



**Faculty of Information Technology
University of Moratuwa
BSc Hons in Information Technology
BSc Hons in Information Technology Management
IN 4410 – Big Data Analytics**

Level 1 - Semester 2

Spark Installation and Introduction

Apache Spark Installation

Apache Spark is an open-source framework that processes large volumes of stream data from multiple sources. Spark is used in distributed computing with machine learning applications, data analytics, and graph-parallel processing.

This guide will show you how to install Apache Spark on Windows 10

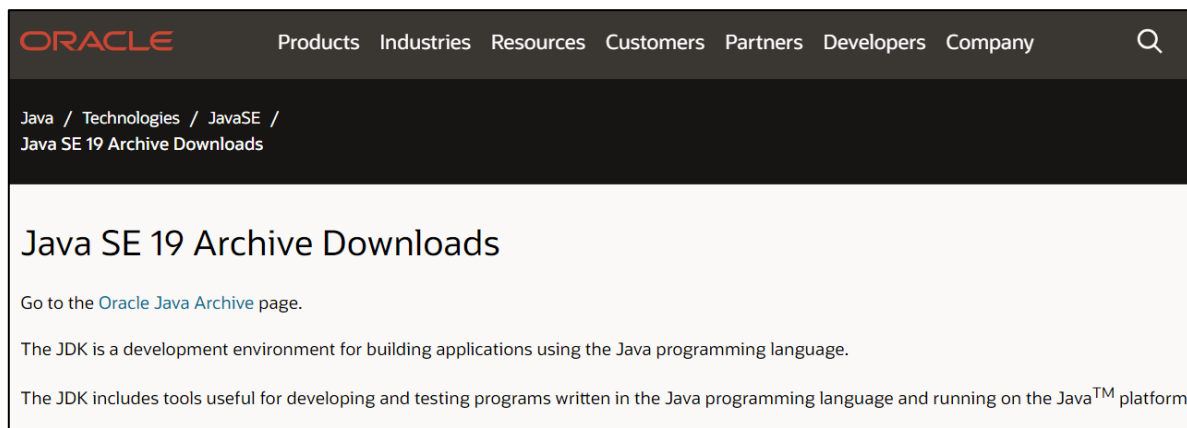
Let's install Apache Spark on your PC!

Pre-requisites

Before we begin install the below must have in your computer
Java, Spark, Winutils

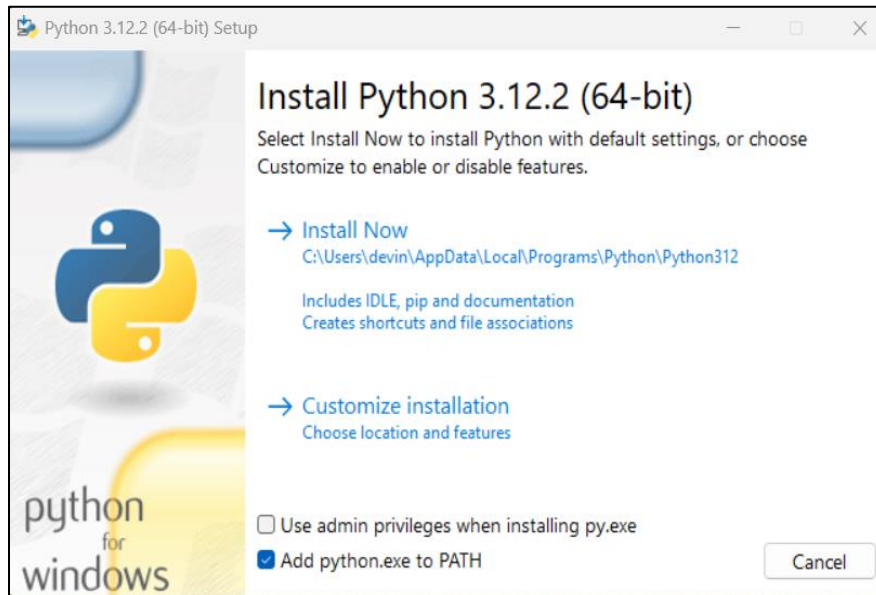
1) Install Java (19)

