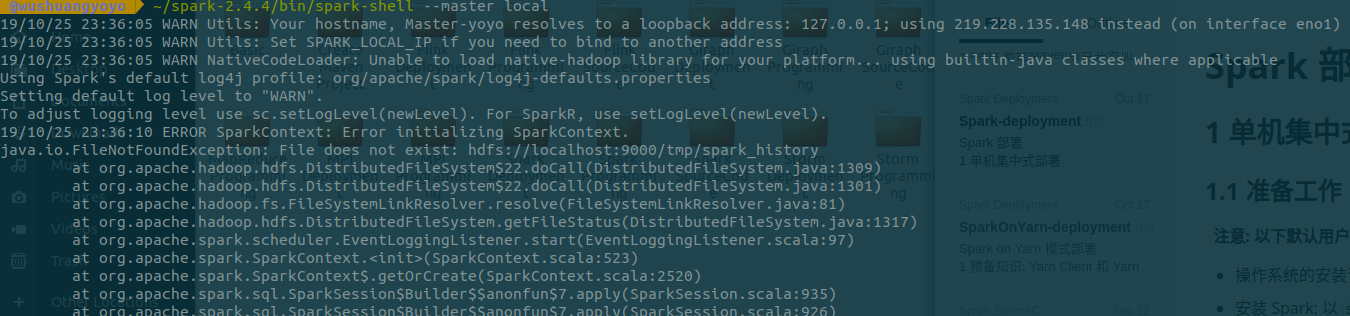
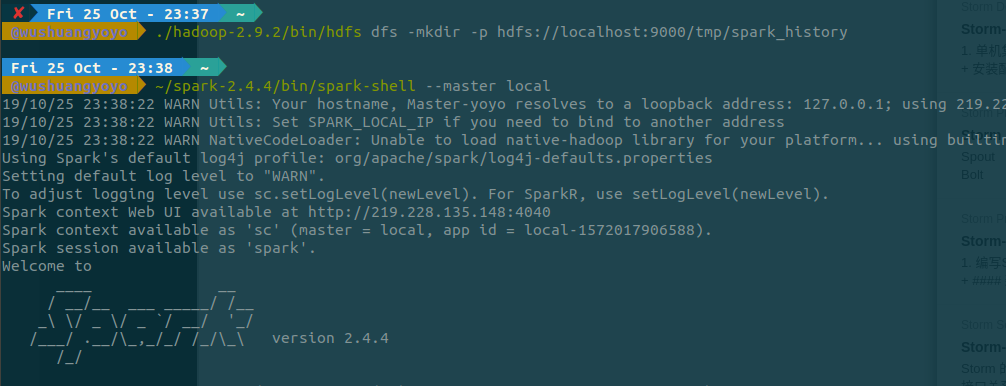
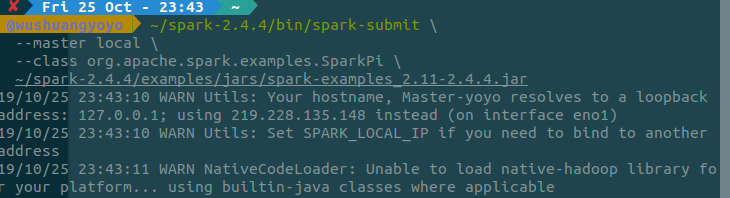
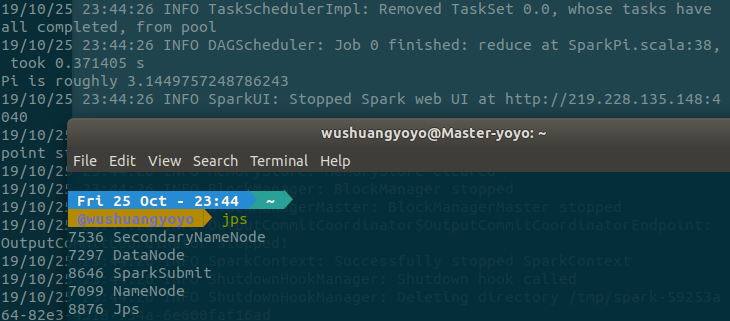
## 一. Spark 部署

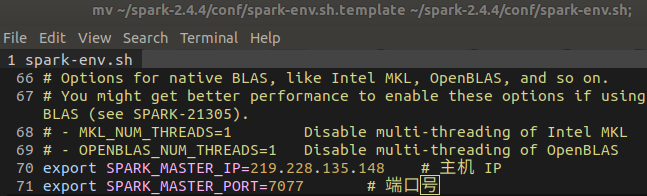
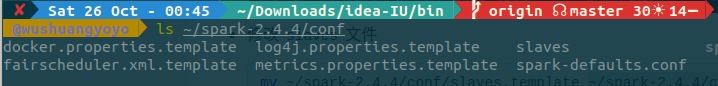
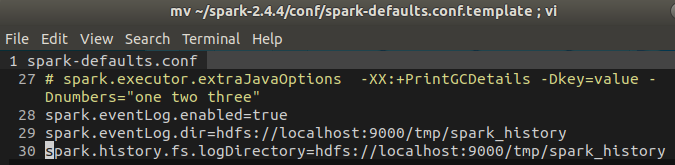
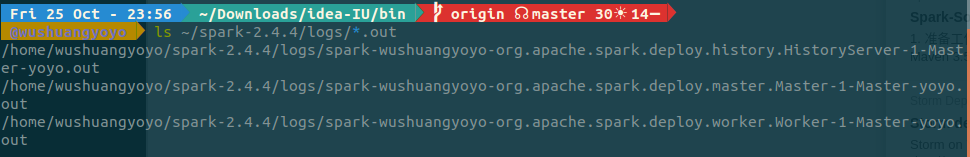
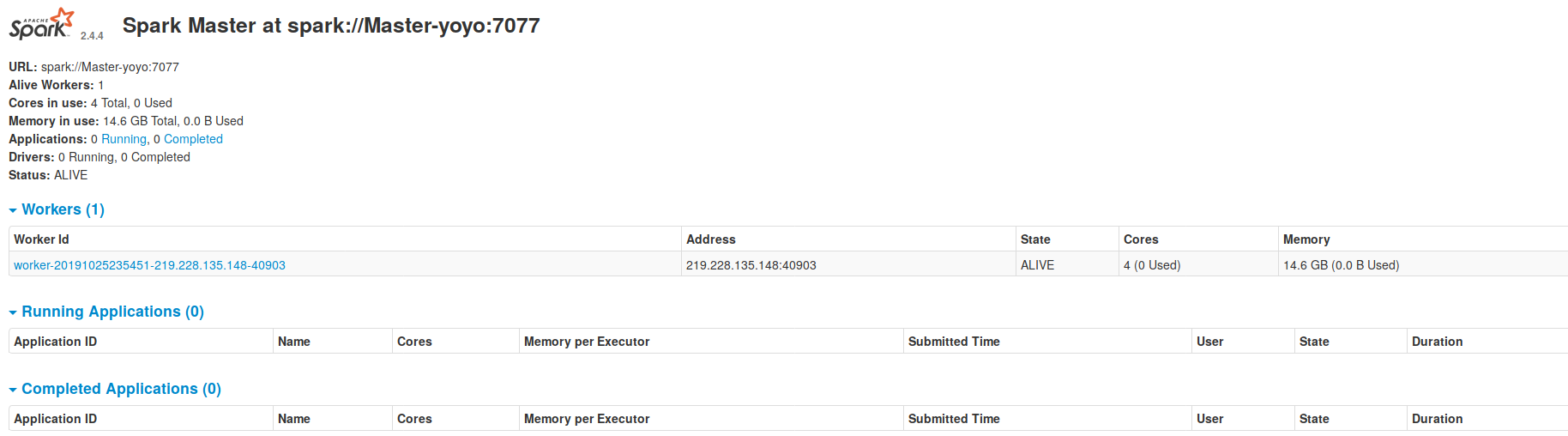
### 1. 单机集中式部署

