

Apache Spark Tutorial-Run your First Spark Program

[REQUEST INFO](#)

What will you learn from this Spark Tutorial?

In this spark scala tutorial you will learn-

- Steps to install spark
- Deploy your own Spark cluster in standalone mode.
- Running your first spark program : Spark word count application.

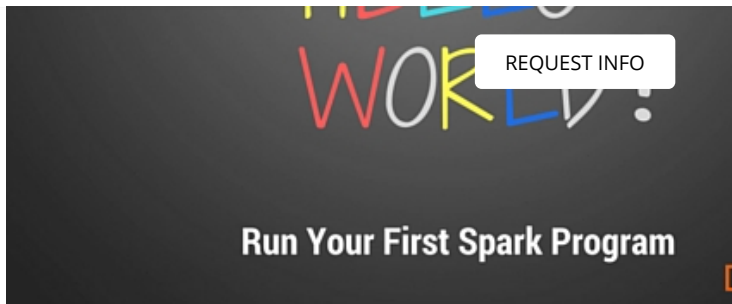
Pre-requisites to Getting Started with this Apache Spark Tutorial

Before you get a hands-on experience on how to run your first spark program, you should have-

- i. Understanding of the entire Apache Spark Ecosystem
- ii. Read the [Introduction to Apache Spark tutorial](#)



Microsoft Professional
Hadoop Certification
Program



Modes of Apache Spark Deployment

Before we begin with the Spark tutorial, let's understand how we can deploy spark to our systems –

1. Standalone Mode in Apache Spark

Spark is deployed on the top of Hadoop Distributed File System (HDFS). For computations, Spark and MapReduce run in parallel for the Spark jobs submitted to the cluster.

2. Hadoop YARN/ Mesos

Apache Spark runs on Mesos or YARN (Yet another Resource Navigator, one of the key features in the second-generation Hadoop) without any root-access or pre-installation. It integrates Spark on top Hadoop stack that is already present on the system.

3. SIMR (Spark in Map Reduce)

This is an add-on to the standalone deployment where Spark jobs can be launched by the user and they can use the spark shell without any

Online courses

- ▶ Hadoop Training
- ▶ Spark Training
- ▶ Data Science in Python
- ▶ Data Science in R
- ▶ Data Science Training
- ▶ Hadoop Training in California
- ▶ Hadoop Training in New York
- ▶ Hadoop Training in Texas
- ▶ Hadoop Training in Virginia
- ▶ Hadoop

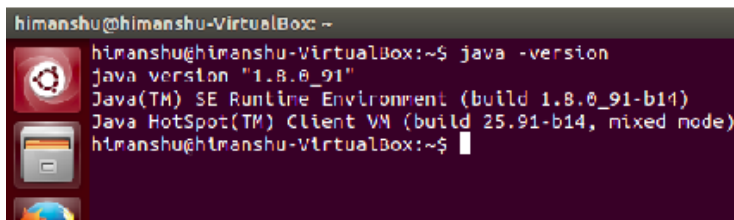


Getting Started with Apache Spark Standalone Mode of Deployment

Step 1: Verify if Java is installed

Java is a pre-requisite software for running Spark Applications. Use the following command to verify if Java is installed -

```
$java -version
```

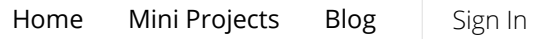


The above screenshot shows the version details of the Java installed on the machine. In case Java is not installed then head on to our [Hadoop Tutorial for Installation](#). Follows the steps listed under "Install Java" section of the Hadoop Tutorial to proceed with the Installation.

Step 2 - Verify if Spark is installed

As Apache Spark is used through Scala programming language, Scala should be installed to proceed with installing spark cluster

[jersey](#)[Hadoop Training in Dallas](#)[Hadoop Training in Atlanta](#)[Hadoop Training in Chicago](#)[Hadoop Training in Canada](#)[Hadoop Training in Charlotte](#)[Hadoop Training in Abudhabi](#)[Hadoop Training in Dubai](#)[Hadoop Training in Detroit](#)[Hadoop Training in Edison](#)[Hadoop Training in Germany](#)[Hadoop Training in](#)



Topics

- Hadoop Training in Sanjose

Learn Hadoop by working on interesting Big Data and Hadoop Projects for just \$9

```
$sudo apt-get install scala
```

Verifying the installation

Step 3: Download and Install Apache Spark:

4 of 15

[Home](#)[Mini Projects](#)[Blog](#)[Sign In](#)[Back to tutorial home](#)[About](#)[Videos](#)[Blogs](#)[Topics](#)

downloads folder. To install spark, extract the tar file using the following comma

[REQUEST INFO](#)

