**

**HUNAN UNIVERSITY**

《数据仓库技术与应用》

实验报告十

|  |  |
| --- | --- |
| **报告名称：** | Spark+Kafka构建实时分析Dashboard |
| **学生姓名：** | 杨超然 |
| **学生学号：** | 202106060220 |
| **专业班级：** | 电商2102班 |
| **学 院：** | 工商管理学院 |
| **指导老师：** | 周中定 |
| **日 期：** | 2023.4.24 |

目录

[一、实验过程 3](#_Toc779743114)

[（一）实验环境配置 3](#_Toc1197303004)

[（二） 数据处理和python操作Kafka 6](#_Toc1149815838)

[2.1数据预处理 6](#_Toc1896933560)

[2.2Python操作Kafka 7](#_Toc220119558)

[（三） Structured Streaming实时处理数据 9](#_Toc1582571172)

[3.1配置Spark开发Kafka环境 9](#_Toc1688719709)

[3.2建立pySpark项目 10](#_Toc1168270411)

[3.3测试程序 11](#_Toc677813156)

[（四）结果展示 12](#_Toc1752449204)

[二．易错点总结 14](#_Toc675553023)

[（一）无法用Kafka创建topic 14](#_Toc273615872)

[（二）无法安装Kafka\_python 15](#_Toc899472477)

[（三）无法导入Kafka库 16](#_Toc1296529706)

[（四） 无法启动startup.sh 16](#_Toc258202633)

[（五） ClassNotFound错误 17](#_Toc1694685891)

[三、学习小记 18](#_Toc510159876)

[（一）实验心得 18](#_Toc1502317108)

[（二）知识点笔记 20](#_Toc1478396377)

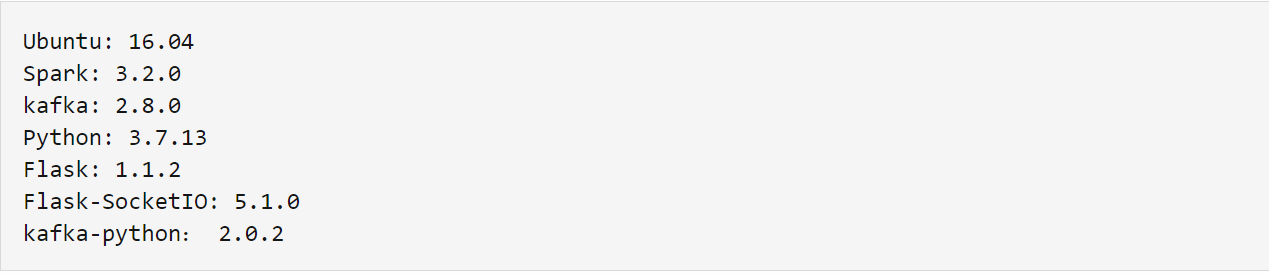
[2.1流数据与流计算 20](#_Toc1022112449)

[2.2 Spark Streaming 21](#_Toc922237990)