#### 1.1 安装并运行 Spark 应用程序

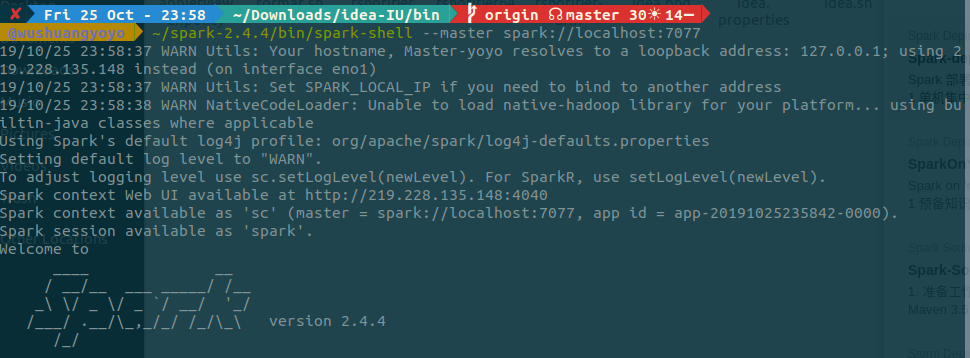
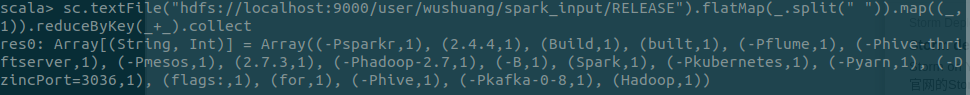
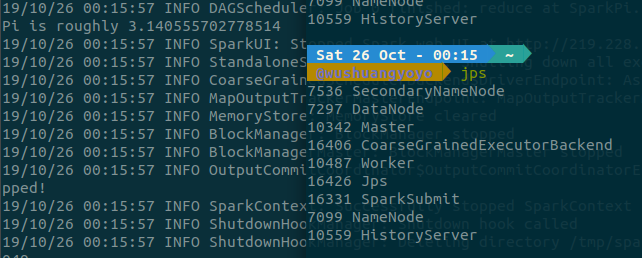
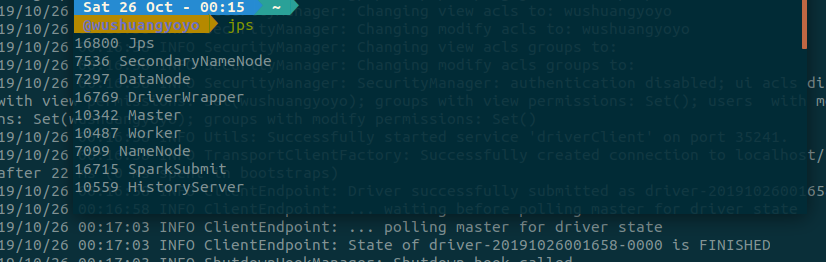
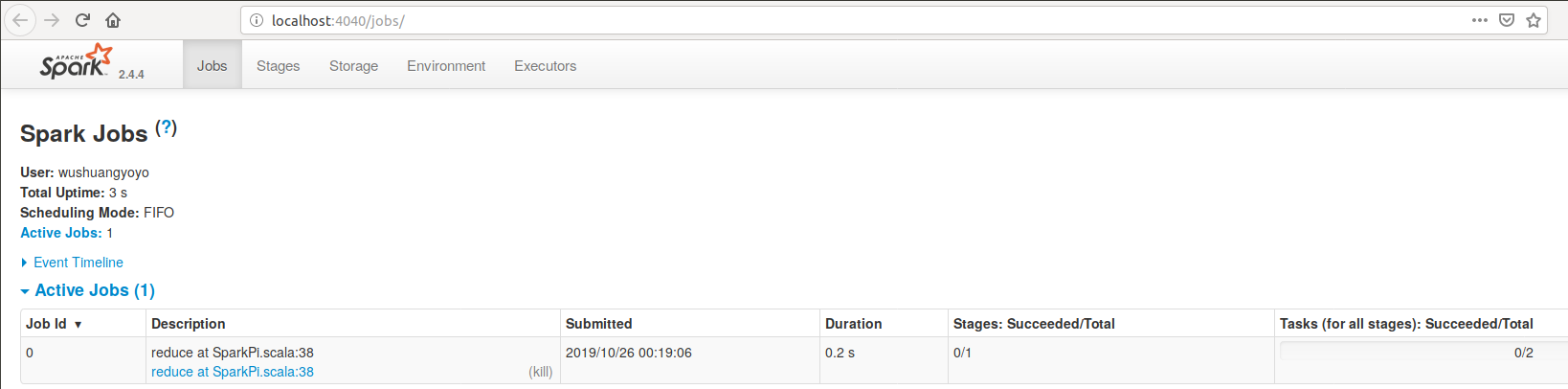
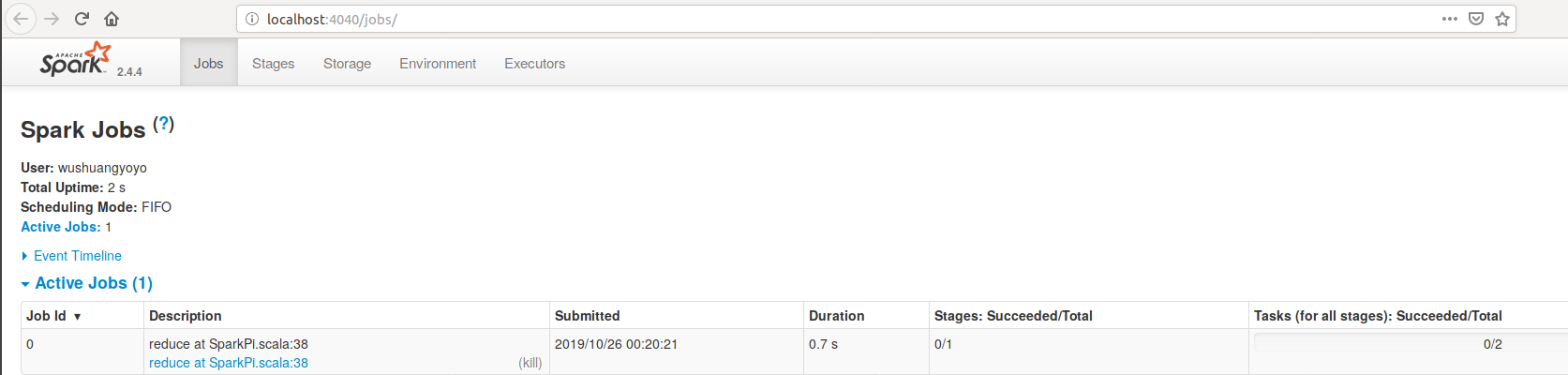
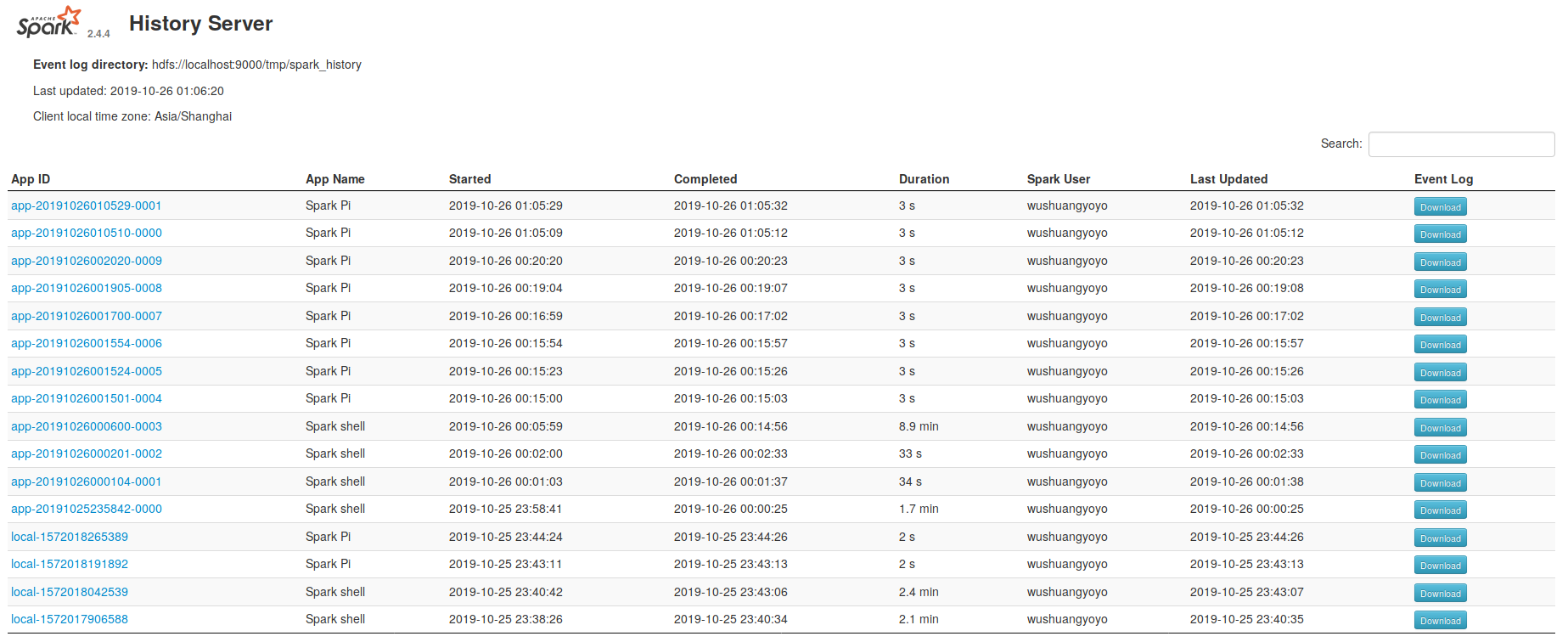
* 通过 Spark-Shell 运行应用程序
  + 进入 Spark-Shell
  + $ ~/spark-2.4.4/bin/spark-shell --master local
  + **遇到问题**：
  + 
  + 问题原因只是因为之前做过一次单机伪分布式的部署，所以这里单机集中式部署会报错，其实只需要把所需要的文件（hdfs://localhost:9000/tmp/spark\_history）新建一下就可以了
  + 
* 在 scala> 后输入 Scala 代码.
* scala> sc.textFile("file:///home/you/spark-2.4.4/RELEASE").flatMap(\_.split(" ")).map((\_,1)).reduceByKey(\_+\_).collect
* 执行后应打印出如下结果
* scala> res2: Array[(String, Int)] = Array((-Psparkr,1), (-B,1), (Spark,1), (-Pkubernetes,1), (-Pyarn,1), (2.4.4,1), (Build,1), (built,1), (-Pflume,1), (-DzincPort=3036,1), (flags:,1), (-Phive-thriftserver,1), (-Pmesos,1), (for,1), (-Phive,1), (-Pkafka-0-8,1), (2.7.3,1), (-Phadoop-2.7,1), (Hadoop,1))
* 如下，运行成功
* 
* 通过提交 Jar 包运行应用程序
* $ ~/spark-2.4.4/bin/spark-submit \  
   --master local \  
   --class org.apache.spark.examples.SparkPi \  
   ~/spark-2.4.4/examples/jars/spark-examples\_2.11-2.4.4.jar
* 运行结果如下图所示:
* 
* 
* 在运行过程中另起一个终端执行 jps 查看进程.此时只会出现 SparkSubmit 进程, 应用程序运行结束后该进程消失. 

