

## BA-BEAD Big Data Engineering for Analytics

# BEAD Workshop Series

## Lab Setup Instructions



**©2015-18 NUS. The contents contained in this document may not be reproduced in any form or by any means, without the written permission of ISS, NUS other than for the purpose for which it has been supplied.**

## Table of Contents

Introduction .....	3
Java Virtual Machine .....	3
Scala Binaries .....	4
Scala IDE .....	5
Download Spark .....	8
WinUtils .....	8
Environment Variables .....	9
Permissions for the folder tmp/hive .....	10
Testing .....	11

# Spark on Windows 10

## Introduction

The goal of this post is to help people to install and run Apache Spark in a computer with window 10 platform without much hassle. This post is composed of pieces of installation content—somewhat similar to LEGO bricks—that you can work around piece by piece. Hope you find the post useful to get started with your experiments on Apache Spark framework.

*Note: If you really want to build a serious prototype, we strongly recommend to install one of the quick start virtual machines mentioned in the classroom.*





Getting Spark installed on Windows involves several steps that I will walk you through here.

## Java Virtual Machine

**Install a JDK:** You need to first install a JDK, that's a Java Development Kit. You can just go to Sun's website and download that and install it if you need to.

Java SE Development Kit 8u181		
You must accept the <a href="#">Oracle Binary Code License Agreement for Java SE</a> to download this software.		
Thank you for accepting the Oracle Binary Code License Agreement for Java SE; you may now download this software.		
Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	72.95 MB	<a href="#">jdk-8u181-linux-arm32-vfp-hflt.tar.gz</a>
Linux ARM 64 Hard Float ABI	69.89 MB	<a href="#">jdk-8u181-linux-arm64-vfp-hflt.tar.gz</a>
Linux x86	165.06 MB	<a href="#">jdk-8u181-linux-i586.rpm</a>
Linux x86	179.87 MB	<a href="#">jdk-8u181-linux-i586.tar.gz</a>
Linux x64	162.15 MB	<a href="#">jdk-8u181-linux-x64.rpm</a>
Linux x64	177.05 MB	<a href="#">jdk-8u181-linux-x64.tar.gz</a>
Mac OS X x64	242.83 MB	<a href="#">jdk-8u181-macosx-x64.dmg</a>
Solaris SPARC 64-bit (SVR4 package)	133.17 MB	<a href="#">jdk-8u181-solaris-sparcv9.tar.Z</a>
Solaris SPARC 64-bit	94.34 MB	<a href="#">jdk-8u181-solaris-sparcv9.tar.gz</a>
Solaris x64 (SVR4 package)	133.83 MB	<a href="#">jdk-8u181-solaris-x64.tar.Z</a>
Solaris x64	92.11 MB	<a href="#">jdk-8u181-solaris-x64.tar.gz</a>
Windows x86	194.41 MB	<a href="#">jdk-8u181-windows-i586.exe</a>
Windows x64	202.73 MB	<a href="#">jdk-8u181-windows-x64.exe</a>

We need the JDK because, even though we're going to be developing in Python or Scala during this course. Even Python gets translated under the hood to Scala code, which is what Spark is developed in natively. And, Scala, in turn, runs on top of the Java interpreter. So, in order to run Python code, you need a Scala system, which will be installed by default as part of Spark. Also, we need Java, or more specifically Java's interpreter, to actually run that Scala code. It's like a technology layer cake. (applications configuration shown after installation below)

 Intel® Hardware Accelerated Execution Manager	Intel Corporation	12/6/2018	646 KB	6.2.0
 ISI ResearchSoft - Export Helper		16/8/2018		
 Java 8 Update 181 (64-bit)	Oracle Corporation	5/10/2018	115 MB	8.0.1810.13
 Java SE Development Kit 8 Update 181 (64-bit)	Oracle Corporation	5/10/2018	345 MB	8.0.1810.13
 JMP Pro 13	SAS Institute Inc.	11/6/2018		13.0.0