(In this spark tutorial, we are using spark-1.3.1-bin-hadoop2.6 version)

```
$ tar xvf spark-1.6.1-bin-hadoop2.6.tgz
```

```
himanshu@himanshu-VirtualBox:~$ cd Downloads
himanshu@himanshu-VirtualBox:~/Downloads$ ls
jdk-8u91-linux-i586.tar.gz  spark-1.6.1-bin-hadoop2.6.tgz  spark-1.6.1.tgz
himanshu@himanshu-VirtualBox:~/Downloads$ tar xzf spark-1.6.1-bin-hadoop2.6.tgz
himanshu@himanshu-VirtualBox:~/Downloads$ ls
jdk-8u91-linux-i586.tar.gz  spark-1.6.1-bin-hadoop2.6  spark-1.6.1-bin-hadoop2.6.tgz
himanshu@himanshu-VirtualBox:~/Downloads$
```

Move the spark downloaded files from the downloads folder to your local system where you plan to run your spark applications. Use the commands:

```
$ sudo su -
Password:
# cd /home/himanshu/Downloads/
# mv spark-1.6.1-bin-hadoop2.6
/usr/local/spark
# exit
```

```
himanshu@himanshu-VirtualBox:~/Downloads$ ls
jdk-8u91-linux-i586.tar.gz  spark-1.6.1-bin-hadoop2.6  spark-1.6.1-bin-hadoop2.6.tgz  sp
himanshu@himanshu-VirtualBox:~/Downloads$ cd
himanshu@himanshu-VirtualBox:~$ sudo su
root@himanshu-VirtualBox:/home/himanshu# cd /home/himanshu/Downloads/
root@himanshu-VirtualBox:/home/himanshu/Downloads# mv spark-1.6.1-bin-hadoop2.6 /usr/local
root@himanshu-VirtualBox:/home/himanshu/Downloads# ls
jdk-8u91-linux-i586.tar.gz  spark-1.6.1-bin-hadoop2.6.tgz  spark-1.6.1.tgz
root@himanshu-VirtualBox:/home/himanshu/Downloads# cd /usr/local
root@himanshu-VirtualBox:/usr/local$ ls
bin  games  hadoop-2.7.2  hadoop-2.7.2-src.tar.gz  include  index.html.1  lib  s
etc  hadoop  hadoop-2.7.2-src  hadoop-2.7.2.tar.gz  index.html  jdk1.8.0_91  man  s
root@himanshu-VirtualBox:/usr/local# cd /usr/local/spark
root@himanshu-VirtualBox:/usr/local/spark# ls
bin  CHANGES.txt  conf  data  ec2  examples  lib  LICENCE  licenses  NOTICE  python  R
root@himanshu-VirtualBox:/usr/local/spark#
```

Spark Configuration

Let's setup the environment variable for Apache

[Home](#)[Mini Projects](#)[Blog](#)[Sign In](#)[Back to tutorial home](#)[About](#)[Videos](#)[Blogs](#)[Topics](#)[/bin](#)

REQUEST INFO

We add the above line `~/bashrc` file and save it. Setting the PATH variable will locate the Spark executables in the location `/usr/local/spark/bin`.

Enrol now for Hands-on [Spark Training](#) to become a Certified Spark Developer

Step 3: Verify Apache Spark installation

Verify the installation using the following command:

```
$spark-shell
```

In case the installation happened successfully, the above command will start Apache Spark in Scala.

```
himanish@himanish-VirtualBox:~/spark-shell$ spark-shell
log4j:WARN No appenders could be found for logger (org.apache.hadoop.metrics2.lib.MutableMetricFactory).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2faq.html#noconfig for more info.
Using spark's repl log4j profiles or /opt/apache/spark/log4j-defaults-repl.properties
to adjust logging level use $setLogLevel('INFO')
Welcome to
      Spark
    version 1.6.1

Using Scala version 2.10.5 (Java HotSpot(TM) Client VM, Java 1.8.0_91)
Type in expressions to have them evaluated.
Type :help for more information.
16/06/00 00:06:06 WARN Utils: Your hostname, himanish-VirtualBox resolves to a loopback address: 127.0.0.1; using
e:eth0)
16/06/00 00:06:06 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
Spark context available as sc.
16/06/00 00:06:45 WARN Connection: BonCP specified but not present in CLASSPATH (or one of dependencies)
16/06/00 00:06:49 WARN Connection: BonCP specified but not present in CLASSPATH (or one of dependencies)
16/06/00 00:07:13 WARN ObjectStore: Version information not found in metastore. hive.metastore.schema.verification
has schema version 1.2.0
16/06/00 00:07:18 WARN ObjectStore: Failed to get database default, returning NoSuchObjectException
Java HotSpot(TM) Client VM warning: You have loaded library /tmp/libnetty_transport_native_epoll1327465713734943635
stack guard. The VM will try to fix the stack guard now.
It's highly recommended that you fix the library with 'execstack -c <libfile>', or link it with '-fstack-protector'.
16/06/00 00:07:42 WARN Connection: BonCP specified but not present in CLASSPATH (or one of dependencies)
16/06/00 00:08:18 WARN Connection: BonCP specified but not present in CLASSPATH (or one of dependencies)
16/06/00 00:08:46 WARN ObjectStore: Version information not found in metastore. hive.metastore.schema.verification
has schema version 1.2.0
16/06/00 00:08:50 WARN ObjectStore: Failed to get database default, returning NoSuchObjectException
SQL context available as sqlContext.

scala>
```