### 2. 单机伪分布式部署

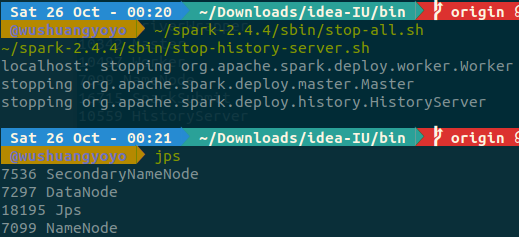
#### 2.1 修改配置并启动Spark服务

* 修改 spark-env.sh 文件
* 
* 修改 slaves 文件
* 
* 修改 spark-defaults.conf 文件
* 
* 启动命令
* $ ~/spark-2.4.4/sbin/start-all.sh # 启动 Spark  
  $ ~/spark-2.4.4/sbin/start-history-server.sh # 启动应用日志服务器
* 结果如下,可通过jps查看进程来检测是否启动成功:
* 
* 也可通过查看 Spark 服务日志
* 
* 访问 Spark Web 界面, 可看到 Master 和 Worker: http://localhost:8080
* 

#### 2.2 运行 Spark 应用程序

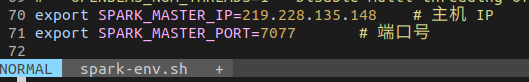
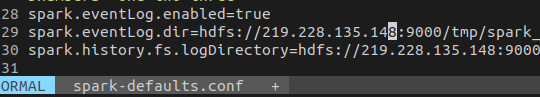
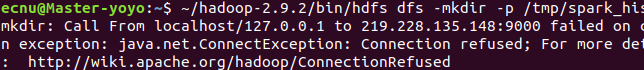
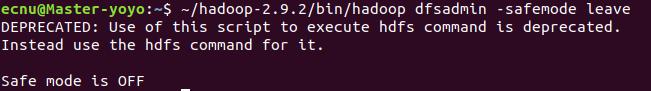
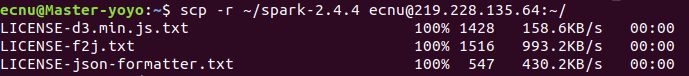
* 通过 Spark-Shell 运行应用程序
  + 准备输入文件
  + 将本地的 ~/spark-2.4.4/RELEASE 上传至 HDFS 的 /user/you/spark\_input 下
  + 进入 Spark-Shell
  + $ ~/spark-2.4.4/bin/spark-shell --master spark://localhost:7077
  + 
* 在 scala> 后输入 Scala 代码.
* scala> sc.textFile("hdfs://localhost:9000/user/you/spark\_input/RELEASE").flatMap(\_.split(" ")).map((\_,1)).reduceByKey(\_+\_).collect
* 执行后应打印出如下结果
* scala> res2: Array[(String, Int)] = Array((-Psparkr,1), (-B,1), (Spark,1), (-Pkubernetes,1), (-Pyarn,1), (2.4.4,1), (Build,1), (built,1), (-Pflume,1), (-DzincPort=3036,1), (flags:,1), (-Phive-thriftserver,1), (-Pmesos,1), (for,1), (-Phive,1), (-Pkafka-0-8,1), (2.7.3,1), (-Phadoop-2.7,1), (Hadoop,1))
* 如图,运行成功:
* 
* 通过提交 Jar 包运行应用程序
* Client 提交模式 (默认), 此模式下 Driver 运行在客户端, 可以在客户端看到应用程序运行过程中的信息
* $ ~/spark-2.4.4/bin/spark-submit \  
   --deploy-mode client \  
   --master spark://localhost:7077 \  
   --class org.apache.spark.examples.SparkPi \  
   ~/spark-2.4.4/examples/jars/spark-examples\_2.11-2.4.4.jar
* 在运行过程中另起一个终端执行 jps 查看进程.运行结果如下图所示:
* 
* Cluster 提交模式, 此模式下 Master 会随机选取一个 Worker 节点启动 Driver, 故在客户端看不到应用程序运行过程中的信息
* $ ~/spark-2.4.4/bin/spark-submit \  
   --deploy-mode cluster \  
   --master spark://localhost:7077 \  
   --class org.apache.spark.examples.SparkPi \  
   ~/spark-2.4.4/examples/jars/spark-examples\_2.11-2.4.4.jar
* 在运行过程中另起一个终端执行 jps 查看进程.运行结果如下图所示: 
* 实时查看应用运行情况
* 在应用运行过程中 (如进入 Spark-Shell 之后), 访问 http://localhost:4040
  + Client 模式:
  + 
  + Cluster 模式:
  + 
* 查看 Spark 应用程序日志
* 在提交一个应用程序后，在 ~/spark-2.4.4/work 下会出现应用程序运行日志
* 
* 查看应用历史记录
* 在应用运行结束后, 访问 http://localhost:18080
* 

#### 2.3 停止服务

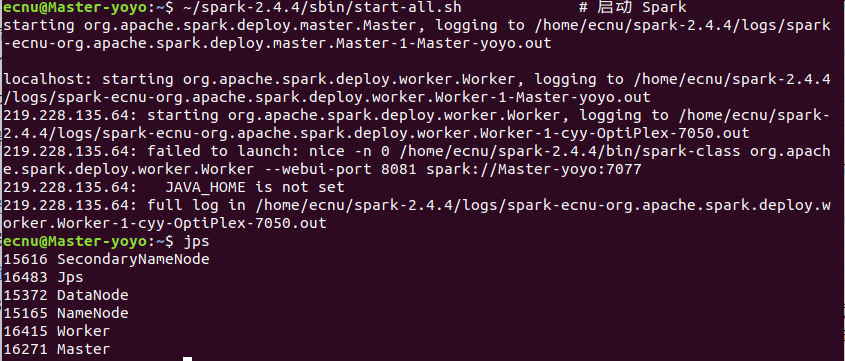
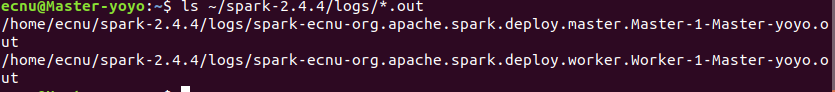
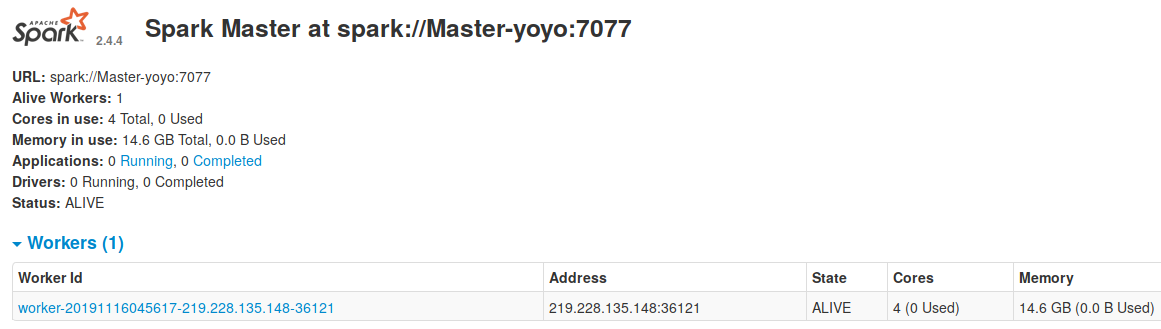
* 停止命令
* $ ~/spark-2.4.4/sbin/stop-all.sh # 停止 Spark  
  $ ~/spark-2.4.4/sbin/stop-history-server.sh # 停止日志服务器
* 

