**第二章 Spark开发环境**

[2.1 Linux环境准备 1](#_Toc15818)

[2.1.1 搭建ubuntu 1](#_Toc20604)

[2.1.2共享文件夹 5](#_Toc5235)

[2.2 Spark快速部署 11](#_Toc27868)

[2.2.1 快速搭建Spark 11](#_Toc28007)

[2.2.2第一个Spark实例 13](#_Toc10850)

[2.2.3 SparkWEB使用 14](#_Toc25546)

[2.3 IDEA 16](#_Toc8697)

[2.3.1 IDEA安装 16](#_Toc13797)

[2.3.2 IDEA的第一个实例（Scala+Maven） 22](#_Toc9638)

[2.3.3 IDEA的第二个实例（Scala） 30](#_Toc20229)

[2.3.4 IDEA打包运行 36](#_Toc14197)

[2.4 小结 40](#_Toc9516)

Spark支持在Windows和Linux上进行部署和开发，本文只详细介绍基于Linux的Spark开发环境的安装步骤。本章将依次介绍Linux环境准备、JDK配置、Spark-shell环境配置以及使用IDEA开发并生成jar包，通过spark-submit提交运行。参考本章，完成Spark开发环境的配置，在此基础上进行进一步的应用开发。

**2.1 Linux环境准备**

Linux是开源操作系统。Linux发行版本众多，总体分为两类，一类是商业公司维护的发行版本，以CentOS为代表，一类是社区组织维护的发行版本，以Debian和Ubuntu为代表。Ubuntu的特点是界面友好、容易上手，是最适合做桌面系统的Linux发型版本。

如果读者使用的操作系统是Linux，那么可以选择跳过本节，直接学习2.2中的配置。本节将介绍在Windows平台上，使用VMware Workstation虚拟机构建Linux（Ubuntu）环境以及必要的环境准备（共享文件夹）。

2.1.1 搭建ubuntu

本文以VMware 14为例，搭建ubuntu系统。首先从网络上下载免费的ubuntu镜像文件。下载地址：<http://mirrors.163.com/ubuntu-releases/>。此地址中，有多个版本的ubuntu镜像可以选择，如图2-1所示。

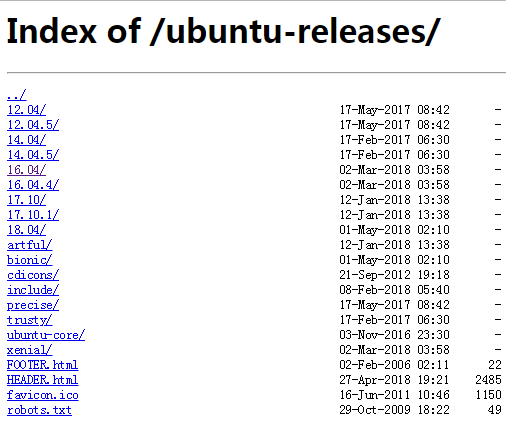


图2-1 ubuntu镜像版本选择

本文中选择16.04版本。点击版本号进入下载页面，如图2-2所示。根据主机安装的Windows操作系统架构（64位/32位），选择对应（amd64/i386）版本的镜像。

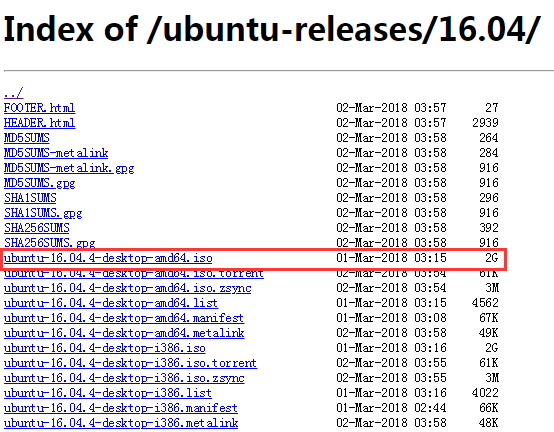


图2-2 ubuntu镜像下载

按照上述步骤镜像下载完成以后，打开VMware，

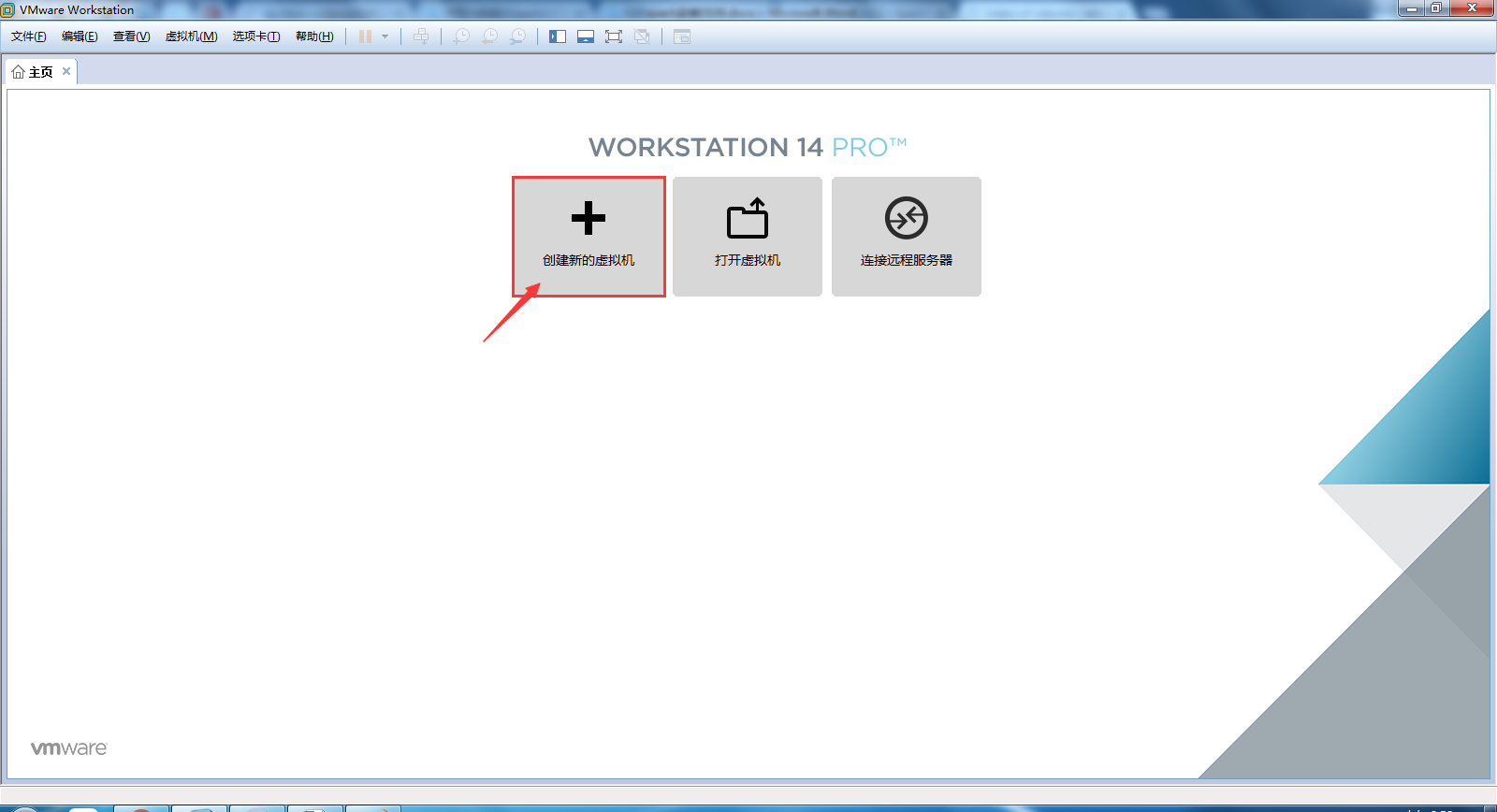


图2-3 Vmware界面

选择上图中的【创建新的虚拟机】，出现如下窗口：

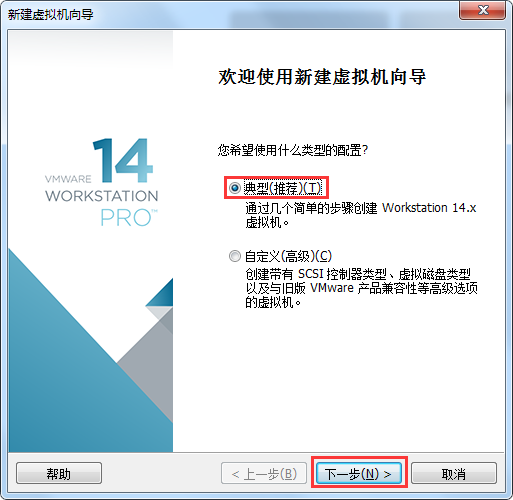


图2-4 新建虚拟机的界面

默认类型为【典型】，点击【下一步】，进入如下窗口：

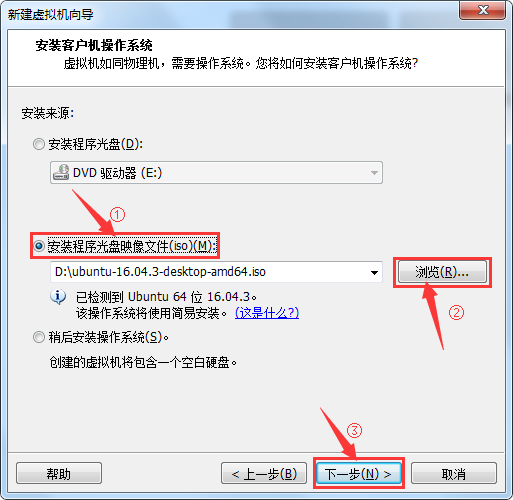


图2-5 安装镜像文件的界面

选择【安装程序光盘映像文件】，点击【浏览】，找到镜像存放位置，然后点击【下一步】，出现如下图所示窗口：



图2-6 设置名称的界面

设定虚拟机名称、用户名以及密码，点击【下一步】，进入虚拟机存放位置设置窗口。虚拟机存放位置可选放在C盘以外的任意盘中，设置完成后，点击【下一步】，进入如下窗口：

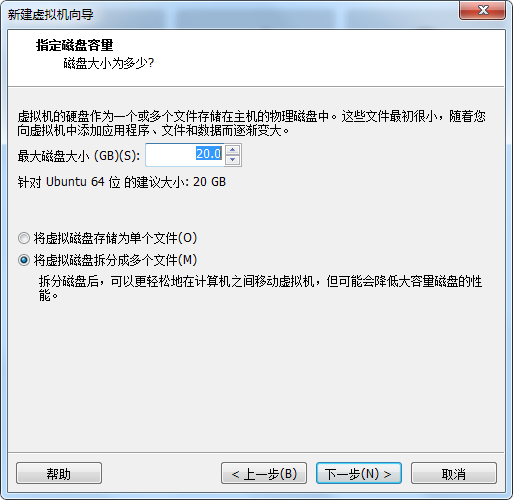


图2-7 设置磁盘大小的界面

设定虚拟机的磁盘容量，默认20GB，选择【下一步】，进入最后一步，如下图所示：



图2-8 自定义硬件的界面

选择【自定义硬件】：设置内存大小。

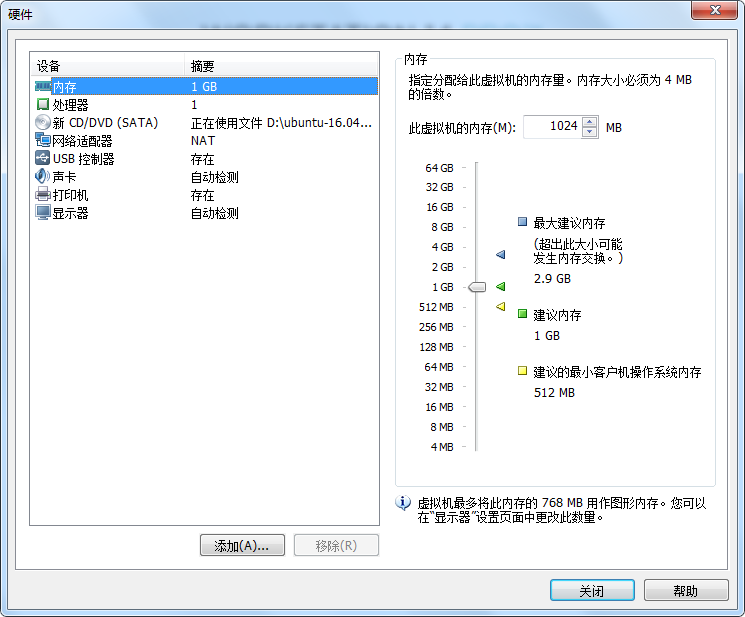


图2-9 设置内存的界面

建议设置内存为1.5G及以上，IDEA的使用需要很大内存，如果开辟比较小，影响IDEA使用的流畅度。

设置完成，点击【关闭】，然后点击【完成】。Linux安装过程自动化程度很高，不需要用户干预，直接等待安装完成即可。到此Spark环境所需要的Ubuntu系统便已搭建完成（安装过程需要等待10分钟左右）。搭建完成以后，启动Ubuntu系统。

2.1.2共享文件夹

虚拟机中下载Spark环境配置所需要的安装包时，经常因为网络问题导致下载失败。而主机中的下载速度更加快速、稳定。因此，本文选择了在主机下载安装包，然后通过共享文件夹，Ubuntu系统复制安装包到指定位置进行下一步安装配置。设置共享文件夹。具体步骤如下:

1. 安装vmware-tools。

点击虚拟机菜单栏中的【虚拟机】，然后选择【重新安装VMware Tools】。在此步骤中，可能此选项为灰色（无法选中）。解决方法为：点击菜单栏中【虚拟机】，然后选择【设置】，进入如图2-11界面。在【硬件】栏下，依次设置框中【CD/DVD】、【CD/DVD 2】、【软盘】的连接属性（图2-11中右侧框中所示），选择【使用物理驱动器】，并设置为【自动检测】，最后点击【确认】。



图2-10 安装tools的界面

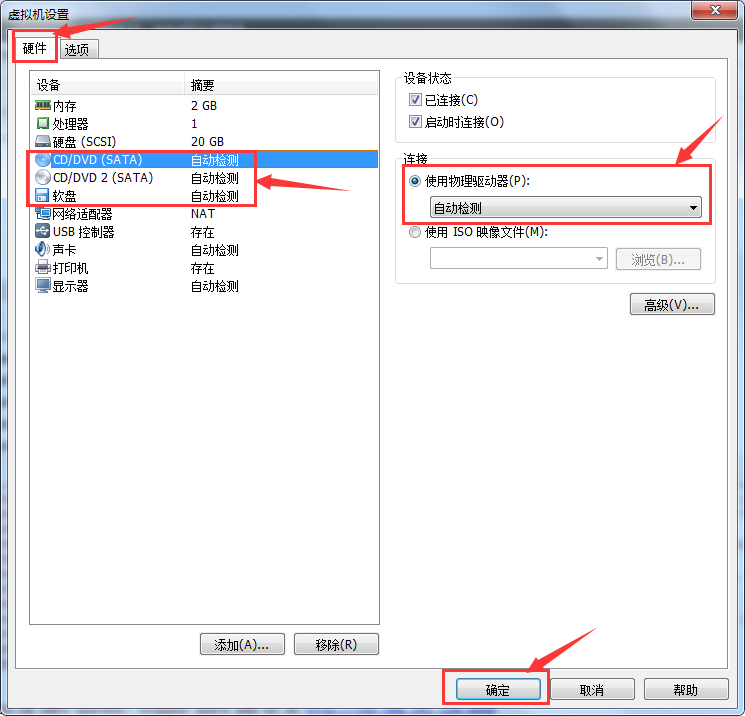


图2-11 安装tools的界面

成功选择【安装VMware Tools】后，弹出如图2-12所示界面，可以看出VmwareTools压缩包下载完成。

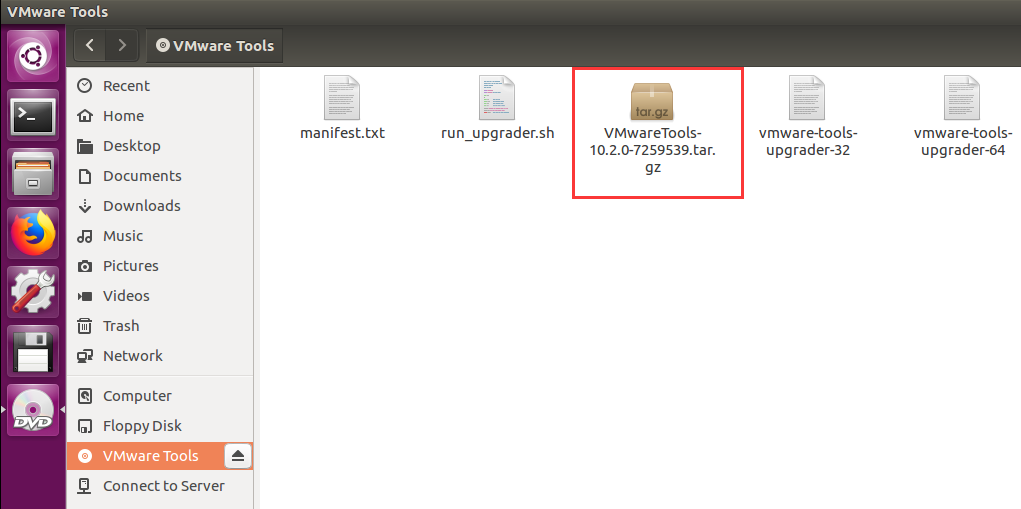


图2-12 安装tools完成的界面

鼠标右键点击 VMwareTools-10.2.0-7259539.tar.gz , 选择【copy to】，出现图2-12界面：

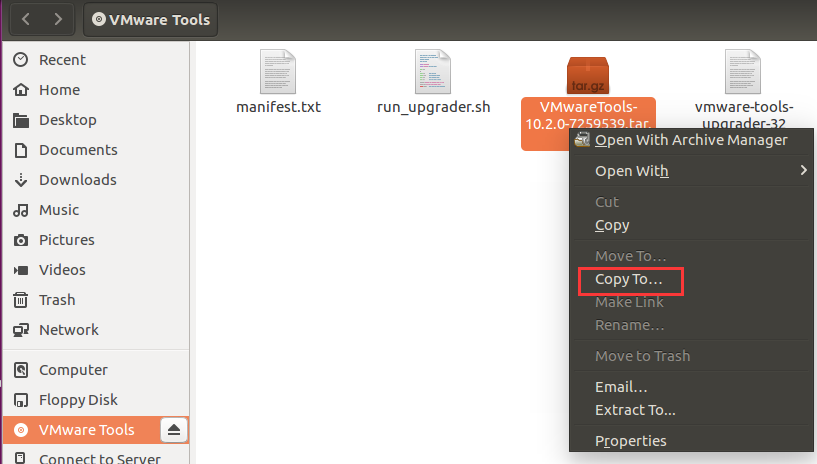


图2-12 复制VmwareTools压缩包

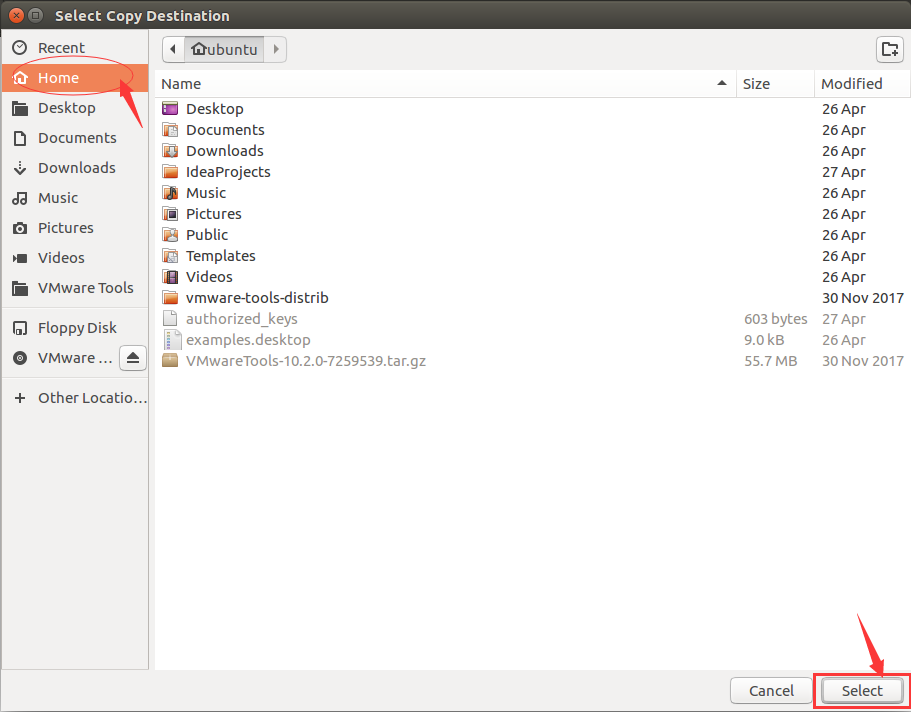


图2-13 复制到Home目录

在上图所示窗口中选择【Home】，最后选择右下角的【Select】。

按照上述步骤复制压缩包完成以后，在Home目录下可以找到 VMwareTools-10.2.0-7259539.tar.gz ，在Home目录下，鼠标右键，选择【Open in Terminal】（打开终端），如下图2-14所示：

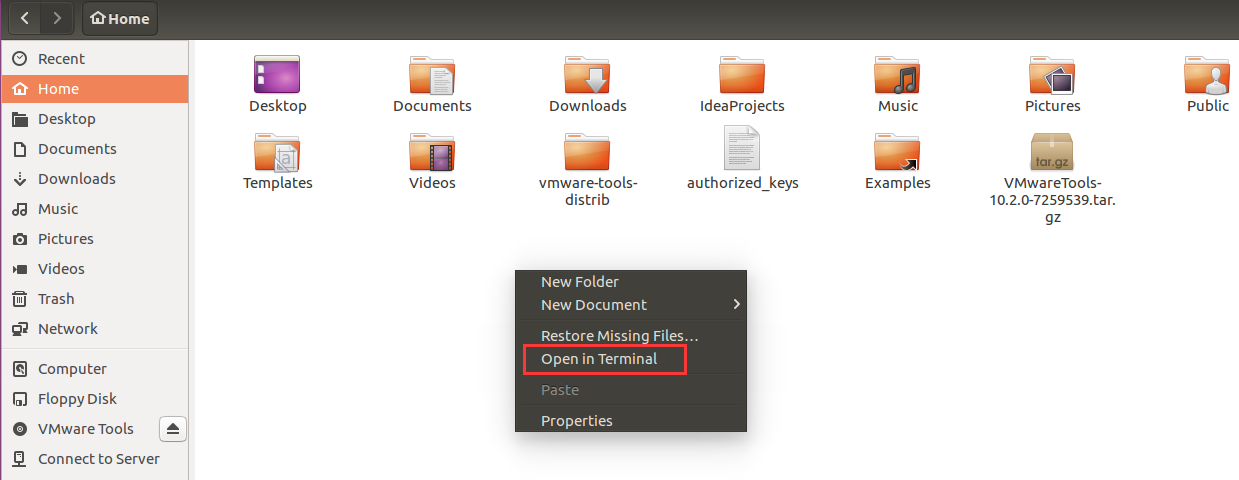


图2-14 打开终端

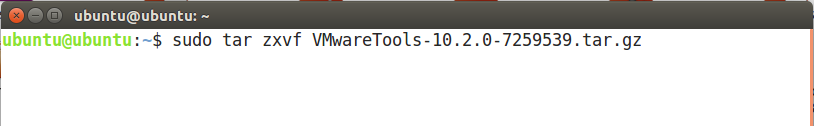


图2-15 解压命令

输入命令：**sudo tar zxvf VMwareTools-10.2.0-7259539.tar.gz** 然后回车执行解压，如图2-15所示。解压完成以后，在Home目录下，出现vmware-tools-distrib文件夹，文件夹内包括内容如图2-16所示：

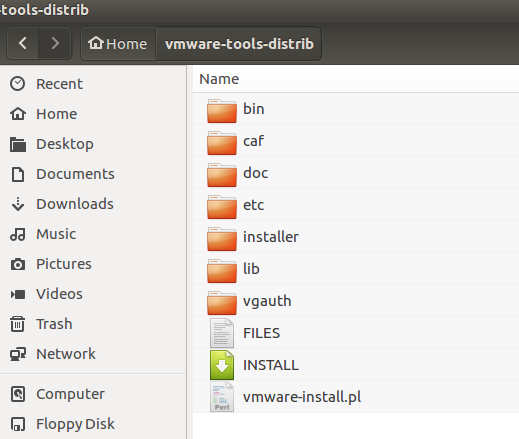


图2-16 vmware-tools-distrib目录

在此文件夹中空白处，鼠标右键选择【Open in Terminal】，输入：**sudo ./vmware-install.pl**

回车键，开始安装。根据提示选择Yes或者No，不确定的功能最好都装上，和hgfs相关的也装上。安装完成

以后重启Ubuntu。

2）设置共享文件夹

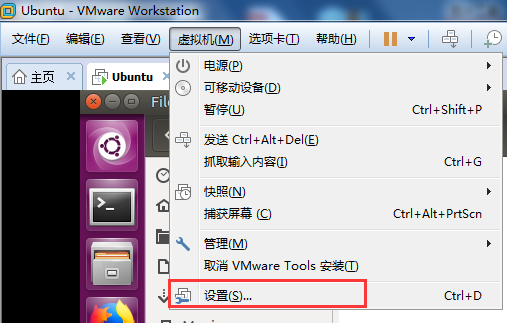


图2-17 设置共享文件夹的界面

选择菜单栏中【虚拟机】，然后选择【设置】，找到图2-18中所示的①【选项】，选择②【共享文件夹】，点击③【添加】，出现【添加共享文件夹向导】弹窗，点击④【下一步】。