# 一、实验过程

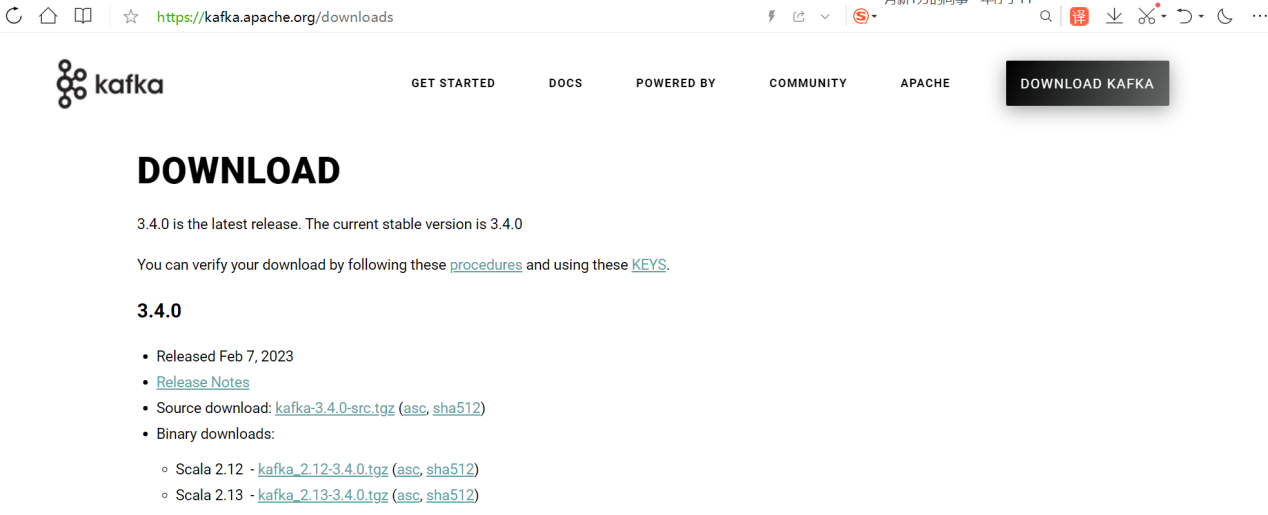
## （一）实验环境配置

本次实验环境如下（版本号仅供参考，可使用不同版本）：



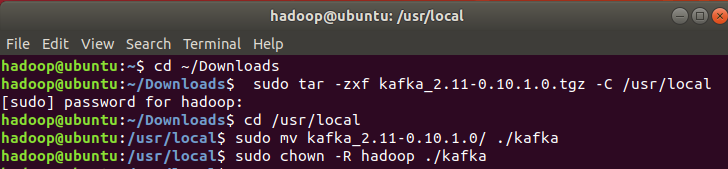
Kafka安装（Ubuntu系统）：

在官网中下载稳定版本0.10.1.0的kafka.此安装包内已经附带zookeeper,不需要额外安装zookeeper：

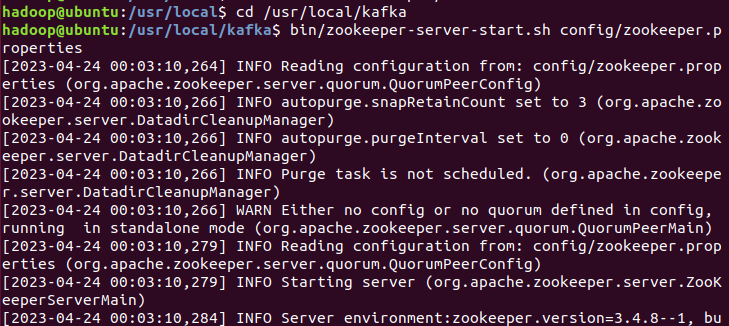




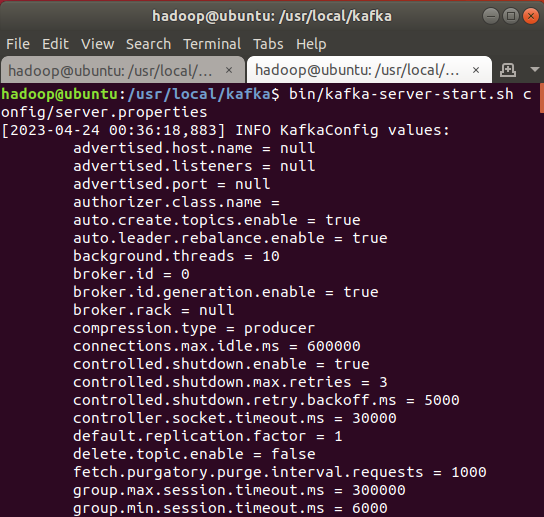
然后在terminal中执行安装命令：



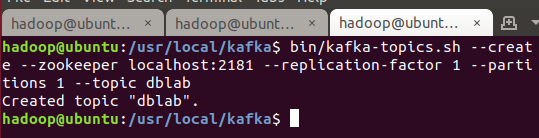
再进行简单的测试：



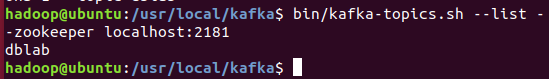
命令执行后不会返回Shell命令输入状态,zookeeper就会按照默认的配置文件启动服务,请千万不要关闭当前终端.启动新的终端，输入如下命令：



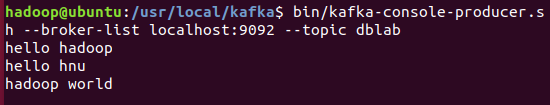
kafka服务端就启动了,请千万不要关闭当前终端。启动另外一个终端,输入如下命令:



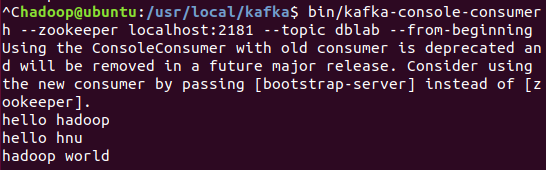
topic是发布消息发布的category,以单节点的配置创建了一个叫dblab的topic.可以用list列出所有创建的topics,来查看刚才创建的主题是否存在：



