*Learn Today. Lead Tomorrow.*

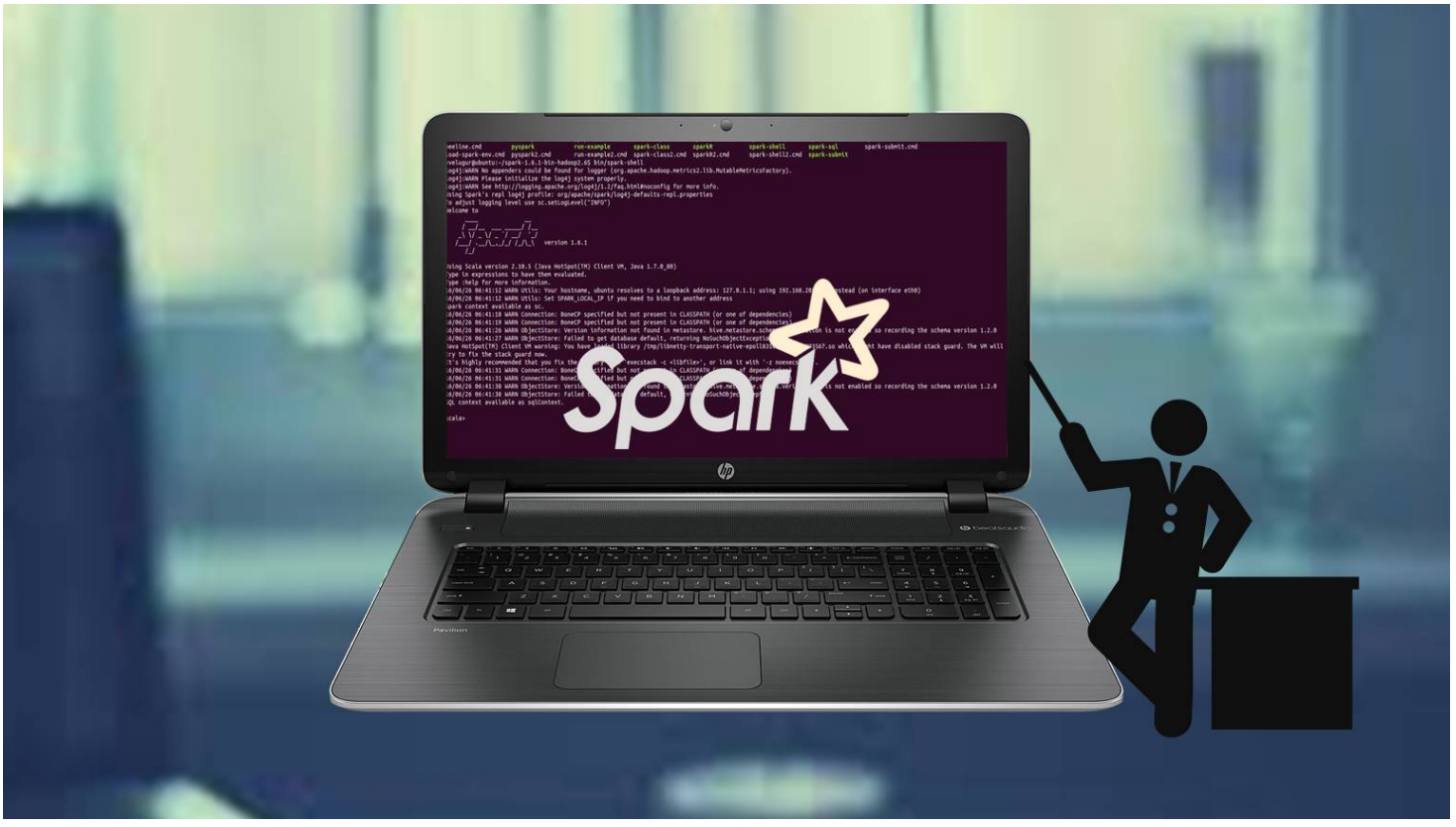



**Spark Tutorials**

ther
DD
RDD

This tutorial will take you through Apache Spark shell commands list to perform common operations of Apache spark. You can create RDDs and perform various transformations and actions like filter operation, partitions, cache, count, collect, etc. We will also discuss integration of spark with Hadoop, how spark reads the data from HDFS and write to HDFS?. This is an Apache spark beginners guide with step by step list of basic spark commands/operations to interact with spark shell.

Before starting you should understanding What is Spark and you must have spark installed, to install Apache spark you can follow this tutorial.




## 2. Apache Spark Shell Commands

**Start the Spark Shell**

```
$bin/spark-shell
```

### 2.1. Create a new RDD

Read File from local filesystem and create an RDD.

```
scala> val data = sc.textFile("data.txt")
```

*Note: sc is the object of SparkContext*

*Note: You need to create a file data.txt in Spark_Home directory*

### 2.2. Create a new RDD

```
scala> val no = Array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
scala> val noData = sc.parallelize(no)
```

### 2.3. Create a new RDD

```
scala> val newRDD = no.map(data => (data * 2))
```

These are three methods to create the RDD. The first method is used when data is already available with the external systems like local filesystem, HDFS, HBase, Cassandra, S3, etc. RDD can be created by calling textFile method of Spark Context with path / URL as the argument.The second approach can be used with the existing collections and the third one is a way to create new RDD from the existing one.

### 2.4. Number of Items in the RDD

Count the number of items available in the RDD. To count the items we need to call Action

```
scala> data.count()
```

### 2.5. Filter Operation

Filter the RDD and create new RDD of items which contains word "DataFlair". To filter, we need to call transformation filter, which will return a new RDD with subset of items

```
scala> val DFData = data.filter(line => line.contains("DataFlair"))
```

### 2.6. Transformation and Action together

For complex requirements, we can chain multiple operations together like filter transformation and count action together

```
scala> data.filter(line => line.contains("DataFlair")).count()
```

### 2.7. Read the first item from the RDD

```
scala> data.first()
```

### 2.8. Read the first 5 item from the RDD

```
scala> data.take(5)
```

### 2.9. RDD partitions

An RDD is made up of multiple partitions, to count number of partitions-

```
scala> data.partitions.length
```

Note: Minimum no. of partitions in the RDD is 2 (by default). When we create RDD from HDFS file then a number of blocks will be equals to the number of partitions.

### 2.10. Cache the file

Caching is the optimization technique. Once we cache the RDD in the memory all future computation will work on the in-memory data, which saves disk seeks and improve the performance.

```
scala> data.cache()
```

RDD will not be cached once you run above operation, you can visit the web UI: http://localhost:4040/storage, it will be blank. RDDs are not explicitly cached once we run cache(), rather RDDs will be cached once we run the Action, which actually needs data read from the disk.

**Let's run some actions**

```
scala> data.count()
```

```
scala> data.collect()
```

Now as we have run some actions on the data file, which needs to be read from the disk to perform those operations. During this process Spark will cache the file, so that for all future operations will get the data from the memory (no need of any disk interaction). Now if we run any transformation or action it will be done in-memory and will be much faster.

### 2.11. Read Data from HDFS file

To read data from HDFS file we can specify complete hdfs URL like hdfs://IP:PORT/PATH

```
scala> var hFile = sc.textFile("hdfs://localhost:9000/inp")
```