0. 前置要求

平台使用前置要求

- 已经注册了华为云的开发者平台，并登记了 ID 信息，获得了华为云代金券

完成作业前置要求

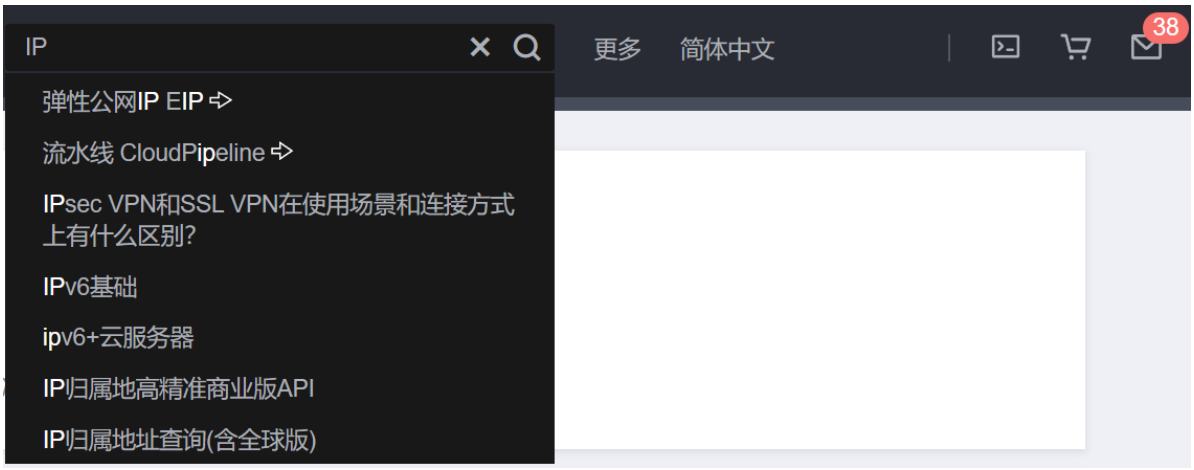
- 安装了 **Java**，jdk 环境为 **1.8 (Java 8)**
- 了解如何构建 **jar** 包，如何配置 **Maven** 环境
- 推荐使用 **IntelliJ IDEA**（非必须）

