# 课程实验四：Spark SQL、Spark Streaming

## 一、实验描述

本实验分为三个部分：第一部分为RDD编程，使用Scala语言编写独立应用程序实现数据去重；第二部分为使用Spark SQL读写数据库，包括在服务器上安装MySQL和通过JDBC连接数据库；第三部分为Spark Streaming编程，使用Kafka数据源编写Spark Streaming程序。

本实验在实验三中配置的环境的基础上进行实验。

## 二、实验目的

1.熟悉使用Scala编写Spark程序；

2.了解Spark RDD的工作原理；

3.掌握在Spark集群上运行程序的方法；

4.学习在Spark中通过JDBC连接MySQL数据库的方法；

5.了解Spark Streaming的工作原理和Kafka的配置方法。

## 三、实验环境

1.服务器节点数量：3；

2.系统版本：Centos 7.5；

3.Hadoop版本：Apache Hadoop 2.7.3；

4.Spark版本：Apache Spark 2.1.1；

5.JDK版本：1.8.0\_131；

6.Scala版本：scala2.11.11；

7.IDEA版本：ideaIC-2017.2.7；

8.MySQL版本：8.0；

9.Kafka版本：0.8.2.1。

## 四、实验步骤

**4.1 RDD编程——编写独立应用程序实现数据去重**

对于两个输入文件A和B，编写Spark独立应用程序，对两个文件进行合并，并剔除其中重复的内容，得到一个新文件C。下面是输入文件和输出文件的一个样例，供参考。**（要求运行截图及代码）**

输入文件A的样例如下：

20170101 x

20170102 y

20170103 x

20170104 y

20170105 z

20170106 z

输入文件B的样例如下：

sp

根据输入的文件A和B合并得到的输出文件C的样例如下：

20170101 x

20170101 y

20170102 y

20170103 x

20170104 y

20170104 z

20170105 y

20170105 z

20170106 z

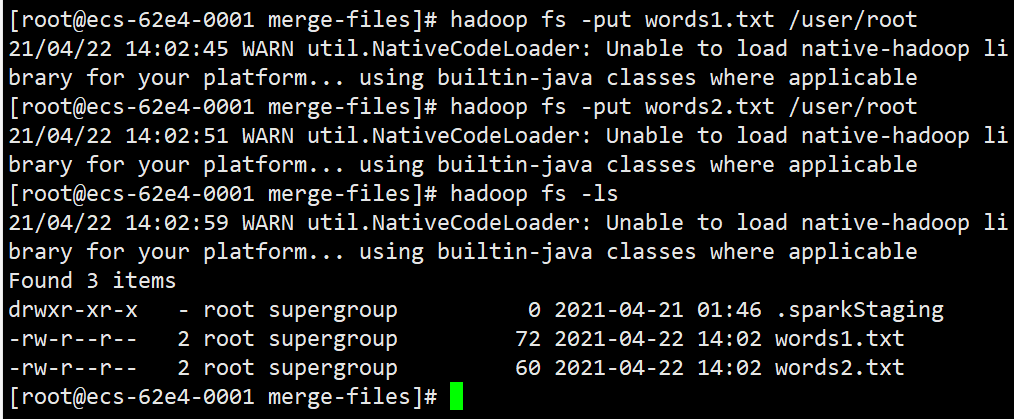
可参考方法：

* 从文件中读取数据创建RDD

val text = sc.textFile(“word.txt”)



可以把txt文件放到与jar包同一路径下，避免出现找不到文件的问题。首先将txt文件通过put命令上传到hdfs中。上传完成后可以用ls命令查看到文件已经上传到hdfs。



* 把RDD写入到文本文件中

text.saveAsTextFile(“writeback”)

此处的writeback应该为hdfs中的一个空文件夹，运行后会在文件夹中生成输出文件，用cat显示其中内容即可。



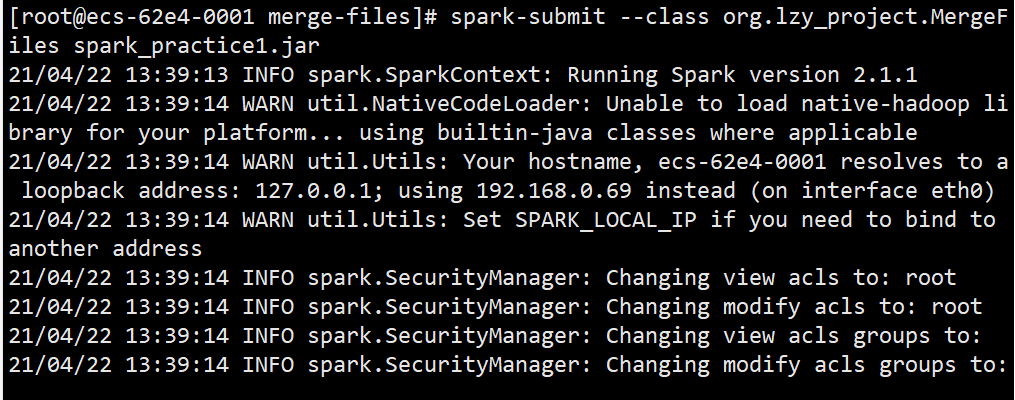
* join()对于给定的两个输入数据集(K,V1)和(K,V2)，只有在两个数据集中都存在的key才会被输出，最终得到一个(K,(V1,V2))类型的数据集。

RDD1.join(RDD2)

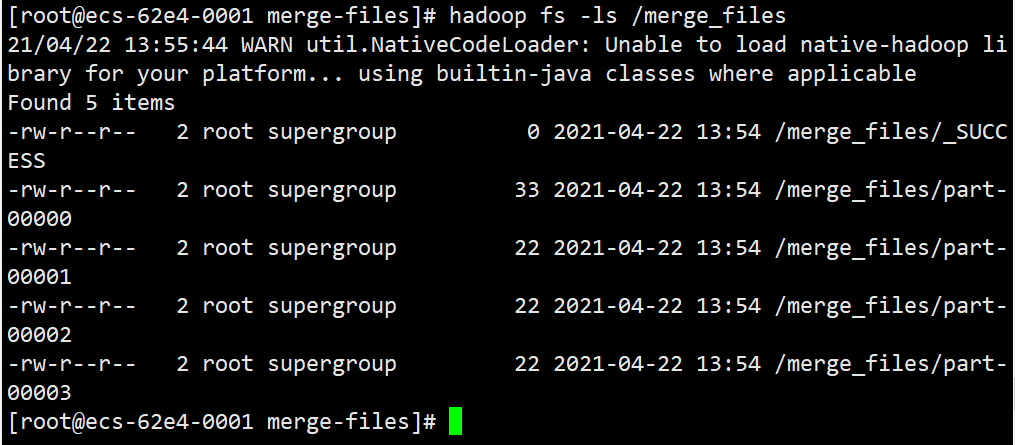
* union()合并变换将两个RDD合并为一个新的RDD，重复的记录不会被剔除。

RDD1.union(RDD2)

