BA-BEAD Big Data Engineering for Analytics

BEAD Workshop Series

Spark Scala IDE Project Setup

Table of Contents

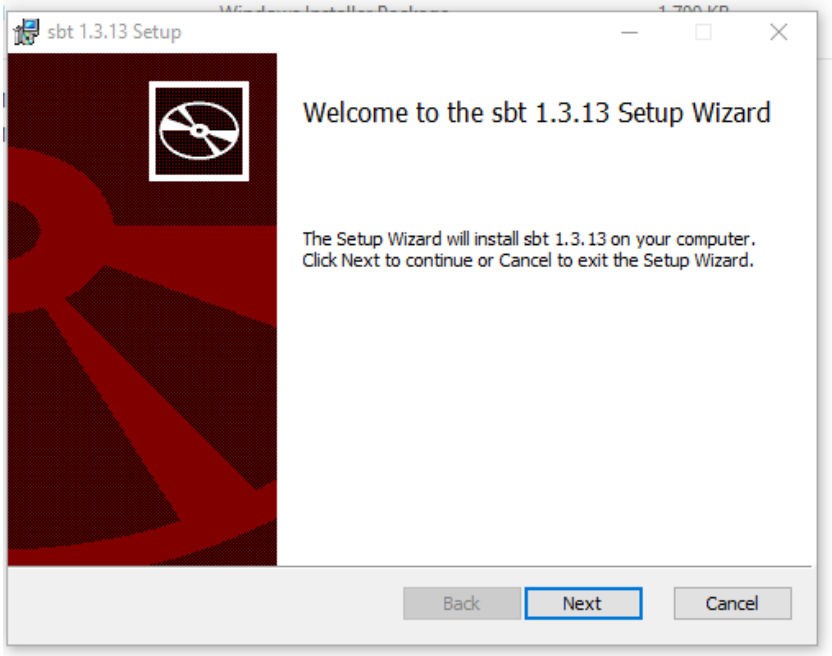
Introduction

This set of instructions would guide you through the configuration of Scala IDE (Eclipse family) and Scala Built Tool download link *(Please note current version is 1.8.2, but workshop may point to 1.3.13 nevertheless steps remain same)* on Windows Platform needful to execute the Spark codes.

Scala Built Tool Installation

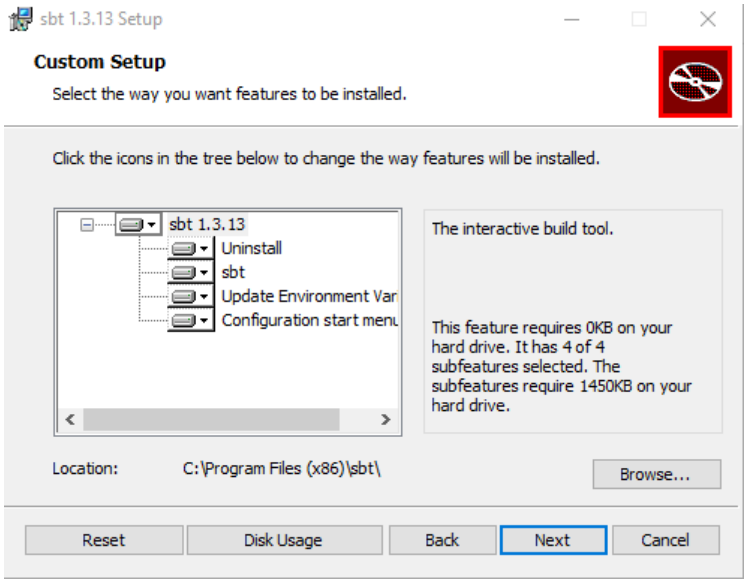
sbt is a build tool for Scala, and Java. It requires Java 1.11 or later and Scala. Make sure you have them both working by testing in the command prompt.

A Scala project build is usually defined via a build.sbt file, which consists of a set of projects. If the required version of libraries demanded by a project is not available locally, the sbt launcher will download it. If this file is not present online, the sbt launcher will choose an arbitrary version from its own repository. Start by double clicking the sbt installer file (usual extention *.msi) and the wizard below pops, click Next

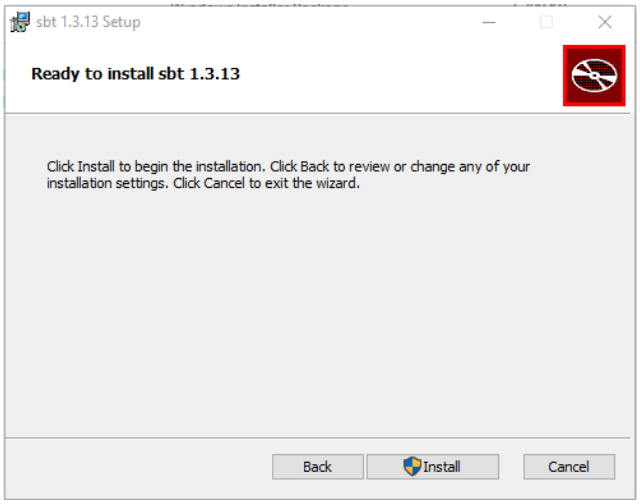


Accept license and click Next

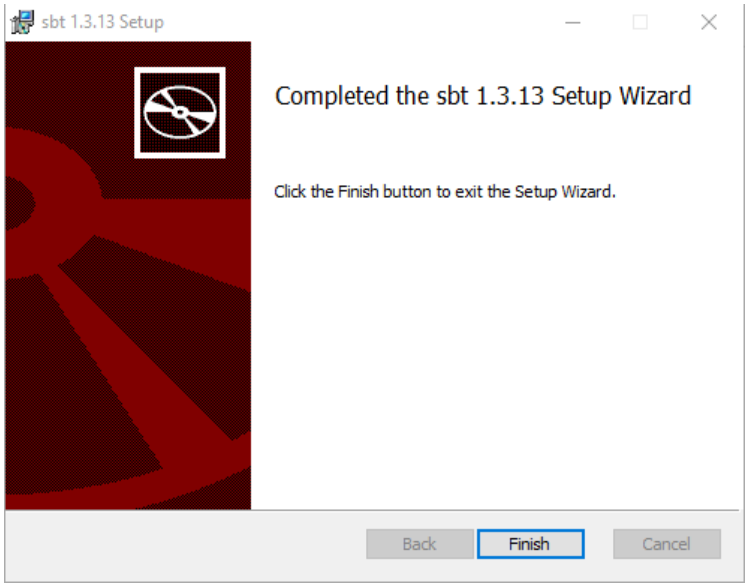
Accept the default folders and click Next



Select Install.