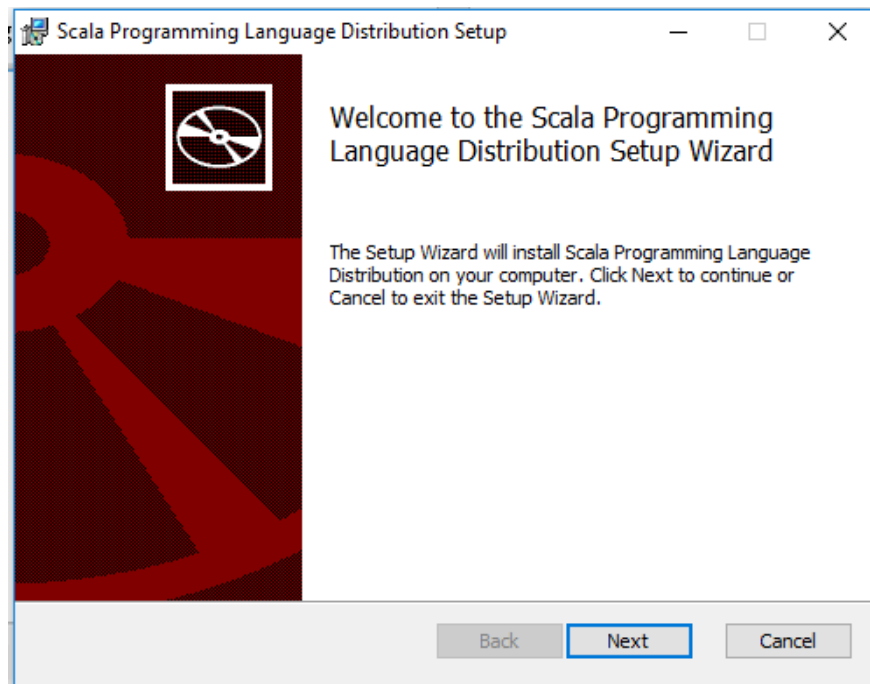
JDK 8 is a superset of JRE 8, and contains everything that is in JRE 8, plus tools such as the compilers and debuggers necessary for developing applets and applications. JRE 8 provides

the libraries, the Java Virtual Machine (JVM), and other components to run applets and applications written in the Java programming language.

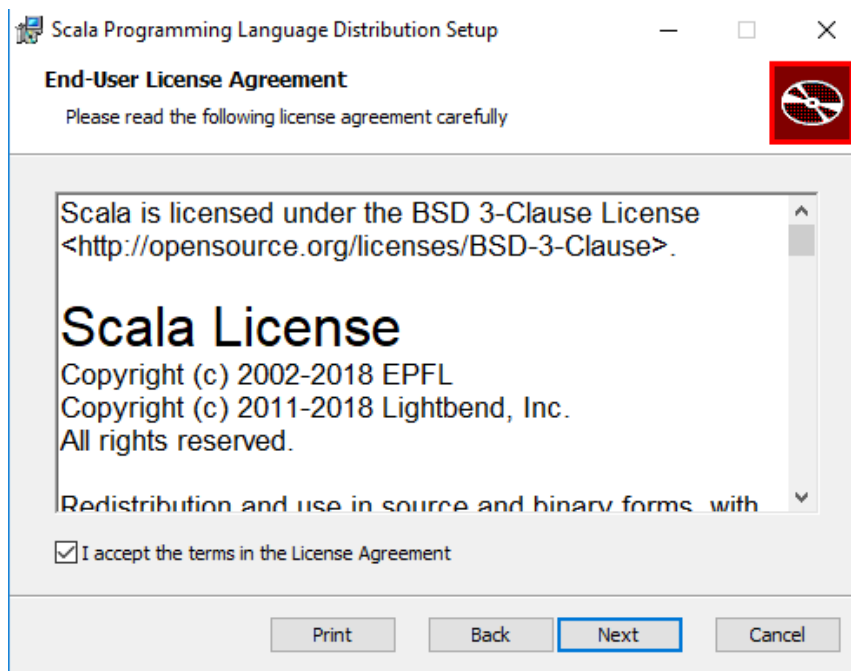
## Scala Binaries

Scala is a modern multi-paradigm programming language designed to express common programming patterns in a concise, elegant, and type-safe way. It smoothly integrates features of object-oriented and functional languages.

