Eclipse 开发环境搭建

刘磊

2020 年 8 月

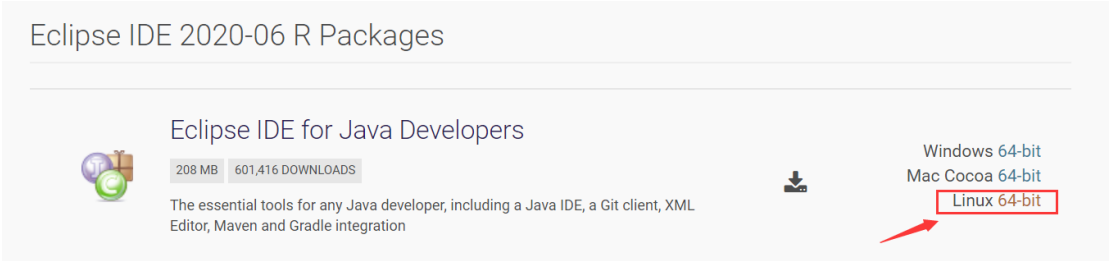
本节介绍开发 Hadoop 程序的 Eclipse 环境的搭建，包括 Eclipse 的安装、Hadoop 插件的安装、Hadoop 插件的使用以及统计词频实例 WordCount 的运行。

1. 安装 Eclipse

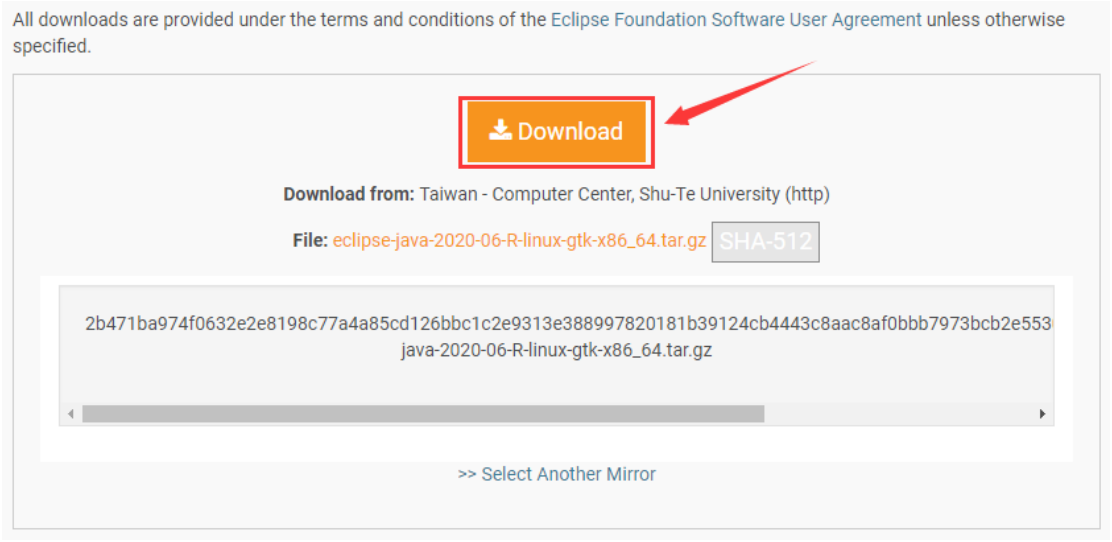
下载安装包

官网下载地址：https://www.Eclipse.org/downloads/packages/release/2020-06/r

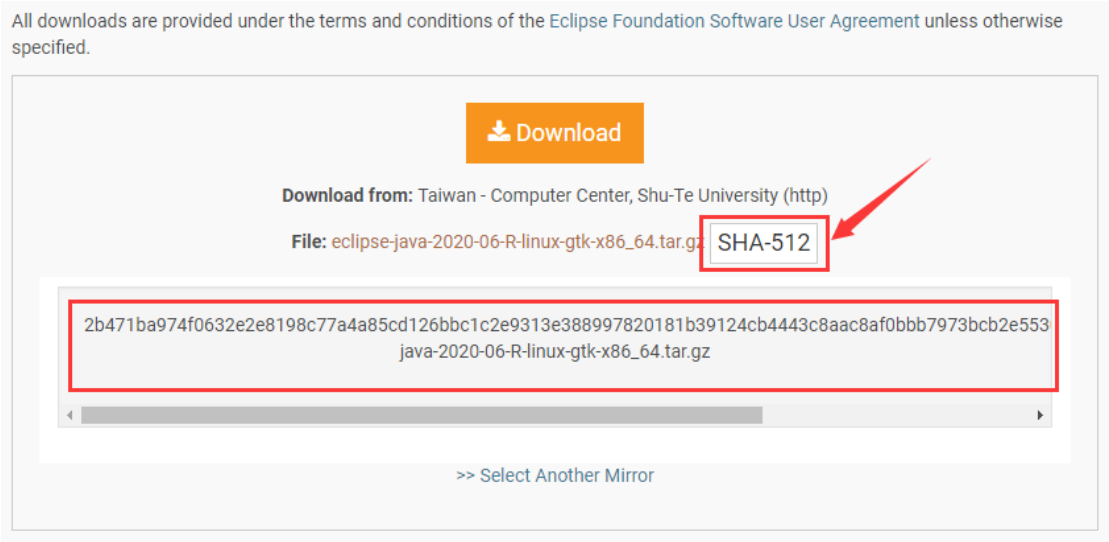
下载 Eclipse IDE for Java Develpers Linux 版本，点击【Linux 64-bit】。



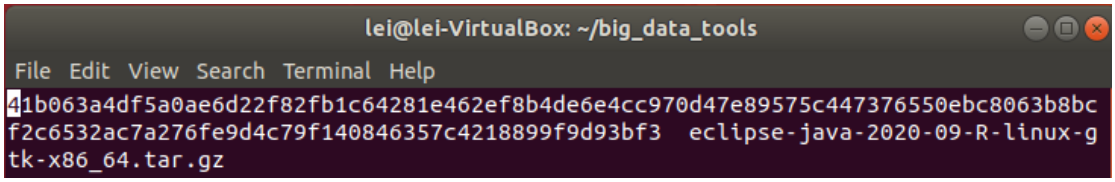
在弹出的页面，点击 Download 进行下载，保存到目录~/big_data_tools 中。



然后，返回上面的下载页面，点击 SHA-512，生成 sha512 散列值。



在~/big_data_tools 中创建文件 Eclipse.sha512 并将红框中的内容复制进去

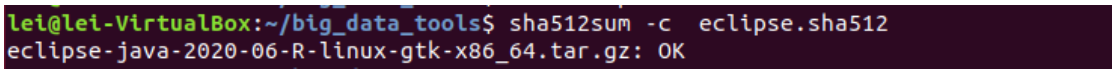


保存退出。

等安装包下载完成以后，在~/big_data_tools 目录下执行如下命令进行完整性校验。

```
sha512sum -c Eclipse.sha512
```

