- **创建Maven项目**
  - 设置相关依赖 pom.xml

```
<dependencies>
        <dependency>
            <groupId>org.apache.hadoop</groupId>
            <artifactId>hadoop-common</artifactId>
            <version>2.7.5</version>
        </dependency>
        <dependency>
            <groupId>org.apache.hadoop</groupId>
            <artifactId>hadoop-client</artifactId>
            <version>2.7.5</version>
        </dependency>
        <dependency>
            <groupId>org.apache.hadoop</groupId>
            <artifactId>hadoop-hdfs</artifactId>
            <version>2.7.5</version>
        </dependency>
        <dependency>
            <groupId>org.apache.hadoop</groupId>
            <artifactId>hadoop-mapreduce-client-core</artifactId>
            <version>2.7.5</version>
        </dependency>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.13</version>
        </dependency>
    </dependencies>
    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.0</version>
                <configuration>
                    <source>1.8</source>
                    <target>1.8</target>
```

```
                    <encoding>UTF-8</encoding>
                    <!--    <verbal>true</verbal>-->
                </configuration>
            </plugin>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-shade-plugin</artifactId>
                <version>2.4.3</version>
                <executions>
                    <execution>
                        <phase>package</phase>
                        <goals>
                            <goal>shade</goal>
                        </goals>
                        <configuration>
                            <minimizeJar>true</minimizeJar>
                        </configuration>
                    </execution>
                </executions>
            </plugin>
        </plugins>
    </build>
</project>
```

- **Map阶段编写**

```
package sz.base.java;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

import java.io.IOException;

/**
 * map方法的生命周期:  框架每传一行数据就被调用一次
 * MapReduce：继承Mapper(分的类型) 形参为<k1,v1><k2,v2>
 * LongWritable:需要序列化，offset 偏移量，每行的偏移量   k1
 * Text：每行的数据，整行数据    v1
```

```
 * Text：输出的每个单词     k2
 * IntWritable：1     v2
 */
public class WordCountMapper extends Mapper<LongWritable, Text, Text, IntWritable> {
    //实现map
    @Override
    protected void map(LongWritable key, Text value, Mapper<LongWritable, Text, Text, IntWritable>.Context context) throws IOException, InterruptedException {
        //1.读取每行数据
        String line = value.toString();
        //2.逗号切分为每行数据
        if (line != null || line.length() != 0) {
            String[] words = line.split(",");
            //3.转换成<单词,1>
            for (String word : words) {
                //context写出到下一步
                context.write(new Text(word), new IntWritable(1));
            }
        }
    }
}

```

- **reduce阶段编写**

```
package sz.base.java;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

/**
 * //生命周期：框架每传递进来一个kv 组，reduce方法被调用一次
 * reduce：
 * Text: k2 单词
 * IntWritable：v2 : 1
 * Text：k3 : 单词
 * IntWritable :  v3 : 单词个数
```

```
 */
public class WordCountReduce extends Reducer<Text, IntWritable, Text, IntWritable> {
    @Override
    protected void reduce(Text key, Iterable<IntWritable> values, Reducer<Text, IntWritable, Text, IntWritable>.Context context) throws IOException, InterruptedException {
        //定义累加的和
        int count = 0;
        for (IntWritable value : values) {
            //累加v2中的每个集合中的值
            count += value.get();
        }
        //context写出去
        context.write(key, new IntWritable(count));
    }
}

```

- **main方法编写**

```
package sz.base.java;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

import java.io.IOException;

public class WordCountMain {
    public static void main(String[] args) {
        try {
            //1.实例化job ， 每个MapReduce都是一个job 创建一个job任务对象,并指定job的名字
            Job job = Job.getInstance(new Configuration(), "wordcount");
            //定义job的jar包参数需要设置一下
            job.setJarByClass(WordCountMain.class);
            //2.设置一下当前输入的格式（数据格式）和输入的文件路径
            job.setInputFormatClass(TextInputFormat.class);
```

```
            TextInputFormat.addInputPath(job, new Path("D:\\logs\\input\\words.txt"));
            //3.设置mapper类和map输出的key和value的类型
            job.setMapperClass(WordCountMapper.class);
            job.setOutputKeyClass(Text.class);
            job.setOutputValueClass(IntWritable.class);
            //4.设置自定义的分区、排序、规约、分组规则，如果不写按照默认

            //5.设置reducer类 和reduce输出的key 和value 的类型
            job.setReducerClass(WordCountReduce.class);
            job.setOutputKeyClass(Text.class);
            job.setOutputValueClass(IntWritable.class);
            //6.设置输出的格式，输出的路径
            job.setOutputFormatClass(TextOutputFormat.class);
            TextOutputFormat.setOutputPath(job, new Path("D:\\logs\\output"));
            //7.等待这个任务的执行
            boolean status = job.waitForCompletion(true);
            //8.如果成功了退出当前job程序
            System.exit(status ? 0 : 1);
        } catch (IOException | InterruptedException | ClassNotFoundException e) {
            e.printStackTrace();
        }

    }
}

```

- 集群运行则把路径修改为hdfs路径

```
TextInputFormat.addInputPath(job, new Path("hdfs://node1:8020/input/wordcount"));
TextOutputFormat.setOutputPath(job, new Path("hdfs://node1:8020/output/wordcount"));
```

- 将程序打成JAR包，然后在集群的任意一个节点上用hadoop命令启动

```
hadoop jar wordcount.jar cn.itcast.WordCountDriver
```

- 如果出现(null) entry in command string: null

```
解决方法：
下载hadoop.dll文件，拷贝到c:\windows\system32目录中即可hadoop.dll
可以在github上下载：https://github.com/4ttty/winutils
各个版本的hadoop.dll好像是通用的。
```