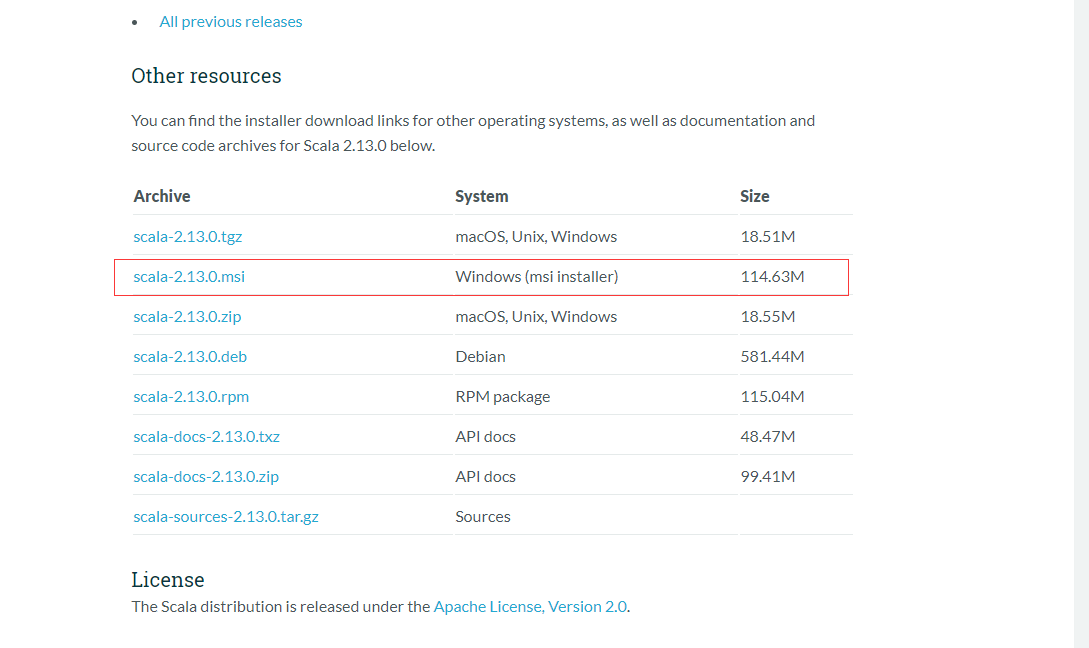
## Windows开发环境构建

### JDK 说明

安装(略)—版本在1.8及之上(如果有多版本的话,在path中将JAVA\_HOME放在最前面)

### Scala安装

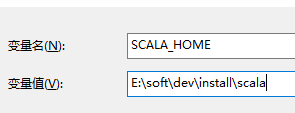
#### 下载地址:<https://www.scala-lang.org/download/> 在页面的最底部