1. 资源购买

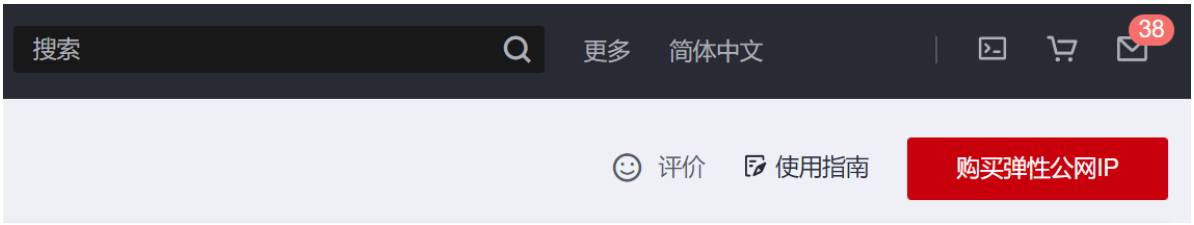
1.1 购买弹性公网 IP

登录华为云账号后，点击控制台进入控制台管理页面。

右上角搜索 IP，点击 `弹性公网IP EIP`。



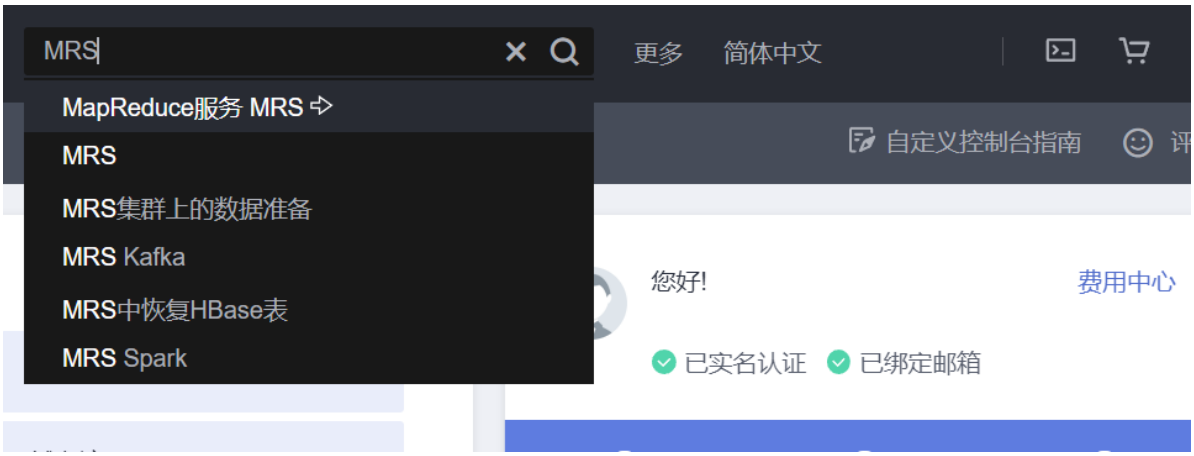
点击 `购买弹性公网IP`。



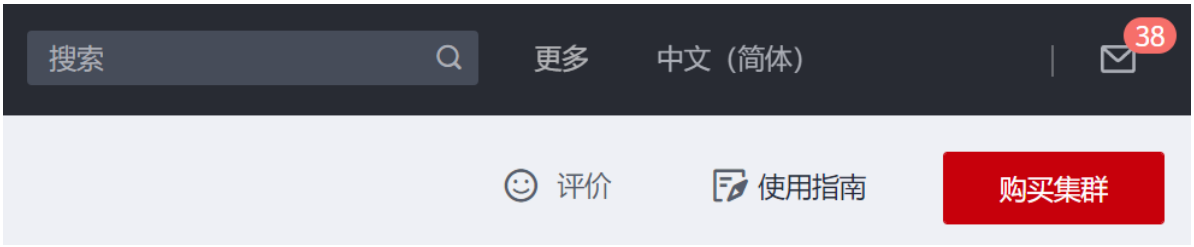
选择如图所示的配置：按需计费、全动态BGP、按流量计费等。

1.2 购买 MRS 集群

右上角搜索 MRS，点击 MapReduce服务 MRS。

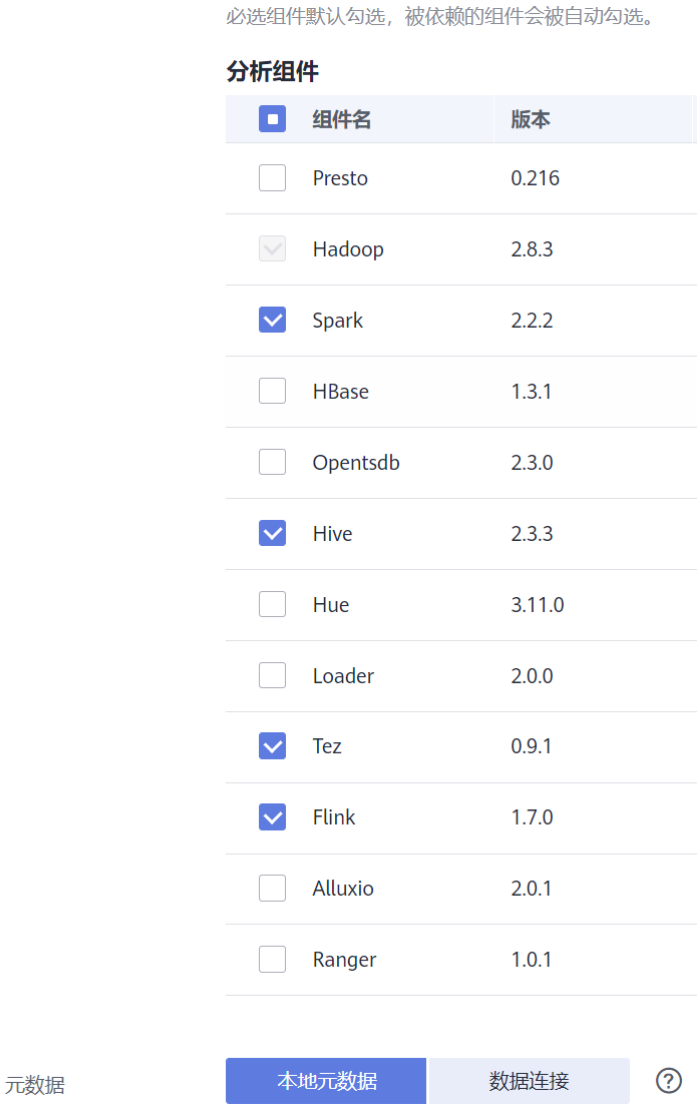


点击 购买集群。



1.2.1 软件配置

选择如下图所示的配置，**集群名称请填写自己的学号和姓名首字母。**



1.2.2 硬件配置