1. Download the Scala binaries for Windows



2. Accept the agreement. Select Next and continue to complete installation.



3. You can verify Scala installation in folder: C:\Program Files (x86)\scala

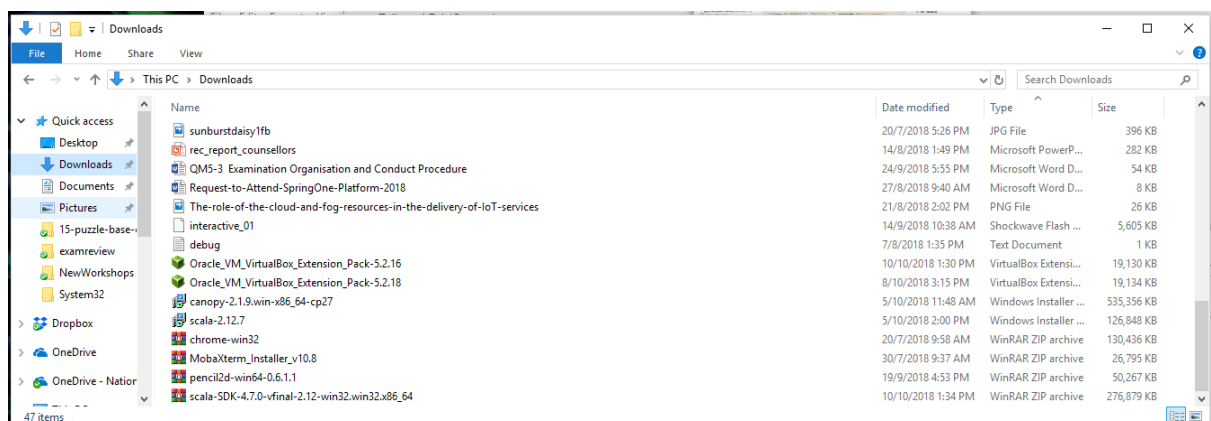
## Scala IDE

Scala IDE provides advanced editing and debugging support for the development of pure Scala and mixed Scala-Java applications. While one free to use the python shell and interactive interpreter such as Jupyter or Spyder, we will assume pure Scala development and try Scala IDE.