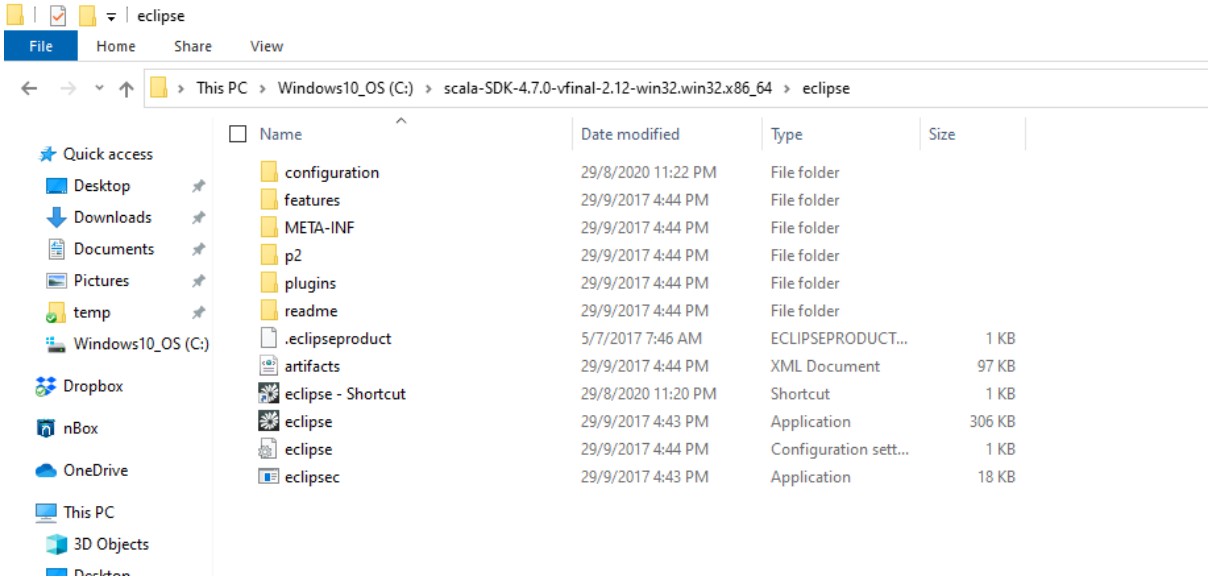
Allow the installation to proceed and select Finish.




```
C:\Users\issspa>sbt -version
sbt version in this project: 1.3.13
sbt script version: 1.3.13
```

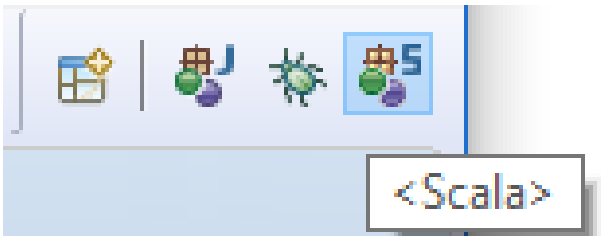
Scala IDE Installation

Eclipse supports a lot of different programming languages. Download the self-extracting zip file and point it to C folder. The below folder structure is observed. Scala IDE is started by double clicking the eclipse application shown below.



After installing Scala IDE, Scala is one of them. Eclipse can optimize its user interface for a specific programming language or environment, by offering different perspectives. Scala IDE adds its own perspective to Eclipse. To switch to the Scala IDE perspective, do the following:

- On the upper right-hand side of the screen, there are a few buttons that represent the last used perspectives. Hover over them; one of them will have the <Scala> tooltip. Click on it:



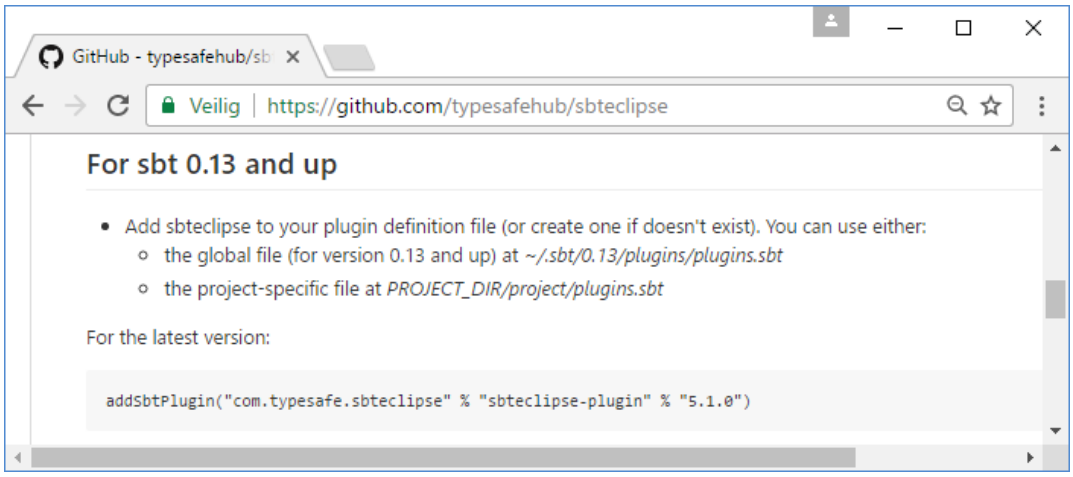
- If the Scala perspective button is unavailable, click on the button with the Open
- Perspective tooltip and select the <Scala> perspective from the list. The button will now be added to the toolbar.

