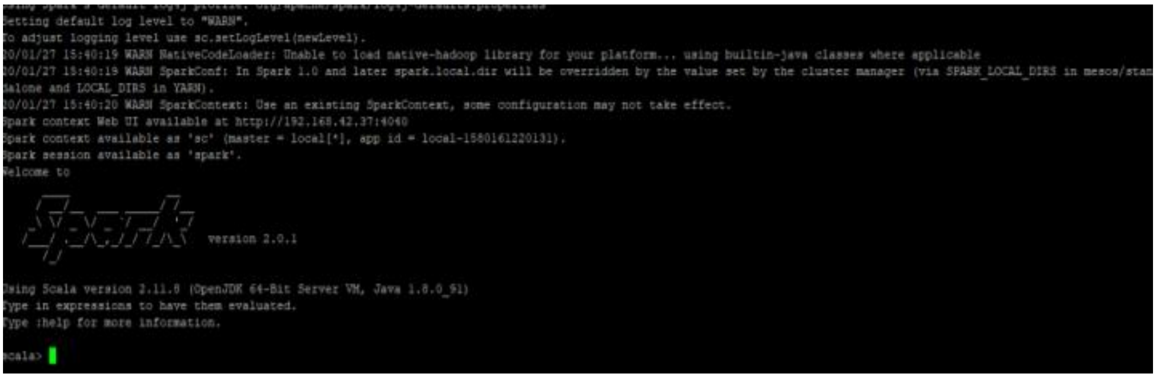
SPARK GUIDELINES:

Steps to invoke Spark-shell:

1. Login with your CSX username and password
2. Put command: spark-shell
   It will open up the shell

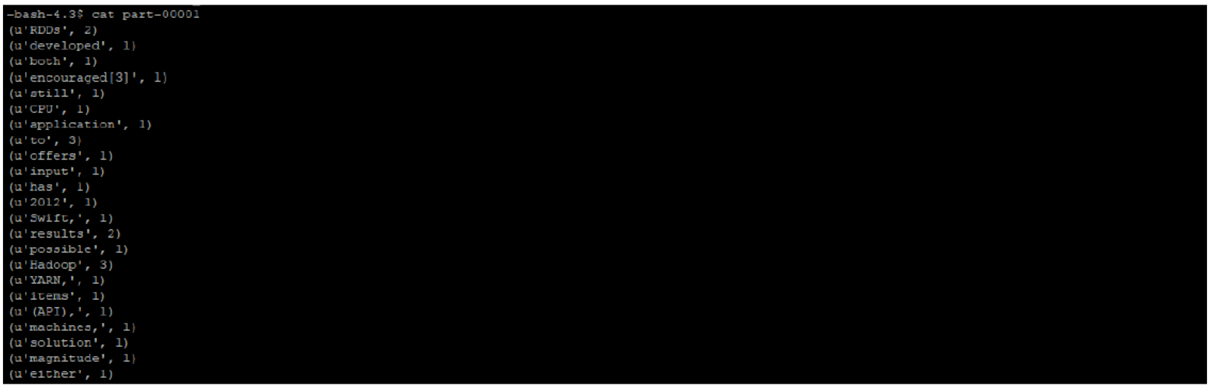


```
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel).
20/01/27 15:40:19 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
20/01/27 15:40:19 WARN SparkConf: In Spark 1.0 and later spark.local.dir will be overridden by the value set by the cluster manager (via SPARK_LOCAL_DIRS in mesos/stan
dalone and LOCAL_DIRS in YARN).
20/01/27 15:40:20 WARN SparkContext: Use an existing SparkContext, some configuration may not take effect.
Spark context Web UI available at http://192.168.42.37:4040
Spark context available as 'sc' (master = local[*], app id = local-1580161220131).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.0.1
      /_/

Using Scala version 2.11.8 (OpenJDK 64-Bit Server VM, Java 1.8.0_91)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

3. For Python : Use pyspark

Steps to Run a spark Application written in Python:

1. Login with your CSX username and password
2. Run the python file using spark-submit command:
   Command: spark-submit pathtoPaythonFile/pythonFile.py
   Ex: spark-submit wordcount.py
3. Checking the output
   Command: cat /output/part-r-00000
   Ex: cat /home/ankaush/spark_python/word_output/part-r-00000