1. Download Scala IDE (green button) installer archive. (<http://scala-ide.org/> )

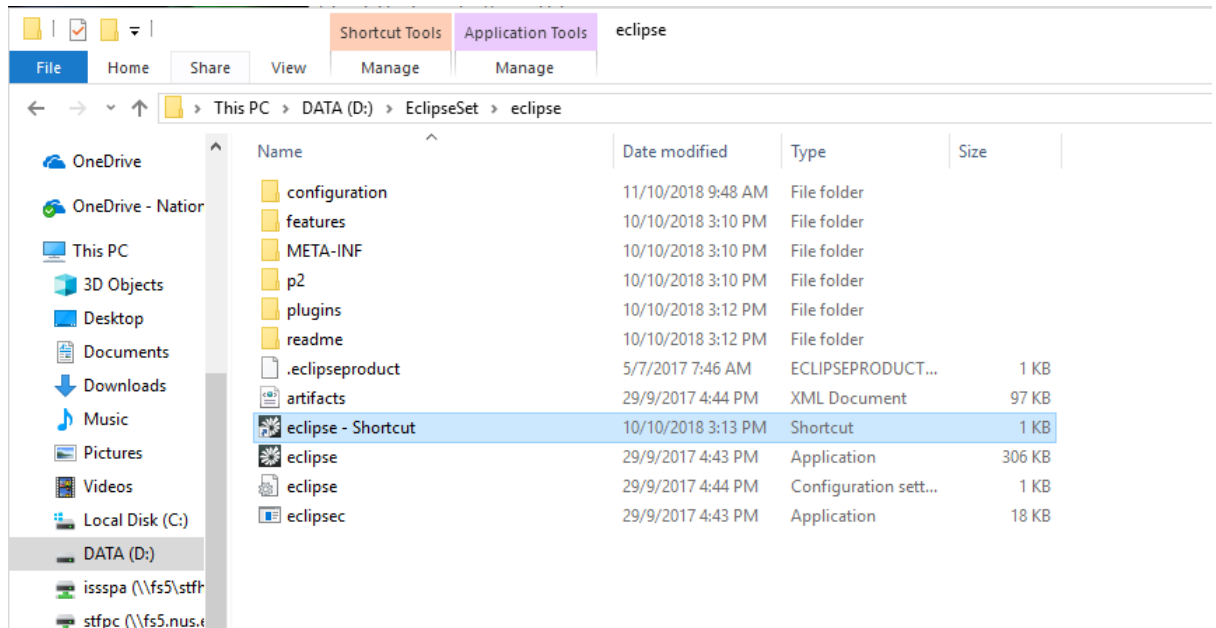


2. Pick Windows 64 bit version
3. Save it under download folders.
4. Choose the installer and file associations.

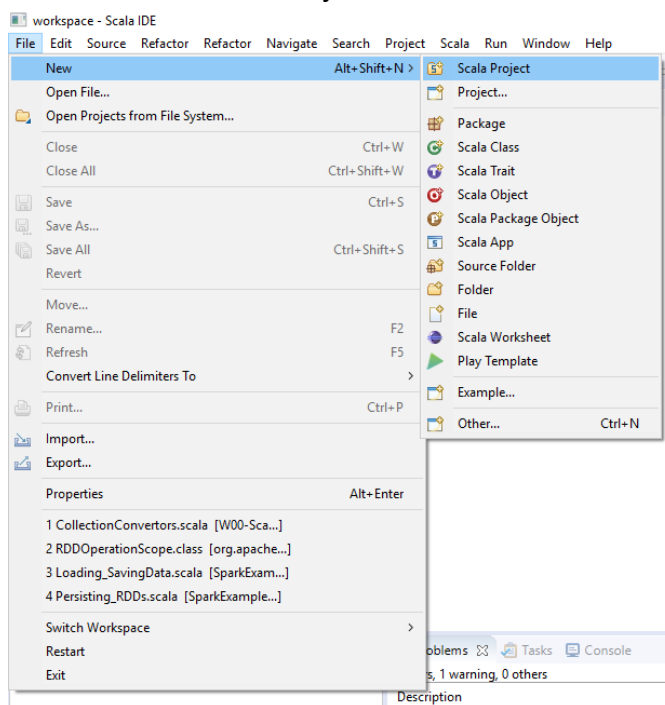


5. Move the archive to D drive and unzip using winrar or 7 zip application. It creates a folder by name eclipse.

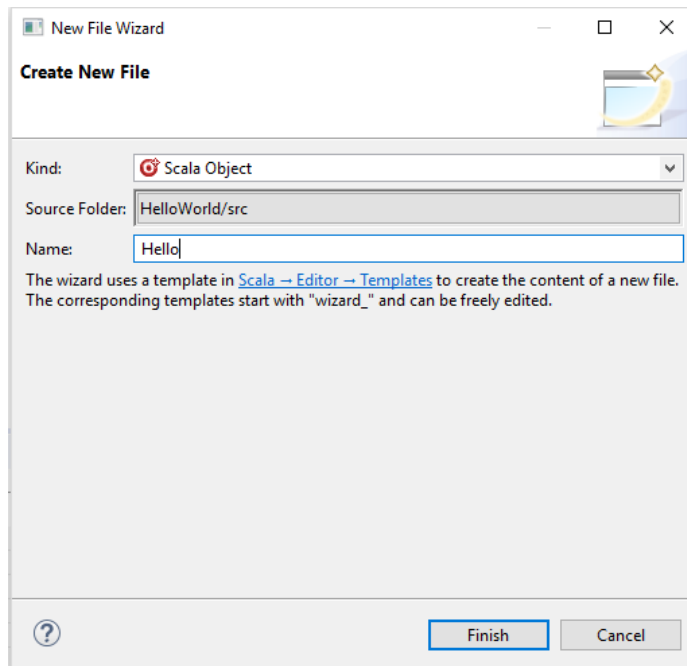
6. Right click on eclipse application and create a short cut.