The IDE's user interface is now optimized for Scala programming.

You can switch to a different perspective at any time. Eclipse IDE is well suited for working with projects that use different programming languages.

Loading the SBTEclipse plugin

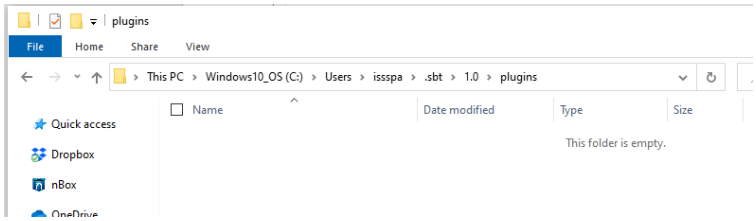
To find out which version of the plugin you have to install, visit the project page at `http://github.com/typesafehub/sbteclipse`:



The GitHub project page has instructions to add the plugin to your eclipse project settings. You should look for a line that starts with `addSbtPlugin`. In my case, it mentioned the following:

```
addSbtPlugin("com.typesafe.sbteclipse" % "sbteclipse-plugin" % "5.1.0")
```

In the `.sbt` subdirectory of `C:/User/XXXX/.sbt/1.0` (in my case issspa), create a new subfolder called **`plugins`** as shown below:



Create a **`plugins.sbt`** file using the following command in the command prompt:

```
> start notepad plugins.sbt
```

It will prompt you to create a new file, answer yes and add the addSbtPlugin line from the git link mention above. The GitHub project page has instructions to add the plugin to your eclipse project settings. You should look for a line that starts with `addSbtPlugin`. In my case, it mentioned via one line as shown below:

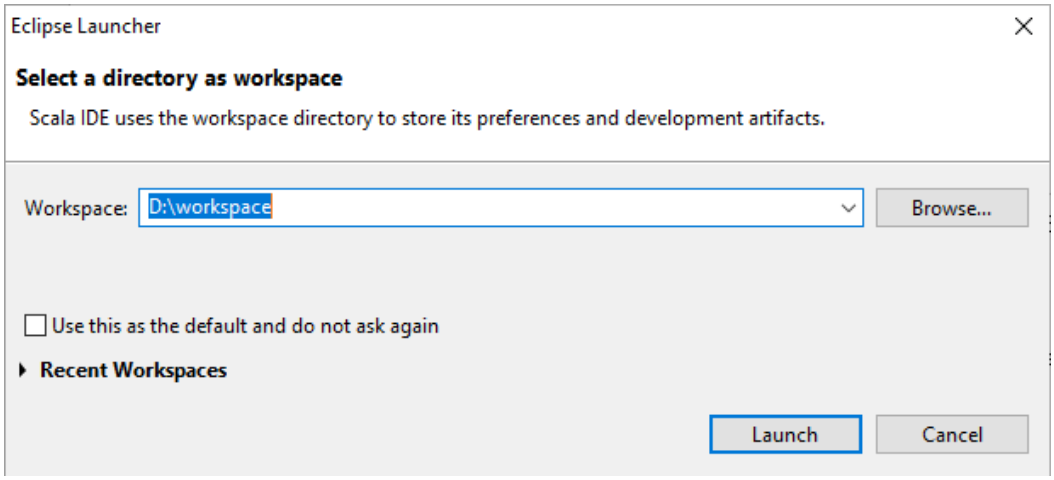
```
addSbtPlugin("com.typesafe.sbteclipse" % "sbteclipse-plugin" % "5.1.0")
```

Save the file and proceed with further setup of project structure.

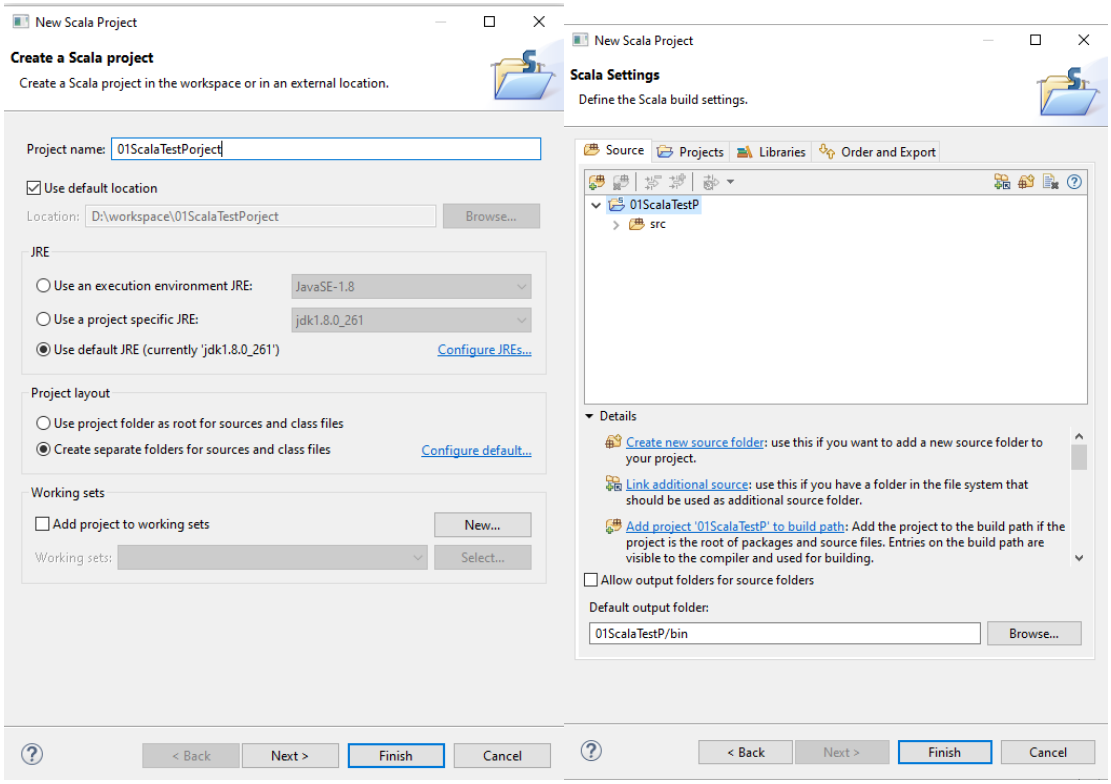
Ensure that the file is saved in the `plugins` subdirectory; otherwise, SBT will not find the file.