```
-bash-4.3$ cat part-00001
(u'RDDs', 2)
(u'developed', 1)
(u'both', 1)
(u'encouraged[3]', 1)
(u'still', 1)
(u'CPU', 1)
(u'application', 1)
(u'to', 3)
(u'offers', 1)
(u'input', 1)
(u'has', 1)
(u'2012', 1)
(u'Swift,', 1)
(u'results', 2)
(u'possible', 1)
(u'Hadoop', 3)
(u'YARN,', 1)
(u'items', 1)
(u'(API),', 1)
(u'machines,', 1)
(u'solution', 1)
(u'magnitude', 1)
(u'either', 1)
```

Example Program:

wordcount.py

import sys

from pyspark import SparkContext, SparkConf

```
if __name__ == "__main__":

    conf = SparkConf().setAppName("Word Count - Python")

    sc = SparkContext(conf=conf)

    # read in text file and split each document into words

    words = sc.textFile("file:///home/ankaush/word_input/").flatMap(lambda line: line.split(" "))

    wordCounts = words.map(lambda word: (word, 1)).reduceByKey(lambda a,b:a +b)

    wordCounts.collect()

    wordCounts.saveAsTextFile("hdfs:///user/ankaush/word_output")
```

Note for the Python Program:

In the example program, the input files are stored in local directory. The program output will be stored in your hdfs directory in the path your will mentiond.

The path of hdfs directory will be : /user/yourusername/*. Ex: /user/ankaush/word_output

Steps to Run a spark Application written in Java:

1. Make a folder say: inputs and put the input files in it
2. Run the Java Jar file like this:
   Command: spark-submit –class “YourJavaClassname” –master local pathtojarfile “pathtoyourinputfiles”
   Ex: spark-submit –class “Wordcount” –master yarn /home/ankaush/wordcount.jar “file:///home/ankaush/inputs”
3. Checking the output:
   Command: cat /output/part-r-00000
   Ex: cat /home/ankaush/java_output/part-00000

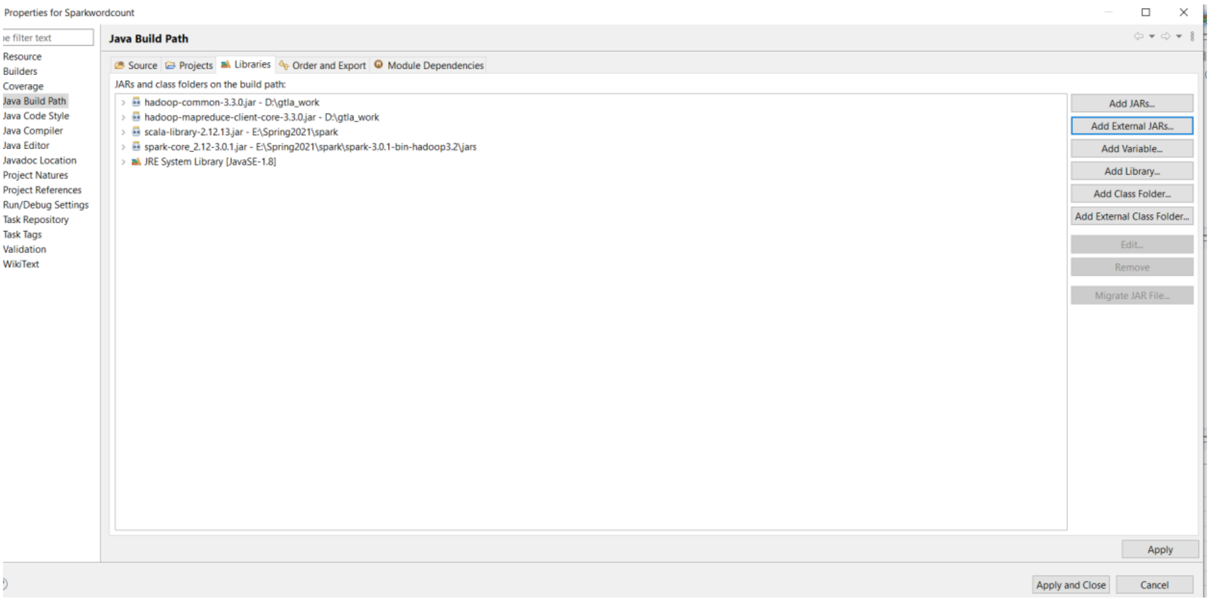
Example Program:

Wordcount.java

