Electrical and Computer Engineering / Software Engineering

CPrE/SE 419: Software Tools for Large-Scale Data Analysis, Spring 2019

Purpose

The goal of this lab is to introduce you to Apache Spark, a fast and general engine for big data processing. Spark provides a basic data abstraction called RDD (Resilient Distributed Dataset) which is a collection of elements, stored in a distributed manner across a cluster. Using RDD transformations, Spark is well suited for data processing in pipelines. Spark also provides rich APIs for RDDs so that users can easily operate data in parallel. In addition, users can optionally persist RDDs in memory so that it will speed up computing when reuse the data.

During this lab, you will learn:

- The Spark platform and usage of its API's (in Java)
- Write program with pipelined jobs to analyze network logs

Submission

Create a single zip archive, named by your last name, with the following and hand it in through canvas:

- The output file for each task generated by your program.
- Commented Code for your program. Include all source files needed for compilation.

Examples

We have 2 examples "WordCount" and "StockPrice", that contain some Spark API usages. We have already seen "WordCount" in Hadoop and Pig, the problem is to count the number of occurrences for each distinct word. In Spark, we first use **flatMap()** method to split each line of text into words and each word is as one element in the RDD. Then we use **mapToPair()** method to transform RDD into PairRDD, that converts each element into <key, value> pair where key is the word and value is one. Finally, we use **reduceByKey()** method to sum all the ones to get the number of counts for each word. Note that the function in **reduceByKey()** method is applied in associative manner, like the Combiner in Hadoop. We also provide another example "StockPrice" in the lecture which analyze the stock prices.

For all the API's on RDD and PairRDD, check the links to their javadocs:

https://spark.apache.org/docs/1.6.0/api/java/org/apache/spark/api/java/JavaRDDLike.html

https://spark.apache.org/docs/1.6.0/api/java/org/apache/spark/api/java/JavaPairRDD.html

For more examples in Java:

https://github.com/apache/spark/tree/master/examples/src/main/java/org/apache/spark/examples

Compile and Submit Application

To compile your Java program, you will need to link Spark libraries and also Hadoop libraries to include them in your class path. The **recommended option** to add the libraries to your program is using Maven. Add the following dependency for Spark library into the pom.xml file in your maven project:

```
<dependencies>


      <!-- https://mvnrepository.com/artifact/org.apache.spark/spark-core -->
      <dependency>
    <groupId>org.apache.spark</groupId>
    <artifactId>spark-core_2.11</artifactId>
    <version>1.6.0</version>
      </dependency>


<!-- https://mvnrepository.com/artifact/org.apache.hadoop/hadoop-common -->
      <dependency>
      <groupId>org.apache.hadoop</groupId>
      <artifactId>hadoop-common</artifactId>
      <version>2.6.0</version>
      <scope>provided</scope>
      </dependency>
      <dependency>
      <groupId>jdk.tools</groupId>
      <artifactId>jdk.tools</artifactId>
      <version>1.8.0_131</version>
      <scope>system</scope>
      <systemPath>C:/Program 
Files/Java/jdk1.8.0_201/lib/tools.jar</systemPath>
      </dependency>
      </dependencies>
      <properties>
      <maven.compiler.source>1.8</maven.compiler.source>
      <maven.compiler.target>1.8</maven.compiler.target>
      </properties>
```

Experiment 1 (40 points)

In this experiment we will modify the word count example, so that the output is sorted by the number of counts in descending order. We use the Gutenberg corpus, as the testing input to your program. It is on the HDFS at the following location: /cpre419/gutenberg. Include the source code and the snapshot of first 10 lines of your output file in your submission.

Hint: To achieve sorting, you can use the **sortByKey()** method. However, key is the word but we want to sort by the counts. You can use the **mapToPair()** method to swap the key and value. You can use Shakespeare corpus to test your program first because it is a small data.

Experiment 2 (60 points)

We will redo the firewall example in lab 5 on pig, except here we will write pipelined jobs in Spark.

Given two input files:

/cpre419/ip_trace – An IP trace file having information about connections received from different source IP addresses, along with a connection ID and time.

The format of IP trace file is:

<Time> <Connection ID> <Source IP> “>” <Destination IP> <protocol> <protocol dependent data>

/cpre419/raw_block - A file containing the connection IDs that were blocked

The format of block file is:

<Connection ID> <Action Taken>

Your task is to regenerate the log file by combining information from others logs that are available. The lost firewall log should contain details of all blocked connections and should be in the following format.

<Time> <Connection ID> <Source IP> <Destination IP> “Blocked”

A. (30pts) Regenerate the firewall file containing details of all blocked connections. You only need to submit your source code and the snapshot of first 10 lines your generated firewall log.

B. (30pts) Based on the previous program, generate a list of all unique source IP addresses that were blocked and the number of times that they were blocked. This list should be sorted (by the script) by the number of times that each IP was blocked in descending order. Submit your code and the snapshot of first 10 lines of your output file.

You can write part A and part B in the same program.

Installing Spark on Mac:

(If having any question, please contact Ashraf Tahmasbi <tahmasbi@iastate.edu>)

To successfully install spark on your system, you need to download and install the following:

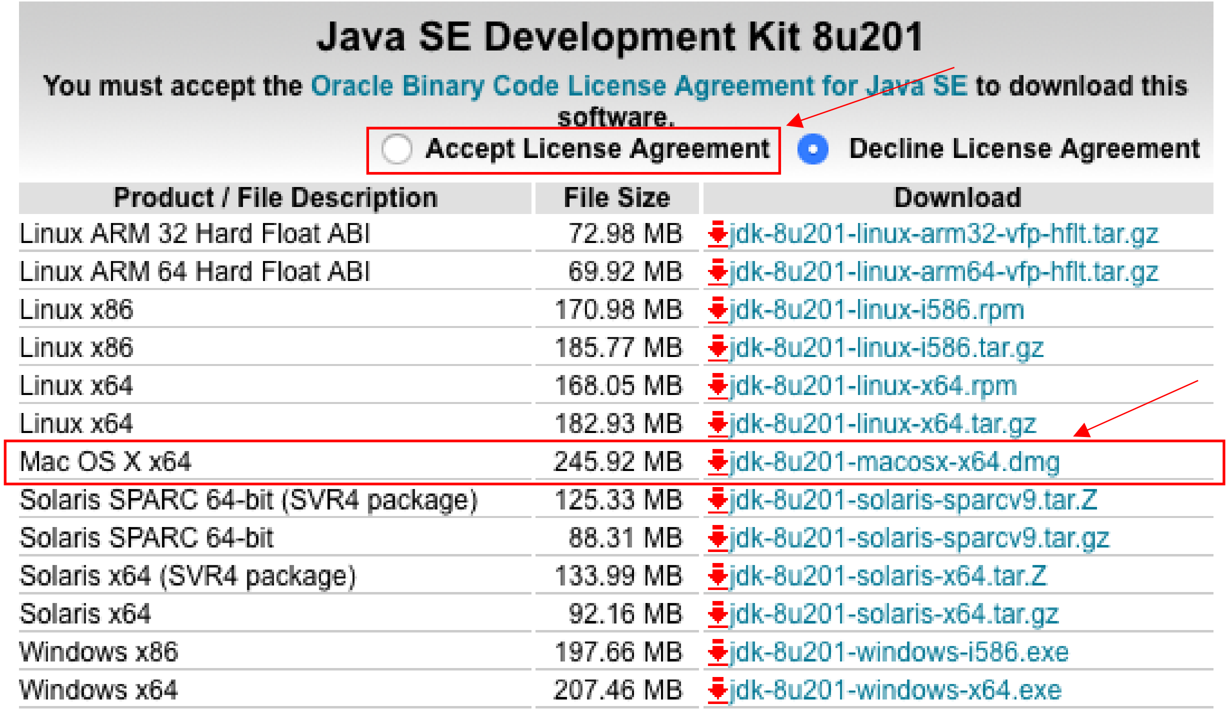
jdk-8u201-macosx-x64.

python-3.6.6-macosx10.9

sbt-1.2.8

scala-2.12.8

spark-2.4.0-bin-hadoop2.7



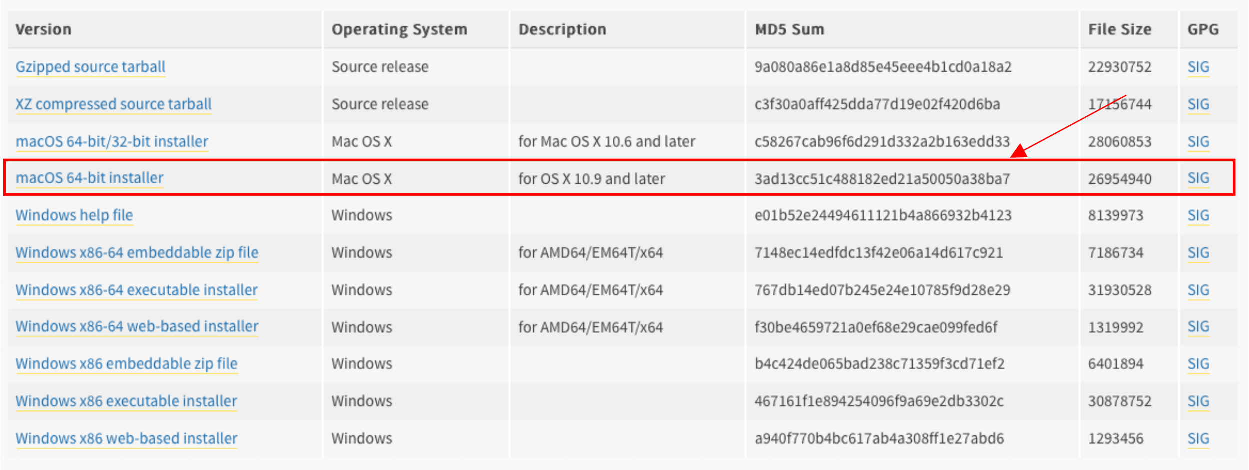
Java SE Development Kit 8u201

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

○ Accept License Agreement ● Decline License Agreement

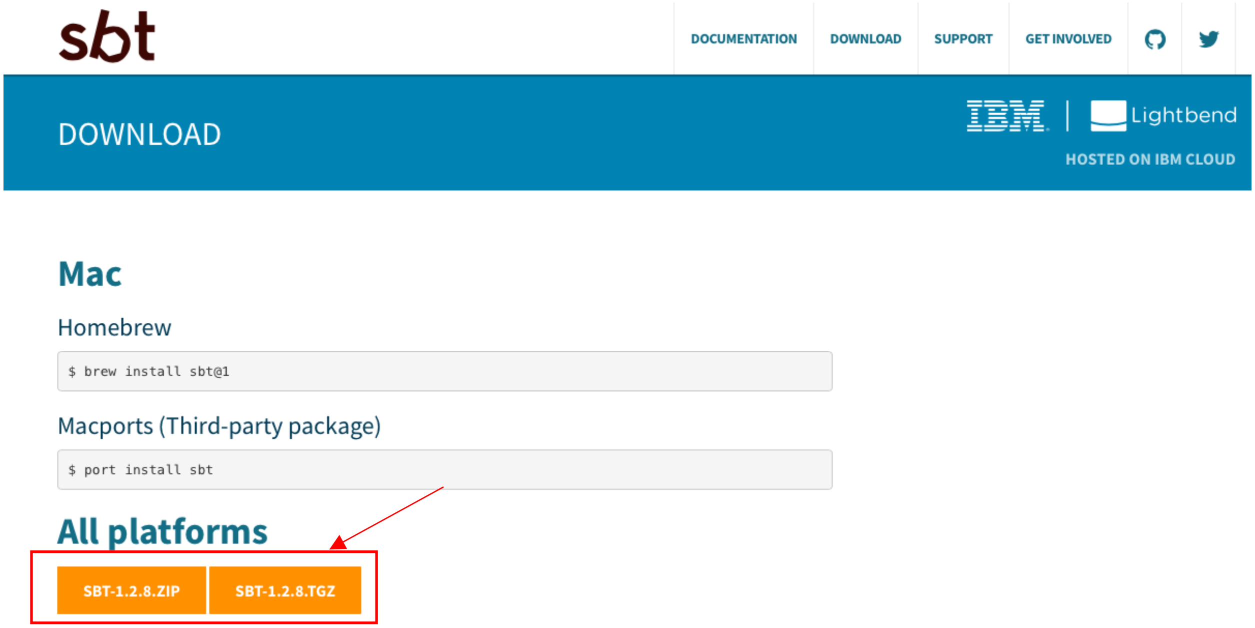
Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	72.98 MB	jdk-8u201-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	69.92 MB	jdk-8u201-linux-arm64-vfp-hflt.tar.gz
Linux x86	170.98 MB	jdk-8u201-linux-i586.rpm
Linux x86	185.77 MB	jdk-8u201-linux-i586.tar.gz
Linux x64	168.05 MB	jdk-8u201-linux-x64.rpm
Linux x64	182.93 MB	jdk-8u201-linux-x64.tar.gz
Mac OS X x64	245.92 MB	jdk-8u201-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	125.33 MB	jdk-8u201-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	88.31 MB	jdk-8u201-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	133.99 MB	jdk-8u201-solaris-x64.tar.Z
Solaris x64	92.16 MB	jdk-8u201-solaris-x64.tar.gz
Windows x86	197.66 MB	jdk-8u201-windows-i586.exe
Windows x64	207.46 MB	jdk-8u201-windows-x64.exe