校验时，生成下载文件的 sha512 值，并和网站提供的值进行对比，如果一致，则返回 OK，否则返回错误信息。如果验证不一致，需要重新下载。



安装

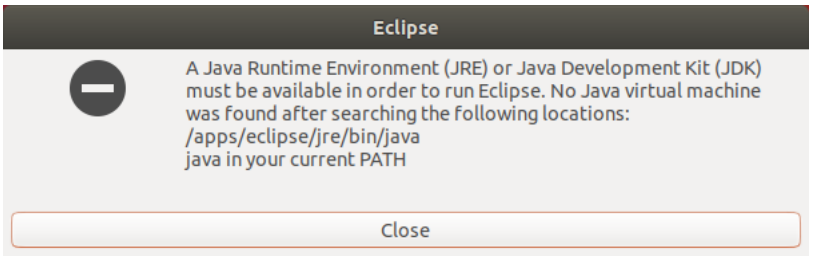
将下载的安装包 Eclipse-java-2020-06-R-linux-gtk-x86_64.tar.gz 复制到目录/apps 并解压。

```
cp ~/big_data_tools/Eclipse-java-2020-06-R-linux-gtk-x86_64.tar.gz /apps
tar zxvf Eclipse-java-2020-06-R-linux-gtk-x86_64.tar.gz
```

建立 jre 软连接

```
mkdir /apps/eclipse/jre
ln -s /apps/java/bin /apps/eclipse/jre/
```

不做这一步的话，Eclipse 无法启动，显示如下信息



创建图标

```
sudo vim /usr/share/applications/eclipse.desktop
```

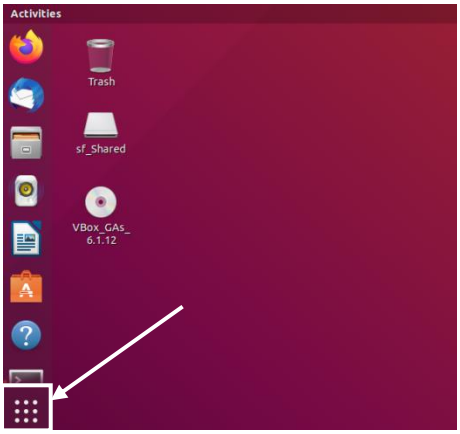
添加内容：

```
[Desktop Entry]
Encoding=UTF-8
Name=eclipse
Comment=eclipse
Exec=/apps/eclipse/eclipse
Icon=/apps/eclipse/icon.xpm
Terminal=false
StartupNotify=true
Type=Application
Categories=Application;Development;
```

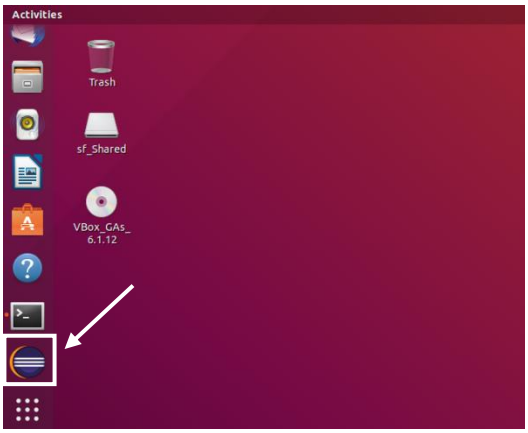
赋予图标文件可执行权限

```
sudo chmod u+x /usr/share/applications/eclipse.desktop
```

点击【显示应用】按钮，搜索 Eclipse，在 Eclipse 图标上右键选择添加到 Favorites。



就可以在 Favorites，找到 Eclipse 图标。



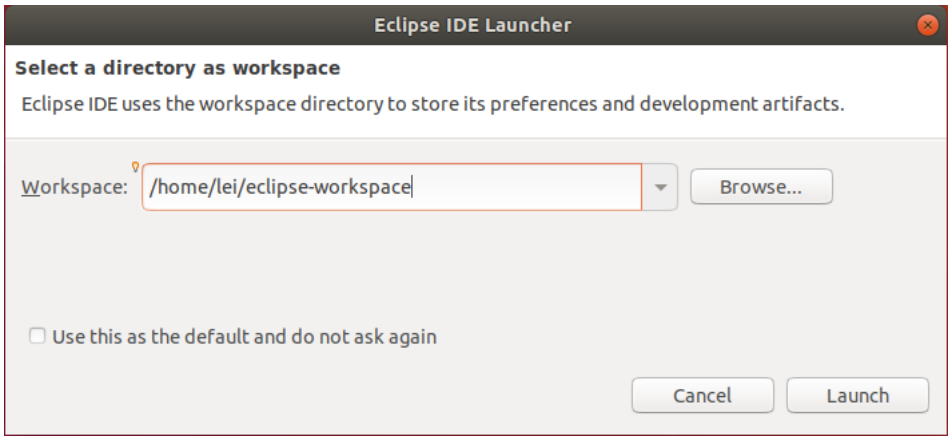
在 Favorites 中点击 Eclipse 图标，如果能够正常打开，说明安装成功。

2. 安装 Hadoop 插件

将插件 hadoop-Eclipse-plugin-2.6.0.jar 拷贝到/apps/Eclipse/plugins 插件目录下。

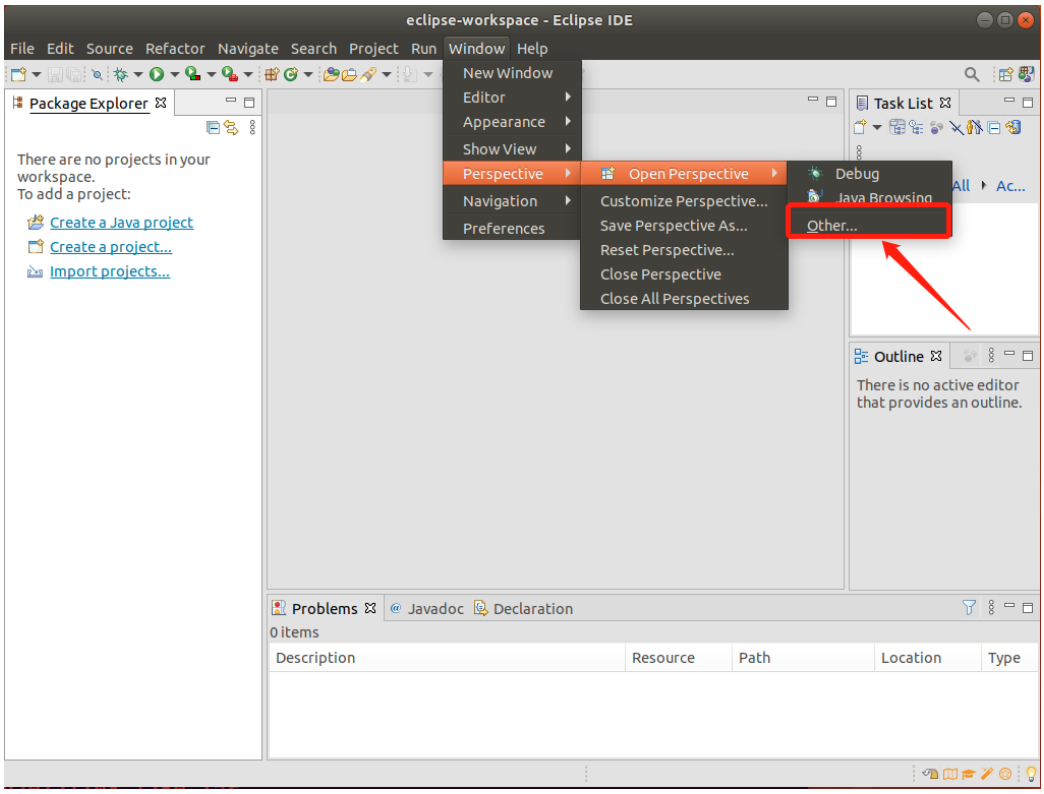
```
cp ~/big_data_tools/hadoop-eclipse-plugin-2.6.0.jar
/apps/eclipse/dropins/
```