7. Send the shortcut to desktop and rename it as you want.  
 8. Open up Scala IDE and launch it in a workspace of your choice (my case D:\workspace).  
 9. File => New => Scala Project



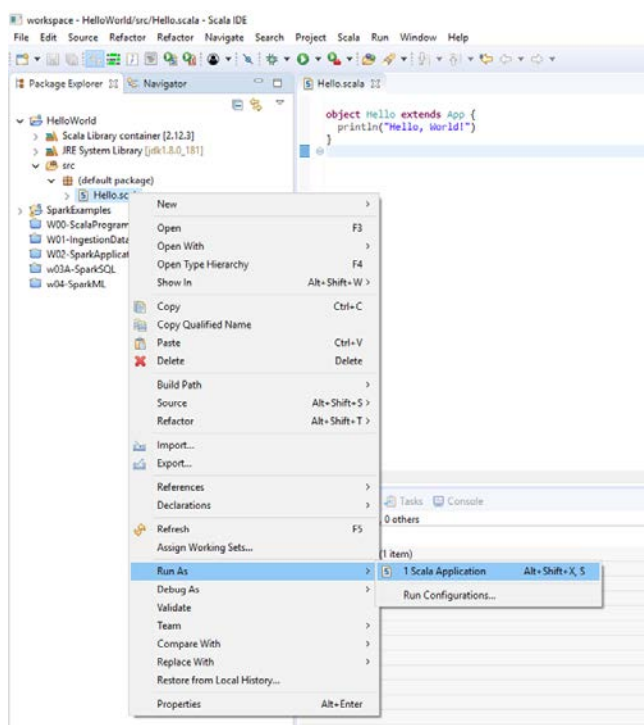
10. Name the project HelloWorld. Select the src folder and right click the context menu to pick New> Scala Object, type “Hello” and click Finish.



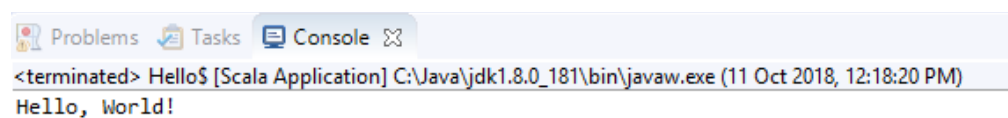
11. Writing code:.. Change the code to the following:

```
object Hello extends App {
  println("Hello, World!")
}
```