Generating a new Scala project

Open Scala IDE. Choose a default workspace folder and click launch. In my case D:\workspace

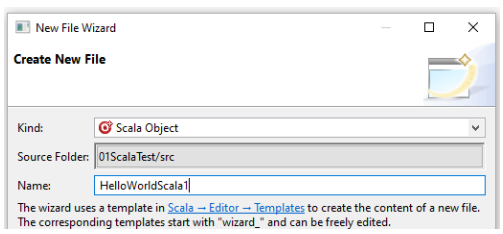


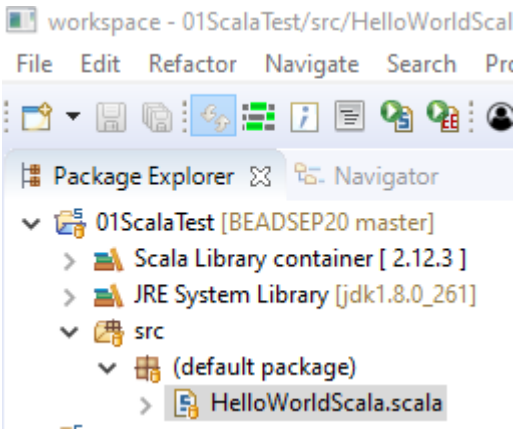
Select File > New > Project from the menu bar and populate the wizard as below:



Edit Project Name: 01ScalaTest, accept the other defaults, click Next and Click Finish.

Select the src folder and choose new > file > scala object. Type name as **HelloWorldScala**

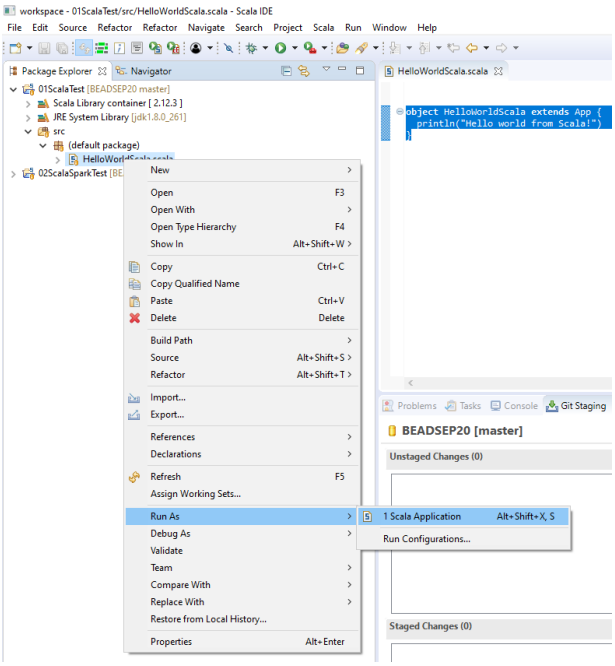




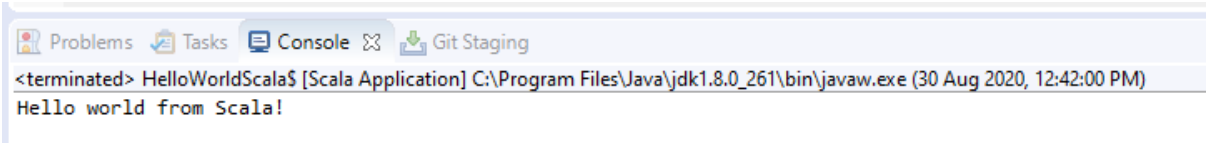
Type the following Scala code inside the newly created Scala Object:

```
object HelloWorldScala extends App {
  println("Hello world from Scala!")
}
```

You can then execute the code by selecting the HelloWorldScala file in project explorer window. Right click, in the menu choose Run As > Scala Application.



