Ex. No. : 10

**Apache Spark**

**Aim:**

To Install Apache Spark, Configure and Deploy in Single Server Standalone mode and run a wordcount scala application on it.

**Procedure**

1.Install Apache Spark

Download Apache Spark from **<https://dlcdn.apache.org/spark/spark-3.5.1/spark-3.5.1-bin-hadoop3.tgz>**

2.Move the downloaded file from downloads to your home directory.

3.Extract the file using

$tar xvf spark-3.5.1-bin-hadoop3-scala2.13.tgz

4.Set the environment variable by modifying .bashrc file

$nano .bashrc

Add the following lines at the end

export SPARK\_HOME=/home/cseds/spark-3.5.1-bin-hadoop3-scala2.13

export PATH=$PATH:$SPARK\_HOME/bin:$SPARK\_HOME/sbin

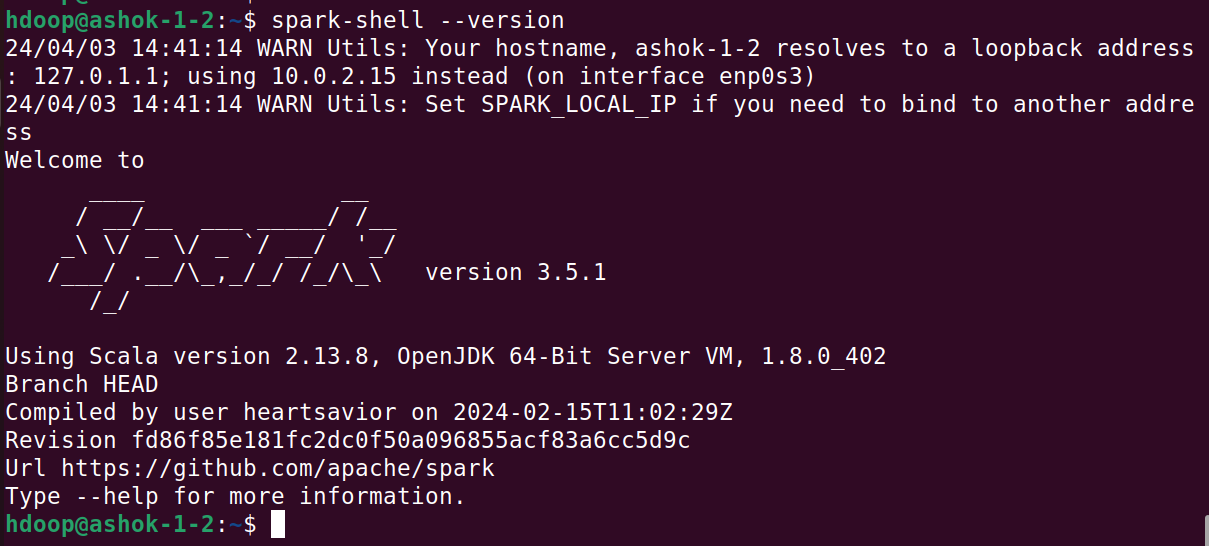
Save the file and exit nano editor

5. To make the above changes to take effect type following command

$source .bashrc

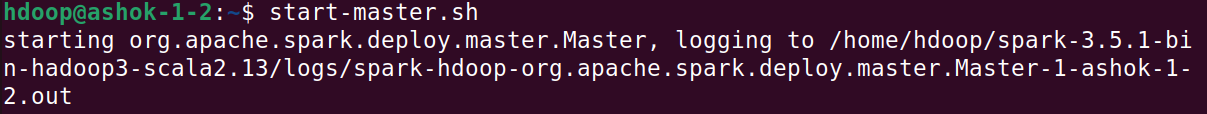
6. Check Spark installed successfully or not by checking version by below command