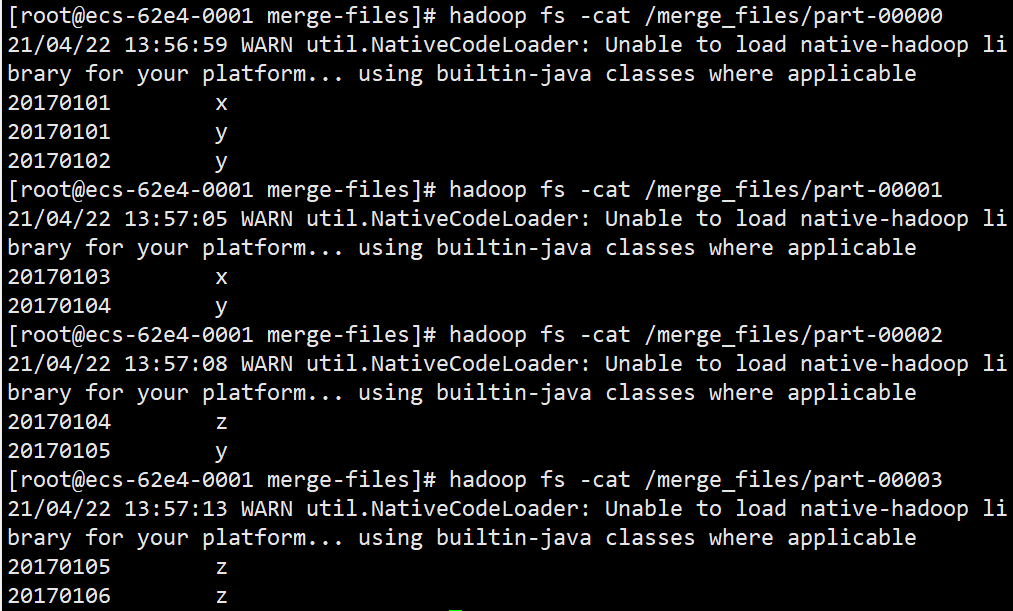
本次实验可以在上一次实验构建的IDEA项目中进行代码的编写。构建方法参考4.2节中的内容。构建完成后，将打好的jar包通过工具上传到服务器，用spark-submit命令运行。