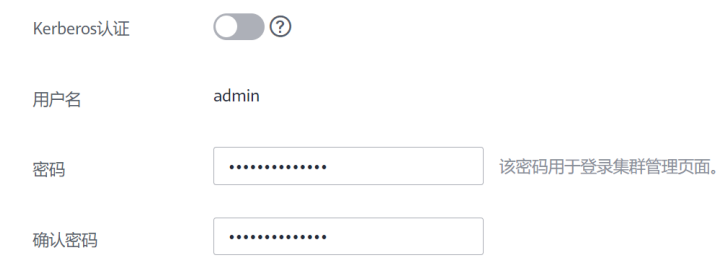
选择如下图所示的配置。

1.2.3 高级配置

`标签`、`主机名前缀`、`弹性伸缩`、`引导操作` 维持默认。

关闭 `告警`。

`Kerberos 认证` 选择关闭，设置 `MRS` 平台的登录密码。



设置服务器 root 用户的登录密码。

点击 确认授权，开通所有的访问规则。

购买成功后，系统将会自动创建集群，请耐心等待集群部署完成。

1.4 检查集群信息

集群配置成功后，概览信息如下图所示。

2. 配置相关资源

2.1 建立 OBS 并行文件系统

如前文所叙步骤，登录控制台，搜索 OBS，进入对象存储服务控制台。点击 并行文件系统 > 创建并行文件系统。

名称任意填写，并按如图所示配置创建一个并行文件系统。

2.2 创建委托策略

控制台处搜索 统一身份认证服务，进入管理页面后，点击左侧 委托。