$spark-shell --version



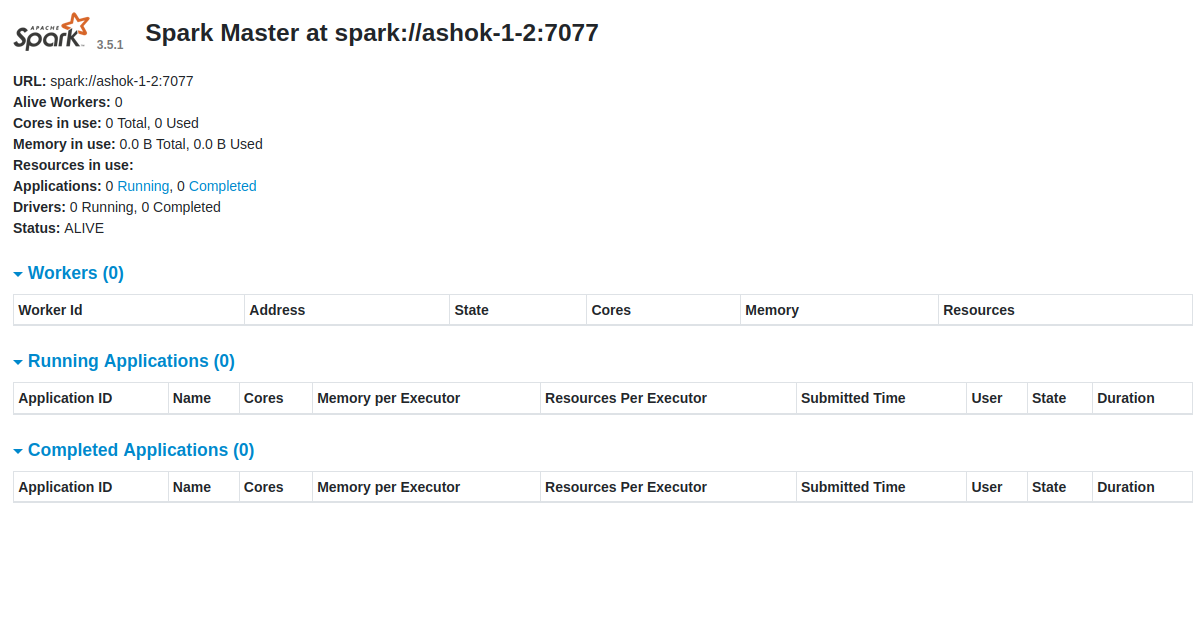
7.Start Standalone Master Server

$start-master.sh



Check the Spark WebUI by typing below command in web browser

<http://localhost:8080>

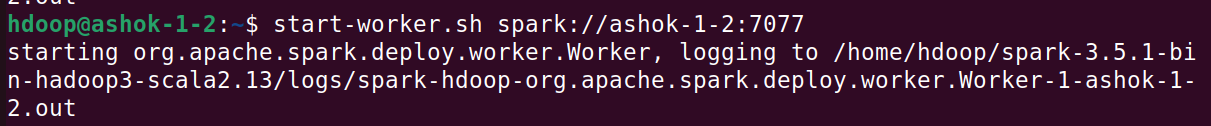


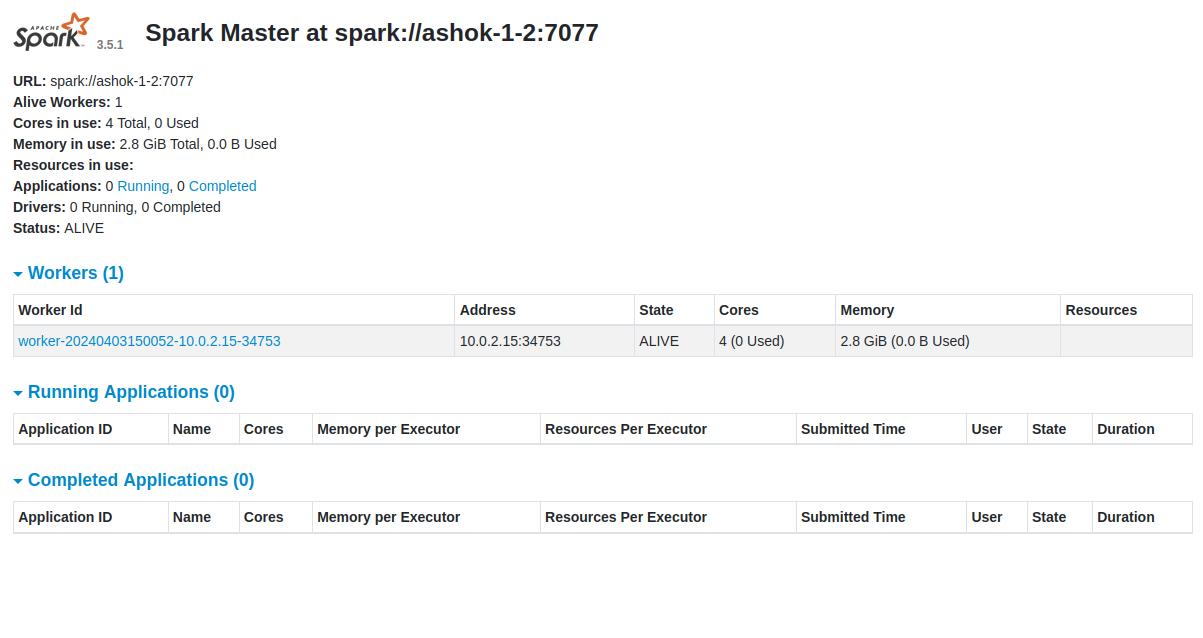