可以在结果中查看到dblab这个topic存在。接下来用producer生产点数据：

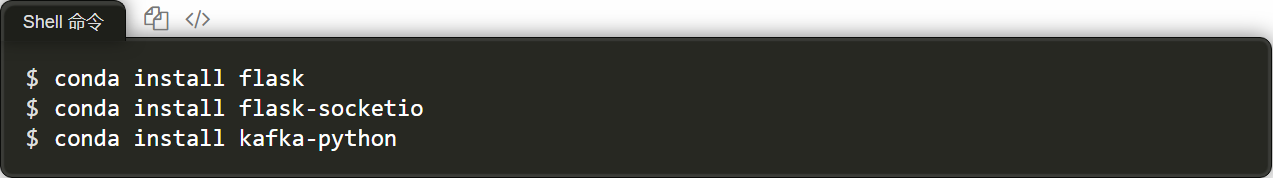


然后再次开启新的终端或者直接按CTRL+C退出。然后使用consumer来接收数据,输入如下命令：

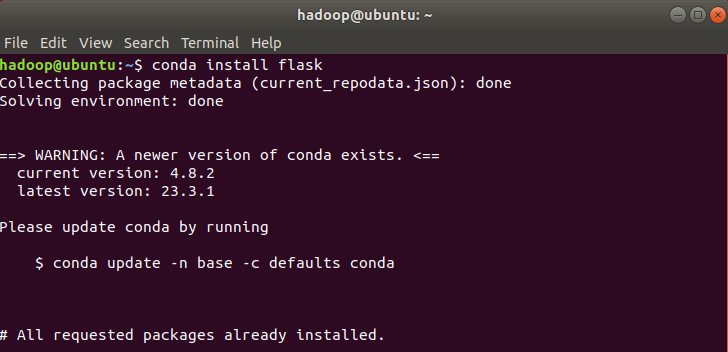


可以看到我们刚刚输入的三行数据都已经被成功接收。

接下来再terminal中安装python依赖库：



安装成功后显示如下：

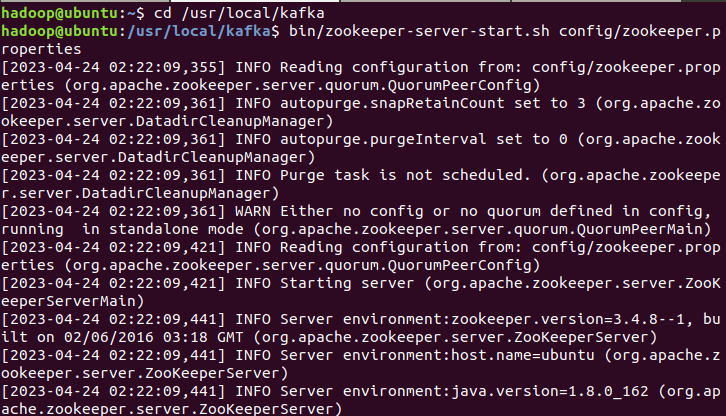


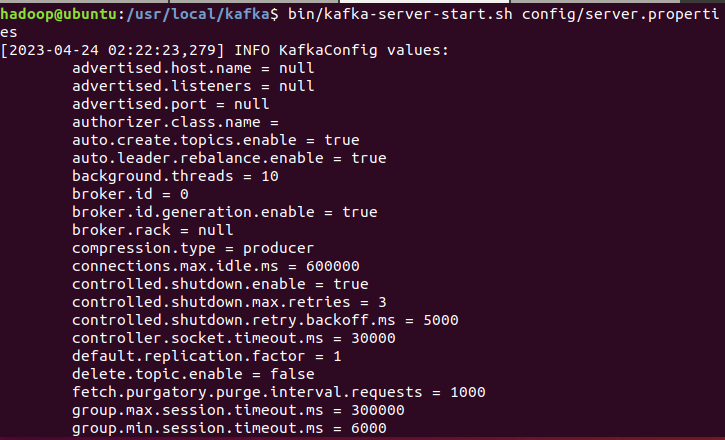
## 数据处理和python操作Kafka

2.1数据预处理

本次实验所使用的数据为data\_format.zip，使用Python对数据进行预处理，并将处理后的数据直接通过Kafka生产者发送给Kafka，所用到的是Python操作Kafka的代码库kafka\_python：

下载完成后，首先开启Kafka：



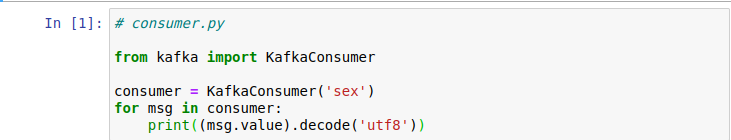


2.2Python操作Kafka

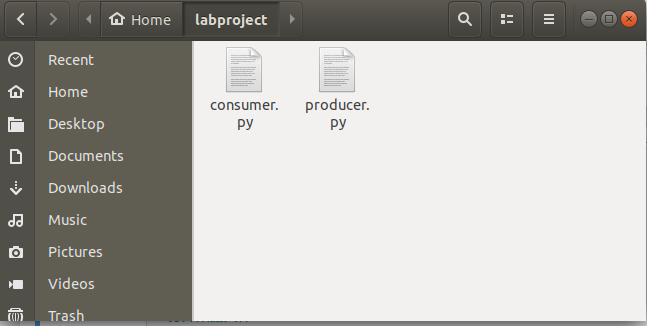
然后，写如下python代码，命名为producer.py：



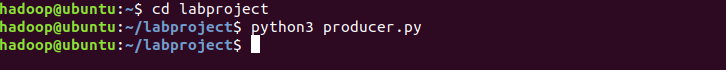
我们可以写一个KafkaConsumer测试数据是否投递成功，代码如下，文件名为consumer.py：