Figure 1- Java SE Development kit 8u201's download page



Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		9a080a86e1a8d85e45eee4b1cd0a18a2	22930752	SIG
XZ compressed source tarball	Source release		c3f30a0aff425dda77d19e02f420d6ba	17156744	SIG
macOS 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.6 and later	c58267cab96f6d291d332a2b163edd33	28060853	SIG
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	3ad13cc51c488182ed21a50050a38ba7	26954940	SIG
Windows help file	Windows		e01b52e24494611121b4a866932b4123	8139973	SIG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	7148ec14edfdc13f42e06a14d617c921	7186734	SIG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	767db14ed07b245e24e10785f9d28e29	31930528	SIG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	f30be4659721a0ef68e29cae099fed6f	1319992	SIG
Windows x86 embeddable zip file	Windows		b4c424de065bad238c71359f3cd71ef2	6401894	SIG
Windows x86 executable installer	Windows		467161f1e894254096f9a69e2db3302c	30878752	SIG
Windows x86 web-based installer	Windows		a940f770b4bc617ab4a308ff1e27abd6	1293456	SIG

Figure 2- python-3.6.6's download page



DOWNLOAD

Mac

Homebrew

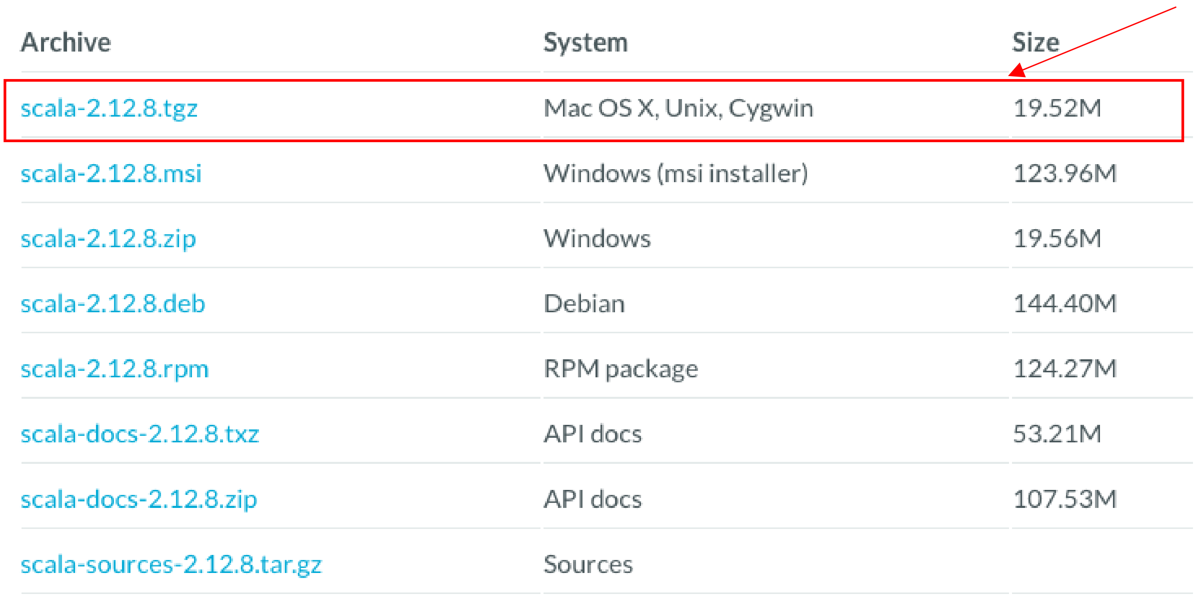
```
$ brew install sbt@1
```

Macports (Third-party package)

```
$ port install sbt
```

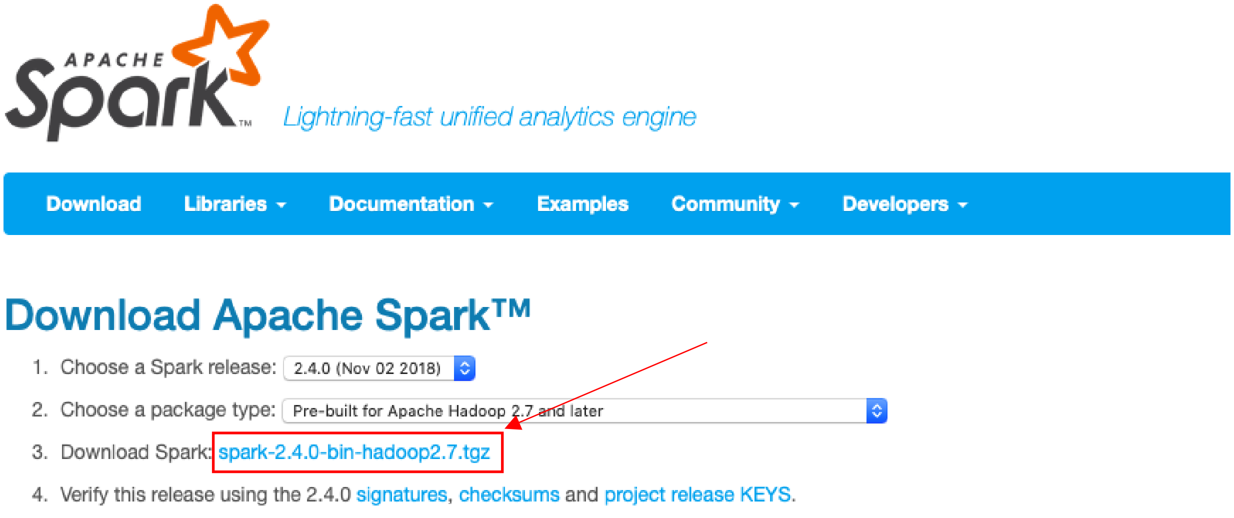
All platforms

Figure 3- sbt's download page



Archive	System	Size