运行完成后，用hadoop fs -ls /输出文件夹命令查看hdfs中的输出结果。



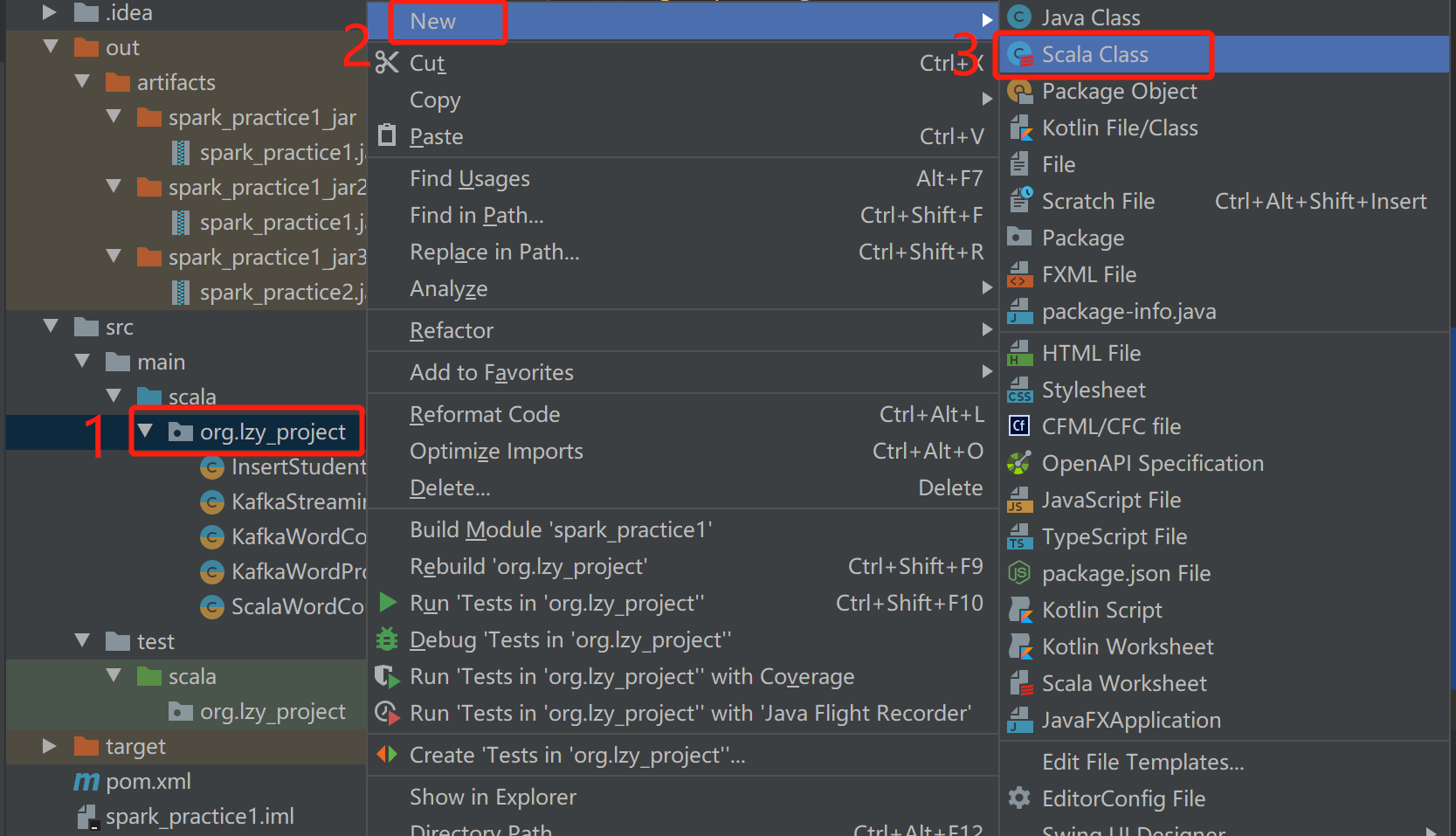
用cat命令分别查看输出文件中的内容。



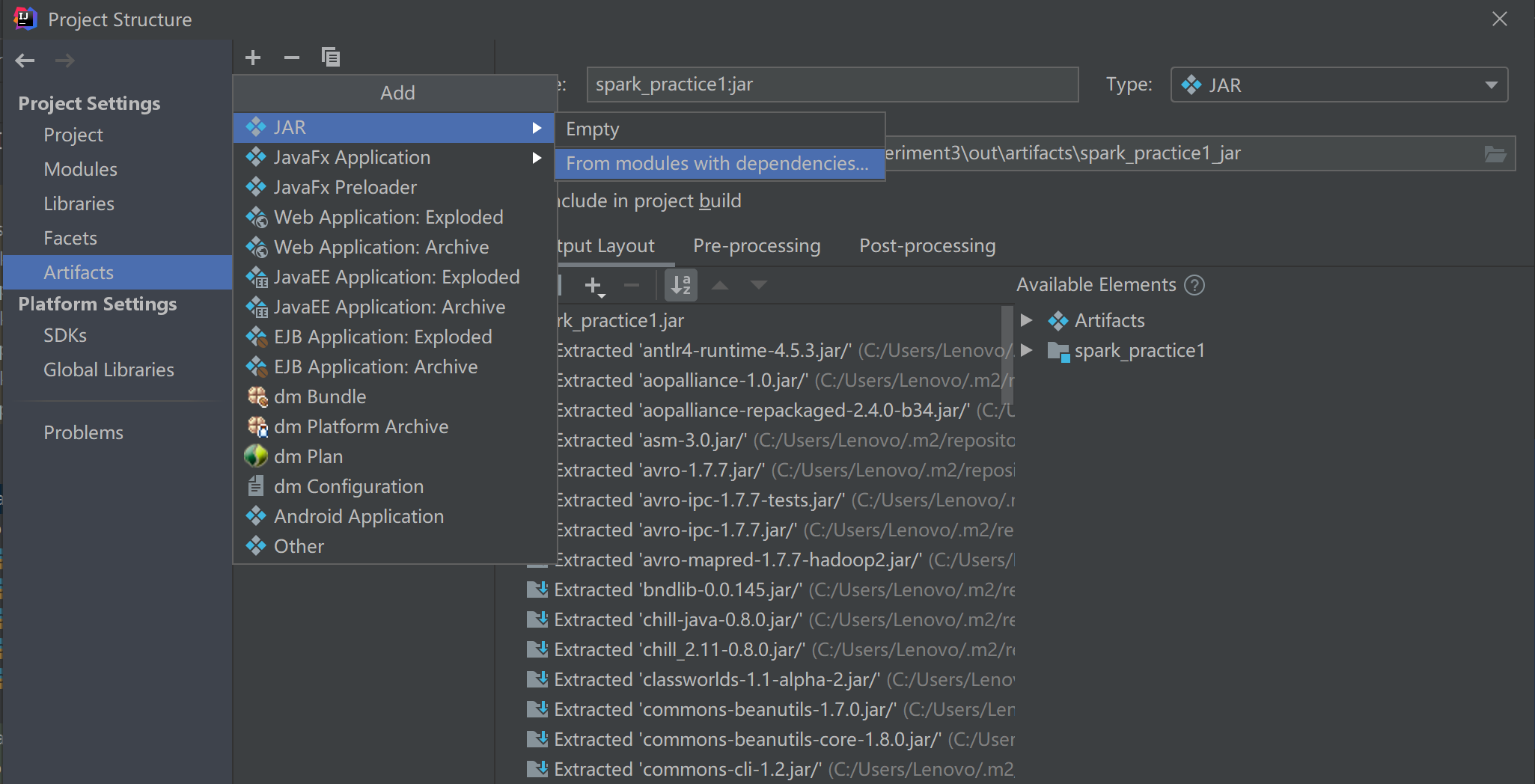
可以看到两个文件被成功合并。

**4.2 IDEA环境构建**

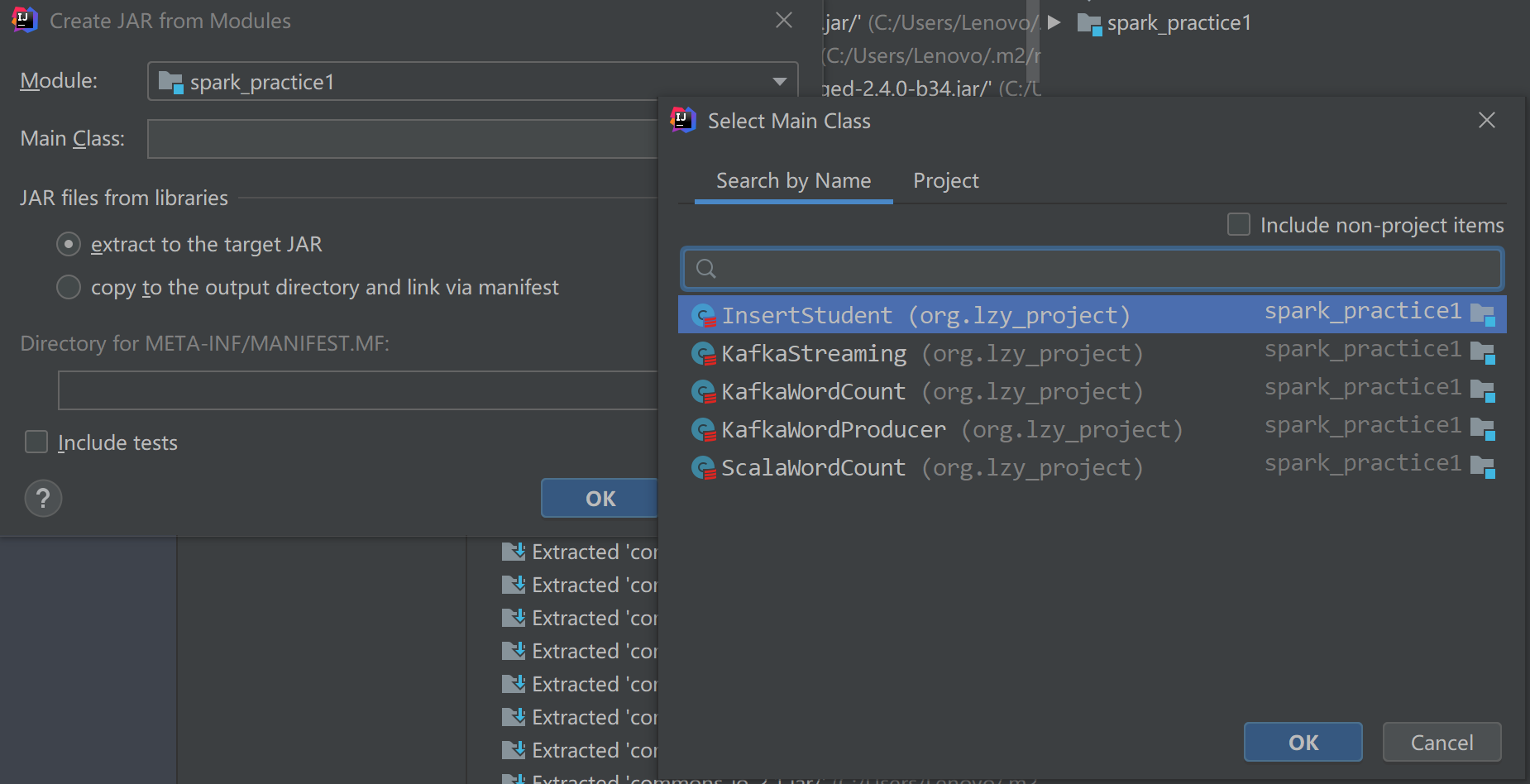
本次实验可以在上一次实验构建的IDEA项目中进行代码的编写。



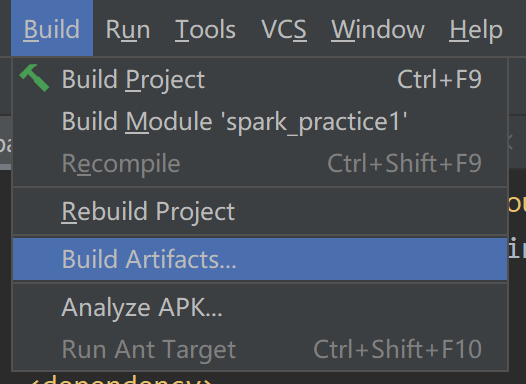
可以在原来的项目中新建一个Scala类，然后在新建的Scala类中编写Scala程序。程序编写完成后，在File->Project Structure中进行构建。点击Artifacts，点击上方的加号，选择JAR，From modules with dependencies。



