**

**HUNAN UNIVERSITY**

《数据仓库技术与应用》

实验报告六

|  |  |
| --- | --- |
| **报告名称：** | 2020美国新冠肺炎疫情数据分析 |
| **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：** | 李信儒 |
| **日 期：** | 2023.3.26 |

目录

[一、实验过程 3](#_Toc1288756177)

[（一）实验环境配置 3](#_Toc605003197)

[1.1安装Spark2.4.0 3](#_Toc2101147081)

[1.2安装Spark(Local模式） 5](#_Toc757899099)

[1.3使用Spark Shell编写代码 7](#_Toc1284646536)

[1.4简单RDD操作 8](#_Toc253743614)

[2.1安装Jupyter Notebook 9](#_Toc1913881203)

[2.2配置Jupyter Notebook 14](#_Toc1591314055)

[2.3运行Jupyter Notebook 15](#_Toc453982647)

[2.4实现Jupyter Notebook与PySpark的交互 16](#_Toc76950338)

[（二） 数据集操作 20](#_Toc519175272)

[2.1数据集下载 20](#_Toc552738743)

[2.2格式转换 22](#_Toc2013280326)

[（三） 使用Spark对数据进行分析 23](#_Toc1450096950)

[3.1完整代码 23](#_Toc2135012494)

[3.2读取文件生成DataFrame 24](#_Toc850728935)

[3.3数据分析 24](#_Toc255088819)

[3.4生成结果文件 25](#_Toc900421521)

[（四） 数据可视化 25](#_Toc67243038)

[4.1可视化工具选择与代码 25](#_Toc577341344)

[4.2结果展示 26](#_Toc1044851462)

[二、易错点总结 32](#_Toc844740315)

[（一）无法上传文件至HDFS 32](#_Toc536083888)

[（二）无法保存结果文件 33](#_Toc1268006451)

[（三） 无法安装pyecharts 33](#_Toc1904192776)

[（四） 无法保存可视化结果 34](#_Toc1966678638)

[三、学习小记 35](#_Toc2047057889)

[（1）认识Spark 35](#_Toc66431836)

[（2）Spark与Hadoop执行流程对比 35](#_Toc1975854859)

[（3） Spark运行架构 36](#_Toc1652981652)

[（4） RDD运行原理 37](#_Toc1814167572)

[（5） RDD在Spark中的运行过程 38](#_Toc741562498)

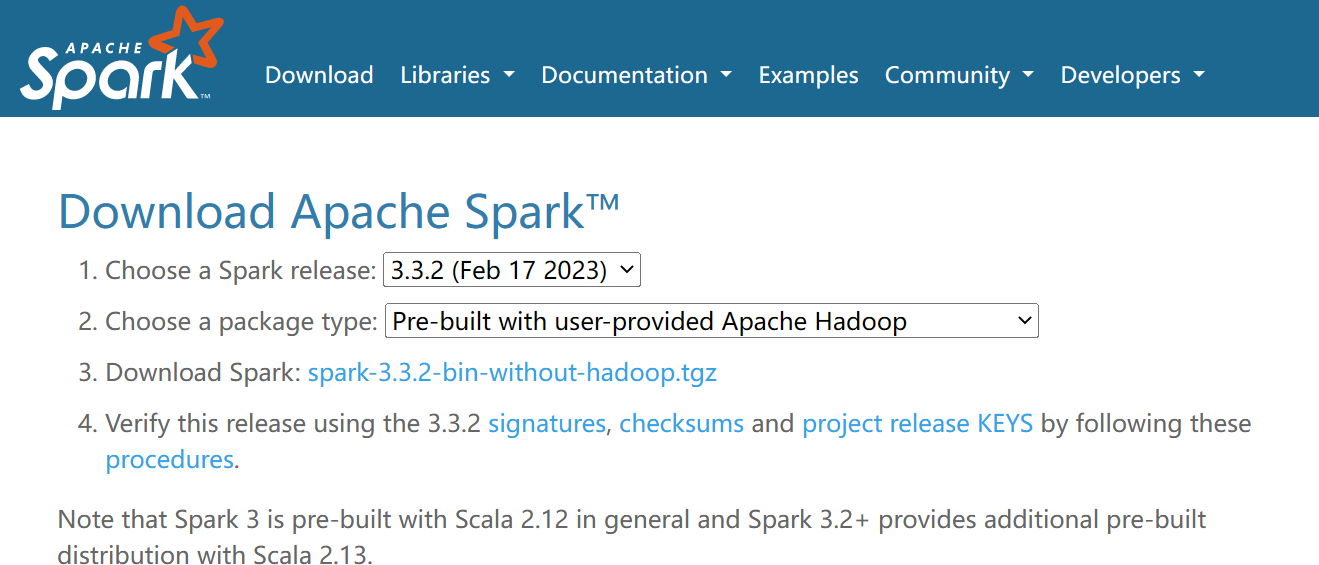
# 一、实验过程

## （一）实验环境配置

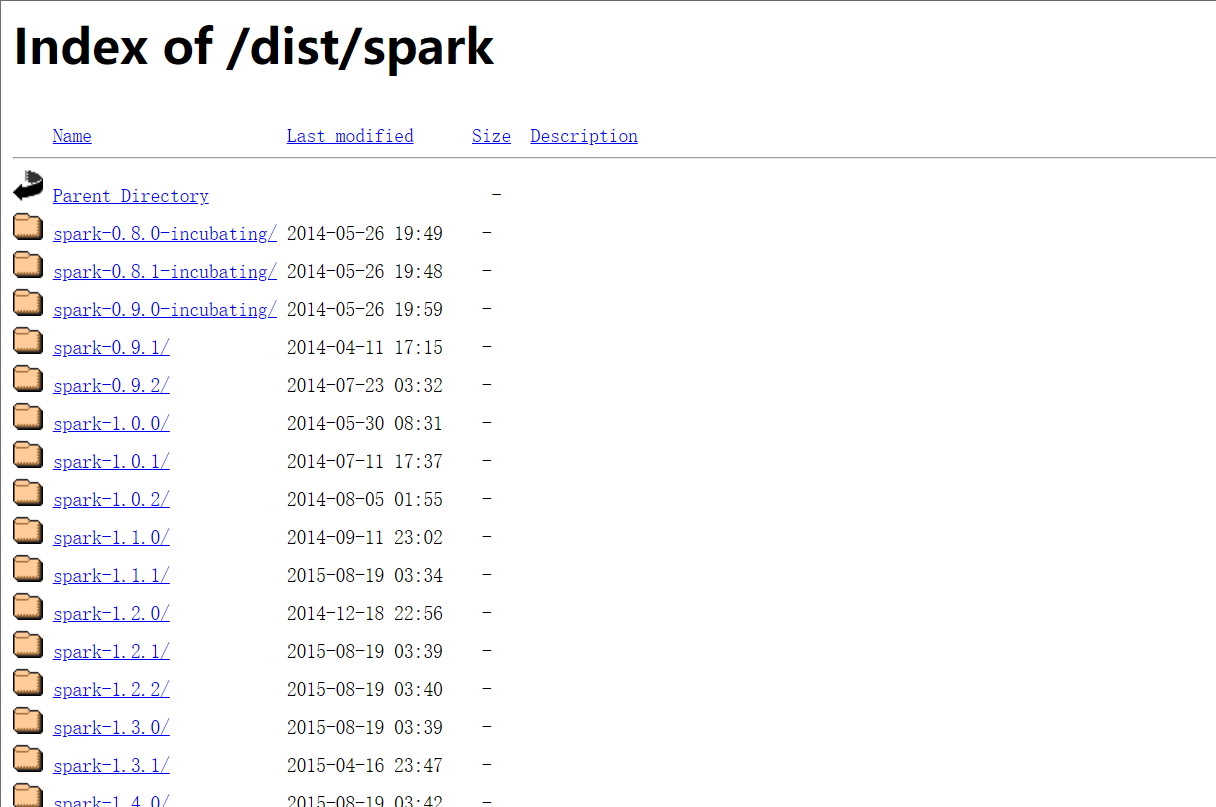
本次实验环境：Linux：Ubuntu 16.04；Hadoop3.1.3；Python：3.6；Spark：2.4.0；Jupyter Notebook

1.1安装Spark2.4.0

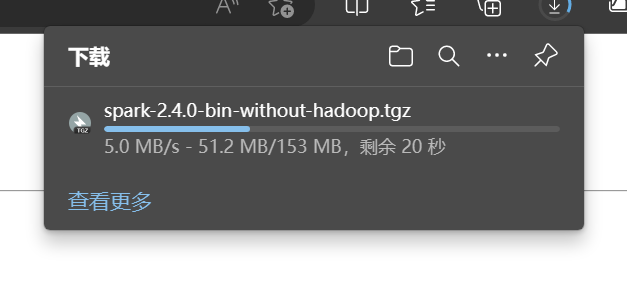
·下载Spark2.4.0安装文件：由于Spark官网只能找到最新版本Spark3.3.2，故笔者选择去自行搜索Spark2.4.0下载路径，以防版本不一致造成后续实验出现问题