<https://www.oracle.com/java/technologies/javase/jdk19-archive-downloads.html>



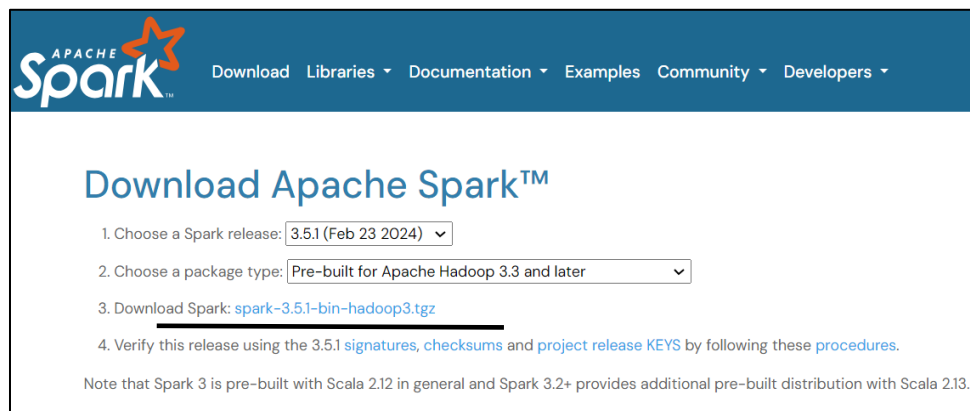
2) Python 3.12.2.

<https://www.python.org/downloads/>



3) Download Spark

<https://spark.apache.org/downloads.html>



1) Click on .tgz file.

2) Click on the first link on the page. Then you can download the file.



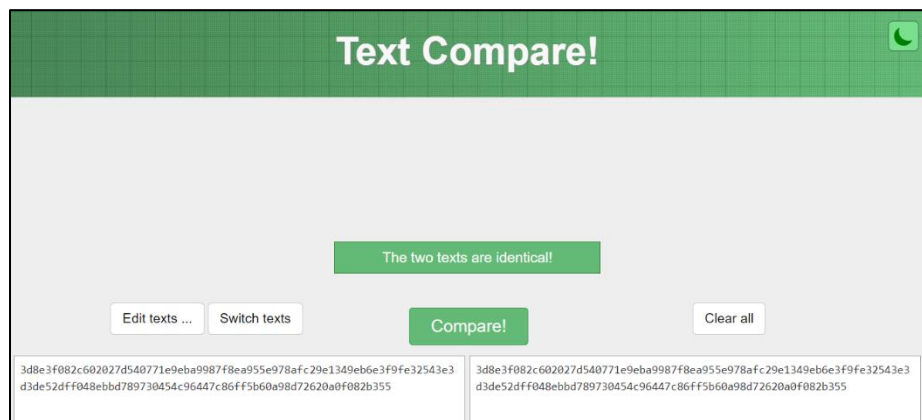
- 3) Create a folder C:/Spark/ and paste the content of the file in it.
- 4) Then **Extract tar.gz to tar file** again **extract tar file to hadoop** file.
- 5) And rename the file as spark-3.5.1

- 6) Check the checksum in command prompt: -
certutil -hashfile C:\Spark\spark-3.5.1-bin-hadoop3.tgz SHA512

