实验三：大数据离线数据分析实践-MR+Spark

刘帅 2020212267

实验内容

本次实验共分为三部分：

- 第一部分为基于Hadoop 集群的MapReduce 大数据批处理实践；
- 第二部分为Spark 的单机伪分布式集群搭建，以及基于该Spark 集群的大数据批处理实践；
- 第三部分为Spark 的完全分布式集群搭建，以及基于该Spark 集群的大数据批处理实践；

一、hadoop mapreduce wordcount程序案例再现

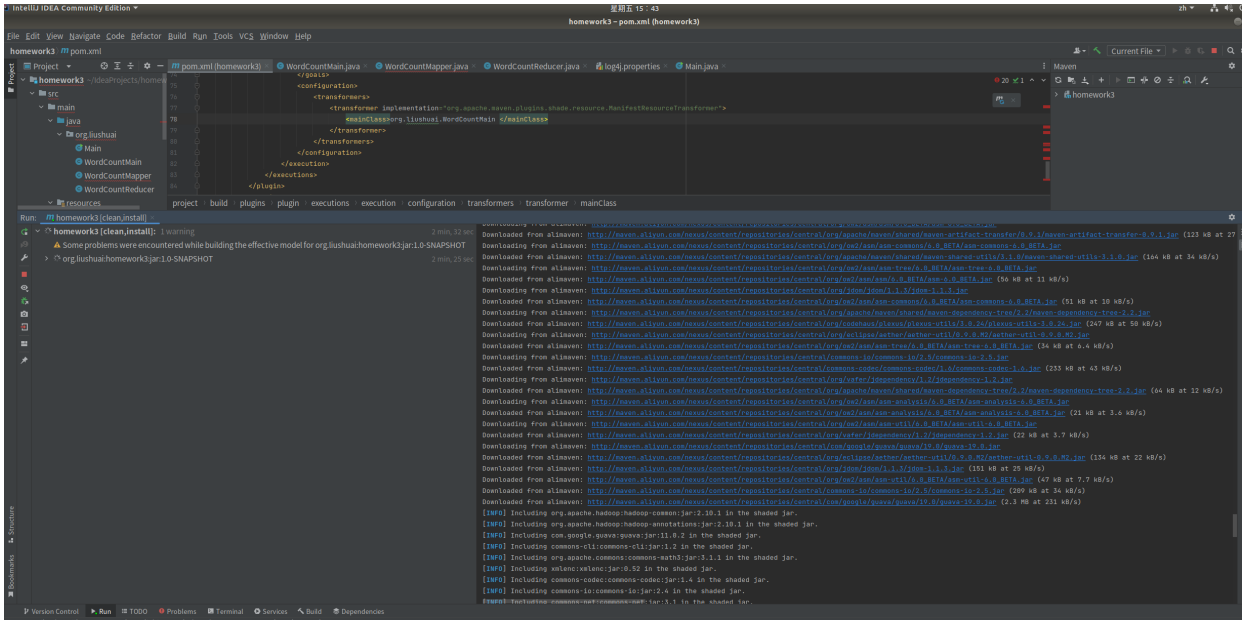
1.1 java项目创建与环境依赖配置

首先根据实验二中的方法创建java项目，并进行maven配置，随后在pom.xml中导入hadoop、slf4j、httpcomponents等相关依赖；

执行

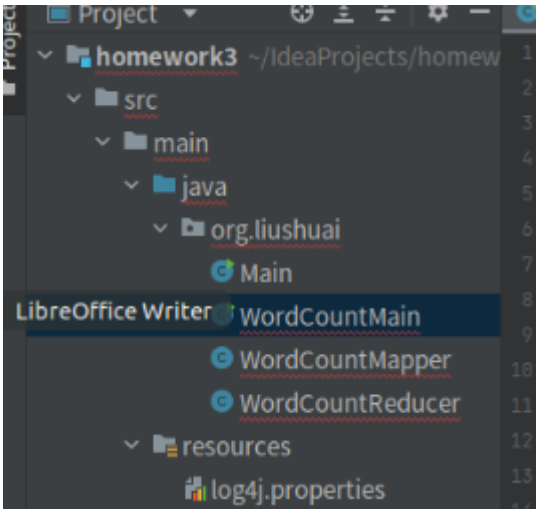
```
mvn clear install
```

将依赖下载到project当中



1.2 编写mapper、reducer、main代码

创建WordCountMain、WordCountMapper、WordCountReducer 三个类



其中:

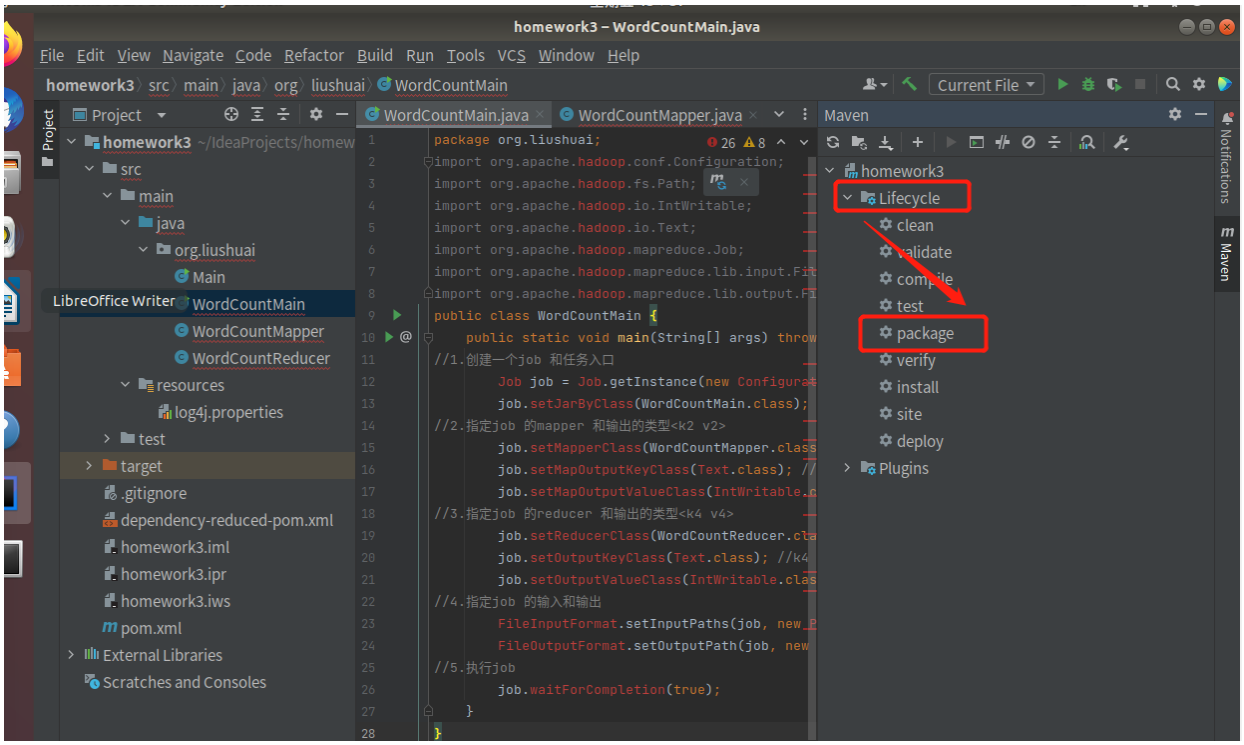
WordCountMain 是 MapReduce 作业的主程序，它负责设置作业的各种参数，例如输入路径、输出路径、Mapper 类、Reducer 类等，并启动作业的执行。