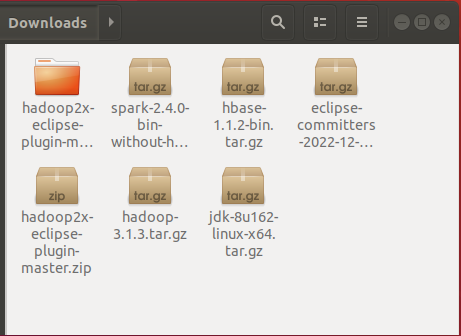


Spark各版本下载地址：http://archive.apache.org/dist/spark

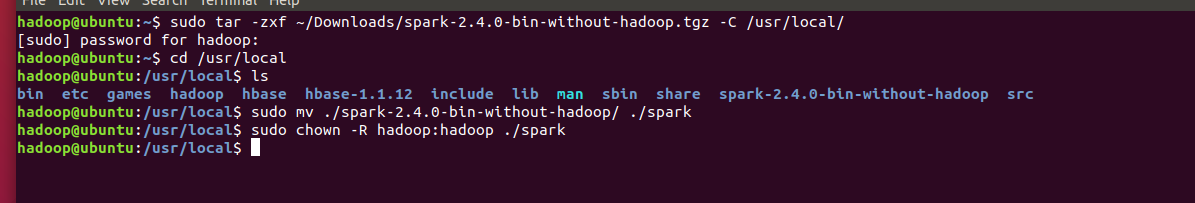




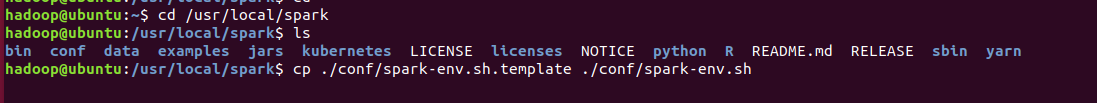




1.2安装Spark(Local模式）

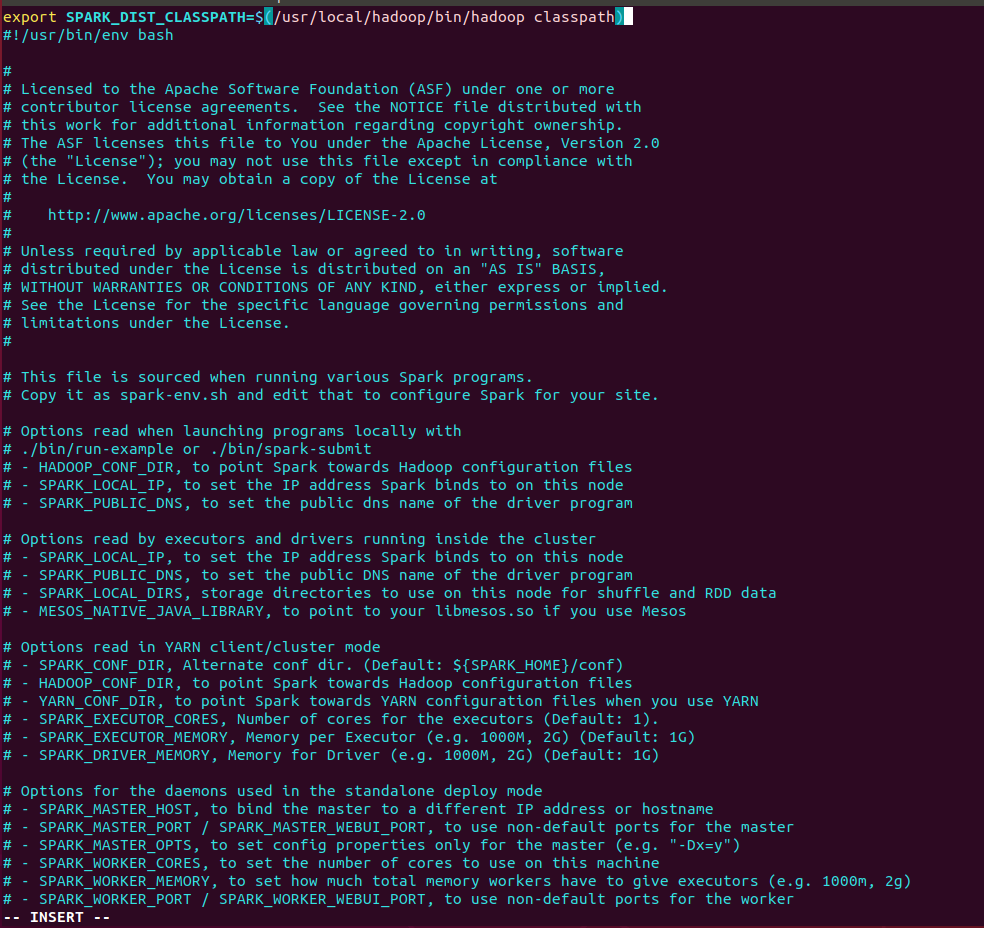


安装后，还需要修改Spark的配置文件spark-env.sh



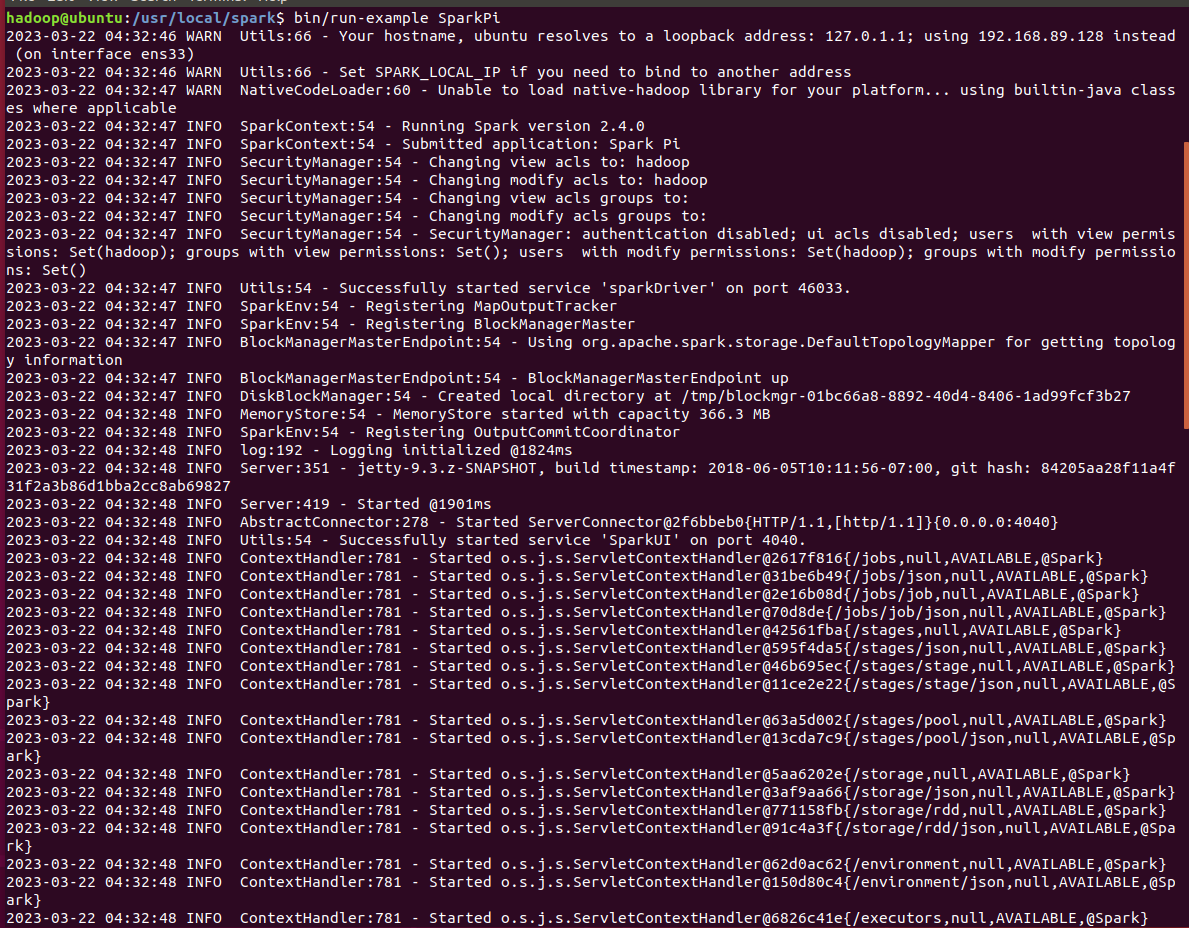
编辑spark-env.sh，(vim ./conf/spark-env.sh)在第一行添加如下配置信息





此时已经配置完成，可以通过运行Spark自带的示例，验证Spark是否安装成功





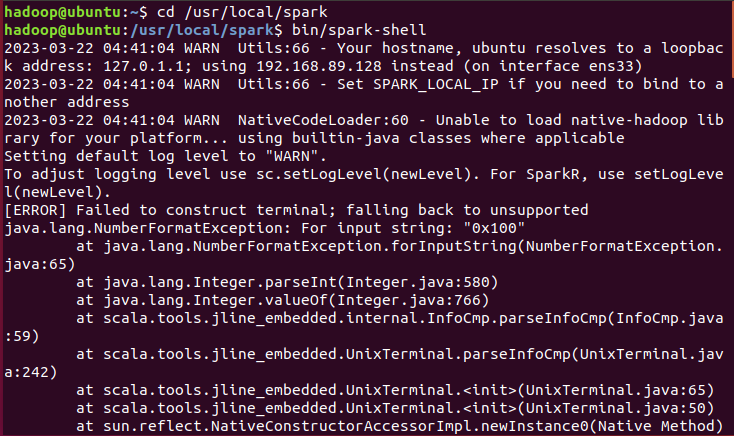
执行时会输出非常多的运行信息，输出结果不容易找到，可以通过 grep 命令进行过滤（命令中的 2>&1 可以将所有的信息都输出到 stdout 中，否则由于输出日志的性质，还是会输出到屏幕中）

运行结果如下，显示的是Π的近似值



### 1.3使用Spark Shell编写代码

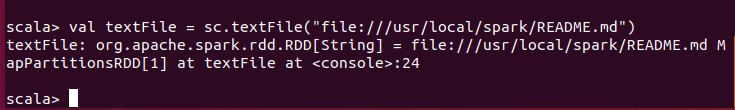
·启动Spark Shell





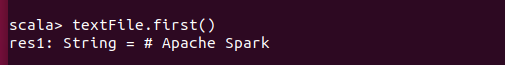
·加载text文件

Spark创建sc，可以加载本地文件和HDFS创建的RDD，这里用Spark自带的本地文件README.md文件测试



1.4简单RDD操作

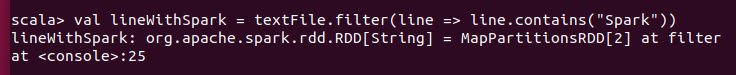
·textFile.first()：获取文件的第一行内容



·textFile.count()：获取文件所有项的计数



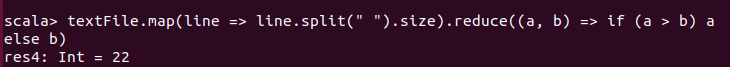
·val lineWithSpark = textFile.filter(line => line.contains(“Spark”)):抽取含“Spark”的行，返回一个新的RDD