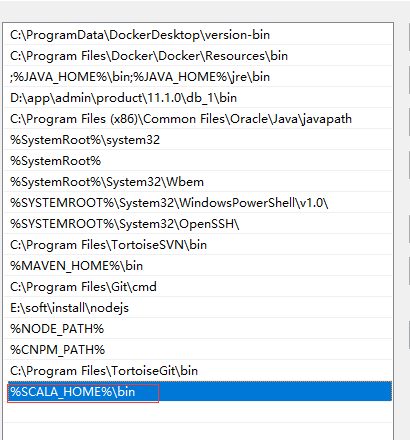


安装过程几乎没什么注意事项，网上说的安装路径不能有空格(如: Program Files),否则安装后使用会报错;仅参考，此处安装未使用含有空格的文件名称

#### 环境变量的配置

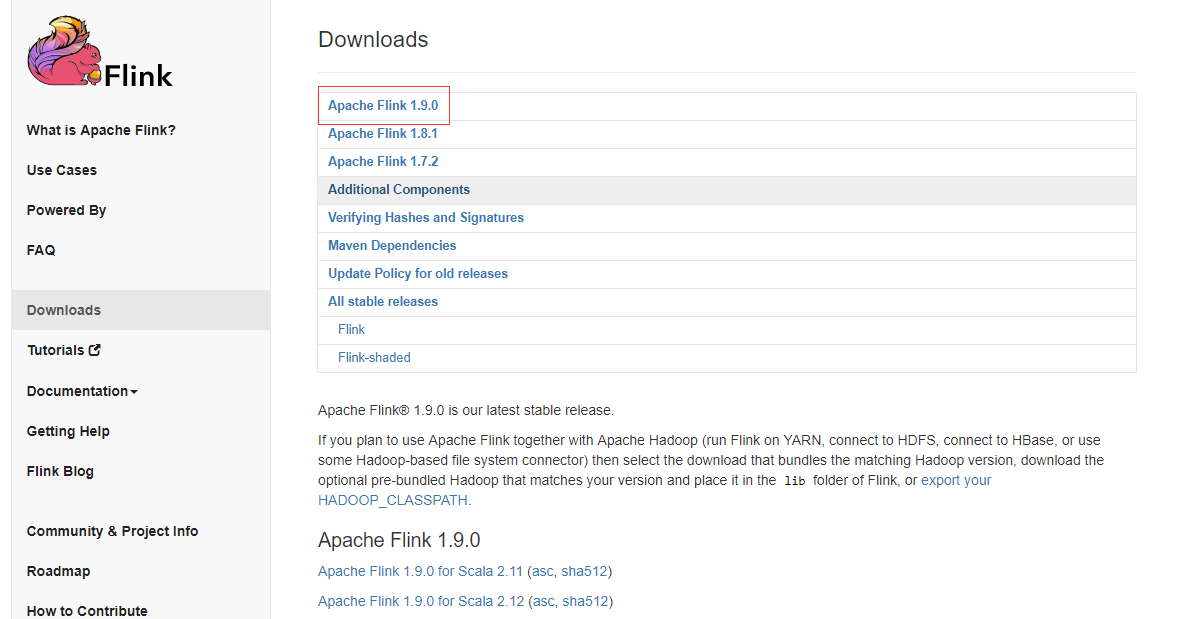
新增环境变量: SCALA\_HOME 值：E:\soft\dev\install\scala



Path地址配置: 

### Flink的下载(JDK1.8及之上)

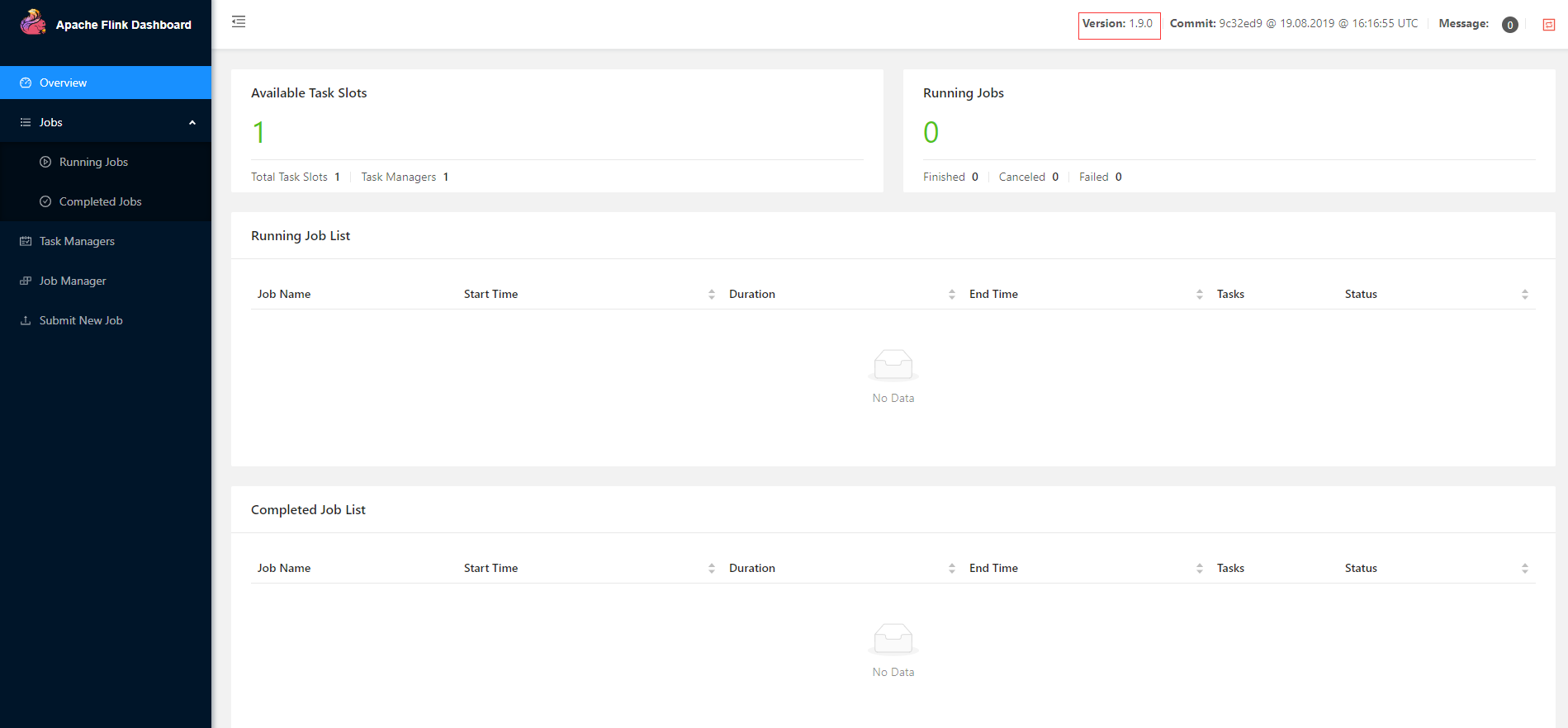
#### ① 下载地址: <https://flink.apache.org/downloads.html>