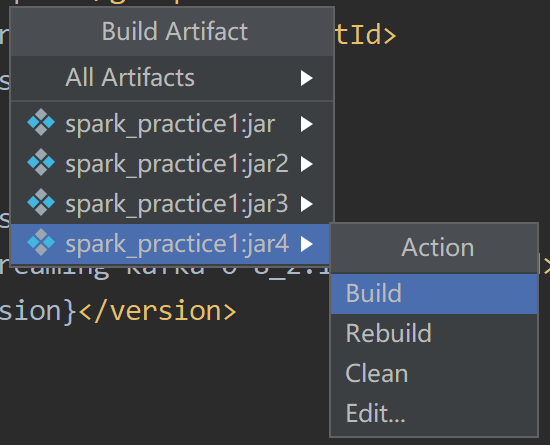
在新打开的窗口中选择主类，可以在已经创建的Scala类中进行选择。创建完成后点击OK。



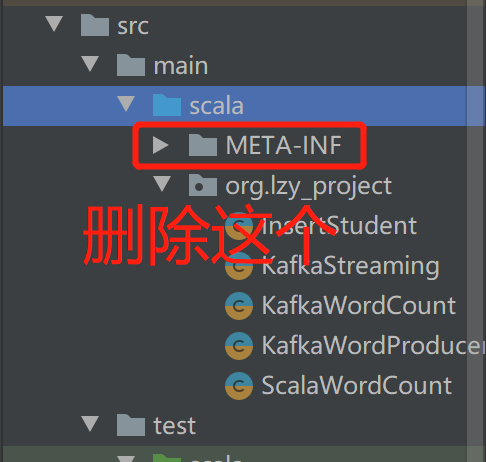
新建Artifacts之后，点击Build->Build Artifacts进行构建。



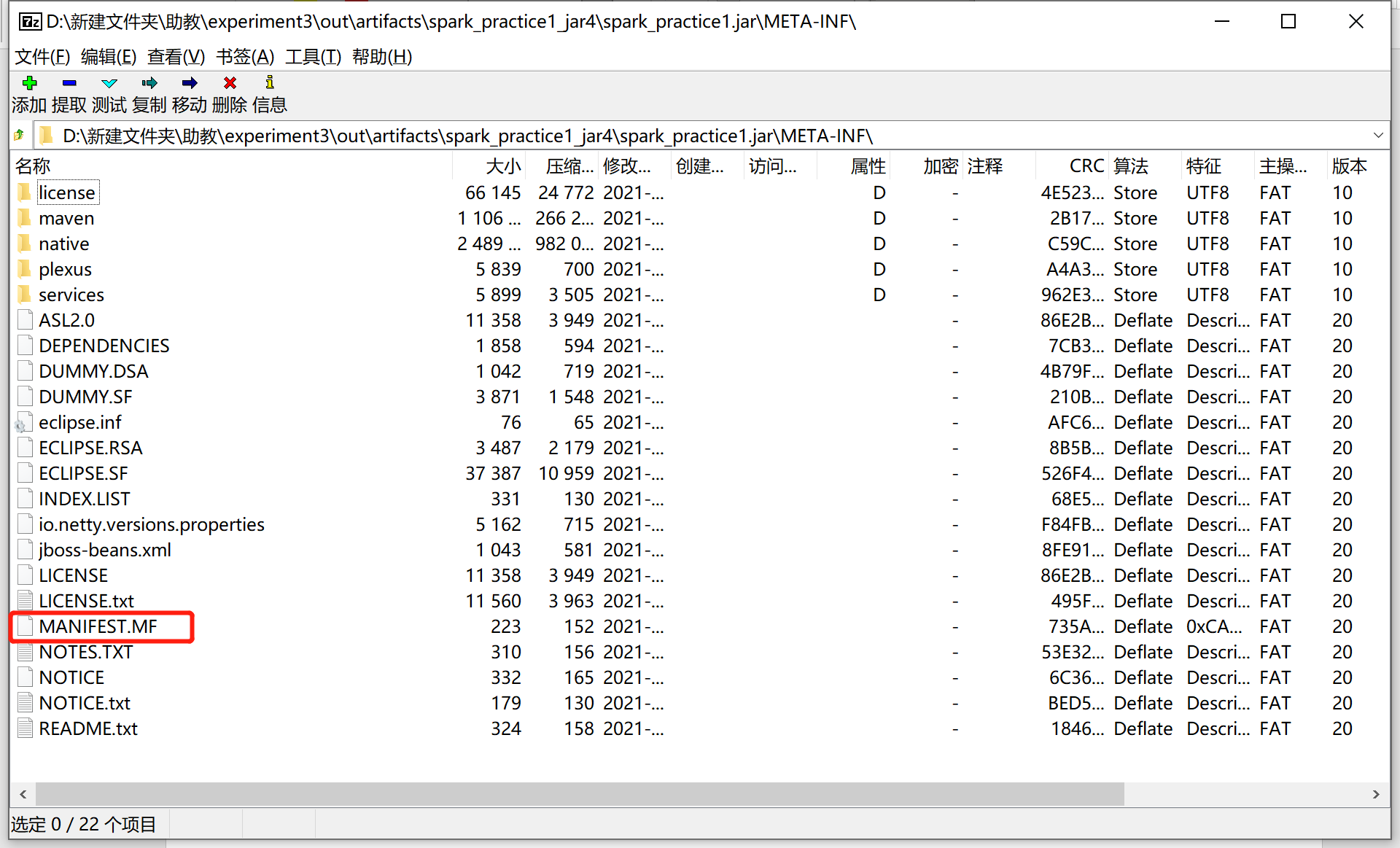
在弹出的窗口中选择刚刚生成的Artifact，点击Build进行构建。



注意：每次构建完之后，要删除项目文件夹下的META-INF文件夹以及文件夹下的所有内容，才能进行下一次构建。



构建完成后可以在项目文件夹下的out/artifacts文件夹下找到生成的jar包，将jar包用压缩工具打开，删除其中META-INF文件夹下的MANIFEST.MF文件。