·lineWithSpark.count()：统计生成新的RDD的行数



·通过组合RDD操作实现简易MapReduce操作



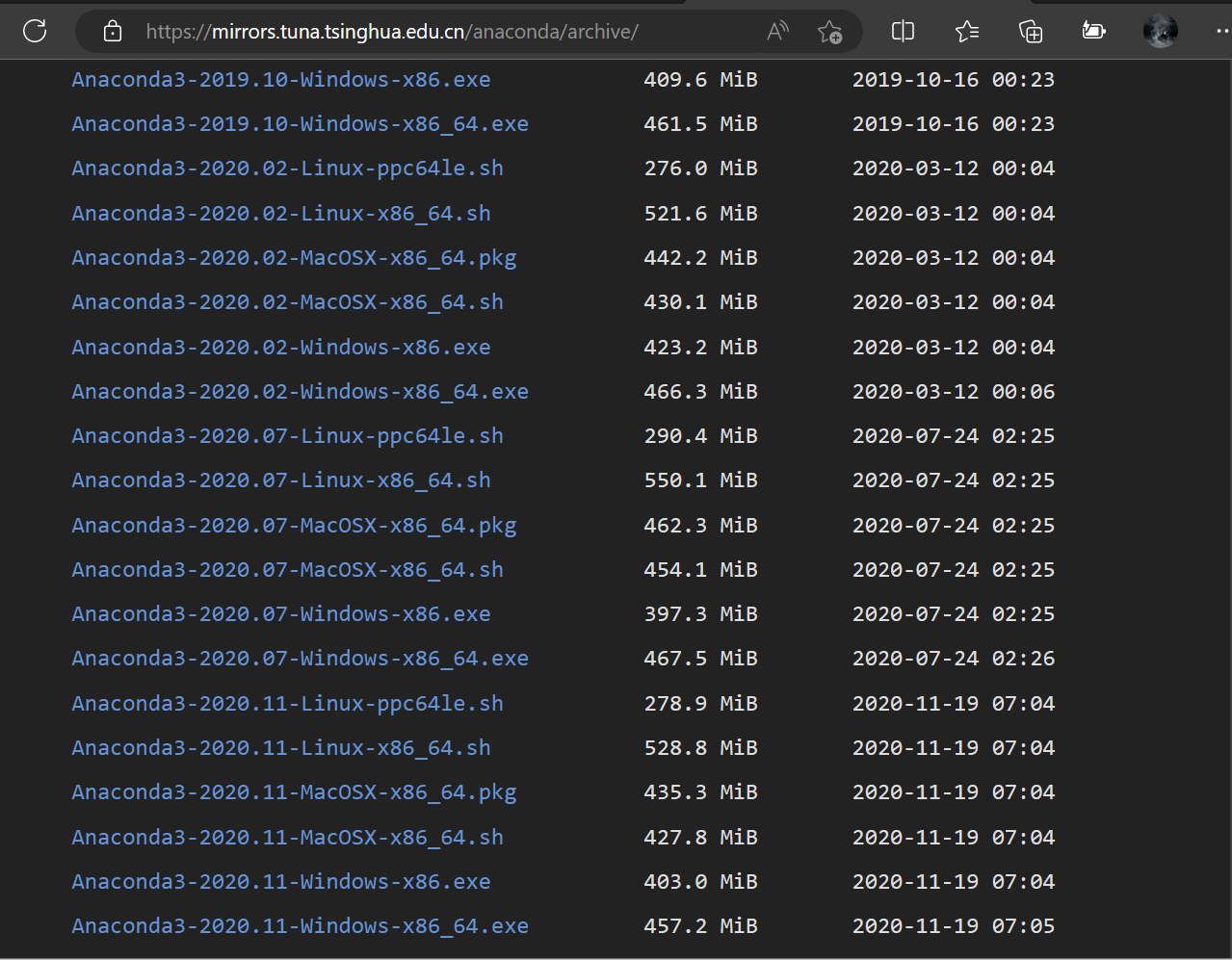
·退出Spark Shell

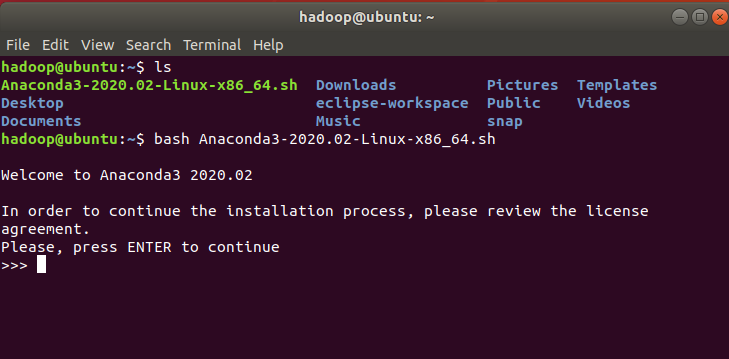


2.1安装Jupyter Notebook

·安装Anaconda

笔者选择在清华大学开源软件镜像站中下载Anaconda（更方便地找到实验对应的版本），找到找到安装文件Anaconda3-2020.02-Linux-x86\_64.sh，下载到本地即可

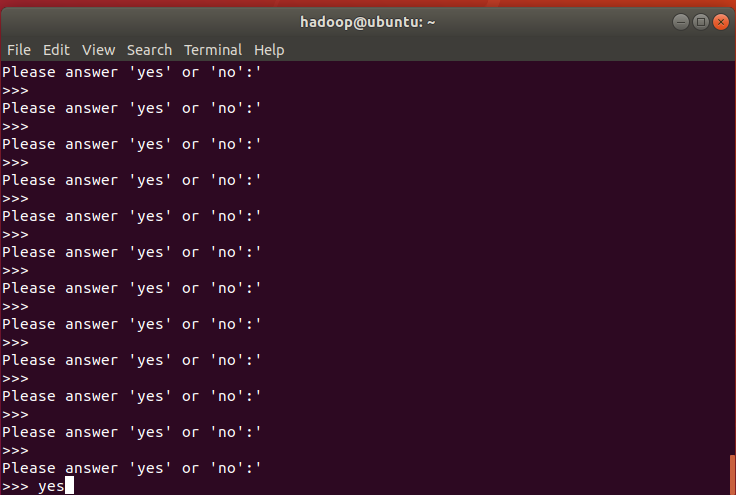




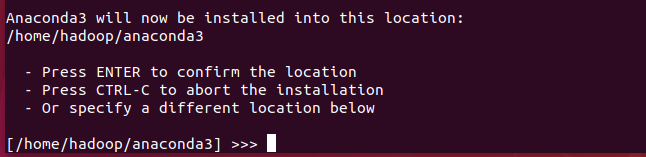
出现许可文件，很长，可以一直按回车到文件的末尾



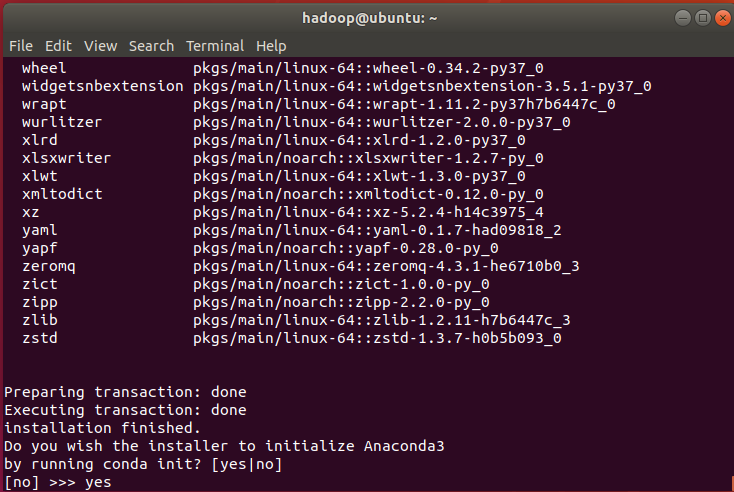
来到末尾以后会出现提示“是否接受条款”，输入yes即可(下图是回车没刹住车)



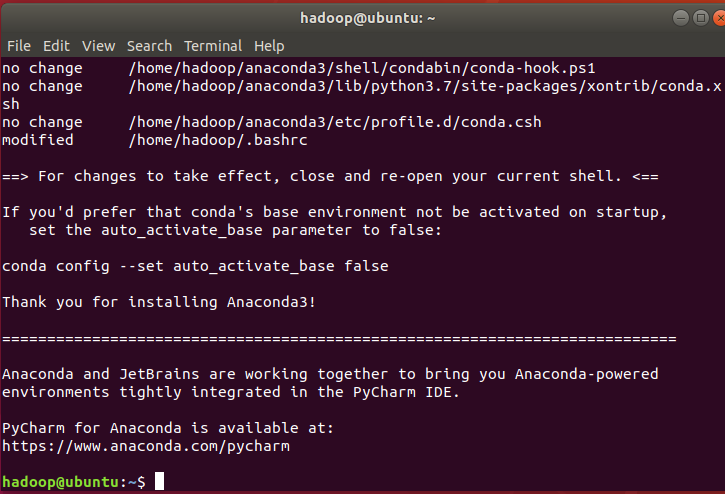
默认安装路径如下



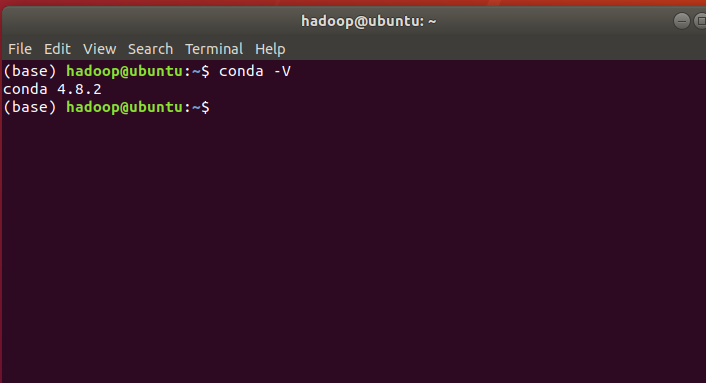
配置初始环境变量，输入yes



安装成功界面如下



查看版本信息



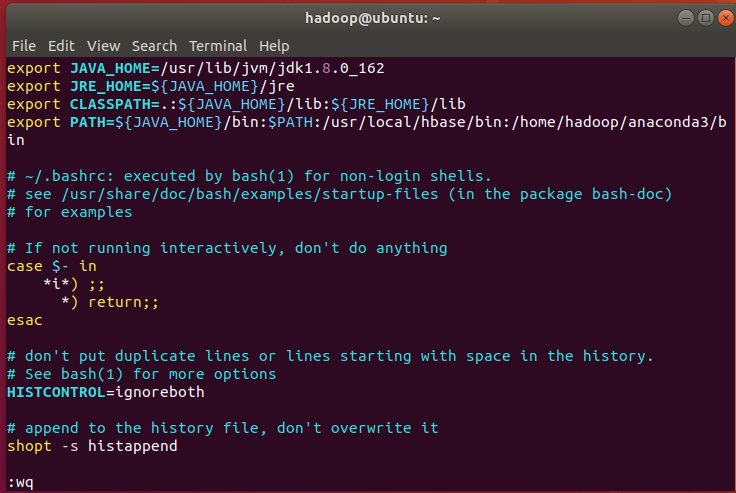
消除terminal前的（base）



执行完这行代码后需要更改anaconda配置



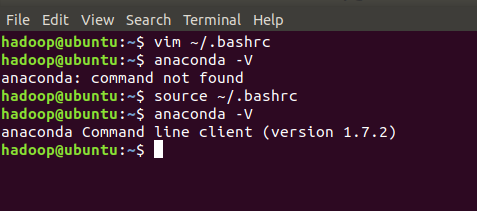
把/home/hadoop/anaconda3/bin加入path的末尾



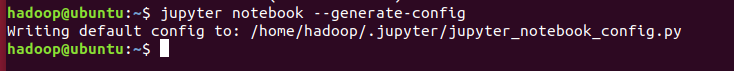
使配置生效

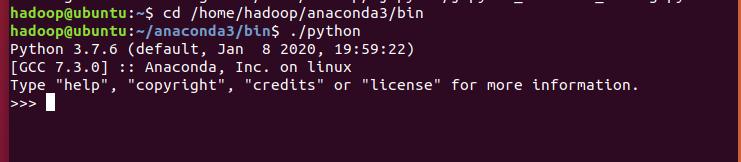


然后就可以用anaconda -V命令了

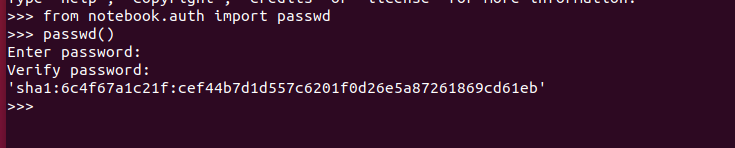


2.2配置Jupyter Notebook

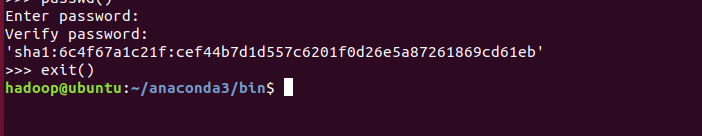




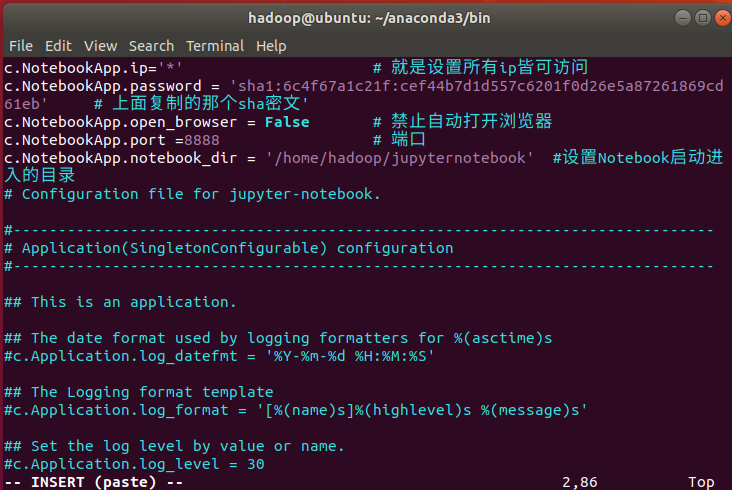
把这个sha1字符串复制粘贴到一个文件中保存起来，后面用于配置密码



'sha1:6c4f67a1c21f:cef44b7d1d557c6201f0d26e5a87261869cd61eb'