The output is similar to the below screenshot



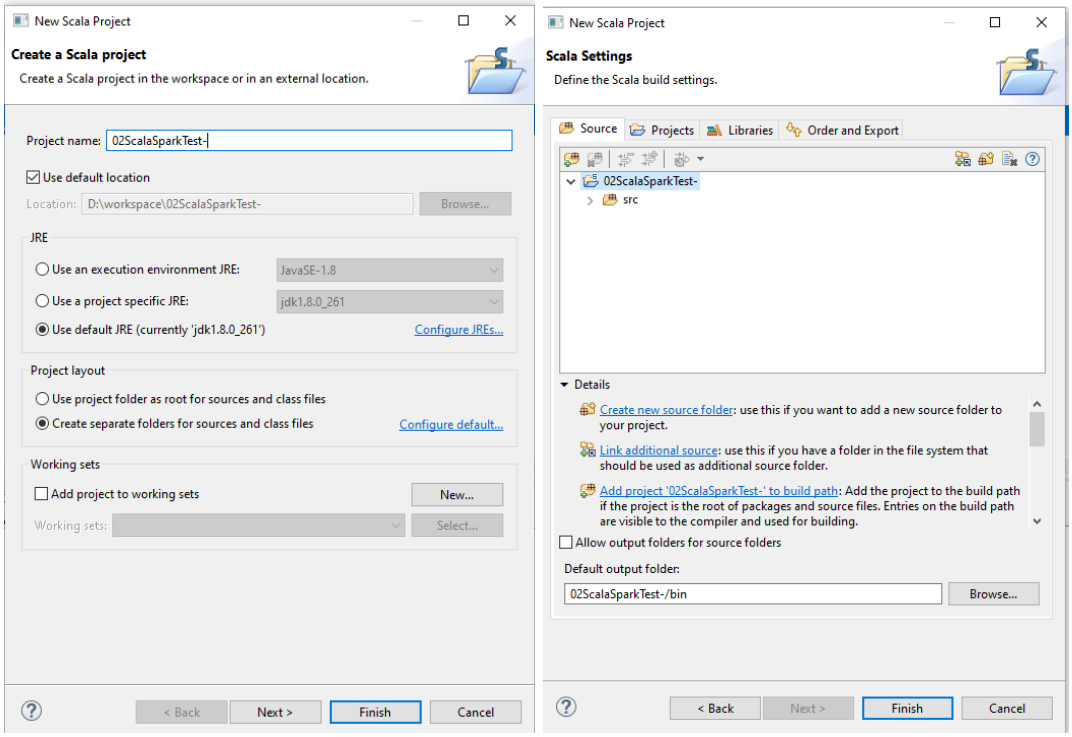
You have tested the Scala IDE and Scala language compiler. Both work fine. As a next step we will proceed to create a Scala project with relevant Spark Libraries link.

Generating a new Spark Scala Project

The whole process is broken into four major steps. (Step 1) Creation of project using Scala IDE. (Step 2) Activate libraries using sbt tool (Step 3) Reset Scala compiler and source folder in Scala IDE and finally (Step 4) Test Scala Spark Code.

Step 1 Create a Scala Project

Select File > New > Project from the menu bar and populate the wizard as below:

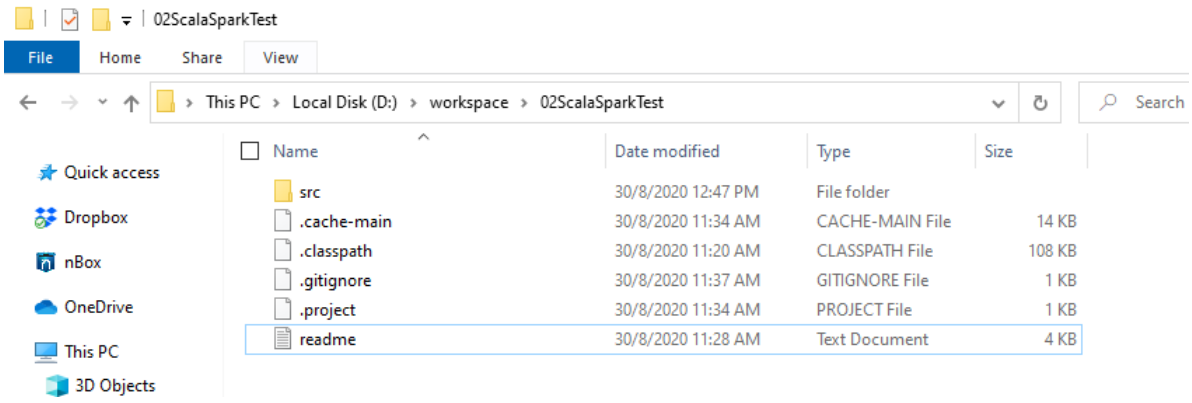


Edit Project Name: 02ScalaSparkTest, accept the other defaults, click Next and Click Finish.

Close the Scala IDE.

Step 2 Activate Spark libraries using sbt

Open windows explorer and point to the folder D:\workspace\02ScalaSparkTest. Here create a build.sbt file using command prompt in the same folder location.




```
> start notepad build.sbt
```

Put the following content inside the file:

```
name := "02ScalaSparkTest"
version := "1.0"
scalaVersion := "2.11.11"
val sparkVersion = "2.2.0"

resolvers ++= Seq(
  "apache-snapshots" at "http://repository.apache.org/snapshots/"
)

libraryDependencies ++= Seq(
  "mysql" % "mysql-connector-java" % "5.1.+",
  "org.apache.spark" %% "spark-core" % sparkVersion,
  "org.apache.spark" %% "spark-sql" % sparkVersion,
  "org.apache.spark" %% "spark-mllib" % sparkVersion,
  "org.apache.spark" %% "spark-streaming" % sparkVersion,
  "org.apache.spark" %% "spark-hive" % sparkVersion
)
```

Create a readme.txt file in the same folder and put any text content inside the same file. In my case readme.txt looks as below:

```
Spark Overview
Apache Spark is a fast and general-purpose cluster computing system.
It provides high-level APIs in Java, Scala, Python and R, and an
optimized engine that supports general execution graphs. It also
supports a rich set of higher-level tools including Spark SQL for
SQL and structured data processing, MLlib for machine learning,
GraphX for graph processing, and Spark Streaming.

Security
Security in Spark is OFF by default. This could mean you are
vulnerable to attack by default. Please see Spark Security before
downloading and running Spark.

Downloading
. . .
```

