

Spark-IoT-Analysis

This repository contains the implementation of a Spark application designed to analyze IoT data samples. It includes code for loading data into HDFS, analyzing it using Spark Shell and Java, and instructions on setting up Spark, compiling, and running the application with Maven and spark-submit.

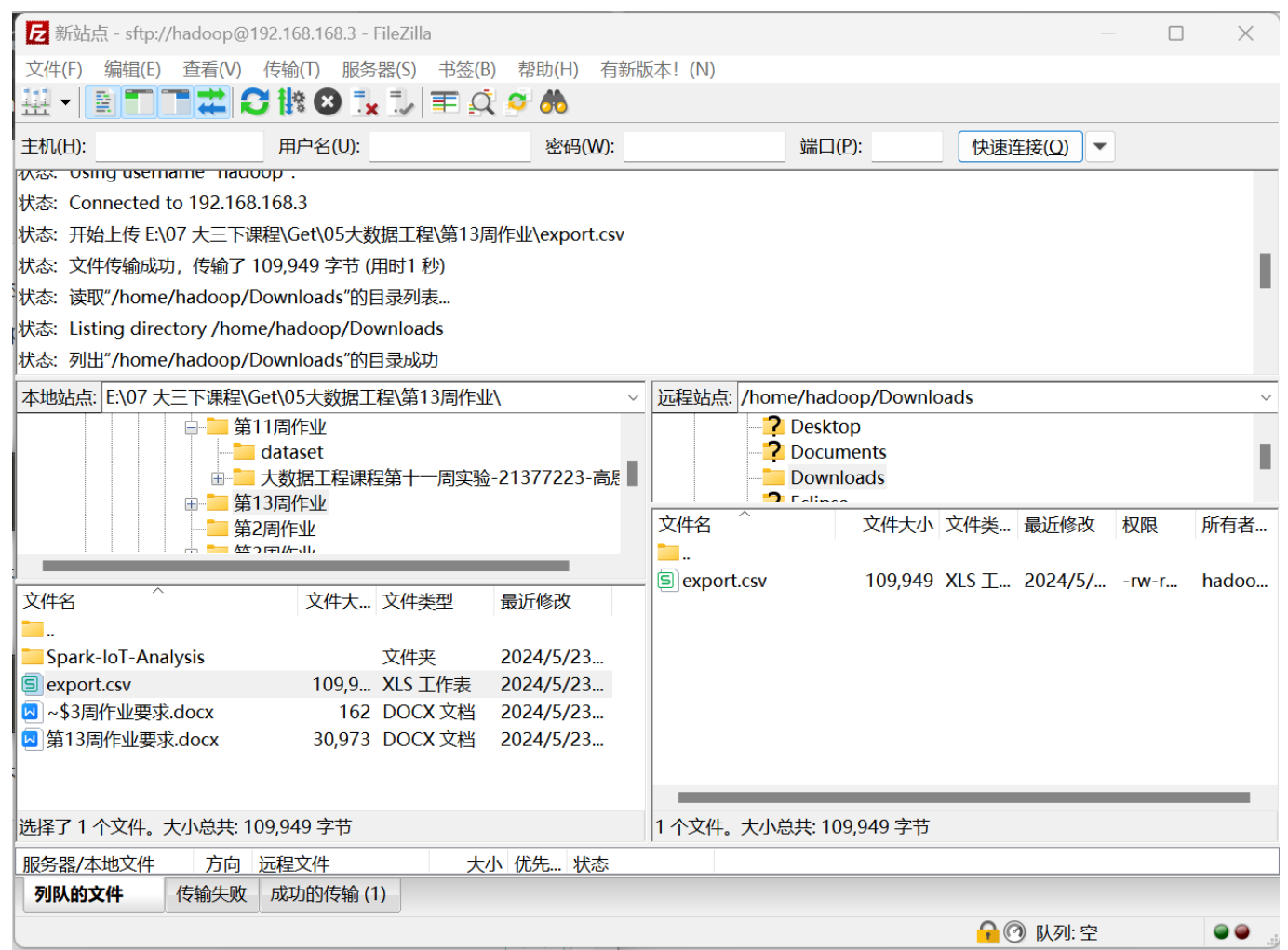
任务说明

在本地机器上安装和部署Spark，利用Spark Shell和Java编写程序，对IOT设备收集的样本数据进行数据统计分析。首先，使用Hadoop Shell命令将数据加载至本地HDFS，然后在Spark平台上进行操作。使用Spark Shell和Java语言进行编程，统计数据文件的行数，并使用Maven编译打包Java应用程序，通过spark-submit运行程序。

具体代码实现

0 Hadoop加载数据文件

下载数据文件，并传至虚拟机：



启动hadoop服务：

```
cd /usr/local/hadoop
./sbin/start-dfs.sh #启动hadoop
```

```
jps
```

```
hadoop@UbuntuGetData: /usr/local/hadoop
hadoop@UbuntuGetData:~/Desktop$ cd /usr/local/hadoop
hadoop@UbuntuGetData:/usr/local/hadoop$ ./sbin/start-dfs.sh
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [UbuntuGetData]
hadoop@UbuntuGetData:/usr/local/hadoop$ jps
2931 DataNode
3143 SecondaryNameNode
2809 NameNode
3258 Jps
hadoop@UbuntuGetData:/usr/local/hadoop$
```