```
import java.util.Arrays;
import org.apache.hadoop.fs.Path;
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.JavaPairRDD;
import org.apache.spark.api.java.JavaRDD;
import org.apache.spark.api.java.JavaSparkContext;
import org.apache.spark.api.java.function.FlatMapFunction;
import org.apache.spark.api.java.function.Function;
import scala.Tuple2;

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

        SparkConf sparkConf = new   SparkConf().setMaster("local").setAppName("JD
Word Counter");

      JavaSparkContext sparkContext = new JavaSparkContext(sparkConf);

      JavaRDD<String> textFile = sparkContext.textFile(args[0]);
      JavaPairRDD<String, Integer> counts = textFile
                .flatMap(s -> Arrays.asList(s.split(" ")).iterator())
                .mapToPair(word -> new Tuple2<>(word, 1))
                .reduceByKey((a, b) -> a + b);
      counts.saveAsTextFile("file:///home/ankaush/java_output");
        }

}
```

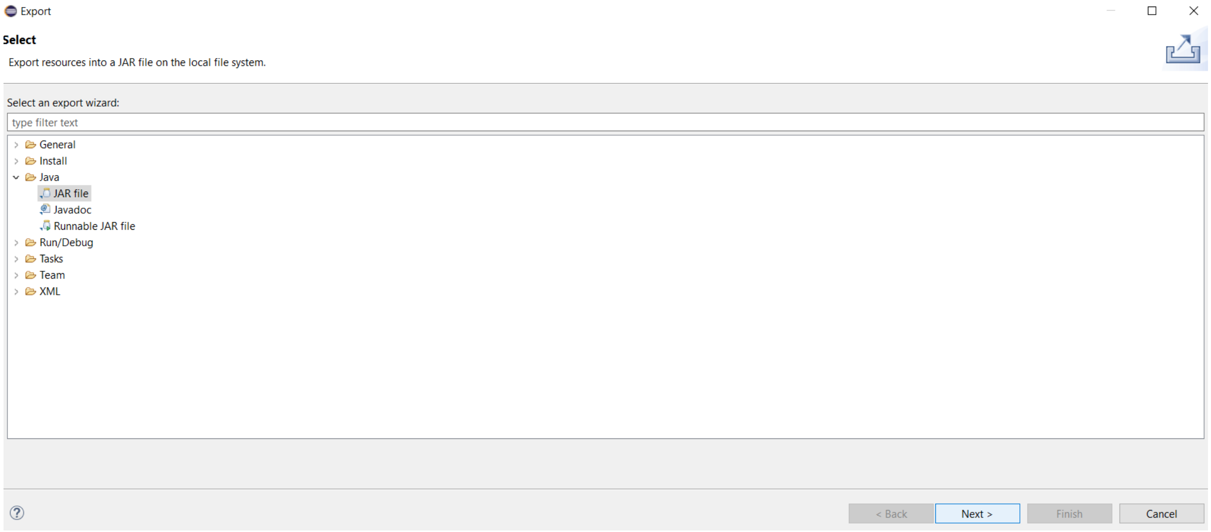
Notes for Java Program:

1. You need to create a Java project in your local IDE like Eclipse or any other and import the these jar files in your project and import these jar files by configure build path
   a) Hadoop-common
   b) Hadoop-mapreduce-client-core
   c) spark-core
   d) Scala-library
2. Steps to add Jar files in your Project in Eclipse:
   a) Right click in your project
   b) Choose Build Path → Configure Build Path → Add External JARs



JARs and class folders on the build path:
- hadoop-common-3.3.0.jar - D:\gtla_work
