

```
(base) [bigdata@localhost ~]$ spark-shell
```

Spark context Web UI available at <http://192.168.150.128:4040>

Spark context available as `sc` (master = local[*], app id = local-1629625151951).

Spark session available as 'spark'.

Welcome to

version 3.0.1

Using Scala version 2.12.10 (OpenJDK 64-Bit Server VM, Java 1.8.0_252)

Type in expressions to have them evaluated.

Type :help for more information.

```
scala> val a=sc.parallelize(1 to 120)
```

```
a: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[0] at parallelize at <console>:24
```

```
scala> a.collect
```

```
res0: Array[Int] = Array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25,
26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53,
54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81,
82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106,
107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120)
```

```
scala> val a=sc.parallelize(1 to 120,10)
```

```
a: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[1] at parallelize at <console>:24
```

```
scala> a.collect
```

```
res1: Array[Int] = Array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25,
26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53,
54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81,
82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106,
107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120)
```

```
scala> a.distinct
```

```
res2: org.apache.spark.rdd.RDD[Int] = MapPartitionsRDD[4] at distinct at <console>:26
```

```
scala> a.partitions.length
```

```
res3: Int = 10
```

```
scala> val mydata = Array(1,2,3,4,5,6,7,8,9,10)
```

```
mydata: Array[Int] = Array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
```

```
scala> val rdd1 = sc.parallelize(mydata)
```

```
rdd1: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[5] at parallelize at <console>:26
```

```

scala> rdd1.foreach{x=>println(x)}
1
2
3
4
5
6
7
8
9
10
scala> val dataSeq = Seq(("Java", 20000), ("Python", 100000), ("Scala", 3000))
dataSeq: Seq[(String, Int)] = List((Java,20000), (Python,100000), (Scala,3000))
scala> val rdd=spark.sparkContext.parallelize(dataSeq)
rdd: org.apache.spark.rdd.RDD[(String, Int)] = ParallelCollectionRDD[6] at parallelize at <console>:25
scala> val rdd=spark.sparkContext.parallelize(Seq(("Java", 20000),
| | ("Python", 100000), ("Scala", 3000)))
rdd: org.apache.spark.rdd.RDD[(String, Int)] = ParallelCollectionRDD[7] at parallelize at <console>:23
scala> rdd.foreach(println)
(Java,20000)
(Python,100000)
(Scala,3000)
scala> val data = 1 to 10000
data: scala.collection.immutable.Range.Inclusive = Range 1 to 10000
scala> val distData = sc.parallelize(data)
distData: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[8] at parallelize at <console>:26
scala> distData.filter(_ < 10).collect()
res6: Array[Int] = Array(1, 2, 3, 4, 5, 6, 7, 8, 9)

```

Wordcount example

```

scala> val book2=sc.textFile("inputfile.txt")
book2: org.apache.spark.rdd.RDD[String] = inputfile.txt MapPartitionsRDD[8] at textFile at
<console>:24
scala> val a=book2.map(x=>x.split(" "))
a: org.apache.spark.rdd.RDD[Array[String]] = MapPartitionsRDD[9] at map at <console>:25
scala> val counts = book2.flatMap(line => line.split(" ")).map(word => (word, 1)).reduceByKey(_+_);
counts: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[12] at reduceByKey at <console>:25
scala> counts.toDebugString
res4: String =
(2) ShuffledRDD[12] at reduceByKey at <console>:25 []
+- (2) MapPartitionsRDD[11] at map at <console>:25 []
|   MapPartitionsRDD[10] at flatMap at <console>:25 []
|   inputfile.txt MapPartitionsRDD[8] at textFile at <console>:24 []
|   inputfile.txt HadoopRDD[7] at textFile at <console>:24 []
scala> counts.cache()
res5: counts.type = ShuffledRDD[12] at reduceByKey at <console>:25
scala> counts.saveAsTextFile ("output1")

```

```
scala>book2.take(5)
res8: Array[String] = Array("people are not as beautiful as they look, ", as they walk or as they talk.,
"they are only as beautiful as they love, ", as they care as they share.)
(base) [bigdata@localhost ~]$ cd output1/
(base) [bigdata@localhost output1]$ ls -l
part-00000
part-00001
_SUCCESS
(base) [bigdata@localhost output1]$ cat part-00000
(talk.,1)
(are,2)
(only,1)
(as,8)
(,1)
(they,7)
(love,,1)
(base) [bigdata@localhost output1]$ cat part-00001
(not,1)
(people,1)
(share,,1)
(or,1)
(care,1)
(beautiful,2)
(walk,1)
(look,,1)
```