打开 Eclipse，设置 Workplace 的路径，然后点击【Launch】。

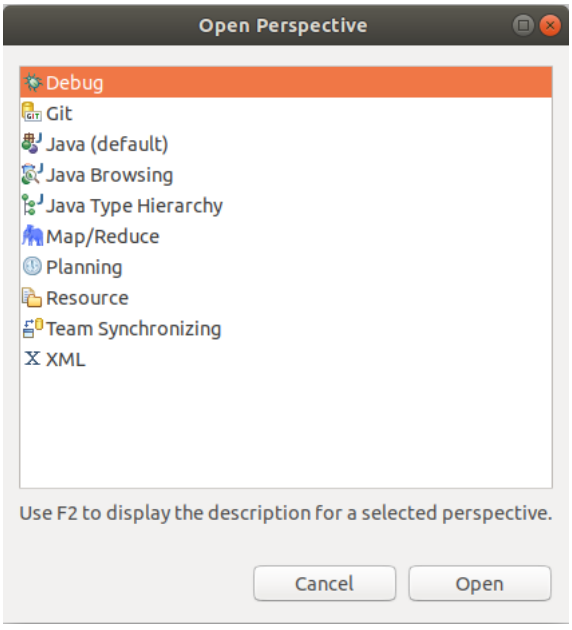


关闭 Welcome 界面，在 Eclipse 菜单中依次选择【Window】 => 【Perspective】 => 【Open Perspective】 => 【Other...】。

注：透视图（Perspective）是一个包含一系列视图和内容编辑器的可视容器。

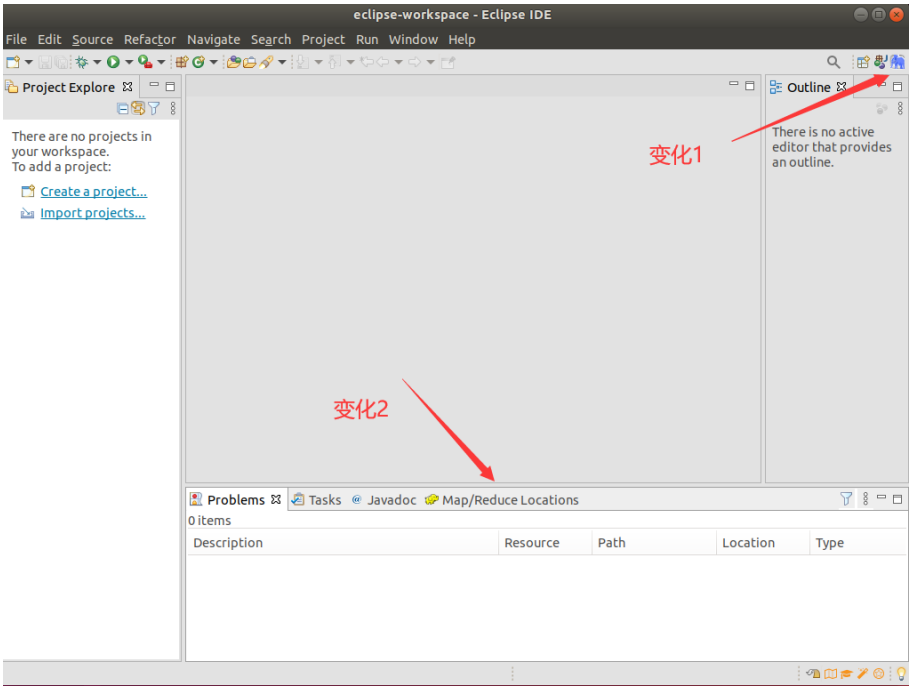


弹出如下图中的窗口。

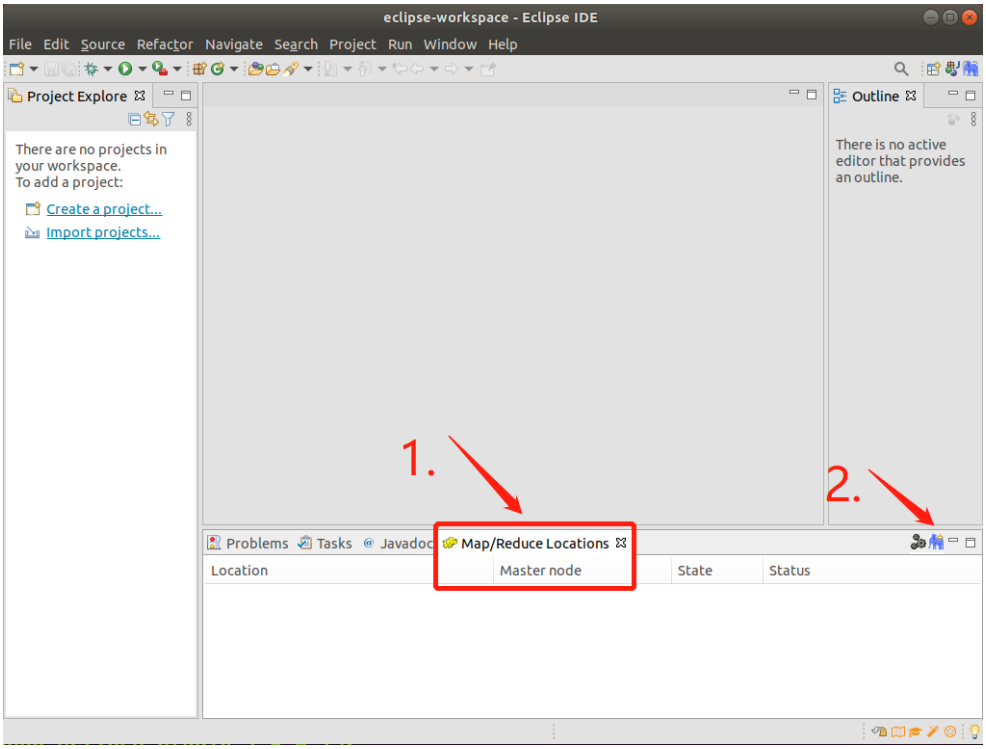


因为我们将 hadoop-Eclipse-plugin-2.6.0.jar 拷贝到了/apps/Eclipse/dropins/下，所以这里多出一下 Map/Reduce 选项。如果是在 Eclipse 打开的状态下拷贝的，需要重启一下才能显出 Map/Reduce 这一项。

选择【Map/Reduce】，并点击【Open】，可以看到窗口中，有两个变化。（右上角操作布局切换、面板窗口）



接下来，添加 Hadoop 配置，连接 Hadoop 集群。在下图中先选中 1，再点击 2.