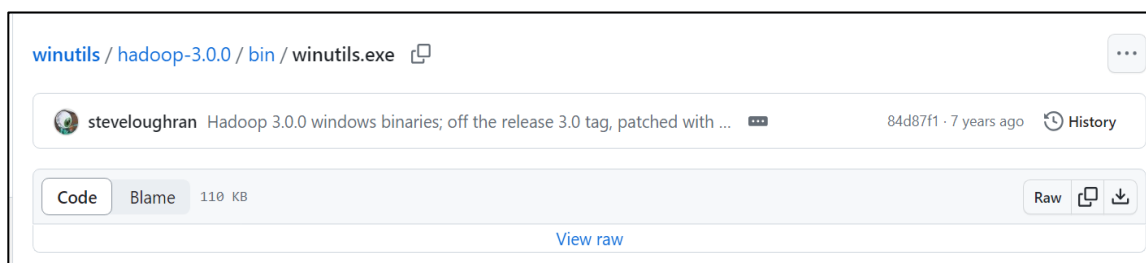
```
C:\Windows\System32>certutil -hashfile C:\Bigdata\spark-3.5.1-bin-hadoop3.tgz SHA512
SHA512 hash of C:\Bigdata\spark-3.5.1-bin-hadoop3.tgz:
3d8e3f082c602027d540771e9eba9987f8ea955e978afc29e1349eb6e3f9fe32543e3d3de52dff048ebbd789730454c96447c86ff5b60a9
8d72620a0f082b355
CertUtil: -hashfile command completed successfully.

C:\Windows\System32>
```

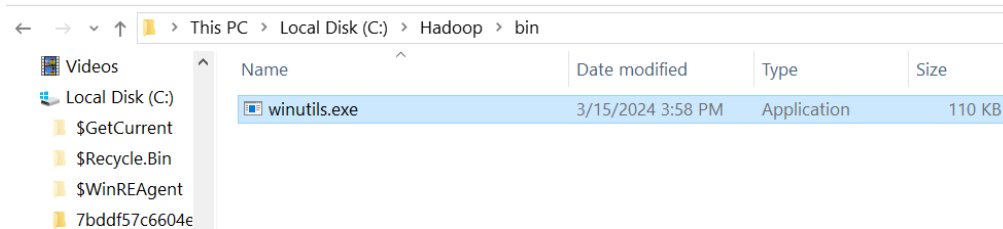
- 7) Compare it with the checksum link in the download link and compare it with a text diff tool.



- 8) Download the winutils.exe file for hadoop3.0.0 from <https://github.com/steveloughran/winutils> and copy it inside %SPARK_HOME%\bin (SPARK_HOME: C:/Spark/bin>



9) Create a new directory C:\Hadoop\bin and paste into it as well.

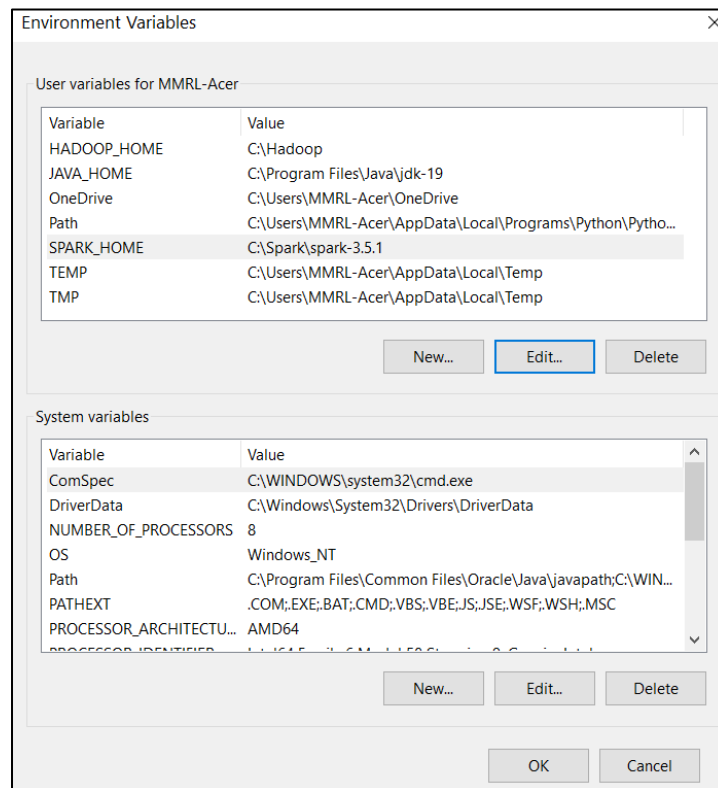


10) Now you can setup the environment variables. Click on “Start” (Windows icon on the taskbar) and type “environment variables”. Now in the opened “System properties” window, click on the “Environment Variables” button.