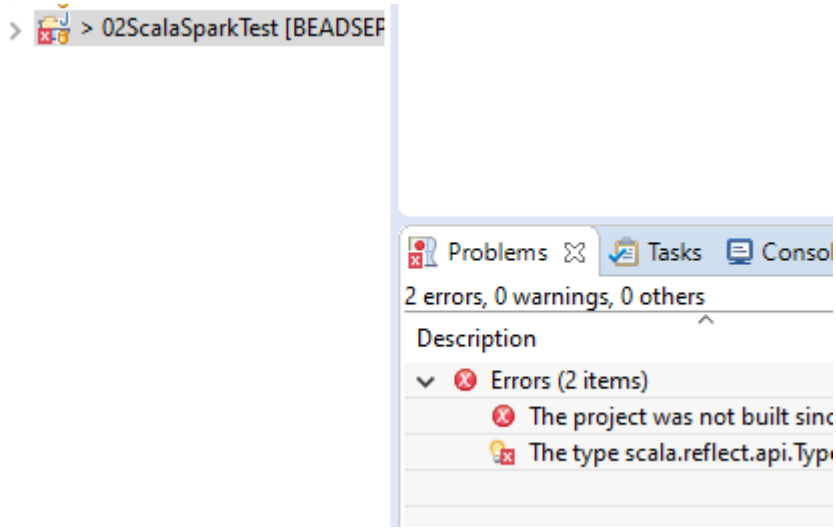
In the same location, create a new command prompt and fire the sbt for eclipse plugin as show below. Run the following command in the D:\workspace\02ScalaSparkTest folder:

```
> sbt eclipse
```

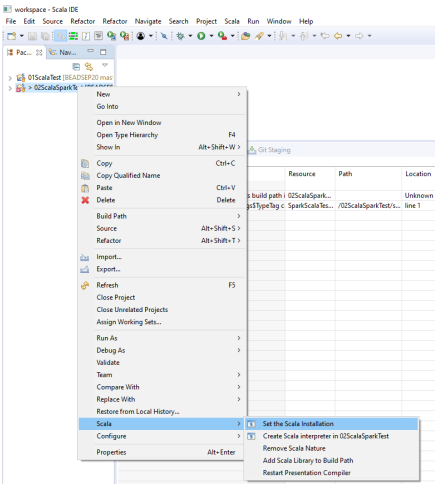
The `sbteclipse` plugin has now generated project files inside the SBT project's directory that Eclipse IDE with the installed Scala IDE plugin can import. SBT will download and activate the plugin. From now on, when starting SBT in this project directory, you will be able to create or update an Eclipse IDE/Scala IDE project.

Step 3: Reset Scala compiler and source folder

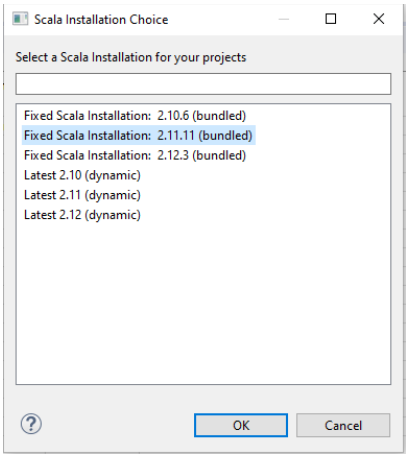
Reopen the ScalaIDE and notice that there are compiler errors for the same project 02ScalaSparkTest.



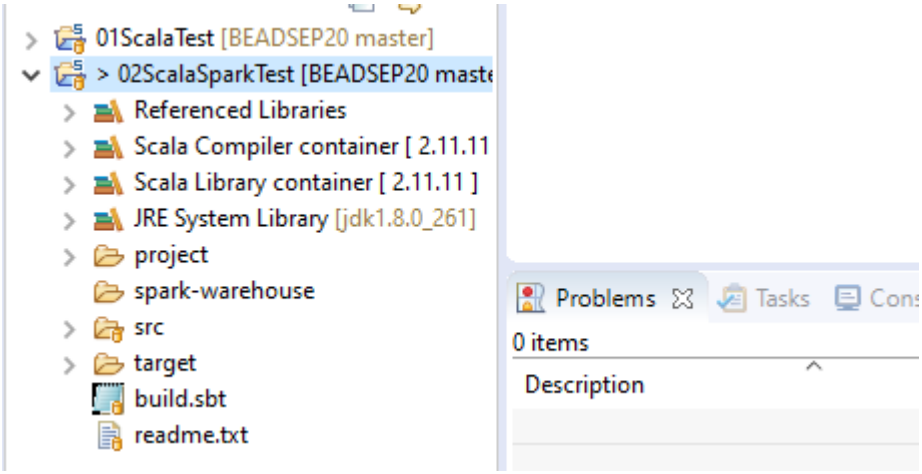
Select the 02ScalaSparkTest project, right click, choose Scala Menu > Scala > Set Scala Installation from the menu.