此处使用版本的是1.9.0

下载后直接解压可使用;

进入解压目录的bin目录，运行start-cluster.bat，启动成功后本地访问<http://localhost:8081>



### idea开发配置

#### 创建MAVEN项目（略）

Pom.xml文件：



只需要依赖flink-java和flink-streaming-java\_${sclac.version};其他的暂时不需要;

#### Scala插件安装

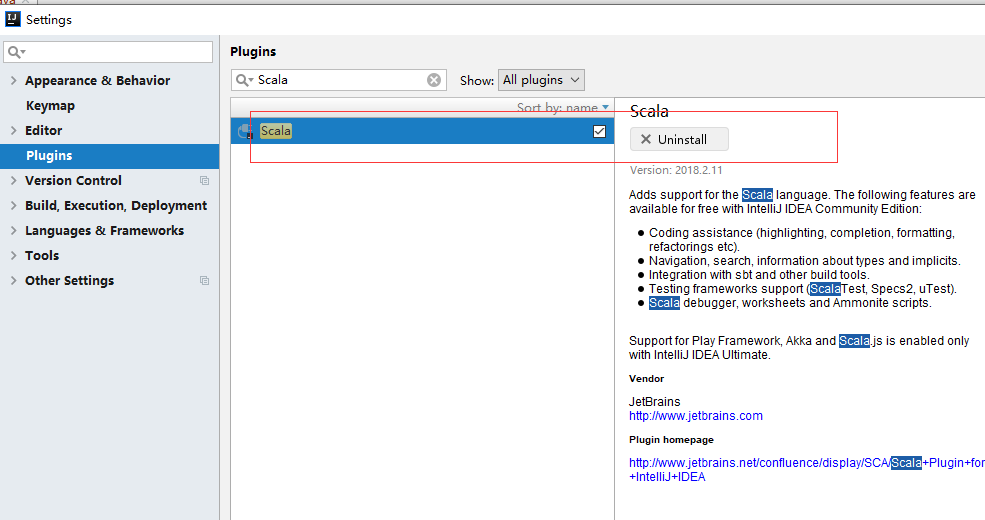
##### 在线安装：

File->Settings->Plugins->Install JetBrains plugins 然后输入Scala;然后安装(时间可能持久较长，视网络情况而定)—；