创建如下图所示委托(委托名称任意)，点击下一步。

在弹出授权页面的搜索框内，搜索 `OBS OperateAccess` 策略，勾选 `OBS OperateAccess`。

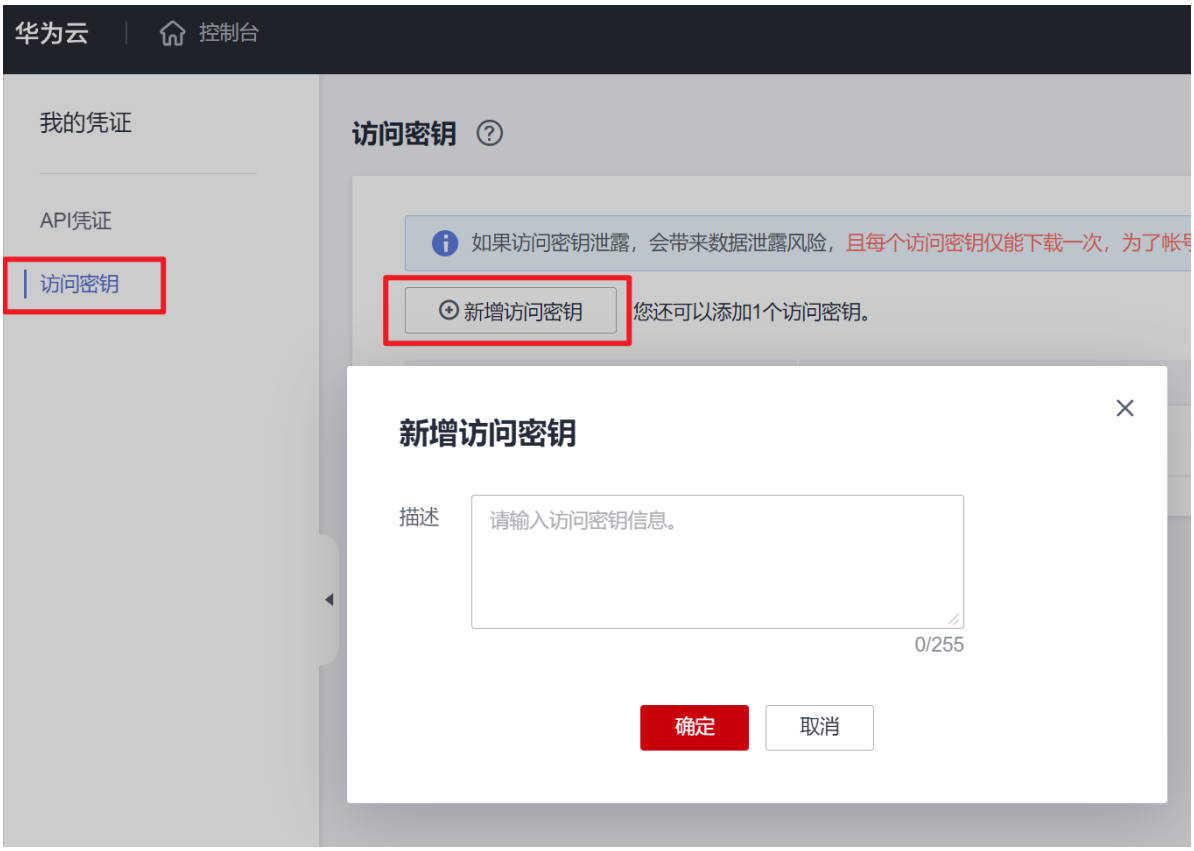
单击下一步，选择权限范围方案，默认选择 `所有资源`，单击 `展开其他方案`，选择 `全局服务资源` 。

在弹出的提示框中单击 `知道了`，开始授权。界面提示授权成功。单击完成，委托成功创建。

2.3 配置 OBS 访问密钥

点击网页右上角个人 ID，点击 `我的凭证`。

点击 `访问密钥`，点击 `+新增访问密钥`，创建一个访问密钥，**并及时下载密钥信息，该密钥信息无法再次获取**。下载后会得到一个名为 `credentials.csv` 的文件，里面记载了用户名，Access Key ID 和 Secret Access Key ID。