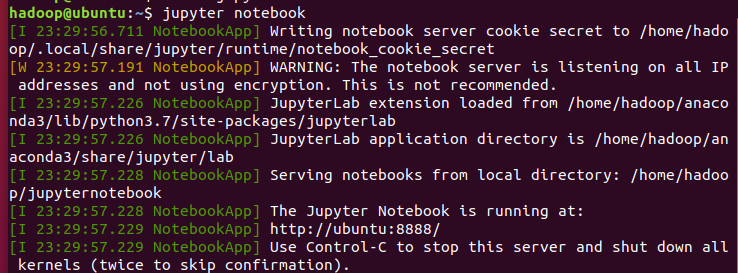


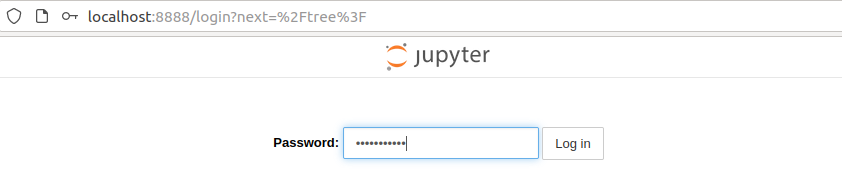


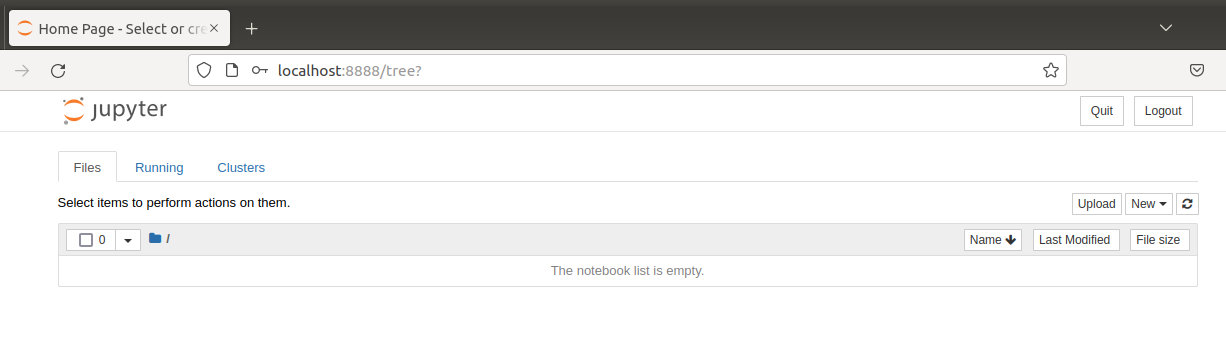




2.3运行Jupyter Notebook

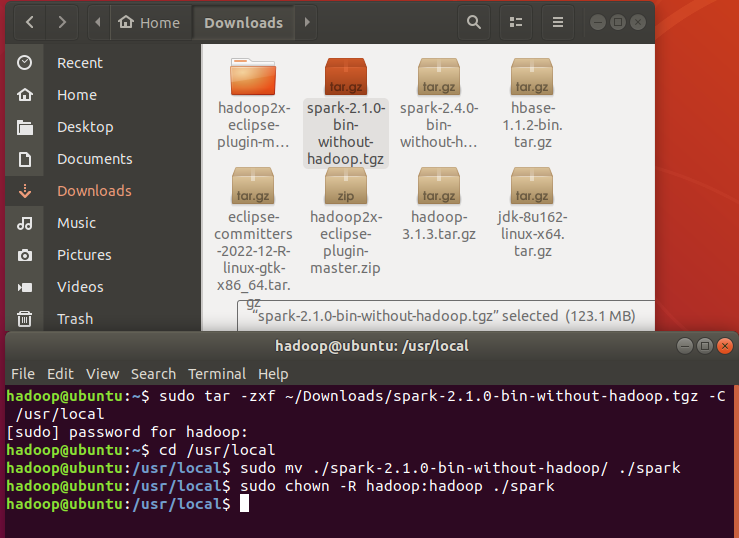


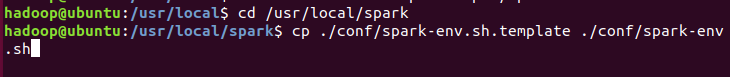




2.4实现Jupyter Notebook与PySpark的交互

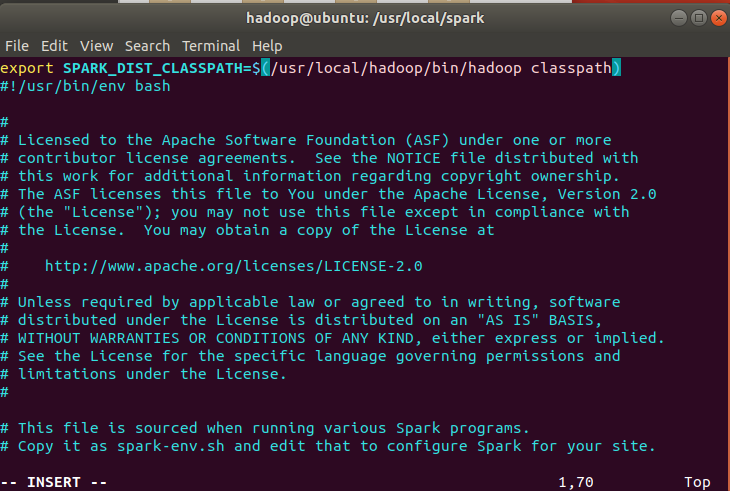
·安装python版的Spark







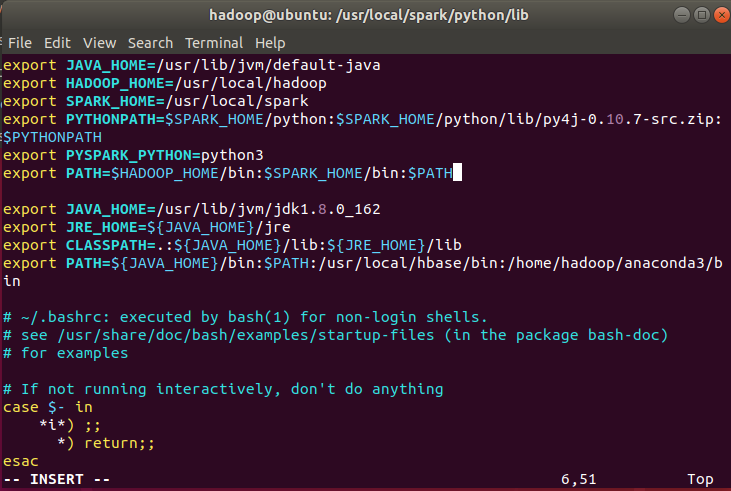
在第一行添加如下配置信息



修改环境变量，注意py4j的zip文件版本的对应性

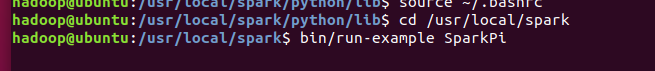


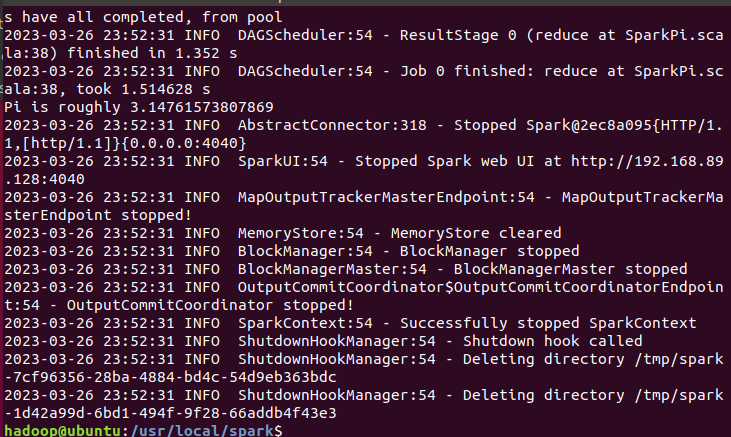






运行自带示例验证是否安装成功

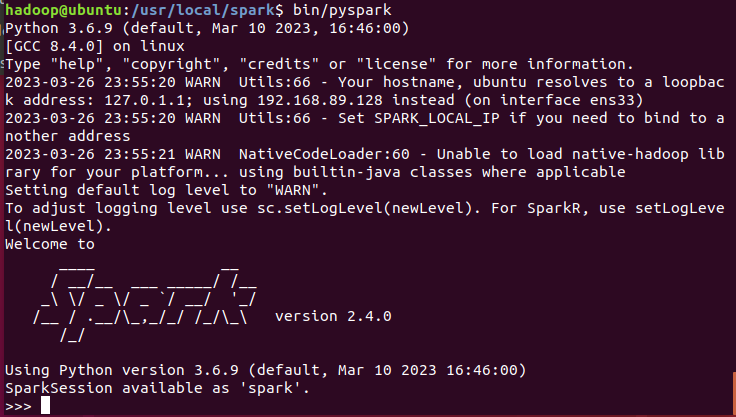




使用grep过滤输出结果



启动pyspark界面如下（>>>是python的标识符）





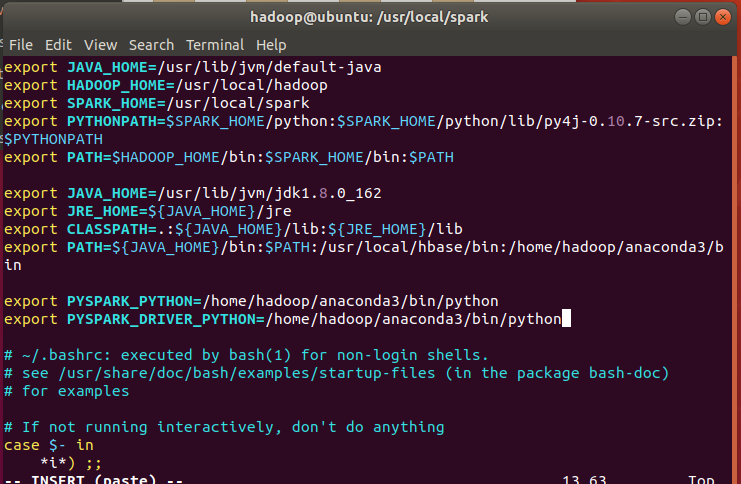
使用exit（）退出pyspark



·实现两者交互

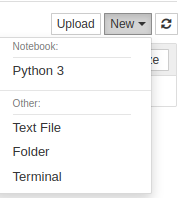


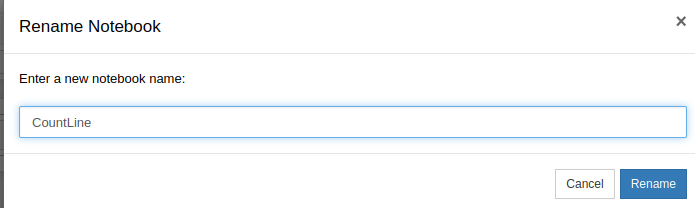
在.bashrc文件中把原来已经存在的一行“export PYSPARK\_PYTHON=python3”删除，然后，在该文件中增加末尾的两行export

