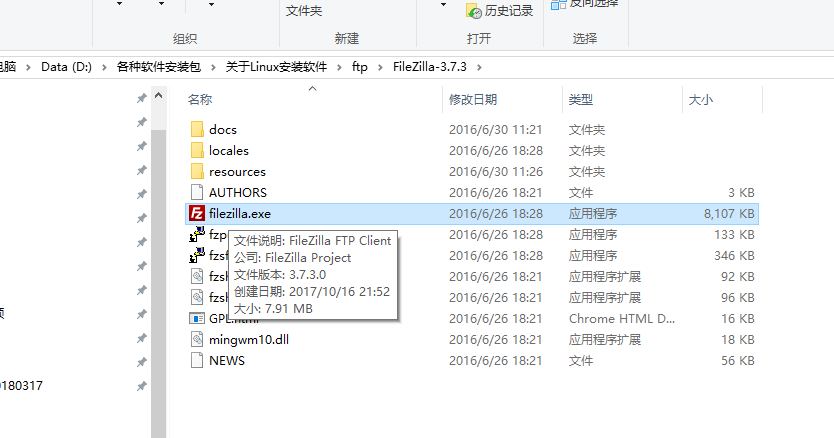
# 一、Jdk8安装

## 1.1、官网下载Linux安装包

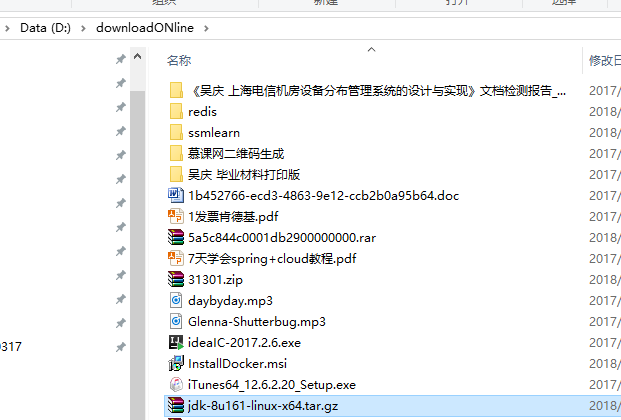
<http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

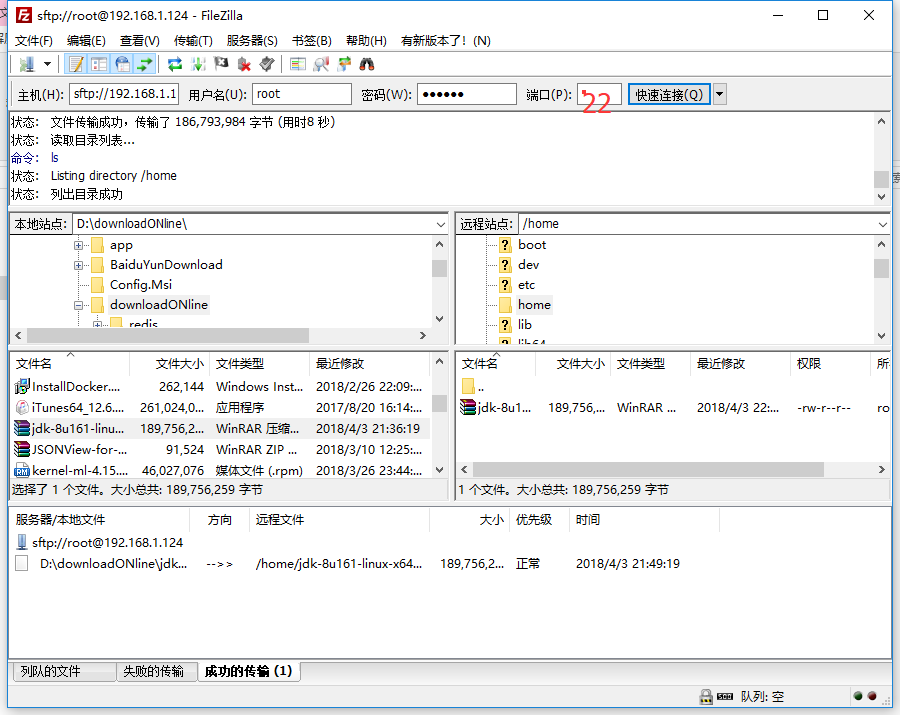


## 1.2、启动ftp传输工具



将下载好的文件通过Ftp工具放到/home目录下





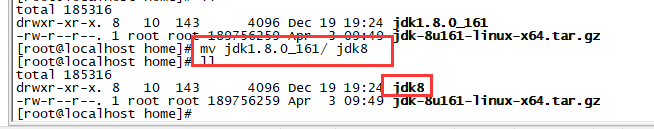
查看传完的列表：



## 1.3、解压Jdk8压缩包

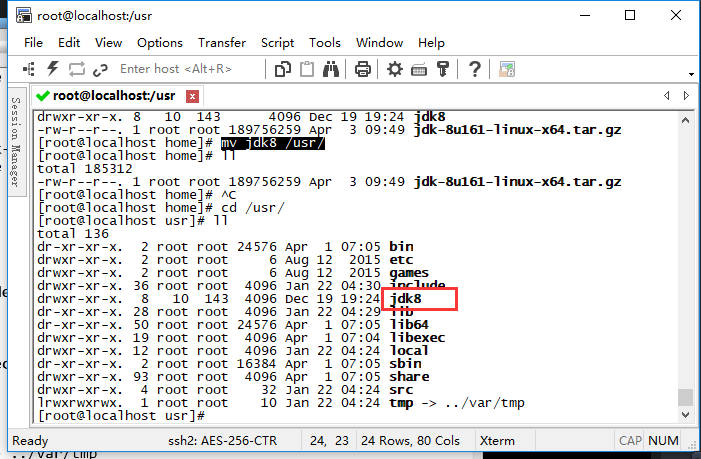


重命名将改为jdk8



把目录移动到usr目录下：

mv jdk8 /usr/



复制所在目录，可以通过pwd查看：



## 1.4、添加jdk配置信息

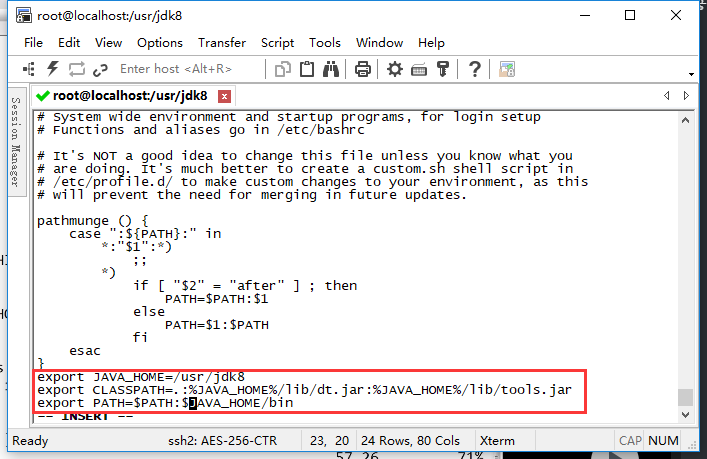
在/etc/profile中

添加：

export JAVA\_HOME=/usr/jdk8

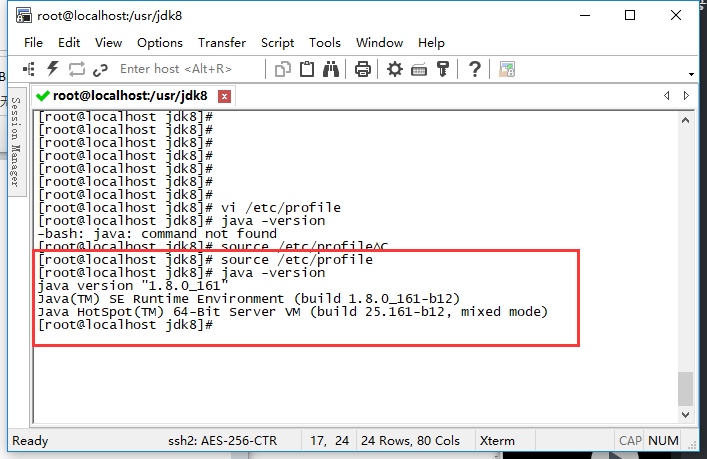
export CLASSPATH=.:%JAVA\_HOME%/lib/dt.jar:%JAVA\_HOME%/lib/tools.jar

export PATH=$PATH:$JAVA\_HOME/bin



修改完记得刷新 我们的配置文件：

source /etc/profile，输入java –version,可以看到:



# 二、单机安装Zookeeper:

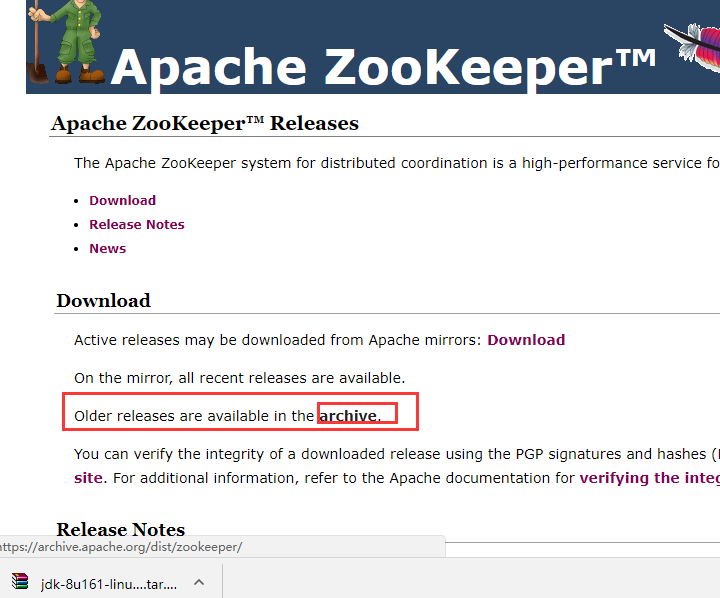
## 2.1、去官网下载安装包



点击Download：

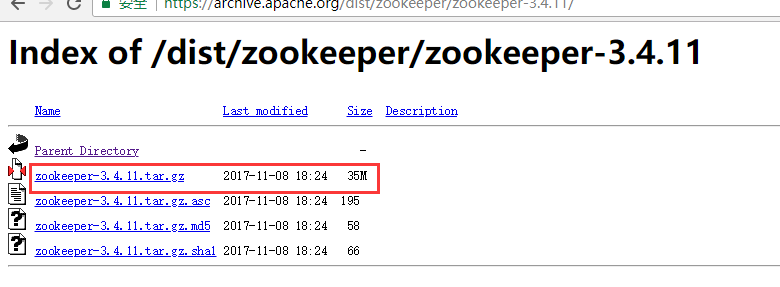


点击archive:



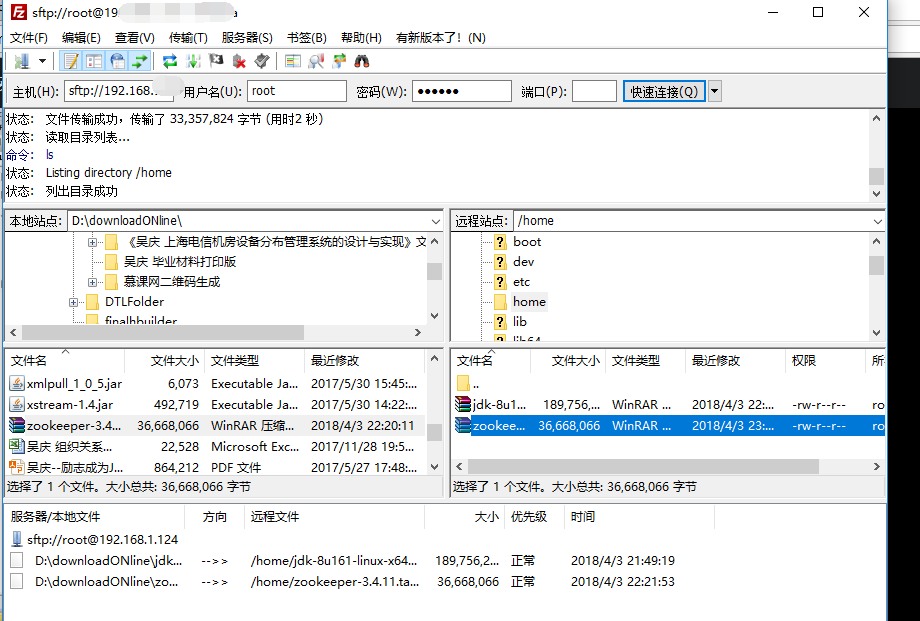
可以看到所有目前稳定的版本都在列表中：

点击3.4.11：



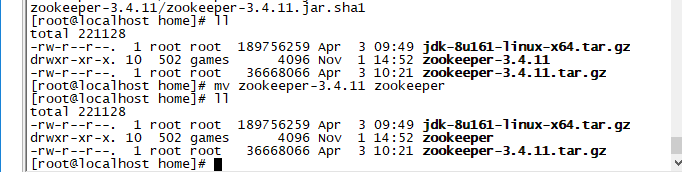
点击下载即可。

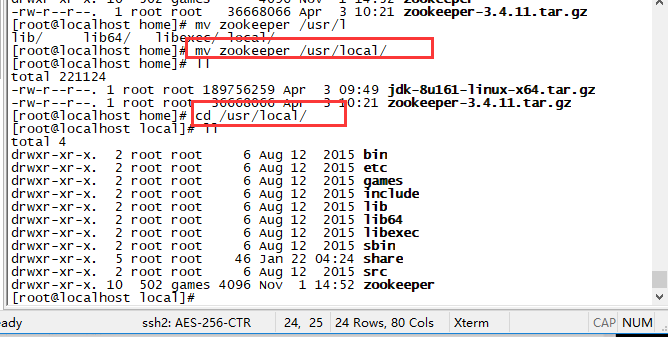
## 2.2、启动ftp传输工具



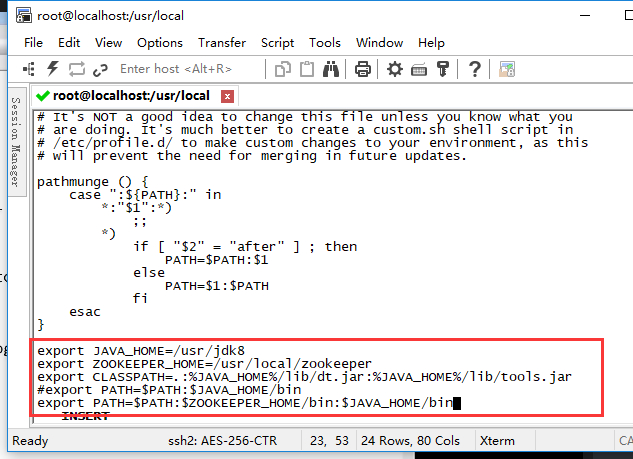
## 2.3、解压安装zookeeper

然后就是一系列的解压，重命名操作，同上面：

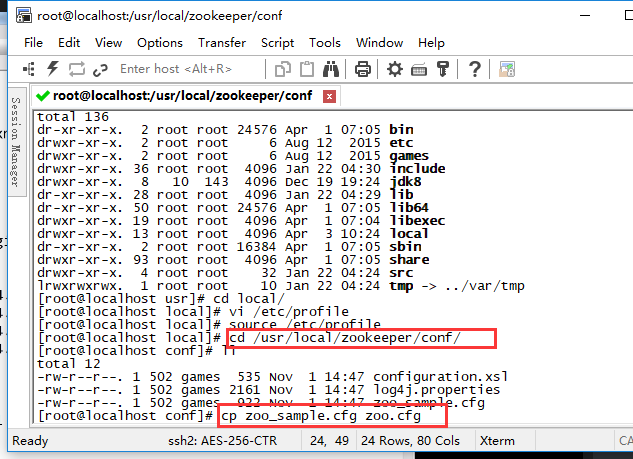




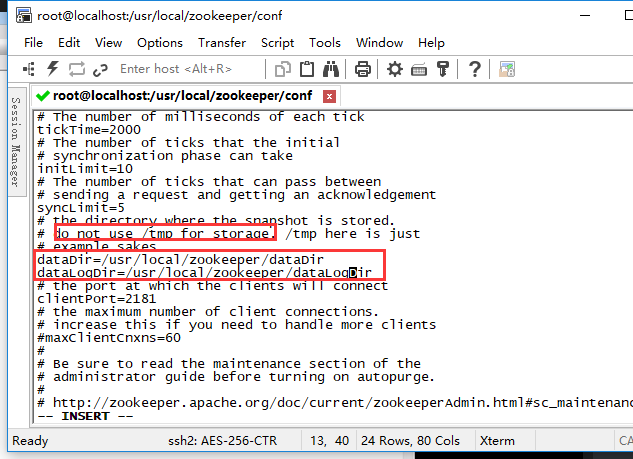
## 2.4、添加Zookeeper配置信息



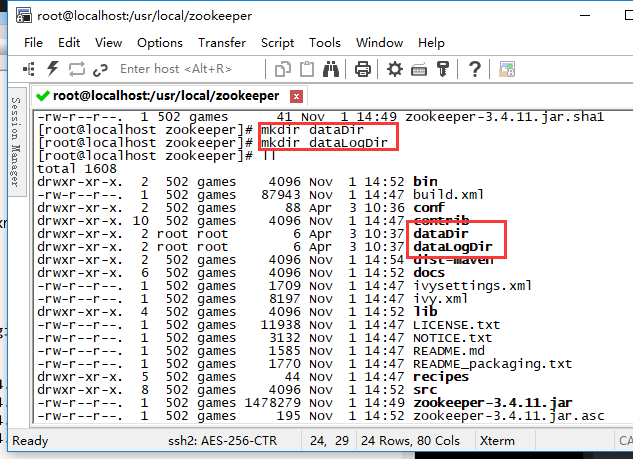
拷贝一份

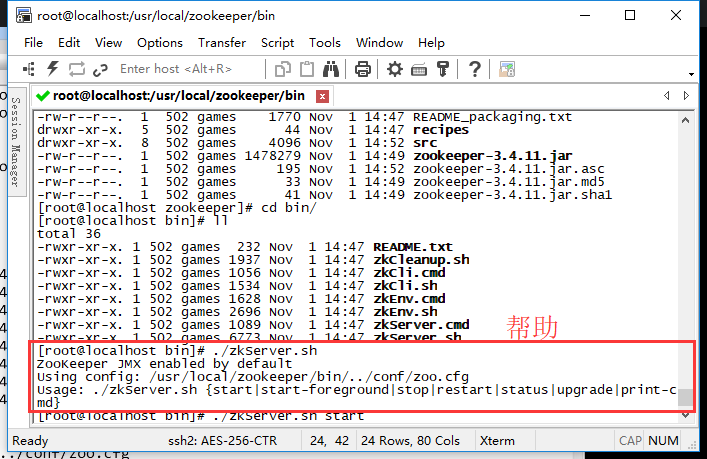


默认是tmp目录，需要修改成如下



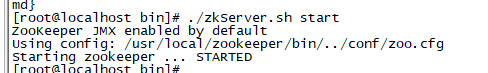
新建我们配置的文件夹名称：





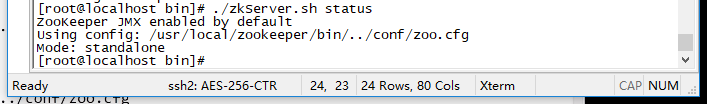
# 三、操作Zookeeper

## 3.1、启动zkServer



## 3.2、查看状态

（我们这 单机状态standalone）

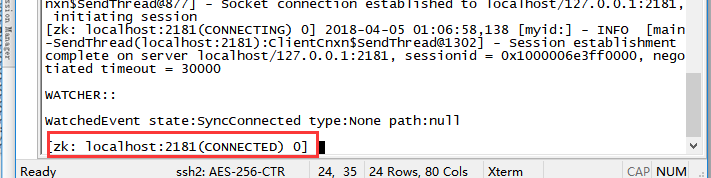


## 3.3、启动zkCli

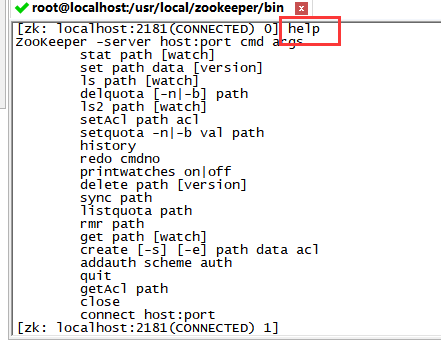
./zkCli.sh



这里再按回车，可以看到：



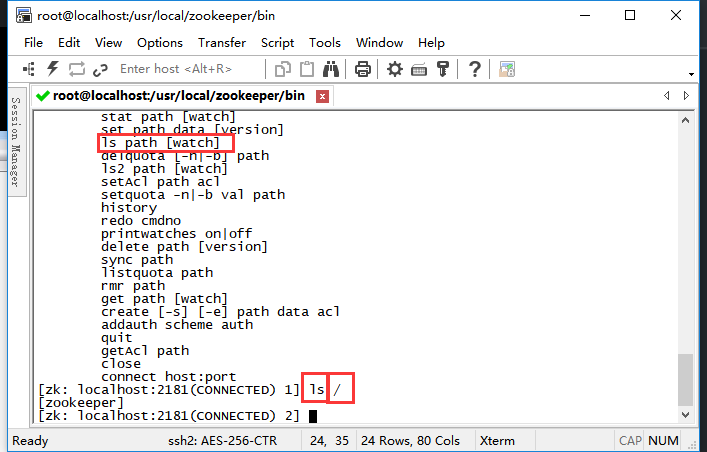
## 3.4、查看帮助



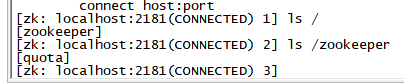
## 3.5、 ls&ls2命令

（1）ls命令：

可以看到我们目前根目录下有个zookeeper



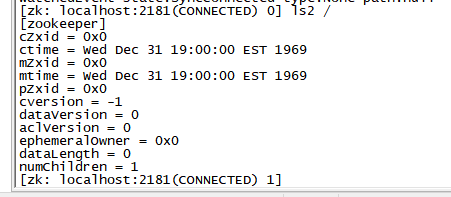
再看zookeeper下有什么：



由上可看出zookeeper下有子节点quota.

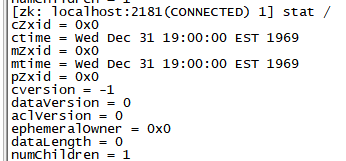
（2）Ls2命令

可以看出当前目录的状态，如创建时间等信息。

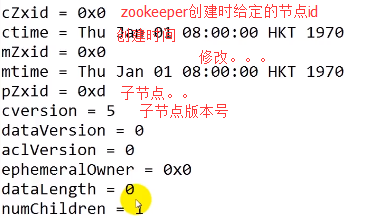


## 3.6、get&stat命令

（1）stat



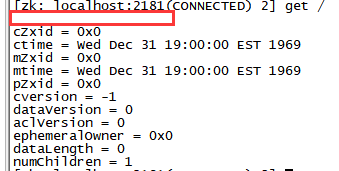
简单说明：



所以Ls2=ls+stat

（2）get

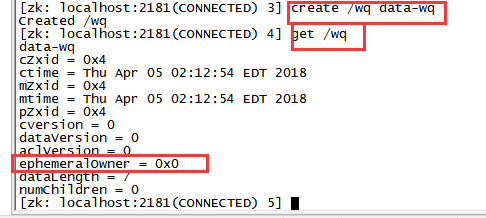
获取当前指定节点中数据：（因为我们当前节点下没东西所以这里是空）



## 3.7、create创建持久/临时/有序节点

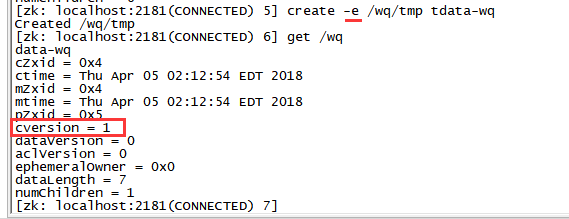
### 3.7.1 create持久

持久化体现在:ephemeralOwner=0x0

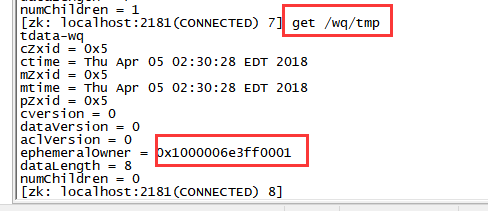


### 3.7.2 Create –e 临时

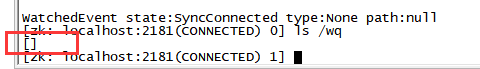
接着我们再创建一个临时节点，如下：



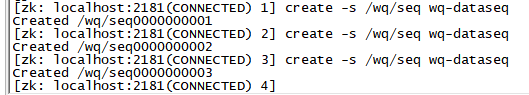
而查看后会发现：



临时节点怎样让它删除呢？这就利用心跳机制，就是退出我们客户端与服务端的连接，即会话断了。按住ctrl+c退出我们的客户端，再重新启动客户端（./zkCli.sh），过了心跳机制的时间段（10秒或20秒后），再输入ls /wq 可以看到我们之前创建的临时节点没了。如下所示：