- hadoop-mapreduce-client-core-3.3.0.jar - D:\gtla_work
- scala-library-2.12.13.jar - E:\Spring2021\spark
- spark-core_2.12-3.0.1.jar - E:\Spring2021\spark\spark-3.0.1-bin-hadoop3.2\jars
- JRE System Library [JavaSE-1.8]

3. Then Create the jar File and put it in your local directory in the server
4. Steps to create the JAR File:
   a) Right click on your project and choose export
   b) Choose JAR file option in under Java



Export

Select

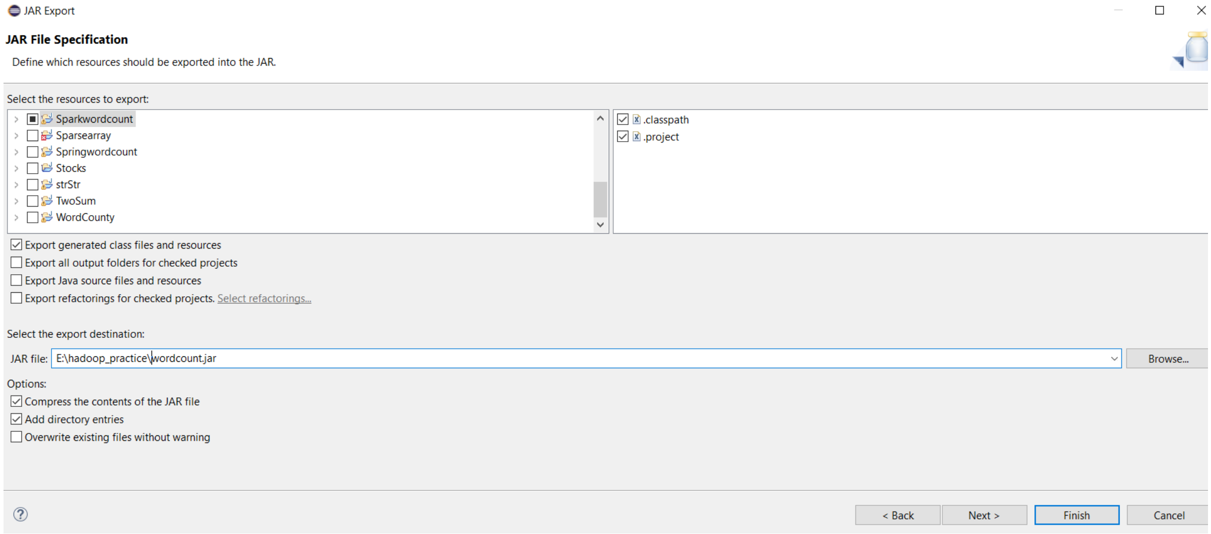
Export resources into a JAR file on the local file system.

Select an export wizard:

type filter text

- General
- Install
- Java
  - JAR file
  - Javadoc
  - Runnable JAR file
- Run/Debug
- Tasks
- Team
- XML

c) In next tab,Choose the JAR file location where you want to store it.



JAR Export

JAR File Specification

Define which resources should be exported into the JAR.

Select the resources to export:

- ■ Sparkwordcount
- □ Sparsearray
- □ Springwordcount
- □ Stocks
- □ strStr
- □ TwoSum
- □ WordCounty

☑ .classpath
☑ .project

☑ Export generated class files and resources
□ Export all output folders for checked projects
□ Export Java source files and resources
□ Export refactorings for checked projects. Select refactorings...

Select the export destination:

JAR file: E:\hadoop_practice\wordcount.jar

Options:

☑ Compress the contents of the JAR file
☑ Add directory entries
□ Overwrite existing files without warning

d) Click Finish. You are done

5. Put this Jar file in your server directory.