8.Start Worker Node (Previously slave node)

$start-worker.sh spark://localhost:7077

Or

$start-worker.sh spark://ashok-1-2:7077





9.Check all started processes running or not by

$jps

10.Start Worker Node and Assign Number of CPU Core

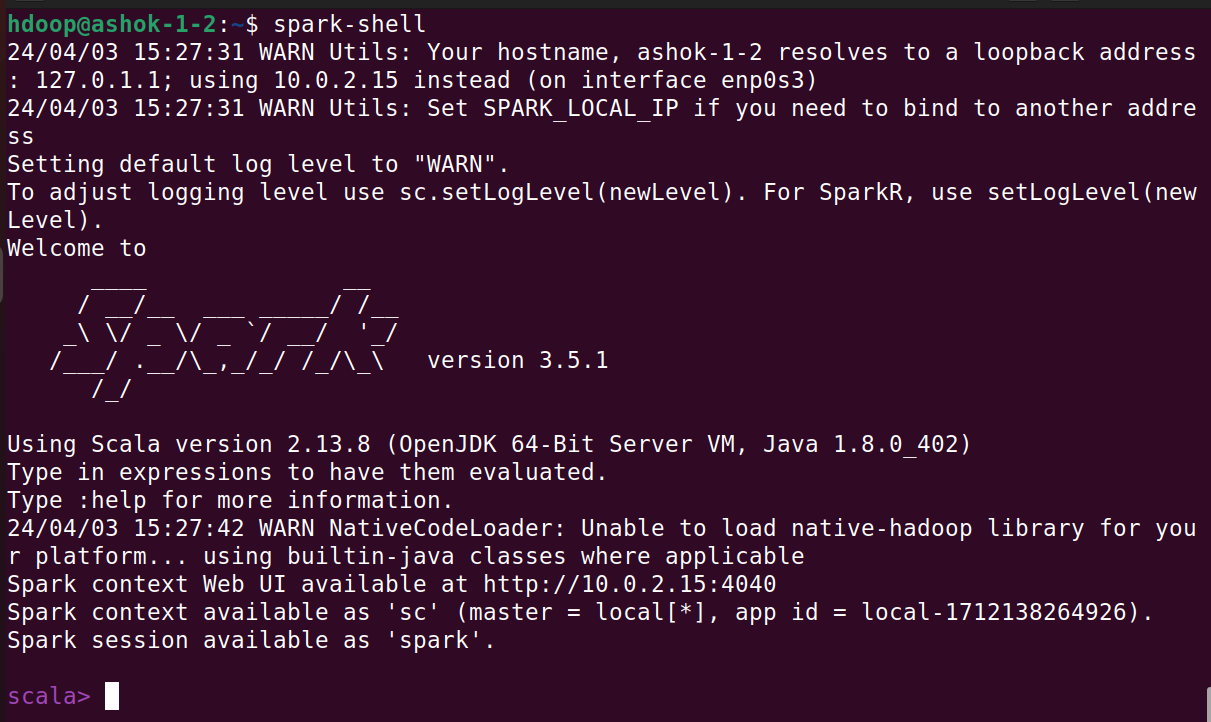
$start-worker.sh -c 2 spark://cseds:7077