##### 离线安装：

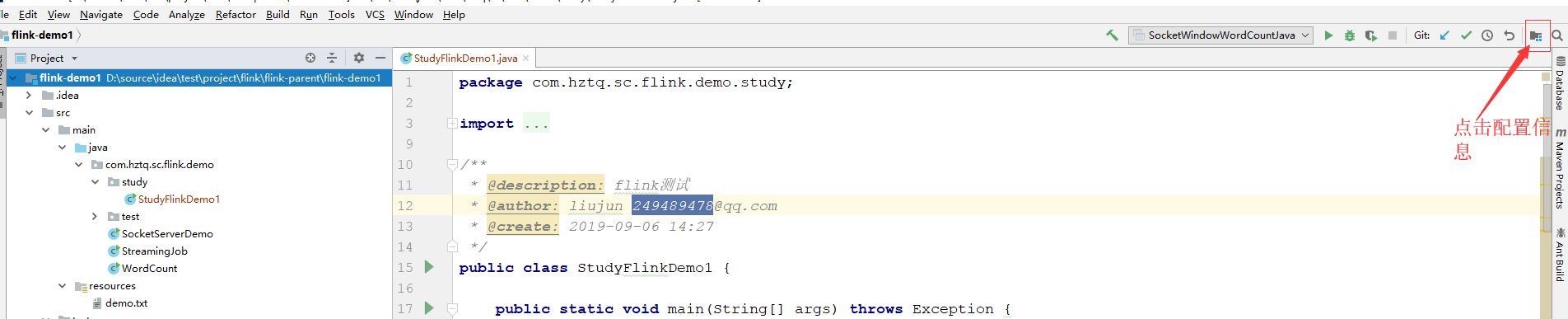
下载插件：[https://plugins.jetbrains.com/plugin/1347-scala 选择相应的版本(idea 、JDK)](https://plugins.jetbrains.com/plugin/1347-scala%20%20选择相应的版本(idea%20、JDK) )–>可用在线安装的方式查看版本;下载后将其解压，将解压的文件夹Scala放入idea安装目录的plugins目录下

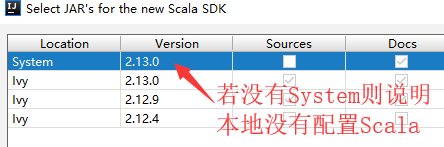
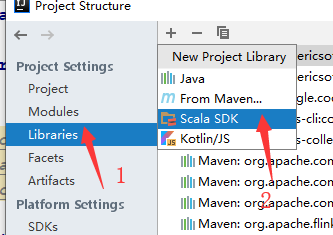
##### 安装完成后(在线安装):



#### 项目依赖配置

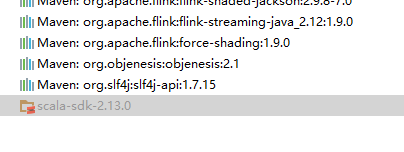
选择项目结构:(如下图)



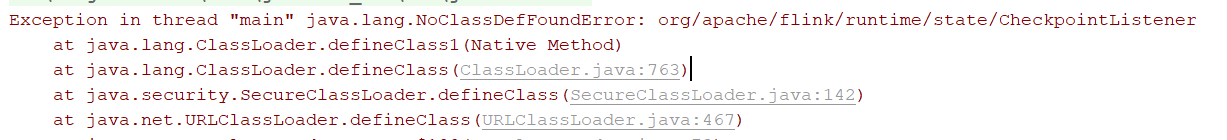


(如果没有Ivy则表示Scala插件安装不成功)