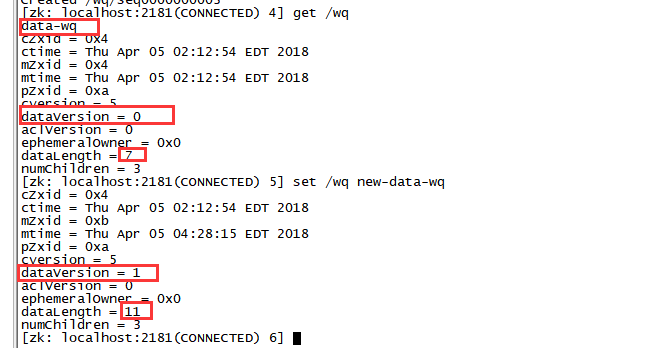
### 3.7.3 create –s 有序



## 3.8、set&delete命令

### 3.8.1 set操作

set更新节点中的数据，如将我们的/wq结点数据从data-wq更改为new-data-wq,改变后对比如下：



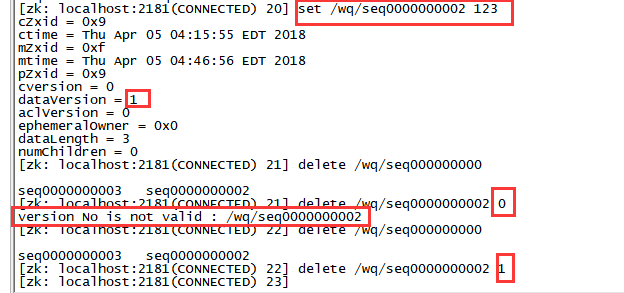
### 3.8.2 set乐观锁操作



### 3.8.3 delete操作



再对第二个节点进行修改操作使之数据版本号由0变为1，再使用老的版本号0对其进行删除，会发现，报错。如果使用新的版本号即可删除成功如下所示：

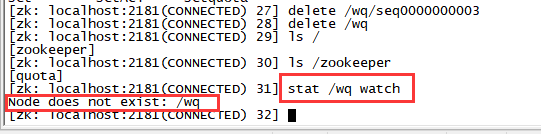


## 3.9、watcher命令行操作

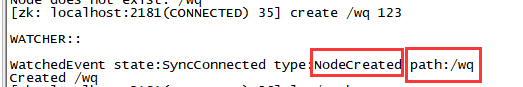
### 3.9.1 父节点操作

#### 3.9.1.1创建触发NodeCreated

首先我们可以先把先前创建过的节点都删除干净（注意先删除子节点），然后输入stat /wq watch用于对我们/wq节点添加watcher事件。



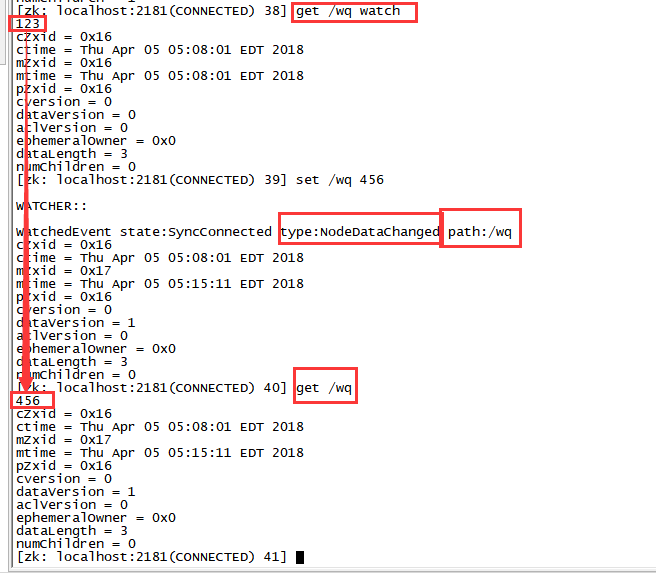
接下来再创建我们的/wq节点，则会发现已经被监听到了，它告诉我们创建了一个节点，其目录为/wq，如下所示：



#### 3.9.1.2修改触发NodeDataChanged

注：watcher是一次性的，触发后立即销毁。

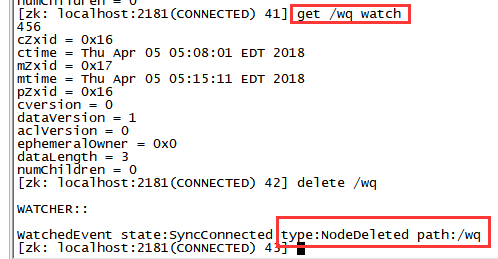
所以我们需要在这里输入get /wq watch再次对/wq节点添加watcher，再修改/wq节点的值，会发现NodeDataChanged事件发生了。如下所示：



#### 3.9.1.3删除触发NodeDeleted

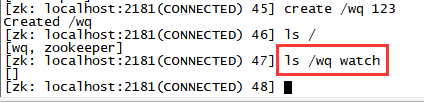
注：watcher是一次性的，触发后立即销毁。

因此需要重新给我们的节点/wq加上watcher事件。再删除会发现监听到我们删除节点的信息，如下所示：

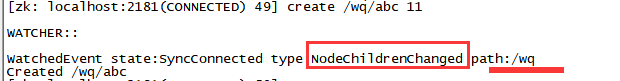


### 3.9.2 子节点操作

#### 3.9.2.1创建触发NodeChildrenChanged

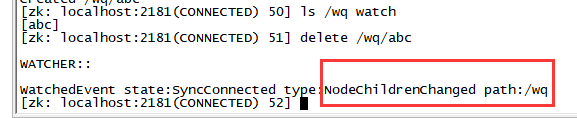


接下来我们可以创建一个子节点，可见watcher成功起作用了，如下所示：

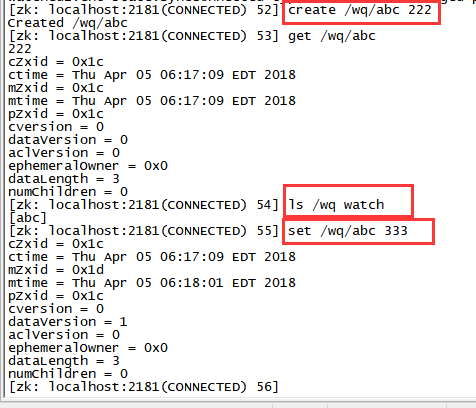


#### 3.9.2.2删除触发NodeChildrenChanged

由于子节点删除对父节点来讲只是发生了变化，具体的操作父节点是不关心的。

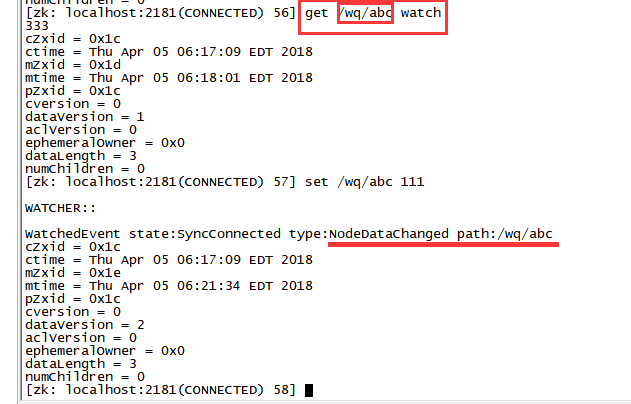


#### 3.9.2.3更新不触发



由上图可以发现并没触发事件。因此说子节点更新不会触发事件。

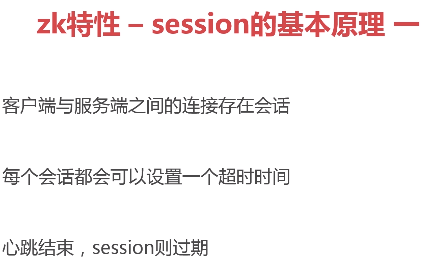
针对子节点修改，则需要将子节点当成父节点来操作，即如下：

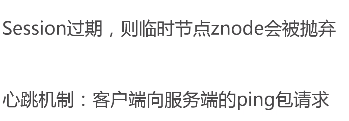


# 四、zk特性

## 4.1、session的基本原理

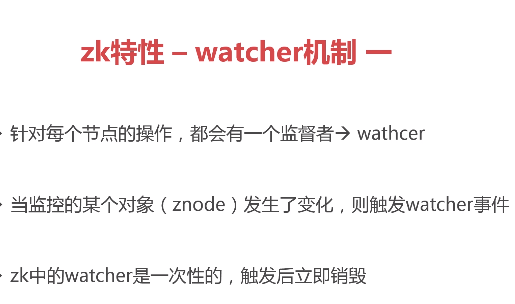
### 4.1.1 原理

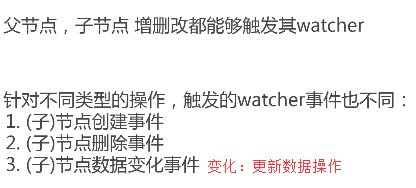




## 4.2、理解watcher机制

### 4.2.1 机制



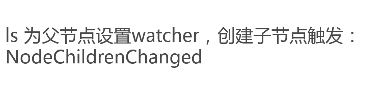


### 4.2.2 事件类型

#### 4.2.2.1父节点事件



#### 4.2.2.2子节点事件



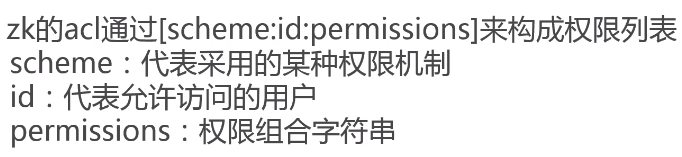


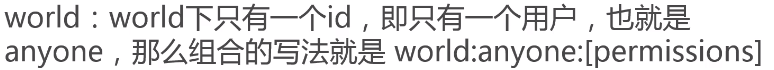


# 五、ACL(access control lists)权限控制

针对节点可以设置相关读写等权限，目的为了保障数据安全性，权限permissions可以指定不同的权限范围以及角色。

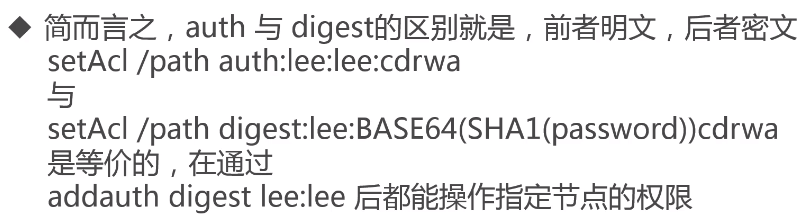
## 5.1、ACL构成











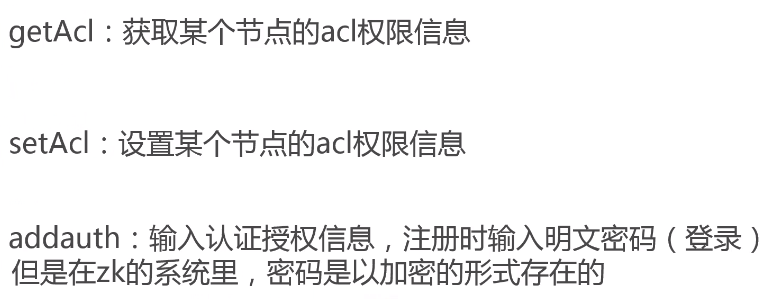


super一般是给运维、研发经理、项目总监用的，通过修改相应文档进行配置后才可以使用。





## 5.2、Acl命令行



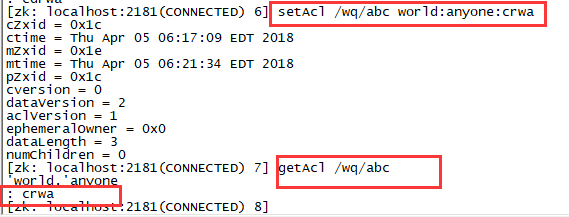
### 5.2.1 getAcl



### 5.2.2 world



默认在节点创建完是允许所有匿名用户对该节点进行操作的。



如上图所示，我们给/wq/abc 设置了crwa(即创建、读、写、管理操作)。

**注：由于我们的/wq默认在创建的时候会拥有cdrwa权限**，此时我们删除/wq/abc肯定是可以的。

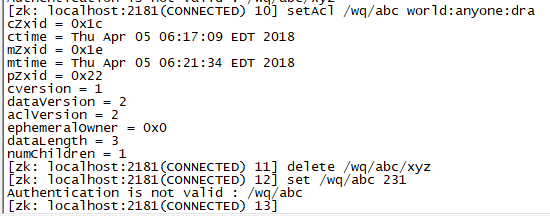
那么怎样来验证我们刚设的crwa权限呢？这里我们需要在/wq/abc下再创建一个子节点即/wq/abc/xyz从而来验证上一步我们给/wq/abc设置不能删除节点的权限。如下所示：



接下来我们测试一些组合：

（1）dra

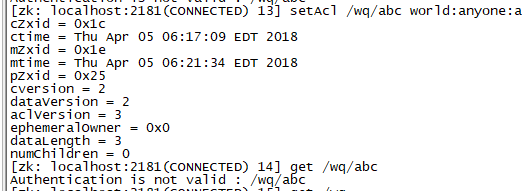
设置dra后，发现可以删除成功。但由于没设修改权限”w”所以我们不能进行修改操作。



