Experiment No :- 06

Title: Apache Spark

Aim: Installation and configuration of Apache Spark on Local Machine. Execute basic commands used in Spark.

Theory:

Apache Spark

What is Apache Spark?

Apache Spark is a multi-language engine for executing data engineering, data science, and machine learning on single-node machines or clusters.

Key features

Batch/streaming data

Unify the processing of your data in batches and real-time streaming, using your preferred language: Python, SQL, Scala, Java or R.

SQL analytics

Execute fast, distributed ANSI SQL queries for dashboarding and ad-hoc reporting. Runs faster than most data warehouses.

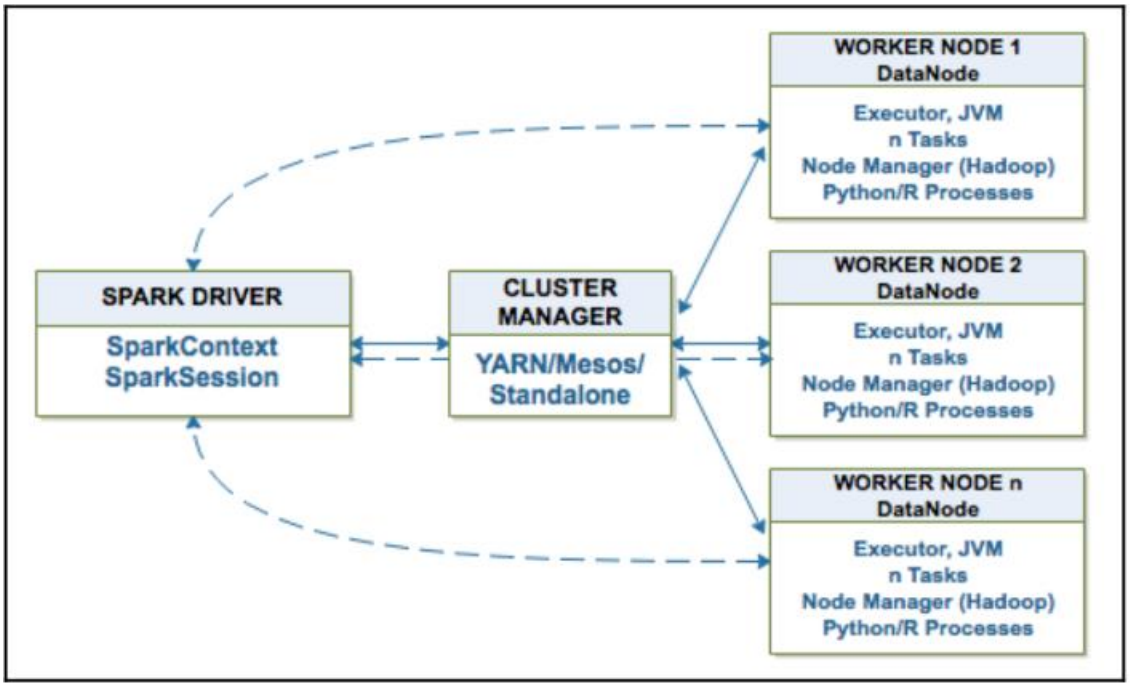
Data science at scale

Perform Exploratory Data Analysis (EDA) on petabyte-scale data without having to resort to downsampling

Machine learning

Train machine learning algorithms on a laptop and use the same code to scale to fault-tolerant clusters of thousands of machines.

The architecture of Spark



Spark's architecture consists of three primary components:

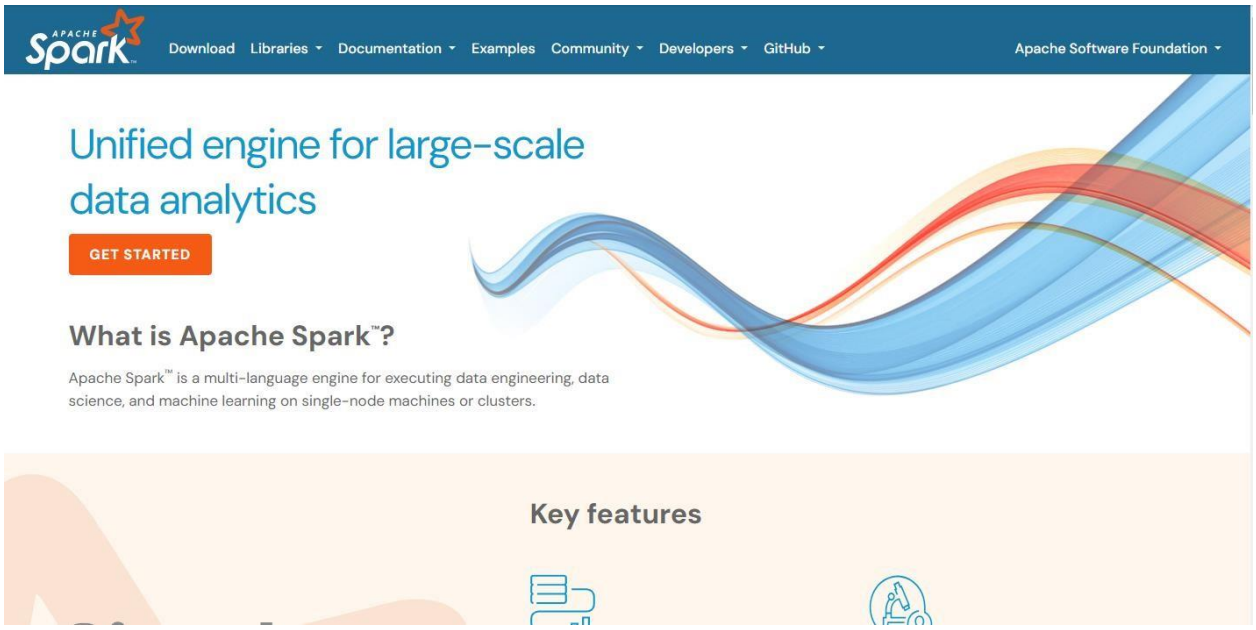
1. SparkSession/SparkContext (Driver): The entry point for Spark applications. The driver creates RDDs, performs operations, and coordinates tasks by sending instructions to worker nodes.

2. Cluster Manager: Manages resources and communication between worker nodes. It can be YARN, Mesos, or run in standalone mode. It handles node administration tasks like starting and stopping nodes.

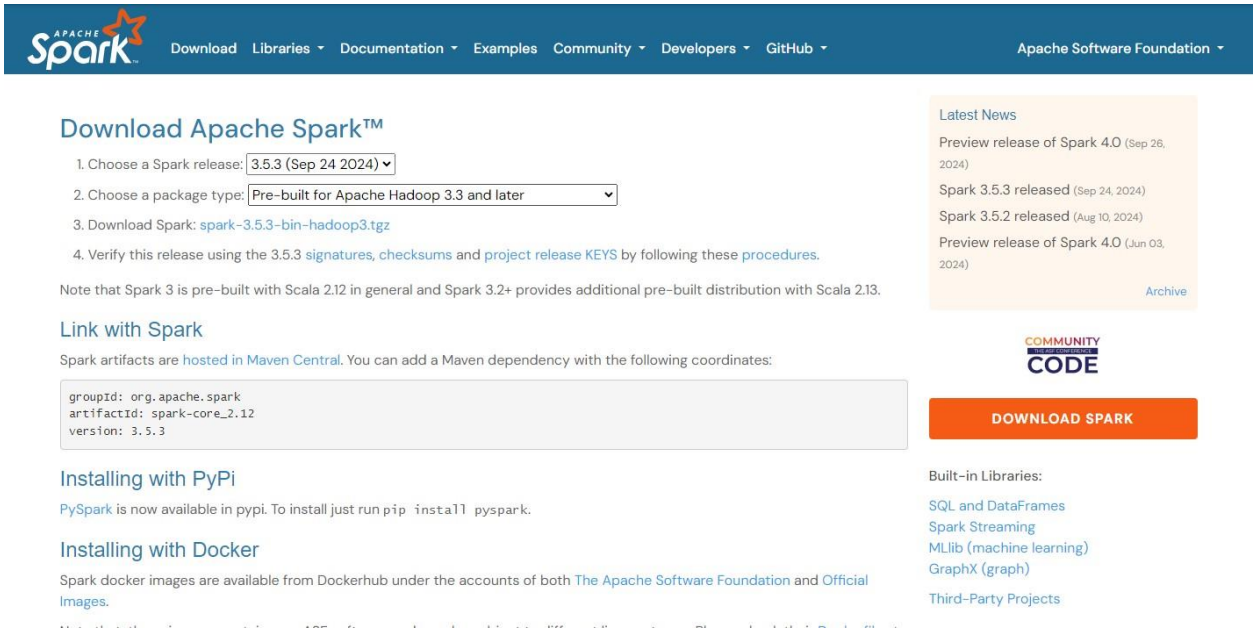
3. Worker Nodes: Hosts Spark's executor processes that perform tasks (actions and transformations). Each application has its own executor process to ensure isolation. The worker nodes include the Executor, JVM, and application-specific processes like Python or R.