### 3. 分布式部署

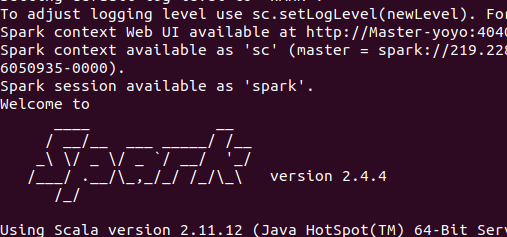
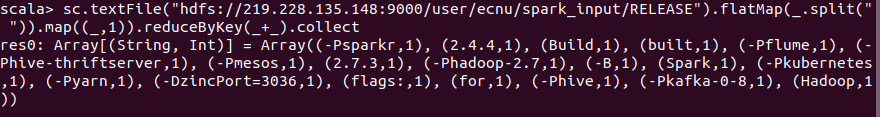
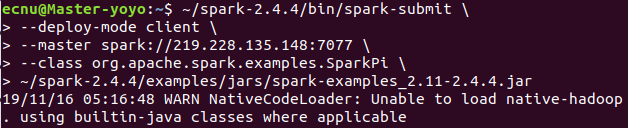
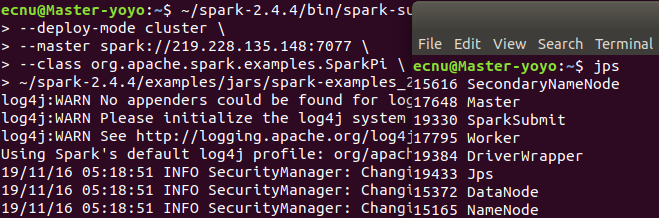
#### 3.1 准备工作&修改配置

* 修改 spark-env.sh 文件
* 
* 修改 slaves 文件, 添加节点 IP
* 219.228.135.148  
  219.228.135.64
* 修改 spark-defaults.conf 文件
* 
* 并在 HDFS 中建立目录 /tmp/spark\_history
* $ ~/hadoop-2.9.2/bin/hdfs dfs -mkdir -p /tmp/spark\_history
* 遇到问题:
* 
* 产生问题的原因 没有打开是 Spark,在打开 Spark 的过程中遇到另一个问题:
* 
* 解决问题方法如下:
* 
* 
* 将配置好的 Spark 拷贝到其它节点
* $ scp -r ~/spark-2.4.4 ecnu@219.228.135.64:~/

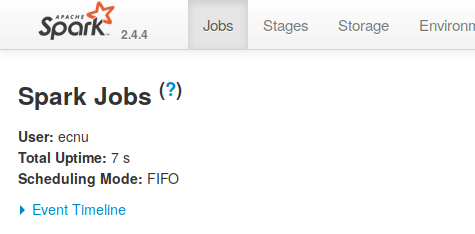
#### 3.3 启动服务&查看服务信息

* 启动命令 (在主节点上执行),执行 jps查看运行进程
* 
* 查看 Spark 服务日志
  + 主节点日志
* 
* 访问 Spark Web 界面, 可看到 Master 和 Worker: http://localhost:8080
* 

#### 3.5 运行 Spark 应用程序

* 通过 Spark-Shell 运行应用程序
  + 准备输入文件
  + 将本地的 ~/spark-2.4.4/RELEASE 上传至 HDFS 的 /user/ecnu/spark\_input 下
  + 
  + 进入 Spark-Shell
  + >>> ~/spark-2.4.4/bin/spark-shell --master spark://219.228.135.148:7077
  + 遇到问题:
  + 
  + hosts 被污染,解决问题方法如下:
  + 
  + 成功,结果如下:
  + 
  + 在 scala> 后输入 Scala 代码. 此处执行的是统计 RELEASE 文件中的单词数量
  + 
* 通过提交 Jar 包运行应用程序
  + Client 提交模式 (默认), 此模式下 Driver 运行在客户端, 可以在客户端看到应用程序运行过程中的信息
  + >>> ~/spark-2.4.4/bin/spark-submit \  
    >>> --deploy-mode client \  
    >>> --master spark://219.228.135.148:7077 \  
    >>> --class org.apache.spark.examples.SparkPi \  
    >>> ~/spark-2.4.4/examples/jars/spark-examples\_2.11-2.4.4.jar
  + 
  + 在运行过程中另起一个终端执行 jps 查看进程. 
* Cluster 提交模式, 此模式下 Master 会随机选取一个 Worker 节点启动 Driver, 故在客户端看不到应用程序运行过程中的信息
* ~/spark-2.4.4/bin/spark-submit \  
   --deploy-mode cluster \  
   --master spark://219.228.135.148:7077 \  
   --class org.apache.spark.examples.SparkPi \  
   ~/spark-2.4.4/examples/jars/spark-examples\_2.11-2.4.4.jar
* 在运行过程中另起一个终端执行 jps 查看进程.
* 

#### 3.6 查看 Spark 程序运行信息

* 实时查看应用运行情况
* 在应用运行过程中 (如进入 Spark-Shell 之后), 访问 http://192.168.1.11:4040
* 
* 查看 Spark 应用程序日志, 在 ~/spark-2.4.4/work 下
* 在提交一个应用程序后，在各子节点的 ~/spark-2.4.4/work 下会出现应用程序运行日志
* 

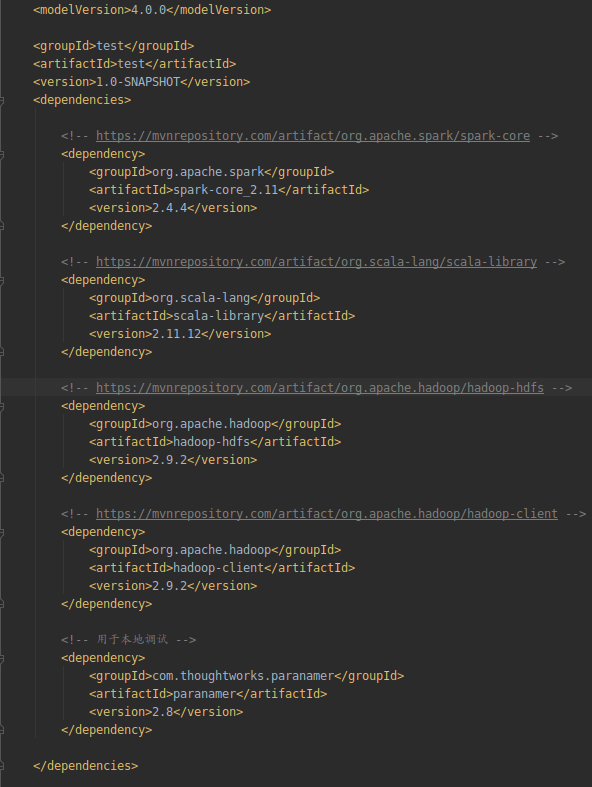
#### 3.7 停止服务

* 停止命令 (在主节点上执行)
* ~/spark-2.4.4/sbin/stop-all.sh # 停止 Spark  
  ~/spark-2.4.4/sbin/stop-history-server.sh # 停止日志服务器

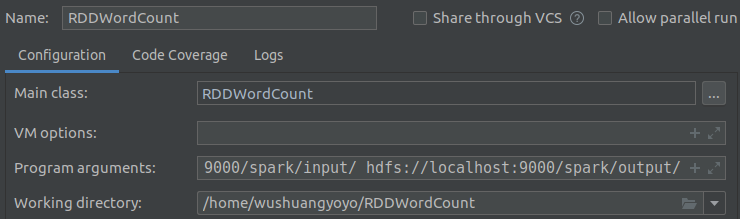
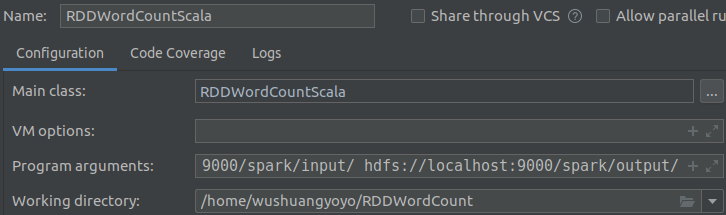
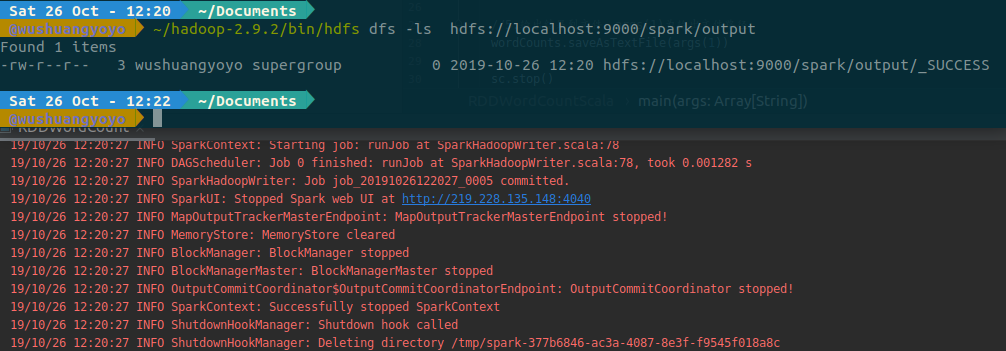
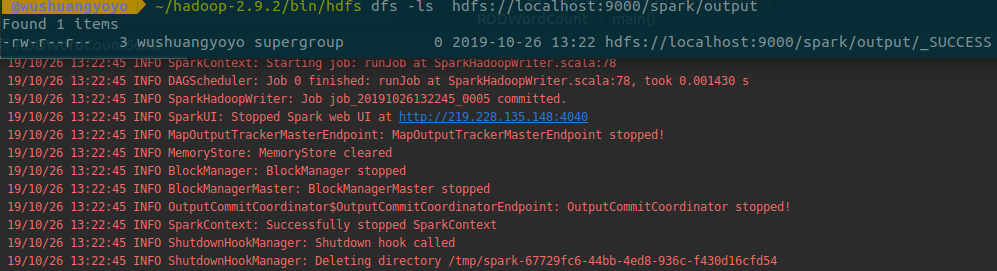
## 二. Spark-RDD-programming