scala-2.12.8.tgz	Mac OS X, Unix, Cygwin	19.52M
scala-2.12.8.msi	Windows (msi installer)	123.96M
scala-2.12.8.zip	Windows	19.56M
scala-2.12.8.deb	Debian	144.40M
scala-2.12.8.rpm	RPM package	124.27M
scala-docs-2.12.8.txz	API docs	53.21M
scala-docs-2.12.8.zip	API docs	107.53M
scala-sources-2.12.8.tar.gz	Sources	

Figure 4- scala's download page

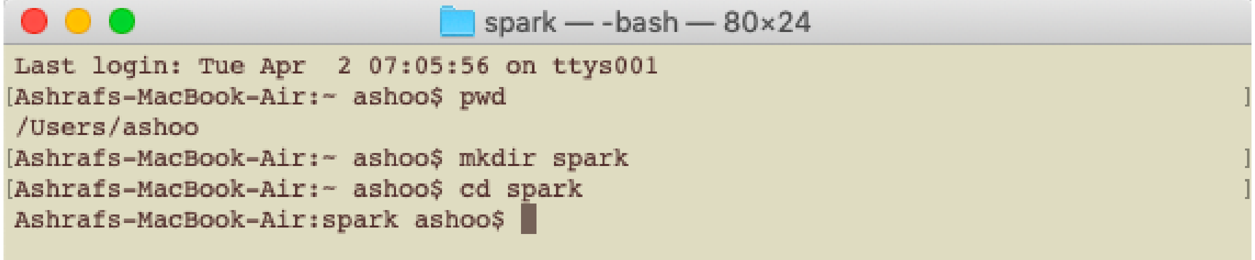


Download Apache Spark™

1. Choose a Spark release: 2.4.0 (Nov 02 2018)
2. Choose a package type: Pre-built for Apache Hadoop 2.7 and later
3. Download Spark: spark-2.4.0-bin-hadoop2.7.tgz
4. Verify this release using the 2.4.0 signatures, checksums and project release KEYS.

Figure 5- spark's download page

directory. After downloading all the required files, create a folder under your HOME directory and call it as “spark”.

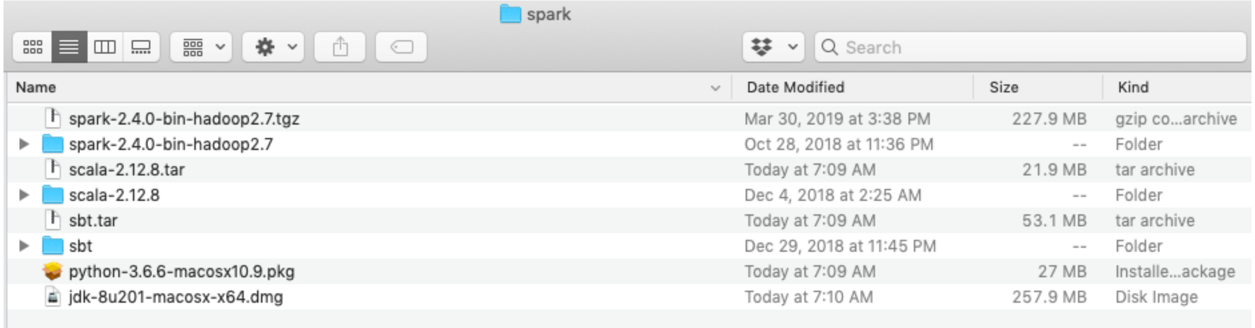


```
Last login: Tue Apr  2 07:05:56 on ttys001
[Ashrafs-MacBook-Air:~ ashoo$ pwd                                              ]
/Users/ashoo
[Ashrafs-MacBook-Air:~ ashoo$ mkdir spark                                      ]
[Ashrafs-MacBook-Air:~ ashoo$ cd spark                                         ]
Ashrafs-MacBook-Air:spark ashoo$ 
```

Figure 6- create a folder called "spark" under your HOME directory

Please note that my Home directory is /Users/ashoo and in the rest of this instruction I may refer to it as as $HOME or ~.

