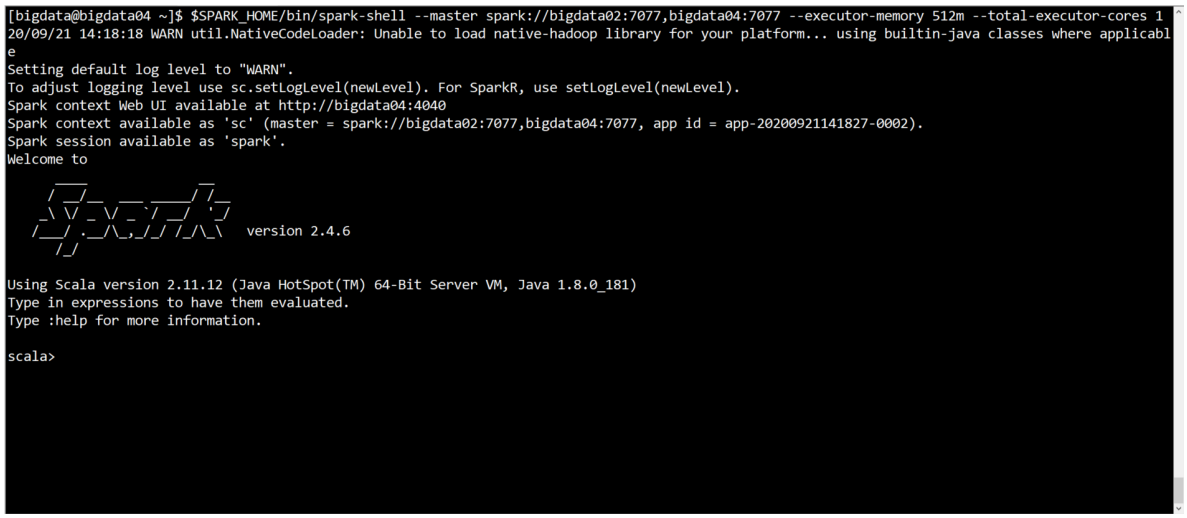
1. 启动Spark Shell

启动命令：

```
spark-shell
```

```
$SPARK_HOME/bin/spark-shell \
--master spark://bigdata02:7077,bigdata04:7077 \
--executor-memory 512M \
--total-executor-cores 2
```



注意上图中的 cores 参数，如果是 0，那么以后这个 spark shell 中运行的代码是不能执行成功的。千万注意。必要要把 cpu cores 和 memory 设置合理。

```
1、executor memory不能超过虚拟机的内存
2、cpu cores不要超过spark集群能够提供的总cpu cores，否则会使用全部。最好不要使用全部。否则其他程序由于没有cpu core可用，就不能正常运行
```

参数说明：

```
--master spark://bigdata02:7077          指定Master的地址
--executor-memory 2G                     指定每个worker可用内存为2G
--total-executor-cores 2                 指定整个集群使用的cup核数为2个
```

注意：

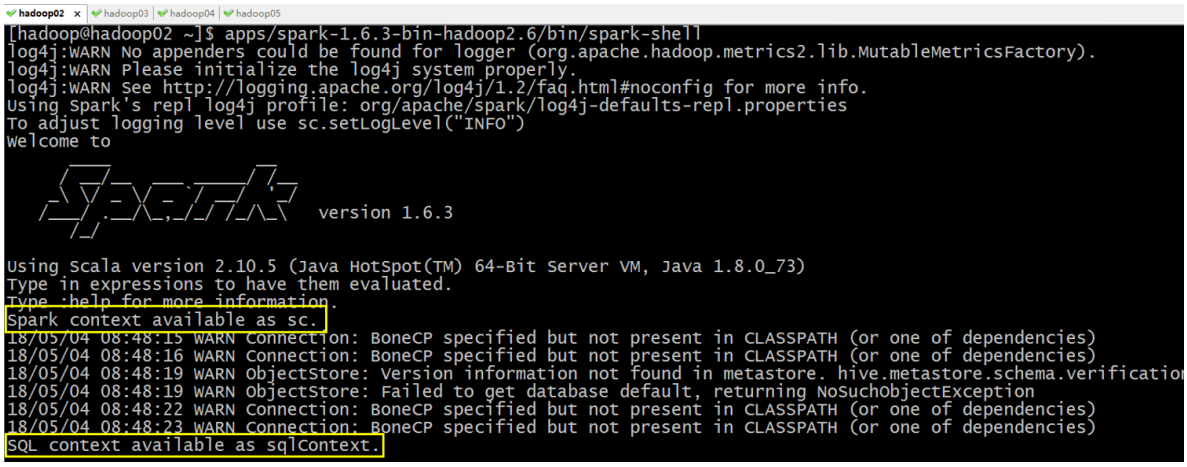
```
如果启动spark shell时没有指定master地址，但是也可以正常启动spark shell和执行spark shell中的程序，其实是启动了spark的local模式，该模式仅在本机启动一个Driver进程，没有与集群建立联系。
```

Spark-2.x

```
Spark Shell 中已经默认将 SparkContext 类初始化为对象 sc
Spark Shell 中已经默认将 SparkSession 类初始化为对象 spark
用户代码如果需要用到，则直接应用 sc，spark 即可
```

Spark-1.x

```
Spark Shell 中已经默认将 SparkContext 类初始化为对象 sc
Spark Shell 中已经默认将 SQLContext 类初始化为对象 sqlContext
用户代码如果需要用到，则直接应用sc，sqlContext即可
```



2. 在Spark Shell中编写WordCount程序

在提交WordCount程序之前，先在HDFS集群中的准备一个文件用于做单词统计：

words.txt内容如下：

```
hello huangbo
hello xuzheng
hello wangbaoqiang
```

把该文件上传到HDFS文件系统中：

```
hadoop fs -mkdir -p /spark/wc/input
hadoop fs -put words.txt /spark/wc/input
```

在 Spark Shell 中提交 WordCount 程序：

```
sc.textFile("hdfs://hadoop277ha/spark/wc/input/words.txt").flatMap(_.split("
")).map((_,1)).reduceByKey(_+_).saveAsTextFile("hdfs://hadoop277ha/spark/wc/outp
ut222")
```

查询验证结果：

```
hadoop fs -ls hdfs://hadoop277ha/spark/wc/output
hadoop fs -cat hdfs://hadoop277ha/spark/wc/output/*
```

说明：

```
sc.textFile("hdfs://hadoop277ha/spark/wc/input/words.txt")
  .flatMap(line => line.split(""))
  .map(word => (word, 1))             // 你给我一个单词，我给变成单词对返回
  .reduceByKey((x, y) => x + y)
  .saveAsTextFile("hdfs://hadoop277ha/spark/wc/output222")

1、sc是SparkContext对象，该对象时提交spark程序的入口

2、textFile("hdfs://hadoop277ha/spark/wc/input/words.txt")是从HDFS中读取数据
    底层而是通过InputFormat去读取（因为数据在HDFS，从HDFS读取数据的规范就是InputFormat）
    RDD[String]   （文件中的一行，就是RDD中的一条数据。）

3、flatMap(_.split(" "))先map再压平
    切割压平
    Array(Array("hello", "a"),  Array("hello", "b")) = Array("hello", "a",
"hello", "b")

4、map((_,1))将单词和1构成元组(word,1)

5、reduceByKey(_+_)按照key进行reduce，并将value累加

6、saveAsTextFile("hdfs://hadoop277ha/spark/wc/output")将结果写入到HDFS对应输出目录中
```