### 1. 编写Spark程序

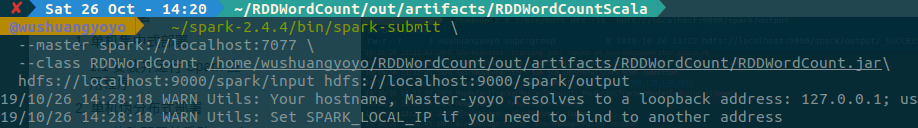
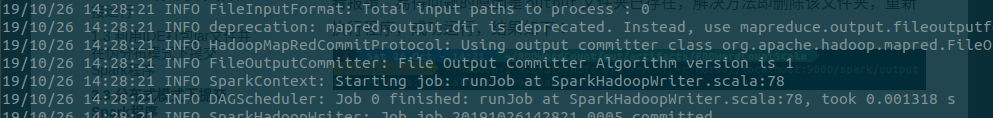
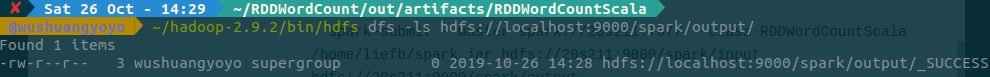
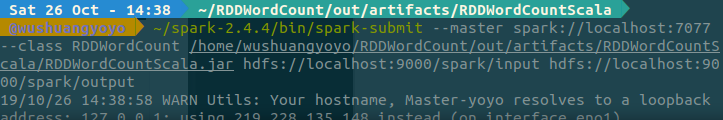
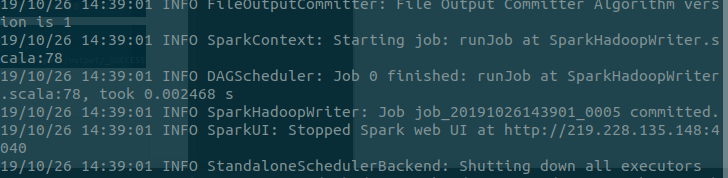
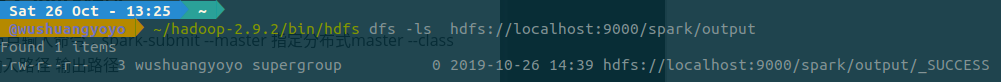
#### 1.1 新建Maven项目并添加pom依赖

* 新建maven项目并修改pom.xml根据计算机的spark和hadoop-hdfs版本进行相关修改。
* 

#### 1.2 IDE环境编写代码并直接运行

* 按照文档所示编写java以及scala代码，其中scala需要导入scala-plugin，配置运行环境如下：
  + Java:
  + 
  + Scala:
  + 
* 运行结果如图：
  + Java:
  + 
  + Scala:
  + 

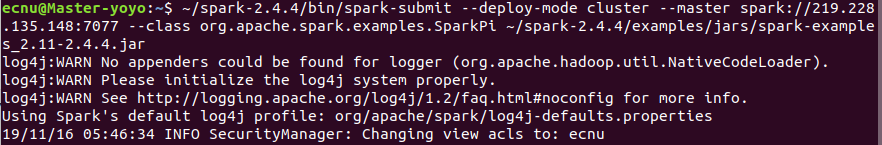
#### 1.3 利用IDE打包jar文件并在伪分布模式下提交Spark程序

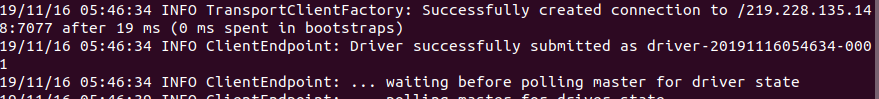
* 在Spark的Master节点输入命令:
  + Java模式：
  + $ ~/spark-2.4.4/bin/spark-submit \  
    --master spark://localhost:7077 \  
    --class RDDWordCount /home/wushuangyoyo/RDDWordCount/out/artifacts/RDDWordCount/RDDWordCount.jar\  
    hdfs://localhost:9000/spark/input hdfs://localhost:9000/spark/output
  + **遇到问题**： fig: 看报错，易得问题的原因是output文件夹已存在，解决方法即删除该文件夹，重新执行程序，成功运行。结果如下：
  + 
  + 
  + 
  + 
  + Scala模式:
  + $ ~/spark-2.4.4/bin/spark-submit \  
     --master spark://localhost:7077 \  
     --class RDDWordCount /home/wushuangyoyo/RDDWordCount/out/artifacts/RDDWordCountScala/RDDWordCountScala.jar\  
     hdfs://localhost:9000/spark/input hdfs://localhost:9000/spark/output
  + 结果如下：
  + 
  + 
  + 

#### 3.3 分布式模式下提交Spark程序

* 如下所示(其中20s211为主机名，也可用ip地址)：

~/spark-2.4.4/bin/spark-submit --master spark://219.228.135.148:7077 --class RDDWordCountScala /home/wushuangyoyo/RDDWordCount/out/artifacts/RDDWordCountScala/RDDWordCountScala.jar hdfs://219.228.135.148:9000/spark/input hdfs://219.228.135.148:9000/spark/output

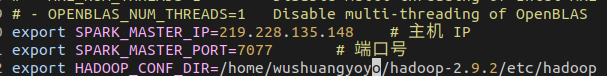
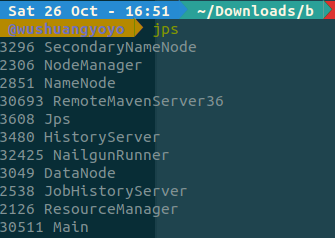




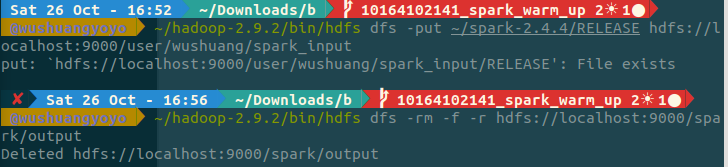
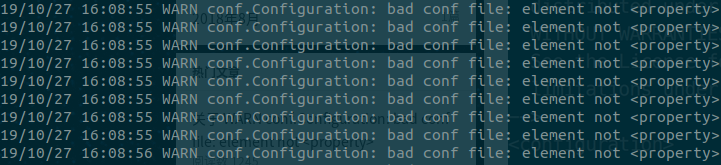
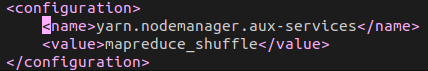
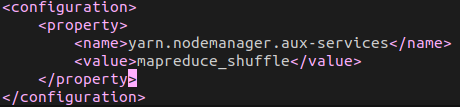
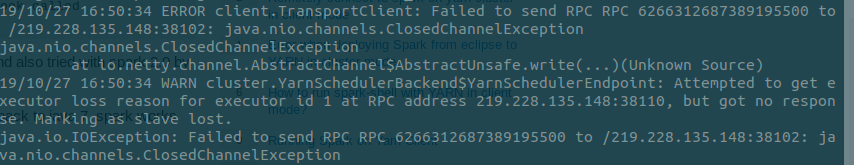
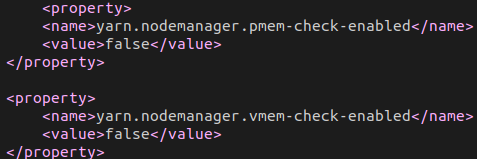
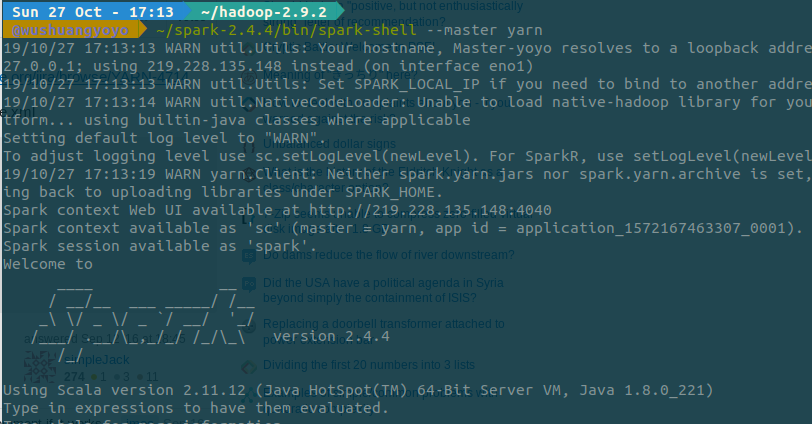
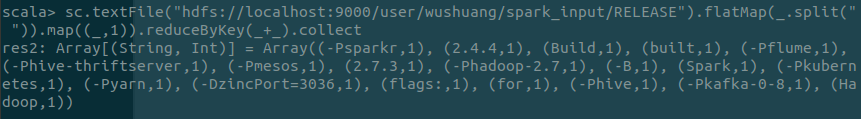
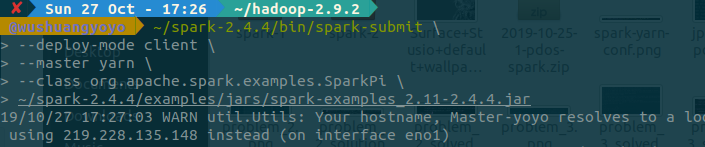
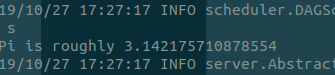
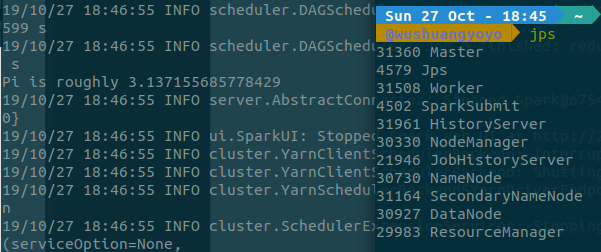
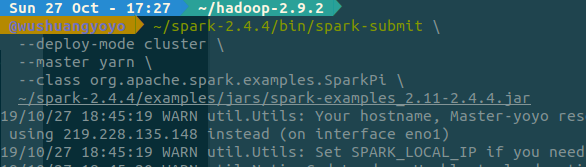
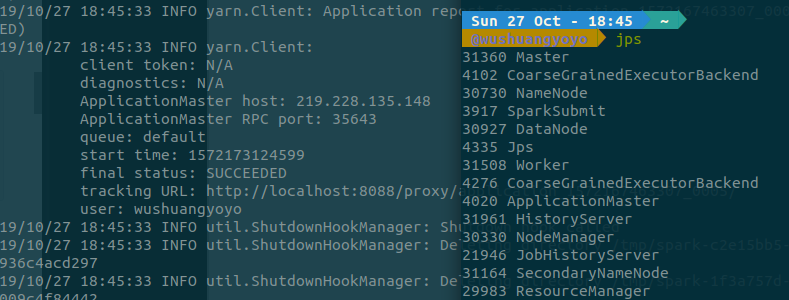
## 三. Spark on Yarn 模式部署