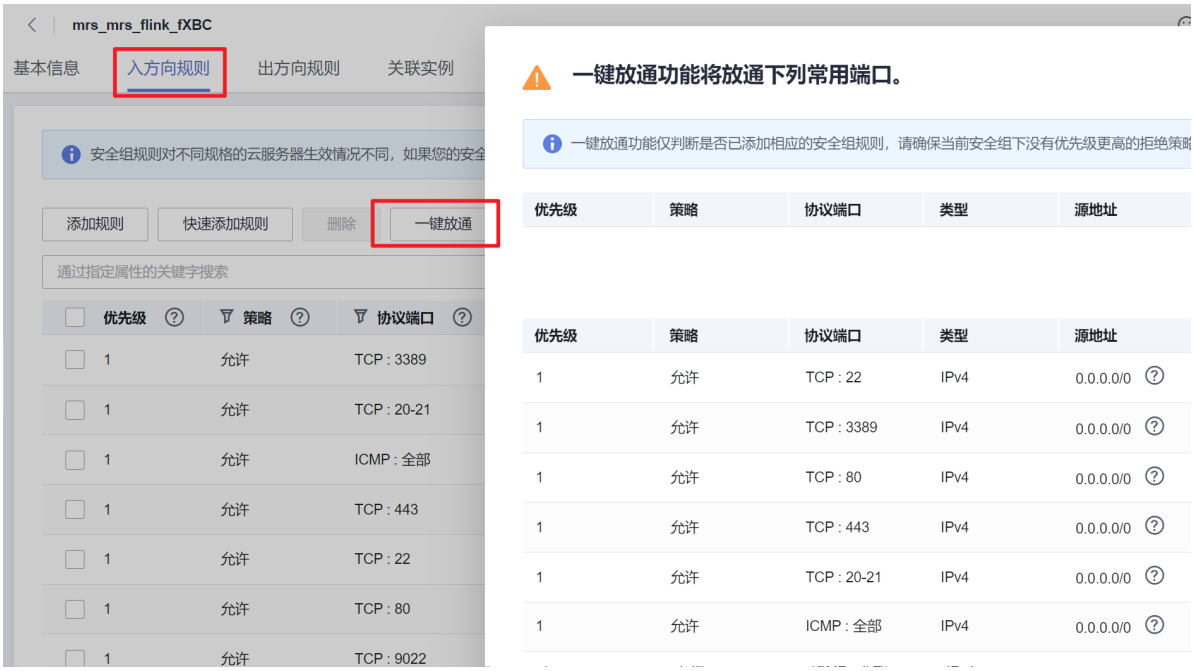


2.4 修改安全组策略

进入 MRS 控制台，进入节点管理页面，展开 master 节点，进入任意一个服务器节点。

点击安全组，点击下方的 配置规则。

点击 入方向规则，点击 一键放通，将如图所示的端口及IP源地址全部放通。



3. 运行样例程序

3.1 通过集群终端运行 Flink 样例程序 WordCount.jar

WordCount.jar 是 Flink 大数据处理系统的“Hello World”。它计算文本集合中单词的频率。该程序分两个步骤工作：首先，文本将文本拆分为单个单词；其次，对单词进行分组和计数。

本节将以 WordCount.jar 程序为例，说明如何通过 shell 指令创建并运行 Flink 作业。

3.1.1 上传数据至 OBS 文件系统

准备一个内容为英语段落的测试文本(文本中仅含大小写字母、空格、换行符)。这里给出一段测试文本作为样例，可以将以下内容复制进 txt 文本文件中：

> WordCount is the “Hello World” of Big Data processing systems. It computes the frequency of words in a text collection. The algorithm works in two steps: First, the texts are splits the text to individual words. Second, the words are grouped and counted.

进入 OBS 控制台页面，在“文件”页签下单击“新建文件夹”，分别新建 input、output 文件夹。在 input 文件夹中上传该 txt 文件。

以 OBS 系统名为 mrs-flink 为例，该 txt 文件的最终访问路径为 `obs://mrs-flink/input/input.txt`。

3.2 运行样例程序

如 2.4 节第一步所示，登录任意一台 master 服务器节点，点击 远程登录，并以 root 账号登录 shell 终端。**本节内容所提到的 OBS 文件系统中的文件名，路径等均为样例，实际操作时请自行对应修改相应的值。**

登录集群客户端节点，进入客户端安装目录。输入以下指令：

```
su - omm
cd /opt/client
source bigdata_env
```

执行以下命令验证集群是否可以访问 OBS。

```
hdfs dfs -ls obs://mrs-flink/input
```

进入 Flink 样例程序目录。

```
cd /opt/client/Flink/flink/examples/batch
```

提交 Flink 作业，指定源文件数据进行消费。其中，`obs.access.key` 和 `obs.secret.key` 为 2.3 节对应的 OBS 访问密钥。

WordCount.jar 输入参数为：`--input <path> --output <path>` 。

```
flink run -m yarn-cluster ./WordCount.jar  -Dfs.obs.access.key=XXXX -
Dfs.obs.secret.key=XXXX --input obs://mrs-flink/input/input.txt --output
obs://mrs-flink/output/output2.txt
```

若作业执行成功，则显示以下输出：

```
...
Cluster started: Yarn cluster with application id application_1654672374562_0011
Job has been submitted with JobID a89b561de5d0298cb2ba01fbc30338bc
Program execution finished
Job with JobID a89b561de5d0298cb2ba01fbc30338bc has finished.
Job Runtime: 12043 ms
```