11.Start Worker Node and Assign Number of CPU Core and RAM

$start-worker.sh -c 2 -m 512M spark://cseds:7077

12.Enter into Spark-shell (Scala Shell)

$spark-shell



**Word Count in Scala**

scala>val myfile = sc.textFile("/home/hdoop/demofile.txt")

scala>myfile.collect()

scala>val splitdata = myfile.flatMap(line => line.split(" "))

scala>splitdata.collect()

scala> val mapdata = splitdata.map(word => (word,1))

scala>mapdata.collect()

scala> val reducedata = mapdata.reduceByKey(\_+\_)

scala> reducedata.collect()

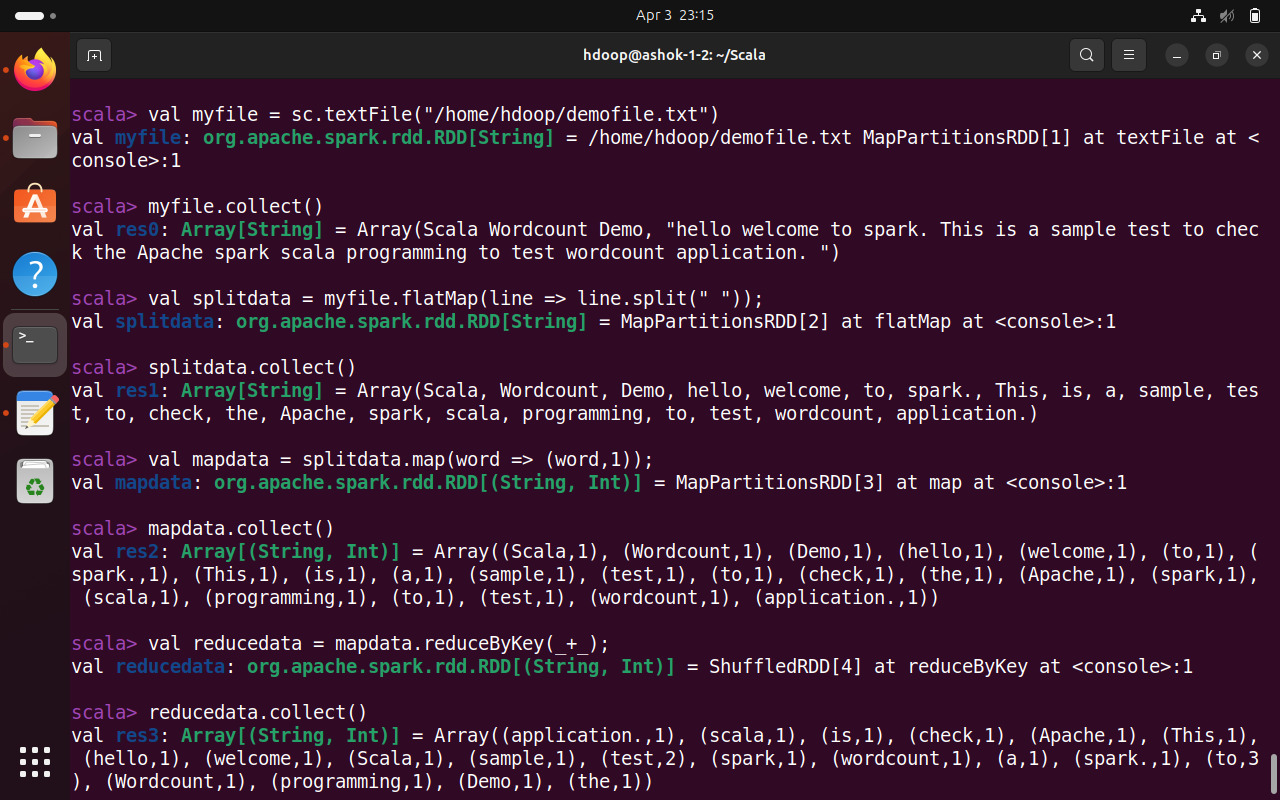
**or**

scala>val myfile = sc.textFile("/home/hdoop/demofile.txt")

scala>val counts = myfile.flatMap(line => line.split(" ")).map(word => (word, 1)).reduceByKey(\_+\_)

scala>counts.saveAsTextFile("/home/hdoop/output")

**Output**



**Example 2 – Creating Spark Dataframe**

import spark.implicits.\_

val data = Seq(("Java", "20000"), ("Python", "100000"), ("Scala", "3000"))

val df = data.toDF()

df.show()

**Example 3:**

**Creating Wordcount Self-contained Application using sbt**

Step1: Create a directory sparkexample and inside create one file and save as **build.sbt** and type following lines inside that file

name := "Example Project"

version := "1.0"

scalaVersion := "2.13.8"

libraryDependencies += "org.apache.spark" %% "spark-sql" % "3.3.2"