然后直接点击OK即可；最后会多一个scala-sdk



不要以为到此就可用运行了，我就在此处掉坑里了，获取运行环境的适合总是出错：

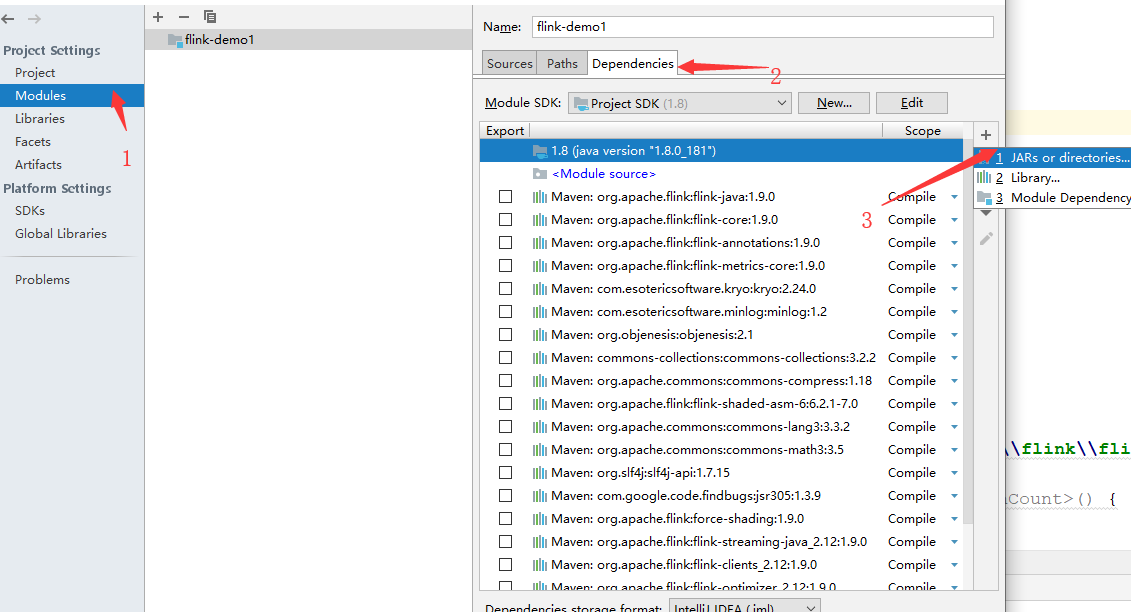


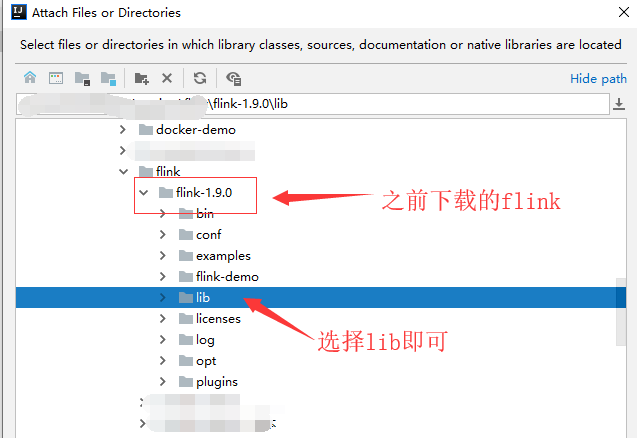
StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnvironment();

避免上面的代码出错，还相应运行环境的配置

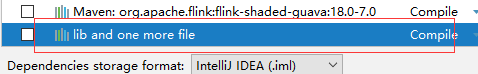
#### Flink在Idea的运行环境配置

选择项目结构:





点击OK,然后依赖中会多一个lib依赖

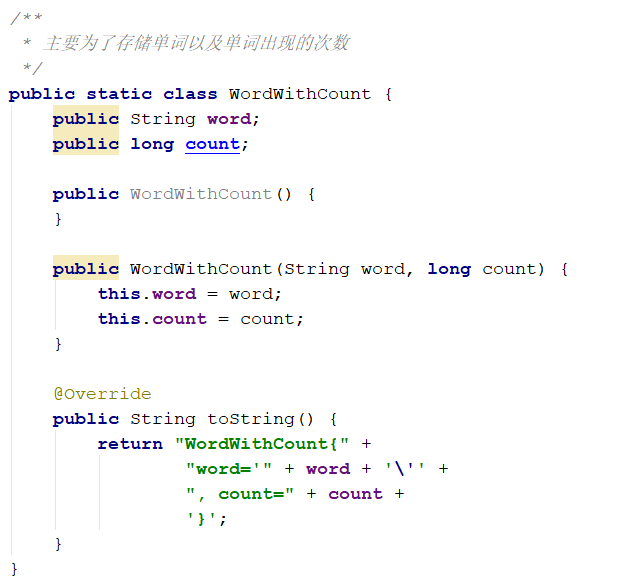


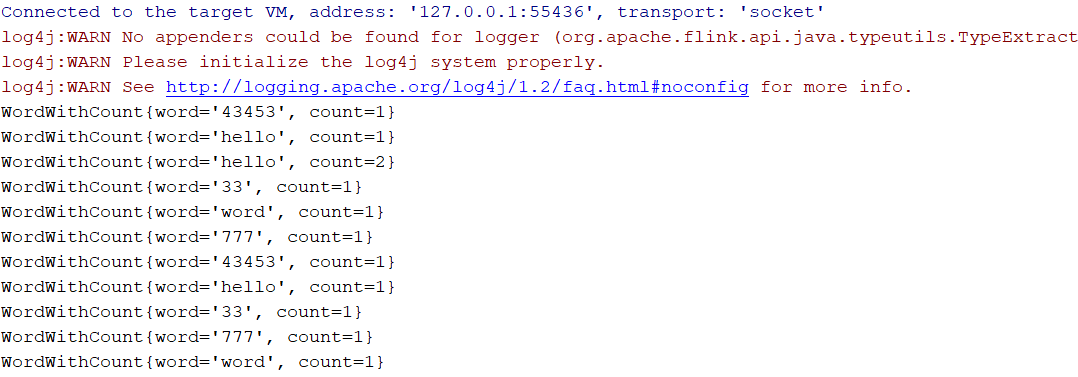
到此，idea可用运行flink了;