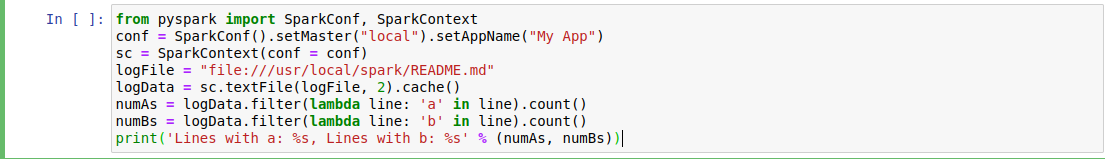




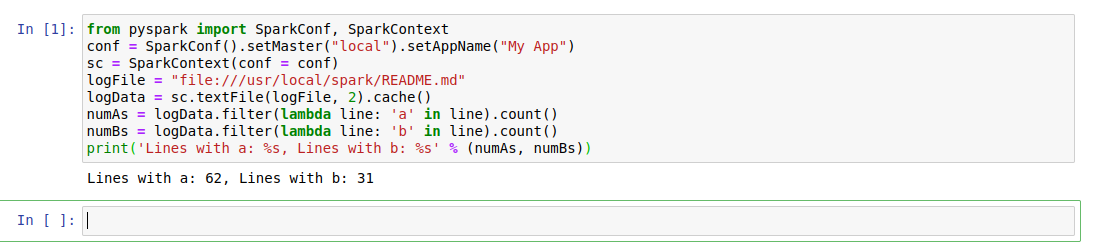
运行Jupyter Notebook 新建文件CountLine,运行如下内容



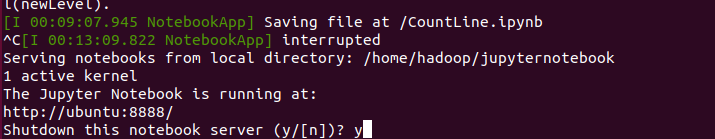




结果如下（注：出现结果后不能再次点运行，要再次运行需要点击刷新，在弹出的界面中点击restart）



如果要关闭退出Jupyter Notebook，可以回到终端界面（正在运行Jupyter Notebook的界面），按Ctrl+C，出现提示，输入字母y，就可以退出了（如下图所示）

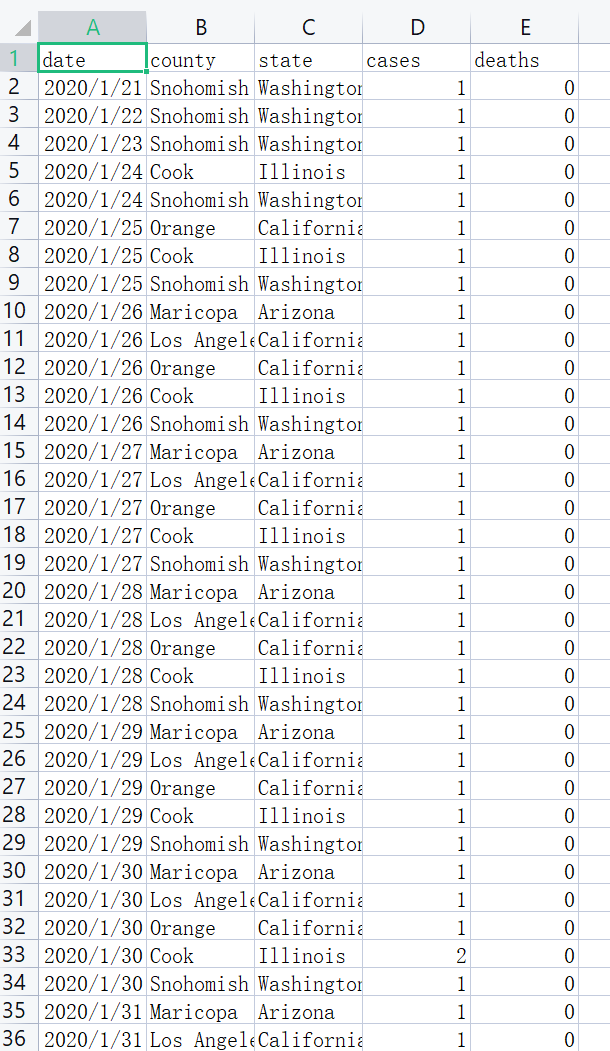


## 数据集操作

2.1数据集下载

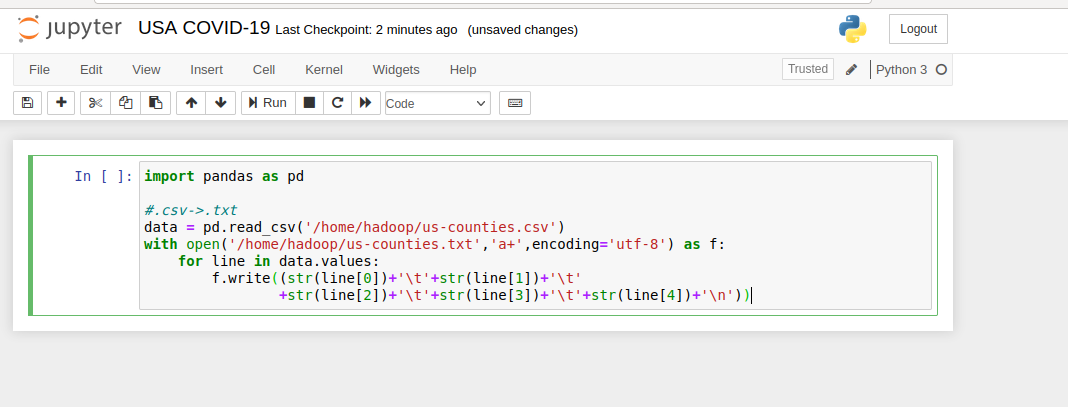
数据集来自Kaggle的美国新冠肺炎数据集



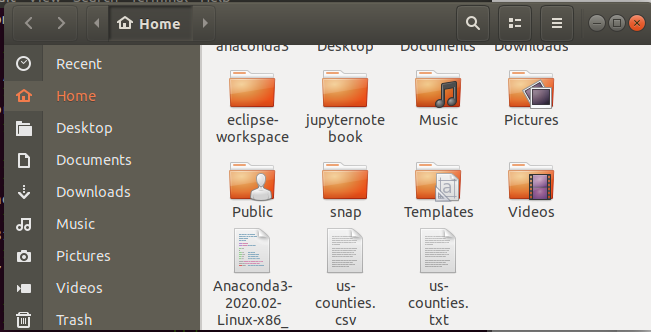


2.2格式转换

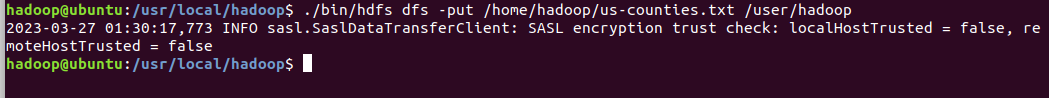
原数据集以.csv文件组织，目的是方便Spark读取生成RDD或DataFrame，因此需要将其转换成txt文件以便于python实现