12. Running it: Right click on Hello object in your code and select Run > Scala Application.  
You're done!



13. Output:

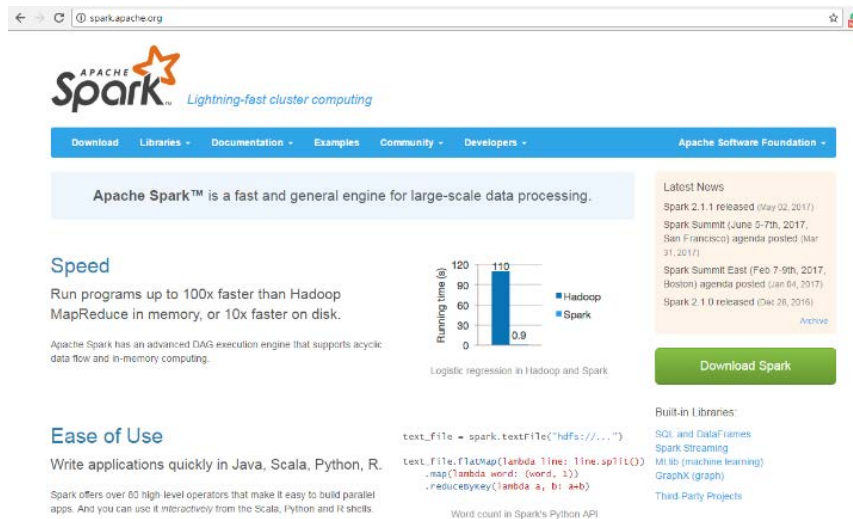




## Download Spark

As we are not going to use Hadoop it makes no difference the version you choose. Fortunately, the Apache website makes available prebuilt versions of Spark that will just run out of the box that are precompiled for the latest Hadoop version. You don't have to build anything, you can just download that to your computer and stick it in the right place and be good to go for the most part.

1. Let's get back to a new browser tab here, head to [spark.apache.org](https://spark.apache.org), and click on the Download Spark button (or <https://spark.apache.org/downloads.html>):



2. Now, we have used Spark 2.3.2 here, but anything beyond 2.0 should work just fine.

## Download Apache Spark™

1. Choose a Spark release: **2.3.2 (Sep 24 2018)**
2. Choose a package type: **Pre-built for Apache Hadoop 2.7 and later**
3. Download Spark: [spark-2.3.2-bin-hadoop2.7.tgz](#)
4. Verify this release using the [2.3.2 signatures and checksums](#) and [project release KEYS](#).

*Note: Starting version 2.0, Spark is built with Scala 2.11 by default. Scala 2.10 users should download the Spark source package and build with Scala 2.10 support.*

Make sure you get a prebuilt version, and select the direct download option so all these defaults are perfectly fine.

3. Now, it downloads a TGZ (Tar in GZip) file. You can use WinRAR to unzip the files. Extract the files to any location in your drive with enough permissions for your user.

## WinUtils

The official release of Hadoop does not include the required binaries (e.g., winutils.exe) necessary to run Apache Hadoop. In order to use Hadoop on Windows, it must be compiled from source. So we must get the 64 bit winutils.exe from a trusted store. I used from [here](#)-feel free to pick.



## Environment Variables

Every process has an environment block that contains a set of environment variables and their values. There are two types of environment variables: user environment variables (set for each user) and system environment variables (set for everyone).

The command processor provides the **set** command to display its environment block or to create new environment variables. You can also view or modify the environment variables by selecting **System** from the **Control Panel**, selecting **Advanced system settings**, and clicking **Environment Variables**. Each environment block contains the environment variables in the following format:

```
Var1=Value1\0
Var2=Value2\0
Var3=Value3\0
...
VarN=ValueN\0\0
```

To set environment variables in Windows 10 and Windows 8:

1. In Search, search for and then select: System (Control Panel)
2. Click the Advanced system settings link.
3. Click Environment Variables. In the section System Variables, find the PATH environment variable and select it. Click Edit. If the PATH environment variable does not exist, click New.
4. In the Edit System Variable (or New System Variable) window, specify the value of the PATH environment variable. Click OK. Close all remaining windows by clicking OK.
5. Reopen Command prompt window, and run your code.
  - **\_JAVA\_OPTION:** We set this variable to the value **-Xmx512M -Xms512M**. It helps with Java Heap Memory problems with the default values pre-set. You are free to increase the memory allocated.
  - **HADOOP\_HOME:** even when Spark can run without Hadoop, the version I downloaded is prebuilt and looks in the code for it. To fix this inconvenience this variable points to the folder containing the **winutils.exe** file (In my case **D:\winutils**)
  - **JAVA\_HOME:** we usually already set this variable when we install java but it is better to verify that exist and is correct. (In my case **C:\Java\jdk1.8.0\_181** – since I avoided the program files owing to the blank space character)
  - **SCALA\_HOME:** the bin folder of the Scala location. If you use the standard location from the installer should be the path **C:\Program Files (x86)\scala**.
  - **SPARK\_HOME:** the bin folder path of where you uncompressed Spark. In my case it is **D:\spark-2.3.2-bin-hadoop2.7**.