WordCountMapper是 MapReduce 作业中的 Mapper 组件，它负责将输入数据拆分为若干个键值对，并对每个键值对执行一次映射操作。在词频统计的场景下，WordCountMapper的任务是将输入的文本数据拆分为单词和出现次数的键值对，其中键是单词，值是 1。

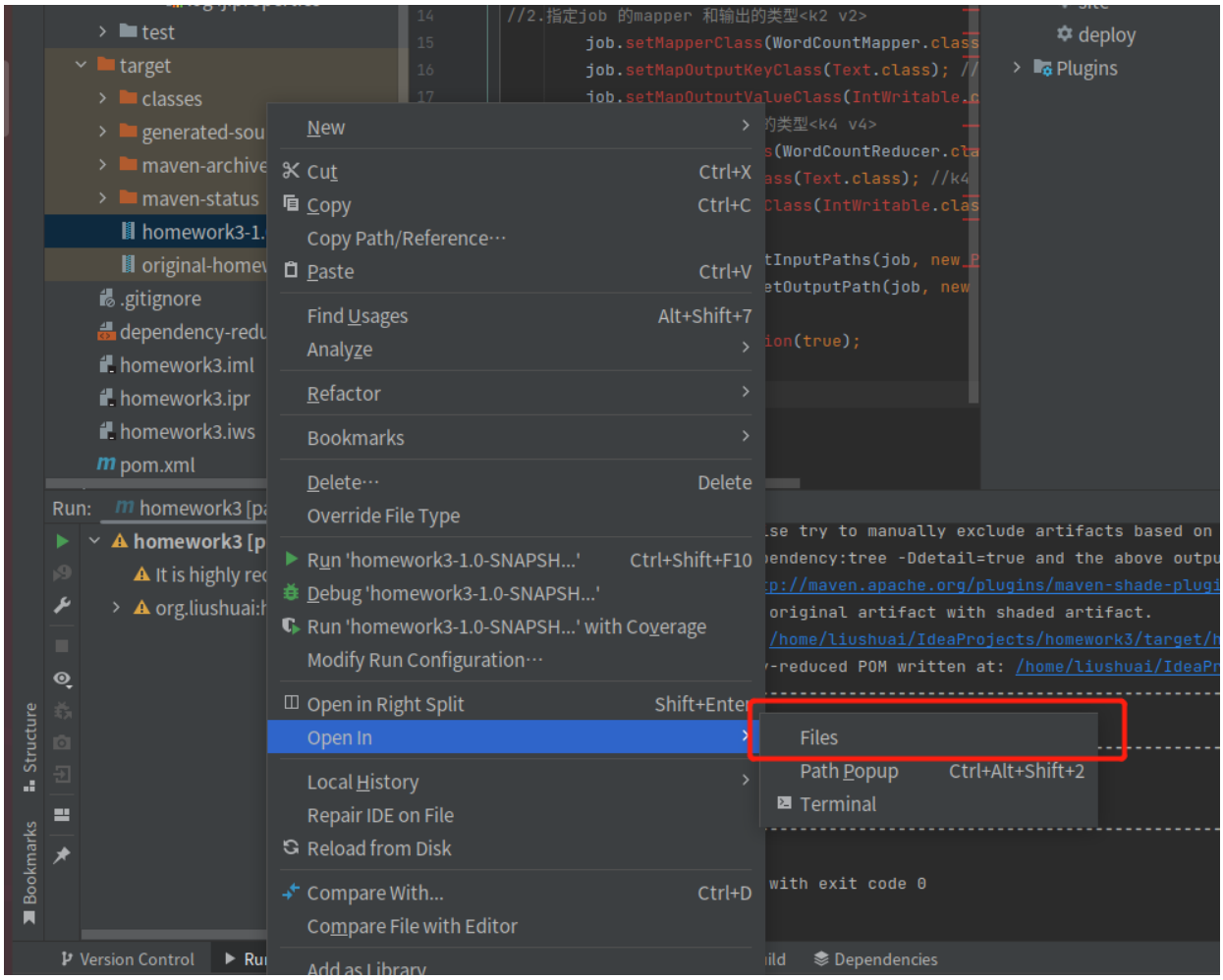
WordCountReducer是 MapReduce 作业中的 Reducer 组件，它负责对 Mapper 的输出进行合并和归约，并生成最终的输出。在词频统计的场景下，wordCountReducer 的任务是将相同单词的键值对进行合并，并将每个单词出现的次数累加，最终输出每个单词出现的总次数。