删除完成后，就可以将jar包上传到服务器上来运行啦。

**4.3 使用Spark SQL读写数据库**

**步骤1**：下载并安装MySQL

下载：wget <https://dev.mysql.com/get/mysql80-community-release-el7-3.noarch.rpm>

安装源：yum localinstall mysql80-community-release-el7-3.noarch.rpm

安装：yum -y install mysql-community-server

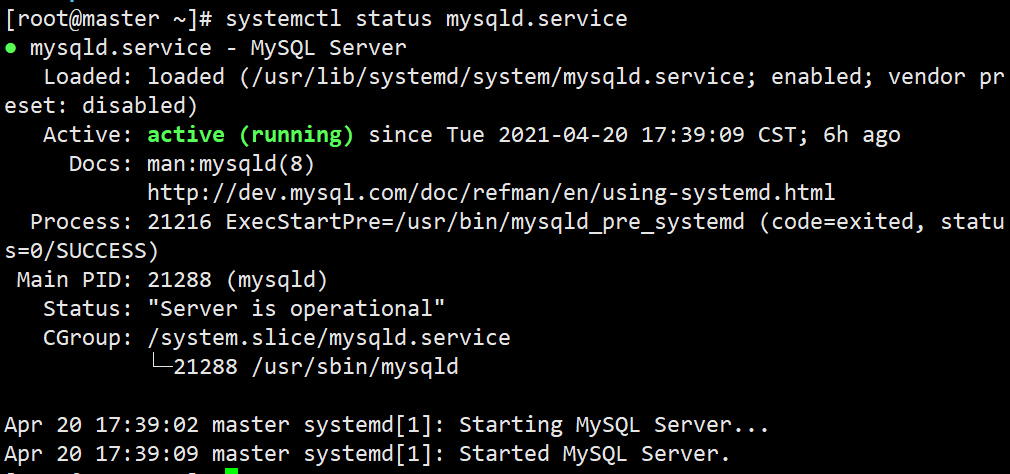
**步骤2**：MySQL数据库设置

首先启动MySQL：

systemctl start mysqld.service

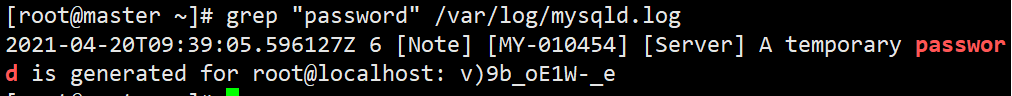
查看MySQL运行状态：

systemctl status mysqld.service



此时MySQL已经开始正常运行，不过要想进入MySQL还得先找出此时root用户的密码，通过如下命令可以在日志文件中找出密码：

grep "password" /var/log/mysqld.log



如下命令进入数据库：

mysql -uroot -p

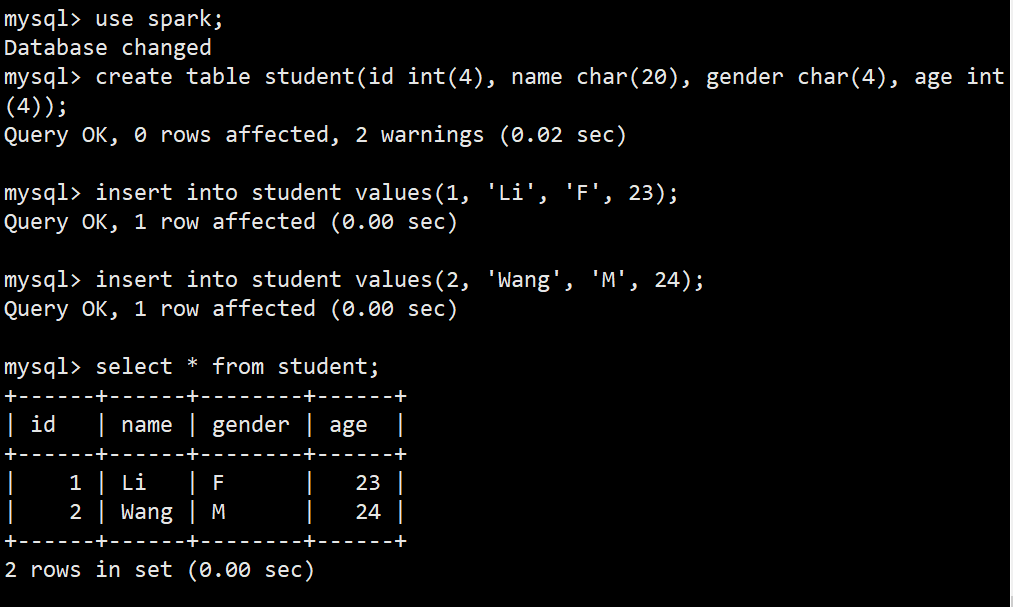
输入初始密码（是上面图片最后面的v)9b\_oE1W-\_e，此时不能做任何事情，因为MySQL默认必须修改密码之后才能操作数据库：

ALTER USER 'root'@'localhost' IDENTIFIED BY 'new password';

其中‘new password’替换成你要设置的密码，注意:密码设置必须要大小写字母数字和特殊符号（,/';:等）,不然不能配置成功。

**步骤3**：通过JDBC连接数据库

在MySQL Shell环境中，输入下面SQL语句完成数据库和表的创建：



要想顺利连接MySQL数据库，还需要使用MySQL数据库驱动程序。请到MySQL官网下载MySQL的JDBC驱动程序，把该驱动程序解压缩到Spark的安装目录下：

wget <https://dev.mysql.com/get/Downloads/Connector-J/mysql-connector-java-8.0.24.tar.gz>

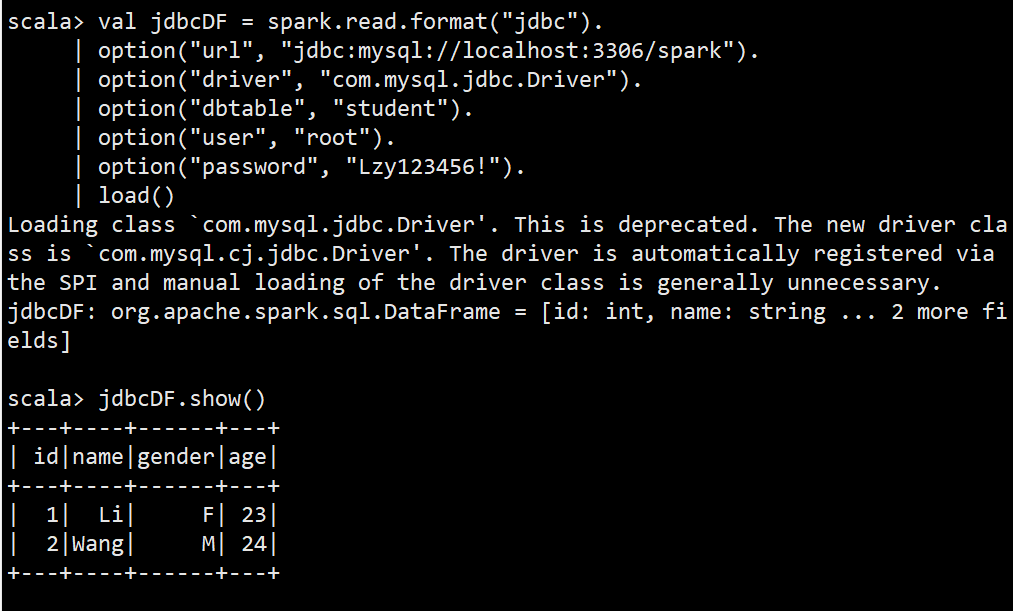
tar -xzvf mysql-connector-java-8.0.24.tar.gz

mv mysql-connector-java-8.0.24/mysql-connector-java-8.0.24.jar /root/spark-2.1.1-bin-hadoop2.7/jars/