创建工作目录，并将数据文件传入工作目录中：

```
./bin/hdfs dfs -ls /
./bin/hdfs dfs -mkdir /user/spark
./bin/hdfs dfs -put /home/hadoop/Downloads/export.csv /user/spark
```

```
hadoop@UbuntuGetData: /usr/local/hadoop
hadoop@UbuntuGetData:/usr/local/hadoop$ ./bin/hdfs dfs -ls /
Found 6 items
drwxr-xr-x - hadoop supergroup 0 2024-03-30 15:01 /hbase
drwxr-xr-x - hadoop supergroup 0 2024-03-07 15:52 /input
drwxr-xr-x - hadoop supergroup 0 2024-03-07 15:54 /output
drwx-wx-wx - hadoop supergroup 0 2024-05-11 20:26 /tmp
drwxr-xr-x - hadoop supergroup 0 2024-05-11 22:05 /user
drwxr-xr-x - hadoop supergroup 0 2024-03-30 17:31 /usr
hadoop@UbuntuGetData:/usr/local/hadoop$ ./bin/hdfs dfs -ls /usr
Found 1 items
drwxr-xr-x - hadoop supergroup 0 2024-03-30 17:31 /usr/local
hadoop@UbuntuGetData:/usr/local/hadoop$ ./bin/hdfs dfs -ls /user
Found 2 items
drwxr-xr-x - hadoop supergroup 0 2024-03-13 19:47 /user/hadoop
drwxr-xr-x - hadoop supergroup 0 2024-05-11 22:12 /user/hive
hadoop@UbuntuGetData:/usr/local/hadoop$ ./bin/hdfs dfs -mkdir /user/spark
hadoop@UbuntuGetData:/usr/local/hadoop$ ./bin/hdfs dfs -ls /user
Found 3 items
drwxr-xr-x - hadoop supergroup 0 2024-03-13 19:47 /user/hadoop
drwxr-xr-x - hadoop supergroup 0 2024-05-11 22:12 /user/hive
drwxr-xr-x - hadoop supergroup 0 2024-05-23 11:17 /user/spark
hadoop@UbuntuGetData:/usr/local/hadoop$ ./bin/hdfs dfs -put /home/hadoop/Downloads/export
.csv /user/spark
2024-05-23 11:20:36,317 INFO sasl.SaslDataTransferClient: SASL encryption trust check: lo
calHostTrusted = false, remoteHostTrusted = false
hadoop@UbuntuGetData:/usr/local/hadoop$ ./bin/hdfs dfs -ls /user/spark
Found 1 items
-rw-r--r-- 1 hadoop supergroup 109949 2024-05-23 11:20 /user/spark/export.csv
hadoop@UbuntuGetData:/usr/local/hadoop$
```

传输完成！

1 安装部署Spark

首先需要下载Spark安装文件。访问[Spark官方下载地址](#)，按照如下图下载：

Download Apache Spark™

1. Choose a Spark release: 3.5.1 (Feb 23 2024) ▼

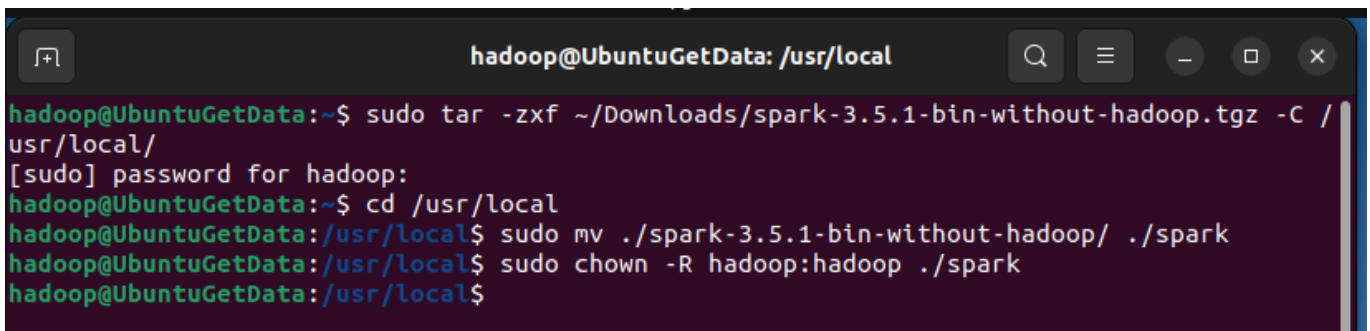
2. Choose a package type:

Pre-built with user-provided Apache Hadoop ▼

3. Download Spark: [spark-3.5.1-bin-without-hadoop.tgz](#)

安装Spark（Local模式），执行以下命令进行解压安装：

```
sudo tar -zxf ~/Downloads/spark-3.5.1-bin-without-hadoop.tgz -C /usr/local/  
cd /usr/local  
sudo mv ./spark-3.5.1-bin-without-hadoop/ ./spark  
sudo chown -R hadoop:hadoop ./spark
```



```
hadoop@UbuntuGetData: /usr/local  
hadoop@UbuntuGetData:~$ sudo tar -zxf ~/Downloads/spark-3.5.1-bin-without-hadoop.tgz -C /usr/local/  
[sudo] password for hadoop:  
hadoop@UbuntuGetData:~$ cd /usr/local  
hadoop@UbuntuGetData:/usr/local$ sudo mv ./spark-3.5.1-bin-without-hadoop/ ./spark  
hadoop@UbuntuGetData:/usr/local$ sudo chown -R hadoop:hadoop ./spark  
hadoop@UbuntuGetData:/usr/local$
```

安装后，还需要修改Spark的配置文件spark-env.sh

```
cd /usr/local/spark  
cp ./conf/spark-env.sh.template ./conf/spark-env.sh
```