1.3 将程序打包成jar包

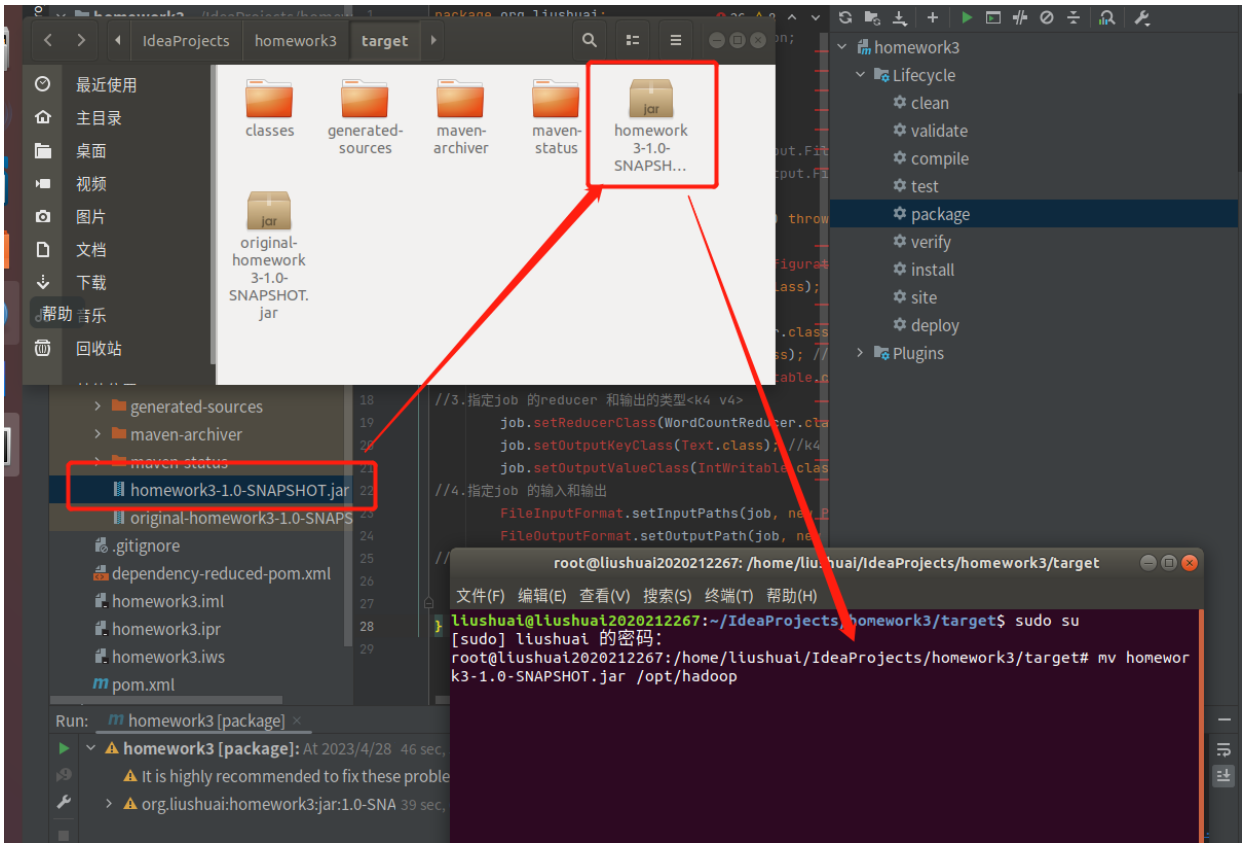
双击Lifecycle 中的package 进行打包



经过maven 的编译打包，即可在我们项目路径的target 目录下生成项目的jar 包。并把这个压缩包在文件系统中打开



可以看到，这个homework3的snapshot包存在于target目录下，我们需要将其移动至opt/hadoop



1.4 执行mapreduce任务，进行词频统计

首先启动hadoop程序，下载测试文件

```
root@liushuai2020212267:/opt/hadoop# start-yarn.sh
starting yarn daemons
starting resourcemanager, logging to /opt/hadoop/logs/yarn-root-resourcemanager-
liushuai2020212267.out
localhost: starting nodemanager, logging to /opt/hadoop/logs/yarn-root-nodemanag
er-liushuai2020212267.out
root@liushuai2020212267:/opt/hadoop# start-dfs.sh
Starting namenodes on [localhost]
localhost: starting namenode, logging to /opt/hadoop/logs/hadoop-root-namenode-l
iushuai2020212267.out
localhost: starting datanode, logging to /opt/hadoop/logs/hadoop-root-datanode-l
iushuai2020212267.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /opt/hadoop/logs/hadoop-root-sec
ondarynamenode-liushuai2020212267.out
root@liushuai2020212267:/opt/hadoop#
```

将测试文件上传至hdfs 的根目录下。

```
etc        lib                         logs          sbin
root@liushuai2020212267:/opt/hadoop# hadoop fs -put /opt/hadoop/ceshi.txt /
root@liushuai2020212267:/opt/hadoop# hadoop fs -ls /
Found 2 items
-rw-r--r--   1 root supergroup       6747 2023-04-28 16:07 /ceshi.txt
drwxr-xr-x   - root supergroup          0 2023-04-25 17:03 /hbase
root@liushuai2020212267:/opt/hadoop#
```

执行mapreduce任务，出现如下日志信息。

```
root@liushuai2020212267:/opt/hadoop# hadoop jar homework3-1.0-SNAPSHOT.jar /ceshi.txt /MRoutput
23/04/28 16:10:12 INFO client.RMProxy: Connecting to ResourceManager at localhost/127.0.0.1:8032
23/04/28 16:10:13 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface
 and execute your application with ToolRunner to remedy this.
23/04/28 16:10:15 INFO input.FileInputFormat: Total input files to process : 1
23/04/28 16:10:16 INFO mapreduce.JobSubmitter: number of splits:1
23/04/28 16:10:17 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1682669079602_0002
23/04/28 16:10:18 INFO conf.Configuration: resource-types.xml not found
23/04/28 16:10:18 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
23/04/28 16:10:18 INFO resource.ResourceUtils: Adding resource type - name = memory-mb, units = Mi, type = COUNTABLE
23/04/28 16:10:18 INFO resource.ResourceUtils: Adding resource type - name = vcores, units = , type = COUNTABLE
23/04/28 16:10:19 INFO impl.YarnClientImpl: Submitted application application_1682669079602_0002
23/04/28 16:10:19 INFO mapreduce.Job: The url to track the job: http://liushuai2020212267:8088/proxy/application_1682669079602_0002/
23/04/28 16:10:19 INFO mapreduce.Job: Running job: job_1682669079602_0002
23/04/28 16:10:47 INFO mapreduce.Job: Job job_1682669079602_0002 running in uber mode : false
23/04/28 16:10:47 INFO mapreduce.Job:  map 0% reduce 0%
23/04/28 16:11:00 INFO mapreduce.Job:  map 100% reduce 0%
23/04/28 16:11:14 INFO mapreduce.Job:  map 100% reduce 100%
23/04/28 16:11:15 INFO mapreduce.Job: Job job_1682669079602_0002 completed successfully
23/04/28 16:11:16 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=18754
                FILE: Number of bytes written=454361
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=6843
                HDFS: Number of bytes written=44
                HDFS: Number of read operations=6
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=10349
                Total time spent by all reduces in occupied slots (ms)=10374
                Total time spent by all map tasks (ms)=10349
                Total time spent by all reduce tasks (ms)=10374
                Total vcore-milliseconds taken by all map tasks=10349
                Total vcore-milliseconds taken by all reduce tasks=10374
                Total megabyte-milliseconds taken by all map tasks=10597376
                Total megabyte-milliseconds taken by all reduce tasks=10622976
        Map-Reduce Framework
                Map input records=1
                Map output records=2000
                Map output bytes=14748
                Map output materialized bytes=18754
                Input split bytes=96
                Combine input records=0
                Combine output records=0
                Reduce input groups=6
                Reduce shuffle bytes=18754
                Reduce input records=2000
                Reduce output records=6
                Spilled Records=4000
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=622
                CPU time spent (ms)=4040
                Physical memory (bytes) snapshot=495661056
```

终端

INFO mapreduce.Job: map 0% reduce 0%表示作业刚开始执行时，Map 和 Reduce 阶段的进度都为 0%。

INFO mapreduce.Job: map 100% reduce 0%表示 Map 阶段已经完成，进度为 100%，而 Reduce 阶段还没有开始执行，进度仍然为 0%。这说明 Map 阶段已经将输入数据全部处理完毕，并将结果输出到 HDFS 中的临时文件中。

INFO mapreduce.Job: map 100% reduce 100% 表示整个作业已经完成，Map 和 Reduce 阶段的进度都已经达到了 100%。这说明 Reduce 阶段已经将 Map 阶段输出的临时结果进行了合并、排序等操作，并将最终结果输出到了指定的目录中。

1.5 查看执行命令后输出的结果

输出结果在hdfs 的MRoutput 文件夹下

```
                Bytes Written=44
root@liushuai2020212267:/opt/hadoop# hadoop fs -ls /
Found 4 items
drwxr-xr-x   - root supergroup          0 2023-04-28 16:11 /MRoutput
-rw-r--r--   1 root supergroup       6747 2023-04-28 16:07 /ceshi.txt
drwxr-xr-x   - root supergroup          0 2023-04-25 17:03 /hbase
drwx------   - root supergroup          0 2023-04-28 16:08 /tmp
root@liushuai2020212267:/opt/hadoop#
```

查看结果：

```
hadoop fs -cat /MRoutput/part-r-00000
```

```
root@liushuai2020212267:/opt/hadoop# hadoop fs -ls /
Found 4 items
drwxr-xr-x   - root supergroup          0 2023-04-28 16:11 /MRoutput
-rw-r--r--   1 root supergroup       6747 2023-04-28 16:07 /ceshi.txt
drwxr-xr-x   - root supergroup          0 2023-04-25 17:03 /hbase
drwx------   - root supergroup          0 2023-04-28 16:08 /tmp
root@liushuai2020212267:/opt/hadoop# hadoop fs -cat /MRoutput/part-r-00000
a       346
b       325
c       335
d       307
hello   346
world   341
root@liushuai2020212267:/opt/hadoop#
```