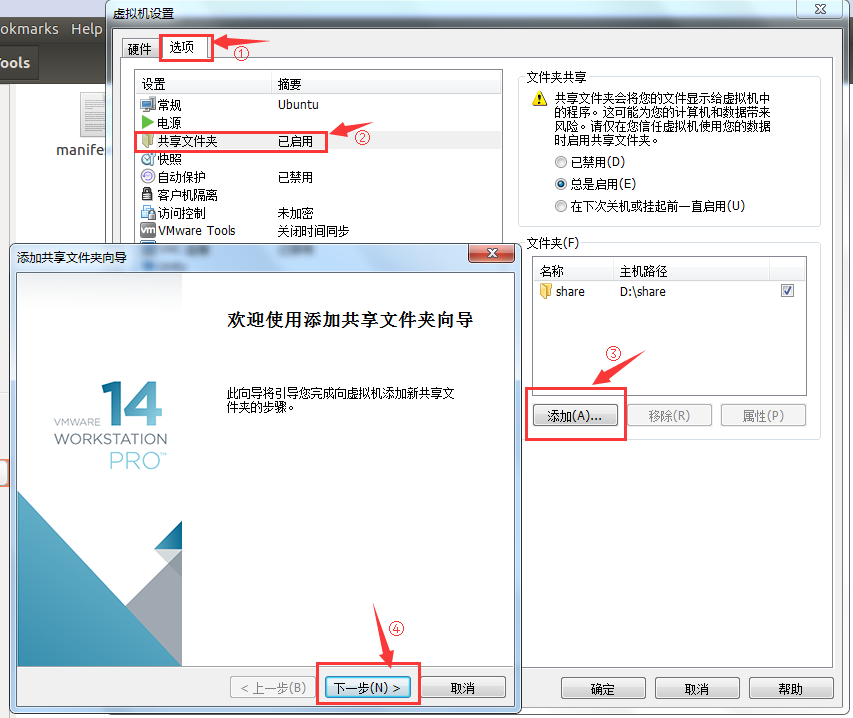


图2-18 添加共享文件夹

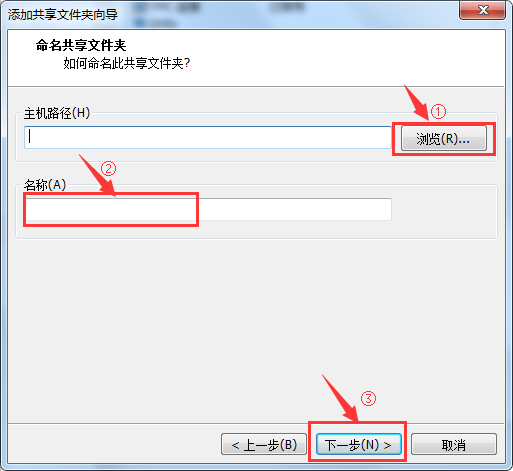


图2-19 设置共享文件夹

点击【浏览】，设置主机中与虚拟机共享文件夹的位置，然后设置【名称】。点击【下一步】。然后，在新出现的界面中选择【启动此共享】，最后选择【完成】。至此，共享文件夹的设置完成。

3）访问共享文件夹

在Ubuntu中，发现在/mnt/hgfs/下，多了一个目录。设置共享文件夹完成。在之后的安装中，可以选择在主机中下载配置环境所需要的压缩包，存放在共享文件夹中，然后在ubuntu系统中的共享文件夹中复制压缩包进行解压安装。

2.2 Spark快速部署

Spark部署主要包括Local（本地）部署、Standalone（伪分布式）部署、YARN部署等。本节将介绍Local（本地）部署。Local模式下部署Spark应用程序比较简单，可以用于检测Spark是否编译安装成功。Spark能通过内置的单机集群调度器在本地运行。在此运行模式下，所有的Spark进程运行在同一个Java虚拟机中。实际上构成一个独立、多线程版本的Spark环境。本地模式适合程序的原型设计、开发、调试及测试。同时，本地运行模式也适用于在单机上进行多核并行计算的实际场景。

**2.2.1 快速搭建Spark**

前期准备：如果读者使用的操作系统为Linux，则直接进入java官网下载JDK，Spark官网下载spark压缩包。具体版本号，下载地址如下表所示：

表2-1 软件清单

|  |  |  |
| --- | --- | --- |
| **软件名** | **版本号** | **下载地址** |
| **Java** | **JDK8及以上**  **（linux版本）** | http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html |
| **Spark** | **2.0及以上** | http://spark.apache.org/downloads.html |

如果使用的是VMware搭建的ubuntu系统，则首先在主机上下载jdk压缩包以及Spark压缩包到共享文件夹中，然后将共享文件夹（/mnt/hdfs/共享文件夹名）中的压缩包复制到/usr/software目录下。

搭建local模式，只需配置JDK和Spark，接下来，将详细介绍具体操作。

**1）JDK**

第一步，安装JDK。在JDK压缩包存放的目录（/usr/software）下鼠标右键点击【Open in Terminal】，打开终端，将jdk压缩包解压到/usr/local，命令：**sudo tar zxvf jdk-8u171-linux-x64.tar.gz -C /usr/local**

第二步，配置环境变量。终端中输入：**sudo gedit ~/.bashrc**，进行环境变量设置。如图2-20所示，

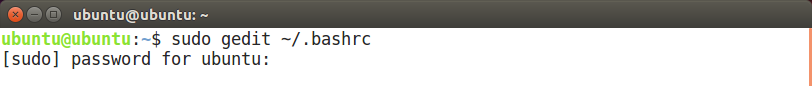


图2-20 环境变量配置

回车运行，在打开的.bashrc文档最后加入如下字段：

**#JAVA**

**export JAVA\_HOME=/usr/local/jdk1.8.0\_171**

**export JRE\_HOME=${JAVA\_HOME}/jre**

**export CLASSPATH=.${JAVA\_HOME}/lib:${JRE\_HOME}/lib**

**export PATH=$PATH:$JAVA\_HOME/bin**

如图2-21中①所示：

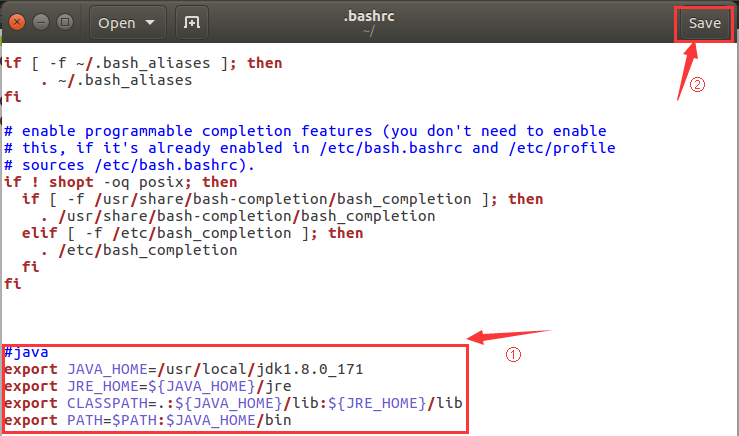
****

图2-21设置jdk

添加上述字段以后，点击②【Save】保存修改。然后在终端中输入：**source ~/.bashrc** 使配置的环境变量生效。

第三步，检测配置。终端中输入：**java -version**，若出现以下结果，则配置成功。

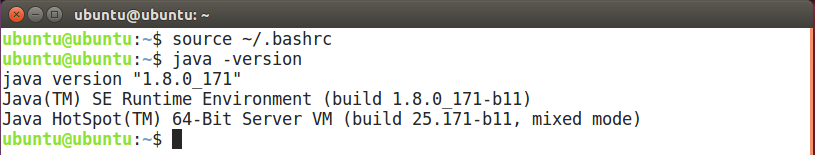


图2-22 检测jdk配置

**2）Spark**

第一步，安装Spark。在Spark压缩包存放的目录（/usr/software）下鼠标右键点击【Open in Terminal】，打开终端，将spark压缩包解压到/usr/local：**sudo tar zxvf spark-2.3.0-bin-hadoop2.7.tgz -C /usr/local**

第二步，配置环境变量。终端中输入：**sudo gedit ~/.bashrc**，回车运行。在.bashrc文档最后加入如下字段：

**#SPARK**

**export SPARK\_HOME=/usr/local/spark-2.3.0-bin-hadoop2.7**

**export PATH=$PATH:$SPARK\_HOME/bin**

添加上述字段以后，点击【Save】保存修改。然后在终端中输入：**source ~/.bashrc** 使配置生效。

第三步，检测配置。终端中输入：**spark-shell** 出现如图2-23所示信息则成功

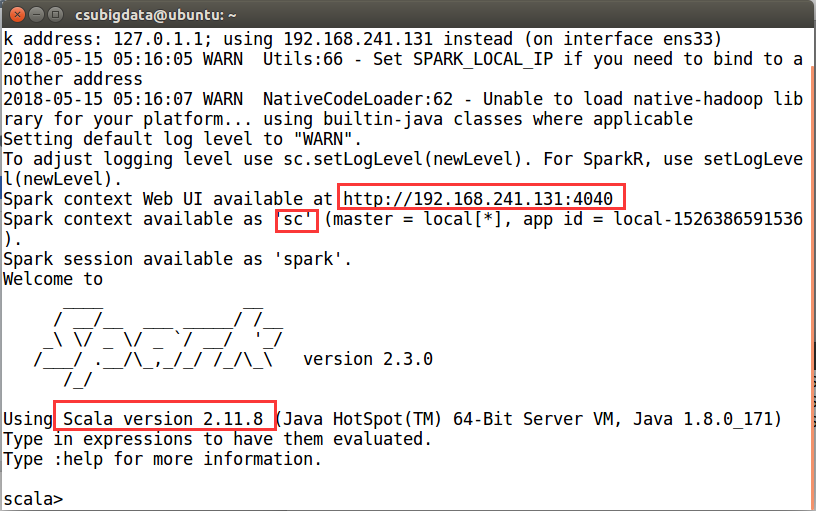


图2-23 spark搭建完成

**2.2.2第一个Spark实例**

根据2.2.1完成搭建Spark的快速搭建以后，本节中，将介绍第一个Spark实例——wordcount，统计文本中各单词出现次数。在ubuntu中，打开终端，输入**spark-shell**，启动spark-shell，成功启动以后，如图2-23所示。

Spark shell中，已经初始化了一个SparkContext对象，可以通过sc这个Scala值来调用这个对象。在spark

shell中，输入的代码会被立即计算，shell能在输入代码时给出实时反馈。命令执行结果的值与类型在代码执行完后会显示出来。接下来，在shell中，运行第一个spark实例，具体如下：

1)创建一个RDD，从本地目录下传入文件，记为rdd1。在shell中输入：

**val rdd1 = sc.textFile(“file:///usr/local/sparkwc”)**

注意：该文件当前在上述路径中不存在。按下回车，结果如图2-23所示

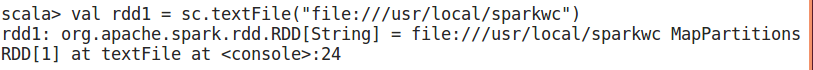


图2-24 本地传入文件操作

根据上图可知，运行结果未报错。因为此操作属于transformation。只记录rdd之间逻辑关系。

2)将rdd1中的文本按 \t 切割，进行分词，生成rdd2。在shell中输入：

**val rdd2 = rdd1.flatMap(\_.split(“\t”))**

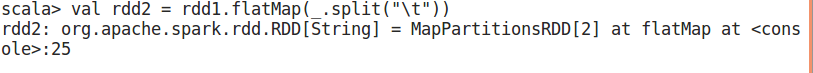


图2-25 rdd1转换操作

在usr/local目录下鼠标右键打开终端，创建一个名为sparkwc的文件，终端中输入命令：**sudo touch sparkwc**

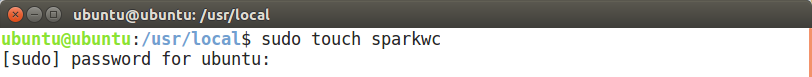


图2-26 新建sparkwc文件

然后，在终端中输入**sudo** **gedit sparkwc**编辑sparkwc内容，如图2-27所示，单词之间用Tab键分开，点击【Save】保存：

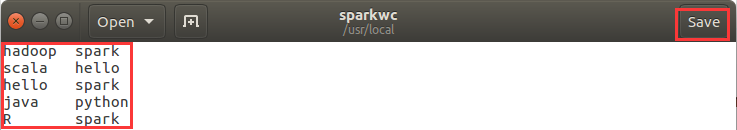


图2-27 编辑sparkwc内容

3)将rdd2中的词转换为形如（单词，1）的元组，生成rdd3。在shell中输入命令：

**val rdd3 = rdd2.map((\_,1))**

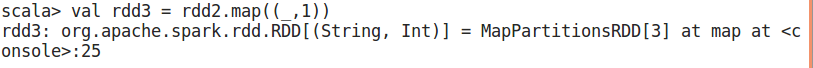


图2-28 rdd2转换操作

4)将rdd3中元组按照key进行累加，生成rdd4：

**val rdd4 = rdd3.reduceByKey(\_+\_)**

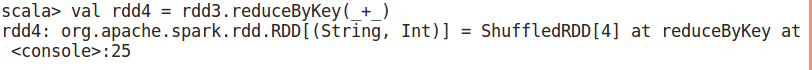


图2-29 rdd3转换操作

5)输出rdd4中结果：

**rdd4.collect**

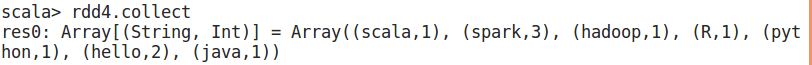


图2-30 rdd4执行操作

不退出spark-shell，下一小节，将介绍SparkWEB的使用。查看第一个spark实例的运行信息。

**2.2.3 SparkWEB使用**

每个Dirver的SparkContext都会启动一个WEB界面，默认端口为4040，在浏览器中输入localhost:4040，可以查看程序的诸多有用信息，包括：调度器Stage、Task列表；RDD大小和内存使用情况统计；环境配置信息；正在运行的执行器信息。Spark-shell执行完2.2.2中代码以后，不退出spark-shell。在ubuntu自带的FireFox浏览器中输入网址：localhost:4040，回车进入SparkWEB监控页面。如下图所示。