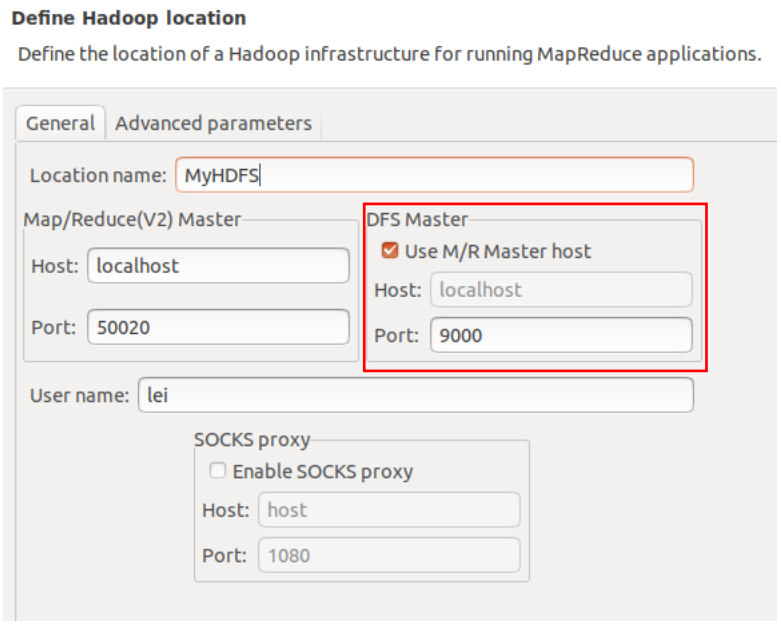
在弹出的窗口中，添加 Hadoop 相关配置。

Location name，是为此配置起的一个名字。

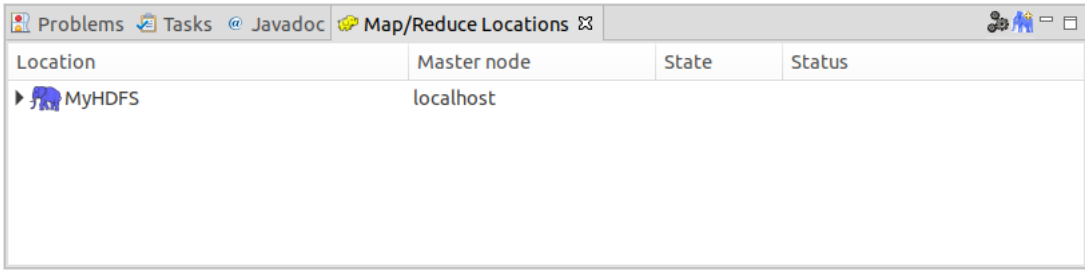
DFS Master，是连接 HDFS 的主机名和端口号。我们在安装 Hadoop 时，在 core-site.xml 文件中进行了设置。

```
<property>
    <name>fs.defaultFS</name>
    <value>hdfs://localhost:9000</value>
</property>
</configuration>
```

Host 为 localhost，端口为 9000。



点击 Finish 保存配置。配置好的 HDFS Location 就显示在下方的窗口中。



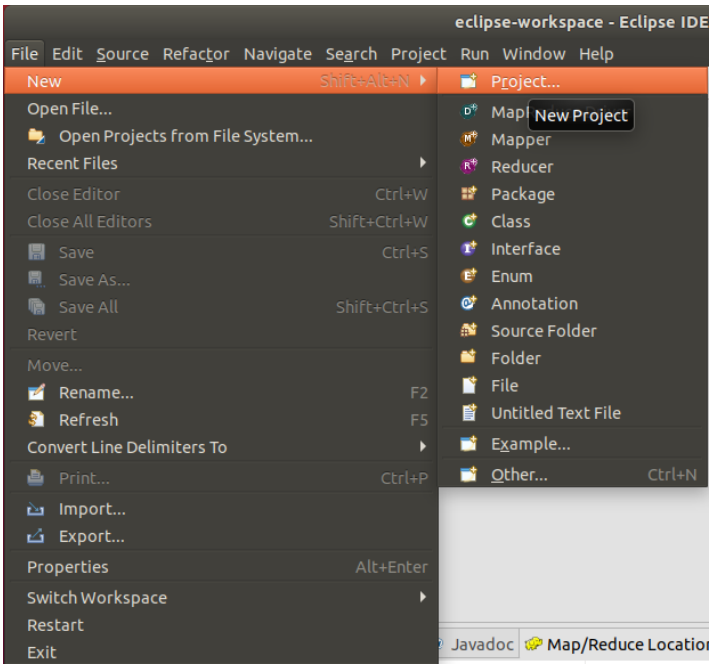
3. Hadoop 插件的使用

开启 HDFS 相关进程。在终端命令行输入 jps 查看进程状态。若不存在 hdfs 相关的进程，如 Namenode、Datanode、secondarynamenode，则需要先切换到 /apps/Hadoop/sbin 目录，启动 hadoop。

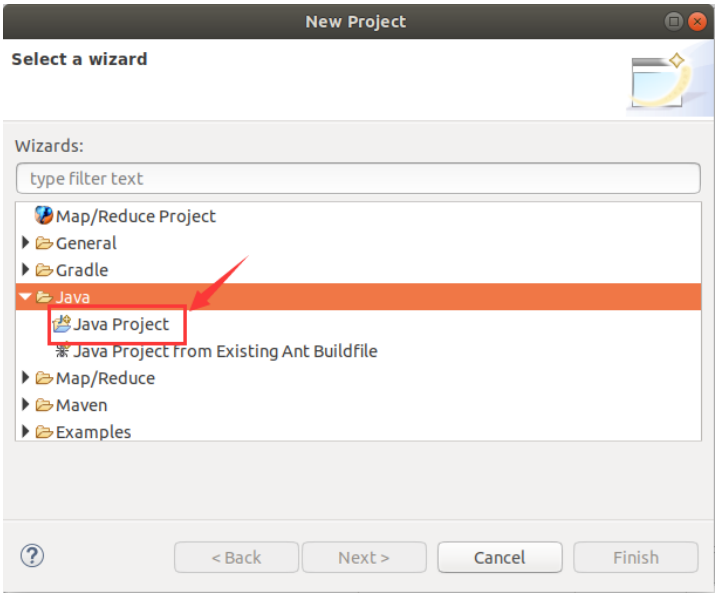
```
cd /apps/hadoop/sbin
```

```
./start-all.sh
```