编辑spark-env.sh文件(vim ./conf/spark-env.sh)，在第一行添加以下配置信息：

```
export SPARK_DIST_CLASSPATH=$(/usr/local/hadoop/bin/hadoop classpath)
```

```
hadoop@UbuntuGetData: /usr/local/spark
#!/usr/bin/env bash
export SPARK_DIST_CLASSPATH=$(/usr/local/hadoop/bin/hadoop classpath)
#
# Licensed to the Apache Software Foundation (ASF) under one or more
# contributor license agreements. See the NOTICE file distributed with
# this work for additional information regarding copyright ownership.
# The ASF licenses this file to You under the Apache License, Version 2.0
# (the "License"); you may not use this file except in compliance with
# the License. You may obtain a copy of the License at
#
```

配置完成后就可以直接使用，不需要像Hadoop运行启动命令。

通过运行Spark自带的示例，验证Spark是否安装成功。

```
cd /usr/local/spark
bin/run-example SparkPi
```

```
hadoop@UbuntuGetData: /usr/local/spark
hadoop@UbuntuGetData: /usr/local/spark$ cd /usr/local/spark
hadoop@UbuntuGetData: /usr/local/spark$ bin/run-example SparkPi
Pi is roughly 3.1373356866784334
hadoop@UbuntuGetData: /usr/local/spark$
```

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）：

```
cd /usr/local/spark
bin/run-example SparkPi 2>&1 | grep "Pi is"
```

```
hadoop@UbuntuGetData: /usr/local/spark
hadoop@UbuntuGetData: /usr/local/spark$ cd /usr/local/spark
hadoop@UbuntuGetData: /usr/local/spark$ bin/run-example SparkPi
Pi is roughly 3.1373356866784334
hadoop@UbuntuGetData: /usr/local/spark$ bin/run-example SparkPi 2>&1 | grep "Pi is"
Pi is roughly 3.150955754778774
hadoop@UbuntuGetData: /usr/local/spark$
```

2 Spark Shell统计文件行数

首先启动spark shell

```
cd /usr/local/spark
bin/spark-shell
```

```
hadoop@UbuntuGetData: /usr/local/spark
hadoop@UbuntuGetData: /usr/local/spark$ cd /usr/local/spark
hadoop@UbuntuGetData: /usr/local/spark$ bin/run-example SparkPi
Pi is roughly 3.1373356866784334
hadoop@UbuntuGetData: /usr/local/spark$ bin/run-example SparkPi 2>&1 | grep "Pi is"
Pi is roughly 3.150955754778774
hadoop@UbuntuGetData: /usr/local/spark$ bin/spark-shell
Spark context Web UI available at http://10.0.2.15:4040
Spark context available as 'sc' (master = local[*], app id = local-1716435727699).
Spark session available as 'spark'.
Welcome to

  ____  _
 / ___|| | | |
| |___| |_| |
 \___ \|  __/
    ___| | | |
   |___|_|_|_|

version 3.5.1

Using Scala version 2.12.18 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_162)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

用spark shell读取数据文件并统计行数：

```
val textFile = sc.textFile("hdfs://localhost:9000/user/spark/export.csv")
val lineCount = textFile.count()
println(s"Total number of lines in the file: $lineCount")
```

```
scala> val textFile = sc.textFile("hdfs://localhost:9000/user/spark/export.csv")
textFile: org.apache.spark.rdd.RDD[String] = hdfs://localhost:9000/user/spark/export.csv
MapPartitionsRDD[3] at textFile at <console>:23

scala> val lineCount = textFile.count()
lineCount: Long = 1001

scala> println(s"Total number of lines in the file: $lineCount")
Total number of lines in the file: 1001