#### 运行验证

Java代码例子（网上找的）,网上是监听端口读取;偷懒一下直接读取文本







代码:

public static void main(String[] args) throws Exception {

//获取运行环境

StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnvironment();

//从文件中读取

DataStreamSource<String> text = env.readTextFile("D:\\source\\idea\\test\\project\\flink\\flink-parent\\flink-demo1\\src\\main\\resources\\demo.txt");

//计算数据

DataStream<WordWithCount> windowCount = text.flatMap(new FlatMapFunction<String, WordWithCount>() {

@Override

public void flatMap(String value, Collector<WordWithCount> out) throws Exception {

String[] splits = value.split("\\s");

for (String word : splits) {

out.collect(new WordWithCount(word, 1L));

}

}

})//打平操作，把每行的单词转为<word,count>类型的数据

.keyBy("word")//针对相同的word数据进行分组

.timeWindow(Time.seconds(2), Time.seconds(1))//指定计算数据的窗口大小和滑动窗口大小

.sum("count");

//把数据打印到控制台

windowCount.print().setParallelism(1);//使用一个并行度

//注意：因为flink是懒加载的，所以必须调用execute方法，上面的代码才会执行

env.execute("streaming word count");

}

/\*\*

\* 主要为了存储单词以及单词出现的次数

\*/

public static class WordWithCount {

public String word;

public long count;

public WordWithCount() {

}

public WordWithCount(String word, long count) {

this.word = word;

this.count = count;

}

@Override

public String toString() {

return "WordWithCount{" +

"word='" + word + '\'' +

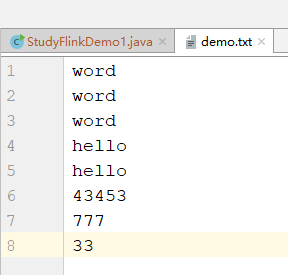
", count=" + count +

'}';