In other words, for this, we just have to place the compiled version of Apache Spark

The most common way to [REQUEST INFO](#) applications on the cluster is to use the shell command `spark-submit`. When using `spark-submit` shell command the spark application need not be configured particularly for each cluster as the `spark-submit` shell script uses the cluster managers through a single interface. `Spark-submit` script has several flags that help control the resources used by your Apache Spark application. `Spark-submit` flags dynamically supply configurations to the Spark Context object.

Let's look at the same [hadoop MapReduce](#) example of Word Count in Apache Spark as well-

The input in the file `input.txt` contains the following text -

In as name to here them deny wise this. As rapid woody my he me which. Men but they fail shew just wish next put. Led all visitor musical calling nor her. Within coming figure sex things are. Pretended concluded did repulsive education smallness yet yet described. Had countryman his pressed shewing. No gate dare rose he. Eyes year if miss he as upon.

Below is the source code for the Word Count program in Apache Spark -

```
import org.apache.spark.SparkContext
```



```
def main(args: Array[String]) {  
    // REQUEST INFO  
  
    val sc = new SparkContext(  
        "local", "Word Count", "/usr/local  
/spark", Nil, Map(),  
        Map())  
    val input =  
    sc.textFile("input.txt")  
    Val count = input.flatMap(line  
⇒ line.split(" "))  
    .map(word ⇒ (word, 1))  
    .reduceByKey(_ + _)  
    count.saveAsTextFile("outfile")  
    System.out.println("OK");  
}  
}
```

Let's understand the word count example in Spark step by step –

Linking with Apache Spark

The first step is to explicitly import the required spark classes into your Spark program which is done by adding the following lines -

```
import org.apache.spark.SparkContext  
import org.apache.spark.SparkContext._  
import org.apache.spark._
```

Creating a Spark Context Object

The next step is to create a Spark context object with the desired spark configuration that tells Apache Spark on how to access a cluster. The below line of code in the word count example

[Home](#)[Mini Projects](#)[Blog](#)[Sign In](#)[Back to tutorial home](#)[About](#)[Videos](#)[Blogs](#)[Topics](#)

"Word Count" – This is the application that you want to run.

REQUEST INFO

"local"- This parameter denotes the master URL to connect the spark application to.

/usr/local/spark- This parameter denotes the home directory of Apache Spark.

Map() – The first map specifies the environment whilst the second one specifies the variables to work nodes.\

Creating a Spark RDD

The next step in the Spark Word count example creates an input Spark RDD that reads the text file input.txt using the Spark Context created in the previous step-

```
val input = sc.textFile("input.txt")
```

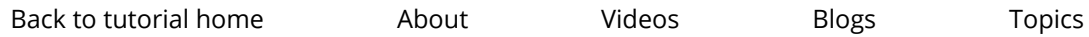
Spark RDD Transformations in Wordcount Example

The below lines of spark application code transform the input RDD to count RDD -

```
Val count = input.flatMap (line => line. Split (" "))  
                  .map (word => (word, 1))  
                  .reduceByKey (_ + _)
```

In the above piece of code, flatMap () is used to tokenize the lines from input text file into words.

Map () method counts the frequency of each word.



REQUEST INFO

Let's start Spark shell

[illegible]

Let's create a Spark RDD using the input file that we want to run our first Spark program on. You should specify the absolute path of the input file-

On executing the above command, the following output is observed -

```
inputFile: org.apache.spark.rdd.RDD[String] = input.Lxl MapPartitionsRDD[1] at Le
```

[Home](#)[Mini Projects](#)[Blog](#)[Sign In](#)[Back to tutorial home](#)[About](#)[Videos](#)[Blogs](#)[Topics](#)

a function that returns a sequence for each element in the list, and flatte REQUEST INFO into the original list.