scala>
```

统计成功！共有1001行！

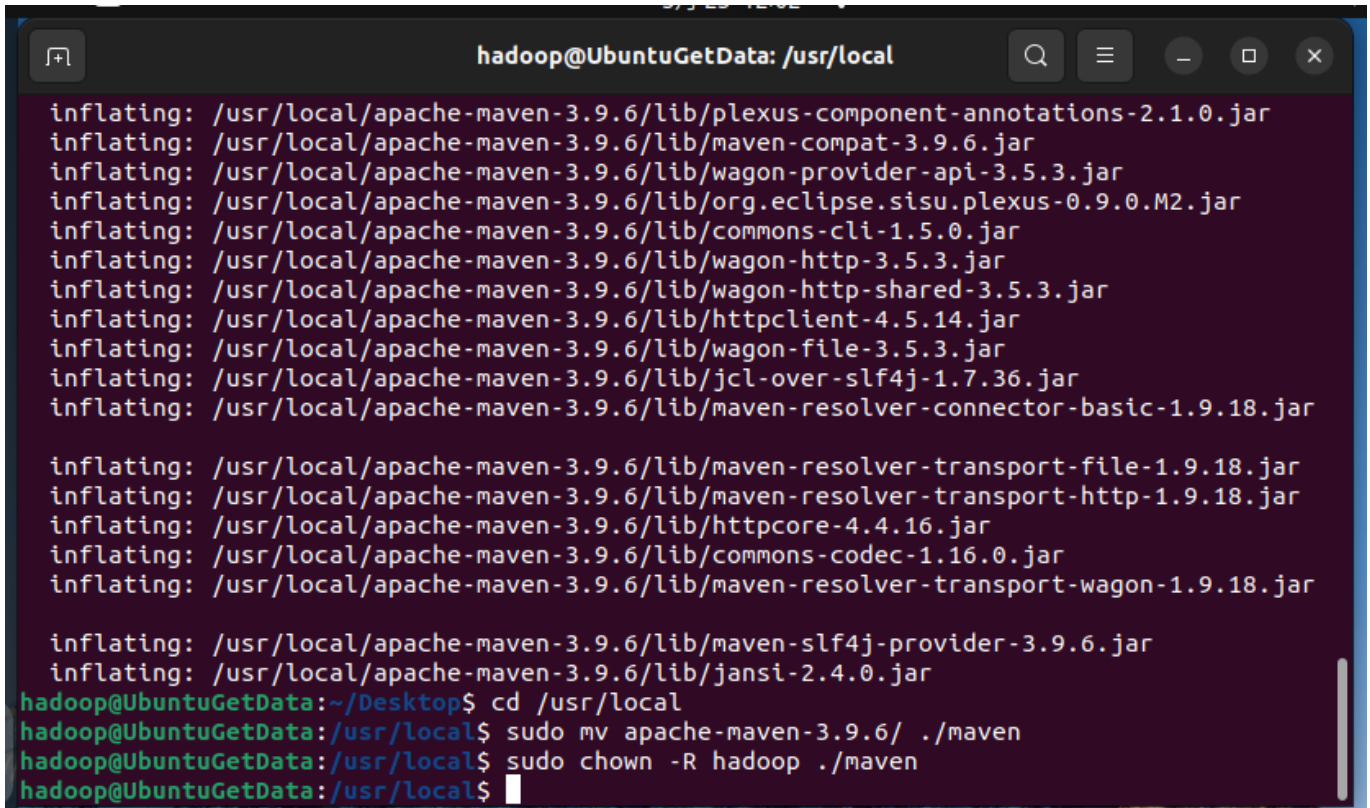
3 maven打包执行Java

首先安装maven：

访问[apache-maven的下载地址](#)，直接点击下载，选择[apache-maven-3.9.6-bin.zip](#)即可。

这里选择安装在/usr/local/maven目录中：

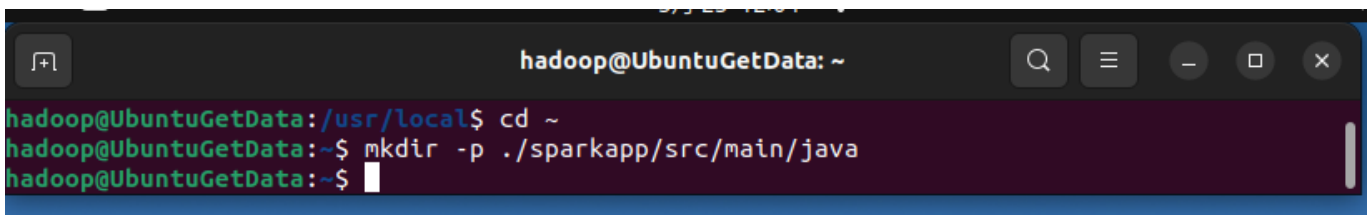

```
sudo unzip ~/Downloads/apache-maven-3.9.6-bin.zip -d /usr/local
cd /usr/local
sudo mv apache-maven-3.9.6/ ./maven
sudo chown -R hadoop ./maven
```

A terminal window titled 'hadoop@UbuntuGetData: /usr/local' showing the process of installing Apache Maven. It lists various JAR files being inflated from the Maven repository, including plexus-component-annotations, maven-compa, wagon-provider-api, org.eclipse.sisu.plexus, commons-cli, wagon-http, wagon-http-shared, httpclient, wagon-file, jcl-over-slf4j, maven-resolver-connector-basic, maven-resolver-transport-file, maven-resolver-transport-http, httpcore, commons-codec, maven-resolver-transport-wagon, maven-slf4j-provider, and jansi. The terminal then shows the user navigating to /usr/local, moving the apache-maven-3.9.6 directory to ./maven, and changing ownership to hadoop.

```
hadoop@UbuntuGetData: /usr/local
inflating: /usr/local/apache-maven-3.9.6/lib/plexus-component-annotations-2.1.0.jar
inflating: /usr/local/apache-maven-3.9.6/lib/maven-compa-3.9.6.jar
inflating: /usr/local/apache-maven-3.9.6/lib/wagon-provider-api-3.5.3.jar
inflating: /usr/local/apache-maven-3.9.6/lib/org.eclipse.sisu.plexus-0.9.0.M2.jar
inflating: /usr/local/apache-maven-3.9.6/lib/commons-cli-1.5.0.jar
inflating: /usr/local/apache-maven-3.9.6/lib/wagon-http-3.5.3.jar
inflating: /usr/local/apache-maven-3.9.6/lib/wagon-http-shared-3.5.3.jar
inflating: /usr/local/apache-maven-3.9.6/lib/httpclient-4.5.14.jar
inflating: /usr/local/apache-maven-3.9.6/lib/wagon-file-3.5.3.jar
inflating: /usr/local/apache-maven-3.9.6/lib/jcl-over-slf4j-1.7.36.jar
inflating: /usr/local/apache-maven-3.9.6/lib/maven-resolver-connector-basic-1.9.18.jar
inflating: /usr/local/apache-maven-3.9.6/lib/maven-resolver-transport-file-1.9.18.jar
inflating: /usr/local/apache-maven-3.9.6/lib/maven-resolver-transport-http-1.9.18.jar
inflating: /usr/local/apache-maven-3.9.6/lib/httpcore-4.4.16.jar
inflating: /usr/local/apache-maven-3.9.6/lib/commons-codec-1.16.0.jar
inflating: /usr/local/apache-maven-3.9.6/lib/maven-resolver-transport-wagon-1.9.18.jar
inflating: /usr/local/apache-maven-3.9.6/lib/maven-slf4j-provider-3.9.6.jar
inflating: /usr/local/apache-maven-3.9.6/lib/jansi-2.4.0.jar
hadoop@UbuntuGetData:~/Desktop$ cd /usr/local
hadoop@UbuntuGetData:/usr/local$ sudo mv apache-maven-3.9.6/ ./maven
hadoop@UbuntuGetData:/usr/local$ sudo chown -R hadoop ./maven
hadoop@UbuntuGetData:/usr/local$
```

在终端执行如下命令创建一个文件夹sparkapp作为应用程序根目录

```
cd ~
mkdir -p ./sparkapp/src/main/java
```

A terminal window titled 'hadoop@UbuntuGetData: ~' showing the user navigating to the home directory and creating the directory structure for the Spark application.

