PRACTICAL-10

AIM:

To install, deploy & configure Apache Spark Cluster. To Select the fields from the dataset using Spark SQL. To explore Spark shell and read from HDFS.

THEORY:

• Apache Spark is an open-source unified analytics engine for large-scale data processing. Spark provides an interface for programming entire clusters with implicit data parallelism and fault tolerance. Originally developed at the University of California, Berkeley's AMPLab, the Spark codebase was later donated to the Apache Software Foundation, which has maintained it since.

• Apache Spark has its architectural foundation in the resilient distributed dataset (RDD), a read-only multiset of data items distributed over a cluster of machines, that is maintained in a fault-tolerant way.

• The Dataframe API was released as an abstraction on top of the RDD, followed by the Dataset API. In Spark 1.x, the RDD was the primary application programming interface (API), but as of Spark 2.x use of the Dataset API is encouraged even though the RDD API is not deprecated. The RDD technology still underlies the Dataset API.

• Spark and its RDDs were developed in 2012 in response to limitations in the MapReduce cluster computing paradigm, which forces a particular linear dataflow structure on distributed programs: MapReduce programs read input data from disk, map a function across the data, reduce the results of the map, and store reduction results on disk.

• Spark's RDDs function as a working set for distributed programs that offers a (deliberately) restricted form of distributed shared memory.

• Inside Apache Spark the workflow is managed as a directed acyclic graph (DAG). Nodes represent RDDs while edges represent the operations on the RDDs.

• Spark facilitates the implementation of both iterative algorithms, which visit their data set multiple times in a loop, and interactive/exploratory data analysis, i.e., the repeated database-style querying of data.

• The latency of such applications may be reduced by several orders of magnitude compared to Apache Hadoop MapReduce implementation. Among the class of iterative algorithms are the training algorithms for machine learning systems, which formed the initial impetus for developing Apache Spark.

• Apache Spark requires a cluster manager and a distributed storage system. For cluster management, Spark supports standalone (native Spark cluster, where you can launch a cluster either manually or use the launch scripts provided by the install package. It isalso possible to run these daemons on a single machine for testing), Hadoop YARN, Apache Mesos or Kubernetes.

• For distributed storage, Spark can interface with a wide variety, including Alluxio, Hadoop Distributed File System (HDFS), MapR File System (MapR-FS), Cassandra, OpenStack Swift, Amazon S3, Kudu, Lustre file system, or a custom solution can be implemented.

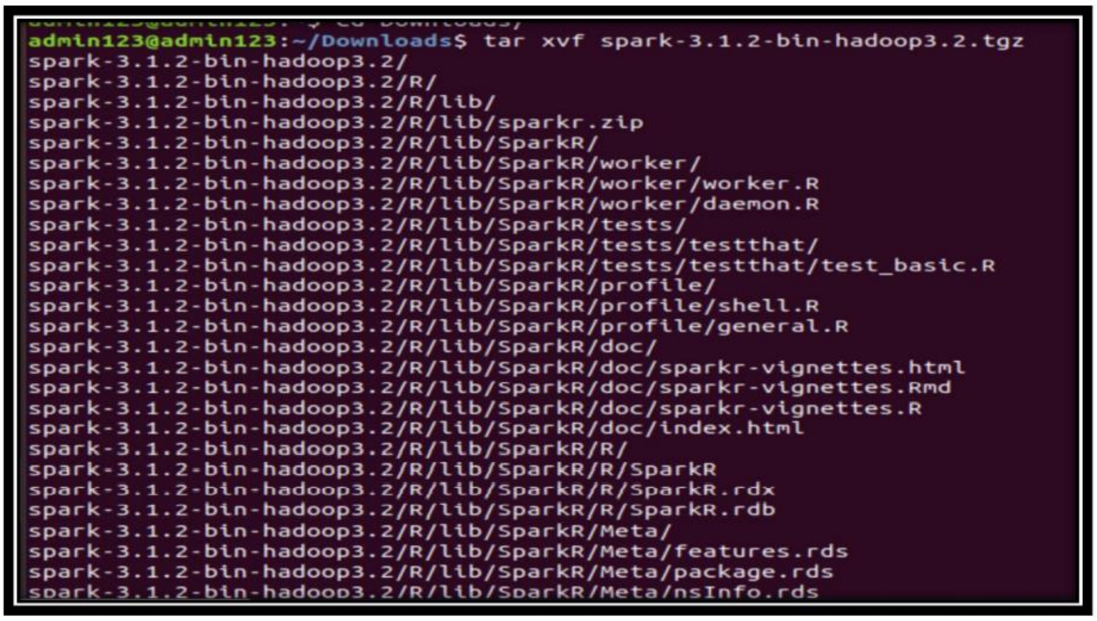
• Spark also supports a pseudo-distributed local mode, usually used only for development or testing purposes, where distributed storage is not required and the local file system can be used instead; in such a scenario, Spark is run on a single machine with one executor per CPU core.