Installation Of Apache Spark:

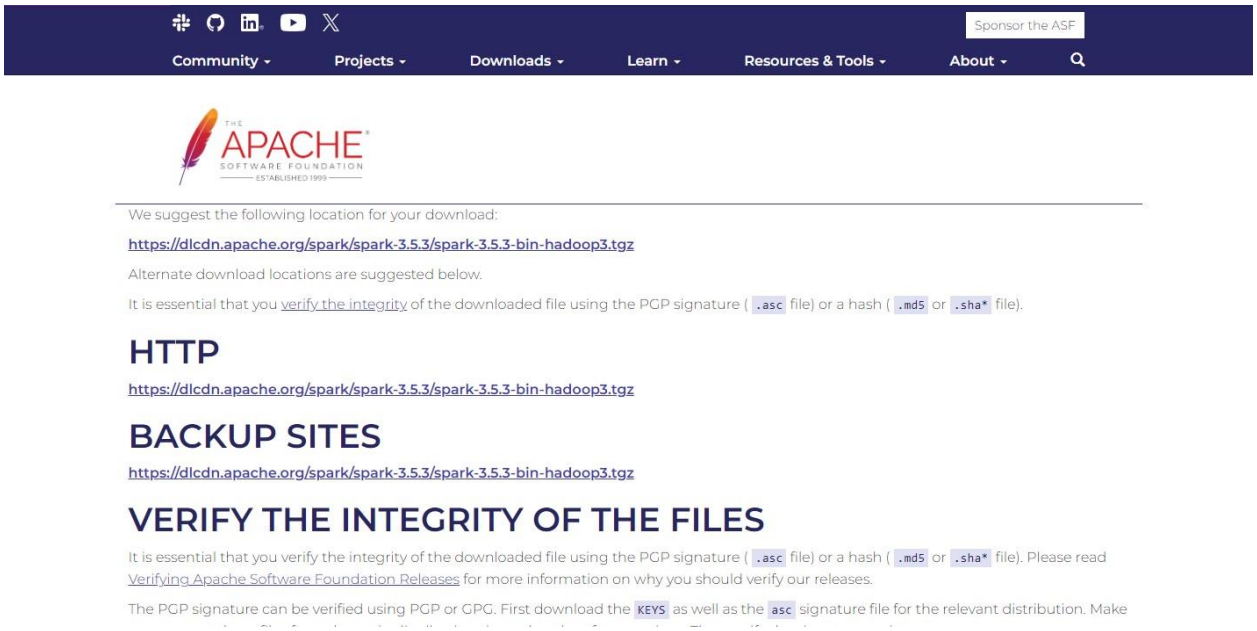
Step 1 :- Visit the official website to download [Apache spark]