```
hadoop@UbuntuGetData:/usr/local$ cd ~
hadoop@UbuntuGetData:~$ mkdir -p ./sparkapp/src/main/java
hadoop@UbuntuGetData:~$
```

在 ./sparkapp/src/main/java 下建立一个名为 CountLines.java 的文件 (vim ./sparkapp/src/main/java/CountLines.java)，添加代码如下，即读取hadoop的文件：

```
/** SimpleApp.java */
import org.apache.spark.api.java.*;
import org.apache.spark.SparkConf;

public class CountLines {
    public static void main(String[] args) {
```

```
// 指定文件位于 HDFS 上的路径
String hdfsFile = "hdfs://localhost:9000/user/spark/export.csv";

// 配置 Spark
SparkConf conf = new SparkConf().setMaster("local").setAppName("File Line Counter");

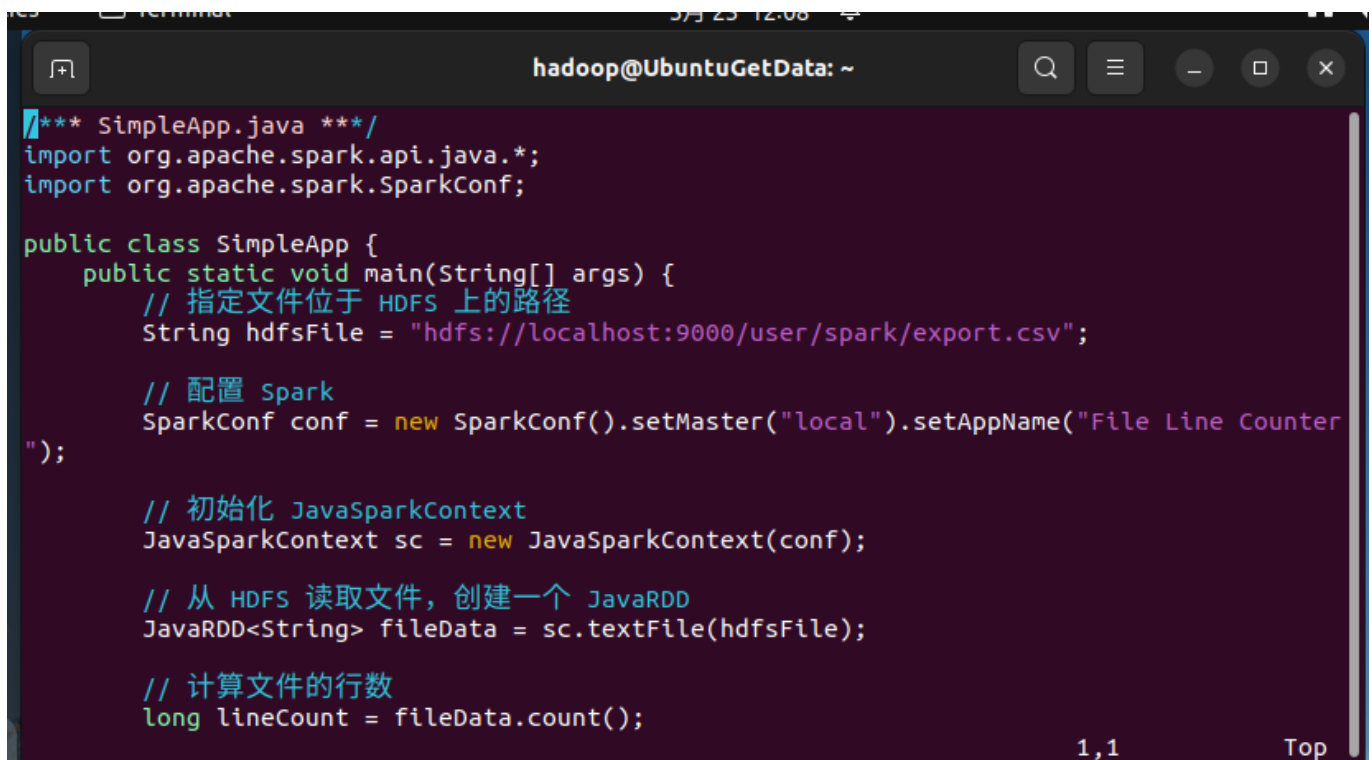
// 初始化 JavaSparkContext
JavaSparkContext sc = new JavaSparkContext(conf);

// 从 HDFS 读取文件，创建一个 JavaRDD
JavaRDD<String> fileData = sc.textFile(hdfsFile);

// 计算文件的行数
long lineCount = fileData.count();

// 输出行数
System.out.println("Total number of lines in the file: " + lineCount);

// 关闭 SparkContext
sc.close();
}
}
```



```
hadoop@UbuntuGetData: ~
/* ** SimpleApp.java ** */
import org.apache.spark.api.java.*;
import org.apache.spark.SparkConf;

package org.apache.spark.examples;

public class SimpleApp {
    public static void main(String[] args) {
        // 指定文件位于 HDFS 上的路径
        String hdfsFile = "hdfs://localhost:9000/user/spark/export.csv";