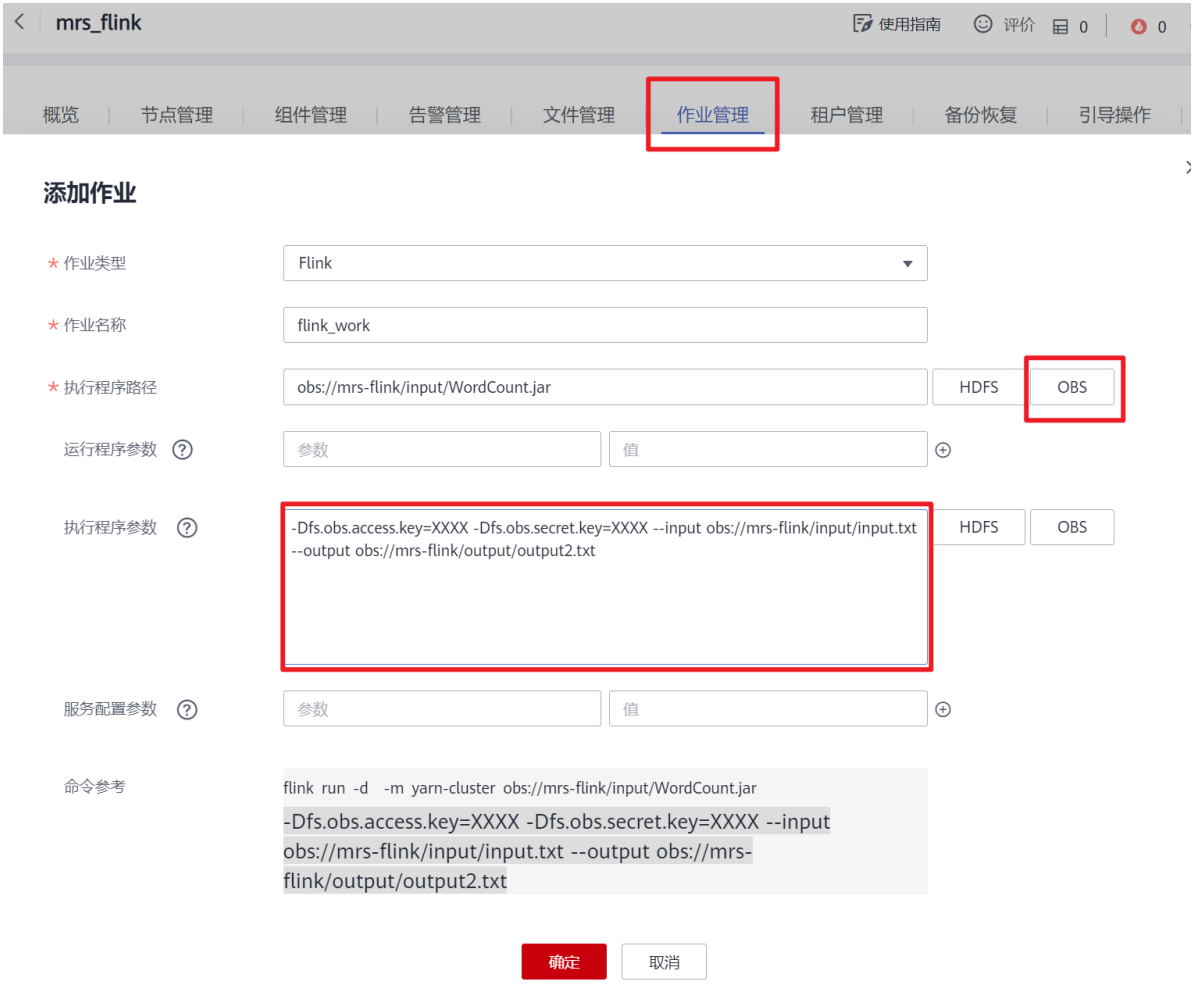
在 OBS 文件系统中指定的结果输出文件中可查看数据分析输出的结果。下载 output.txt，查看输出结果：

```
a 1
algorithm 1
and 1
are 2
big 1
collection 1

...

texts 1
the 6
to 1
two 1
wordcount 1
words 3
works 1
world 1
```