2. Click the Download a release now! link to access the mirrors page.



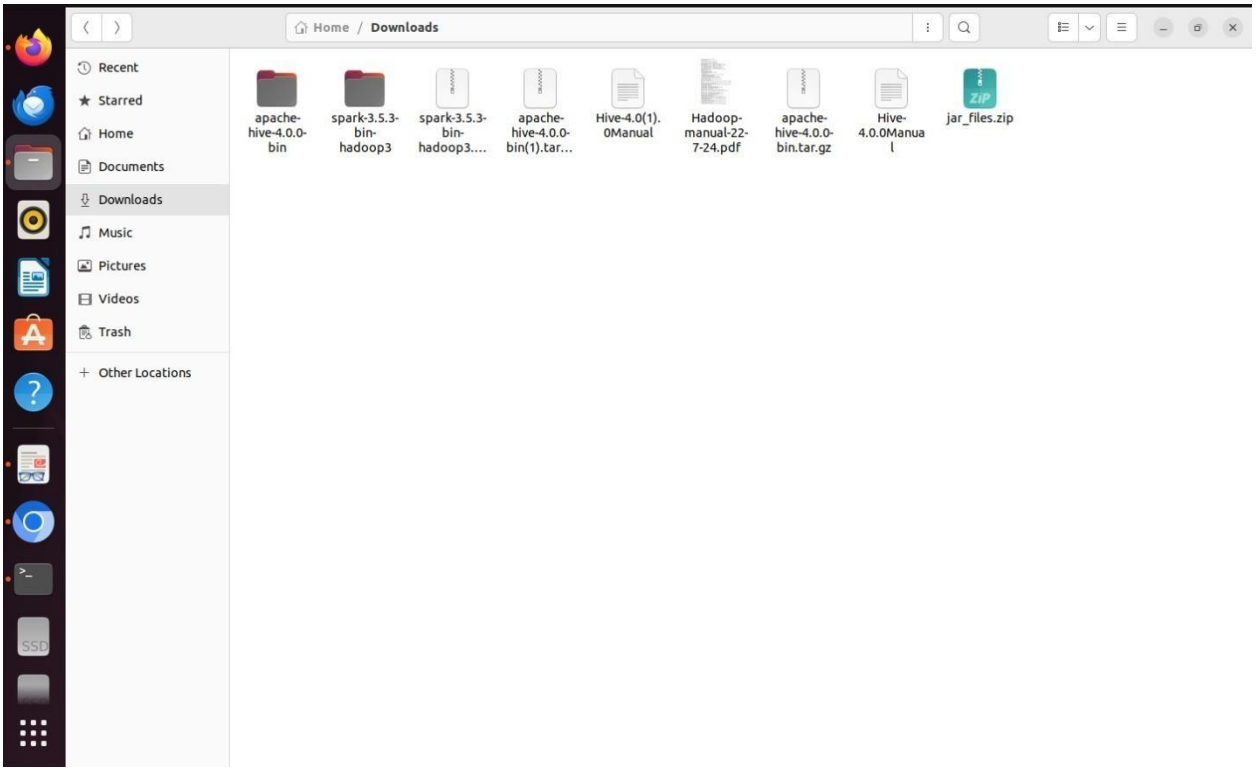
3. Choose the default mirror link.



4. Here to check the installation of apache spark

```
Open  .bashrc  /home/hduser  Save
# some more ls aliases
alias ll='ls -alF'
alias la='ls -A'
alias l='ls -CF'

# Add an "alert" alias for long running commands.  Use like so:
#   sleep 10; alert
alias alert='notify-send --urgency=low -i "$([ $? = 0 ] && echo terminal || echo error)" "$(history|tail -n1|sed -e '\''s/^\s*[0-9]\+\s*//;s/[;&|]\s*alert$//'\'')"'

# Alias definitions.
# You may want to put all your additions into a separate file like
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi
#HADOOP VARIABLES START
export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64
export HADOOP_HOME=/usr/local/hadoop
export PATH=$PATH:$HADOOP_HOME/bin
export PATH=$PATH:$HADOOP_HOME/sbin
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export HADOOP_YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib"
export SPARK_HOME=/home/hduser/Downloads/spark-3.5.3-bin-hadoop3
export PATH=$PATH:$SPARK_HOME/bin
export PATH=$PATH:$SPARK_HOME/sbin
#HADOOP VARIABLES END
sh  Tab Width: 8  Ln 1, Col 1  INS
```

```
hduser@bv-ThinkCentre-neo-50t-Gen-3: ~
  GNU nano 6.2                    /home/hduser/.bashrc
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi
#HADOOP VARIABLES START
export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64
export HADOOP_HOME=/usr/local/hadoop
export PATH=$PATH:$HADOOP_HOME/bin
export PATH=$PATH:$HADOOP_HOME/sbin
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export HADOOP_YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib"
#HADOOP VARIABLES END

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^/ Go To Line
```