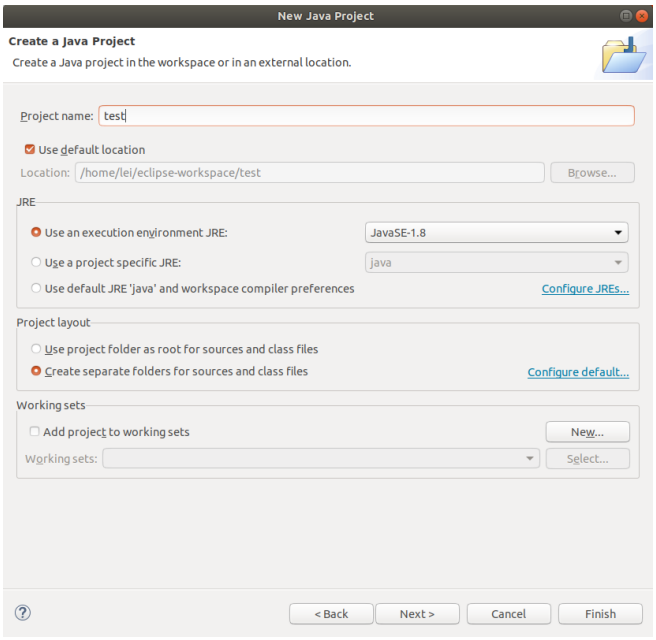
新建一个 Java 工程 test。点击【File】=>【New】=>【Project】，