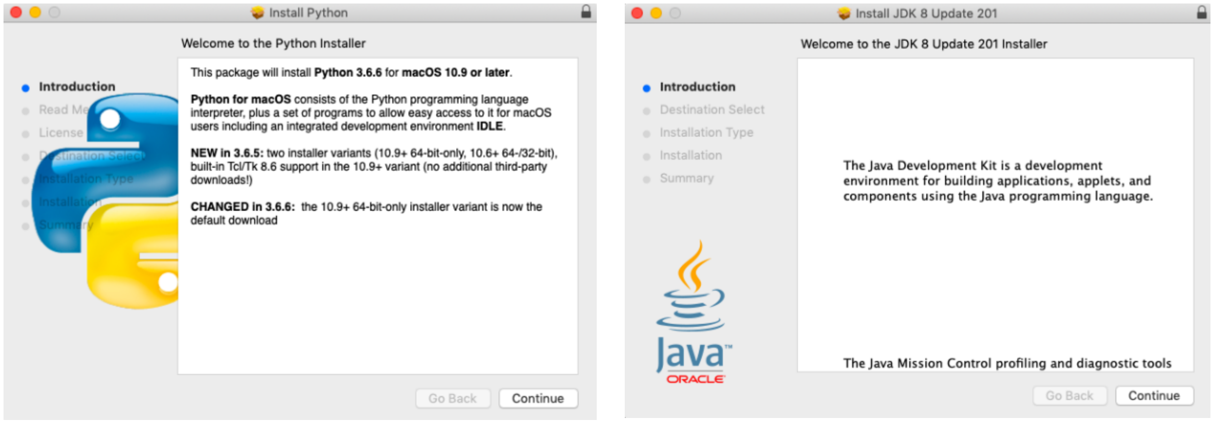
Now, move all the downloaded files into this and extract the compresses files.



Name	Date Modified	Size	Kind
spark-2.4.0-bin-hadoop2.7.tgz	Mar 30, 2019 at 3:38 PM	227.9 MB	gzip co...archive
▶ spark-2.4.0-bin-hadoop2.7	Oct 28, 2018 at 11:36 PM	--	Folder
scala-2.12.8.tar	Today at 7:09 AM	21.9 MB	tar archive
▶ scala-2.12.8	Dec 4, 2018 at 2:25 AM	--	Folder
sbt.tar	Today at 7:09 AM	53.1 MB	tar archive
▶ sbt	Dec 29, 2018 at 11:45 PM	--	Folder
python-3.6.6-macosx10.9.pkg	Today at 7:09 AM	27 MB	Installe...ackage
jdk-8u201-macosx-x64.dmg	Today at 7:10 AM	257.9 MB	Disk Image

Figure 7- spark's folder content

Now, install JDK and Python using the downloaded installer.



Welcome to the Python Installer

This package will install **Python 3.6.6** for **macOS 10.9 or later**.

Python for macOS consists of the Python programming language interpreter, plus a set of programs to allow easy access to it for macOS users including an integrated development environment **IDLE**.

NEW in 3.6.5: two installer variants (10.9+ 64-bit-only, 10.6+ 64-/32-bit), built-in Tcl/Tk 8.6 support in the 10.9+ variant (no additional third-party downloads!)

CHANGED in 3.6.6: the 10.9+ 64-bit-only installer variant is now the default download

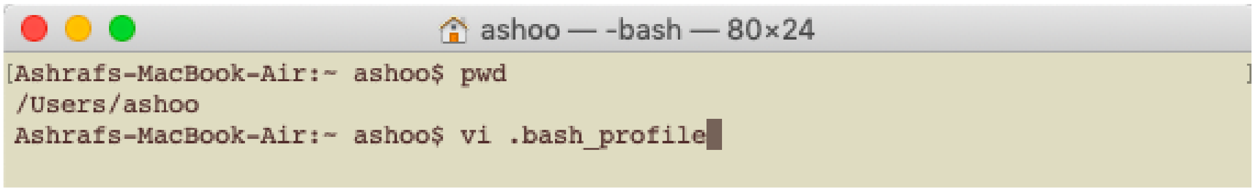
Welcome to the JDK 8 Update 201 Installer

The Java Development Kit is a development environment for building applications, applets, and components using the Java programming language.

The Java Mission Control profiling and diagnostic tools

Figure 8- jdk and python installers

When you are done with installing python and JDK, you need to setup shell environment by editing the ~/.bash_profile file. To this end, open the .bash_profile file, which is located at your Home directory using any text editor. If .bash_profile file doesn't exist, create a file named .bash_profile.

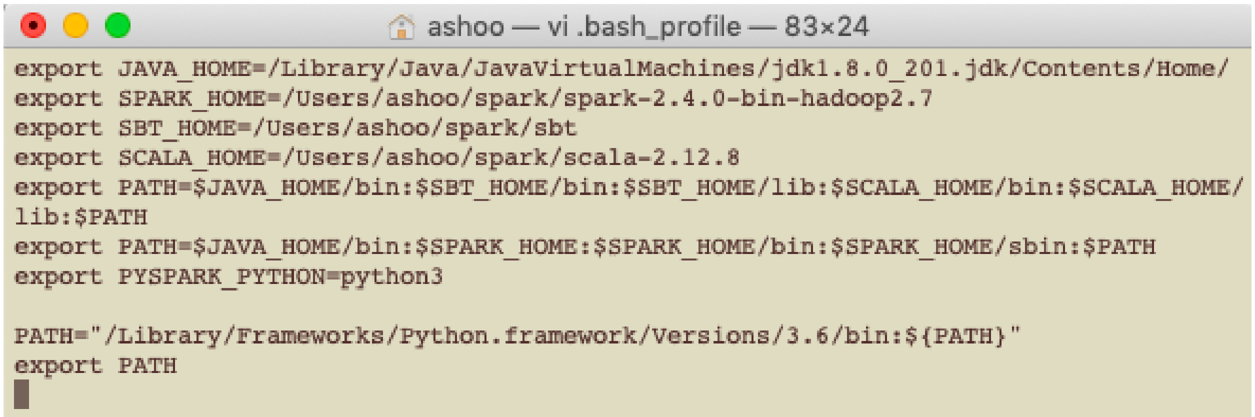


```
[Ashrafs-MacBook-Air:~ ashoo$ pwd
/Users/ashoo
Ashrafs-MacBook-Air:~ ashoo$ vi .bash_profile
```