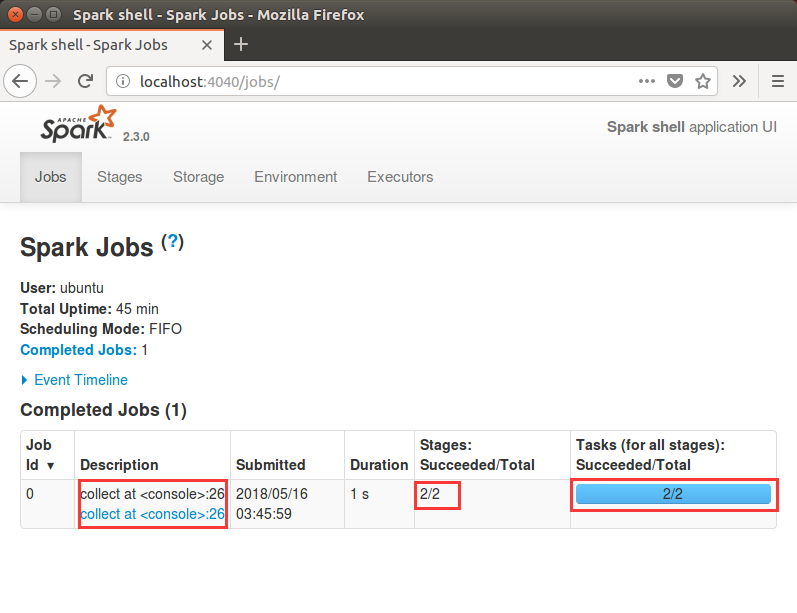


图2-31 SparkWEB

根据上图可以看出，Job完成，提交时间在【Submitted】栏中，运行时间在【Duration】栏中显示，由【Stages】栏和【Tasks】栏中内容可知，任务被拆成了2个Stage和两个Task。点击上图菜单栏中的【Stages】，进入任务Stages详情页面，如图2-32所示，可以看出两个stages划分以及各自执行时间等信息。

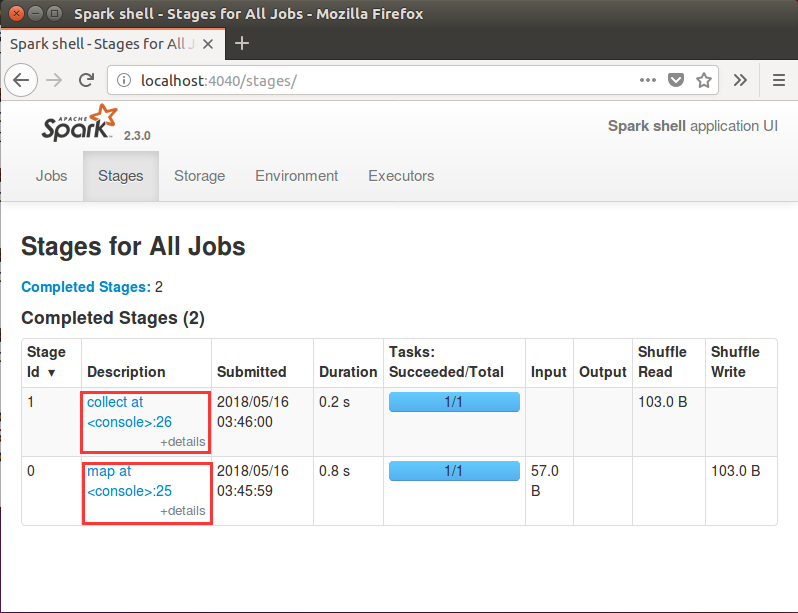


图2-32 查看stage

点击图2-31中【Description】栏中的链接，可以查看spark执行的任务细节。如图2-33所示。

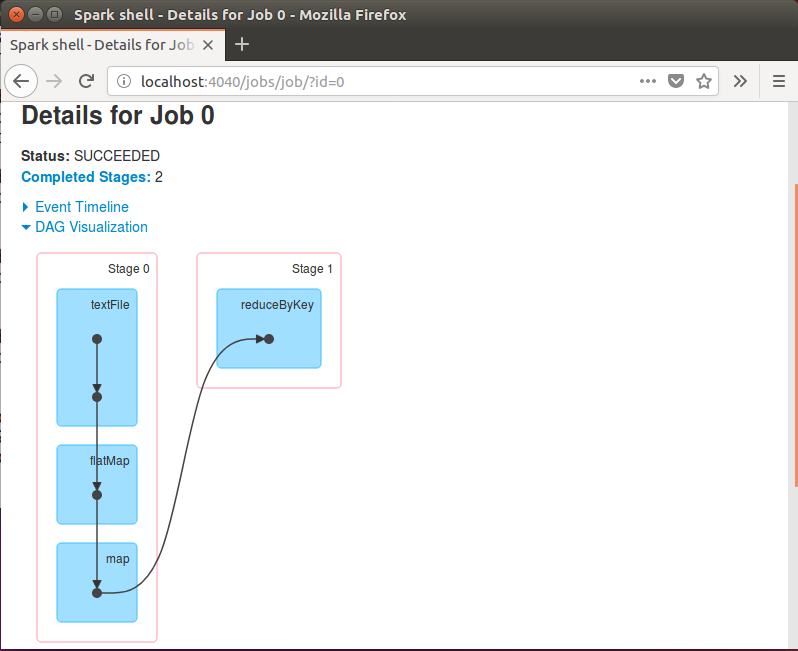


图2-33 details

**2.3 IDEA**

为了在Spark上开发应用程序，需要安装有效的集成开发环境。IDEA是一个通用的集成开发环境，Spark通常采用Scala语言进行开发，而IDEA则是最佳的Scala语言开发环境。本节我们将介绍maven与IDEA的配置安装，以及在IDEA上，使用maven创建并运行第一个Spark程序以及直接创建scala项目的方式创建一个Spark程序，最后，将介绍使用IDEA打包spark程序，通过Spark submit提交运行。

**2.3.1 IDEA安装**

前期准备：如果读者使用的操作系统为Linux，则直接进入maven官网下载JDK，IDEA官网下载IDEA压缩包。具体版本号，下载地址如下表所示：

|  |  |  |
| --- | --- | --- |
| **软件名** | **版本号** | **下载地址** |
| **Maven** | **3.5及以上** | https://maven.apache.org/download.cgi |
| **IDEA** | **Linux版本** | https://www.jetbrains.com/idea/download/#section=linux |

如果使用的是VMware搭建的ubuntu系统，则首先在主机上下载maven压缩包以及IDEA压缩包到共享文件夹中，然后将共享文件夹（/mnt/hdfs/共享文件夹名）中的压缩包复制到/usr/software目录下。

1. **Maven**

第一步，安装Maven。在maven压缩包的存放目录(或/usr/software)下鼠标右键点击【Open in Terminal】，打开终端，将scala压缩包解压到/usr/local，输入命令：**sudo tar zxvf apache-maven-3.5.3-bin.tar.gz -C /usr/local**

第二步，配置环境变量。终端中输入：**sudo gedit ~/.bashrc**，回车运行。在.bashrc文档最后加入如下字段：

**#MAVEN**

**export MAVEN\_HOME=/usr/local/apache-maven-3.5.3**

**export PATH=$PATH:$MAVEN\_HOME/bin**

添加上述字段以后，点击【Save】保存修改。然后在终端中输入：**source ~/.bashrc**使配置生效。

第三步，检测配置。终端中输入**mvn -version** 出现图2-34所示版本信息则成功。

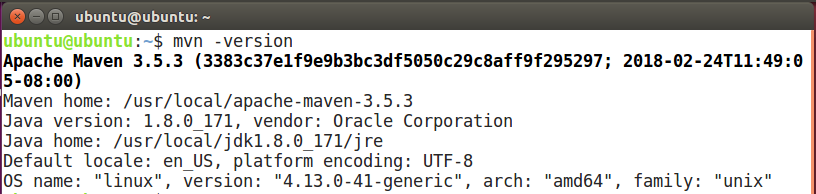


图2-34 maven配置生效

1. **IDEA**

第一步，安装IDEA。在IDEA压缩包的存放目录(/usr/software)下鼠标右键点击【Open in Terminal】，打开终端，将scala压缩包解压到/usr/local，输入命令：**sudo tar zxvf ideaIC-2018.1.2.tar.gz -C /usr/local**

第二步，启动IDEA。在idea的安装目录的bin目录下打开新的终端，输入：**./idea.sh**，出现IDEA启动窗口。

等待数秒以后，出现如图2-35。



图2-35 idea启动

默认选择，然后点击【OK】。然后出现窗口如图2-36。

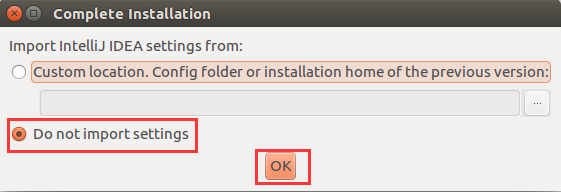


图2-35 idea启动设置①

拉动下拉框到地步，【Accept】选项可选择，则点击【Accept】，出现窗口如图2-37。

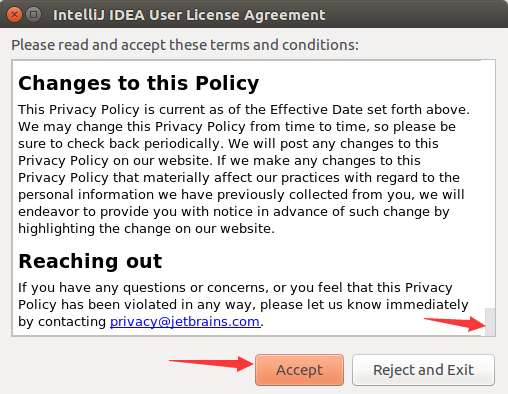


图2-36 idea启动设置②

此界面设置UI风格，本文中选择【IntelliJ】，可以选择窗口右下角【Next】进行其他配置，本书中默认配置，直接选择窗口左下角【Skip Remaining and Set Default】。出现如图2-38。接下来，将进行IDEA的环境配置。

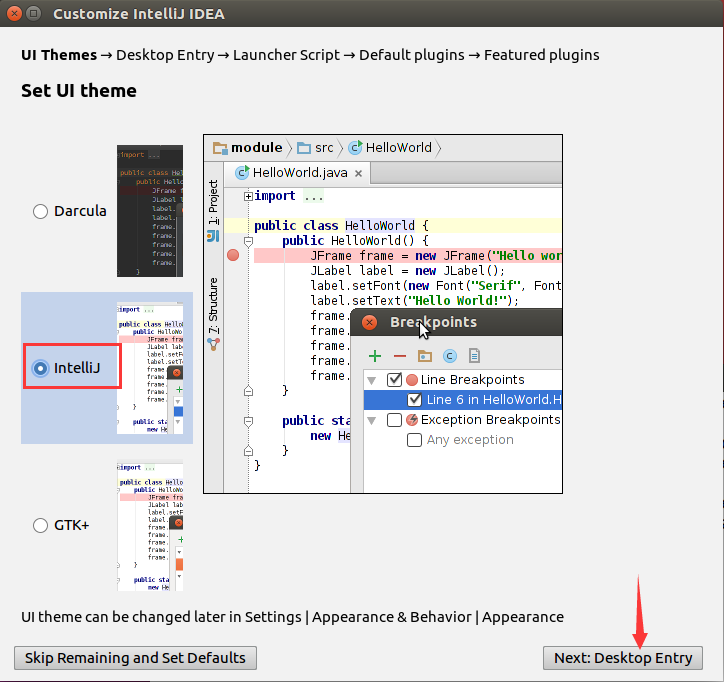


图2-37 idea启动设置③

****

图2-38 IDEA配置

第三步，配置IDEA

1）安装scala插件。如图2-38所示，首先选择【configure】，然后点击【Plugins】

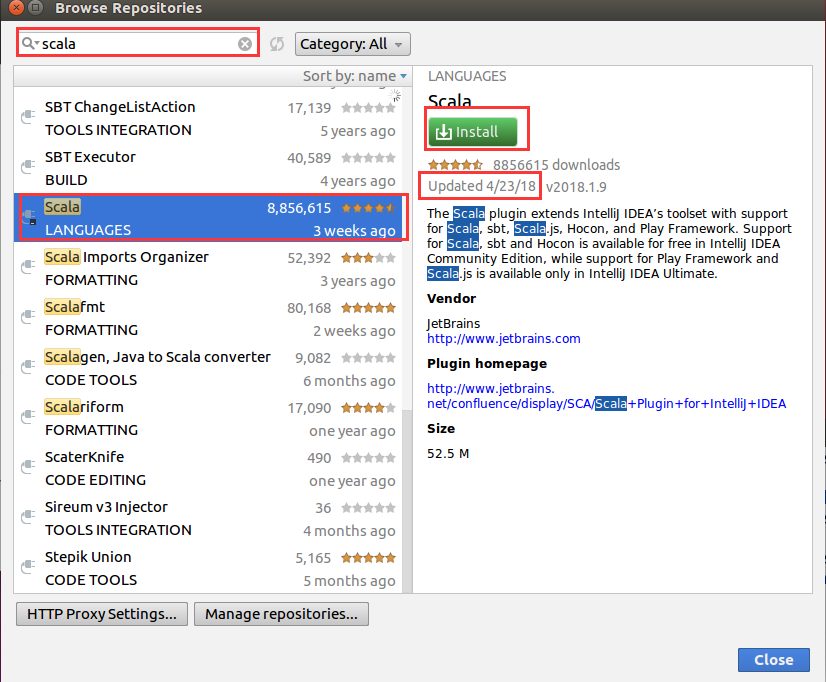


图2-39 安装scala插件

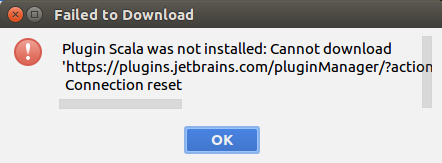


图2-40 安装scala插件失败

出现如图2-39所示。在搜索栏中输入scala，进行下载。IDEA中Scala下载长时间无响应或者搜索失败如图2-40所示。解决方案：我们可以在<http://plugins.jetbrains.com/plugin/?idea&id=1347>中找到对应版本（依照图2-37中检索scala以后，Install按钮下方的Updated \*\*-\*\*-\*时间，在上述网址中下载对应的scala版本，或者根据IDEA解压以后的版本号确定，例如idea182.\*\*\*，选取182对应的版本scala下载即可)