        // 配置 Spark
        SparkConf conf = new SparkConf().setMaster("local").setAppName("File Line Counter");

        // 初始化 JavaSparkContext
        JavaSparkContext sc = new JavaSparkContext(conf);

        // 从 HDFS 读取文件，创建一个 JavaRDD
        JavaRDD<String> fileData = sc.textFile(hdfsFile);

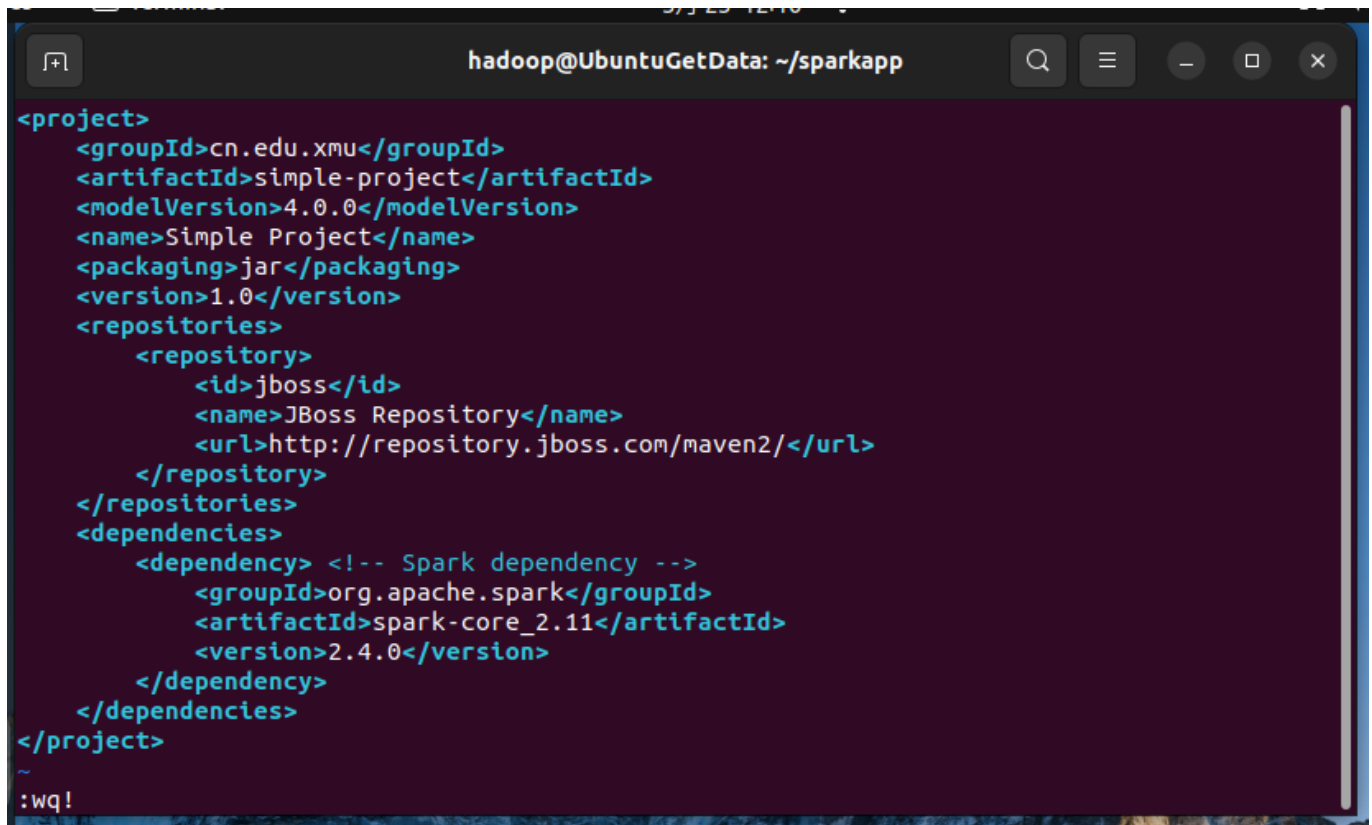
        // 计算文件的行数
        long lineCount = fileData.count();
    }
}
```

通过Maven进行编译打包。在`./sparkapp`目录中新建文件`pom.xml`，命令如下：