在对话框中选择【Java Project】，点击【Next】。

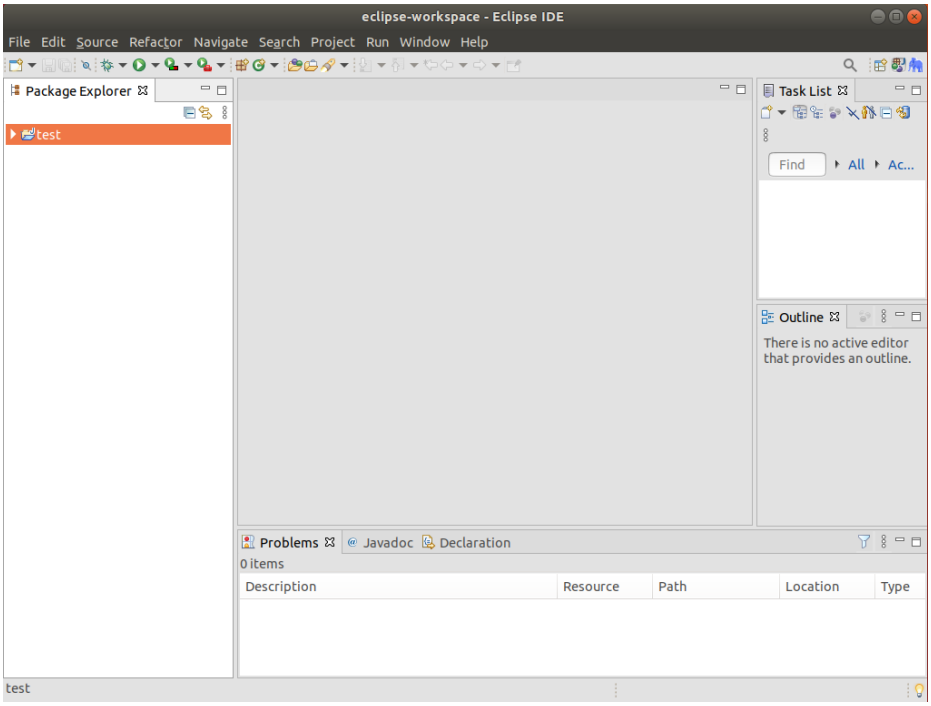


在对话框中输入【Project name】，点击【Finish】。



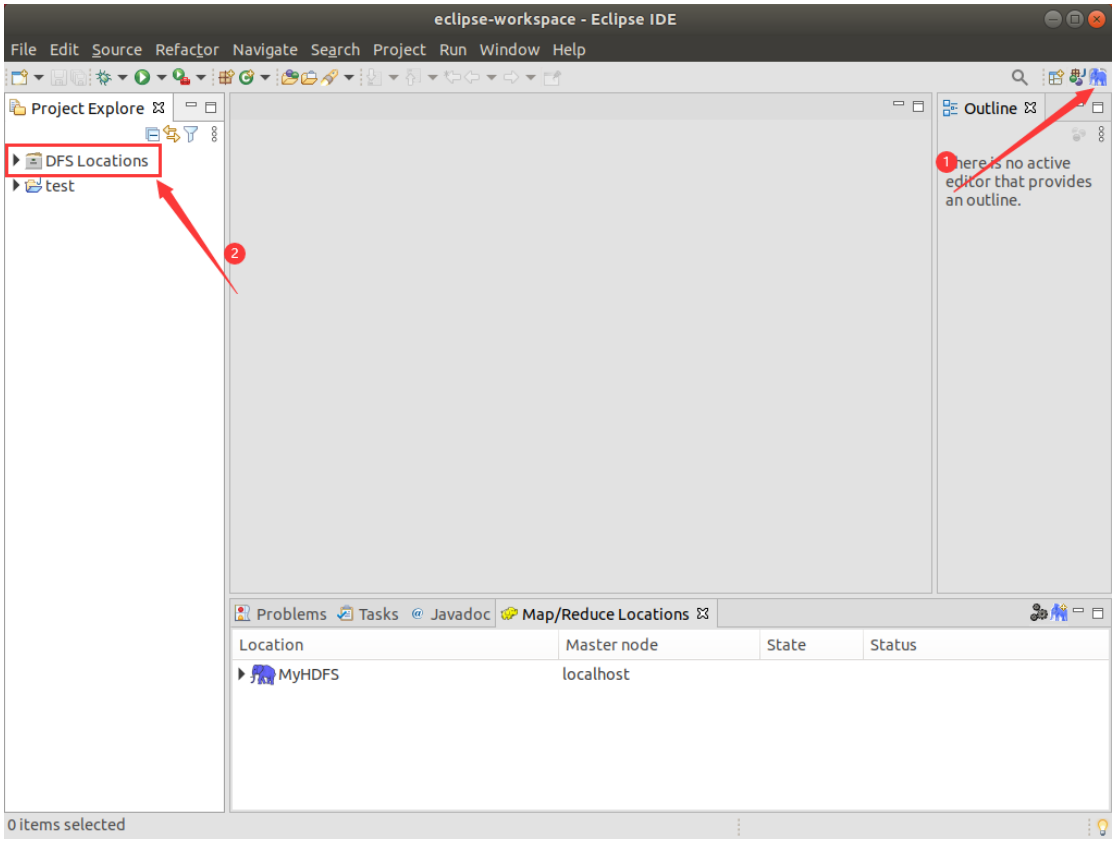
在弹出的窗口中点击【Open Perspective】。

新建的 Project 会显示在左侧的 Package Exploer 中。

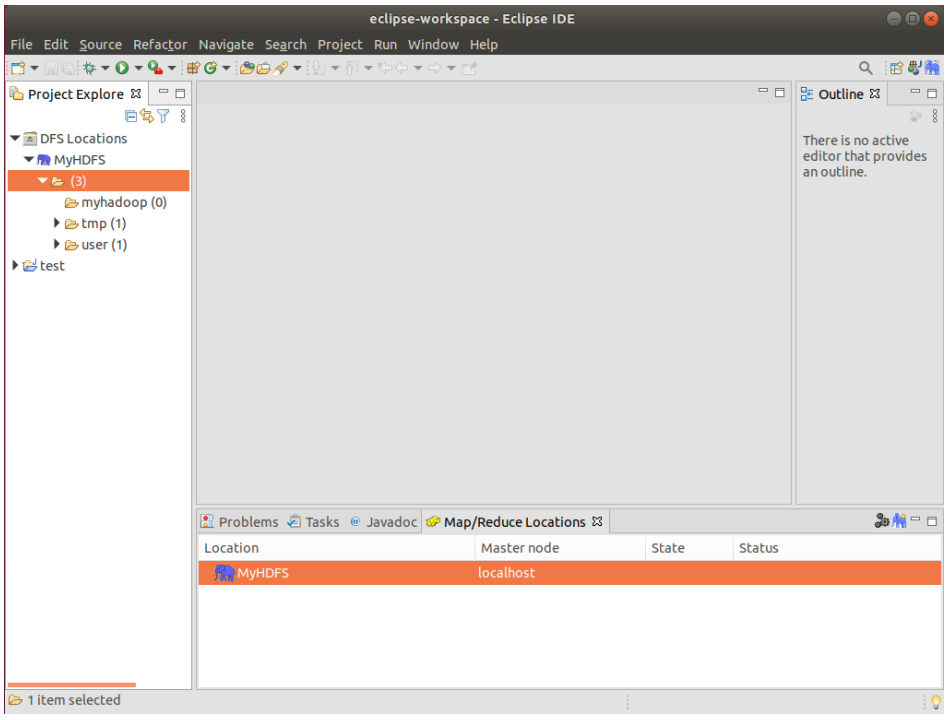


点击右上角的 mapreduce 图标，左侧的 Package Exploer 会切换到 Project Exploer。

DFS Location 也显示在这里。



点开 DFS Locations，可以看到我们 HDFS 上的目录。

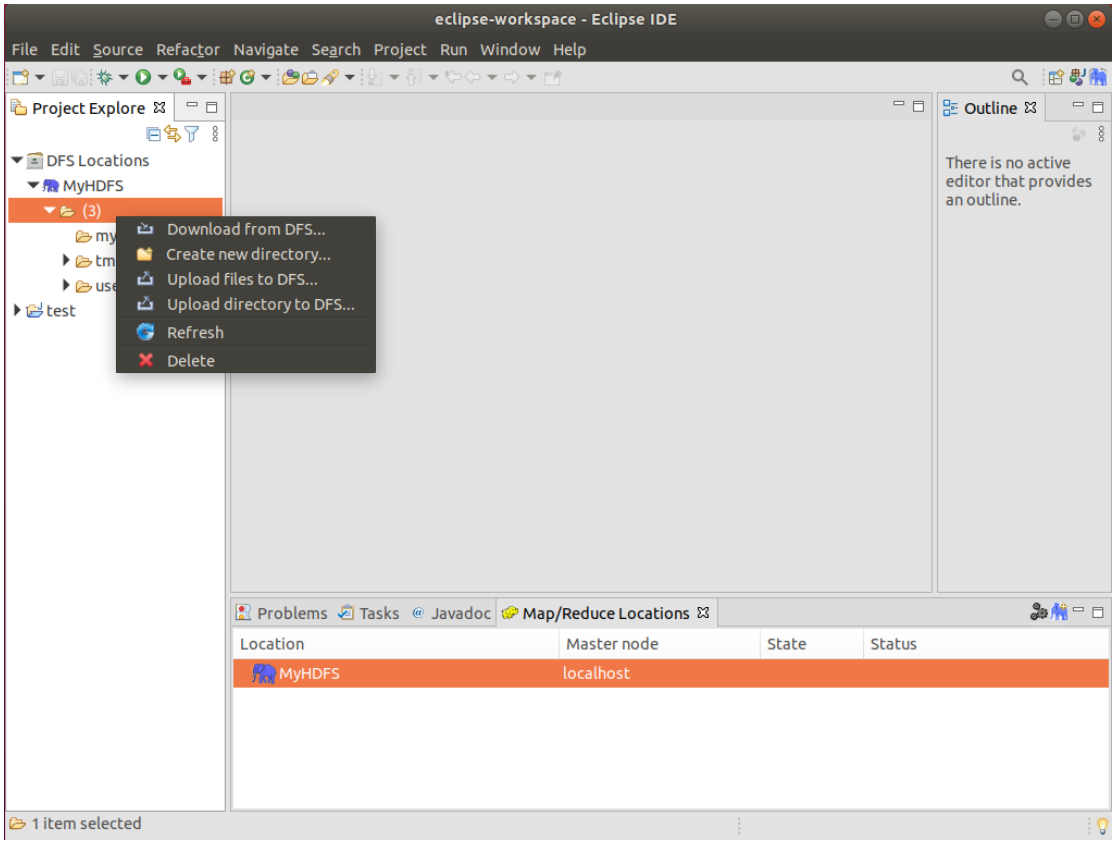


与我们使用命令查看的结果是一样的，说明设置成功。

```
lei@ubuntu:/apps/eclipse$ hadoop fs -ls /
Found 3 items
drwxr-xr-x   - lei supergroup          0 2020-08-25 23:50 /myhadoop
drwx------   - lei supergroup          0 2020-08-25 23:58 /tmp
drwxr-xr-x   - lei supergroup          0 2020-08-25 23:58 /user
```

右键 MyHDFS 下的文件夹，在弹出的菜单中，可以对 HDFS 进行操作，比如上传下载文件或目录，在 HDFS 上创建文件夹，删除文件或文件夹。

操作以后，右键，选择 Refresh，可用刷新 HDFS 目录。（如果刷新不出来，重启 Eclipse 即可）。



4. 运行 WordCount

准备数据文件

在 HDFS 上创建文件夹/input/wordcount

hadoop fs -mkdir /input/wordcount

在/data 目录下创建文件 testfile，

```
vim /data/testfile
```

并写入以下内容

```
hello big data
hello hadoop
hello hdfs
```

```
lei@ubuntu:~$ vim /data/testfile
lei@ubuntu:~$ cat /data/testfile
hello big data
hello hadoop
hello hdfs
```