CODE:

You can download spark from official download.

Then extract the file using tar command.

```
tar xvf spark-*
```



```
admin123@admin123:~/Downloads$ tar xvf spark-3.1.2-bin-hadoop3.2.tgz
spark-3.1.2-bin-hadoop3.2/
spark-3.1.2-bin-hadoop3.2/R/
spark-3.1.2-bin-hadoop3.2/R/lib/
spark-3.1.2-bin-hadoop3.2/R/lib/sparkr.zip
spark-3.1.2-bin-hadoop3.2/R/lib/SparkR/
spark-3.1.2-bin-hadoop3.2/R/lib/SparkR/worker/
spark-3.1.2-bin-hadoop3.2/R/lib/SparkR/worker/worker.R
spark-3.1.2-bin-hadoop3.2/R/lib/SparkR/worker/daemon.R
spark-3.1.2-bin-hadoop3.2/R/lib/SparkR/tests/
spark-3.1.2-bin-hadoop3.2/R/lib/SparkR/tests/testthat/
spark-3.1.2-bin-hadoop3.2/R/lib/SparkR/tests/testthat/test_basic.R
spark-3.1.2-bin-hadoop3.2/R/lib/SparkR/profile/
spark-3.1.2-bin-hadoop3.2/R/lib/SparkR/profile/shell.R
spark-3.1.2-bin-hadoop3.2/R/lib/SparkR/profile/general.R
spark-3.1.2-bin-hadoop3.2/R/lib/SparkR/doc/
spark-3.1.2-bin-hadoop3.2/R/lib/SparkR/doc/sparkr-vignettes.html
spark-3.1.2-bin-hadoop3.2/R/lib/SparkR/doc/sparkr-vignettes.Rmd
spark-3.1.2-bin-hadoop3.2/R/lib/SparkR/doc/sparkr-vignettes.R
spark-3.1.2-bin-hadoop3.2/R/lib/SparkR/doc/index.html
spark-3.1.2-bin-hadoop3.2/R/lib/SparkR/R/
spark-3.1.2-bin-hadoop3.2/R/lib/SparkR/R/SparkR
spark-3.1.2-bin-hadoop3.2/R/lib/SparkR/R/SparkR.rdx
spark-3.1.2-bin-hadoop3.2/R/lib/SparkR/R/SparkR.rdb
spark-3.1.2-bin-hadoop3.2/R/lib/SparkR/Meta/
spark-3.1.2-bin-hadoop3.2/R/lib/SparkR/Meta/features.rds
spark-3.1.2-bin-hadoop3.2/R/lib/SparkR/Meta/package.rds
spark-3.1.2-bin-hadoop3.2/R/lib/SparkR/Meta/nsInfo.rds
```