将代码导出为python文件，放在Home/labproject/目录中：



在terminal中，进入对应目录，输入：



新建一个窗口，输入：

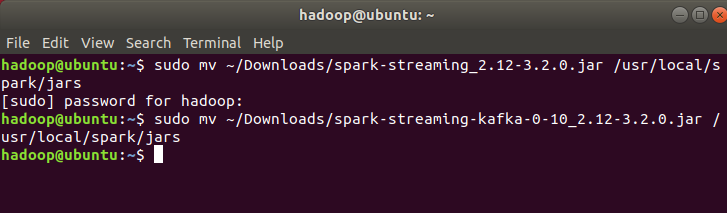


看到屏幕上会输出一行又一行的数字，则表示Python操作Kafka运行成功

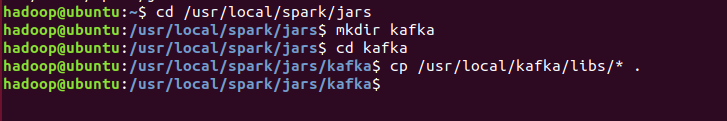
## Structured Streaming实时处理数据

3.1配置Spark开发Kafka环境

下载Spark连接Kafka的代码库。然后把下载的代码库放到目录/usr/local/spark/jars目录下：



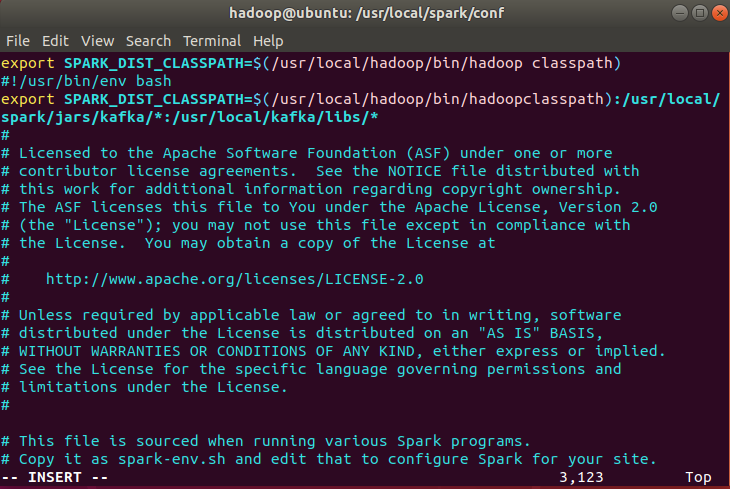
然后在/usr/local/spark/jars目录下新建kafka目录，把/usr/local/kafka/libs下所有函数库复制到/usr/local/spark/jars/kafka目录下：



然后,修改 Spark 配置文件,命令如下：



把 Kafka 相关 jar 包的路径信息增加到 spark-env.sh,修改后的 spark-env.sh 类似如下:

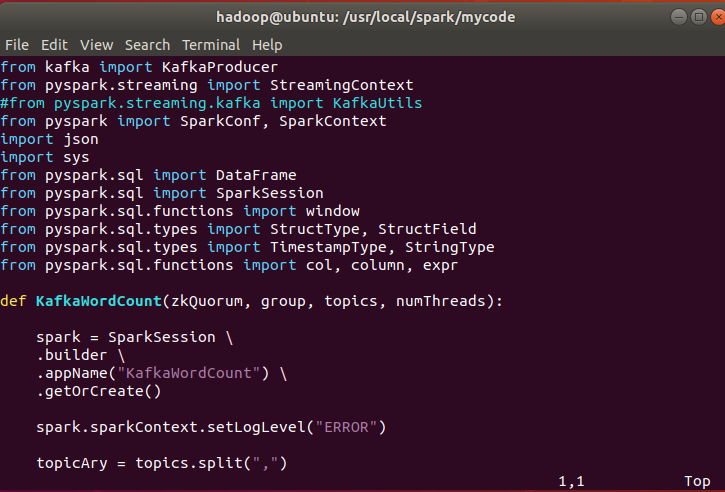


3.2建立pySpark项目

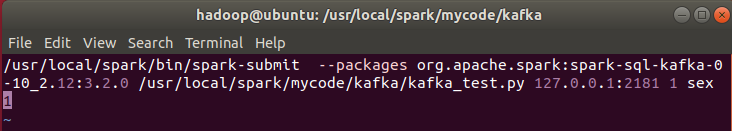
在/usr/local/spark/mycode新建项目目录：



然后在kafka这个目录下创建一个kafka\_test.py文件：



编写好程序之后，接下来编写运行脚本，在/usr/local/spark/mycode/kafka目录下新建startup.sh文件（使用vim命令），输入如下内容：

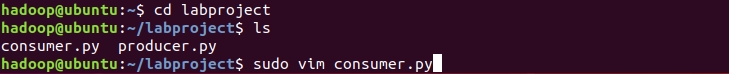


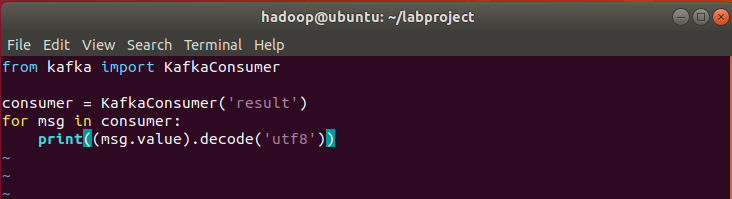
输入sh startup.sh即可执行刚编写好的Structured Streaming程序:

3.3测试程序

下面开启之前编写的KafkaProducer投递消息，然后将KafkaConsumer中接收的topic改为result，验证是否能接收topic为result的消息，更改之后的

KafkaConsumer为：



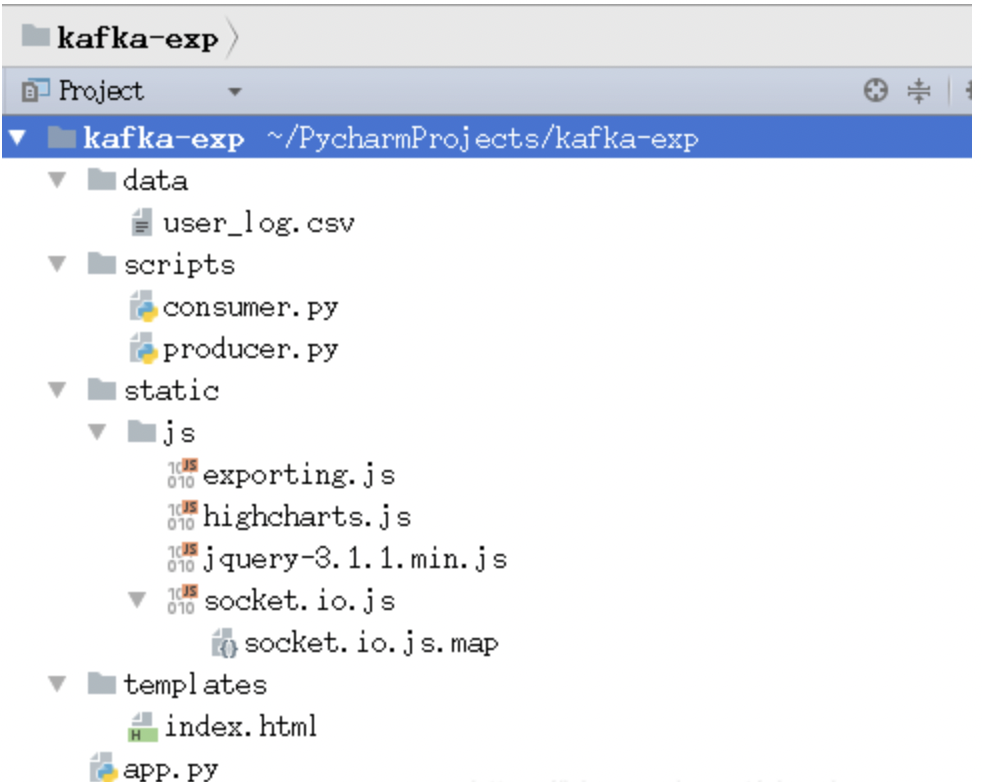


在同时开启Structured Streaming项目，KafkaProducer以及KafkaConsumer后，若可以在KafkaConsumer运行窗口看到如下输出，则表示Structured Streaming程序编写完成：



## （四）结果展示

此部分的运行在PyCharm上进行，代码结构如下：



·data目录存放的是用户日志数据；

·scripts目录存放的是Kafka生产者和消费者；

·static/js目录存放的是前端所需要的js框架；

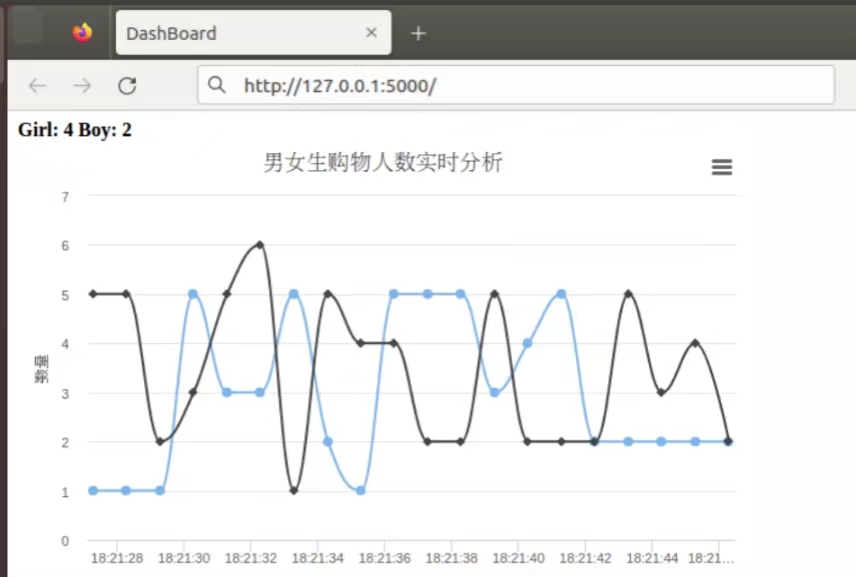
·templates目录存放的是html页面；

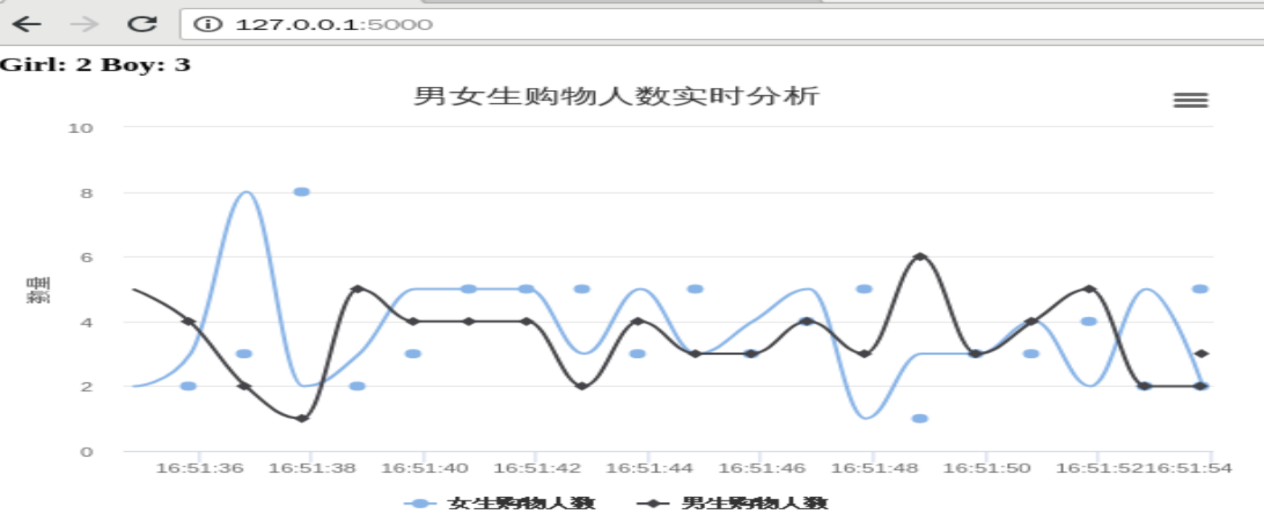
·app.py为web服务器，接收Spark Streaming处理后的结果，并推送实时数据给浏览器；

·External Libraries是本项目所依赖的Python库，是PyCharm自动生成。

·app.py的功能就是作为一个简易的服务器，处理连接请求，以及处理从kafka接收的数据，并实时推送到浏览器

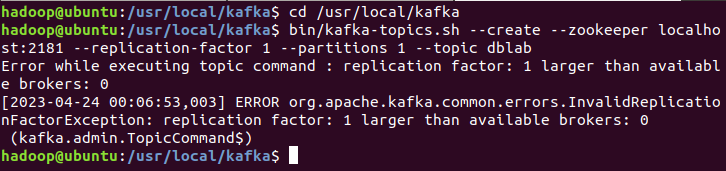
接着，按顺序开启Kafka--开启producer.py模拟数据流--启动Spark Streaming实时处理数据--启动app.py后，使用浏览器访问上图中给出的网址 http://127.0.0.1:5000/ ，就可以看到最终效果图：





# 二．易错点总结

## （一）无法用Kafka创建topic



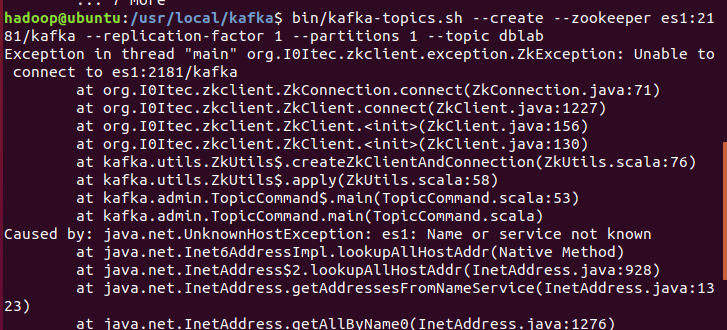
出现该报错的可能原因如下：

Kafka服务处于停止状态。

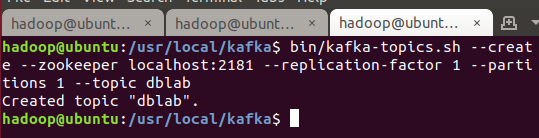
Kafka服务当前可用Broker小于设置的replication-factor。

客户端命令中Zookeeper地址参数配置错误。

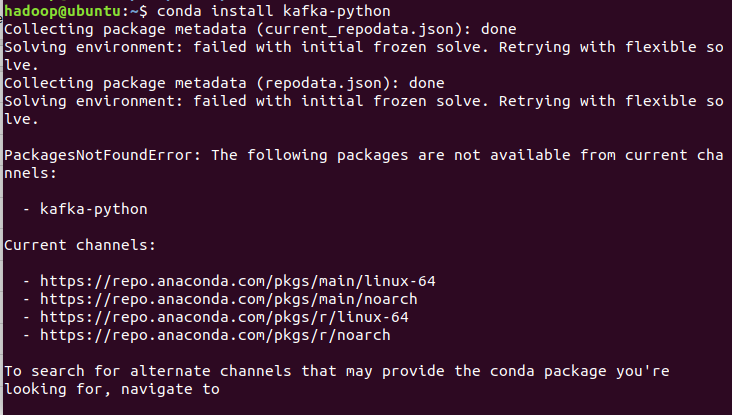
**解决方案：**该报错的原因很可能是第二点，因此我们使得replication-factor小于服务器当前可用Broker即可，修改代码如下：



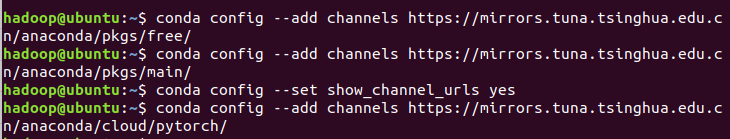
如实在找不到原因，不知道服务器Broker是被什么占用了，就重启Ubuntu吧，重启后成功运行原创建topic的代码，也成功了：