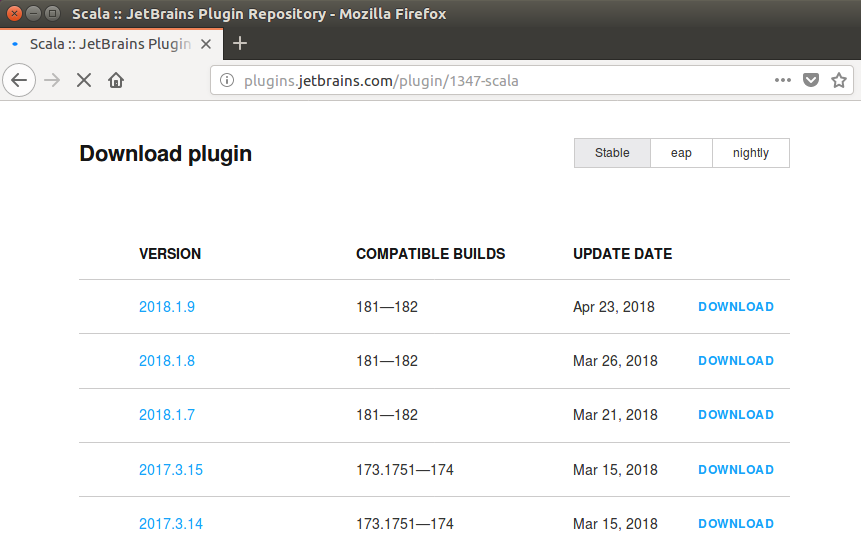
****

图2-41 scala插件选择

下载完插件之后，把.zip格式的scala插件压缩包复制到Intellij的安装目录的plugins目录下；再安装scala插件（注：直接安装zip文件）即可。如下图所示步骤：

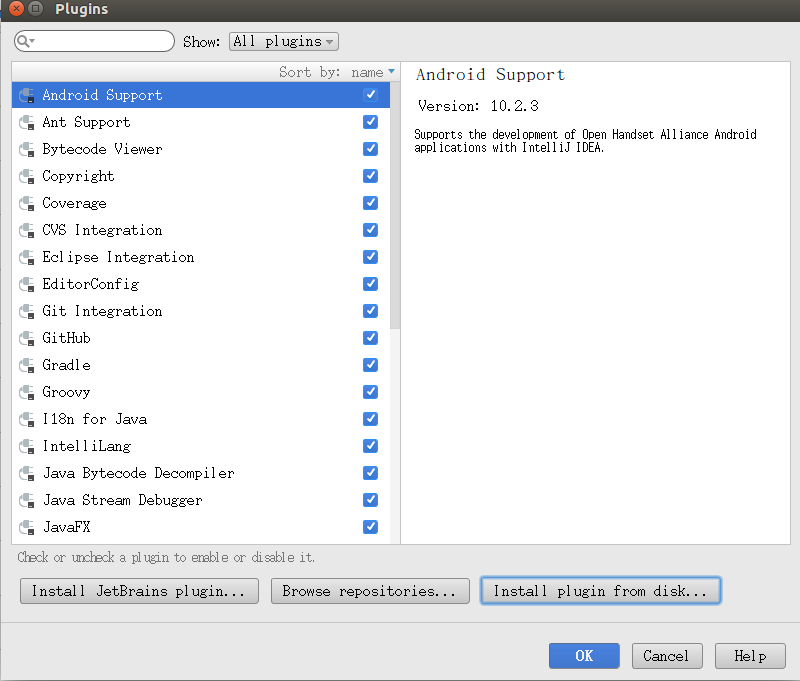
****

图2-42 scala插件从磁盘安装

点击【Install plugin from disk】，找到scala插件存放路径，点击【OK】，出现如下弹窗，点击【Restart】，重启IDEA

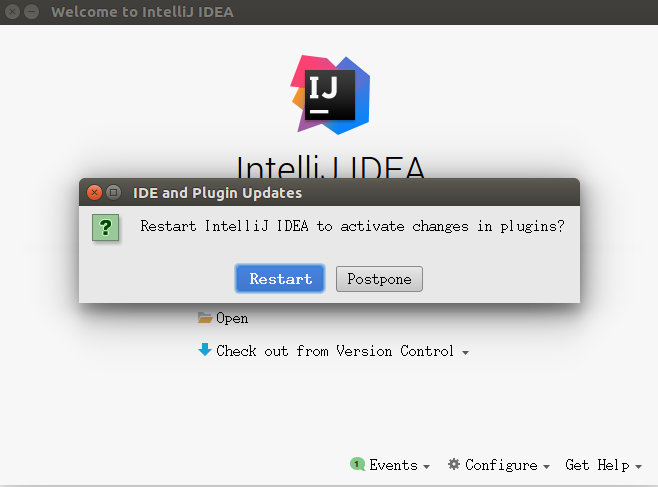


图2-43 重启IDEA

1. **设置全局JDK**

为了不用每次都去配置JDK，先进行一次全局配置。首先在IDEA欢迎界面点击【Configure】，然后在【Project Defaults】的下拉菜单中选择【Project Structure】，如下图所示：

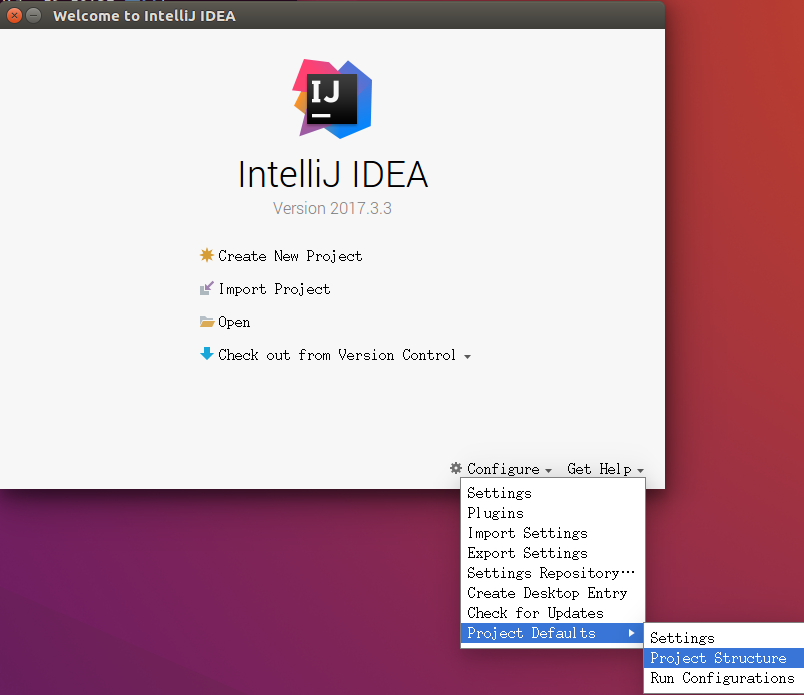


图2-44 配置JDK

出现如下窗口，点击【project】，如图2-45所示，SDK没有默认添加，我们可以点击【New...】，找到jdk的安装路径。导入jdk。最后点击【OK】

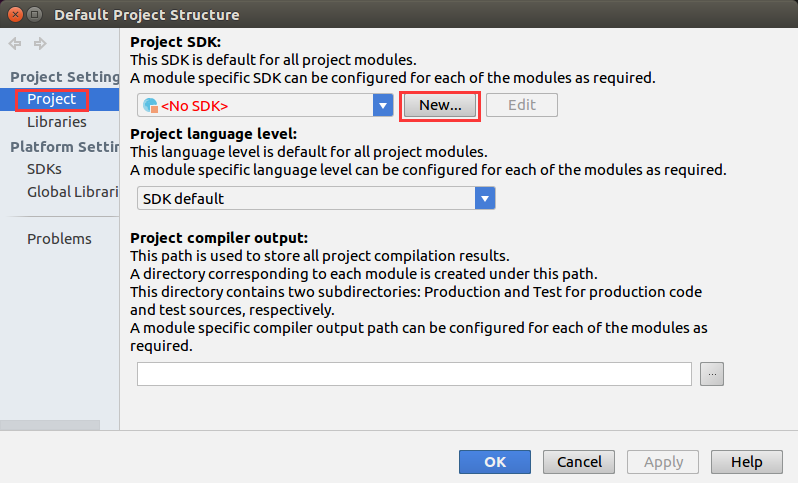


图2-45 JDK配置页面

**2.3.2 IDEA的第一个实例（Scala+Maven）**

根据2.3.1中完成IDEA的配置以后，开始使用IDEA进行开发。接下来，将介绍使用IDEA集成开发环境，通

过maven创建并运行第一个列表求和并输出大于10的数的程序。

在欢迎界面点击【Create New Project】，

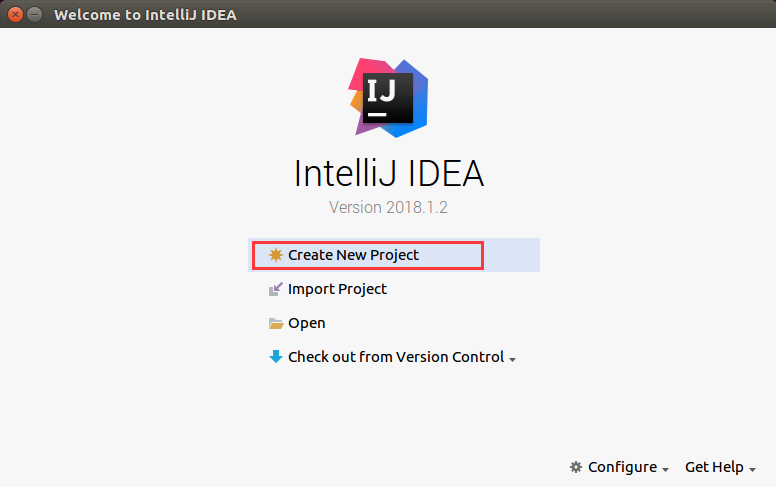


图2-46 创建新项目

在打开的页面左侧边栏中，选择【Maven】，然后在右侧的Project SDK一项中，查看是否是正确的JDK配置项正常来说这一栏会自动填充的，之前已经配置过了全局的Project JDK了，如果这里没有正常显示JDK的话，可以点击右侧的【New…】按钮，然后指定JDK安装路径的根目录即可），

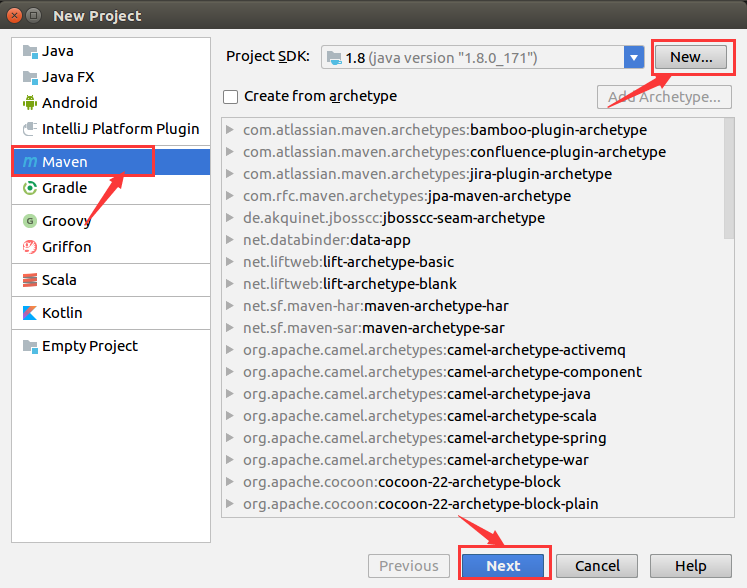


图2-47 创建maven项目

然后点击【Next】，来到Maven项目最重要三个参数的设置页面，这三个参数分别为：

GroupId，ArtifactId和Version。

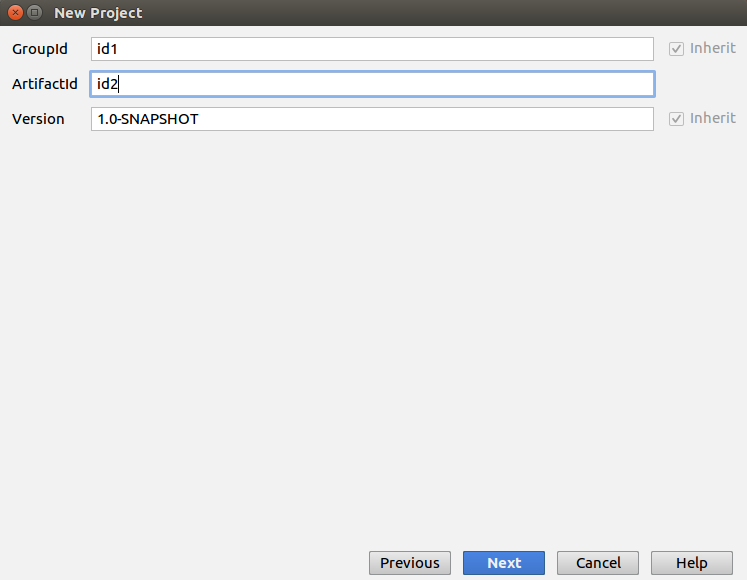


图2-48 id设置页面

点击【Next】，在【Project name】栏中，输入项目名称。在当前界面，也可以修改项目存储的路径。最后，点击【Finish】。

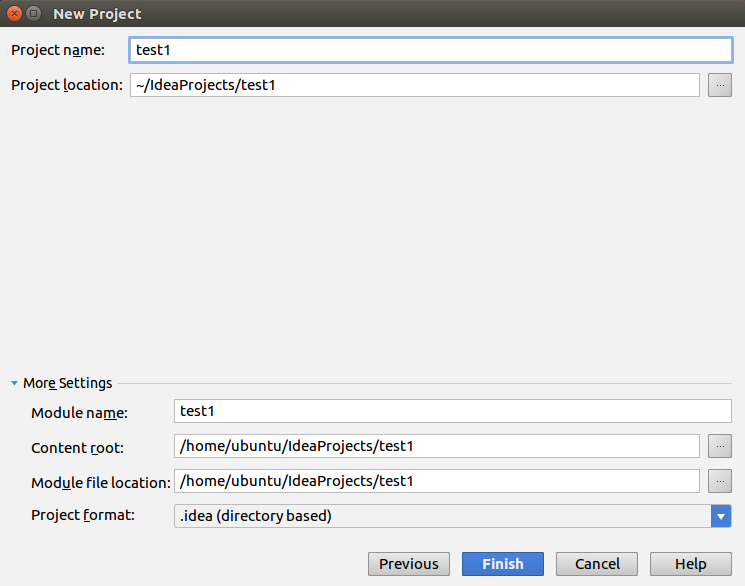


图2-49 项目名设置

为了首次体验IDEA更清爽一些，可以将一些暂时无关的文件和文件夹删除，主要有 main\java, main\resources 和 test 这三个；左键点击【file】，选择【Project Structure】，页面左侧选择【Global Libraries】，然后在中间一栏中有一个绿色的加号标志 +，点击后在下拉菜单中选择 【Scala SDK】。