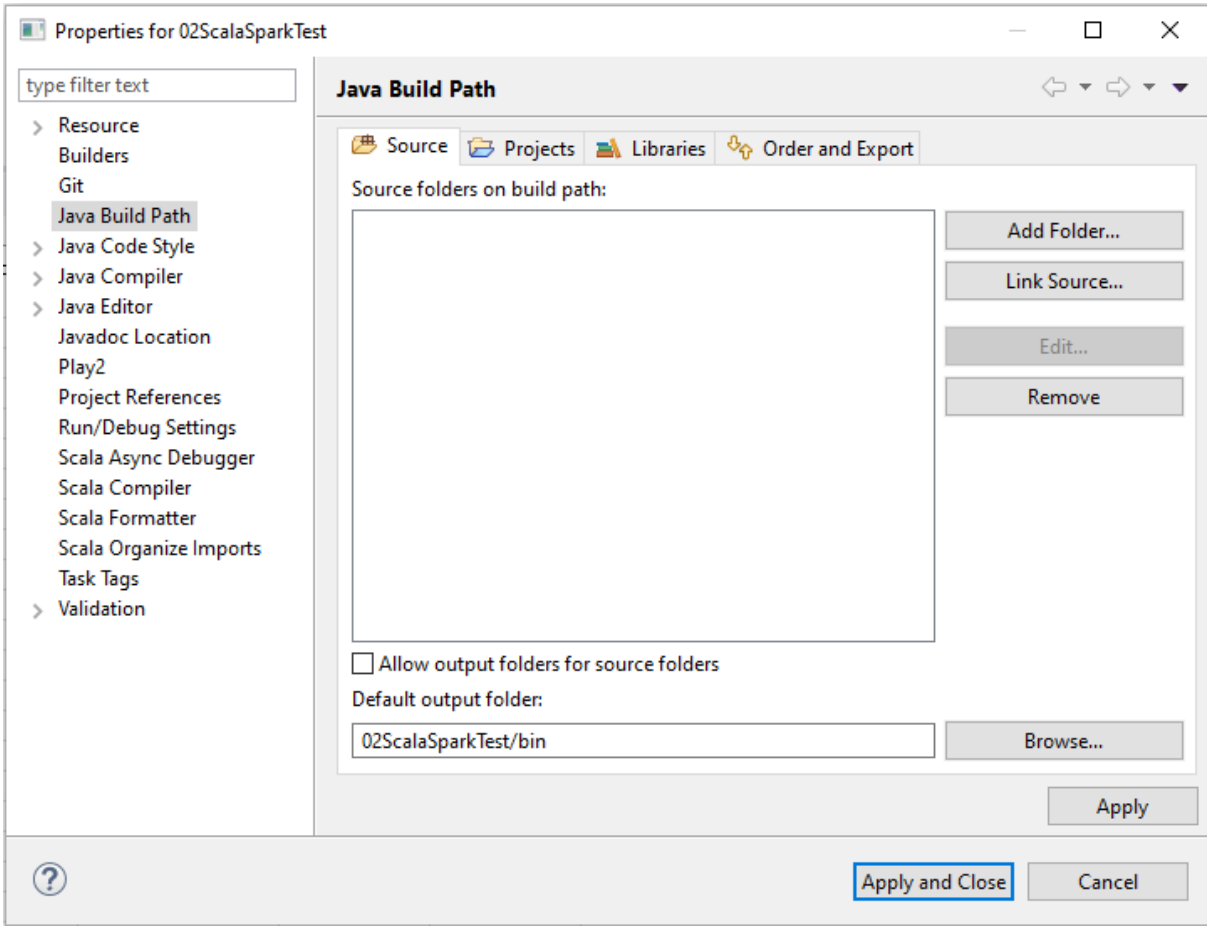
Choose Scala 2.13.X Fixed Bundle from the menu



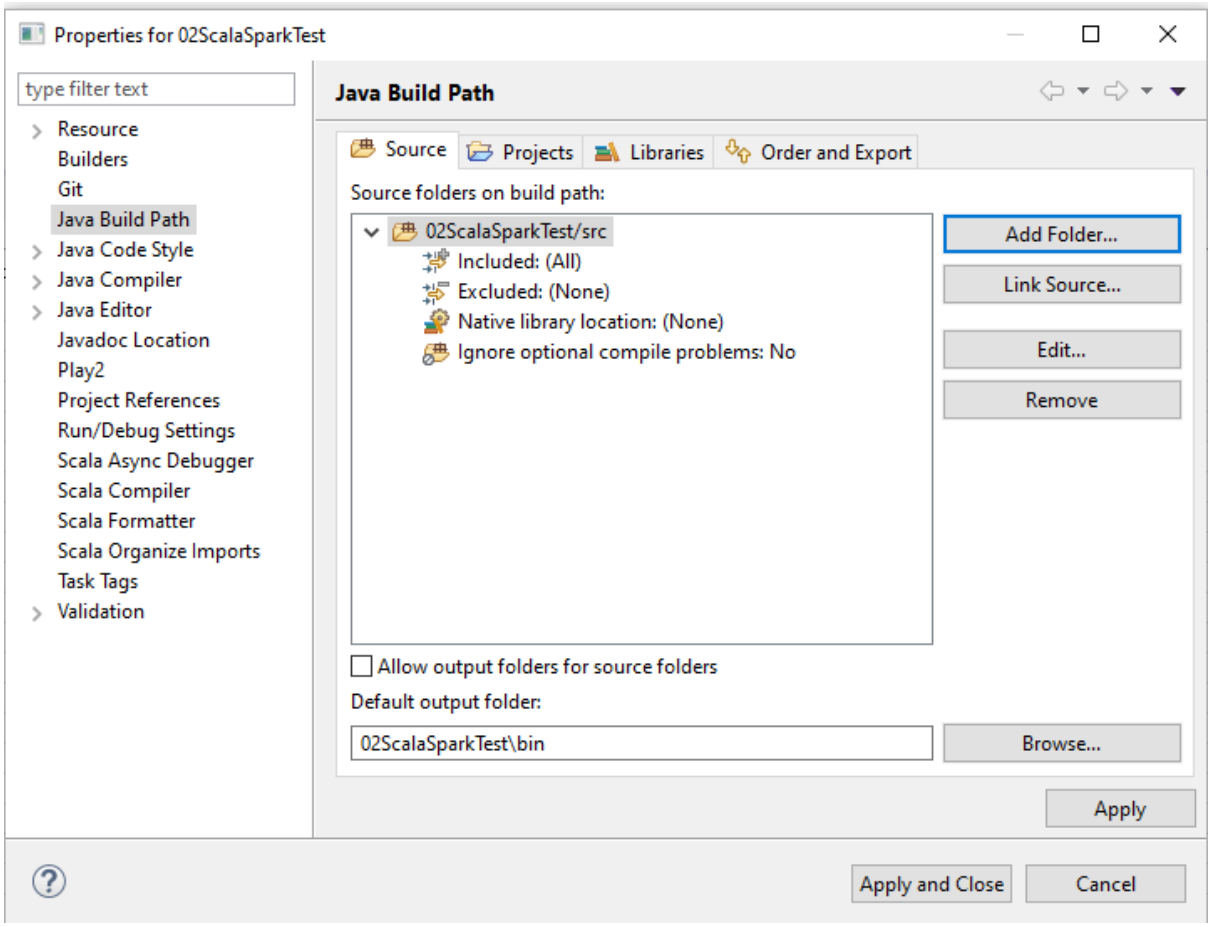
Observe that the problems are cleared of compiler errors.