运行完上述格式转换代码后，进入目录home/hadoop后发现出现了对应的txt文件，之后将会用该文件进行数据分析

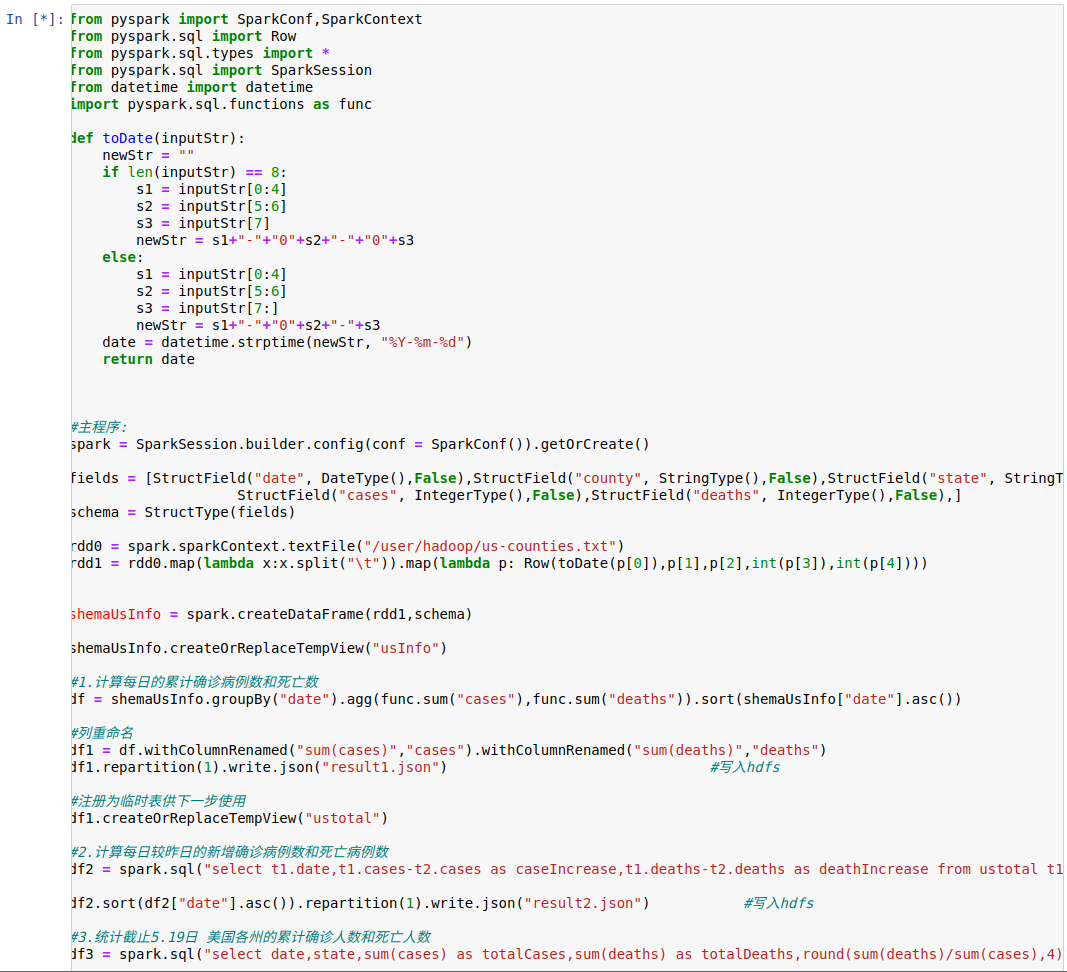


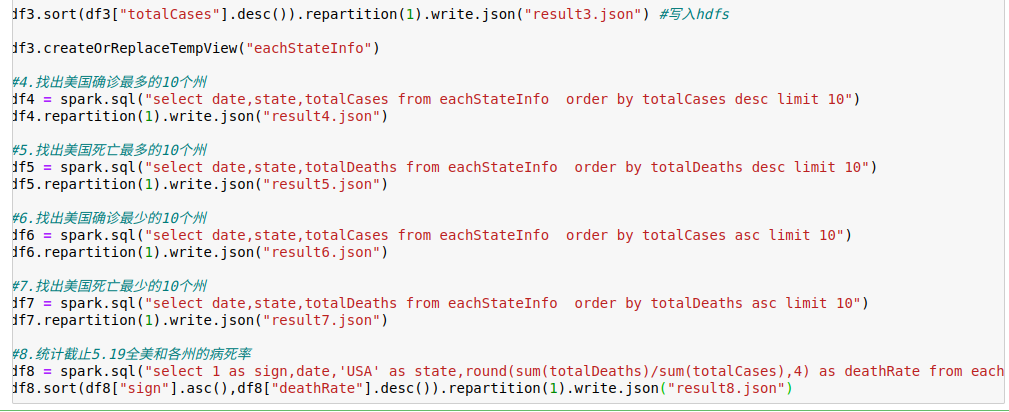
2.3将文件上传至HDFS文件系统（hadoop要保证已经打开，不然会报错）



## 使用Spark对数据进行分析

3.1完整代码





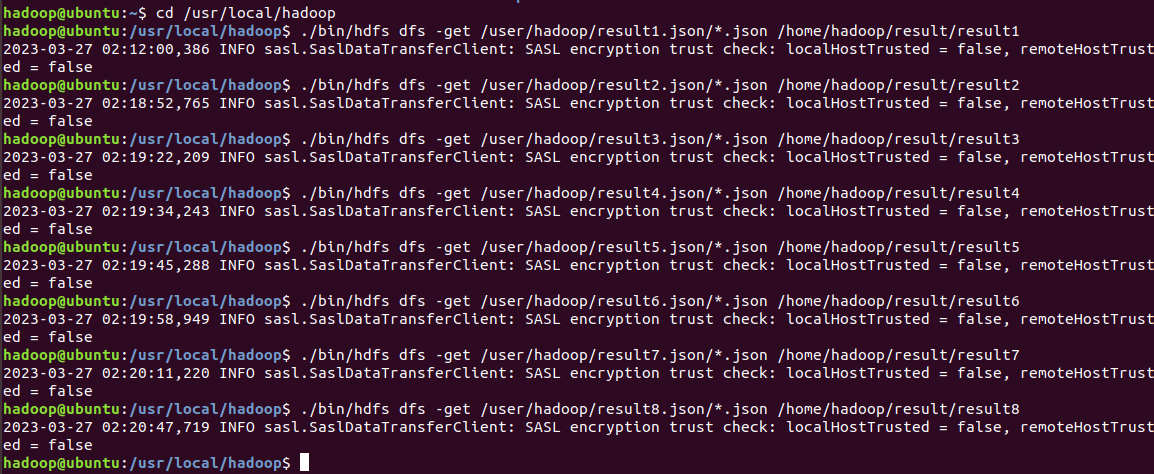
3.2读取文件生成DataFrame



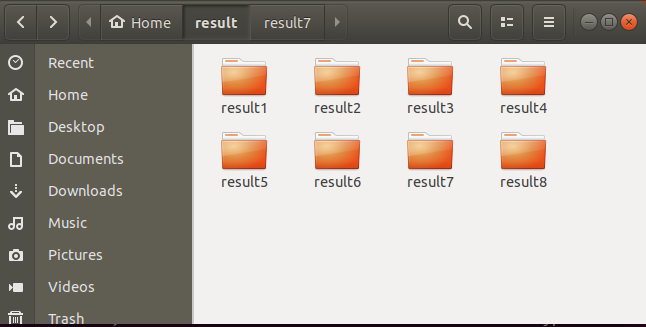
3.3数据分析

本实验主要统计以下8个指标，分别是：  
1) 统计美国截止每日的累计确诊人数和累计死亡人数。做法是以date作为分组字段，对cases和deaths字段进行汇总统计。  
2) 统计美国每日的新增确诊人数和新增死亡人数。因为新增数=今日数-昨日数，所以考虑使用自连接，连接条件是t1.date = t2.date + 1，然后使用t1.totalCases – t2.totalCases计算该日新增。  
3) 统计截止5.19日，美国各州的累计确诊人数和死亡人数。首先筛选出5.19日的数据，然后以state作为分组字段，对cases和deaths字段进行汇总统计。  
4) 统计截止5.19日，美国确诊人数最多的十个州。对3)的结果DataFrame注册临时表，然后按确诊人数降序排列，并取前10个州。  
5) 统计截止5.19日，美国死亡人数最多的十个州。对3)的结果DataFrame注册临时表，然后按死亡人数降序排列，并取前10个州。  
6) 统计截止5.19日，美国确诊人数最少的十个州。对3)的结果DataFrame注册临时表，然后按确诊人数升序排列，并取前10个州。  
7) 统计截止5.19日，美国死亡人数最少的十个州。对3)的结果DataFrame注册临时表，然后按死亡人数升序排列，并取前10个州  
8) 统计截止5.19日，全美和各州的病死率。病死率 = 死亡数/确诊数，对3)的结果DataFrame注册临时表，然后按公式计算。  
在计算以上几个指标过程中，根据实现的简易程度，既采用了DataFrame自带的操作函数，又采用了spark sql进行操作。