将/data/testfile 上传到 HDFS 上/input/wordcount 目录。

```
hadoop fs -put /data/testfile /input/wordcount
```

```
lei@ubuntu:~$ hadoop fs -ls /input/wordcount
Found 1 items
-rw-r--r--   1 lei supergroup         39 2020-09-21 23:20 /input/wordcount/testf
ile
```

我们统计文件/input/wordcount/testfile 出现的词的词频。

准备 jar 包

在~/big_data_tools 目录下创建文件夹 hadoop3libs

mkdir ~/big_data_tools/hadoop3libs

将/apps/hadoop/share/hadoop 目录下的 common，hdfs，mapreduce，yarn 4 个子

目录中的 jar 文件以及这 4 个子目录下 lib 文件夹中的所有 jar 文件复制到

~/big_data_tools/hadoop3lib 中。这些 jar 文件将作为外部的 jar 文件添加到工程中。

完成代码

创建 Project【mr_example】，在项目中创建 Package【sds.mapreduce】，然后创建

Class【WordCount】，将在中间窗口中自动打开 WordCount.java 文件，删除其中的代码，将下面的代码复制其中并保存。

```
package sds.mapreduce;

import java.io.IOException;
import java.util.StringTokenizer;

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.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;

public class WordCount {

        public static class TokenizerMapper extends Mapper<Object, Text,
Text, IntWritable> {

                private final static IntWritable one = new IntWritable(1);
                private Text word = new Text();
                public void map(Object key, Text value, Context context)
throws IOException, InterruptedException {
                        StringTokenizer itr = new
StringTokenizer(value.toString());
                        while (itr.hasMoreTokens()) {
                                word.set(itr.nextToken());
                                context.write(word, one);
                        }
                }
        }

        public static class IntSumReducer extends Reducer<Text, IntWritable,
Text, IntWritable> {
                private IntWritable result = new IntWritable();
                public void reduce(Text key, Iterable<IntWritable> values,
Context context) throws IOException, InterruptedException {
                        int sum = 0;
```

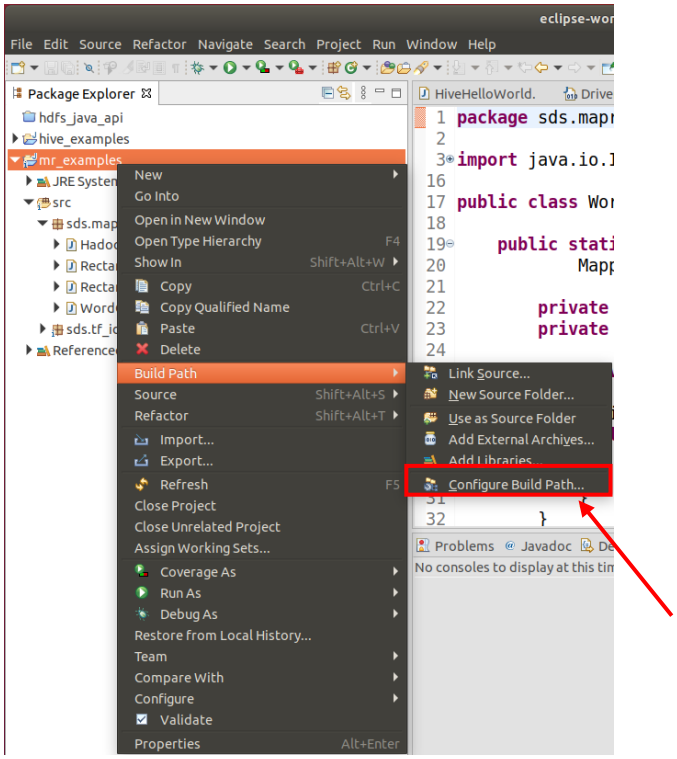
```
                        for (IntWritable val : values) {
                                sum += val.get();
                        }
                        result.set(sum);
                        context.write(key, result);
                }
        }

        public static void main(String[] args) throws Exception {
                Configuration conf = new Configuration();
                String[] otherArgs = new GenericOptionsParser(conf,
args).getRemainingArgs();
                if (otherArgs.length < 2) {
                        System.err.println("Usage: wordcount <in> [<in>...]
<out>");
                        System.exit(2);
                }
                Job job = Job.getInstance(conf, "word count");
                job.setJarByClass(WordCount.class);
                job.setMapperClass(TokenizerMapper.class);
                job.setCombinerClass(IntSumReducer.class);
                job.setReducerClass(IntSumReducer.class);
                job.setOutputKeyClass(Text.class);
                job.setOutputValueClass(IntWritable.class);
                for (int i = 0; i < otherArgs.length - 1; ++i) {
                        FileInputFormat.addInputPath(job, new
Path(otherArgs[i]));
                }
                FileOutputFormat.setOutputPath(job, new
Path(otherArgs[otherArgs.length - 1]));
                System.exit(job.waitForCompletion(true) ? 0 : 1);
        }
}
```