二、Spark 伪分布环境配置

2.1 安装scala并解压

```
root@liushuai2020212267:/opt/hadoop# cd ~
root@liushuai2020212267:~# wget https://downloads.lightbend.com/scala/2.12.15/scala-2.12.15.tgz
--2023-04-28 16:14:15--  https://downloads.lightbend.com/scala/2.12.15/scala-2.12.15.tgz
正在解析主机 downloads.lightbend.com (downloads.lightbend.com)... 54.192.18.53, 54.192.18.34, 54.192.18.74, ...
正在连接 downloads.lightbend.com (downloads.lightbend.com)|54.192.18.53|:443... 已连接。
已发出 HTTP 请求，正在等待回应... 200 OK
长度： 21087658 (20M) [application/octet-stream]
正在保存至：“scala-2.12.15.tgz”

scala-2.12.15.tgz   100%[===================>]  20.11M  5.81MB/s    用时 3.5s

2023-04-28 16:14:20 (5.81 MB/s) - 已保存 “scala-2.12.15.tgz” [21087658/21087658])

root@liushuai2020212267:~# tar -zxvf scala-2.12.15.tgz scala-2.12.15/
scala-2.12.15/
scala-2.12.15/lib/
scala-2.12.15/lib/scala-compiler.jar
scala-2.12.15/lib/scalap-2.12.15.jar
scala-2.12.15/lib/scala-reflect.jar
scala-2.12.15/lib/scala-xml_2.12-1.0.6.jar
scala-2.12.15/lib/jline-2.14.6.jar
scala-2.12.15/lib/scala-parser-combinators_2.12-1.0.7.jar
scala-2.12.15/lib/scala-swing_2.12-2.0.3.jar
scala-2.12.15/lib/scala-library.jar
scala-2.12.15/doc/
scala-2.12.15/doc/licenses/
scala-2.12.15/doc/licenses/mit_jquery.txt
scala-2.12.15/doc/licenses/bsd_asm.txt
scala-2.12.15/doc/licenses/apache_jansi.txt
scala-2.12.15/doc/licenses/bsd_jline.txt
scala-2.12.15/doc/licenses/mit_tools.tooltip.txt
scala-2.12.15/doc/LICENSE.md
scala-2.12.15/doc/License.rtf
scala-2.12.15/doc/README
scala-2.12.15/doc/tools/
scala-2.12.15/doc/tools/scaladoc.html
scala-2.12.15/doc/tools/scalap.html
scala-2.12.15/doc/tools/css/
scala-2.12.15/doc/tools/css/style.css
```

将scala移动至usr目录，并更改环境变量，执行

```
scala -version
```

查看scala是否安装成功

```
root@liushuai2020212267:~# mv scala-2.12.15 /usr/scala-2.12.15
root@liushuai2020212267:~# cd /usr
root@liushuai2020212267:/usr# ls
bin    include  lib      local  scala-2.12.15  src
games  java8    libexec  sbin   share
root@liushuai2020212267:/usr# cd scala-2.12.15/
root@liushuai2020212267:/usr/scala-2.12.15# ls
bin  doc  lib  LICENSE  man  NOTICE
root@liushuai2020212267:/usr/scala-2.12.15#
```

```
root@liushuai2020212267:/usr/scala-2.12.15# vim /etc/profile
root@liushuai2020212267:/usr/scala-2.12.15# source /etc/profile
root@liushuai2020212267:/usr/scala-2.12.15# scala -version
Scala code runner version 2.12.15 -- Copyright 2002-2021, LAMP/EPFL and Lightben
d, Inc.
root@liushuai2020212267:/usr/scala-2.12.15#
```

2.2 安装spark

安装spark，并对文件进行解压，移动至/opt目录中

```
root@liushuai2020212267:~# ^C
root@liushuai2020212267:~# tar -zxf spark-3.2.3-bin-without-hadoop.tgz
root@liushuai2020212267:~# ls
openjdk-8u41-b04-linux-x64-14_jan_2020.tar.gz
scala-2.12.15.tgz
snap
spark-3.2.3-bin-without-hadoop
spark-3.2.3-bin-without-hadoop.tgz
tmp
root@liushuai2020212267:~# mv spark-3.2.3-bin-without-hadoop /opt/spark
root@liushuai2020212267:~#
```

至此，hadoop,hbase,spark的源文件全部置于/opt文件中

```
root@liushuai2020212267:~# cd /opt
root@liushuai2020212267:/opt# ls
hadoop  hbase  spark  vmware-tools-distrib
root@liushuai2020212267:/opt#
```

2.3 配置环境变量

```
fi
export JAVA_HOME=/usr/java8
export PATH=$PATH:$JAVA_HOME/bin
export HADOOP_HOME=/opt/hadoop/
export PATH=$PATH:$HADOOP_HOME/bin
export HBASE_HOME=/opt/hbase
export PATH=$PATH:$HBASE_HOME/bin
export PATH=$PATH:$HADOOP_HOME/sbin
export PATH=~/apache-maven-3.8.8/bin:$PATH
export SCALA_HOME=/usr/scala-2.12.15
export PATH=$PATH:$SCALA_HOME/bin
export SPARK_HOME=/opt/spark
export PATH=$PATH:$SPARK_HOME/bin
```

2.4 配置spark文件

```
root@liushuai2020212267:/opt# vi /etc/profile
root@liushuai2020212267:/opt# source /etc/profile
root@liushuai2020212267:/opt# cp /opt/spark/conf/spark-env.sh.template /opt/spar
k/conf/spark-env.sh
root@liushuai2020212267:/opt# vi /opt/spark/conf/spark-env.sh
```

配置spark-env.sh文件

```
# - OPENBLAS_NUM_THREADS=1   Disable multi-threading of OpenBLAS
export JAVA_HOME=/usr/java8
export SCALA_HOME=/usr/scala-2.12.15
export SPARK_DIST_CLASSPATH=$(/opt/hadoop/bin/hadoop classpath)
export HADOOP_CONF_DIR=/opt/hadoop/etc/hadoop
export SPARK_MASTER_IP=192.168.1.14
export SPARK_MASTER_PORT=7077
export SPARK_MASTER_WEBUI_PORT=8080
:wq
```

启动spark

```
cd /opt/spark
sbin/start-all.sh
```

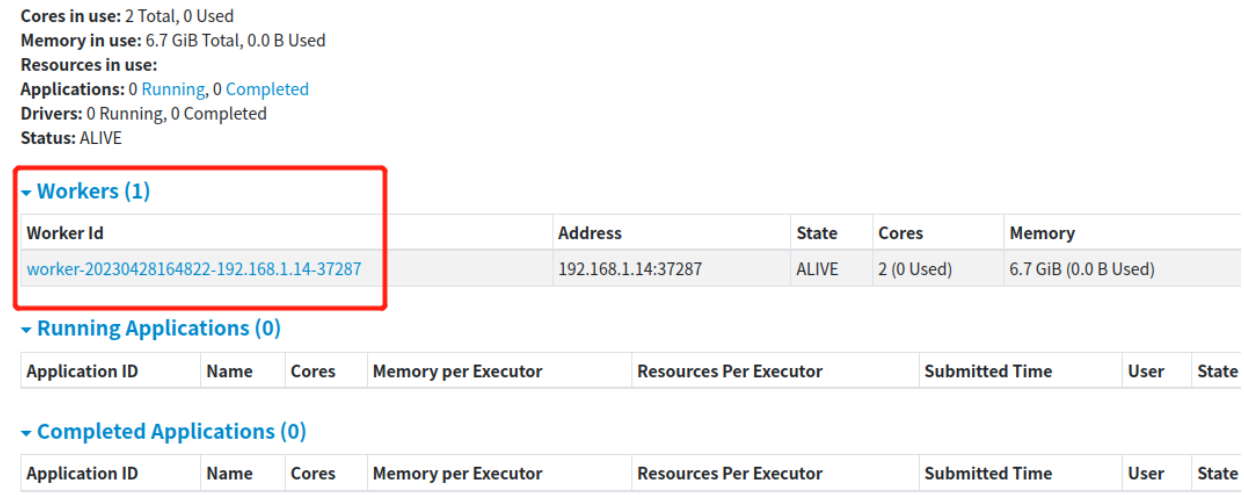
执行jps命令

```
root@liushuai2020212267:/opt/spark# sbin/start-all.sh
starting org.apache.spark.deploy.master.Master, logging to /opt/spark/logs/spark
-root-org.apache.spark.deploy.master.Master-1-liushuai2020212267.out
localhost: starting org.apache.spark.deploy.worker.Worker, logging to /opt/spark
/logs/spark-root-org.apache.spark.deploy.worker.Worker-1-liushuai2020212267.out
root@liushuai2020212267:/opt/spark# jps
3410 ResourceManager
4418 SecondaryNameNode
3570 NodeManager
2163 -- process information unavailable
2756 -- process information unavailable
7878 Worker
3862 NameNode
2870 -- process information unavailable
3994 DataNode
7981 Jps
7742 Master
root@liushuai2020212267:/opt/spark#
```