Figure 9- Open .bash_profile to setup shell environment

Open this file and add the following lines to it:

```
export
JAVA_HOME=/Library/Java/JavaVirtualMachines/jdk1.8.0_201.jdk/Co
ntents/Home/
export SPARK_HOME=/Users/ashoo/spark/spark-2.4.0-bin-hadoop2.7
export SBT_HOME=/Users/ashoo/spark/sbt
export SCALA_HOME=/Users/ashoo/spark/scala-2.12.8
export
PATH=$JAVA_HOME/bin:$SBT_HOME/bin:$SBT_HOME/lib:$SCALA_HOME/bin
:$SCALA_HOME/lib:$PATH
export
PATH=$JAVA_HOME/bin:$SPARK_HOME:$SPARK_HOME/bin:$SPARK_HOME/sbi
n:$PATH
export PYSPARK_PYTHON=python3

PATH="/Library/Frameworks/Python.framework/Versions/3.6/bin:${P
ATH}"
export PATH
```

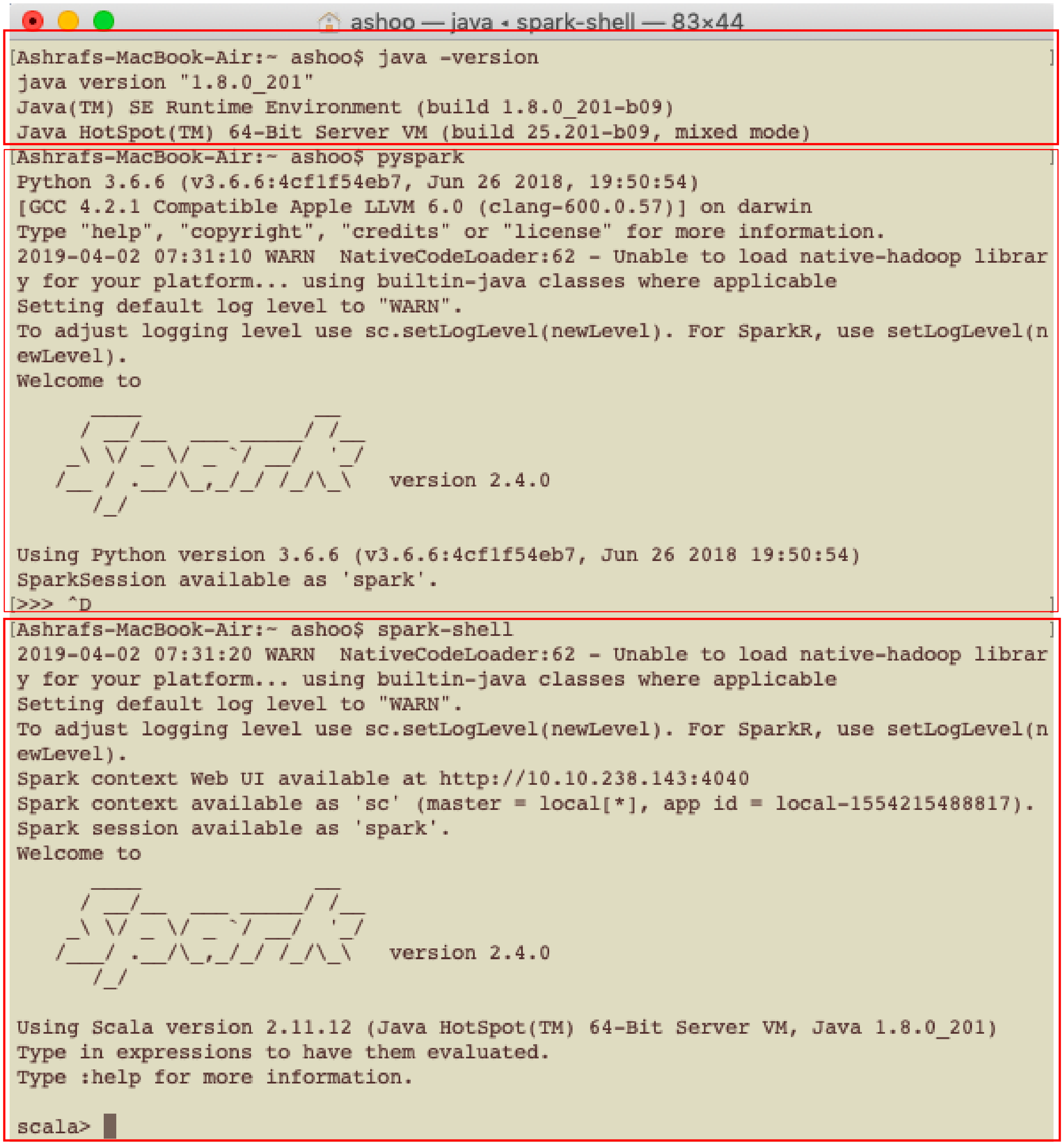


```
export JAVA_HOME=/Library/Java/JavaVirtualMachines/jdk1.8.0_201.jdk/Contents/Home/
export SPARK_HOME=/Users/ashoo/spark/spark-2.4.0-bin-hadoop2.7
export SBT_HOME=/Users/ashoo/spark/sbt
export SCALA_HOME=/Users/ashoo/spark/scala-2.12.8
export PATH=$JAVA_HOME/bin:$SBT_HOME/bin:$SBT_HOME/lib:$SCALA_HOME/bin:$SCALA_HOME/
lib:$PATH
export PATH=$JAVA_HOME/bin:$SPARK_HOME:$SPARK_HOME/bin:$SPARK_HOME/sbin:$PATH
export PYSPARK_PYTHON=python3

PATH="/Library/Frameworks/Python.framework/Versions/3.6/bin:${PATH}"
export PATH
```

Figure 10- .bash_profile file

After editing .bash_profile file, you have to reload it. To do so type source ~/.bash_profile in your terminal or quit and reopen the terminal program.