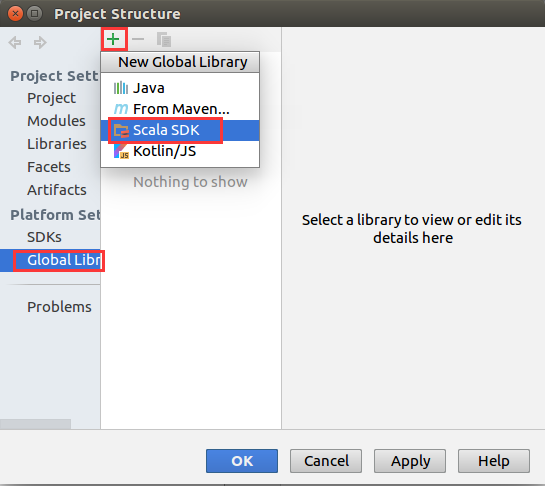
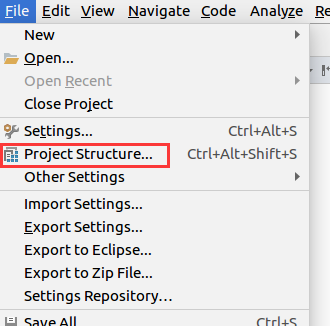


图2-50 sdk设置

然后在打开的对话框中选择IDEA下载的Scala插件，选择版本。

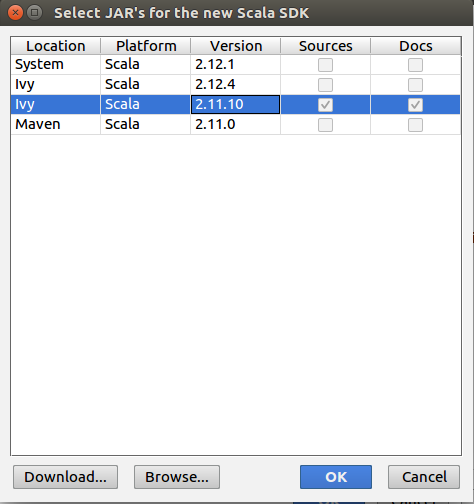


图2-51 sdk选择

点击【OK】确定，这时候会在中间一栏位置处出现Scala的SDK，在其上鼠标右键点击后选择【Copy to Project Libraries…】，将Scala SDK添加到项目的默认Library中去。具体操作如下图：

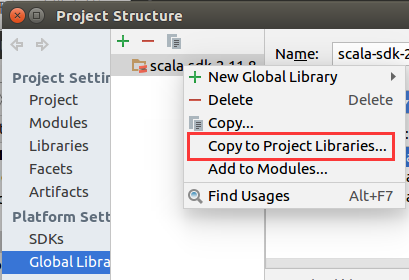


图2-52 sdk设置

为了避免出现scala版本冲突的问题，（在写依赖关系中，指明的scala版本，在spark上运行时，出现版本的问题），故在页面右侧有选择版本，选择2.11，加载出来的Scala会自动匹配为该版本。

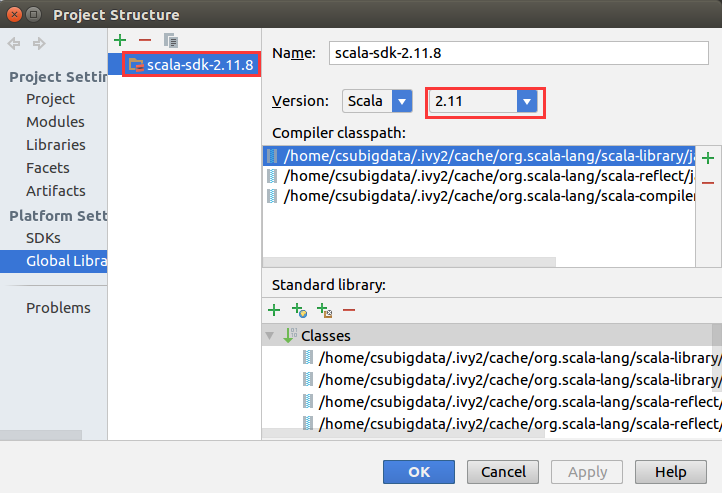


图2-53 配置全局SDK

在main上鼠标右键点击【Mark Directory as】，然后点击【Sources Root】。将main文件夹标记为一个源文件的根目录，然后在其内的所有代码中的 package ，其路径就从这个根目录下开始算起。如下图所示：

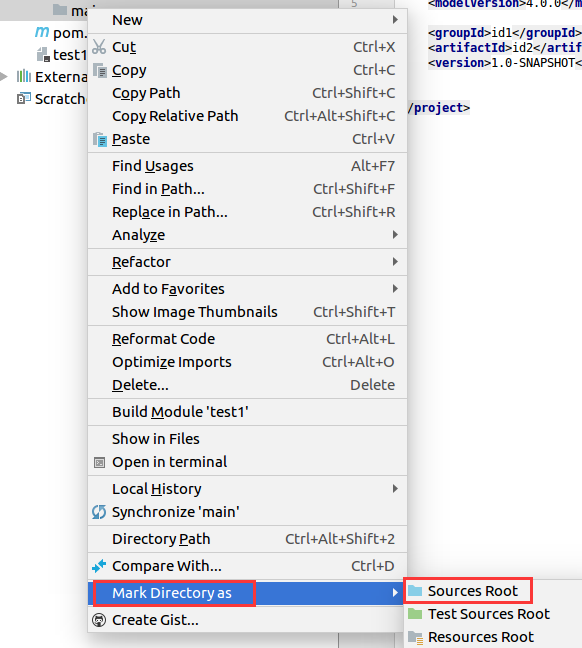


图2-54 设置根目录

将Scala的框架添加到这个项目中，方法是在左侧栏中的项目名称上鼠标右键点击【Add Framework Support…】，

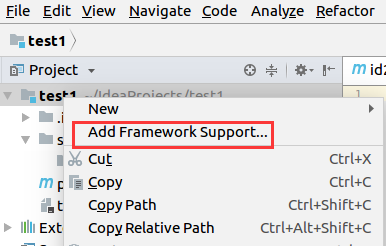


图2-55 添加scala框架①

然后在打开的对话框左侧边栏中，勾选Scala前面的复选框，然后点击确定即可，如下图（前提是上文中所述步骤都已正确走通，否则你很有可能看不到Scala这个选项的，如果看不到Scala选项，依照之前步骤，重新导入scala SDK后等待2分钟左右，重启IDEA即可解决。）

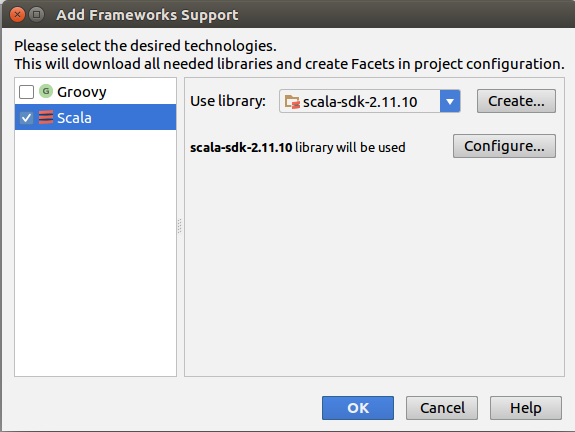


图2-56 添加scala框架②

在已经标记好为源文件根目录的main文件夹上，鼠标右键选择 【New】，然后选择【Scala Class】，如下图所示

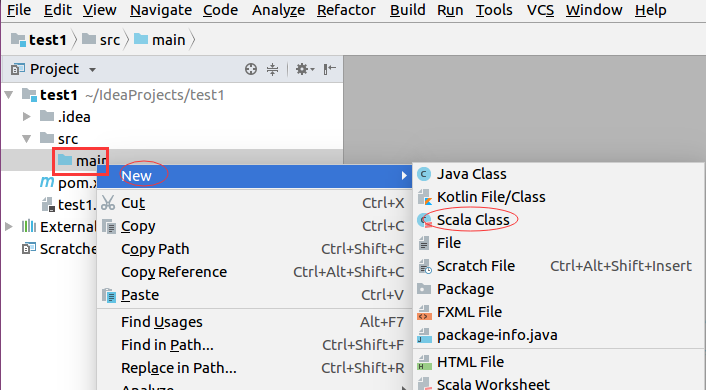


图2-57 新建scala项目①

随后在Name栏中，设置程序的名称，并且选择点击下拉框，设置为Object，如图2-49所示。然后点击【OK】完成创建。一切正常的话，将会打开这个Object 代码界面。如下图2-58所示：

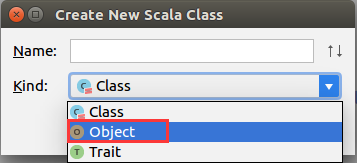
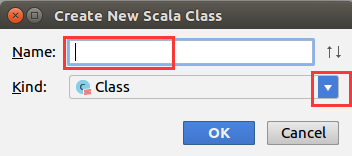


图2-58 新建scala项目②

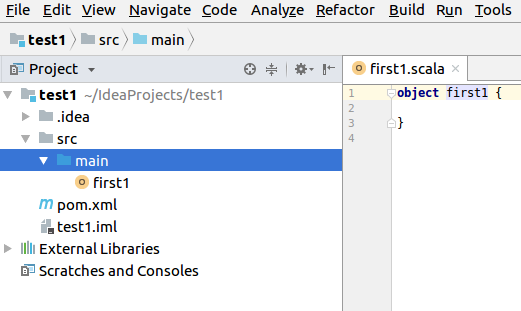


图2-59 新建scala程序

想要运行在spark 上，还需要导入相关依赖。打开pom.xml文件添加如下依赖。

**<properties>**

**<spark.version>2.3.0</spark.version>**

**<scala.version>2.11</scala.version>**

**</properties>**

**<dependencies>**

**<dependency>**

**<groupId>org.apache.spark</groupId>**

**<artifactId>spark-core\_${scala.version}</artifactId>**

**<version>${spark.version}</version>**

**</dependency>**

**</dependencies>**

如下图所示：

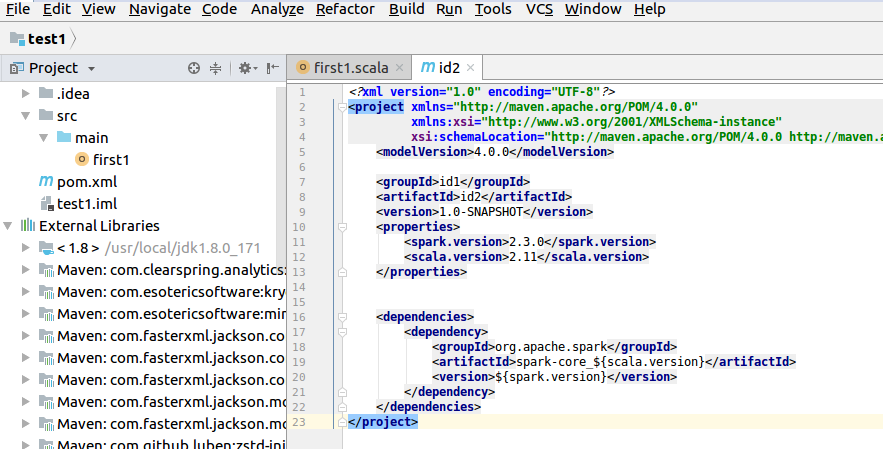


图2-60 修改pow.xml

注意版本号：

**<spark.version>2.3.0</spark.version>**

**<scala.version>2.11</scala.version>**

其中spark version要与系统变量中安装的spark版本一致，Scala version要与之前通过idea下载的版本一致，但 scala-2.a.b（a，b都为版本号，仅保留2.a），例如，scala-2.11.1此处写2.11