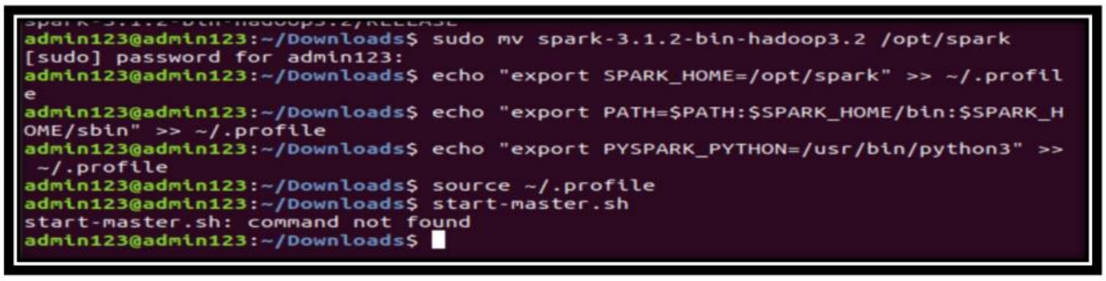
Finally, move the unpacked directory spark-3.0.1-bin-hadoop2.7 to the opt/spark directory.

```
sudo mv spark-3.0.1-bin-hadoop2.7 /opt/spark
```

Before starting a master server, you need to configure environment variables. There are a few Spark home paths you need to add to the user profile.

```
echo "export PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin" >> ~/.profile

echo "export PYSPARK_PYTHON=/usr/bin/python3" >> ~/.profile
```

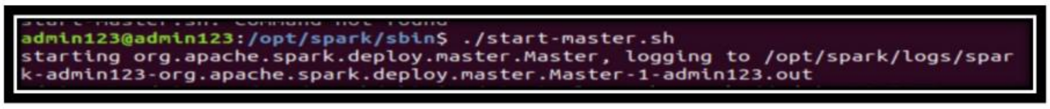


```
admin123@admin123:~/Downloads$ sudo mv spark-3.1.2-bin-hadoop3.2 /opt/spark
[sudo] password for admin123:
admin123@admin123:~/Downloads$ echo "export SPARK_HOME=/opt/spark" >> ~/.profil
e
admin123@admin123:~/Downloads$ echo "export PATH=$PATH:$SPARK_HOME/bin:$SPARK_H
OME/sbin" >> ~/.profile
admin123@admin123:~/Downloads$ echo "export PYSPARK_PYTHON=/usr/bin/python3" >>
 ~/.profile
admin123@admin123:~/Downloads$ source ~/.profile
admin123@admin123:~/Downloads$ start-master.sh
start-master.sh: command not found
admin123@admin123:~/Downloads$
```

Now that you have completed configuring your environment for Spark, you can start a master server.

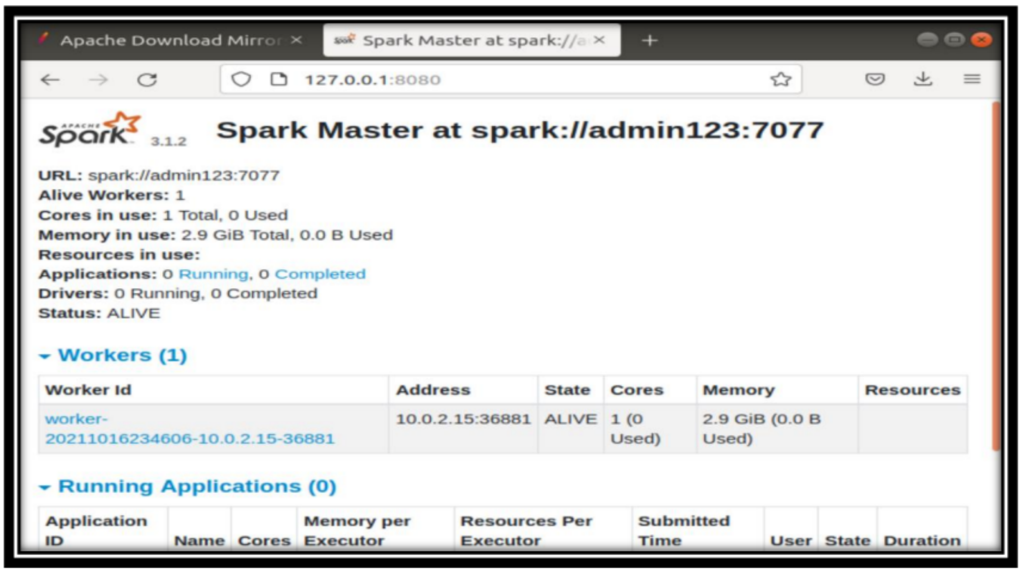
In the terminal, type:

```
start-master.sh
```



```
admin123@admin123:/opt/spark/sbin$ ./start-master.sh
starting org.apache.spark.deploy.master.Master, logging to /opt/spark/logs/spar
k-admin123-org.apache.spark.deploy.master.Master-1-admin123.out
```