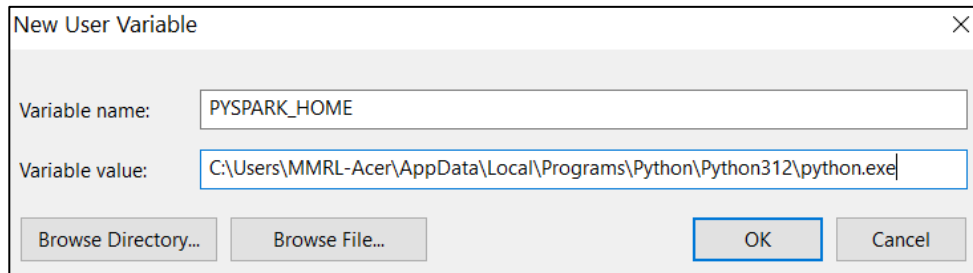
- Click on “New” under user variables.

11) Now add JAVA_HOME, SPARK_HOME and HADOOP_HOME to env variables.

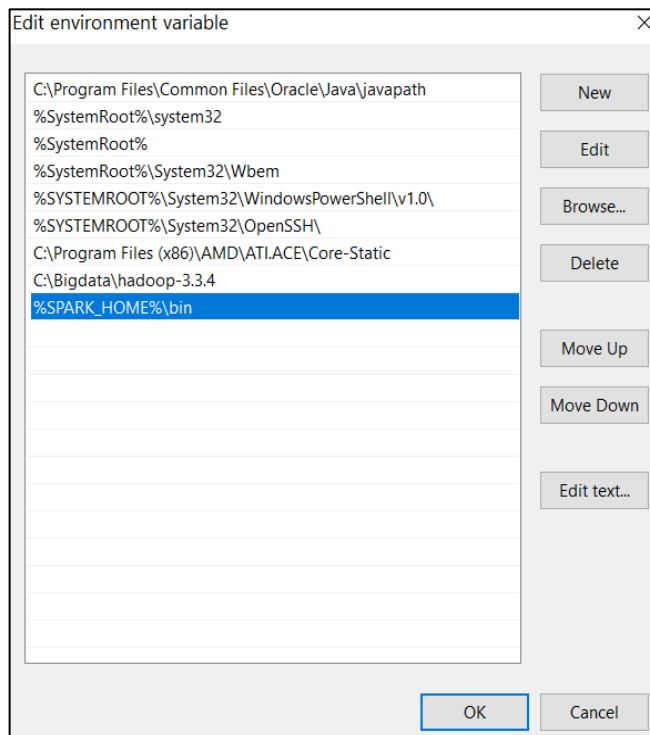
- JAVA_HOME - C:\Program Files\Java\jdk-19
- SPARK_HOME - C:\Spark\spark-3.5.1
- HADOOP_HOME - C:\Hadoop



- Enter “PYSPARK_HOME” as the variable name and provide the path to your Hadoop directory. (Eg: - If the path to the folder is C:\Bigdata\Spark\spark-3.5.1, then your variable value should be
- C:\Users\MMRL-Acer\AppData\Local\Programs\Python\Python312\python.exe). Click OK.



- 12) Then click on the “Path” variable under the system variables and click “Edit”.
- Then click **New** put the spark home path inside the environment variable list and then click ok.



Also add these paths

- %JAVA_HOME%\bin
- %HADOOP_HOME%\bin

14) Also Verify the installation by checking the version of Java and Python:

The output should print **Python 3.9.1**.