启动一个spark shell。启动spark shell时，必须指定MySQL连接驱动jar包，命令如下：

./bin/spark-shell --jars /root/spark-2.1.1-bin-hadoop2.7/jars/mysql-connector-java-8.0.24.jar --driver-class-path /root/spark-2.1.1-bin-hadoop2.7/jars/mysql-connector-java-8.0.24.jar

spark.read.format(“jdbc”)操作可以实现对MySQL数据库的读取。执行以下命令连接数据库，读取数据并显示：**（要求贴出命令和结果的截图）**

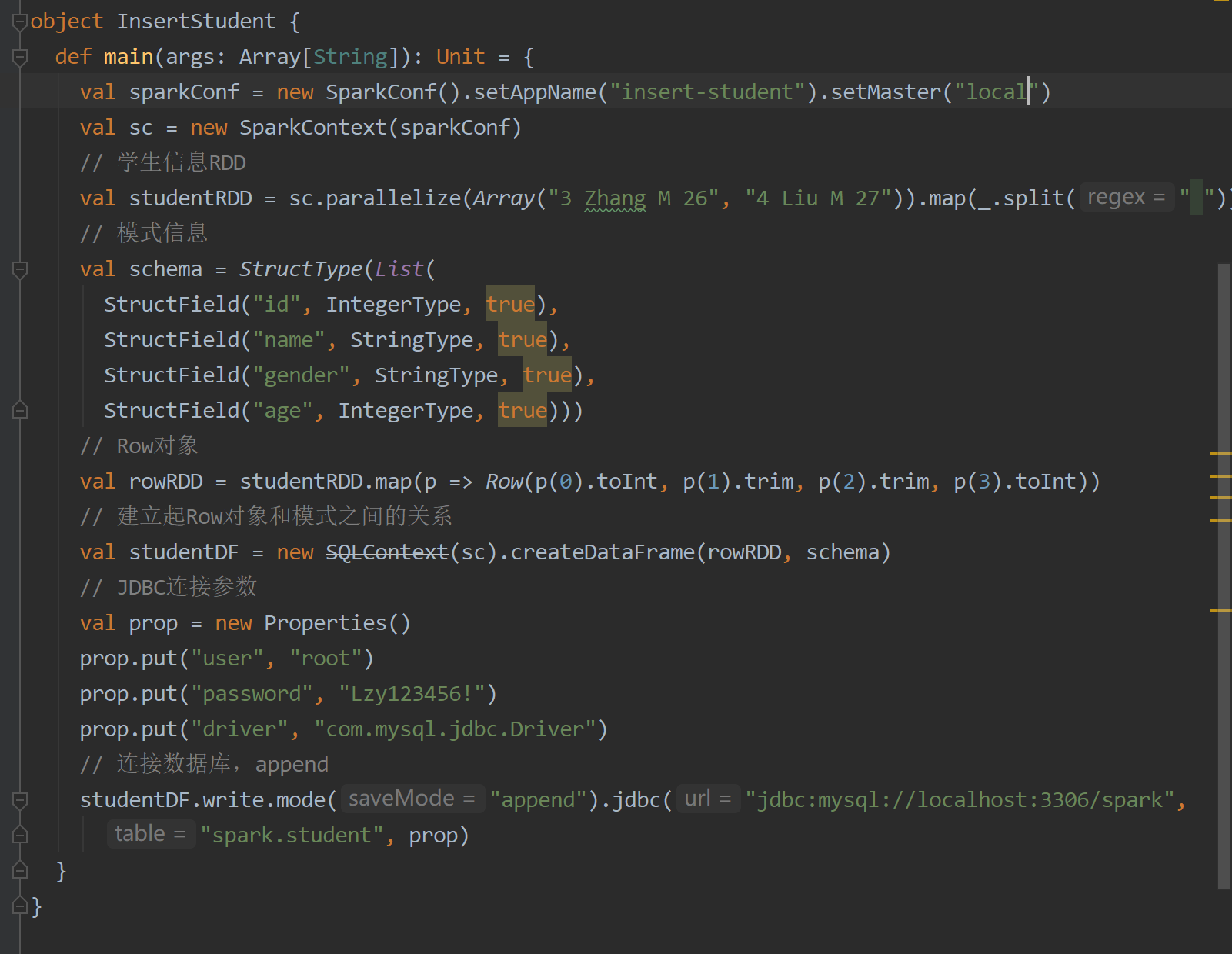


在通过JDBC连接MySQL数据库时，需要通过option()方法设置相关的连接参数，下表给出了各个参数的含义。

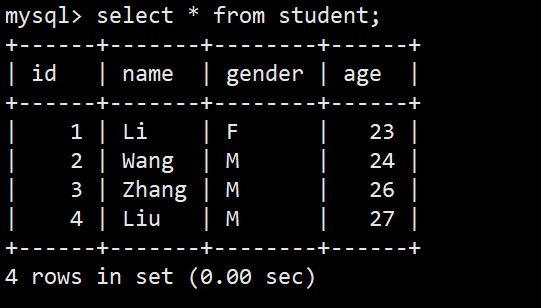
|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数的值 | 含义 |
| url | jdbc:mysql://localhost:3306/spark | 数据库的连接地址 |
| driver | com.mysql.jdbc.Driver | 数据库的JDBC驱动 |
| dbtable | student | 所要访问的表 |
| user | root | 用户名 |
| password |  | 用户密码 |

**步骤4**：向MySQL数据库写入数据

在idea中编写向数据库中插入数据的代码：



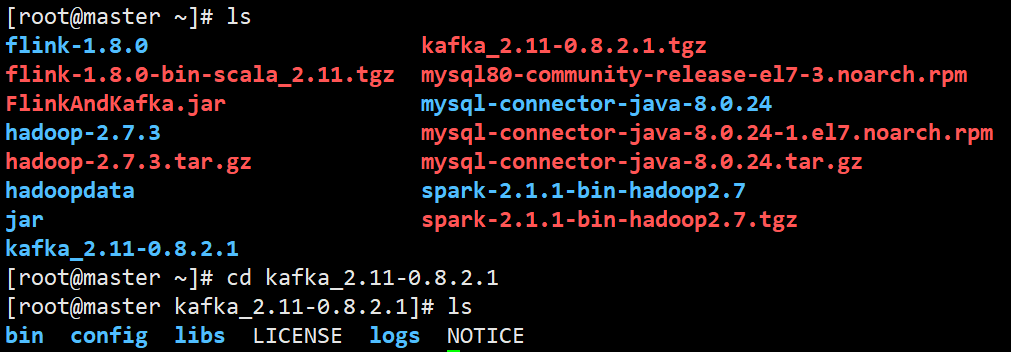
打包并删除jar包中的MANIFEST.MF文件，将jar包上传到服务器，使用spark-submit命令运行程序。运行之后，可以到MySQL shell环境中使用sql语句查询student表，可以发现新增加的两条记录。**（可以在命令行里打印一些文字表示运行成功，要求截图中包含代码和运行结果）**