出现worker和master两进程，其中，masker进程负责管理整个 Spark 集群的资源和作业调度等工作。而worker进程主要负责在本地节点上启动和管理 Spark Executor 进程，执行具体的计算任务。

2.5 前端可视化

打开localhost:8080, web 界面显示有一个节点，配置成功



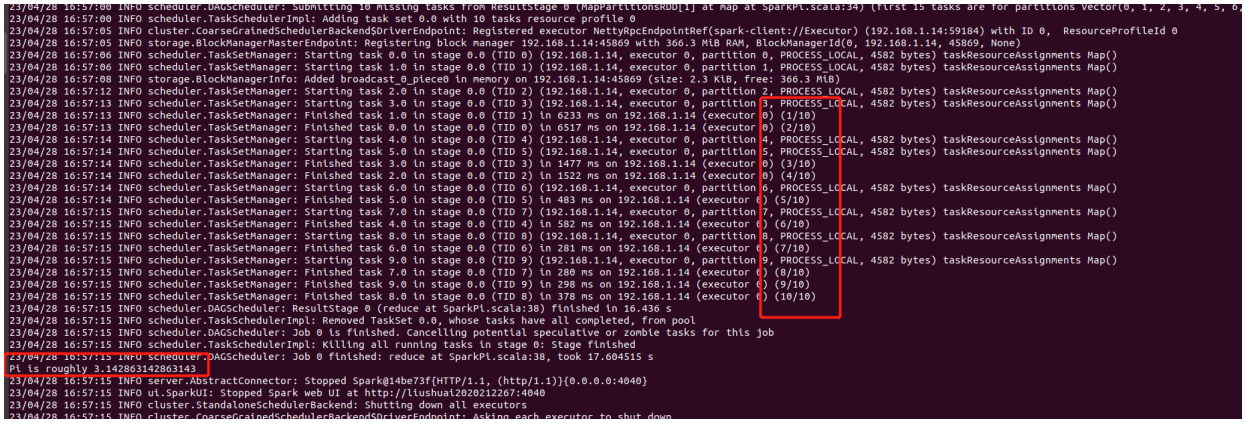
2.6 用自带样例进行测试

```
bin/spark-submit
--master spark://主机名:7077
--class org.apache.spark.examples.SparkPi
/opt/spark/examples/jars/spark-examples_2.12-3.2.3.jar 10
```

在 Spark 集群上提交 Spark 应用程序的命令，其中，

- `bin/spark-submit` 是提交 Spark 应用程序的命令。
- `--master spark://主机名:7077` 指定 Spark 应用程序将运行在名为 "主机名" 的 Spark 主节点上，并且主节点的端口为 7077。
- `--class org.apache.spark.examples.SparkPi` 指定 Spark 应用程序的入口类为 `org.apache.spark.examples.SparkPi`。
- `/opt/spark/examples/jars/spark-examples_2.12-3.2.3.jar` 是 Spark 应用程序的 JAR 文件路径。
- `10` 是 SparkPi 应用程序的参数，表示计算 Pi 的精度。

该命令将在 Spark 集群上提交一个名为 `SparkPi` 的应用程序，该应用程序将计算 Pi 的值，并将结果返回给驱动程序。



2.7 参数信息的解释：

- `storage.BlockManagerInfo`：表示在 BlockManager 中添加了一个大小为 2.3 KiB 的广播变量。
- `scheduler.TaskSetManager`：表示应用程序的任务集管理器，包括启动和完成任务的信息，以及任务的详细信息，如任务在哪个节点上执行、任务的资源分配等。

三、本地运行wordcount程序

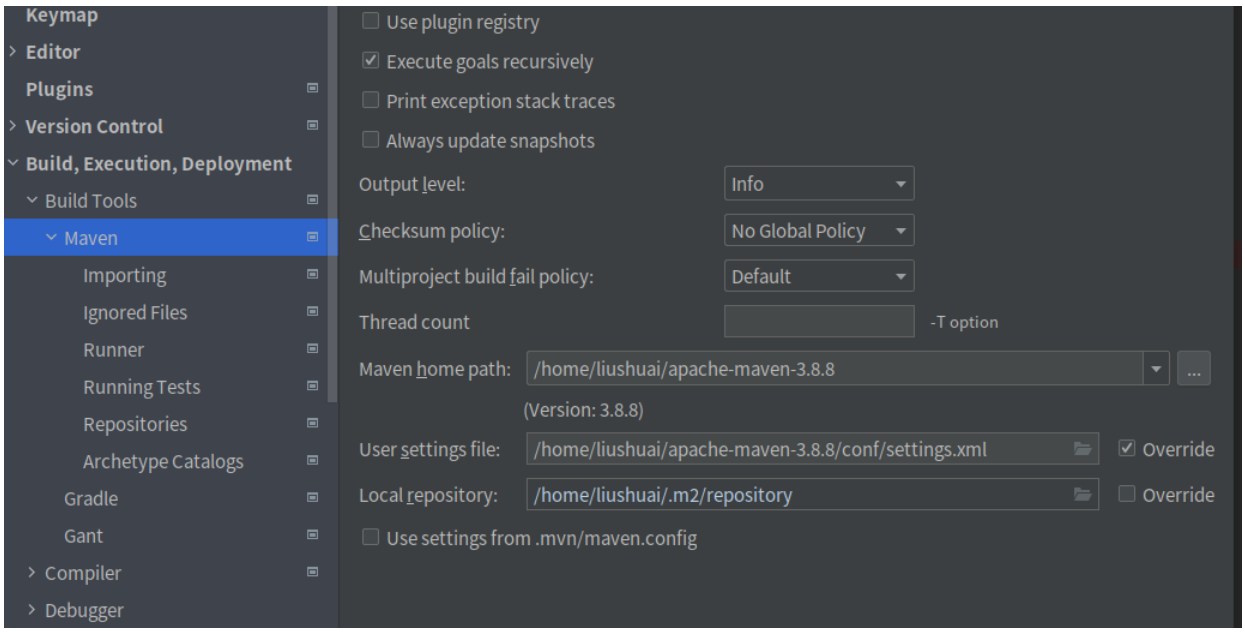
创建maven工程，并进行配置，同时更改importing和runner设置，在IDEA中设置maven编译时忽略HTTPS的SSL证书验证