The output should print **java version "19.0.2" 2023-01-17**

- 1) Open a new command-prompt window using the right-click and **Run as administrator**: To start Spark, enter:

```
Administrator: Command Prompt - spark-shell
```

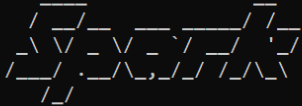
(c) Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>spark-shell

24/03/15 16:12:14 WARN Shell: Did not find winutils.exe: java.io.FileNotFoundException: Could not locate Hadoop executable: C:\Bigdata\hadoop-3.3.4\bin\winutils.exe -see <https://wiki.apache.org/hadoop/WindowsProblems>

Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).

Spark context Web UI available at http://host.docker.internal:4040
Spark context available as 'sc' (master = local[*], app id = local-1710499351670).
Spark session available as 'spark'.
Welcome to

 version 3.5.1

Using Scala version 2.12.18 (Java HotSpot(TM) 64-Bit Server VM, Java 19.0.2)
Type in expressions to have them evaluated.
Type :help for more information.

scala>

To exit Spark and close the Scala shell, press **ctrl-d** in the command-prompt window.

You can make sure whether the spark is working with python within this command

```
pyspark
```

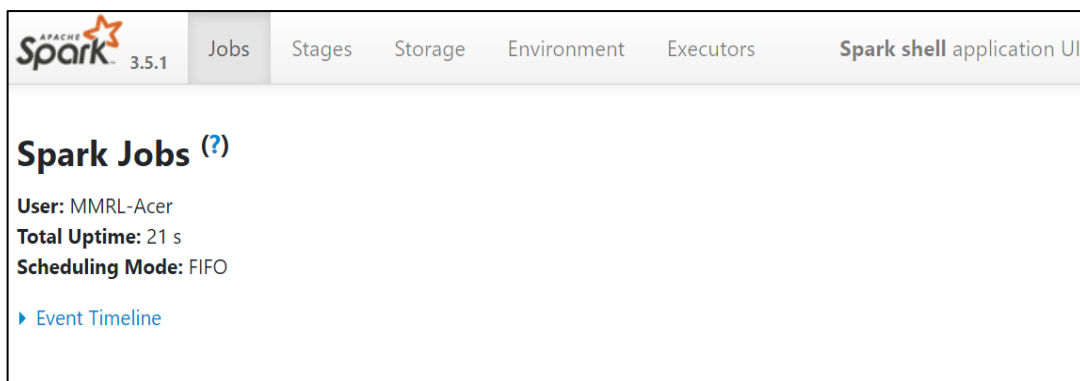
```
C:\WINDOWS\system32>pyspark
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Welcome to

  ____      _
 / ___|  _ \| | | |
 \___ \| |_) | |_| |
  ___) | |_) | | | |
 |____|_|_|\___|_|_|_|

version 3.5.1

Using Python version 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024 21:26:36)
Spark context Web UI available at http://host.docker.internal:4040
Spark context available as 'sc' (master = local[*], app id = local-1710734049187).
SparkSession available as 'spark'.
>>> print(spark.version)
3.5.1
>>>
```

- 2) Open a web browser and navigate to <http://localhost:4040/>. You can replace **localhost** with the name of your system. You should see an Apache Spark shell Web UI. The example below shows the Executors page



Congratulations!

Spark Introduction

Test Spark

Resilient Distributed Datasets

Resilient Distributed Datasets (**RDDs**) are the primary data structure in Spark. RDDs are reliable and memory-efficient when it comes to parallel processing. By storing and processing data in RDDs, Spark speeds up MapReduce processes.

In this example, we will launch the Spark shell and use Scala to read the contents of a file. You can use an existing file, such as the *README* file in the Spark directory, or you can create your own. We created **test** with some text.