```
cd ~/sparkapp
vim pom.xml
```

在pom.xml文件中添加内容如下，声明该独立应用程序的信息以及与Spark的依赖关系：

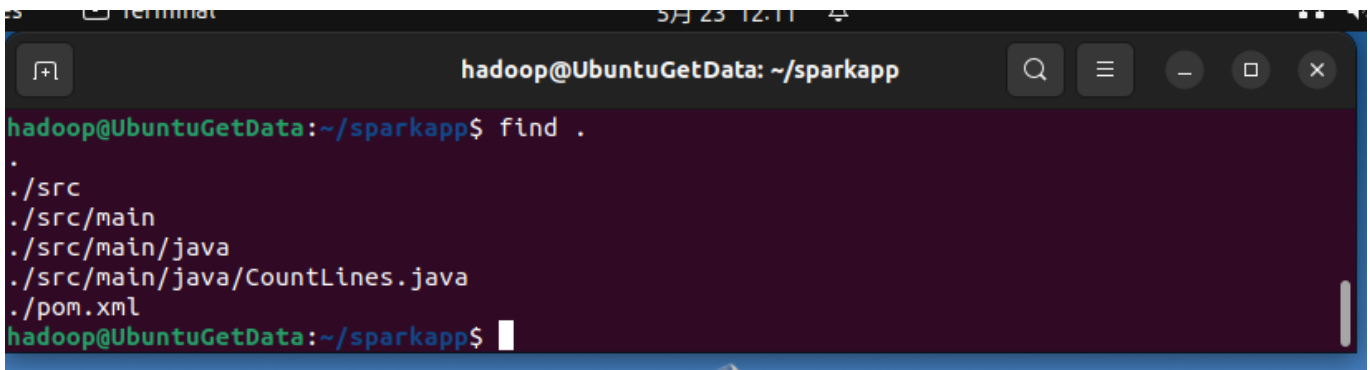
```
<project>
  <groupId>cn.edu.xmu</groupId>
  <artifactId>simple-project</artifactId>
  <modelVersion>4.0.0</modelVersion>
  <name>Simple Project</name>
  <packaging>jar</packaging>
  <version>1.0</version>
  <repositories>
    <repository>
      <id>jboss</id>
      <name>JBoss Repository</name>
      <url>http://repository.jboss.com/maven2/</url>
    </repository>
  </repositories>
  <dependencies>
    <dependency> <!-- Spark dependency -->
      <groupId>org.apache.spark</groupId>
      <artifactId>spark-core_2.11</artifactId>
      <version>2.4.0</version>
    </dependency>
  </dependencies>
</project>
```

A screenshot of a terminal window with a dark background. The title bar shows 'hadoop@UbuntuGetData: ~/sparkapp'. The terminal displays the same XML content as the code block above, with syntax highlighting. At the bottom of the terminal, there is a prompt character '~' followed by ':wq!'.

```
<project>
  <groupId>cn.edu.xmu</groupId>
  <artifactId>simple-project</artifactId>
  <modelVersion>4.0.0</modelVersion>
  <name>Simple Project</name>
  <packaging>jar</packaging>
  <version>1.0</version>
  <repositories>
    <repository>
      <id>jboss</id>
      <name>JBoss Repository</name>
      <url>http://repository.jboss.com/maven2/</url>
    </repository>
  </repositories>
  <dependencies>
    <dependency> <!-- Spark dependency -->
      <groupId>org.apache.spark</groupId>
      <artifactId>spark-core_2.11</artifactId>
      <version>2.4.0</version>
    </dependency>
  </dependencies>
</project>
~
:wq!
```

为了保证maven能够正常运行，先执行如下命令检查整个应用程序的文件结构:

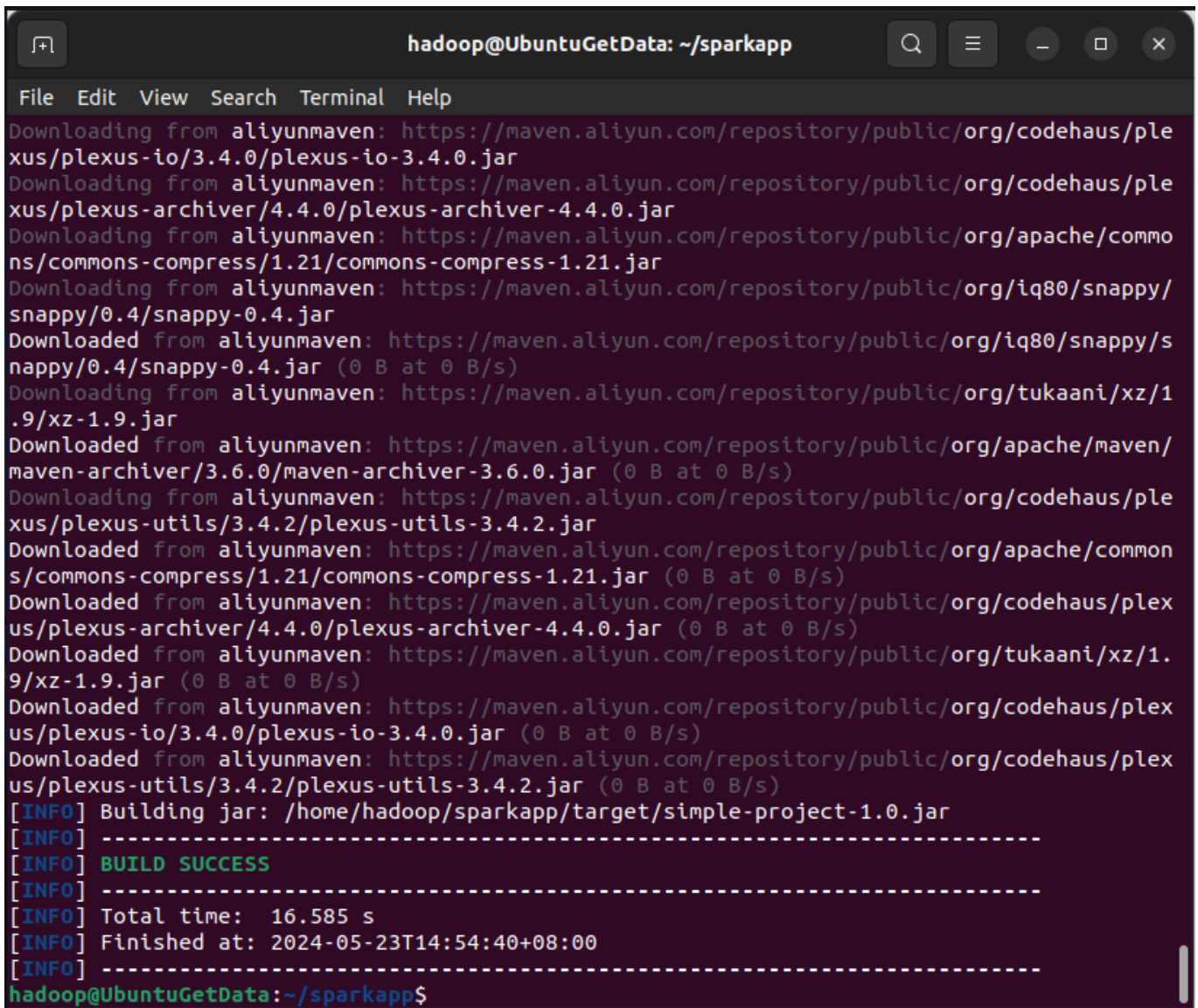