Now, the installation is complete. To test the installation do as follow:

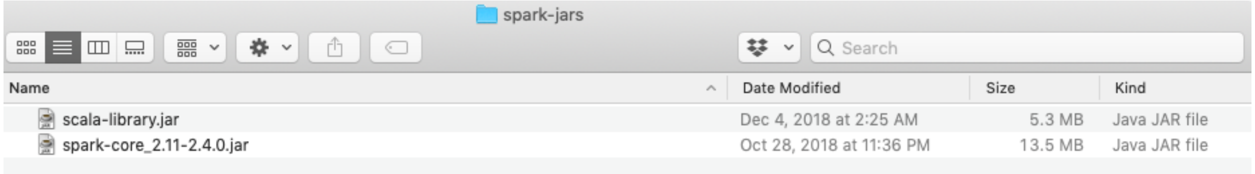


```
[Ashrafs-MacBook-Air:~ ashoo$ java -version
 java version "1.8.0_201"
 Java(TM) SE Runtime Environment (build 1.8.0_201-b09)
 Java HotSpot(TM) 64-Bit Server VM (build 25.201-b09, mixed mode)
[Ashrafs-MacBook-Air:~ ashoo$ pyspark
 Python 3.6.6 (v3.6.6:4cf1f54eb7, Jun 26 2018, 19:50:54)
 [GCC 4.2.1 Compatible Apple LLVM 6.0 (clang-600.0.57)] on darwin
 Type "help", "copyright", "credits" or "license" for more information.
 2019-04-02 07:31:10 WARN  NativeCodeLoader:62 - Unable to load native-hadoop librar
 y for your platform... using builtin-java classes where applicable
 Setting default log level to "WARN".
 To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(n
 ewLevel).
 Welcome to
       ____              __
      / __/__  ___ _____/ /__
     _\ \/ _ \/ _ `/ __/  '_/
    /__ / .__/\_,_/_/ /_/\_\   version 2.4.0
       /_/

 Using Python version 3.6.6 (v3.6.6:4cf1f54eb7, Jun 26 2018 19:50:54)
 SparkSession available as 'spark'.
[>>> ^D
[Ashrafs-MacBook-Air:~ ashoo$ spark-shell
 2019-04-02 07:31:20 WARN  NativeCodeLoader:62 - Unable to load native-hadoop librar
 y for your platform... using builtin-java classes where applicable
 Setting default log level to "WARN".
 To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(n
 ewLevel).
 Spark context Web UI available at http://10.10.238.143:4040
 Spark context available as 'sc' (master = local[*], app id = local-1554215488817).
 Spark session available as 'spark'.
 Welcome to
       ____              __
      / __/__  ___ _____/ /__
     _\ \/ _ \/ _ `/ __/  '_/
    /___/ .__/\_,_/_/ /_/\_\   version 2.4.0
       /_/

 Using Scala version 2.11.12 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_201)
 Type in expressions to have them evaluated.
 Type :help for more information.

 scala> 
```

Figure 11- check installation was successful

Note, to exit spark-shell or pyspark you can use CTRL-D.

Note1: If you are not using maven project you need to include the following jar files into your project.

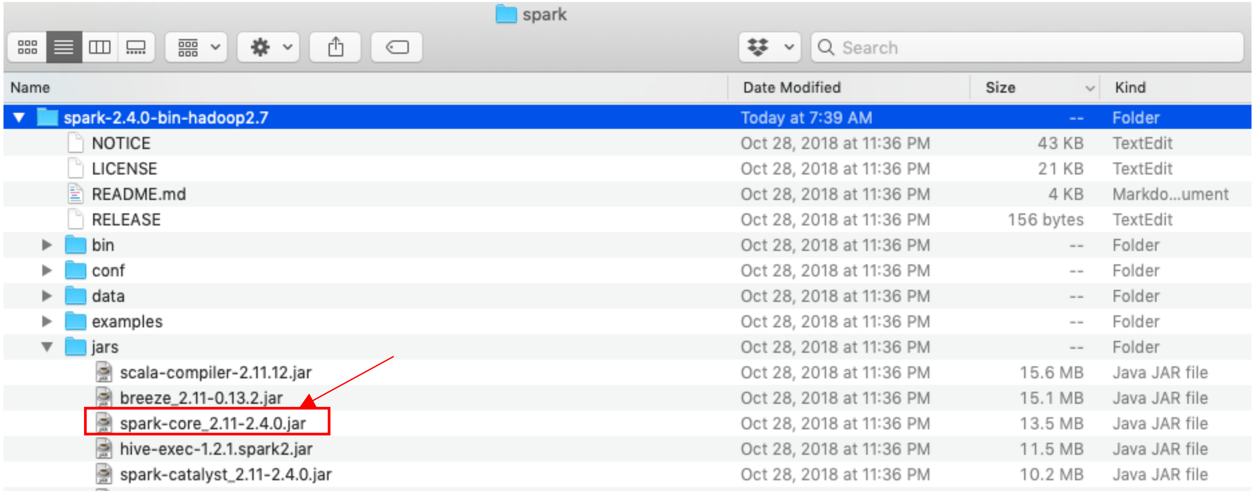


Name	Date Modified	Size	Kind
scala-library.jar	Dec 4, 2018 at 2:25 AM	5.3 MB	Java JAR file
spark-core_2.11-2.4.0.jar	Oct 28, 2018 at 11:36 PM	13.5 MB	Java JAR file

Figure 12- external jar files to include in your project

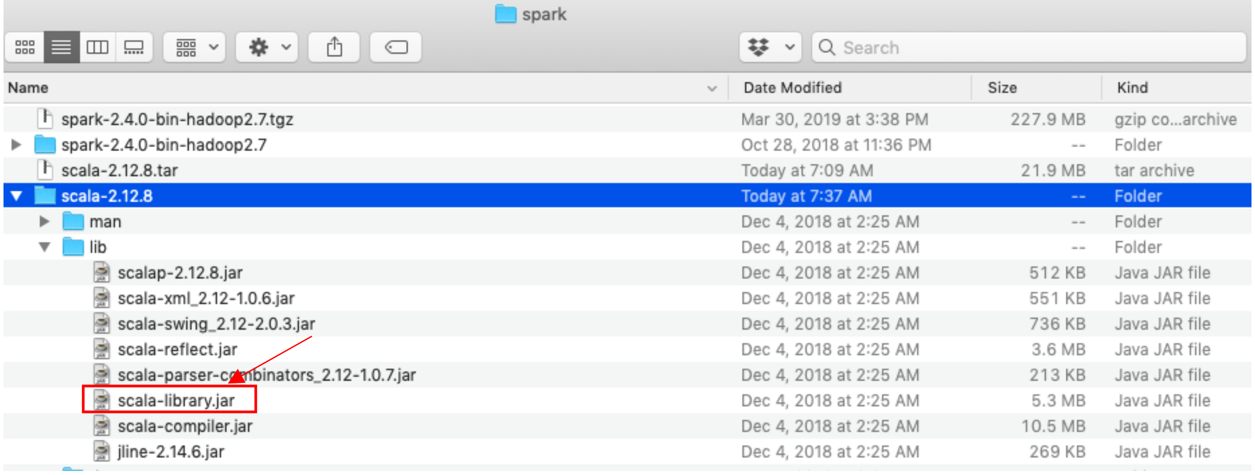
You can find these jar files through the following paths:

Spark → scala-2.12.8 → lib and Spark → spark-2.4.0-bin-hadoop2.7 → jars



Name	Date Modified	Size	Kind
spark-2.4.0-bin-hadoop2.7	Today at 7:39 AM	--	Folder
NOTICE	Oct 28, 2018 at 11:36 PM	43 KB	TextEdit
LICENSE	Oct 28, 2018 at 11:36 PM	21 KB	TextEdit
README.md	Oct 28, 2018 at 11:36 PM	4 KB	Markdo...ument
RELEASE	Oct 28, 2018 at 11:36 PM	156 bytes	TextEdit
bin	Oct 28, 2018 at 11:36 PM	--	Folder
conf	Oct 28, 2018 at 11:36 PM	--	Folder
data	Oct 28, 2018 at 11:36 PM	--	Folder
examples	Oct 28, 2018 at 11:36 PM	--	Folder
jars	Oct 28, 2018 at 11:36 PM	--	Folder
scala-compiler-2.11.12.jar	Oct 28, 2018 at 11:36 PM	15.6 MB	Java JAR file
breeze_2.11-0.13.2.jar	Oct 28, 2018 at 11:36 PM	15.1 MB	Java JAR file
spark-core_2.11-2.4.0.jar	Oct 28, 2018 at 11:36 PM	13.5 MB	Java JAR file
hive-exec-1.2.1.spark2.jar	Oct 28, 2018 at 11:36 PM	11.5 MB	Java JAR file
spark-catalyst_2.11-2.4.0.jar	Oct 28, 2018 at 11:36 PM	10.2 MB	Java JAR file

Figure 13- spark-core-2.11-2-4.0.jar file



Name	Date Modified	Size	Kind
spark-2.4.0-bin-hadoop2.7.tgz	Mar 30, 2019 at 3:38 PM	227.9 MB	gzip co...archive
spark-2.4.0-bin-hadoop2.7	Oct 28, 2018 at 11:36 PM	--	Folder
scala-2.12.8.tar	Today at 7:09 AM	21.9 MB	tar archive
scala-2.12.8	Today at 7:37 AM	--	Folder
man	Dec 4, 2018 at 2:25 AM	--	Folder
lib	Dec 4, 2018 at 2:25 AM	--	Folder
scalap-2.12.8.jar	Dec 4, 2018 at 2:25 AM	512 KB	Java JAR file
scala-xml_2.12-1.0.6.jar	Dec 4, 2018 at 2:25 AM	551 KB	Java JAR file
scala-swing_2.12-2.0.3.jar	Dec 4, 2018 at 2:25 AM	736 KB	Java JAR file
scala-reflect.jar	Dec 4, 2018 at 2:25 AM	3.6 MB	Java JAR file
scala-parser-combinators_2.12-1.0.7.jar	Dec 4, 2018 at 2:25 AM	213 KB	Java JAR file
scala-library.jar	Dec 4, 2018 at 2:25 AM	5.3 MB	Java JAR file
scala-compiler.jar	Dec 4, 2018 at 2:25 AM	10.5 MB	Java JAR file
jline-2.14.6.jar	Dec 4, 2018 at 2:25 AM	269 KB	Java JAR file

Figure 14- scala-library.jar file

Note2: After writing your code and creating your jar file, you can run it using the following command:

```
spark-submit --class <Class Name> <jar file> <required
arguments>
```

Installing Spark on Windows:

(If having any question, please contact Xu Teng <xuteng@iastate.edu>)

1. Install **Java JDK 1.8.x** for Windows
   Download link: https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
   Note: Remember the Path you install the Java (let’s call it ***install_JAVA_PATH***)
2. Configure Java
   a. Open ***Control Panel*** -> click ***System and Security*** -> click ***System*** -> click ***Advanced system settings*** -> click ***Environment Variables*** under ***Advanced***
   b. Click ***New…*** under System variables. Add Variable name with ***JAVA_HOME*** and Variable value with ***install_JAVA_PATH***.
   c. Select ***Path*** variable under System variables, click ***Edit…***
   d. Click ***New*** and enter %JAVA_HOME%\bin
3. Configure winutils.exe
   a. Download from link: https://github.com/steveloughran/winutils/tree/master/hadoop-2.7.1/bin. You can find winutils.exe under this repository.
   b. Create a new folder under C: drive, named winutils. Then create another new folder, named bin, under C:\winutils. And copy winutils.exe to C:\winutils\bin.
   c. Add new system variable (same as 2.a and 2.b above) whose name is ***HADOOP_HOME*** and value is ***C:\winutils***
4. Configure Spark
   a. Download from link: https://spark.apache.org/downloads.html. Choose Spark release 2.3.3 and Pre-built for Apache Hadoop 2.7 and later. Then click Download Spark, and select one mirror site for download.
   b. Unzip download file and find folder ***spark-2.3.3-bin-hadoop2.7***.
   c. Copy ***spark-2.3.3-bin-hadoop2.7*** to C: drive
   d. Add new system variable (same as 2.a and 2.b above) whose name is ***SPARK_HOME*** and value is ***C:\spark-2.3.3-bin-hadoop2.7***
   e. Select ***Path*** variable under System variables, click ***Edit…***
   f. Click ***New*** and enter %SPARK_HOME%\bin

Eventually, Spark has been installed and configured on our local. Open your terminal and cd into the directory you copied Spark files to (in our case, ***C:\spark-2.3.3-bin-hadoop2.7***). Then type ***spark-shell***.