（2）a

可以发现不能进行读写、删除、创建操作了。

不能读操作，如下所示：



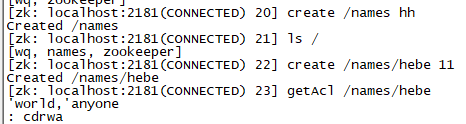
不能创建节点，如下图所示：



其余演示，自行练习，这里就不一一贴图了。

### 5.2.3 Auth

第一步，先初始化一些节点，在根目录下新建节点/names,再在/names下新建节点hebe并查看其权限，如下所示：

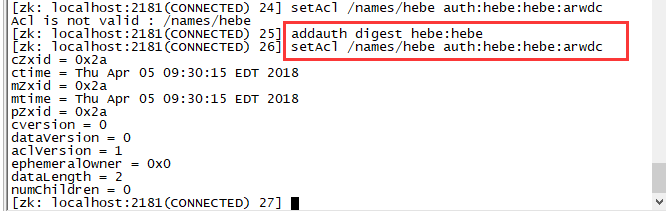


注:下面演示一下错误操作，直接给当前节点验证用户名hebe密码hebe，会出现如下错误信息：



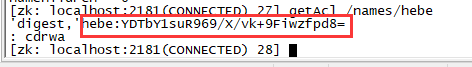
这是由于我们的用户名密码并没有进行注册（即登录信息并未注册）。

那么就需要我们将该用户名/密码添加到库里，再登录并设置权限就行了，如下所示：



获取该节点Acl,并将我圈出部分即账户信息hebe:YDTbY1suR969/X/vk+9Fiwzfpd8=

拷贝（选中右键粘贴即可，不要ctrl+c，因为会退出）到本地记事本中，待会有用到。如下所示：

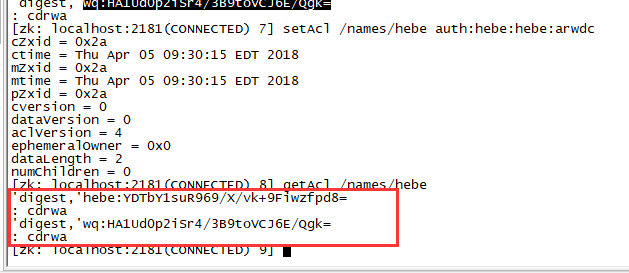


这里提一下，（此案例一直没演示成功那个setAcl …auth:hebe:hebe:awd的操作，真的炸了）如下所示：  
//img.mukewang.com/szimg/5ac62e0a0001b4ff08540100.jpg

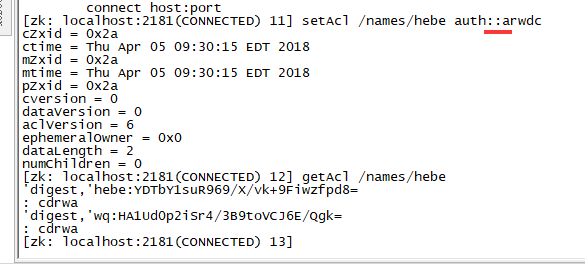
后来终于知道啥原因了在上一步拷贝密码的时候我用了ctrl+c退出了客户端，没重启。重启 ./zkServer restart这个时候我再./zkCli.sh，进入客户端，这个时候之前的用户相当于退出了，然后你get这个会报错。

上一步的坑终于跳过了，接下来说下后面添加的用户都会使用第一个用户的账号信息。

首先，为了演示，新加一个用户wq:wq,（ wq:HA1Ud0p2iSr4/3B9toVCJ6E/Qgk=）它的Acl信息如下：

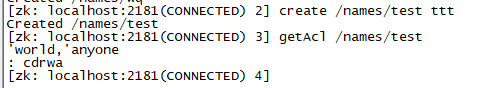


即这里**省略用户密码信息**也是可以进入的。默认使用的是当前第一个的用户信息。

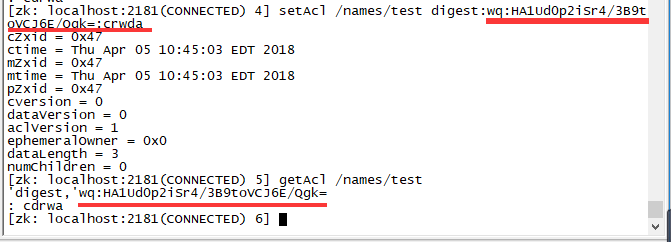


### 5.2.4 digest

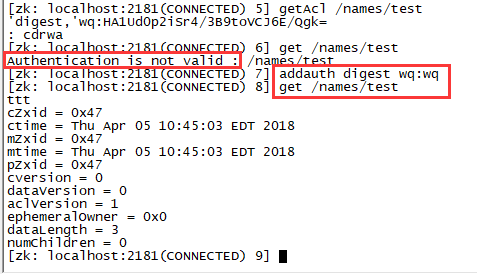
使用digest首先我们需要退出当前用户，即ctrl+c退出客户端。再./zkCli.sh运行客户端。接下来在/names节点下新建test节点，如下所示：



设置Acl，用户信息就是上面我们拷贝的wq的用户信息，如下所示：

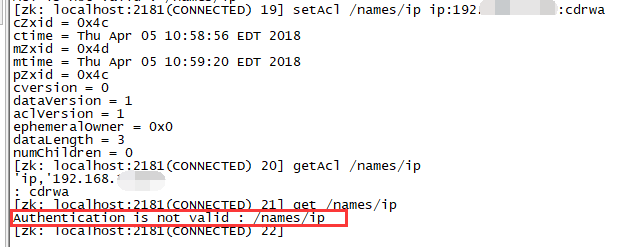


登录addauth digest wq:wq,注意登录我们也不可能让用户输入暗文登录，所以还是使用明文密码登录。如下所示：



### 5.2.5 使用Ip

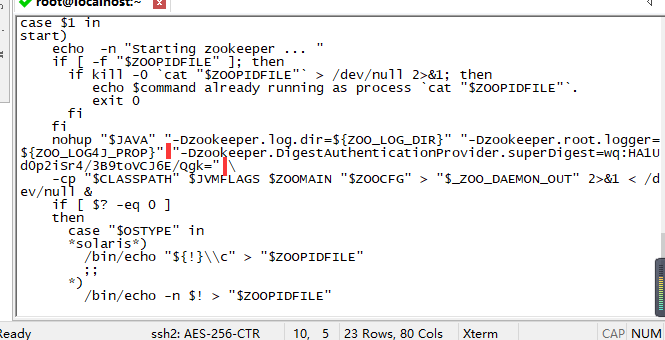
这里由于我们是Linux环境目前只演示怎么配置，等后面再详细说明怎么使用Java客户端来成功访问到我们的节点数据，配置Acl如下所示：



### 5.2.6 使用Super

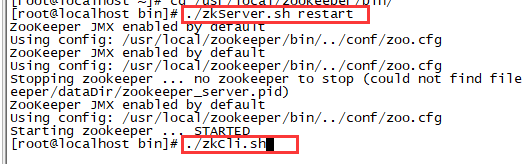
（1）首先需要vi zkServer.sh 利用/查询nohup，紧接着添加超级管理员配置信息，如下所示：



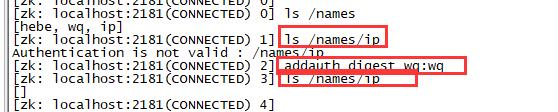


-Dzookeeper.DigestAuthenticationProvider.superDigest=super:/o7XEyClGiOcKnJmPVFVYU8AQdw=我们之前设过的wq的用户密码。

（2）重启zkServer.sh,再重启客户端zkCli.sh，如下所示：



（3）接下来验证超级管理员权限



通过上图，我们可以发现，一开始我们是没登录超级管理员，所以查看ip节点是没有权限的（因为上一步我们给ip节点加了固定ip权限才能操作），所以在使用addauth digest wq:wq登录我们的超级管理员后，就可以任意妄为地操作我们的节点了。

## 5.3Acl使用场景

（1）开发、测试环境分离，开发者无权操作测试库的节点，只能看；

（2）生产环境上控制指定ip的服务可以访问相关节点，防止混乱。

# 六、ZK四字命令

## 6.1 Four Letter words

（1）zk可以通过它自身提供的简写命令来和服务器进行交互；

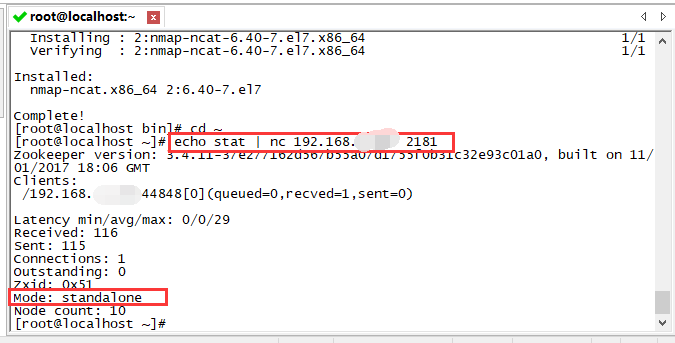
（2）需要使用到nc命令，安装:yum install nc

（3）echo [command] | nc [ip] [port]

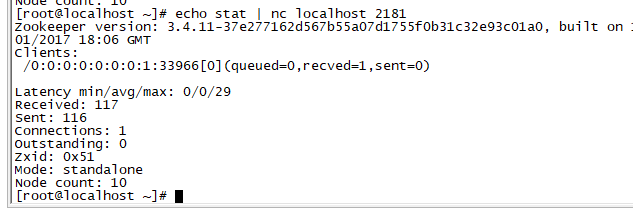
## 6.2常用四字命令

### 6.2.1 stat

查看zk的状态信息，以及是否mode。



使用localhost是一样的效果，如下所示：



### 6.2.2 ruok

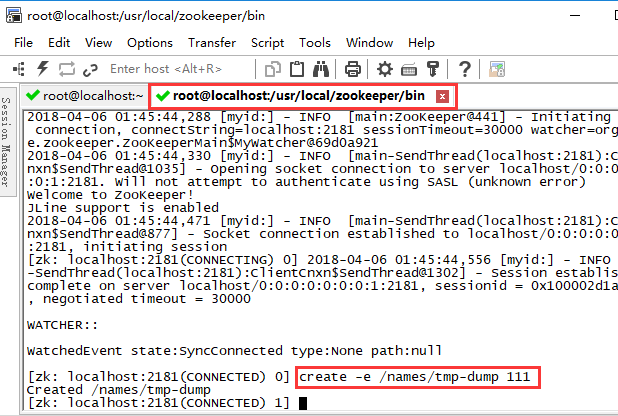
查看当前zkserver是否启动，返回imok 🡪 Are you ok? I ‘m ok.，如下所示：



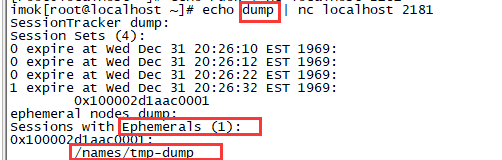
### 6.2.3 dump

dump用于列出未经处理的会话和临时节点。

（1）首先我们新建一个窗口启动客户端再创建一个临时节点，如下所示：

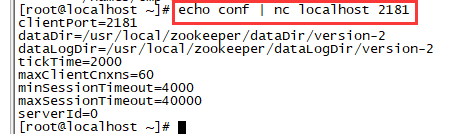


（2）接下来使用dump命令，查看临时会话以及临时节点信息，如下所示：



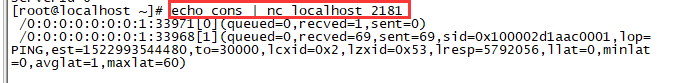
### 6.2.4 conf

conf用于查看服务器配置。



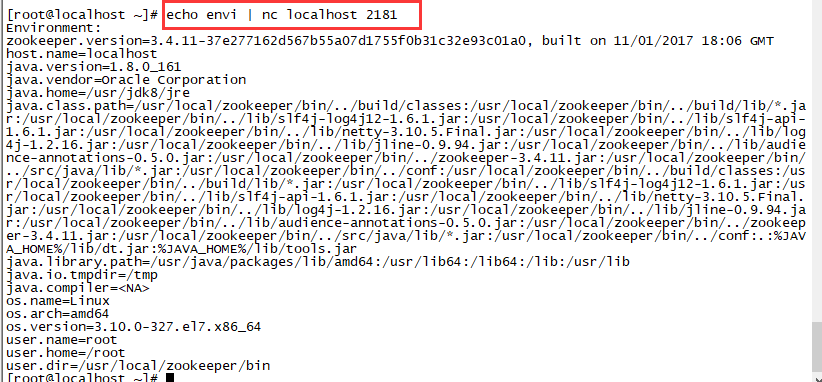
### 6.2.5 cons

cons用于展示连接到服务器的客户端信息。



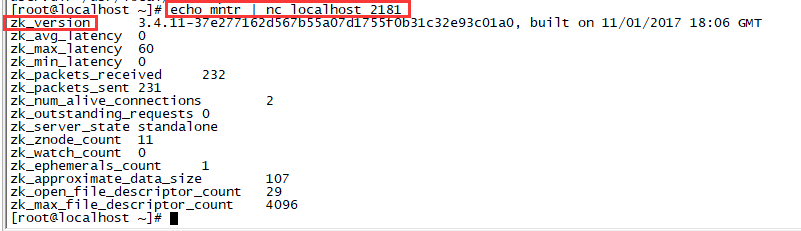
### 6.2.6 envi

用于查看相关的环境变量信息，如下所示。

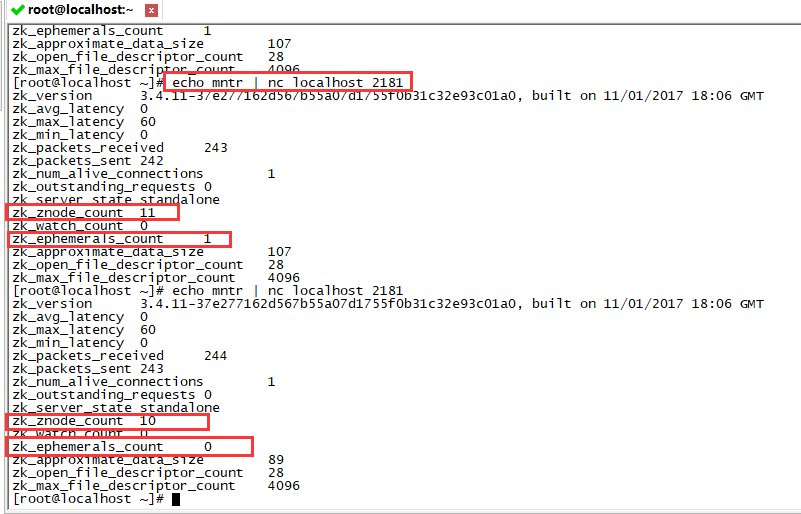


### 6.2.7 mntr

监控zk健康信息。



然后我把刚才的会话窗口关了，会发现：

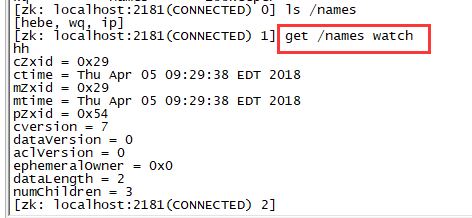


注：关闭以后查看可以发现临时节点个数还是1，这是因为zk的session心跳机制（前面章节有提及），需要等待一段时间再监控发现临时节点没了。

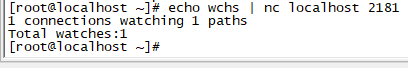
### 6.2.8 wchs

展示watch的信息。

（1）首先为了验证，我们需要重新登录客户端，并设置一个监听如下：



（2）然后使用命令发现监听个数信息，如下所示：



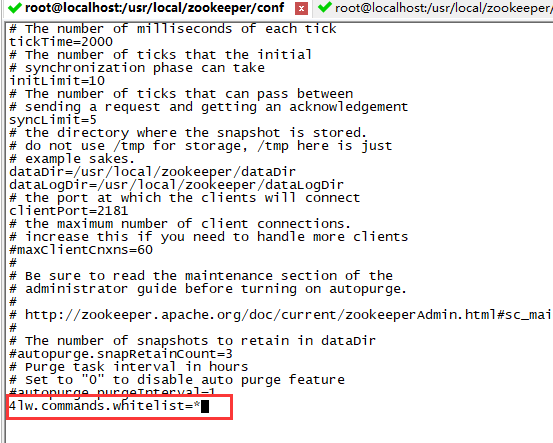
### 6.2.9 wchc&wchp

查看session与watch 及path与watch信息。

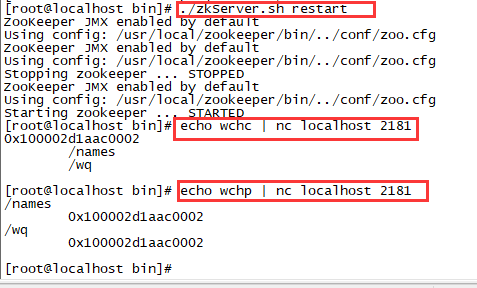
（1）首先，为了演示我又给wq节点加了个监听，然后我分别使用wchc、wchp命令发现都提示如下错误信息：



这时，我们需要注意，由于我们的zk版本3.4.11（官网文档中有说3.4.10以上直接使用就会报上述错误信息）。因此，我们需要配置，需要在zoo.conf（cd /usr/local/zookeeper/conf/下）中添加如下配置信息（4lw.commands.whitelist=\*）：



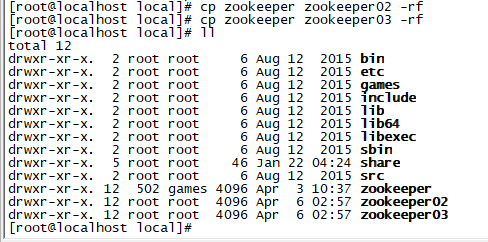
接着:wq保存下，再重启zkServer.sh即可生效，如下所示：



# 七、搭建集群

## 7.1、单机伪分布式安装zookeeper集群

（1）拷贝两份zookeeper

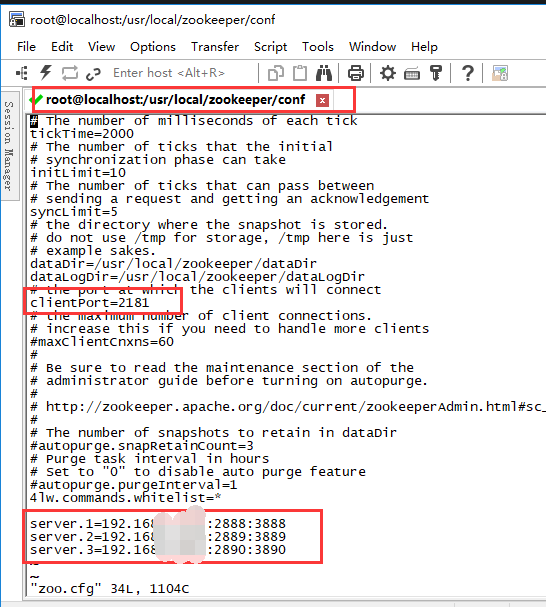


修改添加配置文件如下：

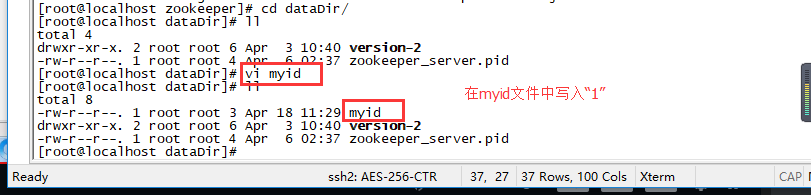
server.1=192.168.1.124:2888:3888

server.2=192.168.1.124:2889:3889

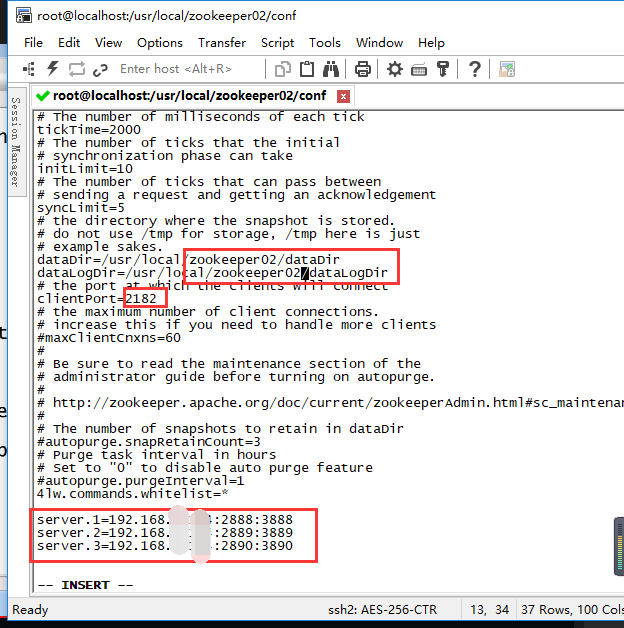
server.3=192.168.1.124:2890:3890



在dataDir下新建一个Myid文件，里面输入”1”,如下所示：

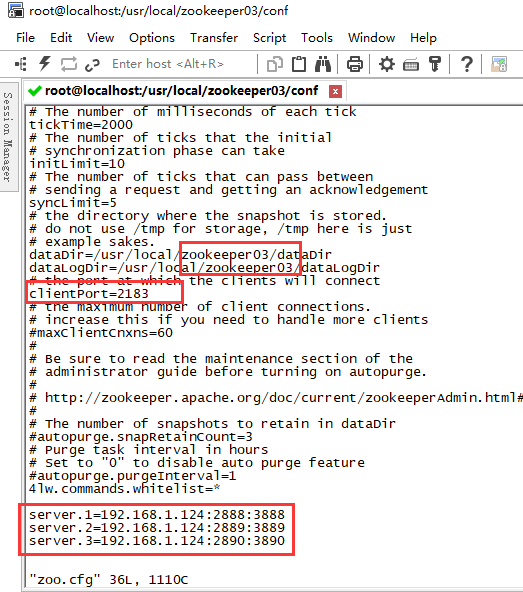


（2）第二台节点配置需要修改端口信息、数据缓存目录，添加相关配置信息，如下所示：



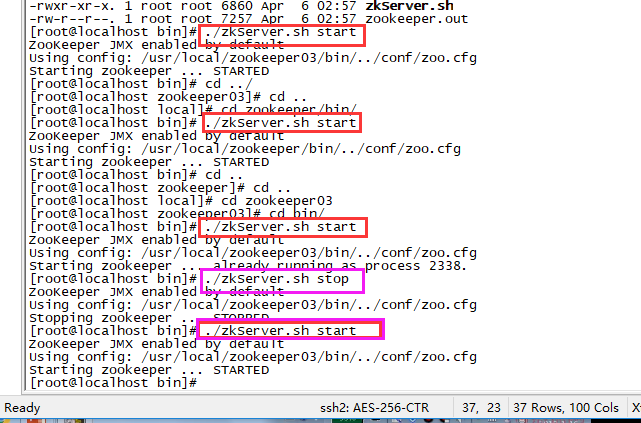
接着在第二台节点中的dataDir目录下创建myid文件，内容设置为”2”即可。

（3）设置第三台节点



并在dataDir下添加myid文件，内容设置为“3”即可。

（4）启动三个节点服务器

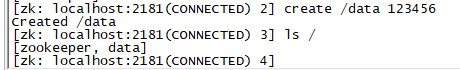


（5）启动客户端

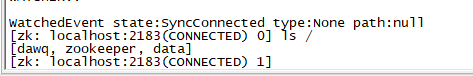
随便连一个，如下所示：



随便找一个节点服务器，创建一个节点，如下所示：

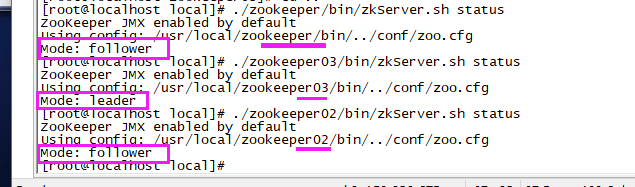


然后创建完毕后，再连接其他的节点（比如2183）查看会发现已经同步了我们新添加的节点data!

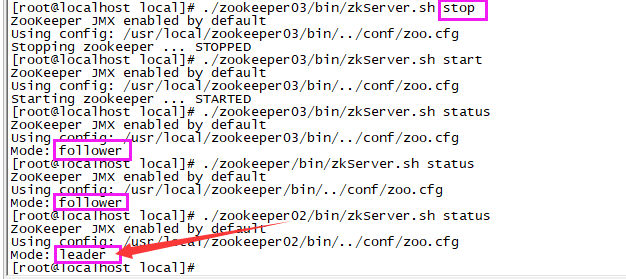


## 7.2、测试集群角色以及选举

直接利用./zkServer.sh status获取当前节点的集群角色，可以看出我们的zookeeper03节点是leader主节点，其他两个为follower从节点，如下所示：



接下来，我们可以模拟宕机操作，将03节点停止下再启动会发现它从主节点变成了从节点，而新的主节点又变成了02，如下所示：



# 八、ZooKeeper原生Java API使用

## 8.1、建立客户端与zk服务端的连接

需要把上一章节的三台节点服务器都启动哦。不然连不上的哟。

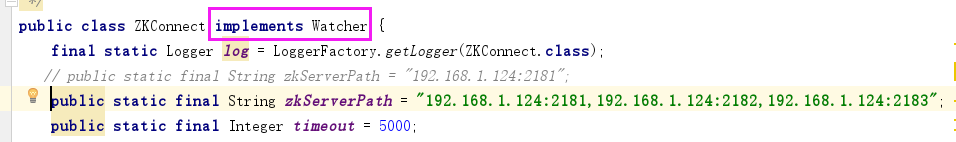


然后通过命令连接即可。

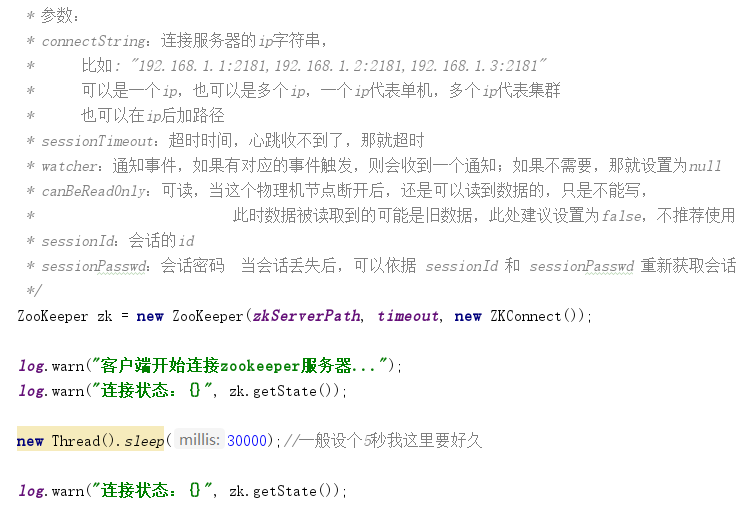
接下来，利用java连接zk服务端。

对应代码：项目zookeeper-starter/ZKConnect.java：

（1）定义超时常量、服务端地址，这里我启用的是伪分布式集群，然后实现Watcher接口重写processOn方法，如下所示：



@Override  
**public void** process(WatchedEvent watchedEvent) {  
 ***log***.warn(**"接受到watch通知：{}"**, watchedEvent);  
}



由于我这里连接比较慢，我将线程睡眠时间设为了30秒，这才成功接收到了通知，如下所示：



## 8.2、Zk会话重连机制

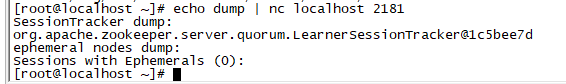
（1）大致思路：

通过第一次连获取返回的sessionId和sessionPassword再作为参数进行第二次连接。对应代码：项目zookeeper-starter/ZKConnectSessionWatcher.java

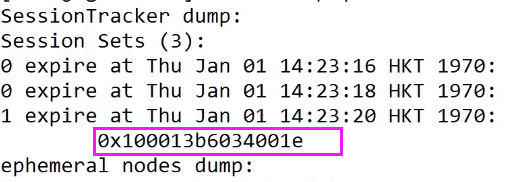


（2）怎么查看有无会话

第一步可以通过echo dump | nc localhost 2181 查看是否存在会话，可以看出刚开始并没有，如下所示：



接下来可以debug，再输入上一步命令，即可验证，如下所示：



## 8.3、创建Zk节点

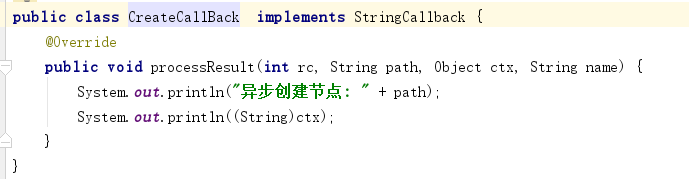
### 8.3.1 同步创建

result = zookeeper.create(path, data, acls, CreateMode.PERSISTENT);//同步创建

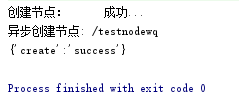
### 8.3.2 异步创建

String ctx = **"{'create':'success'}"**;  
**zookeeper**.create(path, data, acls, CreateMode.***PERSISTENT***, **new** CreateCallBack(), ctx);

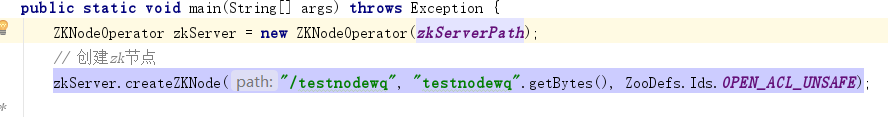
异步回调方法如下：



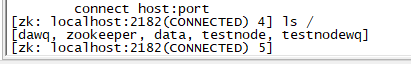
控制台输出结果如下：



两种对应的main方法如下：



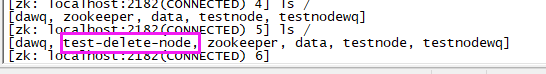
创建完可以查看节点已经成功创建，如下所示：



## 8.4、删除节点

### 8.4.1 同步删除

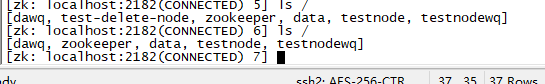
（1）先利用上一节创建节点方法来创建一个节点供删除测试，如下所示：



（2）调用删除方法，删除后查看节点是否存在，如下所示：

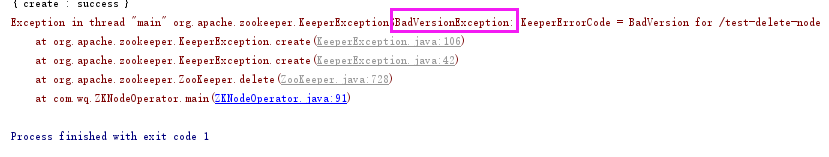
zkServer.getZookeeper().delete(**"/test-delete-node"**, 0);*//0代表版本号*

可以看出，我们的节点test-delete-node已经被删除了，如下所示：

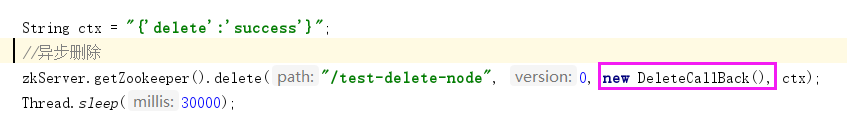


这里，如果版本号改成不是当前节点数据的版本号（如）2，会报BadVersion相关错误，如下所示：

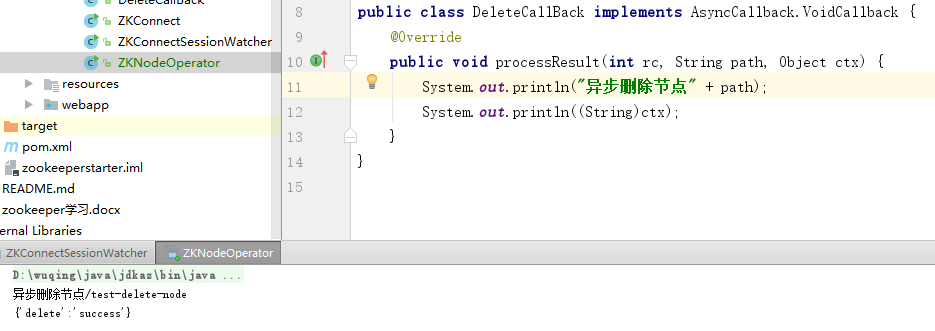




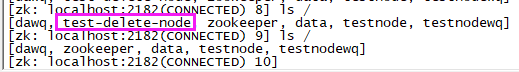
### 8.4.2 异步删除



新建DeleteCallBack 实现VoidCallBack接口，并重写回调方法，如下所示：



通过以上异步删除节点操作，可以看出，我们的测试删除节点已经成功删除，如下所示：



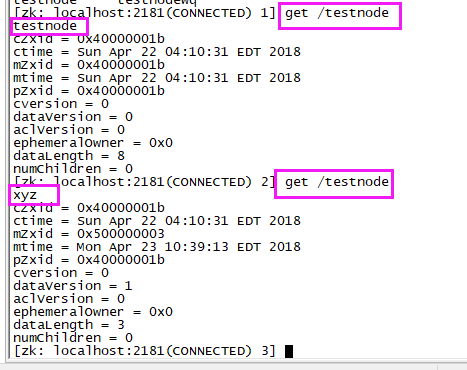
## 8.5、修改节点

### 8.5.1 同步修改

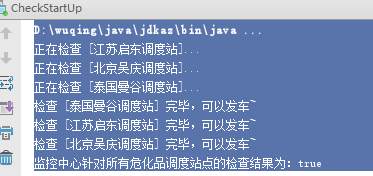
Stat status = zkServer.getZookeeper().setData(**"/testnode"**, **"xyz"**.getBytes(), 0);//要修改的节点数据版本号为0

System.***out***.println(status.getVersion());//输出1

通过以上代码，已经成功同步修改节点，如下所示：

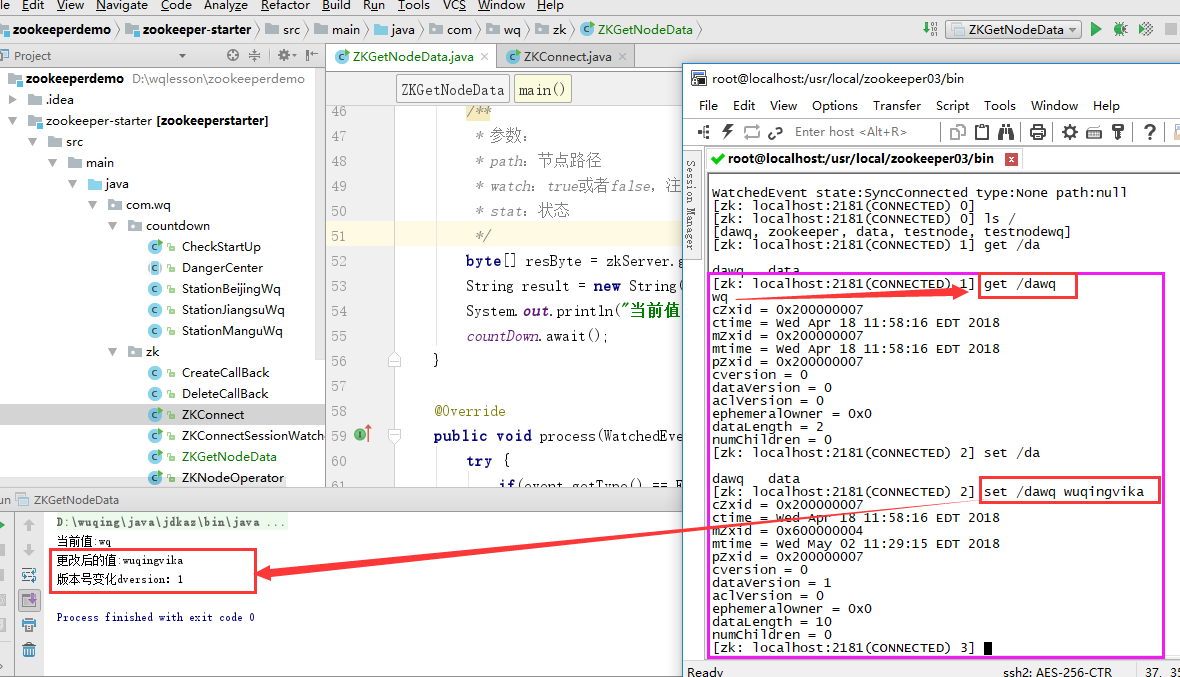


## 8.6、CountDownLatch代码示例



## 8.7、获取节点数据

代码ZKGetNodeData.java

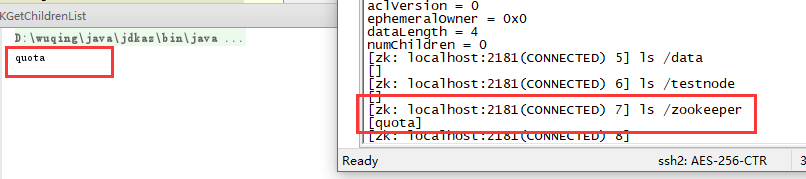


## 8.8、获取子节点列表

### 8.8.1 同步获取

|  |
| --- |
| List<String> strChildList = zkServer.getZookeeper().getChildren(**"/zookeeper"**, **true**); **for** (String s : strChildList) {  System.***out***.println(s); } |

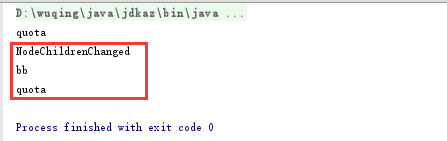
可以看出我们/zookeeper节点下面子节点有quota，如下所示:



接下来，再在/zookeeper节点下创建/bb节点，如下所示：



接下来我们的程序监听后输出如下信息：



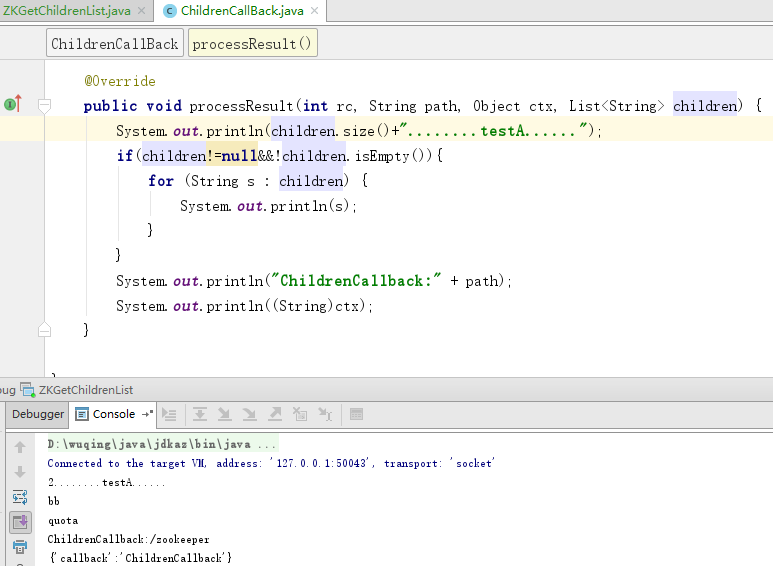
### 8.8.2 异步获取

（1）第一种异步获取

*// 异步调用*

|  |
| --- |
| String ctx = **"{'callback':'ChildrenCallback'}"**; zkServer.getZookeeper().getChildren(**"/zookeeper"**, **true**, **new** ChildrenCallBack(), ctx); |

异步回调方法如下：

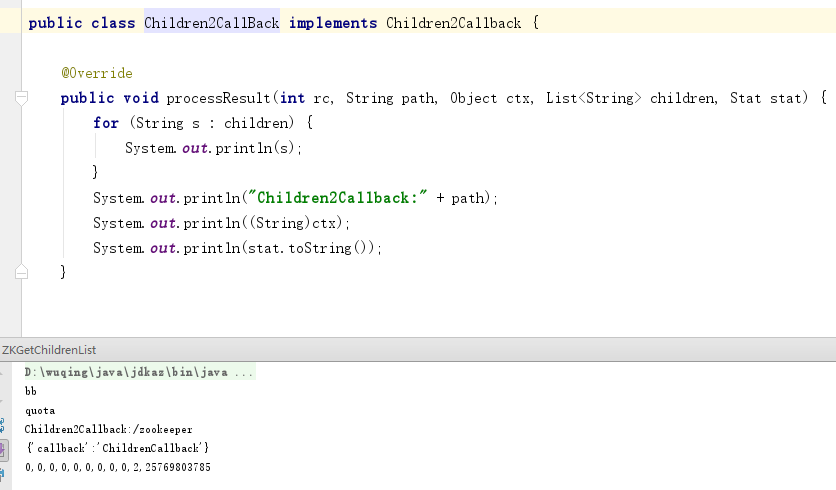


（2）第二种异步获取

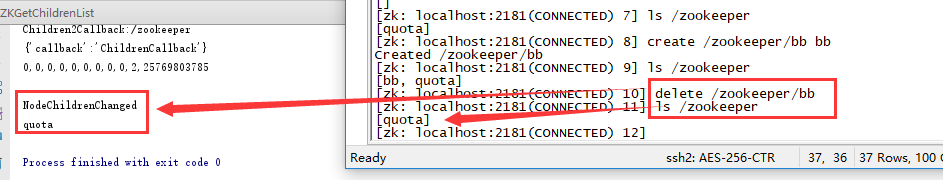
*// 异步调用*

|  |
| --- |
| String ctx = **"{'callback':'ChildrenCallback'}"**; zkServer.getZookeeper().getChildren(**"/zookeeper"**, **true**, **new** Children2CallBack(), ctx); |

异步回调方法如下：



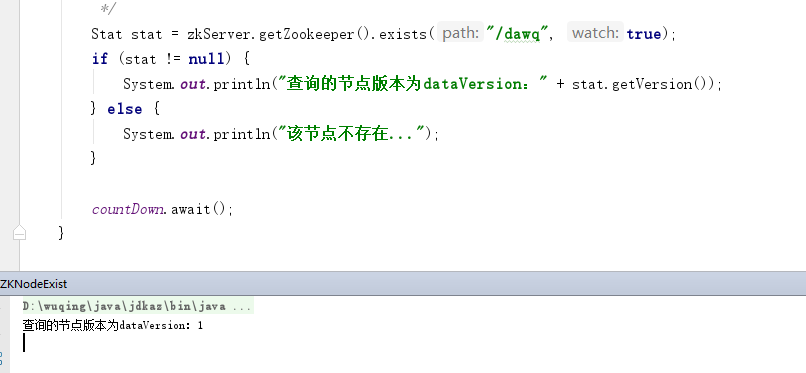
删除一个节点并测试监听，可以看到能成功监听到节点变化，如下所示：



注：如果修改子节点数据，是不会被监听到，原理见前期章节。

## 8.9、判断zk节点是否存在

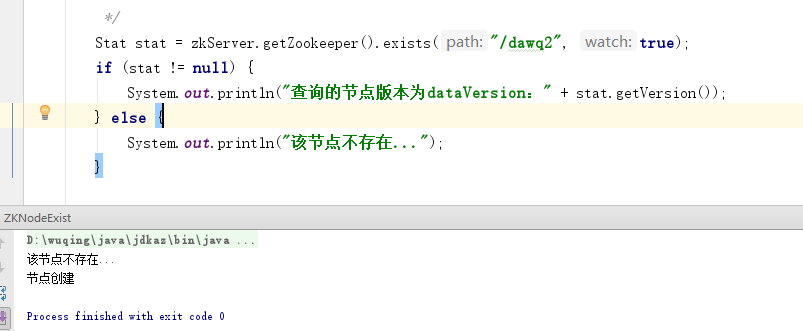
代码：ZKNodeExist



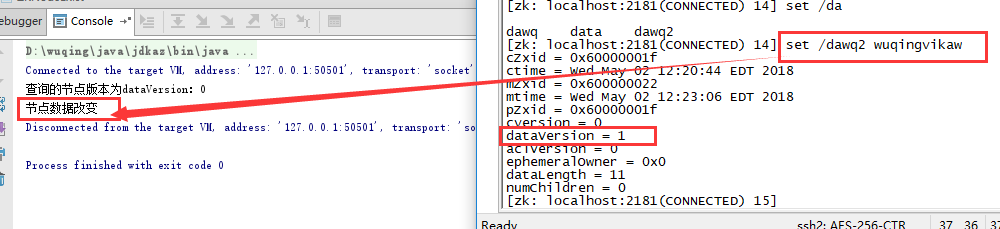
如果给个不存在的节点，会输出不存在信息，如下所示:



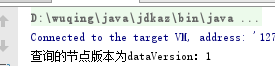
但此时watcher监听还是存在的，下面我们创建那个不存在的节点/tawq2,发现监听信息如下所示:



那么我们再重新启动程序再修改节点信息，会出现怎样的效果呢，如下所示：

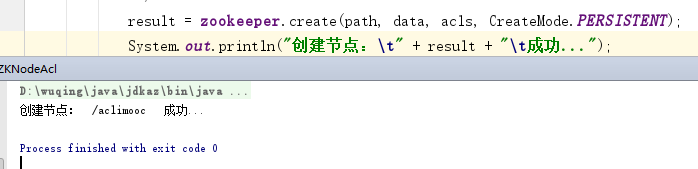


然后再运行看下，我们的dataversion输出为1，如下所示：



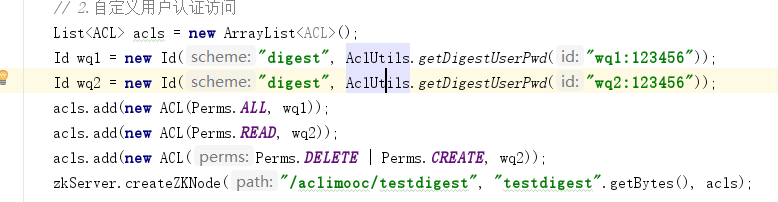
## 8.10、ACL权限列表

### 8.10.1 默认匿名权限

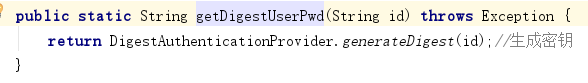


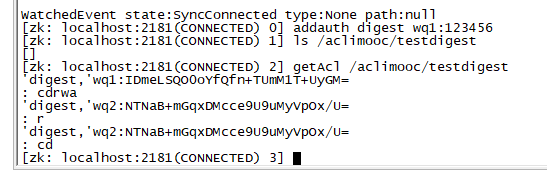
### 8.10.2 自定义用户权限

（1）自定义权限并通过用户权限创建节点

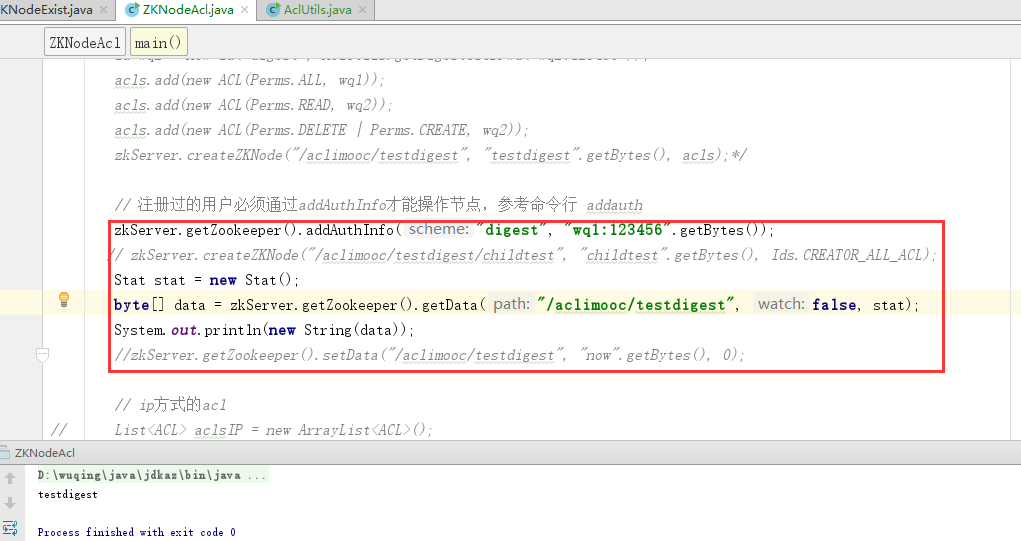


这里AclUtils定义了一个生成密钥的方法，如下所示：

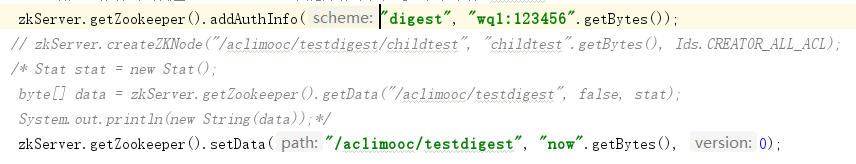




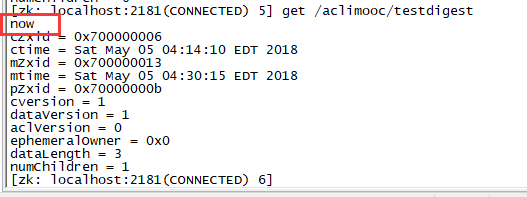
（2）登录读取权限的用户读取节点数据



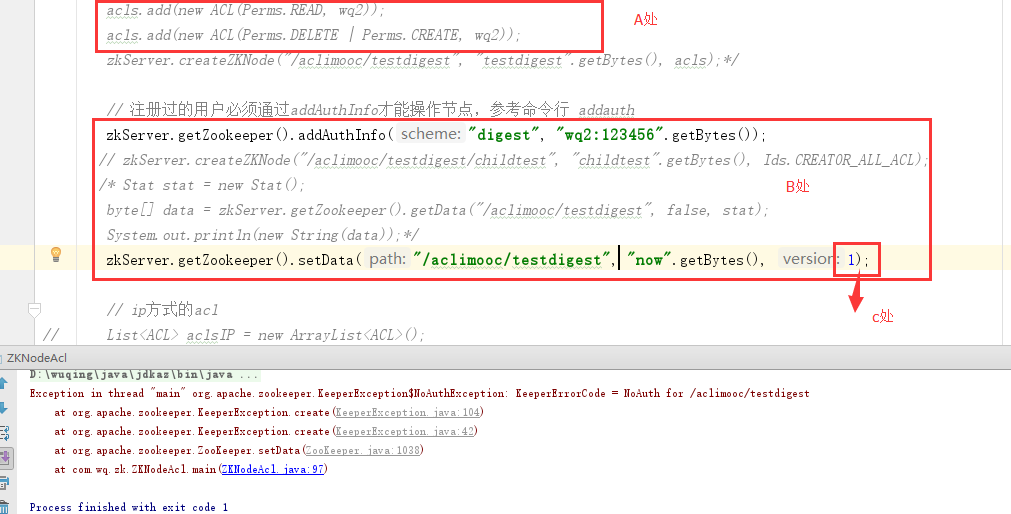
（3）为节点设值



可以看到节点数据已被成功修改，如下所示：



这里我们可以测试下没有修改权限的用户，会出现怎样的结果呢？



如预期的结果一样，因为在一开始自定义权限时，未给“wq2”这个用户添加修改权限（见截图A处），所以会报没权限的错误信息（见截图中的控制台信息）。

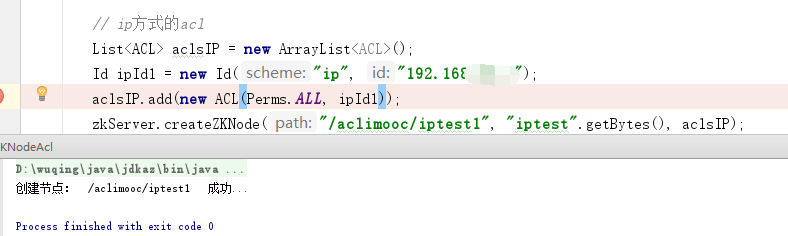
这里，截图C处由于上一小节我们修改过数据，所以这里要注意davaVersion已经变为1了。

### 8.10.3 ip权限

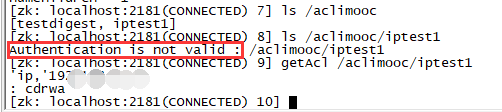
（1）首先通过打开cmd输入“ipconfig”查看自己的ip地址，如下所示：

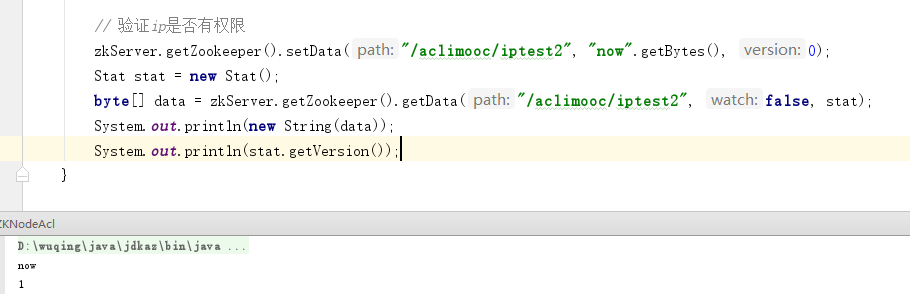


（2）将本地ip地址添加至ip权限列表中，如下所示：



（3）验证是否有权限





这里注意下，由于我ip地址写错了，所以重新创建了个iptest2节点。可以看到通过本地ip所在的客户端能够成功修改节点数据并获取节点数据版本。

## 8.11 总结

通过本章的学习，我们可以发现zk原生api的不足之处：

（1）超时重连，不支持自动，需要手动操作；

（2）Watch注册一次后会失效；

（3）不支持递归创建节点。

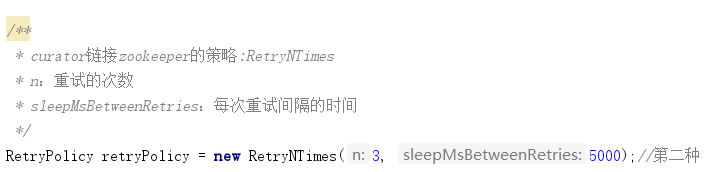
所以，下一章我们将学习Apache curator的zk java客户端，它是Apache的开源项目，解决了原生api的watcher的注册一次就失效的问题且更加简单易用。

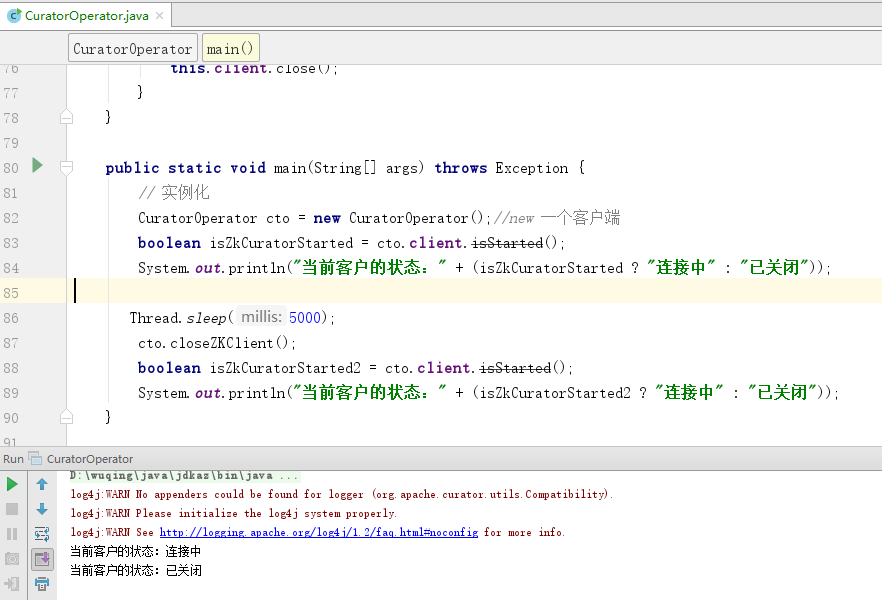
# 九、Apache curator

Apache curator提供了更多解决方案且实现简单：比如分布式锁，它也提供了常用的Zookeeper工具类，其编程风格更爽。

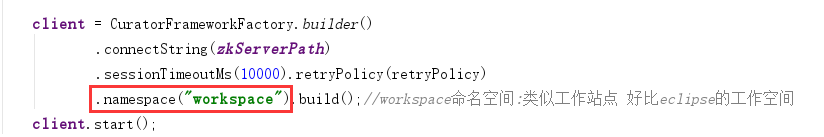
## 9.1、建立Curator与zkServer的连接

方式有多种，这里只测试一种，如下所示：



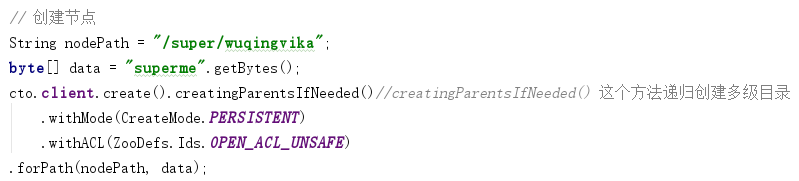


## 9.2、zk命名空间以及创建节点

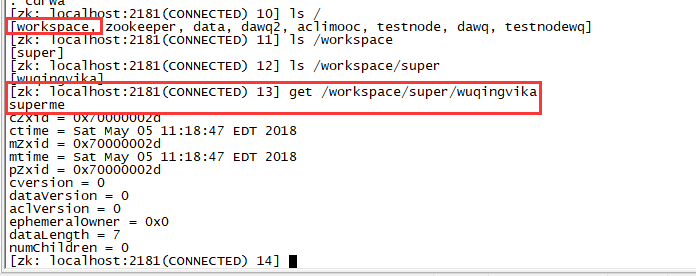


以后创建节点会默认在此空间下。

下面演示创建多级目录“/super/wuqingvika”在“workspace”命名空间下，程序实现如下所示：

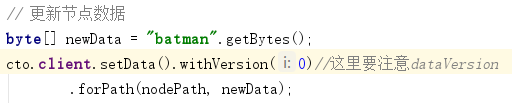


创建成功，通过命令行查看如下所示：

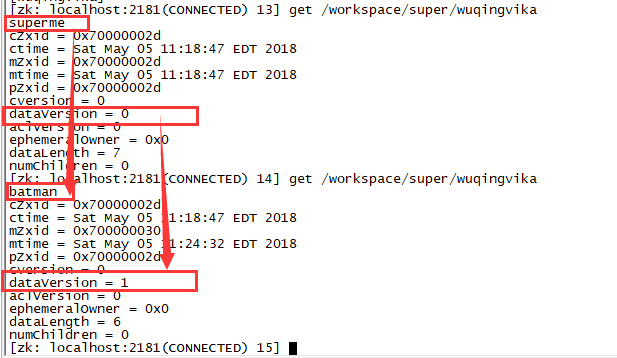


## 9.3、修改节点数据及删除节点

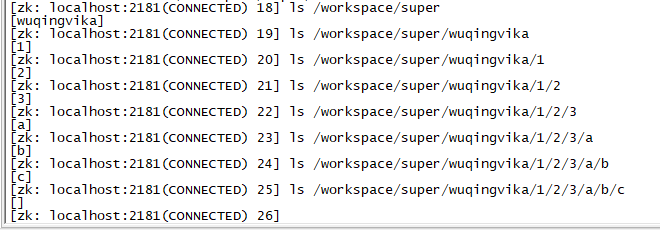
（1）更新节点数据



通过上面程序运行，成功修改节点数据，如下所示：



（2）为了演示多级目录删除，这里我重新创建一下多级目录，如下所示：



然后我再执行删除“/super/wuqingvika”代码如下所示：



接下来通过命令查看”super“下的所有节点都被删除了，如下所示：



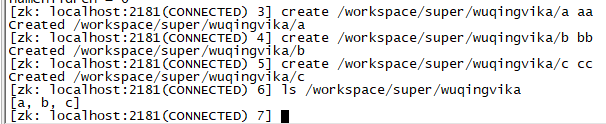
## 9.4、获取节点数据

由于上一小节我把节点都删了，所以我重新创建了节点再执行读取操作的，读取节点数据如下所示：



## 9.5、获取子节点数据

这里我为了演示我特意在“super/wuqingvika”下创建了节点a、b、c分别对应节点数据aa、bb、cc，如下所示：



遍历子节点，打印结果如下所示：



## 9.6、判断节点是否存在

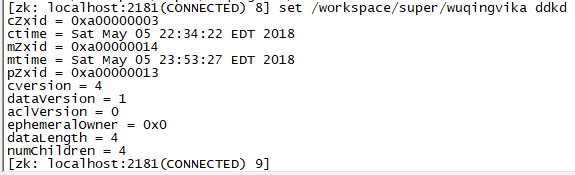


如果放一个不存在的节点路径，则会打印空，如下所示：

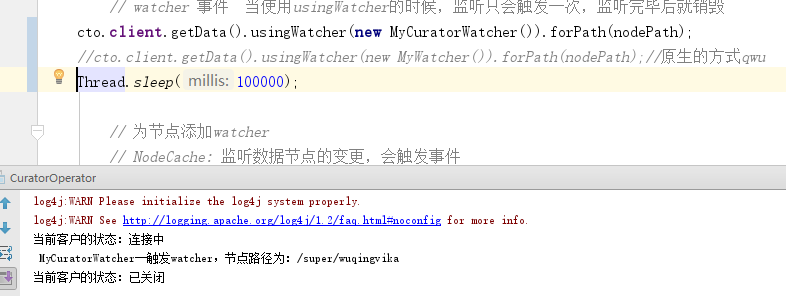


## 9.7、UsingWatcher

为了演示，我修改了”/super/wuqingvika”的节点数据，如下所示：



接下来程序中可以看出能成功监听到修改事件，如下所示：

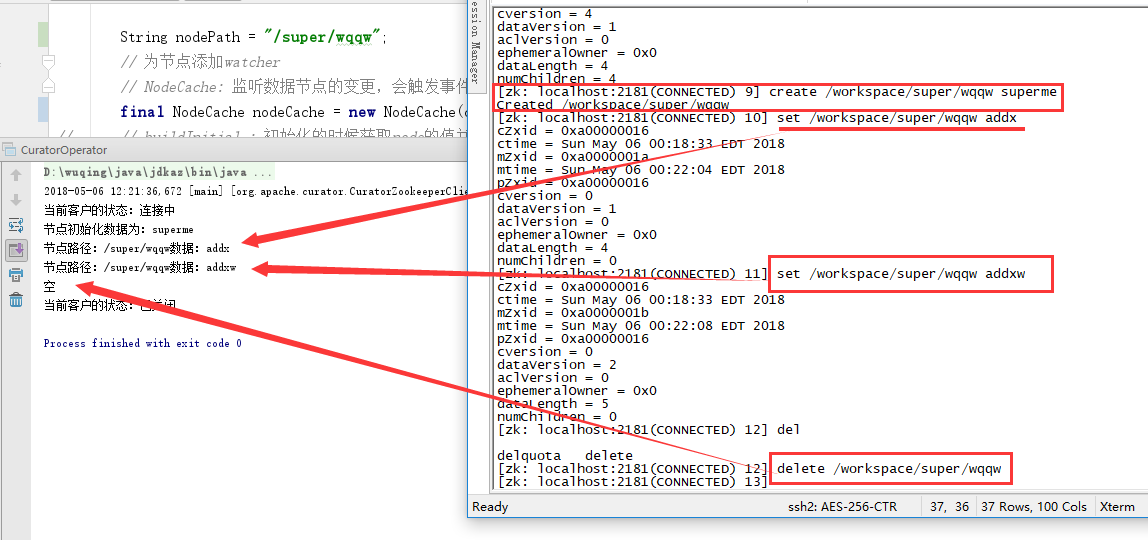


注意：使用usingWatcher的时候，监听只会触发一次，监听完毕后就销毁。

## 9.8、nodeCache一次注册N次监听

如下是一次注册N次监听的代码：



接下来，命令行进行修改节点操作和删除节点操作，如下所示：  


## 9.9、pathChildrenCache子节点监听

代码如下：



接下来便是通过命令行对子节点的一些操作，可以看到相应的监听事件打印信息，如下所示：