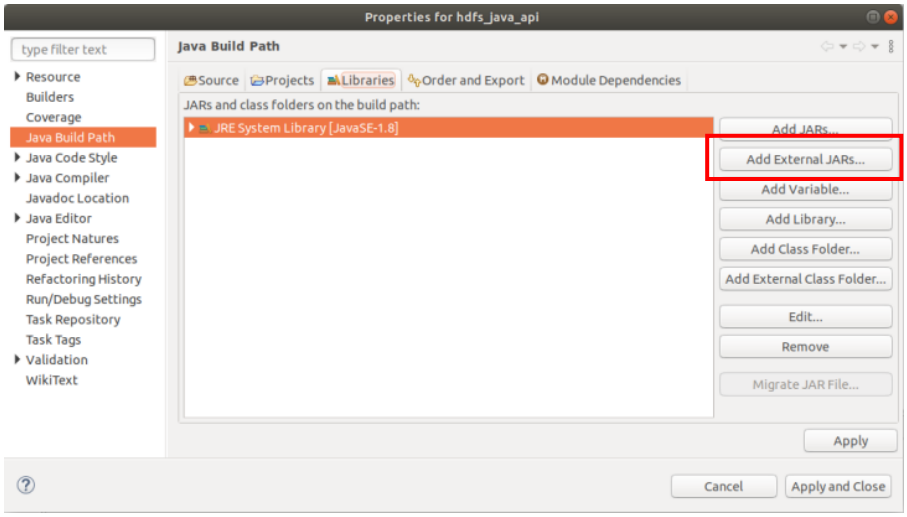
添加外部 jar 文件

在 Package Explorer 工程名 mr_example 上点击右键，选择【Build Path】=>

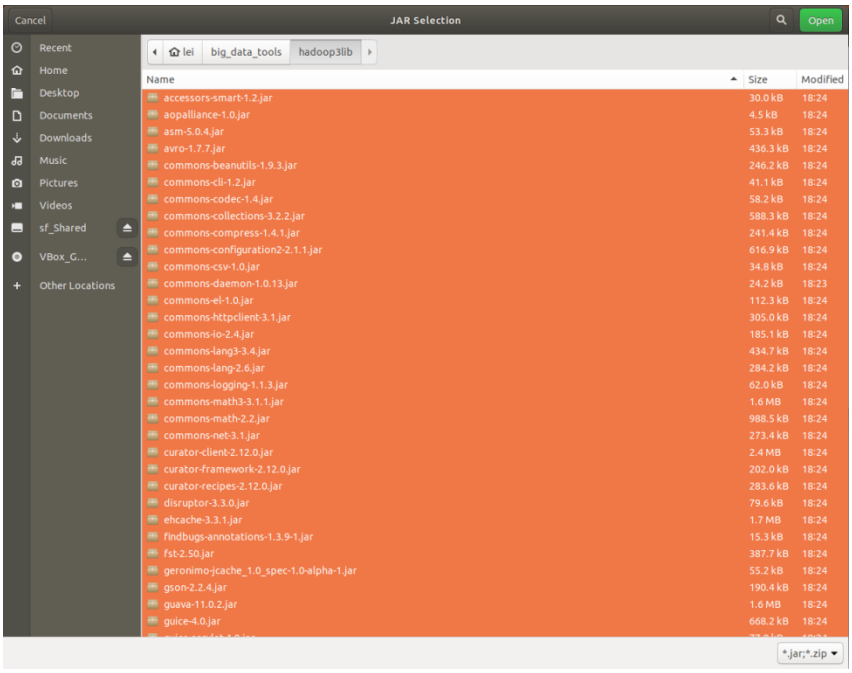
【Configure Build Path...】。



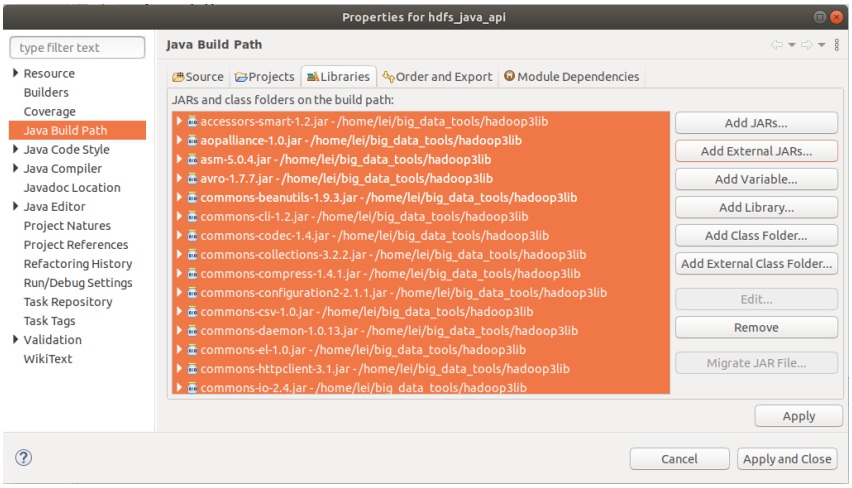
在【Java Build Path】对话框中，选择【Libraries】标签，点击右侧的【Add External JARs...】按钮。



在弹出的 JAR 文件选择窗口，切换到目录~/big_data_tools/hadoop3libs，全选目录中的 JAR 文件。点击【Open】。

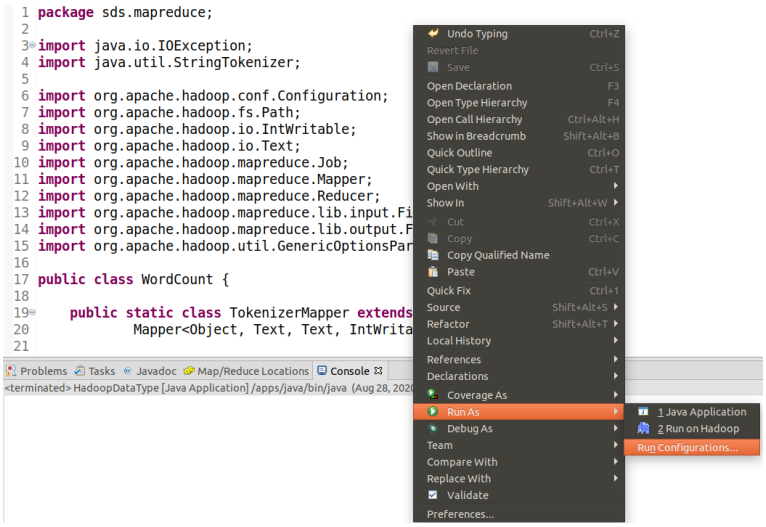


回到【Java Build Path】对话框，点击【Apply and Close】，完成添加 jar 包。



运行代码

因为代码中设置了从命令行获取参数，所以运行时，需要提供参数的值。在 Eclipse 中点击右键，选择【Run As】=>【Run Configurations...】，

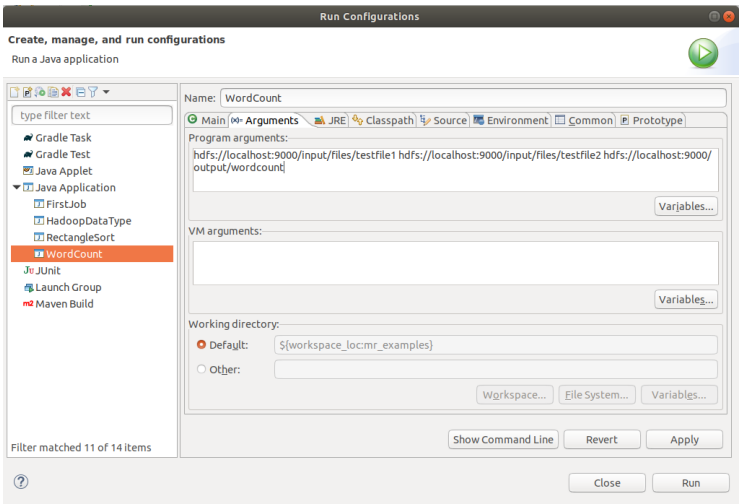


在弹出的【Run configurations】对话框中，从左边的列表中选择我们要运行的 Java Application 【WordCount】。然后在右边选择【Arguments】标签，在【Program arguments】中提供需要的参数值，也就是我们要统计词频的输入文件和存放结果的输出目录，各个参数用空格隔开。**注意如果输出目录在 HDFS 上已存在，需要先删除，否则会报错。**

hdfs://localhost:9000/input/wordcount/testfile

hdfs://localhost:9000/output/wordcount

这里 HDFS 上的地址需要写全，前面需要加上 hdfs://localhost:9000/。设置好以后点击【Run】。



运行结束以后，我们可以在【Project Explorer】中的【DFS Locations】中看到，在

/output 目录下生成了子目录/wordcount，双击打开其中的文件 part-r-00000，可以看到词频统计结果与我们预期的结果是一致的。