Now that a worker is up and running, if you reload Spark Master's Web UI, you should see it on the list:



Spark Master at spark://admin123:7077

URL: spark://admin123:7077
Alive Workers: 1
Cores in use: 1 Total, 0 Used
Memory in use: 2.9 GiB Total, 0.0 B Used
Resources in use:
Applications: 0 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers (1)

Worker Id	Address	State	Cores	Memory	Resources
worker-20211016234606-10.0.2.15-36881	10.0.2.15:36881	ALIVE	1 (0 Used)	2.9 GiB (0.0 B Used)	

Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration

In this single-server, standalone setup, we will start one slave server along with the master server.

To do so, run the following command in this format:

```
start-slave.sh spark://master:port
```

The master in the command can be an IP or hostname.

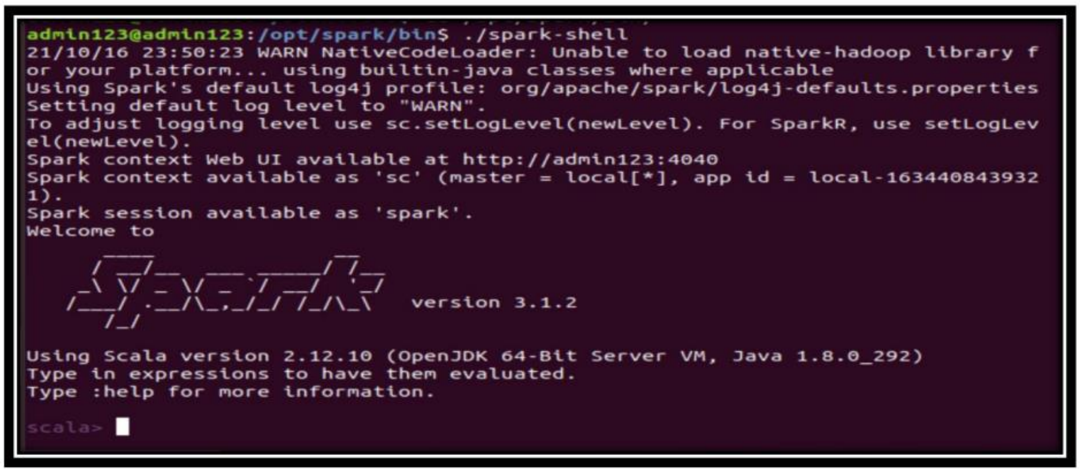
In our case it is ubuntu1:

```
start-slave.sh spark://ubuntu1:7077
```

After you finish the configuration and start the master and slave server, test if the Spark shell works.

Load the shell by entering:

```
spark-shell
```



```
admin123@admin123:/opt/spark/bin$ ./spark-shell
21/10/16 23:50:23 WARN NativeCodeLoader: Unable to load native-hadoop library f
or your platform... using builtin-java classes where applicable
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLev
el(newLevel).
Spark context Web UI available at http://admin123:4040
Spark context available as 'sc' (master = local[*], app id = local-163440843932
1).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.1.2
      /_/

Using Scala version 2.12.10 (OpenJDK 64-Bit Server VM, Java 1.8.0_292)
Type in expressions to have them evaluated.
Type :help for more information.

scala> 
```

CONCLUSION:

In this practical, we learnt about spark and installed it and configured it. We explored the spark shell as well.