**4.4 Spark Streaming实验**

**步骤1**：安装Kafka

将kafka\_2.11-0.8.2.1.tgz上传到服务器，然后用tar命令解压。



**步骤2**：启动Kafka

首先用yum install screen命令安装screen，然后就可以使用screen来切换终端窗口而不用建立多个连接。

**screen -S 窗口名**：**新建一个自定义窗口名的窗口**

**screen -r 窗口名**：**切换到指定的窗口**

**在screen窗口中按下ctrl+a d即可离开窗口**

首先使用screen -S kafka1创建一个新窗口，输入以下命令启动Zookeeper服务：

cd kafka\_2.11-0.8.2.1

./bin/zookeeper-server-start.sh config/zookeeper.properties

执行上面命令之后，终端窗口会返回一系列信息，然后停住不动，此时Zookeeper服务已经启动，正在处于服务状态。此时可使用ctrl+a d来退出此窗口。

继续使用screen -S kafka2创建另一个新窗口，输入以下命令启动Kafka服务：

cd kafka\_2.11-0.8.2.1

./bin/kafka-server-start.sh config/server.properties

执行上面命令之后会和上一步有同样的效果，同样使用ctrl+a d退出窗口。这样，Kafka就会在后台运行，即使关闭了这个终端，Kafka也会在后台一直运行。

**步骤3**：测试Kafka是否正常工作

下面测试一下Kafka是否可以正常使用，输入以下命令创建一个自定义名称为“wordsendertest”的topic：

cd kafka\_2.11-0.8.2.1

./bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic wordsendertest

这个topic叫wordsendertest，2191是Zookeeper默认的端口号，--partition是topic里面的分区数，--replication-factor是备份的数量，在Kafka集群中使用。

下面用生产者来产生一些数据，在当前终端内输入下面命令，启动生产者：

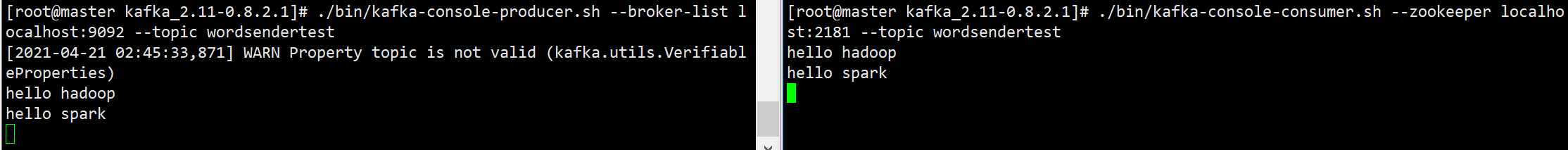
./bin/kafka-console-producer.sh --broker-list localhost:9092 --topic wordsendertest

接下来再打开一个终端，在终端内输入下面命令，启动消费者：

cd kafka\_2.11-0.8.2.1

./bin/kafka-console-consumer.sh --zookeeper localhost:2181 --topic wordsendertest