3.4生成结果文件



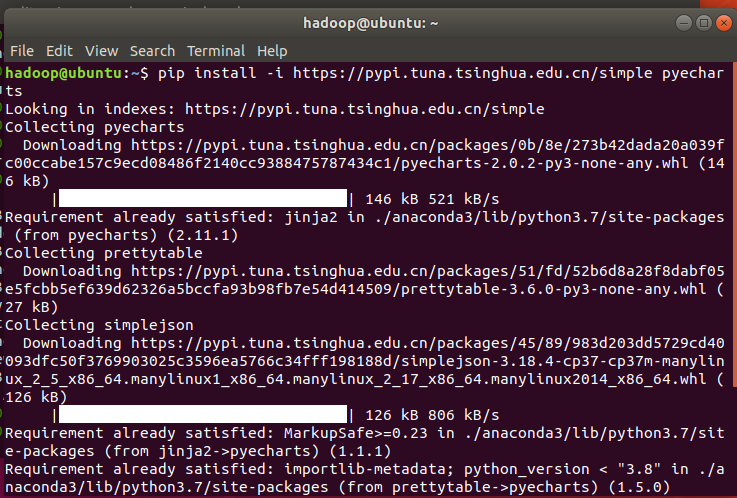
对于result2等文件，代码同理，改变一下路径即可

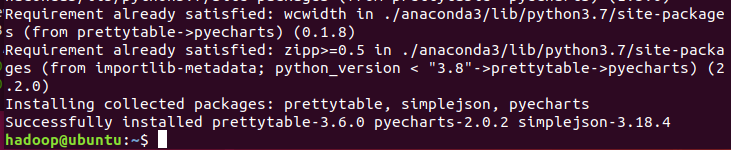


## 数据可视化

4.1可视化工具选择与代码

笔者进行了换源，直接使用pip install pyecharts会因为网关等问题难以成功





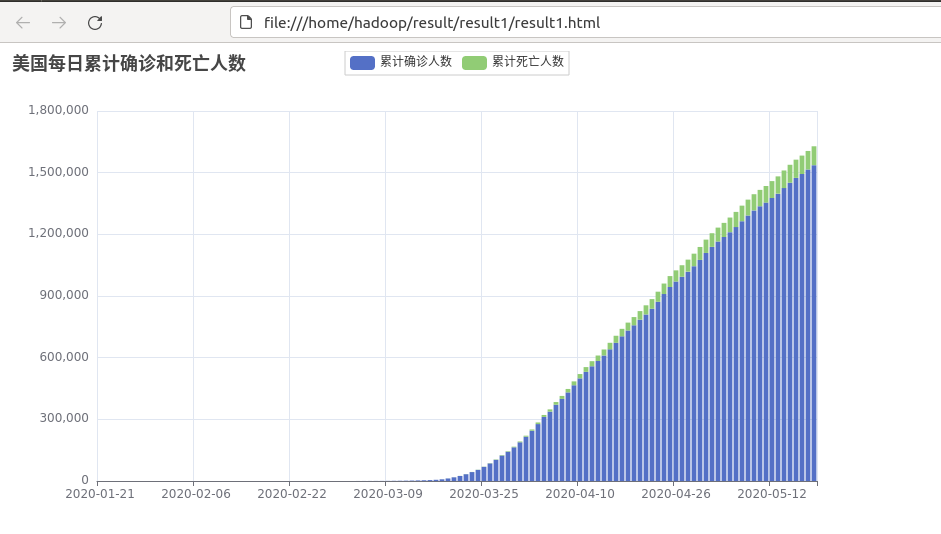
4.2结果展示

（部分代码如下）

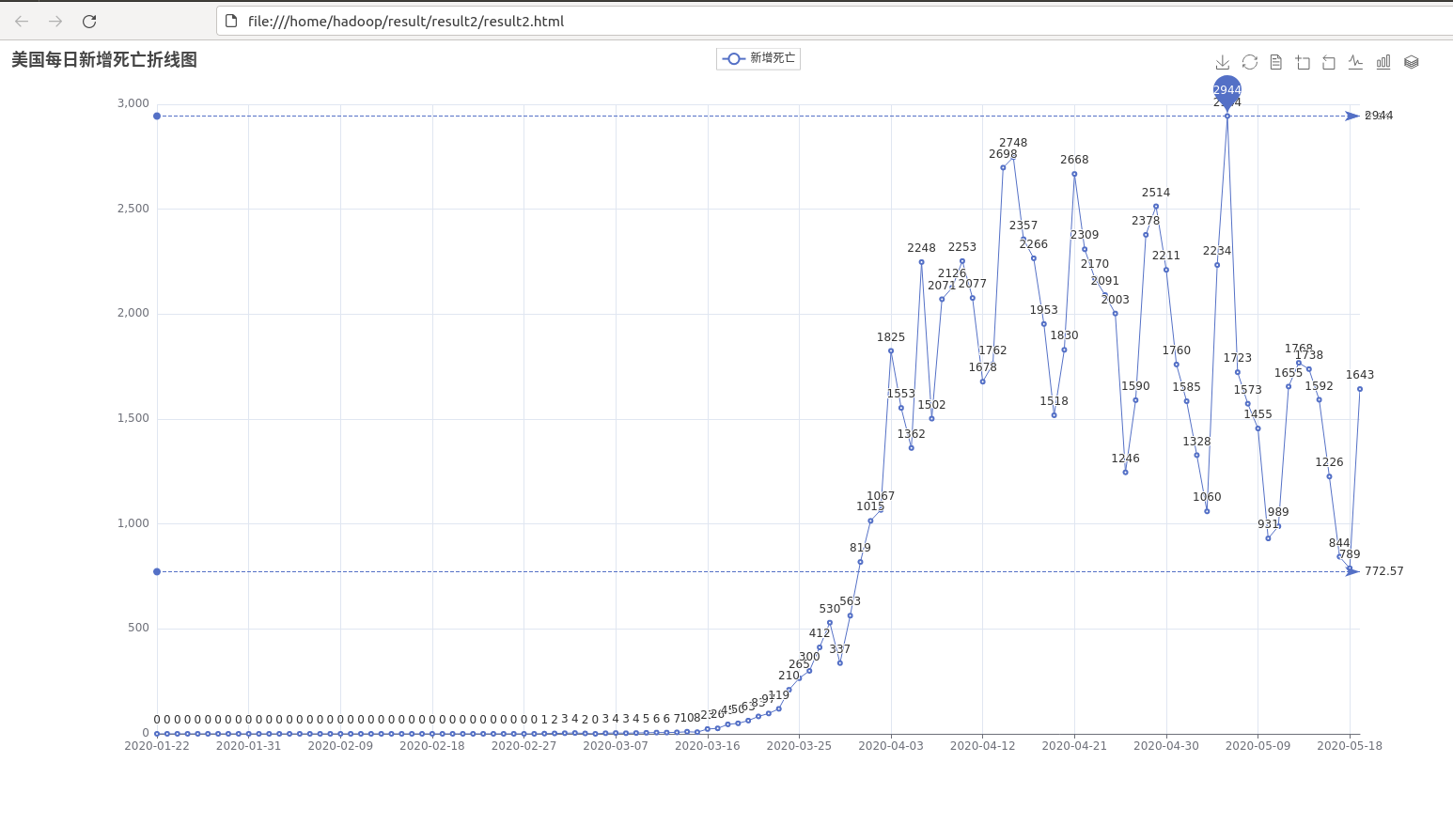


可视化数据如下

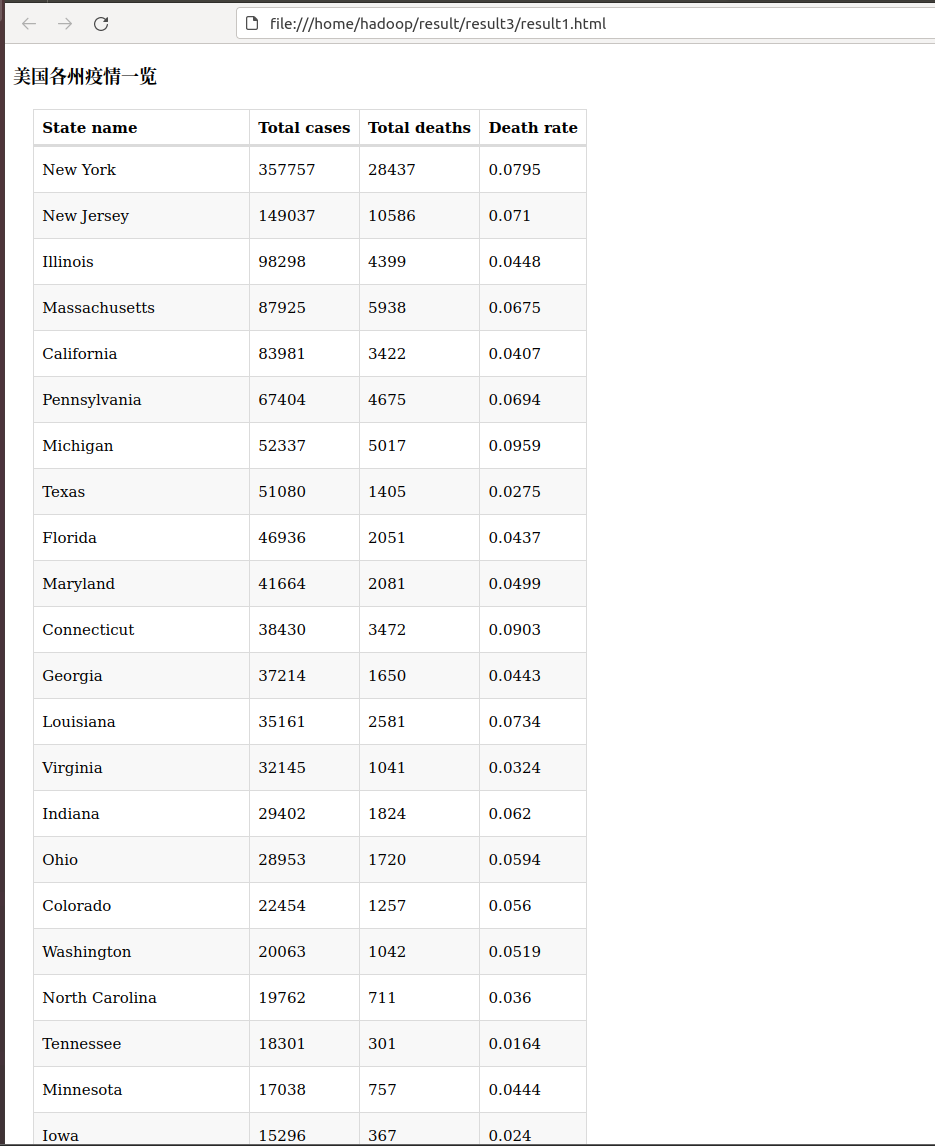
·美国累计确诊和死亡人数



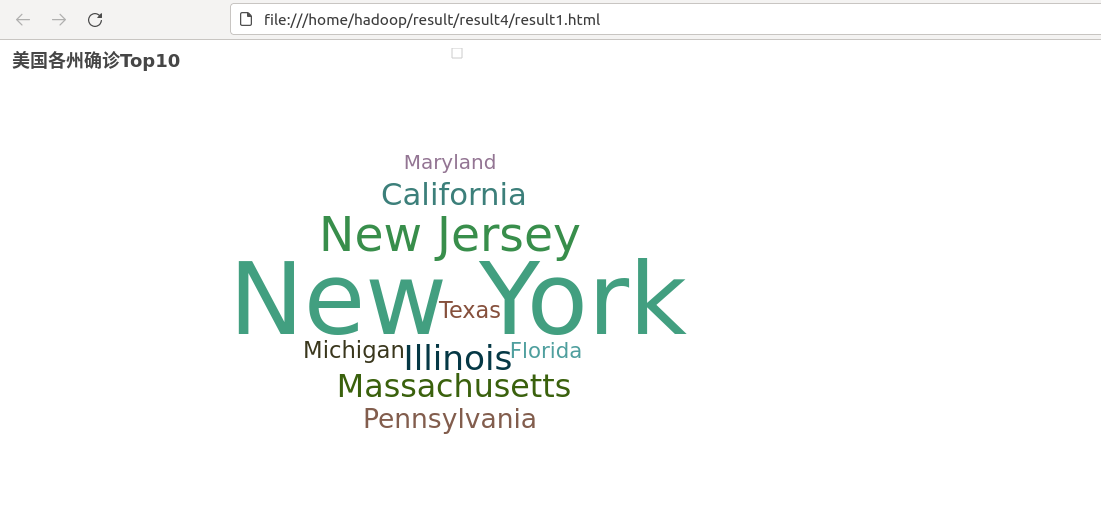
·美国每日新增死亡折线图



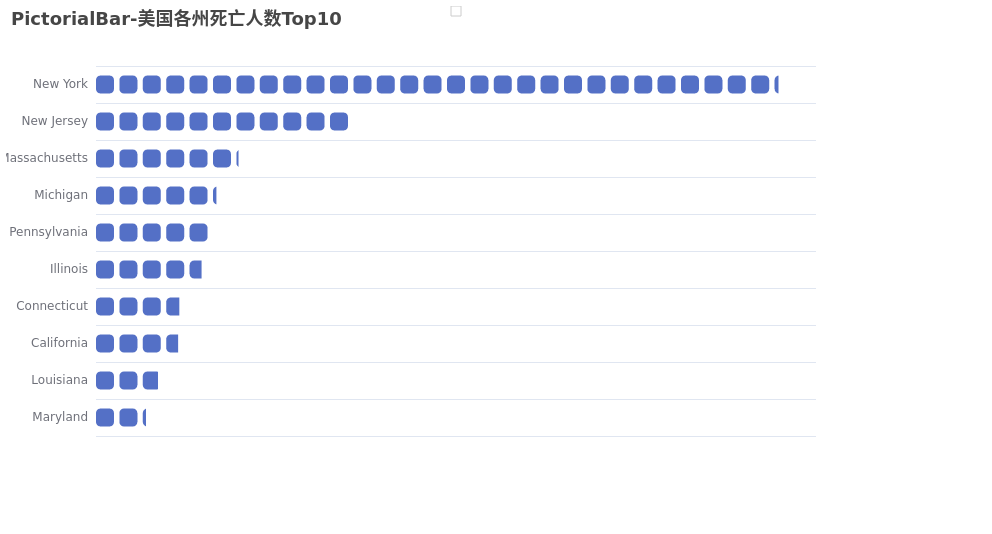
·美国各州疫情一览



·美国各州确诊Top10

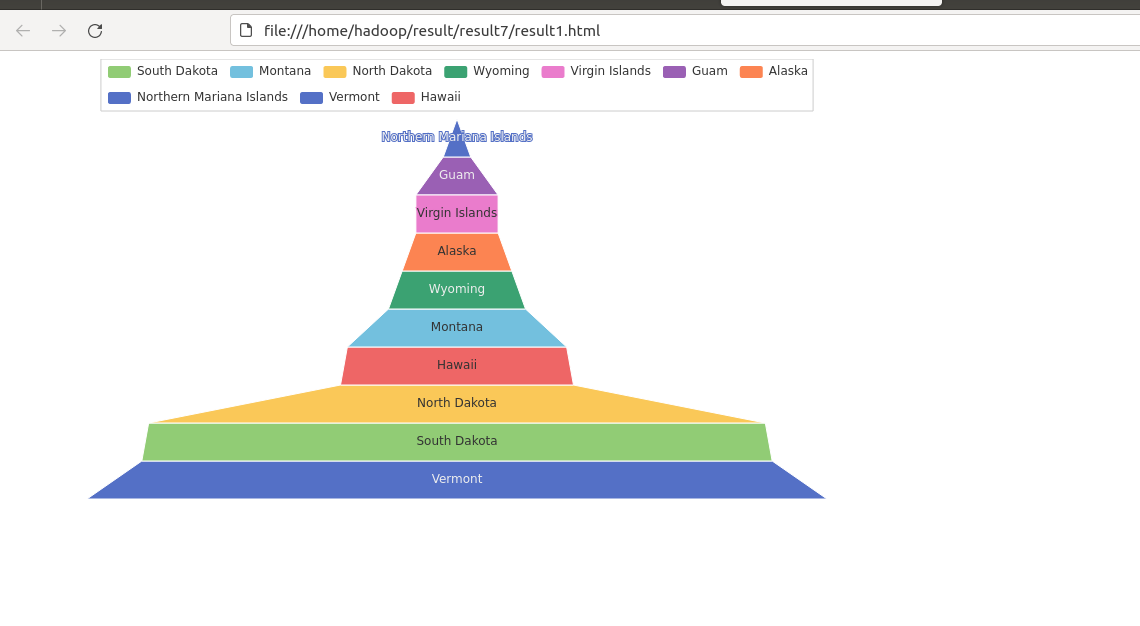


·美国各州死亡人数Top10

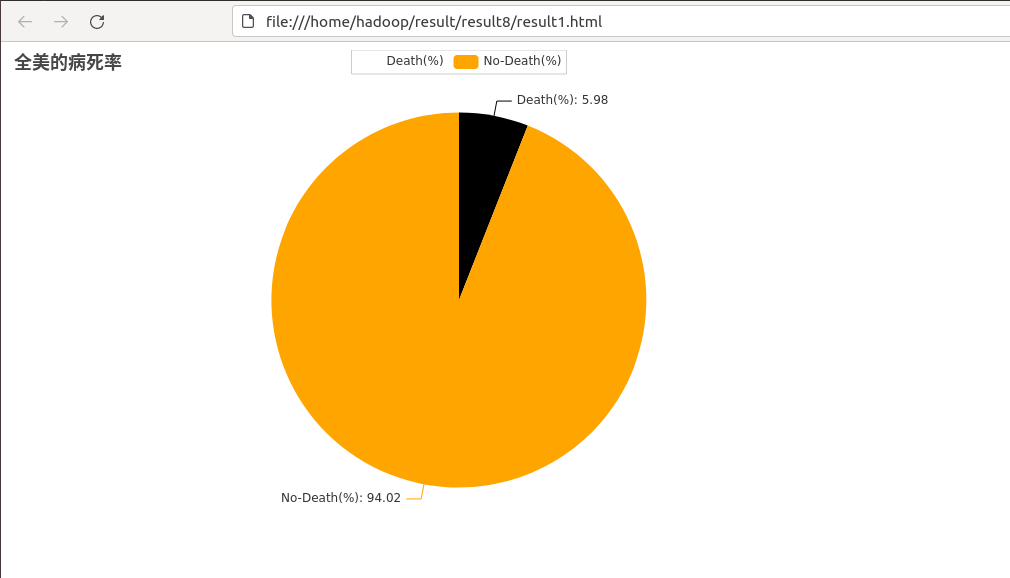


·美国确诊最少的10个州



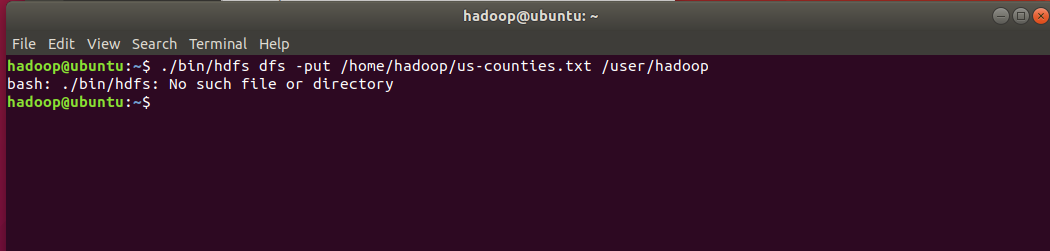


·全美病死率

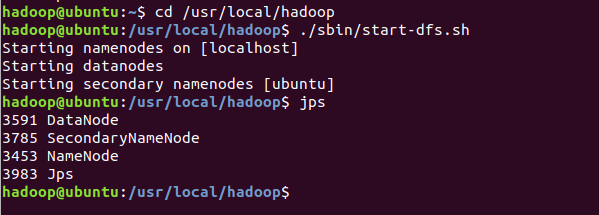


# 二、易错点总结

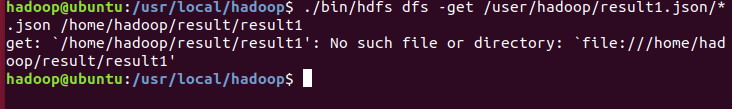
## （一）无法上传文件至HDFS



**解决方案：**启动Hadoop。



## （二）无法保存结果文件



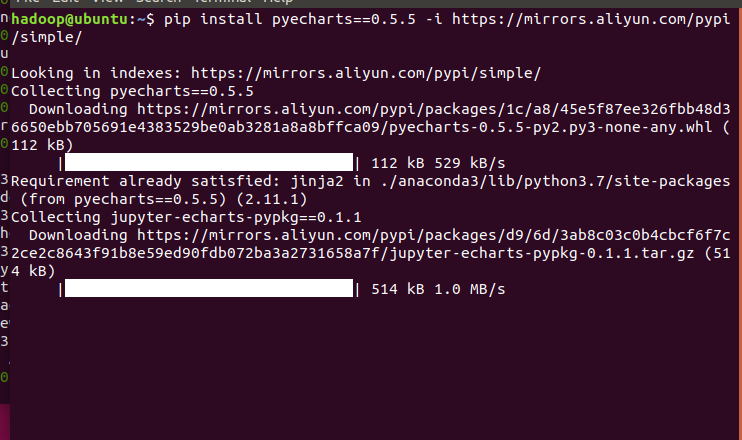
**解决方案：**电脑中不存在/result/result1/这个路径，自己对着这个路径创建相应的文件夹再执行shell命令即可。

## 无法安装pyecharts

直接使用pip install pyecharts命令进行安装时，大概率会因为网关原因而无法响应或者安装失败

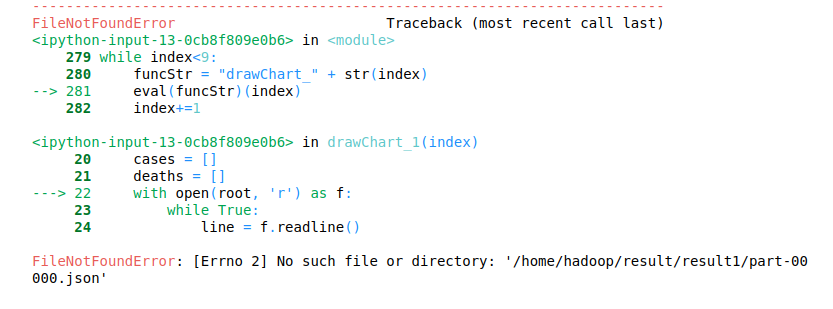


**解决方案：**换源（笔者下载的是阿里云的源）



## 无法保存可视化结果

这个报错的大致意思就是没有part-00000.json这个文件（因为其他路径都是对得上的）进入文件夹查看json文件，发现其格式是part-00000-一长串字符.json，