## （二）无法安装Kafka\_python



**解决方案：**更换镜像源--更新conda到最新版本--执行conda update --all：





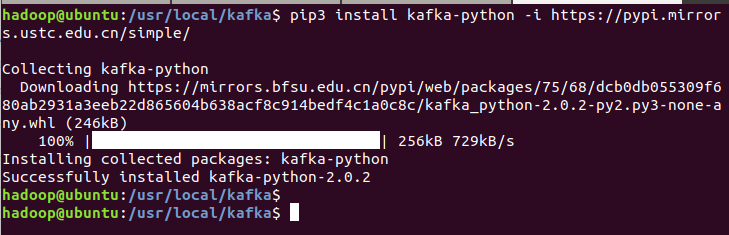


再安装Kafka\_python库，就不报错了

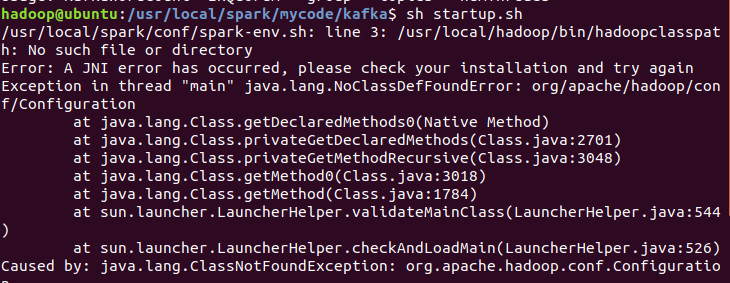
## （三）无法导入Kafka库



**解决方案：**因为笔者一开始以为Kafka\_python已经安装好了，但其实没有（于是就有了错误二），使用中科大镜像源安装库成功后，再import库就不会报错：



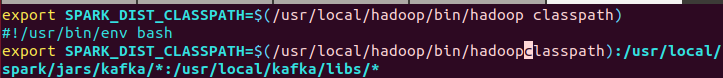
## 无法启动startup.sh



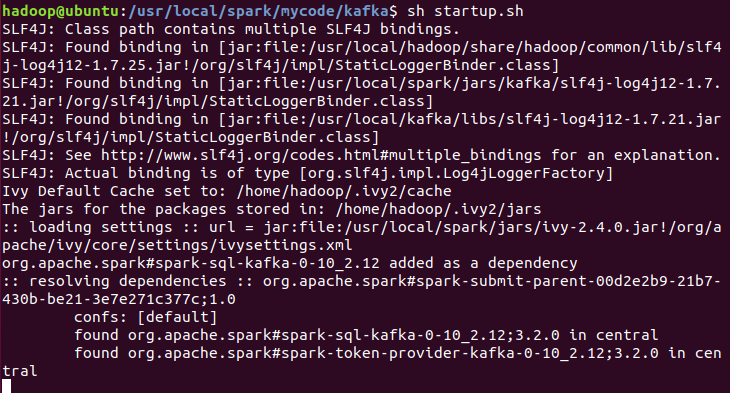
**解决方案：**这个报错可能是由java与javac版本不一致造成的，但笔者的版本是一致的，且其中也提及了一条笔者完全没有创建和使用过的路径，想来报错的原因大概率又是导入jar包时的路径有问题了：



我们回到spark-env.sh文件中，果不其然，时export的路径有问题，hadoopclasspath中间啊是有空格的：

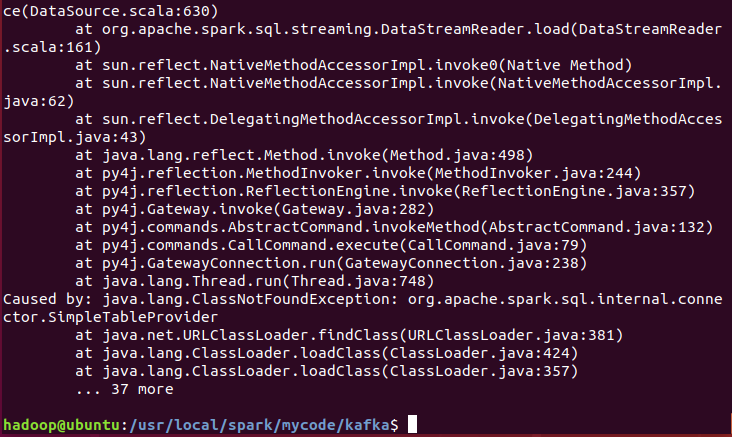


修改后，就可以正常启动startup.sh了



## ClassNotFound错误

在使用Structured Steaming处理实时数据时，出现如下报错：



**解决方案：**这个报错很麻烦，涉及到版本之间不兼容的问题，特别是需要用到的软件一多，就更加不好处理。完全按照指南对应版本来安装相关软件，还是会出现不兼容，因此笔者的处理方法是，新开一个虚拟机，安装最新版本且适配的软件。

# 三、学习小记

## （一）实验心得

这次实验的大致逻辑为：下载数据集，并向Kafka发送订单数据，Kafka通过生成窗口数据后，由Spark进行数据处理后再将窗口数据推送给Kafka，然后借由Flask,Flask-SocketIO推送、处理数据后推送给客户端进行可视化展示。

指南中的操作概览图很清晰地展示了此次实验的流程架构，因此在此进行展示：



由于实验逻辑明晰，因此操作起来也不难理解。但是在实验过程中也出现了许许多多的问题——一会儿是环境配置有问题，一会儿哪个库明明已经安装好了却无法导入......最头大的还是ClassNotFound问题导致Structured Streaming无法启动，研究了好半天都无法解决，最后还是在室友的帮助下才发现原来指南上的参考版本这么乱......许多软件之间根本无法适配，因此，又从头开始，重开虚拟机去装适配软件，来来回回搞了好几天。只能说，自己对流计算还只是一知半解，停留在知道个大概的流程，但对数据处理等过程的细节还是不够熟悉，导致这次试验过程中的代码有些地方对我来说是比较难懂的。同时，粗枝大叶也是一个毛病，让我在执行实验操作的过程中没太在意一些重要代码的含义或是版本之间的匹配性，因此在这次较为综合的实验中便四处碰壁，且出现的报错往往需要调试很长一段时间。

当下时代，对于大多数企业而言，建立数据仓库系统，存储静态数据用于支持决策分析已经成为必须，但与此同时，实时获取来自不同数据源的海量数据，经过实时分析处理，获得有价值的信息的方式，也逐步兴起。

所以，基于流计算的实时数据分析越来越被重视与被需要，因为需要即时追踪用户的需求、进行迅速的响应，就需要对数据进行动态分析与处理，如关注某种行为的增长变化趋势，以及打车软件即时更新车辆位置信息等，都离不开数据的实时刷新。因此，通过这次试验，掌握一定的实时分析技术，是具有深刻的现实意义的。