- b. Each word is read and key-value pairs are created for each one of them using **map** transformation. This will assign the value '1' to each of the work-keys.
- c. Finally the values of similar keys are added to get the final word count using **reduceByKey** function.

```
scala> val counts = inputfile.  
flatMap (line => line. Split (" "  
").map (word => (word, "  
1)).reduceByKey (_+_)
```

You will get the following output:

```
scala> val counts = inputfile.flatMap(line => line.split(" ").map(word => (word, 1)).  
counts: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[4] at reduceByKey at <co
```

The next step is to store the output in a text file and exit the spark shell.

```
scala> counts.saveAsTextFile  
( "output")
```

```
scala> counts.saveAsTextFile("output")  
scala> exit
```

Go to the output directory (location where you have created the file named output). Use 'ls'



Using the cat command, print the contents of the output file to find the occurrence of each word in the input.txt file -

```
himanshu@himanshu-VirtualBox:~$ cd output
himanshu@himanshu-VirtualBox:~/output$ ls
part-000000 _SUCCESS
himanshu@himanshu-VirtualBox:~/output$ cat part-000000
(fail,1)
(Led,1)
(next,1)
(nor,1)
(rose,1)
(smallness,1)
(countryman,1)
(are.,1)
(Had,1)
(here,1)
(woody,1)
(put.,1)
```

[PREVIOUS](#)[NEXT](#)



Home

Mini Projects

Blog

Sign In

Back to tutorial home

About

Videos

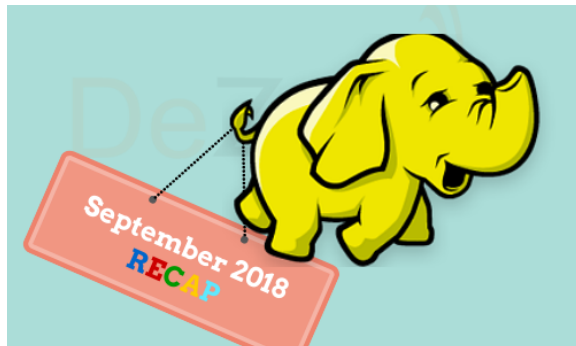
Blogs

Topics

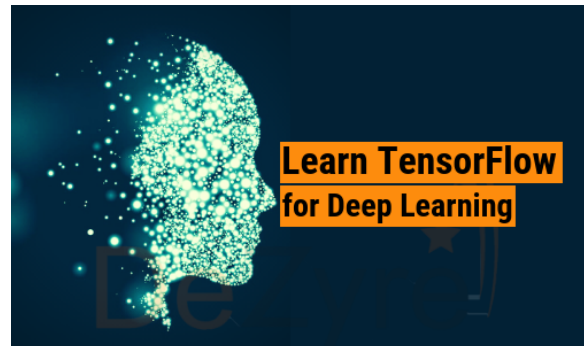
REQUEST INFO

Data Cleaning in Python

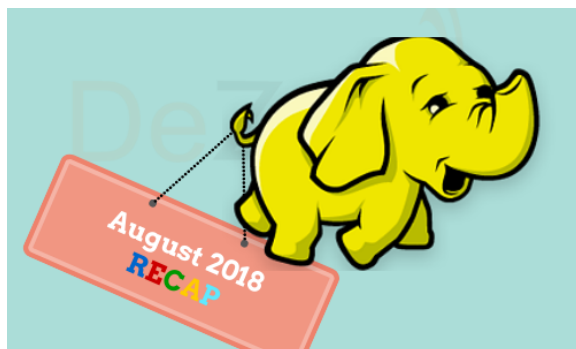
Python Pandas Dataframe Tutorials



Recap of Hadoop News for September 2018



Introduction to TensorFlow for Deep Learning



Recap of Hadoop News for August 2018



AWS vs Azure-Who is the big winner in the cloud war?

Other Tutorials

Big Data and Hadoop Training Courses in Popular Cities

- Microsoft Big Data and Hadoop Certification

➤ Hadoop Training in Texas

➤ Hadoop Training in California

➤ Hadoop Training in Dallas

➤ Hadoop Training in Chicago

➤ Hadoop Training in Charlotte

➤ Hadoop Training in Dubai

➤ Hadoop Training in Edison

➤ Hadoop Training in Fremont

➤ Hadoop Training in San Jose
- Hadoop Training in New Jersey

➤ Hadoop Training in New York

➤ Hadoop Training in Atlanta

➤ Hadoop Training in Canada

➤ Hadoop Training in Abu Dhabi

➤ Hadoop Training in Detroit

➤ Hadoop Trainging in Germany

➤ Hadoop Training in Houston

➤ Hadoop Training in Virginia

➤ Hadoop Training in Washington



[Home](#) [Mini Projects](#) [Blog](#) [Sign In](#)

[Back to tutorial home](#) [About](#) [Videos](#) [Blogs](#) [Topics](#)

REQUEST INFO