在importing中添加

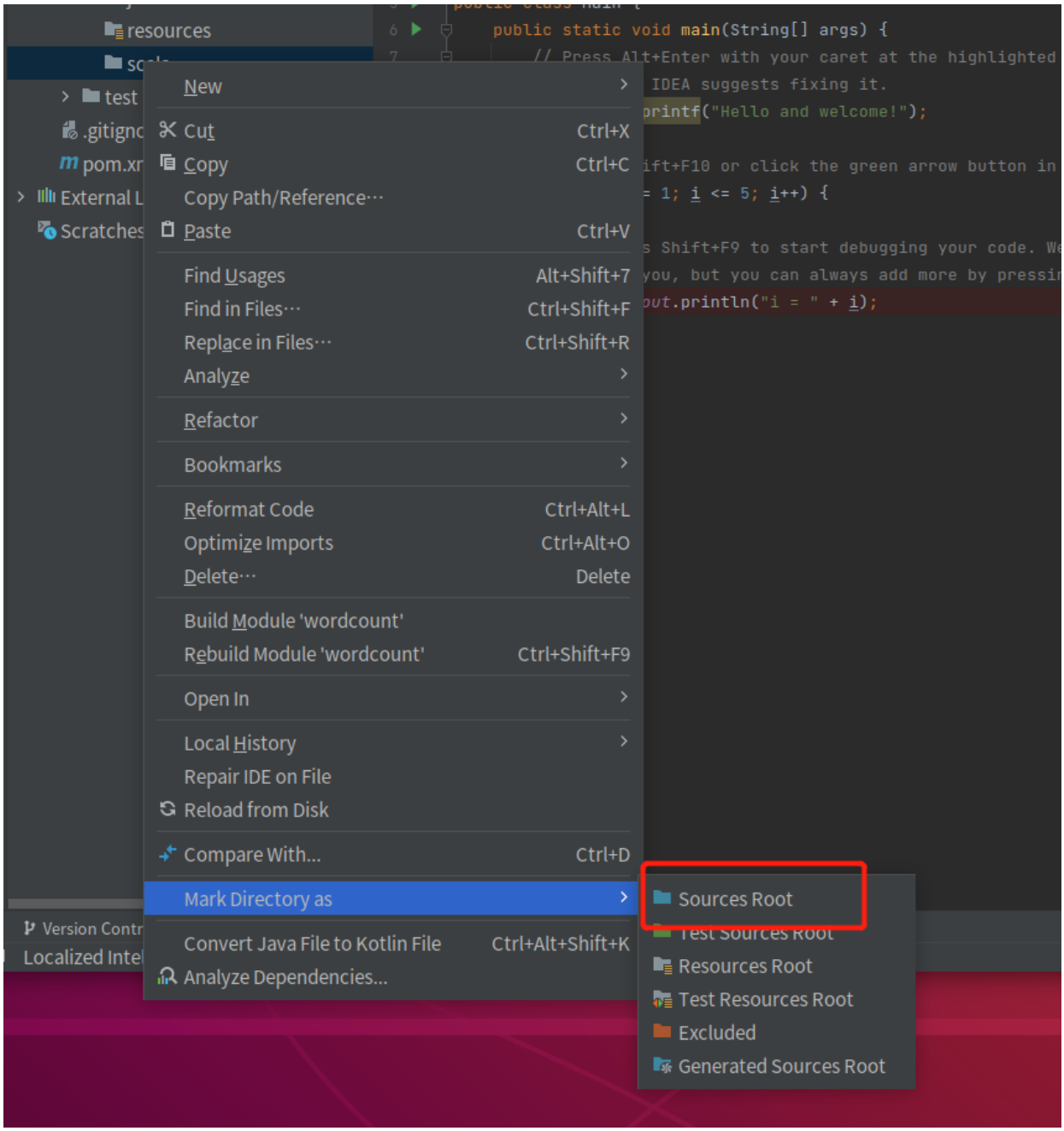
```
-Dmaven.wagon.http.ssl.insecure=true -Dmaven.wagon.http.ssl.allowall=true
```

在runner中添加

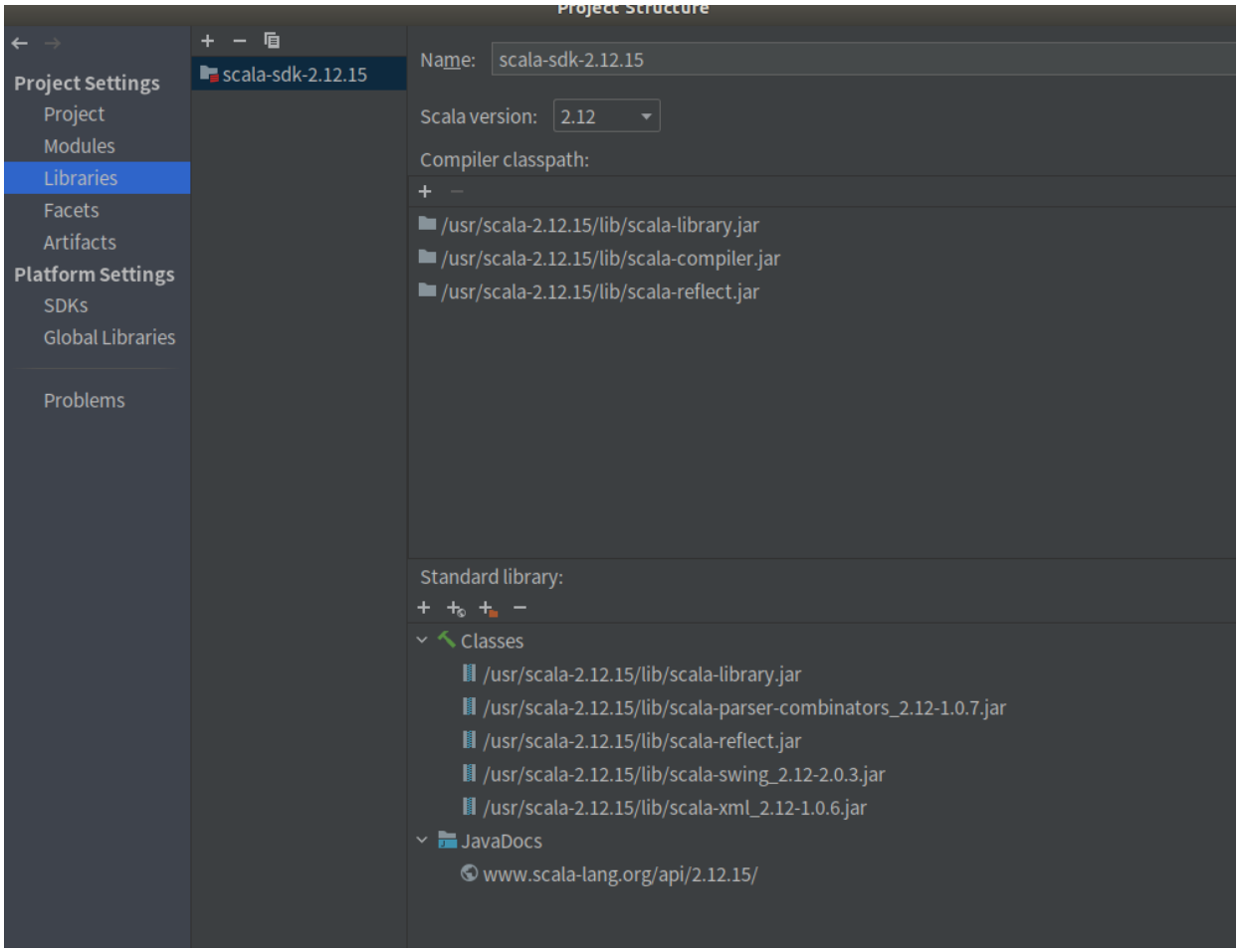
```
-Dmaven.wagon.http.ssl.insecure=true -Dmaven.wagon.http.ssl.allowall=true
-Dmaven.wagon.http.ssl.ignore.validity.dates=true
```