执行后，就可以在生产者的终端内输入一些单词，这些单词就是数据源，会被Kafka捕捉到以后发送给消费者，可以看到在消费者侧会显示输入的单词。



步骤4：编写消费者程序

编写代码时要在pom.xml的dependency中加入以下内容：

<dependency>

<groupId>org.apache.spark</groupId>

<artifactId>spark-streaming\_2.11</artifactId>

<version>${spark.version}</version>

</dependency>

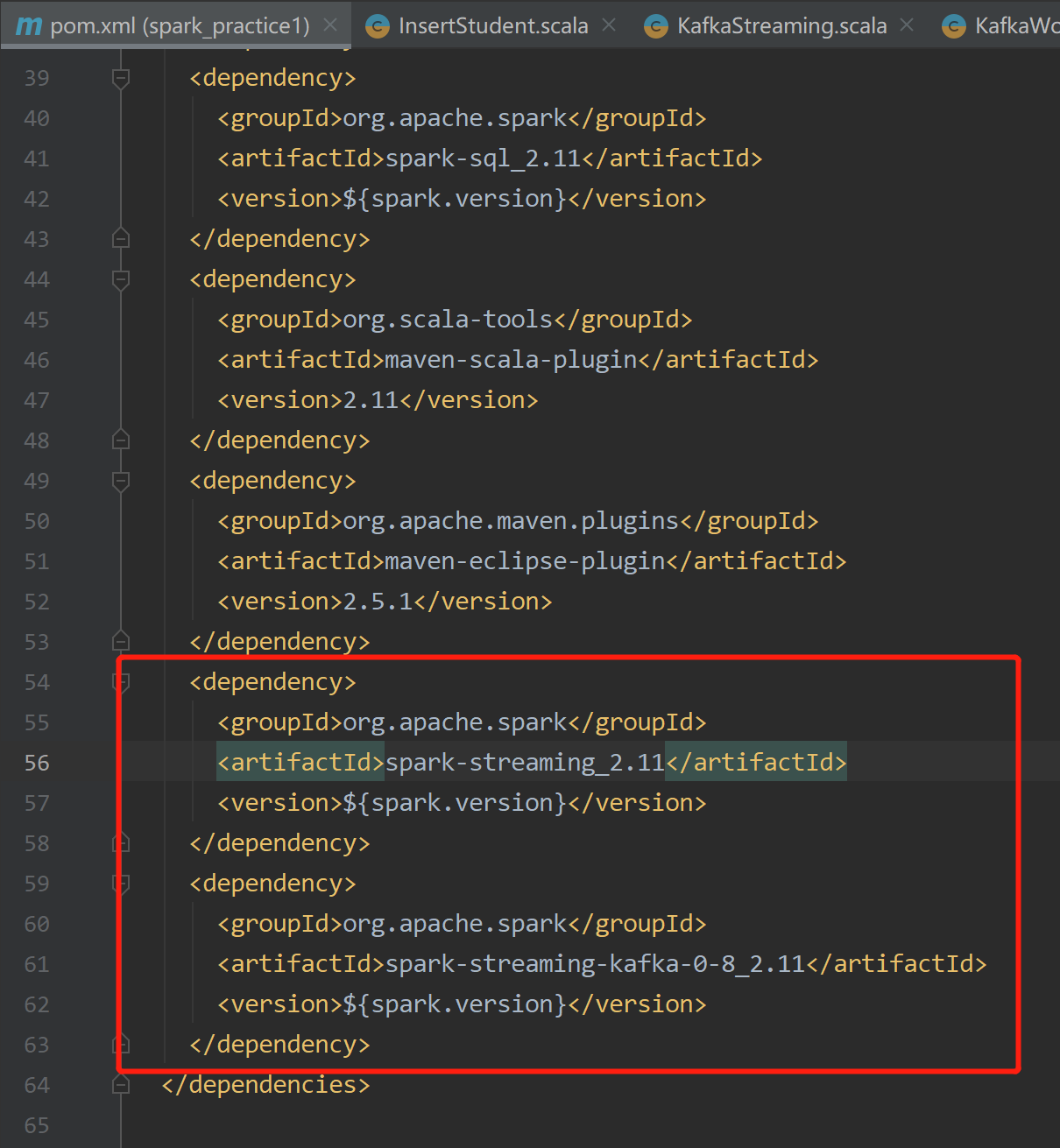
<dependency>

<groupId>org.apache.spark</groupId>

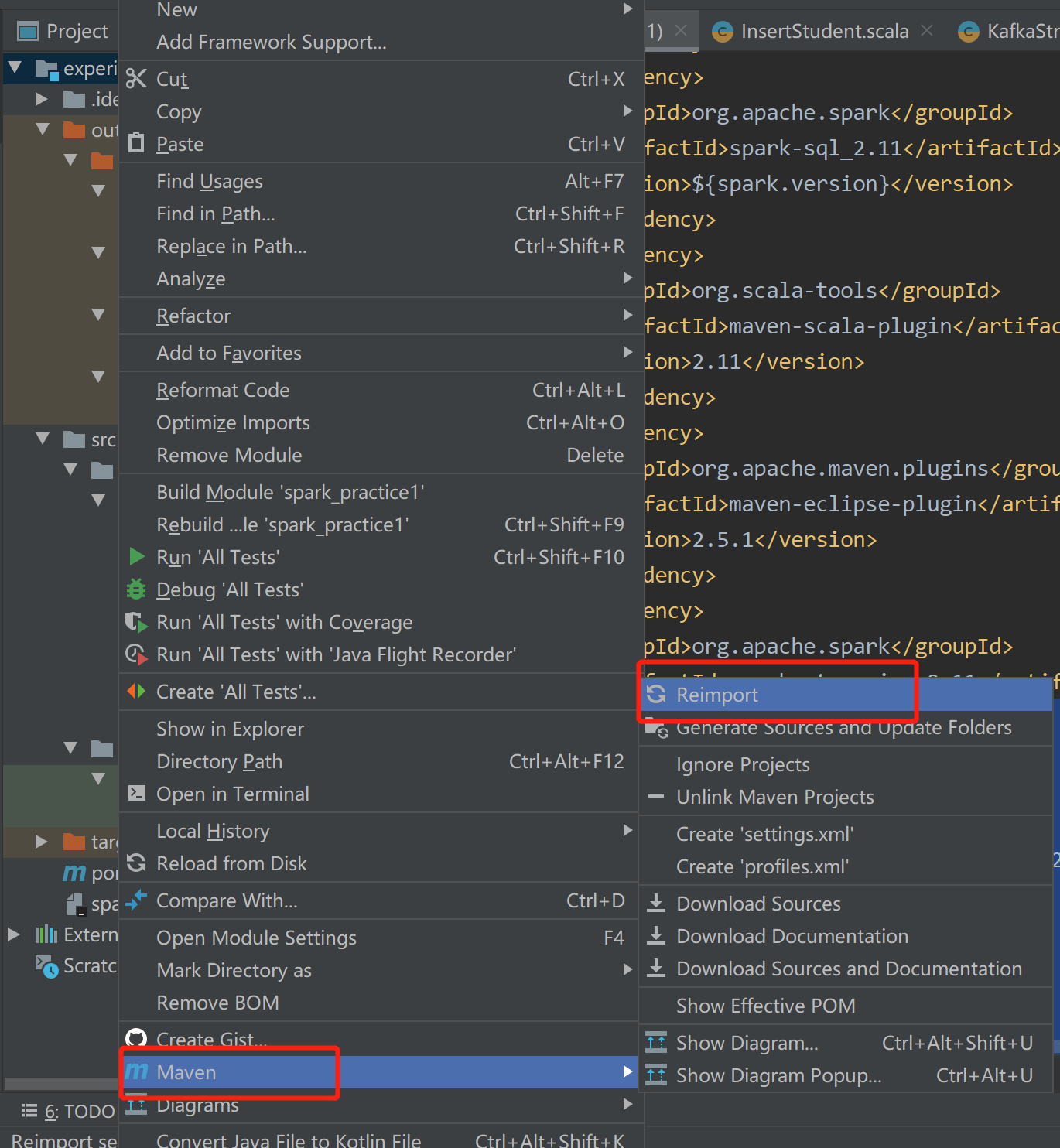
<artifactId>spark-streaming-kafka-0-8\_2.11</artifactId>

<version>${spark.version}</version>

</dependency>



添加完成后右键项目名，选择Maven->Reimport重新导入所需依赖。



下图是Spark Streaming的参考代码：



代码第13行中local[2]表示本地多线程运行；

第14行创建了一个StreamingContext对象，在对数据流进行分段时，每10秒切成一个分片；

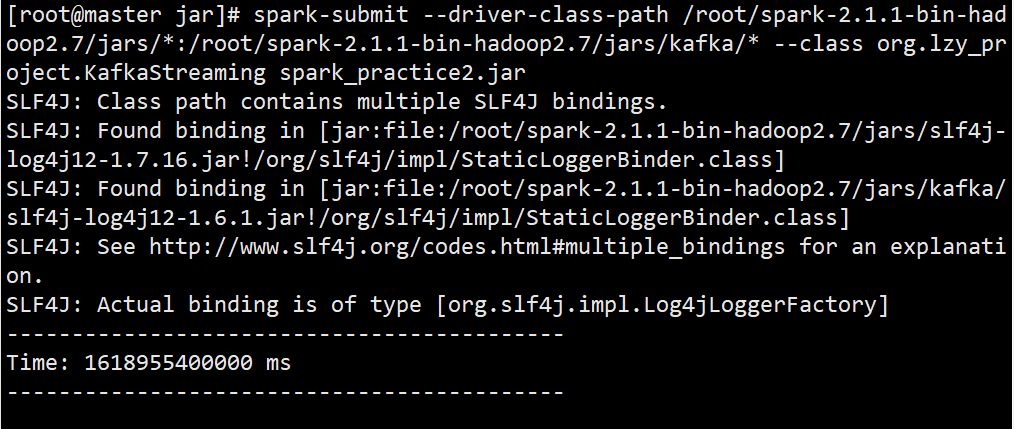
第21行使用createStream方法创建了一个Receiver用来接收数据；

第28行用start方法来开始接收数据和处理流程；

第29行用awaitTermination()方法来等待流计算进程结束。

打包并删除jar包中的MANIFEST.MF文件，将jar包上传到服务器，使用spark-submit命令运行程序。

spark-submit --driver-class-path /root/spark-2.1.1-bin-hadoop2.7/jars/\*:/root/spark-2.1.1-bin-hadoop2.7/jars/kafka/\* --class org.lzy\_project.KafkaStreaming spark\_practice2.jar （其中的项目名称改成自己的）



出现上图的结果说明消费者程序运行成功，此时需要再打开一个生产者终端，在新打开的终端中运行生产者：

cd kafka\_2.11-0.8.2.1

./bin/kafka-console-producer.sh --broker-list localhost:9092 --topic wordsendertest

此时可以在生产者终端中输入一些单词，可以看到消费者程序会对生产者中输入的单词统计词频。**（要求截图中包含生产者和消费者终端的输出）**