When you add an EXE path as an environment variable, you can access the program from any command line. The command line in Windows being the Command Prompt, you can open a Command Prompt in any location and run commands. Which paths you add is entirely up to you since you know which programs you need to access from the Command Prompt.

So after you have introduced all the above said environment variables, the last one to modify is PATH

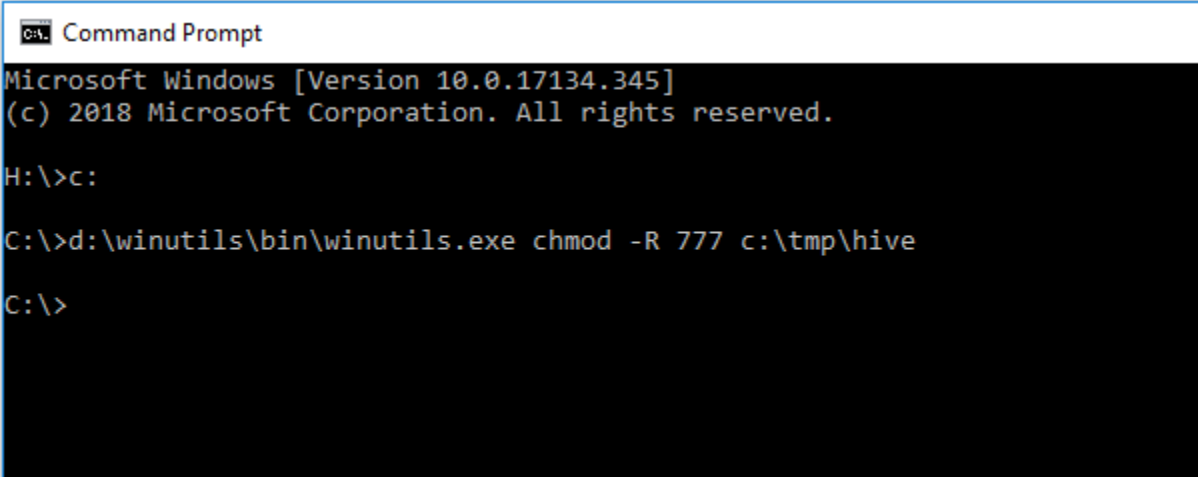
- **PATH:** We set this variable to include all the new variables set previously.

```
%JAVA_HOME%\bin  
%SCALA_HOME%\bin  
%HADOOP_HOME%\bin  
%SPARK_HOME%\bin
```

### Permissions for the folder tmp/hive

After we set everything, the shell tries to find the folder tmp/hive. So when you want to run the spark-shell "C:\tmp\hive" needs permission, I have to set the 777 permissions for it. In theory you can do it with the advanced sharing options of the sharing tab in the properties of the folder, but I did it in this way from the command line using winutils

Open a command prompt **as administrator** and type:



```
Command Prompt  
Microsoft Windows [Version 10.0.17134.345]  
(c) 2018 Microsoft Corporation. All rights reserved.  
H:\>c:  
C:\>d:\winutils\bin\winutils.exe chmod -R 777 c:\tmp\hive  
C:\>
```



```
SparkContextExample.scala
+ import org.apache.spark.SparkContext

object SparkContextExample {

  def main(args: Array[String]) {
    val stocksPath = "D:/data/stocks.txt"
    Logger.getLogger("org").setLevel(Level.ERROR)
    val sc = new SparkContext("local[*]", "Some random code")
    val data = sc.textFile(stocksPath, 2)
    val totalLines = data.count()
    println("Total number of Lines: %s".format(totalLines))
  }
}
```

Problems Tasks Console

<terminated> SparkContextExample\$ [Scala Application] C:\Java\jdk1.8.0\_181\bin\javaw.exe (11 Oct 2018, 1:59:43 PM)  
Total number of Lines: 57391

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



-----END OF DOCUMENT-----