### 1 单机伪分布式部署

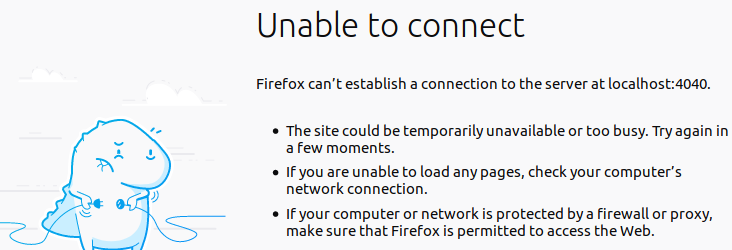
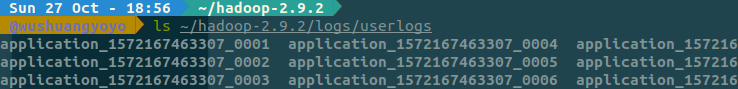
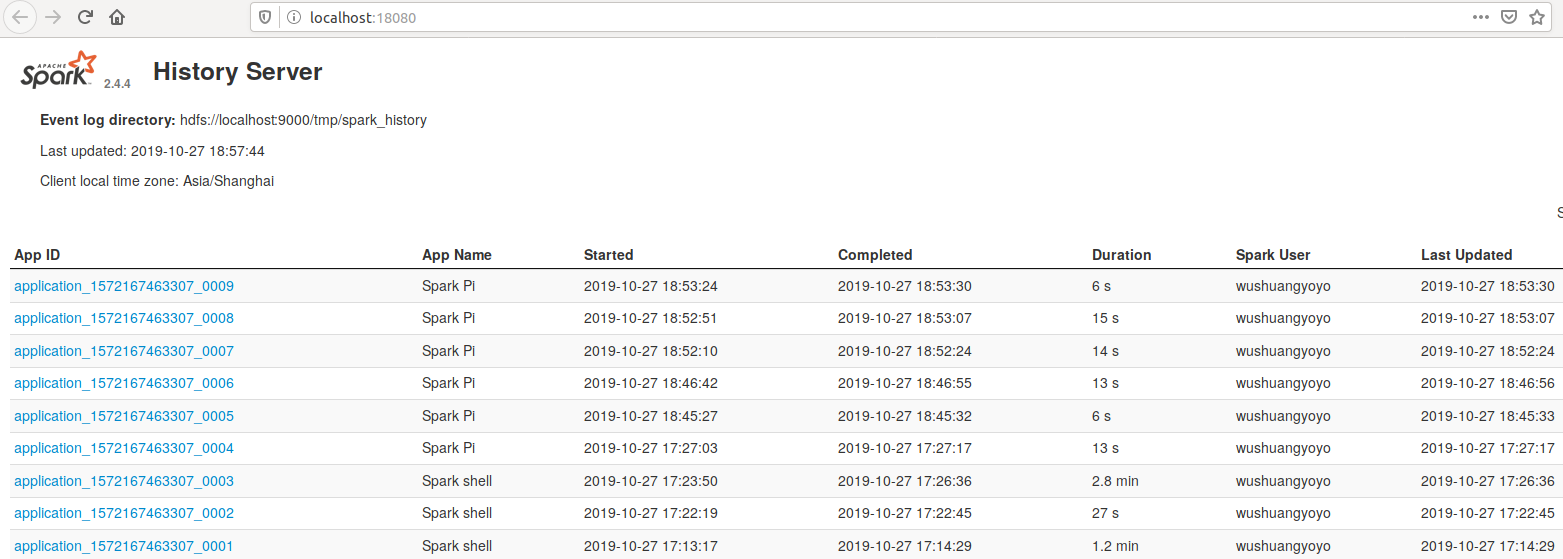
#### 1.1 修改配置并启动Spark服务

* 修改 spark-env.sh 文件, 使 Spark 能够读取 Yarn 配置，结果如下：
* 
* 启动命令。通过jps查看服务信息来确认启动成功：
* 

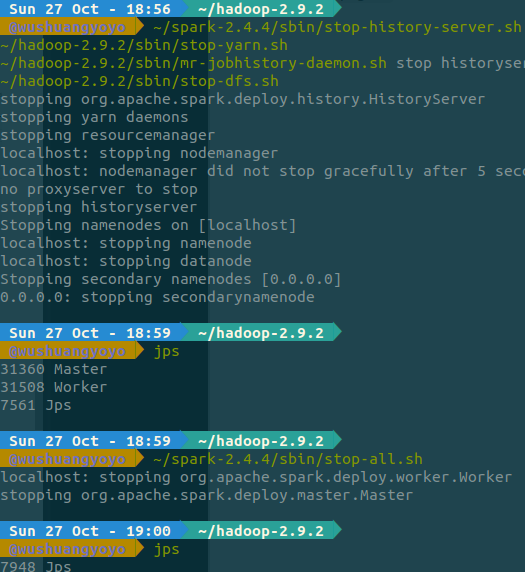
#### 1.2 运行 Spark 应用程序

* 通过 Spark-Shell 运行应用程序
  + 准备输入文件
  + 由于之前的实验过程中很多次传输过所需文件，这里只需要删除一下spark/output和目录下的文件。
  + 
  + 进入 Spark-Shell
  + $ ~/spark-2.4.4/bin/spark-shell --master yarn
  + **出现问题**,如下:
  + 
  + 查出原因是hadoop的配置文件中有的Configuration里设置了和但是没有写框架:
  + 
  + 解决方法为将之修改为:
  + 
  + 重启hadoop-2.9.2下的yarn.再次运行,出现**另一个问题**:
  + 
  + 问题原因如网站[YARN-4714](https://issues.apache.org/jira/browse/YARN-4714)所说,是一个blocker问题,虚拟内存使用过高.
  + 解决方法为在Hadoop 配置文件 yarn-site.xml 中添加取消虚拟内存的检查设置,具体如下图所示:
  + 
  + 再次格式化,重新运行.成功,结果如下:
  + 
  + 在 scala> 后输入 Scala 代码.
* scala> sc.textFile("hdfs://localhost:9000/user/ecnu/spark\_input/RELEASE").flatMap(\_.split(" ")).map((\_,1)).reduceByKey(\_+\_).collect
* 运行成功,结果如下: 
* 通过提交 Jar 包运行应用程序
  + Client 提交模式 (默认), 此模式下 Driver 运行在客户端, 可以在客户端看到应用程序运行过程中的信息
  + 
  + 成功,结果如下:
  + 
  + 在另一个terminal中运行jps,查看此时运行的进程:
  + 
* Cluster 提交模式, 此模式下 ResourceManager 会随机选取一个 NodeManager 所在节点在其上启动 ApplicationMaster.由于Driver 运行在 ApplicationMaster 中, 故在客户端看不到应用程序运行过程中的信息:
* 
* 在另一个terminal中运行jps,查看此时运行的进程:
* 