}

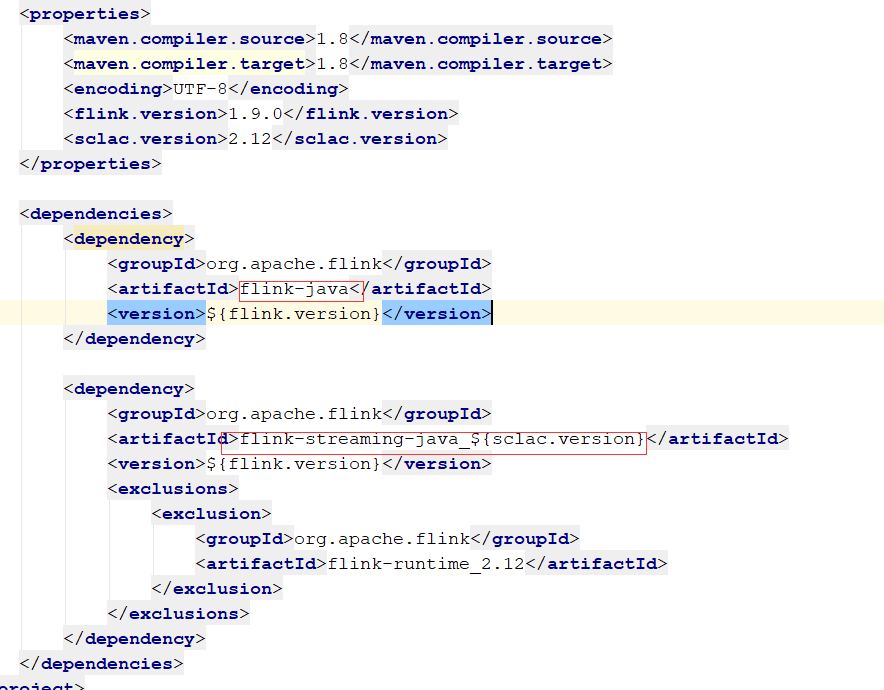
}

Demo.txt



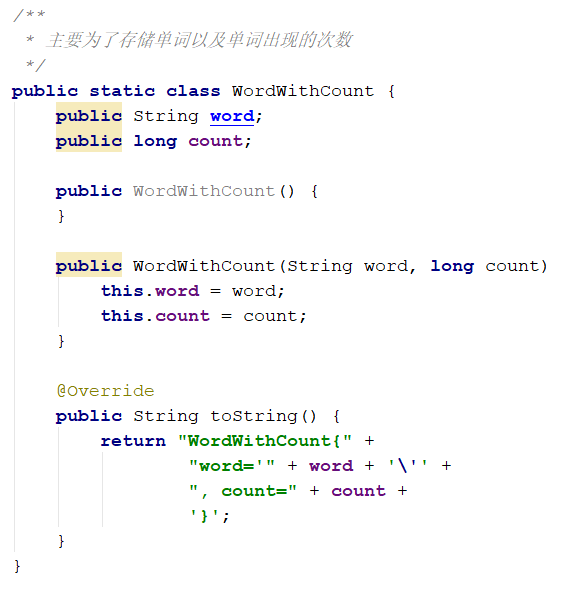
## 二．Flink 第一个demo

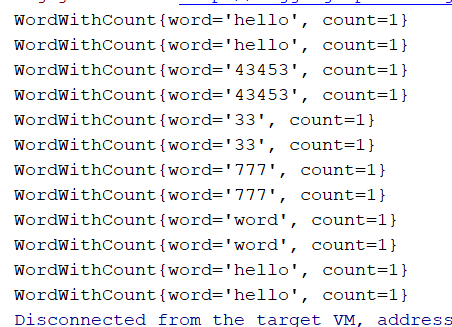
Pom.xml文件中只需要两个依赖包



Java代码





输出结果：

如果能够成功，则开发环境能够操作flink

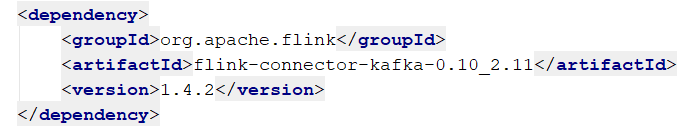
## 三．Flink-kafka整合(处理OGG数据)

### 1.ogg数据处理,大概流程图:

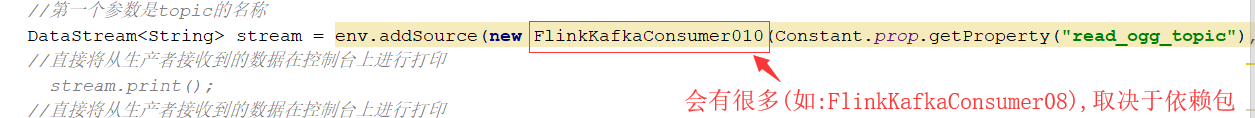


### 2.开发依赖说明

flink和kafka的整合需要一个关键依赖包：org.apache.flink

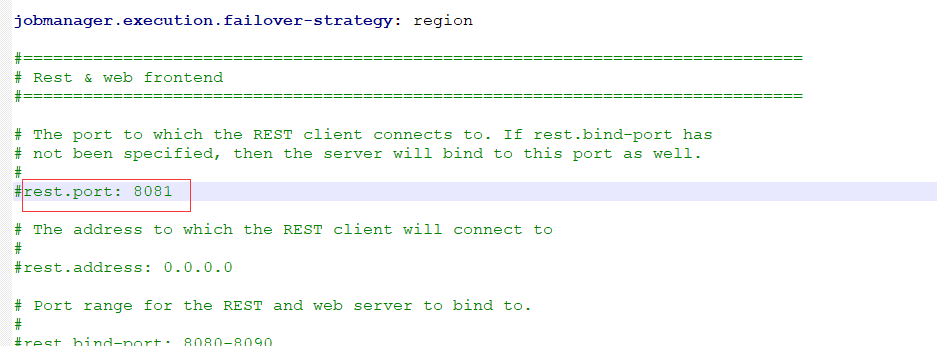


目前使用flink-connector-kafka-0.10\_2.11版本,不同版本会影响flink与kafka的操作工具类



## 部分配置说明

### 1. Flink web端口修改



修改conf文件下的flink-conf.yaml文件

Yaml文件中描述了端口（rest.port:）修改此属性则可用修改web的端口；默认8081