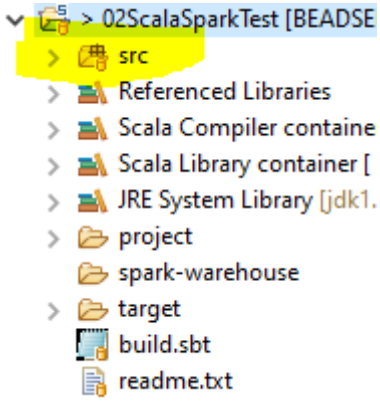
Reset the source folder (src in this case) in the ScalaIDE by selecting the 02ScalaSparkTest project and right clicking. Choose properties. Choose Java Build Path in the left menu and choose the source tab as displayed below:



Select Add Folder. Point to the src folder. This will activate the src folder as source of codes that the IDE will execute. The selection looks as shown in the screen below. Choose Appy and Close to finalise the settings.



The final project settings including src as code folder looks as below:



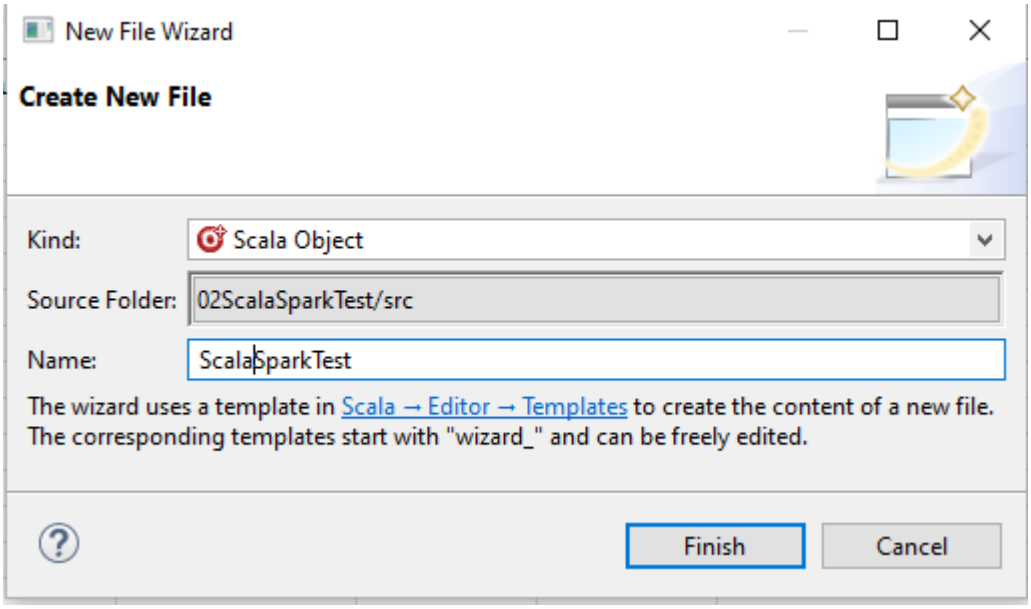
Step 4: Test the Scala Spark Codes

The final step is to test the spark codes inside Scala IDE. For this we need to create a new Scala Object. Choose the following from menu:

File > New > Scala Object

Input the name as SparkScalaTest

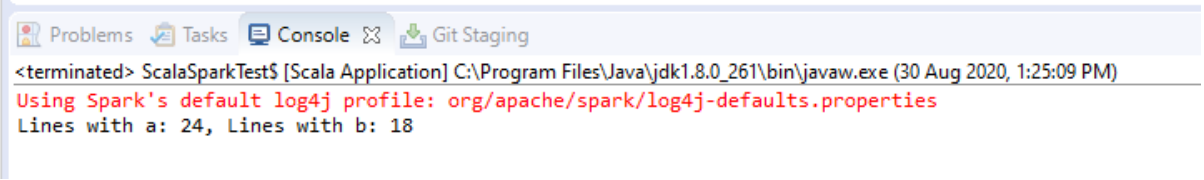
Click Finish



Type the code inside the Scala editor as below:

```
import org.apache.spark.sql.SparkSession
import org.apache.log4j.Level
import org.apache.log4j.Logger
object ScalaSparkTest extends App {
    Logger.getLogger("org").setLevel(Level.OFF)
    // Should be some file on your system
    val logFile = "D:/workspace/02ScalaSparkTest/readme.txt"
    val spark = SparkSession.builder.appName(getClass.getSimpleName)
                .master("local[2]").getOrCreate()
    val logData = spark.read.textFile(logFile).cache()
    val numAs = logData.filter(line => line.contains("a")).count()
    val numBs = logData.filter(line => line.contains("b")).count()
    println(s"Lines with a: $numAs, Lines with b: $numBs")
    spark.stop()
}
```

Execute the file by selecting the ScalaSparkTest file, right click, Run As and Choosing Scala Application, yields the below output



Conclusion

Thus we have effectively learnt to set up the workspace and run Spark framework via Scala, sbt and Scala IDE tools.

Have fun coding. Cheers to the craft of creating clean code.