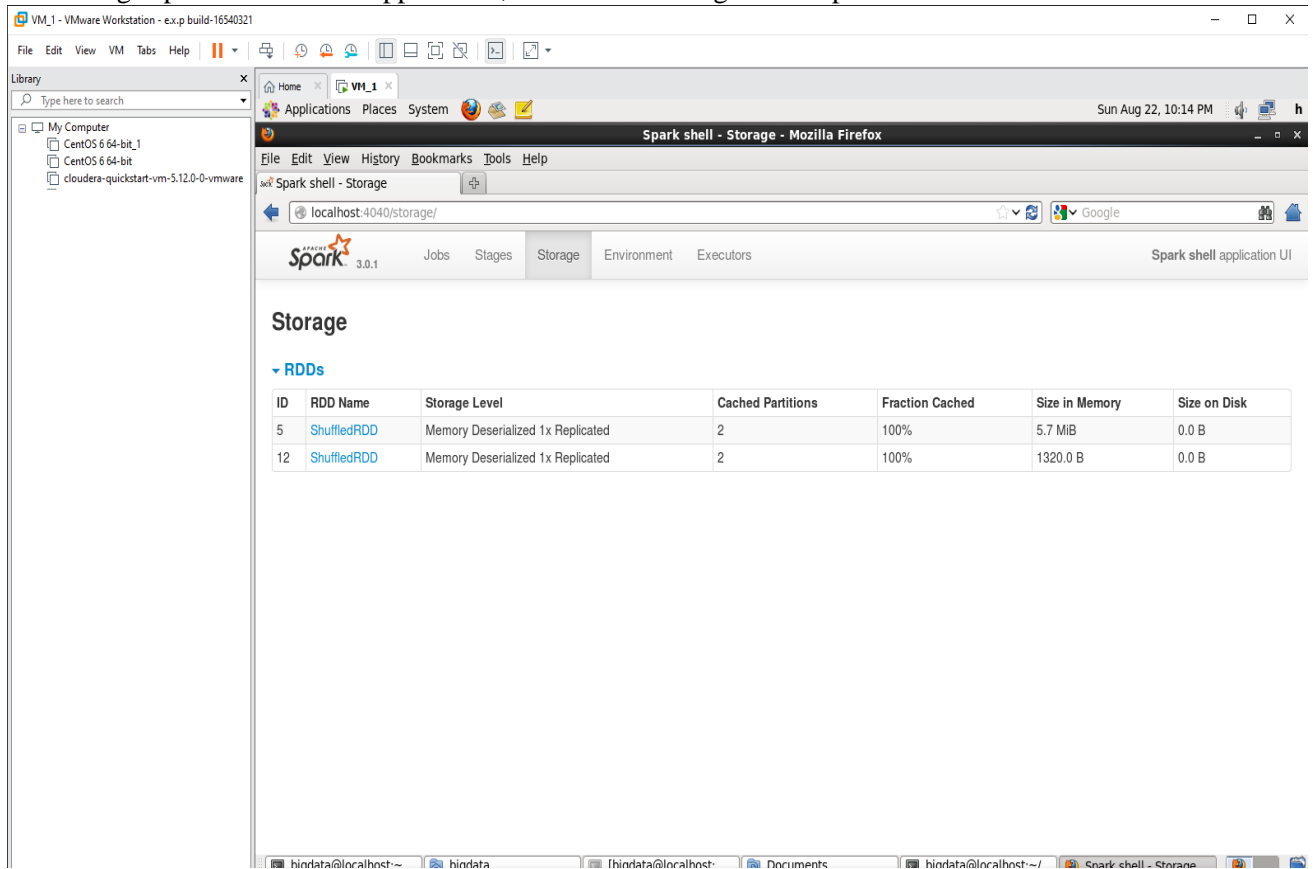
Check Spark WebUI through <http://localhost:4040>

The screenshot shows the Spark WebUI interface in a Mozilla Firefox browser window. The page title is "Spark shell - Spark Jobs - Mozilla Firefox". The URL bar shows "localhost:4040/jobs/". The interface includes a sidebar with "My Computer" and "CentOS 6 64-bit" options. The main content area displays "Spark Jobs (5)" with a table of completed jobs. The table has columns for Job Id, Description, Submitted, Duration, Stages: Succeeded/Total, and Tasks (for all stages): Succeeded/Total. The jobs listed are:

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
4	take at <console>:26 take at <console>:26	2021/08/22 22:09:08	14 ms	1/1	1/1
3	take at <console>:26 take at <console>:26	2021/08/22 22:09:08	15 ms	1/1	1/1
2	runJob at SparkHadoopWriter.scala:78 runJob at SparkHadoopWriter.scala:78	2021/08/22 22:08:29	0.3 s	2/2	4/4
1	take at <console>:26 take at <console>:26	2021/08/22 21:27:22	20 ms	1/1	1/1
0	runJob at SparkHadoopWriter.scala:78 runJob at SparkHadoopWriter.scala:78	2021/08/22 21:27:05	3 s	2/2	4/4

The interface also shows "User: bigdata", "Total Uptime: 53 min", "Scheduling Mode: FIFO", and "Completed Jobs: 5". There are pagination controls at the bottom of the table.

The storage space used for the application, which are running on the Spark shell.



The screenshot shows a VMware Workstation window titled "VM_1 - VMware Workstation - e.x.p build-16540321". Inside the VM, a Mozilla Firefox browser is open to the "Spark shell - Storage" page. The page displays the "Storage" tab of the Spark shell application UI, which lists RDDs and their storage details.

The table shows the following data:

ID	RDD Name	Storage Level	Cached Partitions	Fraction Cached	Size in Memory	Size on Disk
5	ShuffledRDD	Memory Deserialized 1x Replicated	2	100%	5.7 MiB	0.0 B
12	ShuffledRDD	Memory Deserialized 1x Replicated	2	100%	1320.0 B	0.0 B