## （二）知识点笔记

2.1流数据与流计算

流数据：即数据以大量、快速、时变的流形式持续到达，兴起于Web应用、网络监控、传感监测等领域。数据具有如下特点：

·具有数据快速持续到达，潜在大小也许是无穷无尽的

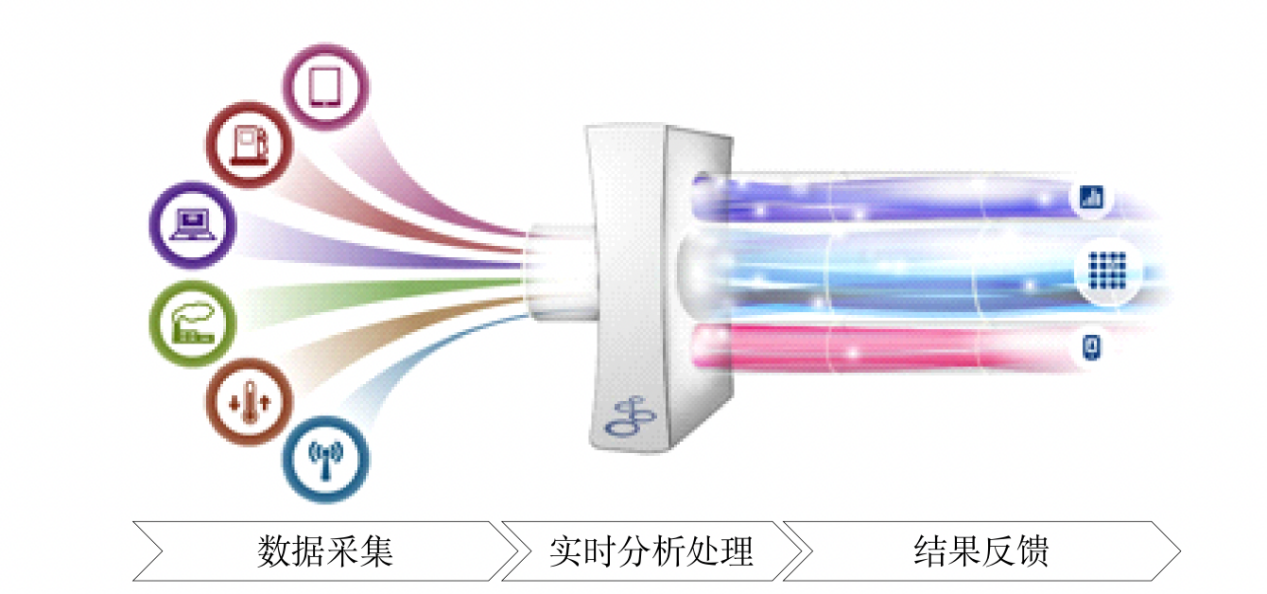
·数据来源众多，格式复杂

·数据量大，但是不十分关注存储，一旦经过处理，要么被丢弃，要么被归档存储

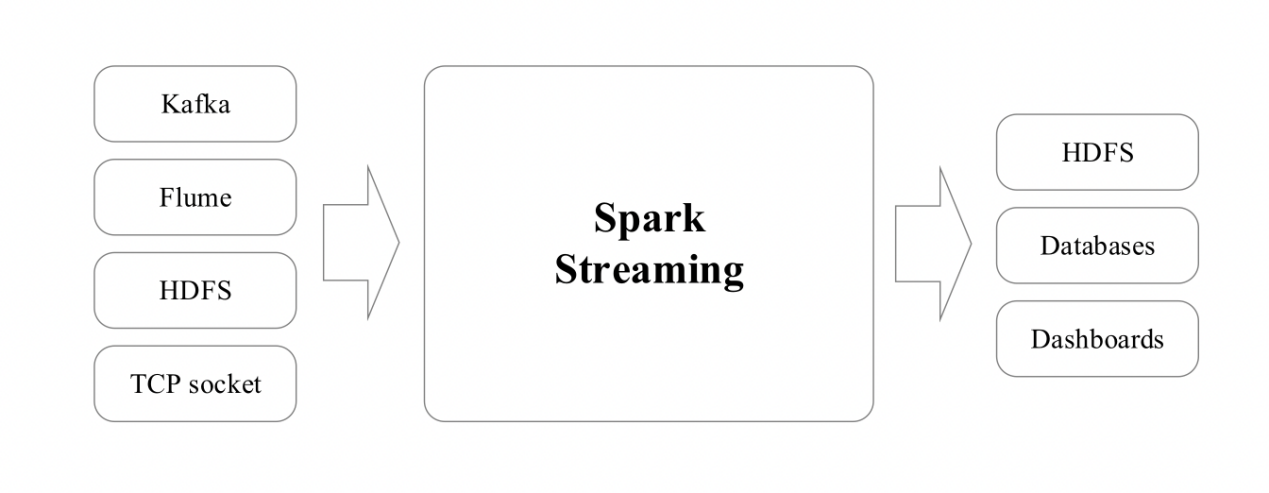
·注重数据的整体价值，不过分关注个别数据

·数据顺序颠倒，或者不完整，系统无法控制将要处理的新到达的数据元素的顺序

流计算：实时获取来自不同数据源的海量数据，经过实时分析处理，获得有价值的信息。



2.2 Spark Streaming

Spark Streaming可整合多种输入数据源，如Kafka、Flume、HDFS，甚至是普通的TCP套接字。经处理后的数据可存储至文件系统、数据库，或显示在仪表盘里

其基本原理是将实时输入数据流以时间片（秒级）为单位进行拆分，然后经Spark引擎以类似批处理的方式处理每个时间片数据：