```
cd ~/sparkapp  
find .
```



```
hadoop@UbuntuGetData: ~/sparkapp  
hadoop@UbuntuGetData:~/sparkapp$ find .  
.  
./src  
./src/main  
./src/main/java  
./src/main/java/CountLines.java  
./pom.xml  
hadoop@UbuntuGetData:~/sparkapp$
```

通过如下代码将这整个应用程序打包成Jar:

```
cd ~/sparkapp  
/usr/local/maven/bin/mvn package
```

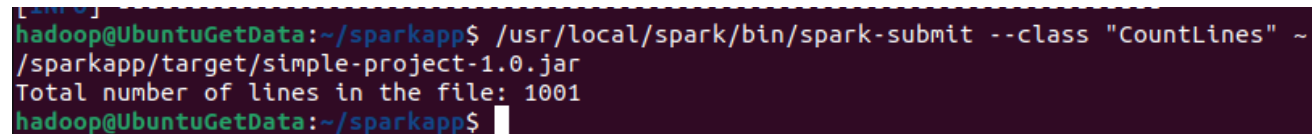


```
hadoop@UbuntuGetData: ~/sparkapp  
File Edit View Search Terminal Help  
Downloading from aliyunmaven: https://maven.aliyun.com/repository/public/org/codehaus/plexus/plexus-io/3.4.0/plexus-io-3.4.0.jar  
Downloading from aliyunmaven: https://maven.aliyun.com/repository/public/org/codehaus/plexus/plexus-archiver/4.4.0/plexus-archiver-4.4.0.jar  
Downloading from aliyunmaven: https://maven.aliyun.com/repository/public/org/apache/commons/commons-compress/1.21/commons-compress-1.21.jar  
Downloading from aliyunmaven: https://maven.aliyun.com/repository/public/org/iq80/snappy/snappy/0.4/snappy-0.4.jar  
Downloaded from aliyunmaven: https://maven.aliyun.com/repository/public/org/iq80/snappy/snappy/0.4/snappy-0.4.jar (0 B at 0 B/s)  
Downloading from aliyunmaven: https://maven.aliyun.com/repository/public/org/tukaani/xz/1.9/xz-1.9.jar  
Downloaded from aliyunmaven: https://maven.aliyun.com/repository/public/org/apache/maven/maven-archiver/3.6.0/maven-archiver-3.6.0.jar (0 B at 0 B/s)  
Downloading from aliyunmaven: https://maven.aliyun.com/repository/public/org/codehaus/plexus/plexus-utils/3.4.2/plexus-utils-3.4.2.jar  
Downloaded from aliyunmaven: https://maven.aliyun.com/repository/public/org/apache/commons/commons-compress/1.21/commons-compress-1.21.jar (0 B at 0 B/s)  
Downloaded from aliyunmaven: https://maven.aliyun.com/repository/public/org/codehaus/plexus/plexus-archiver/4.4.0/plexus-archiver-4.4.0.jar (0 B at 0 B/s)  
Downloaded from aliyunmaven: https://maven.aliyun.com/repository/public/org/tukaani/xz/1.9/xz-1.9.jar (0 B at 0 B/s)  
Downloaded from aliyunmaven: https://maven.aliyun.com/repository/public/org/codehaus/plexus/plexus-io/3.4.0/plexus-io-3.4.0.jar (0 B at 0 B/s)  
Downloaded from aliyunmaven: https://maven.aliyun.com/repository/public/org/codehaus/plexus/plexus-utils/3.4.2/plexus-utils-3.4.2.jar (0 B at 0 B/s)  
[INFO] Building jar: /home/hadoop/sparkapp/target/simple-project-1.0.jar  
[INFO] -----  
[INFO] BUILD SUCCESS  
[INFO] -----  
[INFO] Total time: 16.585 s  
[INFO] Finished at: 2024-05-23T14:54:40+08:00  
[INFO] -----  
hadoop@UbuntuGetData:~/sparkapp$
```

编译打包成功！

通过将生成的jar包通过spark-submit提交到Spark中运行，如下命令：

```
/usr/local/spark/bin/spark-submit --class "CountLines" ~/sparkapp/target/simple-project-1.0.jar
```

A terminal window screenshot showing the execution of the spark-submit command. The prompt is 'hadoop@UbuntuGetData:~/sparkapp\$'. The command entered is '/usr/local/spark/bin/spark-submit --class "CountLines" ~/sparkapp/target/simple-project-1.0.jar'. The output is 'Total number of lines in the file: 1001'. The prompt returns to 'hadoop@UbuntuGetData:~/sparkapp\$'.

```
hadoop@UbuntuGetData:~/sparkapp$ /usr/local/spark/bin/spark-submit --class "CountLines" ~/sparkapp/target/simple-project-1.0.jar
Total number of lines in the file: 1001
hadoop@UbuntuGetData:~/sparkapp$
```

执行成功！