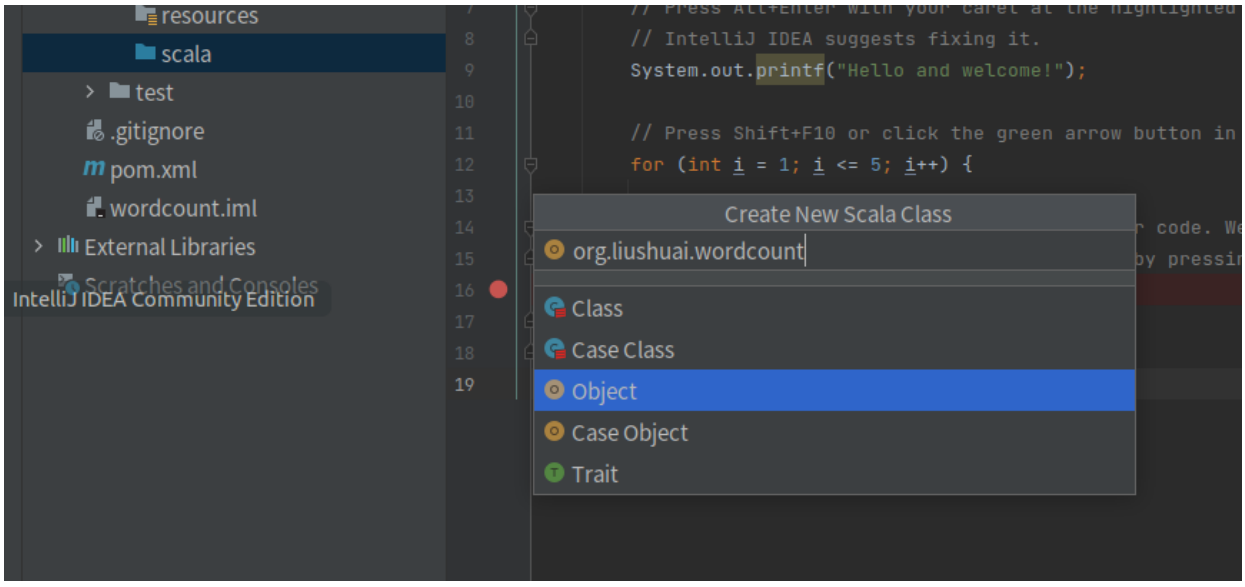
创建scala目录，并将其设置为sources root



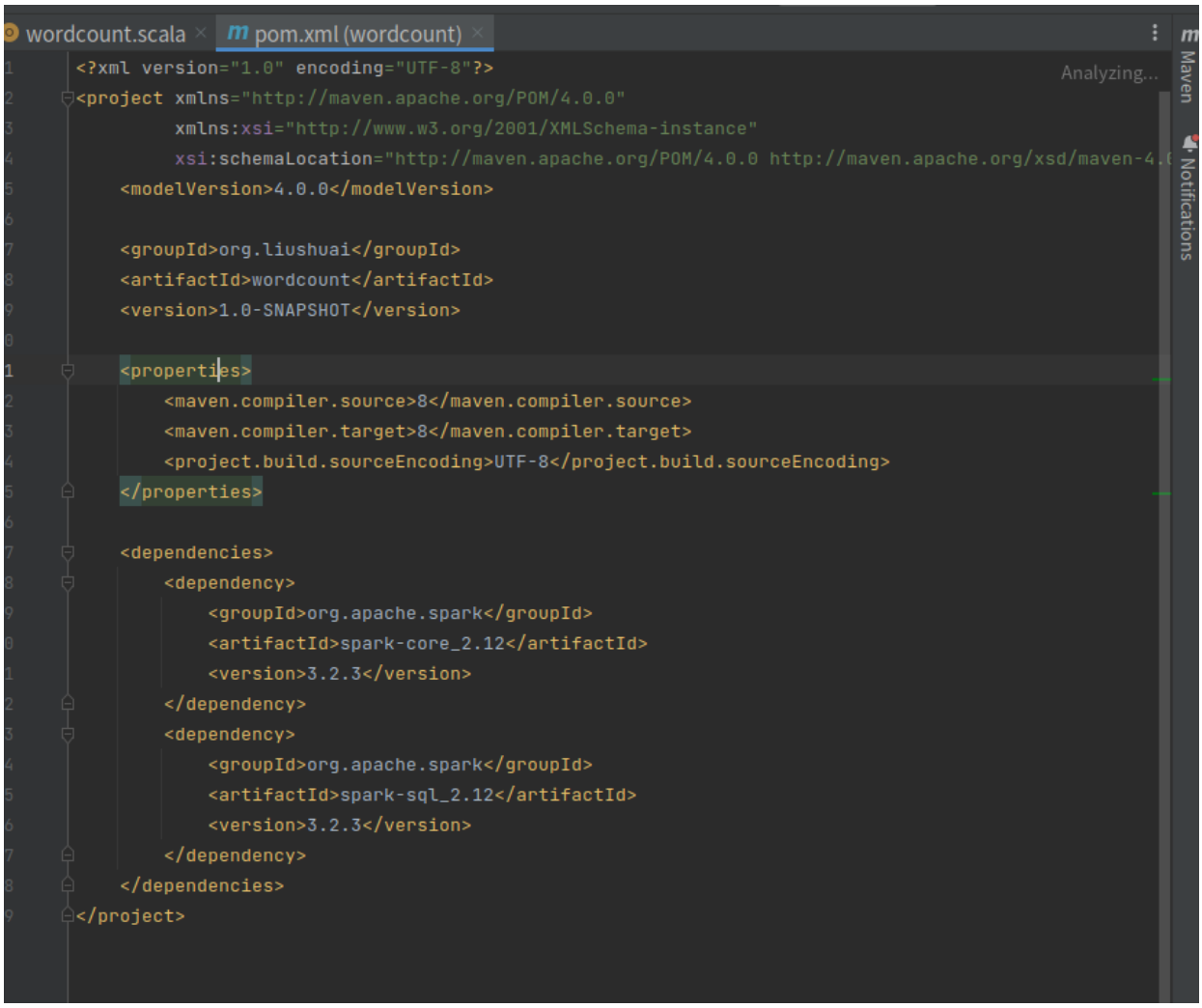
添加scala library，首先需要在plugins中下载scala插件，这样才能在libraries中找到scala-sdk