1. Open a command-prompt window and navigate to the folder with the file you want to use and launch the Spark shell.
2. Change the directory to spark shell

```
val x = sc
```

```
scala> val x = sc
x: org.apache.spark.SparkContext = org.apache.spark.SparkContext@7689b31
scala>
```

1. First, state a variable to use in the Spark context with the name of the file. Remember to add the file extension if there is any.

```
val x=sc.textFile("C:/Bigdata/Spark/spark-3.5.1/README.md")
```

```
scala> val x=sc.textFile("C:/Bigdata/Spark/spark-3.5.1/README.md")
x: org.apache.spark.rdd.RDD[String] = C:/Bigdata/Spark/spark-3.5.1/README.md MapPartitionsRDD[1] at
  textFile at <console>:23
scala>
```


2. The output shows an RDD is created. Then, we can view the file contents by using this command to call an action:

```
x.take(11).foreach(println)
```

```
scala> x.take(11).foreach(println)
# Apache Spark

Spark is a unified analytics engine for large-scale data processing. It provides
high-level APIs in Scala, Java, Python, and R, and an optimized engine that
supports general computation graphs for data analysis. It also supports a
rich set of higher-level tools including Spark SQL for SQL and DataFrames,
MLlib for machine learning, GraphX for graph processing,
and Structured Streaming for stream processing.

<https://spark.apache.org/>

scala>
```

This command instructs Spark to print 11 lines from the file you specified.

3. To perform an action on this file (**value x**), add another value **y**, and do a map transformation.

For example, you can print the characters in uppercase mode with this command:

```
val y = x.map(_.toUpperCase)
```

```
scala> val y = x.map(_.toUpperCase)
y: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[2] at map at <console>:23
scala>
```

The system creates a child RDD in relation to the first one. Then, specify how many lines you want to print from the value **y**:

```
y.take(11).foreach(println)
```

```
scala> y.take(11).foreach(println)
# APACHE SPARK

SPARK IS A UNIFIED ANALYTICS ENGINE FOR LARGE-SCALE DATA PROCESSING. IT PROVIDES
HIGH-LEVEL APIS IN SCALA, JAVA, PYTHON, AND R, AND AN OPTIMIZED ENGINE THAT
SUPPORTS GENERAL COMPUTATION GRAPHS FOR DATA ANALYSIS. IT ALSO SUPPORTS A
RICH SET OF HIGHER-LEVEL TOOLS INCLUDING SPARK SQL FOR SQL AND DATAFRAMES,
PANDAS API ON SPARK FOR PANDAS WORKLOADS, MLLIB FOR MACHINE LEARNING, GRAPHX FOR GRAPH PROCESSING,
AND STRUCTURED STREAMING FOR STREAM PROCESSING.

<HTTPS://SPARK.APACHE.ORG/>

scala>
```

The output prints 11 lines of the README.md file in the uppercase mode.

Test RDD and a Data Frame

In this example, we can create RDD and Data frame

1. We can create RDD in 3 ways, we will use one way to create RDD.

Define any list then parallelize it. It will create RDD. Below is code and copy paste it one by one on the command line.

```
val list = Array(1,2,3,4,5)
val rdd = sc.parallelize(list)
```

Above will create RDD.

```
scala> val list = Array(1,2,3,4,5)
list: Array[Int] = Array(1, 2, 3, 4, 5)

scala> val rdd = sc.parallelize(list)
rdd: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[3] at parallelize at <console>:24

scala>
```

2. Now we will create a Data frame from RDD. Follow the below steps to create Data frame.

```
import spark.implicits._  
  
val df = rdd.toDF("id")
```

```
scala> import spark.implicits._  
import spark.implicits._  
  
scala> val df = rdd.toDF("id")  
df: org.apache.spark.sql.DataFrame = [id: int]  
  
scala>
```

3. Above code will create Data frame with id as a column. To display the data in the Data frame use below command.

```
df.show()
```

It will display the below output.

```
scala> df.show()  
+----+  
| id |  
+----+  
|  1 |  
|  2 |  
|  3 |  
|  4 |  
|  5 |  
+----+  
  
scala>
```

When done, exit the shell using **ctrl-d**.

Conclusion

You should now have a working installation of Apache Spark on Windows 10 with all dependencies installed. Get started running an instance of Spark in your Windows environment.