#### 1.3 查看 Spark 程序运行信息

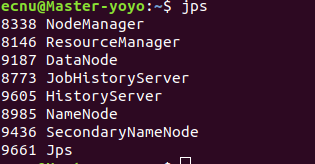
* 实时查看应用运行情况
* 在应用运行过程中 (如进入 Spark-Shell 之后), 访问 http://localhost:4040
  + Client:
  + 
  + Cluster:
  + 由于Driver不是运行在本地的,所以没法登陆.
  + 
* 查看 Spark 应用程序日志
* 在提交一个应用程序后，在 ~/hadoop-2.9.2/logs/userlogs 下会出现应用程序运行日志
* 
* 查看应用历史记录
* 在应用运行结束后, 访问 http://localhost:18080
* 
* web\_logs

#### 1.4 停止服务

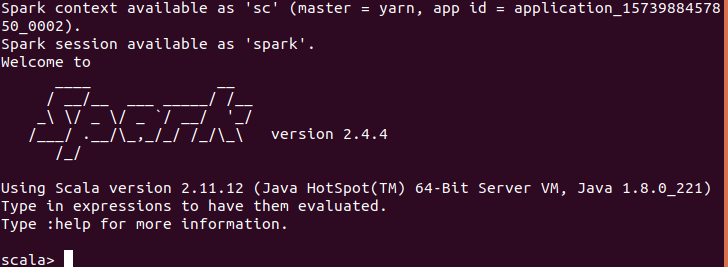
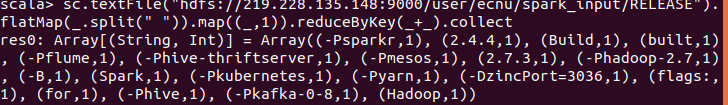
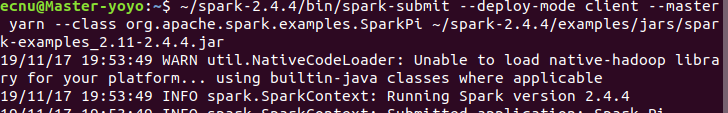
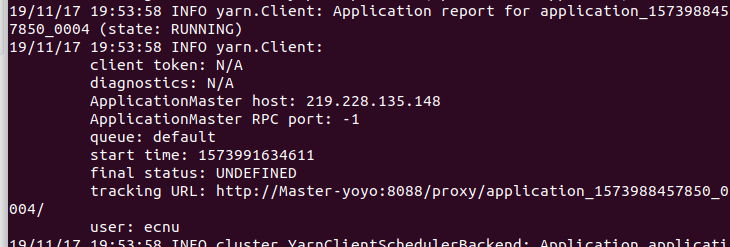
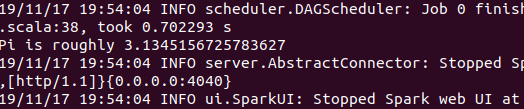
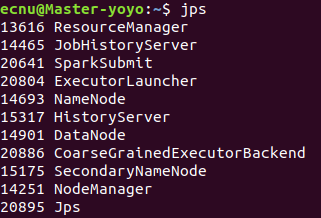
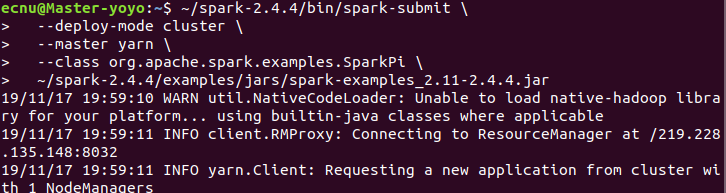
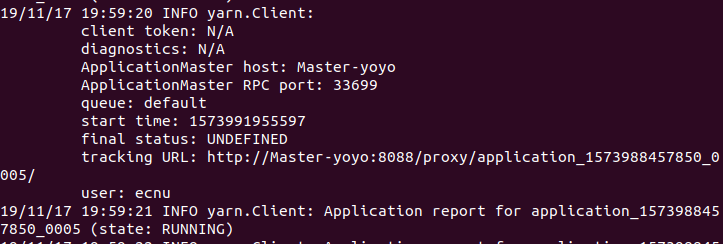
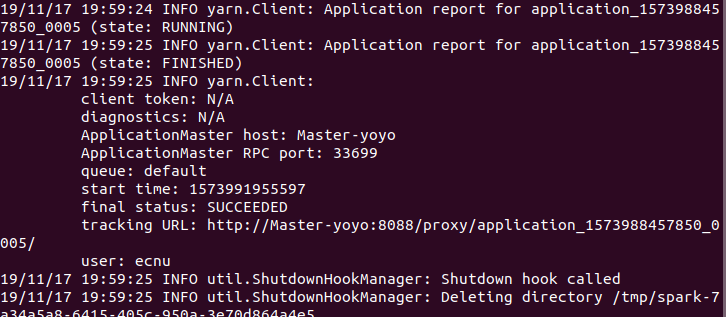
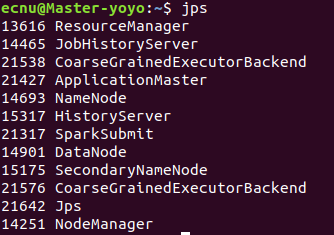


### 2 分布式部署

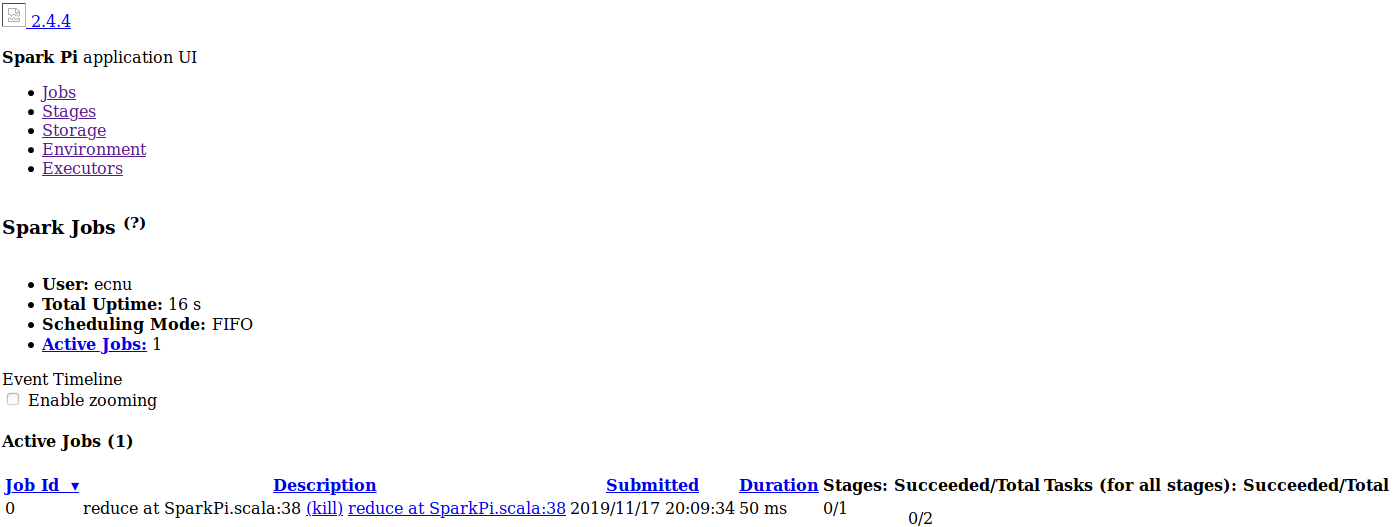
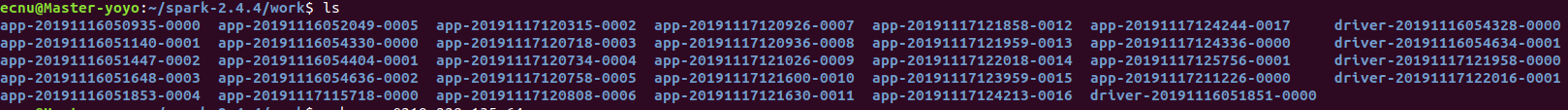
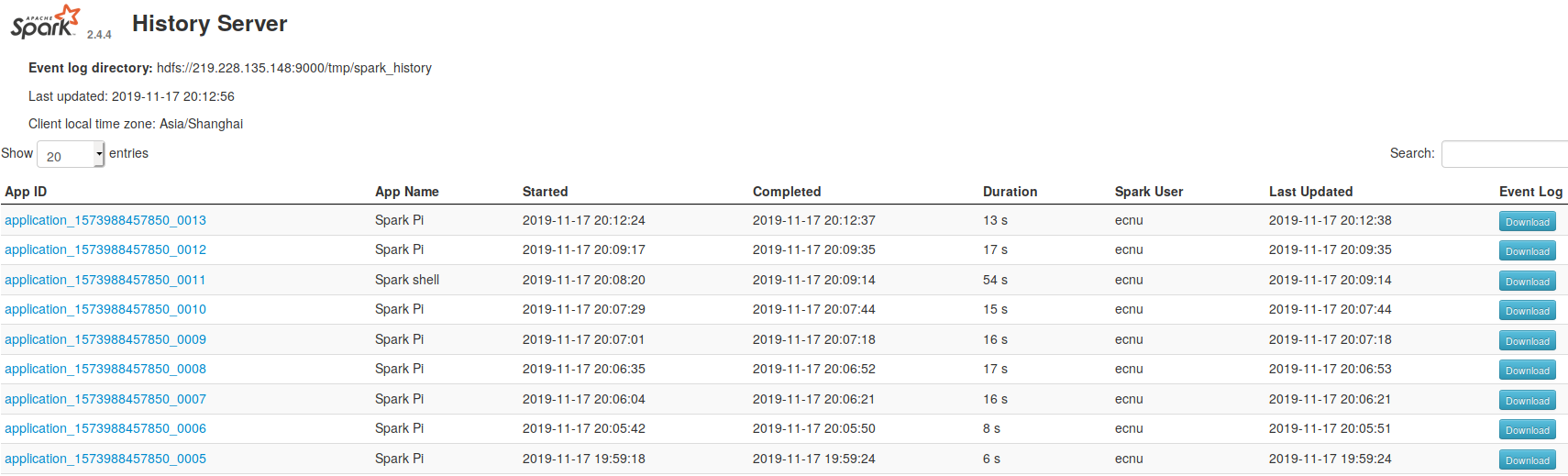
#### 2.1 准备工作与启动服务