写入依赖后，切记导入import包，否则依赖关系中的代码出现错误

点击【Enable Auto-Import】



图2-61 自动导入

耐心等待maven导入依赖包，左侧如下图，则配置依赖成功。

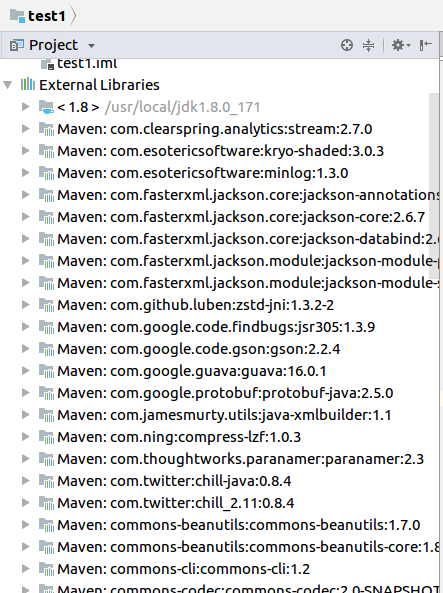


图2-62 导入成功

在main目录下的first1.scala中，然后输入如下代码：

Import org.apache.spark.{SparkConf,SparkContext}

object first {

def main(args: Array[String]) {

val conf = new SparkConf().setAppName("mySpark").setMaster("local")

val sc =new SparkContext(conf)

val rdd =sc.parallelize(1 to 6).map(\_ \* 3)

val mappedRDD=rdd.filter(\_ > 10).collect()

println("sum = " + rdd.reduce(\_ + \_))

for(arg <- mappedRDD)

print(arg + " ")

println()

sc.stop()}}

在空白处，右键选择【Run】出现如下结果，则表示案例成功运行：

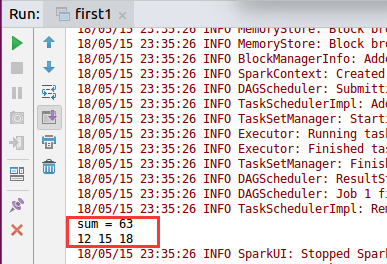
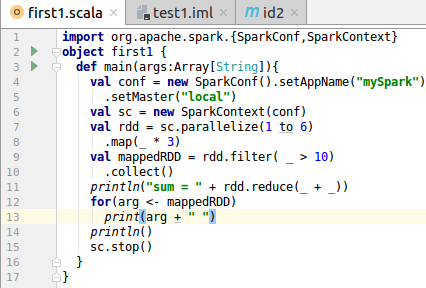


图2-63 第一个程序运行结果

**2.3.3 IDEA的第二个实例（Scala）**

本节中，将介绍使用IDEA直接创建scala项目，并运行wordcount，进一步演示IDEA进行开发的流程。首先，创建一个项目，名为wordcount。

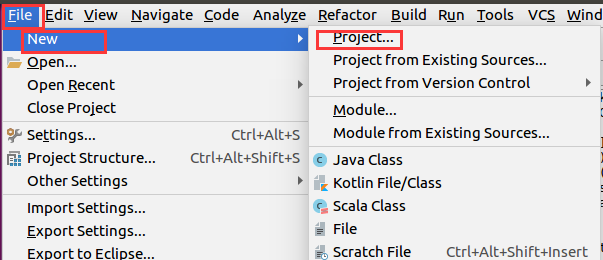


图2-64 新建项目①

如上图2-64所示，点击菜单栏【File】，然后点击【New】，再点击【Project】，然后弹出如图2-65界面，选择【Scala】，然后点击【IDEA】，最后点击【Next】。接下来进入如图2-66窗口，设置项目名，JDK和SDK（如果IDEA环境完成配置，JDK和SDK会自动填充），填写完信息，点击【Finish】。

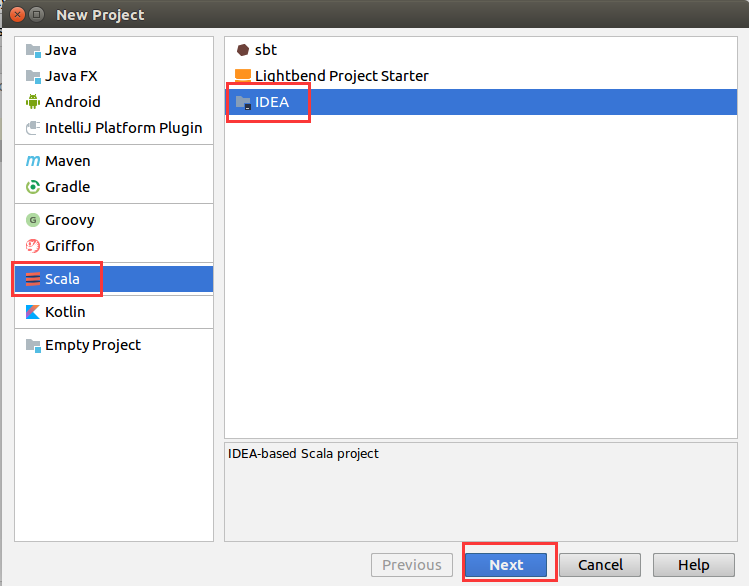


图2-65 新建项目②

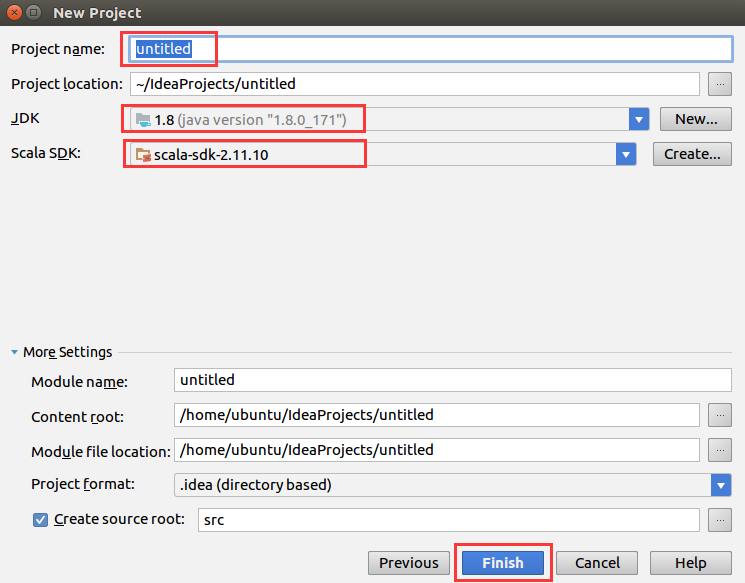


图2-66 新建项目③

创建项目完成以后，在如图2-67所示的文件src上右键点击，新建一个【Scala Class】，并将类名定义为WordCount，类别修改为Object（此步骤与2.3.2相同），



图2-67 新建项目④

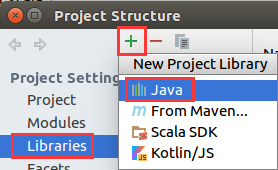


图2-68 导入jar包①

创建项目完成以后，需要引入必要的jar包。点击菜单栏【File】，然后点击【Project Structure】，进入环境配置页，选择【Global Libraries】，鼠标右键选择出现的SDK，鼠标右键点击【Copy to Project Libraries…】，将SDK加入到此项目的环境中。然后如图2-68，选择【Libraries】，然后点击中间的绿色【+】，点击【Java】，然后出现如图2-69的路径选择。选择Spark安装目录下的jars文件夹，最近点击【OK】，出现2-70窗口，点击【OK】。至此Jar包导入完成。

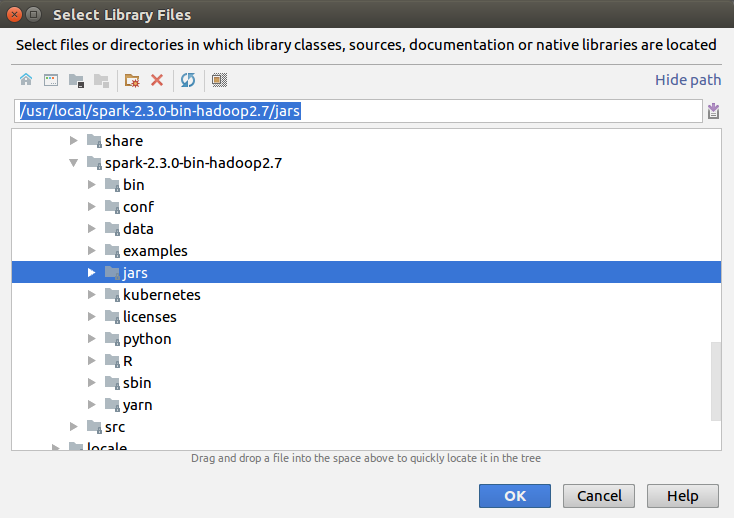


图2-69 导入jar包②

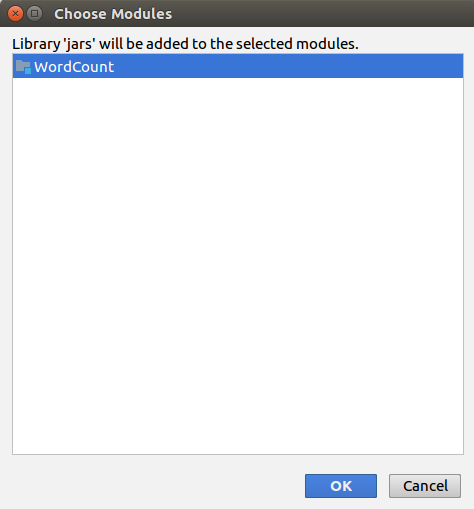


图2-70 导入jar包③

开始编写代码，在新创建的WordCount.scala中输入以下代码：

import org.apache.spark.{SparkConf,SparkContext}

object WordCount {

def main(args: Array[String]) {

val conf = new SparkConf().setAppName("mySpark").setMaster("local")

val sc = new SparkContext(conf)

val rdd = sc.textFile(args(0))

val wordcount = rdd.filtMap(\_.split(“\t”)).map((\_,1))

.reduceByKey(\_ + \_)

for(arg <- wordcount.collect())

print(arg + " ")

println()

wordcount.saveAsText(args(1))

sc.stop()

}

}

如下图2-66所示，

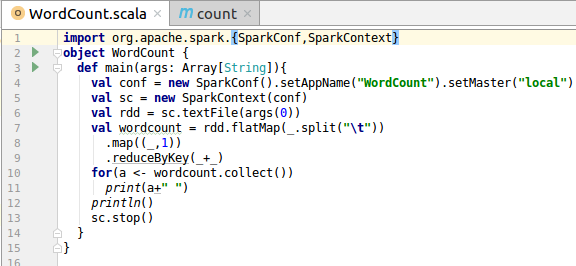


图2-71 WordCount.scala

在代码中，需要参数args[0]，且表示文件路径。需要在运行项目时，指定参数内容。接下来，将展示wordcount项目运行流程：

**第一步，代码编译**

代码在运行前，首先要进行编译，单击菜单【Bulid】，然后点击【Make Project】或者使用【Ctrl+F9】组合键对代码进行编译。结果在Event Log中显示

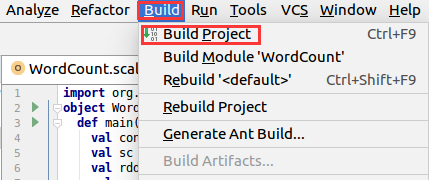


图2-72 IDEA编译项目

**第二步，运行调试配置**

编译通过以后，单击菜单【Run】，然后点击【Edit Configuration】可打开配置窗口。如下图所示，在本节编写的wordcount程序中需要一个参数，即单词计数的文件路径

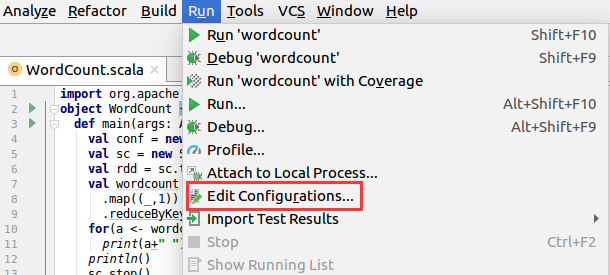


图2-73运行配置

在出现的Run/Debug Configuration窗口中，添加一个【Aplication】，具体步骤：首先，如图2-74，点击【+】创建一个【Application】，然后在如图2-75的窗口中，设置【Main Class】，点击右侧的【...】，选择项目中的WordCount.scala路径，然后在【Program arguments】栏中输入参数。最后点击【OK】。

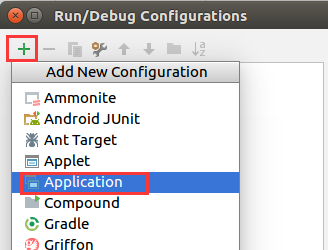


图2-74 创建一个Application

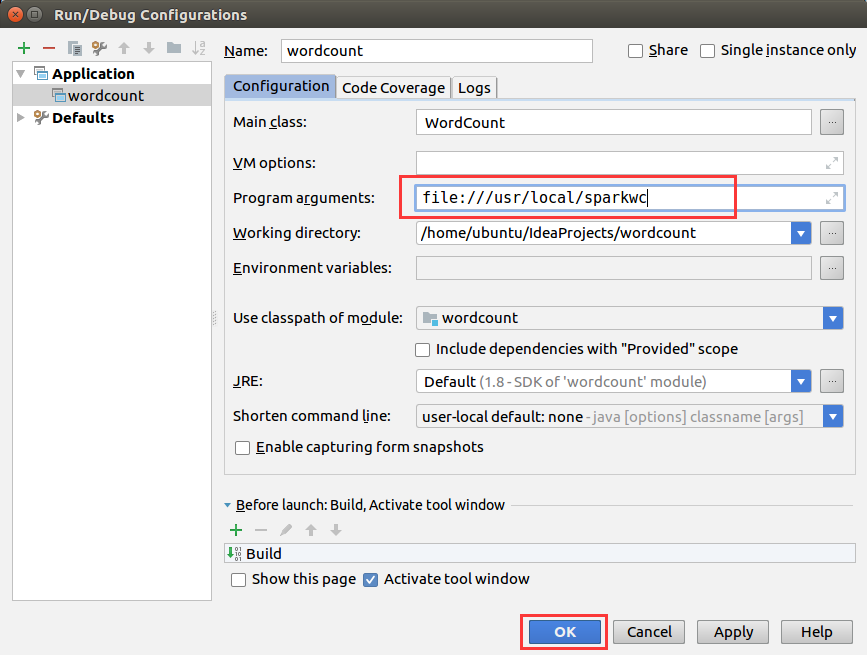


图2-75 设置参数

**第三步，IDEA直接运行**

在代码空白处，鼠标右键点击，然后选择【Run】，如图2-76。

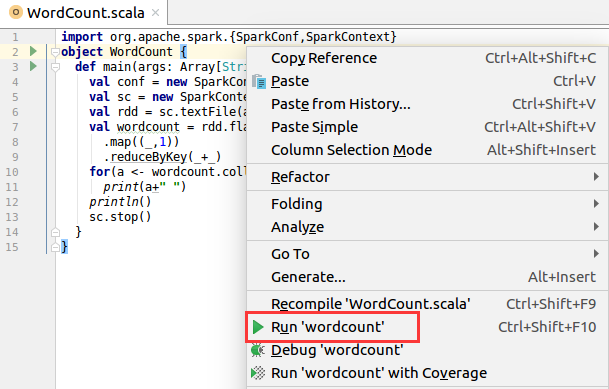


图2-76 运行

**第四步，查看结果**

在IDEA界面下方，将出现运行结果。运行过程需要等候数十秒，红色字体为运行日志，黑色字体为运行结果，如下图2-77所示。

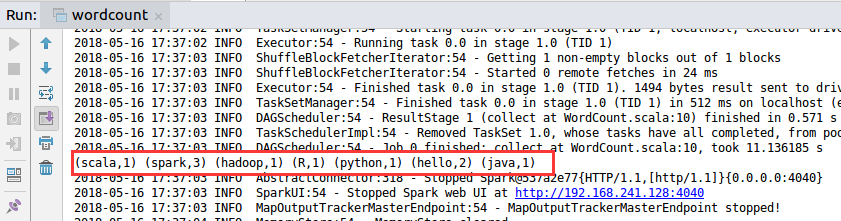


图2-77 运行结果

**2.3.4 IDEA打包运行**

在2.3.3小节中创建的WordCount项目基础上，本小节将介绍使用IDEA打包项目，使用spark submit提交spark集群运行。打包2.3.2中项目的过程与本小节介绍的流程相同。具体流程为：