3.3 使用华为 MRS 平台提交 WordCount.jar 程序

进入 OBS 文件系统平台，上传附件中的 WordCount.jar 程序。

进入 MRS 平台，点击作业管理，添加作业，并进行如图所示的操作（相应参数自行修改）：



运行成功后，进入 OBS 平台可以看到 output/output2.txt 文件已经生成。

4. WordCount.jar 源码分析

代码如下：

```
package org.example;

import org.apache.flink.api.common.functions.FlatMapFunction;
import org.apache.flink.api.java.DataSet;
import org.apache.flink.api.java.ExecutionEnvironment;
import org.apache.flink.api.java.tuple.Tuple2;
import org.apache.flink.api.java.utils.ParameterTool;
import org.apache.flink.core.fs.FileSystem;
import org.apache.flink.util.Collector;


public class BatchJob {

    public static void main(String[] args) throws Exception {
        String input = null;
        String output = null;

        ParameterTool params = ParameterTool.fromArgs(args);

        try {
            input = params.getRequired("input");
```

```
            output = params.getRequired("output");
        } catch (RuntimeException e) {
            System.out.println("Argument Error");
            e.printStackTrace();

            return;
        }

        ExecutionEnvironment env =
ExecutionEnvironment.getExecutionEnvironment();

        env.setParallelism(1);

        DataSet<String> text = env.readTextFile(input);

        DataSet<Tuple2<String, Integer>> counts = text.flatMap(new
Tokenizer()).groupBy(0).sum(1);

        counts.writeAsText(output, FileSystem.WriteMode.OVERWRITE);


        env.execute("Flink Batch Java API Skeleton");
    }

    public static class Tokenizer implements FlatMapFunction<String,
Tuple2<String, Integer>> {

        @Override
        public void flatMap(String value, Collector<Tuple2<String, Integer>>
out) throws Exception {
            String[] tokens = value.toLowerCase().split("\\W+");

            for (String token : tokens) {
                if (token.length() > 0) {
                    System.out.println(token);
                    out.collect(new Tuple2<String, Integer>(token, 1));
                }
            }
        }
    }
}
```

Flink 程序的第一步是创建一个 `StreamExecutionEnvironment` 。这是一个入口类，可以用来设置参数和创建数据源以及提交任务：

```
ExecutionEnvironment env = ExecutionEnvironment.getExecutionEnvironment();
```

为了便于演示，将结果输出于同一个文件中，设置并行度为 1：

```
env.setParallelism(1);
```

这创建了一个字符串类型的 `DataSet`。`DataSet` 是 Flink 中做数据处理的 API，上面定义了非常多常见的操作(如，过滤、转换、聚合、窗口、关联等)。本示例中，首先要将字符串数据解析成单词和次数(使用`Tuple2<String, Integer>`表示)，第一个字段是单词，第二个字段是次数，次数初始值都设置成了 1。实现了一个 `flatmap` 来做解析的工作，因为一行数据中可能有多个单词。

即：处理数据，切分(flatMap，split)，分组(groupBy)，统计(累加 sum)。

```
DataSet<String> text = env.readTextFile(input);

DataSet<Tuple2<String, Integer>> counts = text.flatMap(new
Tokenizer()).groupBy(0).sum(1);
```

`flatMap` 实现的接口 `Tokenizer()` 如下：

```
public static class Tokenizer implements FlatMapFunction<String, Tuple2<String,
Integer>> {

    @Override
    public void flatMap(String value, Collector<Tuple2<String, Integer>> out)
throws Exception {
        String[] tokens = value.toLowerCase().split("\\W+");

        for (String token : tokens) {
            if (token.length() > 0) {
                System.out.println(token);
                out.collect(new Tuple2<String, Integer>(token, 1));
            }
        }
    }
}
```