**Step 2: Create a text file with some contents and store it inside sparkexample folder**

**Step 3: Create WordCount.scala file and store it in location sparkexample/src/main/scala/**

import org.apache.spark.{SparkConf, SparkContext}

object WordCount {

def main(args: Array[String]): Unit = {

// Create a SparkConf object to configure the Spark Application

val conf = new SparkConf().setAppName("WordCount").setMaster("local")

// Create a SparkContext object

val sc = new SparkContext(conf)

// Load the input data

val input = sc.textFile("demofile.txt")

// Split each line into words

val words = input.flatMap(line => line.split(" "))

// Map each word to a tuple (word, 1) to represent word count

val wordCounts = words.map(word => (word, 1))

// Reduce by key to sum up the counts for each word

val wordCountsReduced = wordCounts.reduceByKey((count1, count2) => count1 + count2)

// Save the results to a text file

wordCountsReduced.saveAsTextFile("output")

// Stop the SparkContext

sc.stop()

}

}

**Execution Steps**

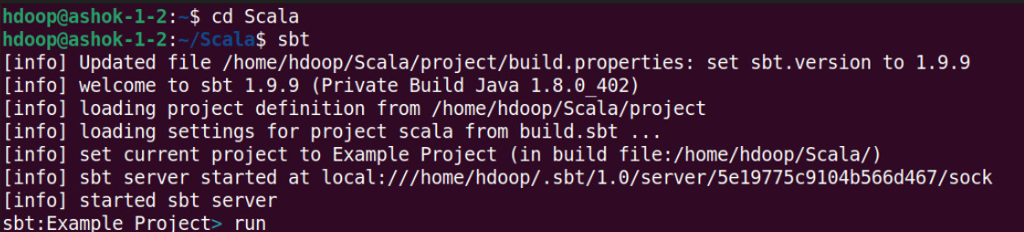
**You should be inside sparkexample folder to run. For execution of above application type sbt**

hdoop@ashok-1-2:~/sparkexample$ sbt

upon successful launch of sbt it will diplay below line

**[info] started sbt server**

**sbt:Example Project> run**

****

**Result**

Apache Spark has been successfully installed and configured in single server standalone mode and wordcount program using scala has successfully implemented and result is verified.