* 启动命令 (在 Yarn 主节点上执行)，并在主节点的terminal中执行jps来验证是否成功执行服务
* ~/hadoop-2.9.2/sbin/start-yarn.sh  
  ~/hadoop-2.9.2/sbin/mr-jobhistory-daemon.sh start historyserver  
  ~/hadoop-2.9.2/sbin/start-dfs.sh  
  ~/spark-2.4.4/sbin/start-history-server.sh  
    
  jps
* 运行成功，结果如下：
* 

#### 2.4 运行 Spark 应用程序

* 通过 Spark-Shell 运行应用程序
  + 准备输入文件
  + 将本地的 ~/spark-2.4.4/RELEASE 上传至 HDFS 的 /user/you/spark\_input 下
  + 进入 Spark-Shell
  + $ ~/spark-2.4.4/bin/spark-shell --master yarn
  + 
  + 并在 scala> 后输入 Scala 代码. 此处执行的是统计 RELEASE 文件中的单词数量
  + scala> sc.textFile("hdfs://219.228.135.148:9000/user/ecnu/spark\_input/RELEASE").flatMap(\_.split(" ")).map((\_,1)).reduceByKey(\_+\_).collect
  + 结果如下：
  + 
* 通过提交 Jar 包运行应用程序
  + Client 提交模式 (默认), 此模式下 Driver 运行在客户端, 可以在客户端看到应用程序运行过程中的信息
  + >>> ~/spark-2.4.4/bin/spark-submit \  
    >>> --deploy-mode client \  
    >>> --master yarn \  
    >>> --class org.apache.spark.examples.SparkPi \  
    >>> ~/spark-2.4.4/examples/jars/spark-examples\_2.11-2.4.4.jar
  + 
* 
* 
* 同时，在运行过程中另起一个终端执行 jps 查看进程。
* 结果如下：   
   
  + Cluster 提交模式, 此模式下 ResourceManager 会随机选取一个 NodeManager 所在节点在其上启动 ApplicationMaster, Driver 运行在 ApplicationMaster 中, 故在客户端看不到应用程序运行过程中的信息
  + ~/spark-2.4.4/bin/spark-submit \  
     --deploy-mode cluster \  
     --master yarn \  
     --class org.apache.spark.examples.SparkPi \  
     ~/spark-2.4.4/examples/jars/spark-examples\_2.11-2.4.4.jar
* 
* 
* 
* 同时，在运行过程中另起一个终端执行 jps 查看进程。
* 结果如下：
* 

#### 2.5 查看 Spark 程序运行信息

* 实时查看应用运行情况
* 在应用运行过程中，访问 http://219.228.135.148:4040。得到如下结果：
* 
* 查看 Spark 应用程序日志
* 不知为何，我的spark分布式运行变成了standalone的模式，所有的logs都存在了NodeManager节点的 spark安装目录/work/目录，结果如下：
* 
* 查看应用历史记录
* 在应用运行结束后, 访问 http://219.228.135.148:18080
* 

#### 2.6 停止服务

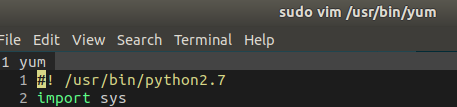
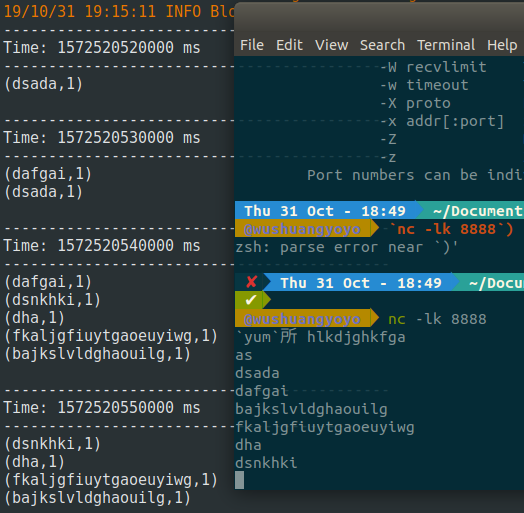
* 停止命令 (在 Yarn 主节点上执行)
* ~/spark-2.4.4/sbin/stop-history-server.sh  
  ~/hadoop-2.9.2/sbin/stop-yarn.sh  
  ~/hadoop-2.9.2/sbin/mr-jobhistory-daemon.sh stop historyserver  
  ~/hadoop-2.9.2/sbin/stop-dfs.sh

## 四. Spark-Streaming-Programming

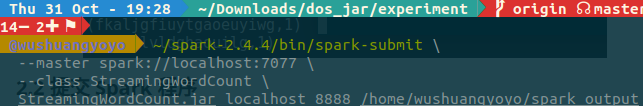
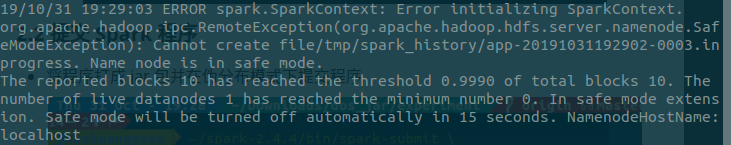
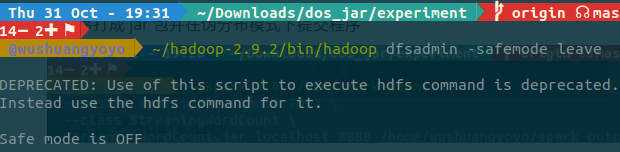
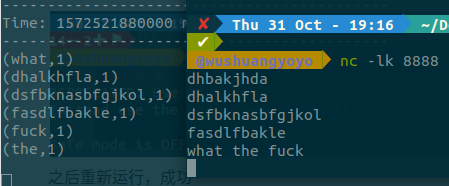
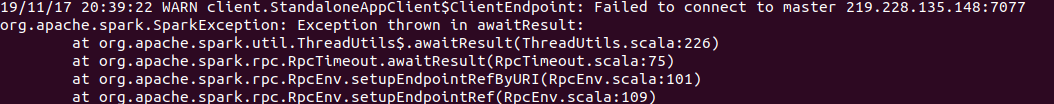
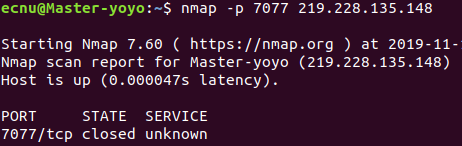
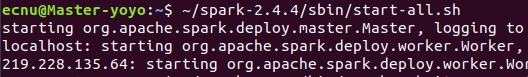
### 1 编写程序

新建 Maven 项目，设置依赖，添加类并编程。

#### 1.1 IDEA 中运行

* 安装 Netcat (yum install nc), 并选择一个端口号 (如8888) 开启 Netcat 服务 (nc -lk 8888)
* 遇到问题：
* 
* 这种情况出现的原因是yum所需python版本不兼容，于是修改/usr/bin/yum：
* 
* 再次运行，发现可以运行，但是需要重新配置repositories，非常繁琐，于是直接使用apt，也便于管理，运行成功，结果如下：
* 
* 
* 配置运行环境，设IP 地址为开启 Netcat 的 IP, 端口号为开启 Netcat 的端口号。运行程序，成功。结果如下：
* 

#### 1.2 提交 Spark 程序

* 将程序打成 jar 包并在伪分布模式下提交程序
* 
* 出现问题：
* 
* 出现这种情况的原因是NameNode发现集群中DataNode丢失达到一定比例，于是会进入安全模式，此时只允许查看数据不允许对数据进行任何操作。解决方法为：
* 
* 之后重新运行，成功。结果如图：
* 
* 分布式模式下提交程序
* ~/spark-2.4.4/bin/spark-submit \  
   --master spark://219.228.135.148:7077 \  
   --class StreamingWordCount \  
   /home/ecnu/Downloads/StreamingWordCount.jar 219.228.135.148 8888 hdfs://219.228.135.148:9000/user/ecnu/spark\_output
* 遇到问题：
* 
* 问题的描述是ip地址为219.228.135.148（即Master）端口为7077的节点无法访问，通过nmap判断7077是否被占用：
* 
* 发现并没有，于是重新审理整个过程，发现master和slaves没有启动，于是重新启动spark：
* 
* 重新运行，成功。结果如下：
* 