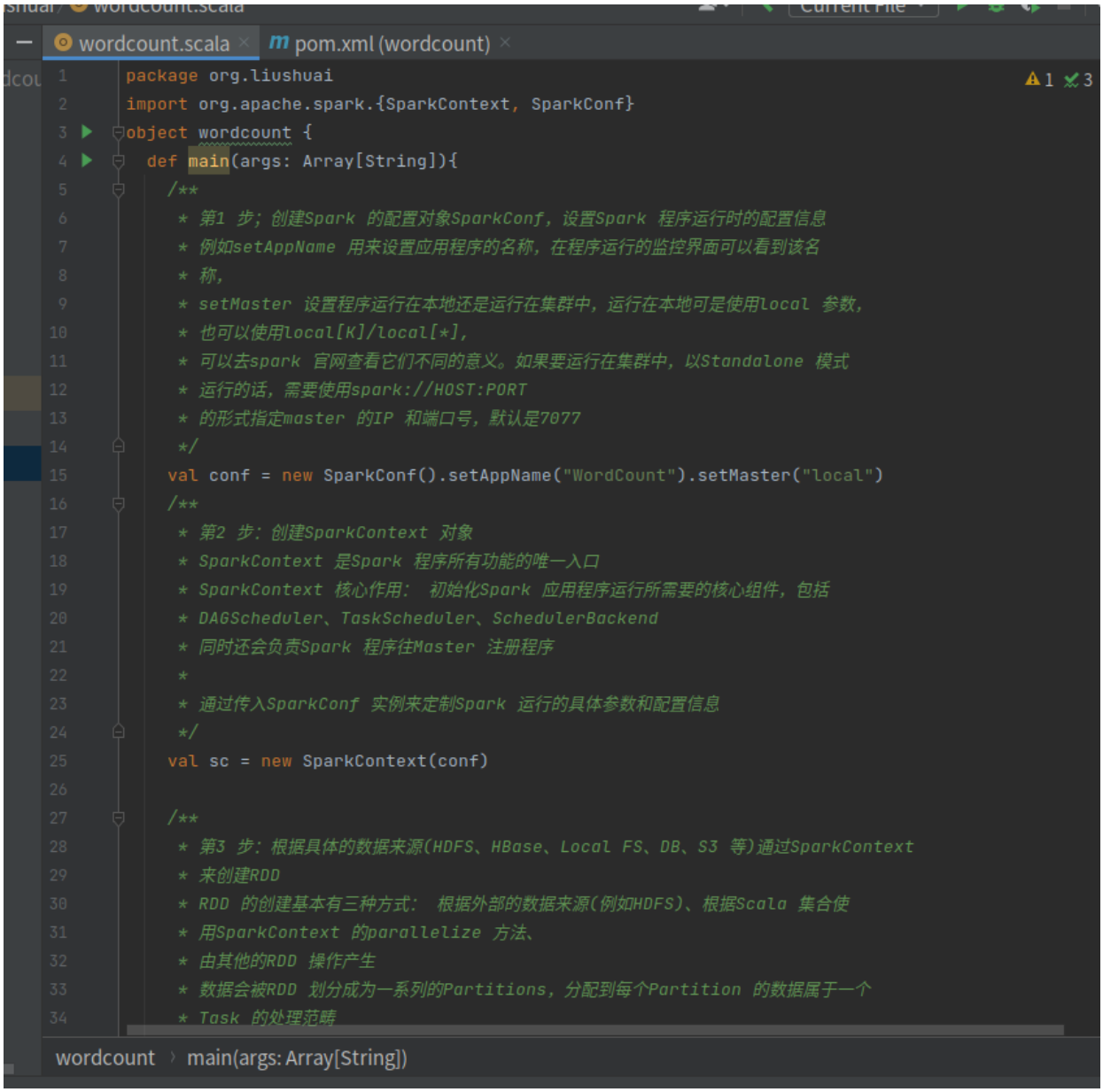
新建scala class



设置pom.xml文件，导入spark依赖

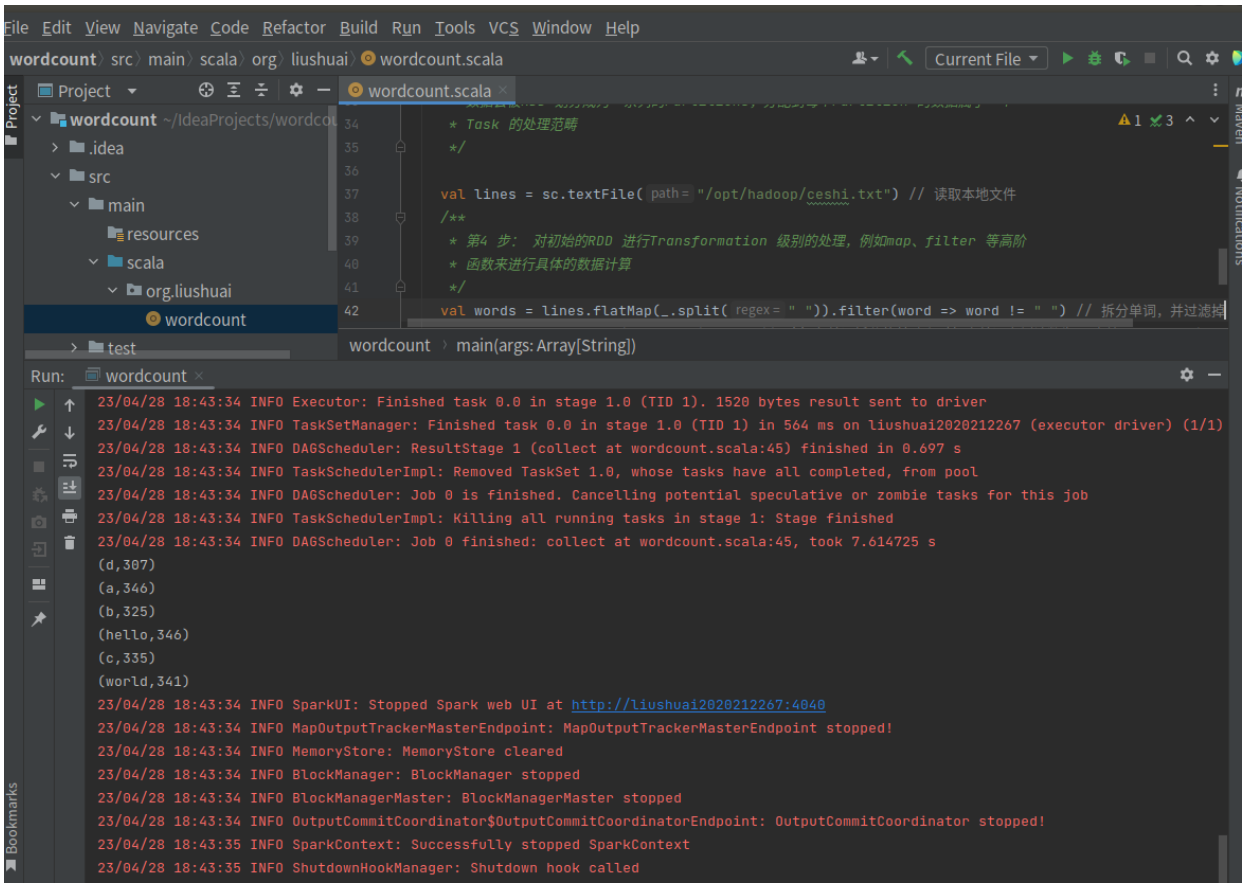

wordcount.scala
pom.xml (wordcount)
Analyzing...
Maven
Notifications
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.
<modelVersion>4.0.0</modelVersion>
<groupId>org.liushuai</groupId>
<artifactId>wordcount</artifactId>
<version>1.0-SNAPSHOT</version>
<properties>
<maven.compiler.source>8</maven.compiler.source>
<maven.compiler.target>8</maven.compiler.target>
<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>
<dependencies>
<dependency>
<groupId>org.apache.spark</groupId>
<artifactId>spark-core_2.12</artifactId>
<version>3.2.3</version>
</dependency>
<dependency>
<groupId>org.apache.spark</groupId>
<artifactId>spark-sql_2.12</artifactId>
<version>3.2.3</version>
</dependency>
</dependencies>
</project>

编写scala程序：



将测试文件上传到/opt/hadoop/ceshi.txt中，通过scala类进行读取并计算

在idea中运行wordcount程序，并查看运行结果



四、性能分析

mapruduce方法和spark方法实现wordcount功能的性能分析如下：

- 实现方法

  MapReduce 的实现通常需要编写 Map 和 Reduce 函数，并将代码打包成可执行的 Jar 包提交到 Hadoop 集群上运行。而 Spark 的实现则可以使用 Spark 提供的 RDD 和高级 API，通过编写 Scala、Python 或 Java 代码来实现。

- 性能表现

  Spark 相对于 MapReduce 有一些优势。首先，Spark 的计算模型是基于内存的，可以将数据存储在内存中进行高效的计算，而 MapReduce 的计算模型则是基于磁盘的，需要将数据从磁盘中读取到内存中进行计算，因此 Spark 的计算速度通常要快于 MapReduce。其次，Spark 提供了更丰富的数据处理和转换操作，可以方便地进行数据清洗、过滤、排序、聚合等操作，相对而言更加灵活和方便。

- 大规模场景处理

  对于大规模数据集的处理，MapReduce 效果更优，因为它可以通过横向扩展来处理大规模数据集，而 Spark 在处理大规模数据集时需要考虑内存的限制和数据分区等问题。此外，MapReduce 也可以运行在各种 Hadoop 生态系统中，包括 HDFS、YARN、HBase 等等，具有更广泛的适用性。