**解决方案：**把代码中root对应的part-00000.json改成与文件夹中相对应的文件名（笔者命名格式为part-00000-1.json，对应result1文件夹，再在代码中做对应修改即可）

# 三、学习小记

## （1）认识Spark

Spark最初由美国加州伯克利大学（UCBerkeley）的AMP实验室于2009年开发，是基于内存计算的大数据并行计算框架，可用于构建大型的、低延迟的数据分析应用程序，Spark具有如下特点：

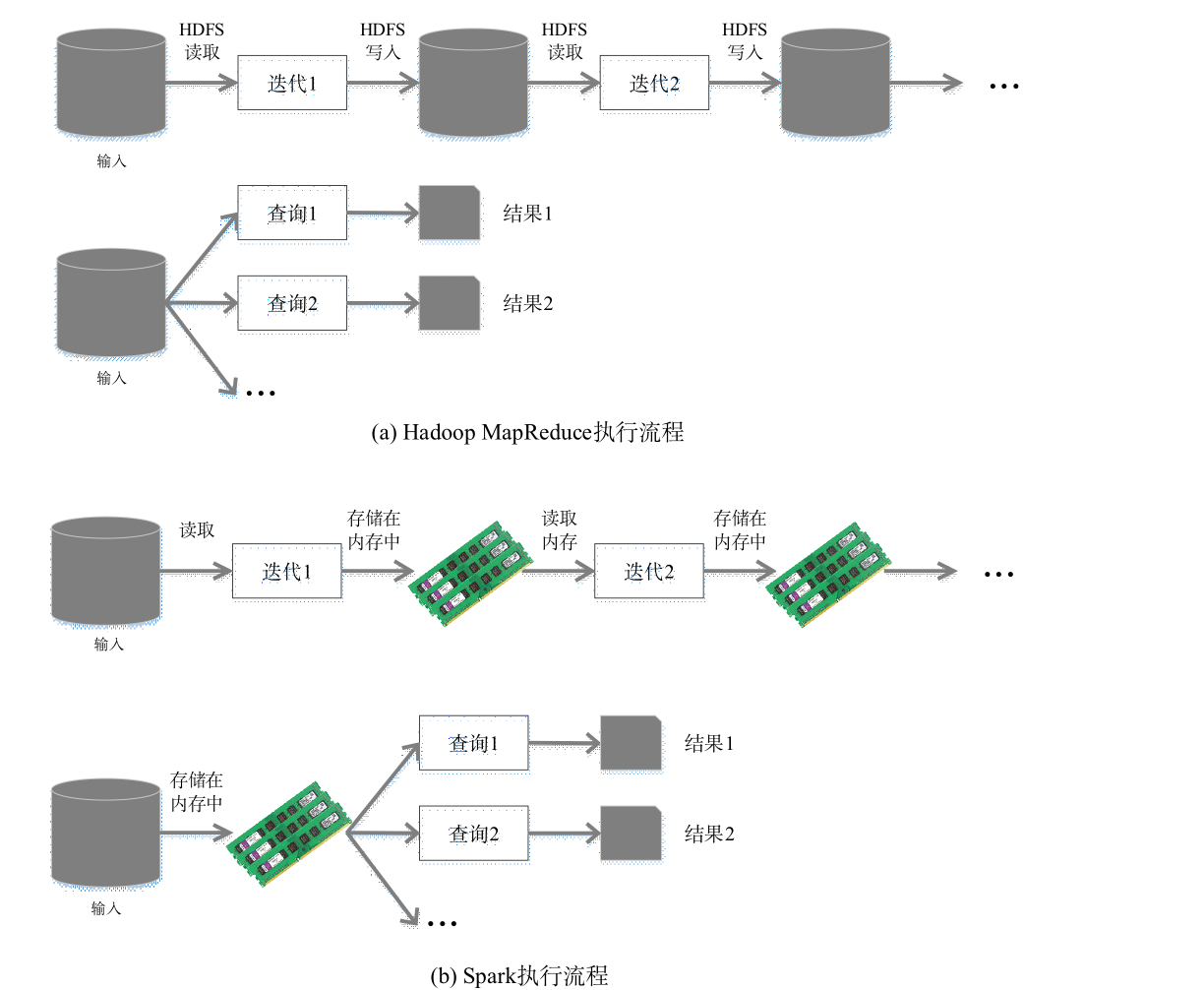
·运行速度快：使用DAG执行引擎以支持循环数据流与内存计算

·容易使用：支持使用Scala、Java、Python和R语言进行编程，可以通过Spark Shell进行交互式编程

·通用性：Spark提供了完整而强大的技术栈，包括SQL查询、流式计算、机器学习和图算法组件

·运行模式多样：可运行于独立的集群模式中，可运行于Hadoop中，也可运行于Amazon EC2等云环境中，并且可以访问HDFS、Cassandra、HBase、Hive等多种数据源

## （2）Spark与Hadoop执行流程对比



## Spark运行架构

·Spark运行架构运行架构包括集群资源管理器（Cluster Manager）、运行作业任务的工作节点（Worker Node）、每个应用的任务控制节点（Driver）和每个工作节点上负责具体任务的执行进程（Executor）

·资源管理器可以自带或Mesos或YARN



## RDD运行原理

·一个RDD就是一个分布式对象集合，本质上是一个只读的分区记录集合，每个RDD可分成多个分区，每个分区就是一个数据集片段，并且一个RDD的不同分区可以被保存到集群中不同的节点上，从而可以在集群中的不同节点上进行并行计算

·RDD提供了一种高度受限的共享内存模型，即RDD是只读的记录分区的集合，不能直接修改，只能基于稳定的物理存储中的数据集创建RDD，或者通过在其他RDD上执行确定的转换操作（如map、join和group by）而创建得到新的RDD

RDD典型对执行过程如下：

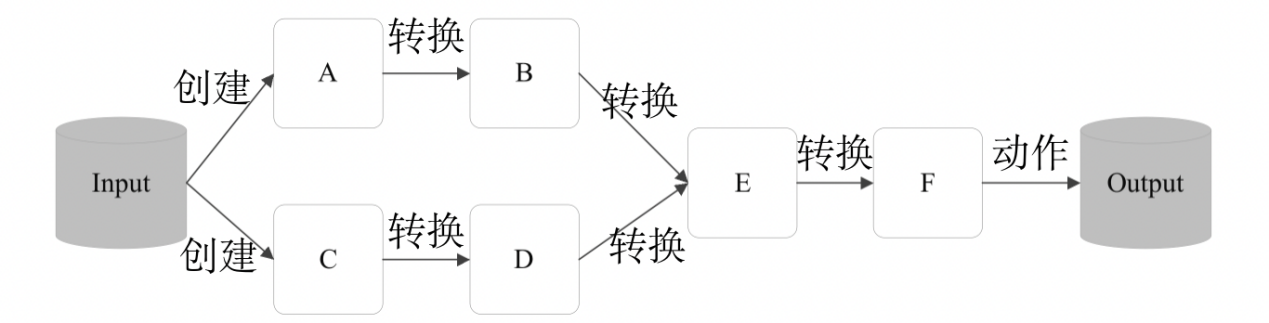
·RDD读入外部数据源进行创建

·RDD经过一系列的转换（Transformation）操作，每一次都会产生不同的RDD，供给下一个转换操作使用

·最后一个RDD经过“动作”操作进行转换，并输出到外部数据源

·这一系列处理称为一个Lineage（血缘关系），即DAG拓扑排序的结果

优点：惰性调用、管道化、避免同步等待、不需要保存中间结果、每次操作变得简单



## RDD在Spark中的运行过程

1. 创建RDD对象
2. SparkContext负责计算RDD之间的依赖关系，构建DAG
3. DAGScheduler负责把DAG图分解成多个Stage，每个Stage中包含了多个Task，每个Task会被TaskScheduler分发给各个WorkerNode上的Executor去执行。