5. To run the basic commands

```
Type :help for more information.

scala> val data = seq(1,2,3,4,5)
<console>:22: error: not found: value seq
       val data = seq(1,2,3,4,5)
                  ^

scala> val data = seq(1,2,3,4,5)
<console>:22: error: not found: value seq
       val data = seq(1,2,3,4,5)
                  ^

scala> val data = Seq(1,2,3,4,5)
data: Seq[Int] = List(1, 2, 3, 4, 5)

scala> val rdd =sc.parallelize(data)
rdd: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[0] at parallelize at <console>:24

scala> rdd.collect()
res0: Array[Int] = Array(1, 2, 3, 4, 5)

scala> res0:Array[int]=Array(1,2,3,4,5)
<console>:1: error: ';' expected but '=' found.
       res0:Array[int]=Array(1,2,3,4,5)
                      ^

scala> val evenNumber = rdd.filter(_%2==0)
evenNumber: org.apache.spark.rdd.RDD[Int] = MapPartitionsRDD[1] at filter at <console>:23

scala> evenNumber.collect()
res1: Array[Int] = Array(2, 4)

scala> val rdd = sc.parallelize(seq(1,2,3,4,5))
<console>:23: error: not found: value seq
Error occurred in an application involving default arguments.
       val rdd = sc.parallelize(seq(1,2,3,4,5))
                                ^

scala> val rdd = sc.parallelize(Seq(1,2,3,4,5))
rdd: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[2] at parallelize at <console>:23

scala> val sum = rdd.reduce(_+_)
sum: Int = 15

scala>
```

```
scala> val data = seq(1,2,3,4,5)
<console>:22: error: not found: value seq
       val data = seq(1,2,3,4,5)
                  ^

scala> val data = Seq(1,2,3,4,5)
data: Seq[Int] = List(1, 2, 3, 4, 5)

scala> val rdd =sc.parallelize(data)
rdd: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[0] at parallelize at <console>:24

scala> rdd.collect()
res0: Array[Int] = Array(1, 2, 3, 4, 5)

scala> res0:Array[int]=Array(1,2,3,4,5)
<console>:1: error: ';' expected but '=' found.
       res0:Array[int]=Array(1,2,3,4,5)
                      ^

scala> val evenNumber = rdd.filter(_%2==0)
evenNumber: org.apache.spark.rdd.RDD[Int] = MapPartitionsRDD[1] at filter at <console>:23

scala> evenNumber.collect()
res1: Array[Int] = Array(2, 4)

scala> val rdd = sc.parallelize(seq(1,2,3,4,5))
<console>:23: error: not found: value seq
Error occurred in an application involving default arguments.
       val rdd = sc.parallelize(seq(1,2,3,4,5))
                                ^

scala> val rdd = sc.parallelize(Seq(1,2,3,4,5))
rdd: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[2] at parallelize at <console>:23

scala> val sum = rdd.reduce(_+_)
sum: Int = 15

scala> val rdd = sc.parallelize(Seq(1,2,3,4,5))
rdd: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[3] at parallelize at <console>:23

scala> val divide =rdd.reduce(_/_)
divide: Int = 0

scala>
```

```
scala> rdd.collect()
res0: Array[Int] = Array(1, 2, 3, 4, 5)

scala> res0:Array[int]=Array(1,2,3,4,5)
<console>:1: error: ';' expected but '=' found.
       res0:Array[int]=Array(1,2,3,4,5)
                      ^

scala> val evenNumber = rdd.filter(_%2==0)
evenNumber: org.apache.spark.rdd.RDD[Int] = MapPartitionsRDD[1] at filter at <console>:23

scala> evenNumber.collect()
res1: Array[Int] = Array(2, 4)

scala> val rdd = sc.parallelize(seq(1,2,3,4,5))
<console>:23: error: not found: value seq
Error occurred in an application involving default arguments.
       val rdd = sc.parallelize(seq(1,2,3,4,5))
                                ^

scala> val rdd = sc.parallelize(Seq(1,2,3,4,5))
rdd: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[2] at parallelize at <console>:23

scala> val sum = rdd.reduce(_+_)
sum: Int = 15

scala> val rdd = sc.parallelize(Seq(1,2,3,4,5))
rdd: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[3] at parallelize at <console>:23

scala> val divide =rdd.reduce(_/_)
divide: Int = 0

scala> val rdd = sc.parallelize(Seq(1,2,3,4,5))
rdd: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[4] at parallelize at <console>:23

scala> val subtract =rdd.reduce(_-_)
subtract: Int = -13

scala> val rdd = sc.parallelize(Seq(1,2,3,4,5))
rdd: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[5] at parallelize at <console>:23

scala> val mul =rdd.reduce(_*_)
mul: Int = 120

scala>
```