**第一步，生成JAR包**

打包运行需要重新打开项目配置页，具体操作是：点击菜单栏【File】，然后点击【Project Structure】，进入项目配置页，如下图所示。

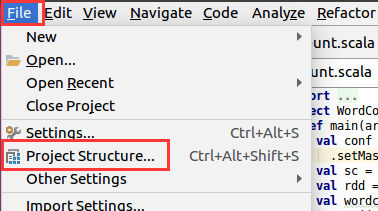


图2-78 打开配置页

选择【Artifacts】，然后在界面的中间操作界面中选择【+】号，选择添加【JAR】包的【From modules with dependencies】，如下图所示。

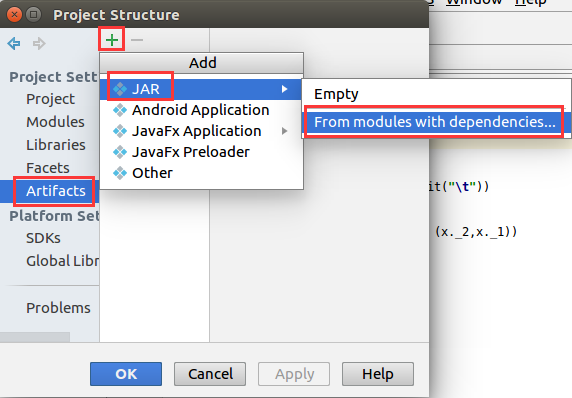


图2-79 设置Artifacts

然后出现图2-80界面，设置【Main Class】，点击右侧的【...】，出现路径选择。如图2-81所示。

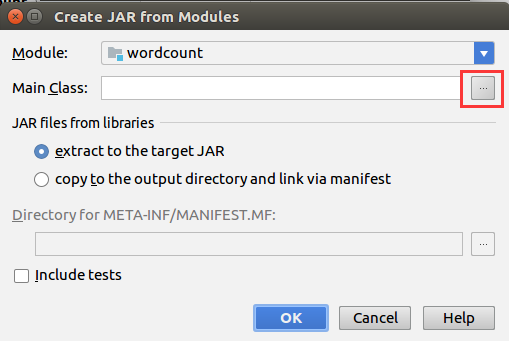


图2-80 设置主函数入口

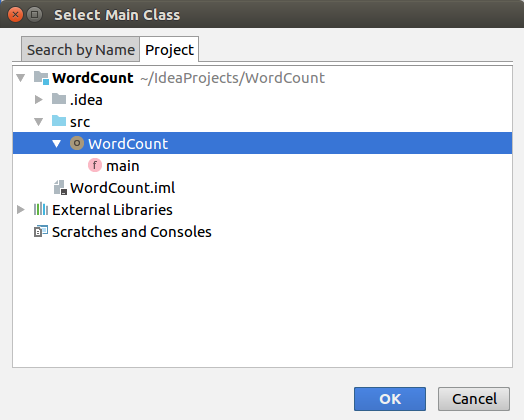


图2-81 选择wordcount路径

点击【Project】，然后在项目wordcount目录下，找到WordCount，点击，最后选择【OK】，然后回到图2-82，点击【OK】。配置页的具体配置完成，出现如下图所示结果。可以在最右侧【Output directory】中修改jar包的保存位置。注意：jar包的保存位置，在spark submit运行时会有用。

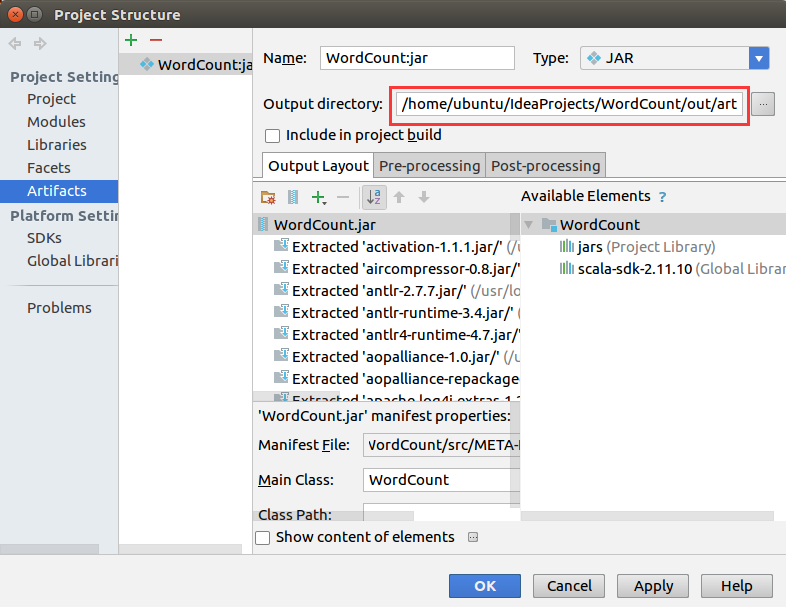


图2-82 打包配置

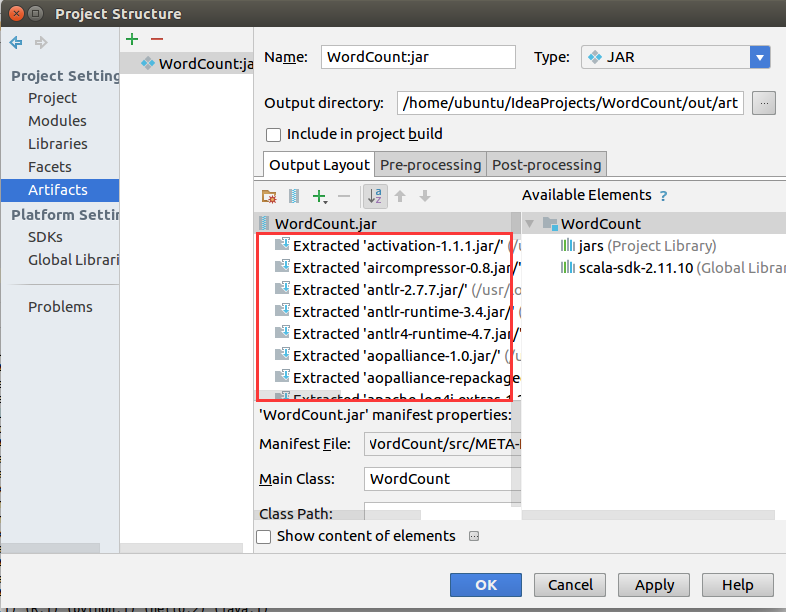


图2-83 打包配置

删除图2-83中红框内的多余的部分，仅保留WordCount.jar以及’WordCount’ Compile output。可以利用全选按键Ctrl+A，选中红框内全部选项，然后使用Ctrl+鼠标左键进行反选，即按住Ctrl键的同时，用鼠标左键分别点击WordCount.jar以及’WordCount’ Compile output，从而保留这两项，最后点击红框上方的【-】执行删除。

点击【OK】，回到项目页面，需要注意的是，打包之前，删除IDEA直接运行时，代码中，对conf的setMaster(“lcoal”)配置可以选择删除，因为在spark submit运行jar包时，会重新指定运行模式。在菜单栏单击【Build】，然后点击【Build Artifacts】生成项目的打包文件。如下图所示：

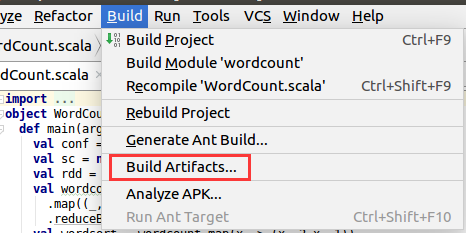


图2-83 jar包生成

等待几分钟以后，jar生成，存储位置在图2-84中框中路径。在ubuntu系统中，根据该路径，找到jar包。本文中，路径为/home/ubuntu/IdeaProjects/wordcount/out/artifacts/WordCount.jar。如下图所示。

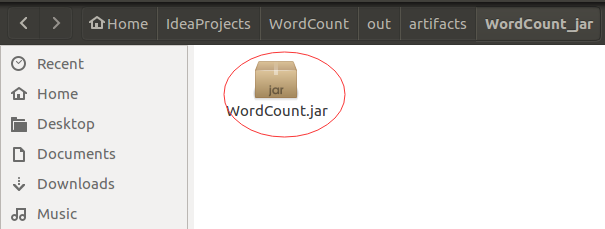


图2-84 jar包位置

将jar复制到spark安装的根目录下。在图2-84中，新建一个终端，然后输入：

**sudo cp WordCount.jar /usr/local/spark-2.3.0-bin-hadoop2.7**

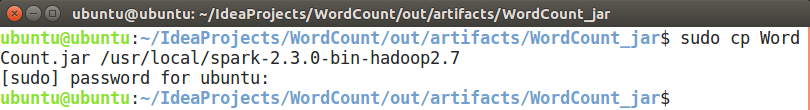


图2-85 复制jar包

**第二步，Spark submit运行jar包**

通过spark-submit提交运行jar包的命令格式为：

./bin/spark-submit

--class main-class //需要运行的程序的主类，应用程序的入口点

--master master-url //Master URL，下面会有具体解释

--deploy-mode deploy-mode //部署模式

... # other options

application-jar //应用程序JAR包

[application-arguments] //传递给主类的主方法的参数

在本小节的实例中，main-class为WordCount，master-url为local，deploy-node和other options缺省，application-jar为WordCount.jar，[application-arguments]为单词计数文件的目录：[file:///usr/local/sparkwc。](file://usr\\local\\sparkwc。)

将本小节生成的jar包提交spark-submit运行的步骤为：在spark的安装目录（/usr/local/ spark-2.3.0-bin-hadoop2.7）下，打开一个终端，输入：

**bin/spark-submit --class WordCount --master local WordCount.jar [file:///usr/local/sparkwc](file://usr\\local\\sparkwc)**

可以查看到运行结果如下图所示，结果为红框中内容：

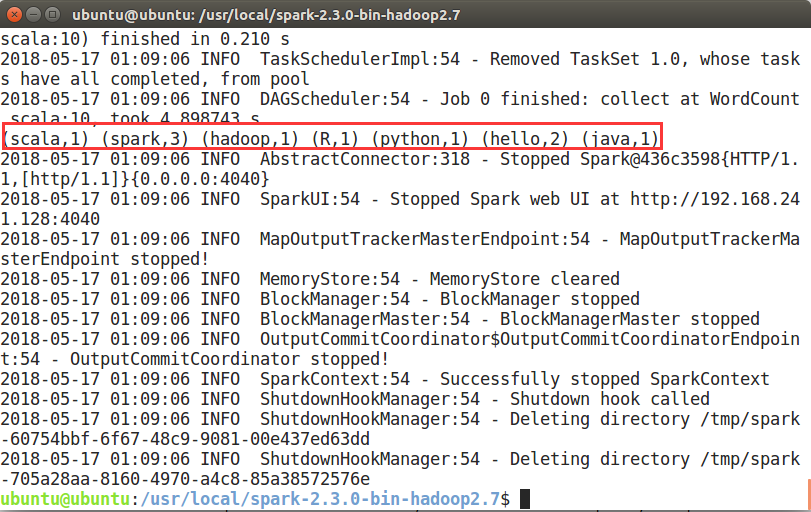


图2-86 spark-submit运行jar包

**2.4 小结**

Spark提供了丰富的简单实例源代码，存放在Spark安装目录下examples文件夹中。通过阅读源代码并进行Spark开发实践，可以更加深入理解Spark核心原理。在本章中，先介绍了Linux环境准备，然后在此基础上详细介绍了Spark的local部署模式的流程；IDEA创建并运行Spark实例的流程；IDEA对编译好的代码进行打包，接着将打包程序通过Spark-submit提交运行的流程。Spark实战开发环境，在本章中已基本搭建完成。利用该环境，可以快速开发应用程序和阅读源代码。

2.5 wrfewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewewew