Apache Spark with DataBricks on Azure

Trainer - Jeganathan Swaminathan <jegan@tektutor.org>

Updated on 16th Dec 2020

Lab Setup

Apache Spark Installation in CentOS 7.7.1908 (Core) 64-bit

First make sure you have logged in as root user

```
su -
```

As a root user, you may install JDK 1.8 as shown below

```
yum install -y epel-release
yum install -y java-1.8.0-openjdk.x86_64-devel
```

In case your CentOS box couldn't recognize the above java install package, you may try the below command

```
yum list available \*openjdk\*
```

Now look for the openjdk 1.8.0 that has "devel" in it, the devel packages will install JDK(Java Development Kit). JDK comes with java(interpreter - used to run Java applications) and javac(compiler - compile Java applications)

```
yum install -y "the package you copied from yum list goes here"
```

You may now download Scala as shown below

```
cd /root
wget https://scala-lang.org/files/archive/scala-2.13.4.tgz
tar xvf scala-2.13.4.tgz
mv scala-2.13.4 /usr/lib
ln -s /usr/lib/scala-2.13.4 /usr/lib/scala
export PATH=$PATH:/usr/lib/scala/bin
```

You may now check the version of Scala

```
scala -version
Scala code runner version 2.13.4 -- Copyright 2002-2020, LAMP/EPFL and Lightbend, Inc
```

You may now download Apache Spark as shown below

```
cd /root
wget https://archive.apache.org/dist/spark/spark-3.0.1/spark-3.0.1-bin-hadoop2.7.tgz
tar xvf spark-3.0.1-bin-hadoop2.7.tgz
cd spark-3.0.1-bin-hadoop2.7/
pwd
```

Add the below in /root/.bashrc

```
export SPARK_HOME=/root/spark-3.0.1-bin-hadoop2.7
export PATH=$PATH:/usr/lib/scala/bin:$SPARK_HOME/bin:$SPARK_HOME/sbin
source /root/.bashrc
```

Start Apache Spark

```
start-master.sh
```

Access Apache Spark page from web browser

http://172.16.124.243:8080/

You need to replace the above IP address with your CentOS IP address.

In case you have setup Apache Spark cluster on your office laptop/desktop which is behind proxy, you need to open up ports 6066, 7077, 8080 and 8081 as shown below.

```
firewall-cmd --permanent --zone=public --add-port=6066/tcp
firewall-cmd --permanent --zone=public --add-port=7077/tcp
firewall-cmd --permanent --zone=public --add-port=8080-8081/tcp
firewall-cmd --reload
```

In case you have setup the Apache Spark cluster on the Azure/AWS Cloud, then you need to open up the above ports in the Networking Inbound ports section in case of Azure. In case of AWS cloud VM, you need to open up the ports in the Security Group Inbound ports.

Installing Apache Maven

Apache Maven is a build tool that helps in compiling, packing Java applications and managing its application dependencies. This is required in case you are planning to write the Apache Spark Driver application in Java, otherwise this can be ignored.

```
cd /root
```

```
wget
https://mirrors.estointernet.in/apache/maven/maven-3/3.6.3/binaries/apache-maven-3.6.3
-bin.zip
```

```
unzip apache-maven-3.6.3-bin.zip
```

Installing Git version control software

Though it isn't mandatory to install this software for Apache Spark, I would strongly encourage you to install this as you may clone my GitHub to get the lab exercises in case you wish to download the lab exercises code, otherwise it is completely optional.

```
yum install -y git
```

Installing Unix tree utility

This is a handy tool to display the folder structure, this is optional but a very useful one.

```
yum install -y tree
```

Installing Simple Build Tool (SBT)

SBT is the build tool used to package Scala applications as jar files. The packaged jar file can be executed in Apache Spark cluster. Hence we need this build tool.

```
cd /root
wget https://github.com/sbt/sbt/releases/download/v1.4.5/sbt-1.4.5.zip
unzip sbt-1.4.5.zip
```

Add the below exports into /root/.bashrc

```
export SPARK_HOME=/root/spark-3.0.1-bin-hadoop2.7
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.272.b10-1.el7_9.x86_64
export M2_HOME=/root/apache-maven-3.6.3
export
PATH=$PATH:/usr/lib/scala/bin:$SPARK_HOME/bin:$SPARK_HOME/sbin:$M2_HOME/bin:/root/sbt/
bin
```

To apply the environment settings without restarting Linux terminal, you may try

```
source /root/.bashrc
```

You may check if all the tools are in path now

```
javac -version
java -version
mvn -version
scala --version
```

Installing MySQL Server 5.7 on CentOS

```
wget https://dev.mysql.com/get/mysql57-community-release-el7-9.noarch.rpm
sudo rpm -ivh mysql57-community-release-el7-9.noarch.rpm
sudo yum install mysql-server
sudo systemctl enable mysqld
sudo systemctl start mysqld
sudo systemctl status mysqld
```

Find the temporary password that was generated during the MySQL Server installation

```
grep 'temporary password' /var/log/mysqld.log
```

Change the root password with your preferred password

```
sudo mysql_secure_installation
```

I used Root@123 as the password.

```
Estimated strength of the password: 100
```

```
Change the password for root ? ((Press y|Y for Yes, any other key for No) : No

Remove anonymous users? (Press y|Y for Yes, any other key for No) : No

Disallow root login remotely? (Press y|Y for Yes, any other key for No) : Y
Success.
 ... skipping.

Remove test database and access to it? (Press y|Y for Yes, any other key for No) : Y
 - Dropping test database...
Success.

Reload privilege tables now? (Press y|Y for Yes, any other key for No) : Y
Success.
```

You may now try to login to mysql server via mysql client as shown below

```
mysql -u root -p
```

When prompted for password, type the password you had set, in my case it is Root@123

mysql > CREATE DATABASE tektutor;

mysql > USE tektutor;

mysql > CREATE TABLE books(id int NOT NULL, title VARCHAR(50), fname VARCHAR(25), lname VARCHAR(25), author VARCHAR(100), PRIMARY KEY (id));

mysql> SHOW TABLES;

mysql > DESCRIBE books;

Testing your Apache Spark Setup

In your /root directory, create a Test folder, the name of the folder could be anything, hence it isn't mandatory to name it as Test, feel free to name the folder to whatever you wish.

```
mkdir Test
cd Test
mkdir -p src/main/scala
touch build.sbt
```

Type the below into build.sbt file and save it

```
name := "Simple Project"
version := "1.0"
```

```
scalaVersion := "2.12.10"
libraryDependencies += "org.apache.spark" %% "spark-sql" % "3.0.1"
```

Now create a file by name SimpleApp.scala under src/main/scala/

```
/* SimpleApp.scala */
import org.apache.spark.sql.SparkSession

object SimpleApp {
  def main(args: Array[String]) {
    val logFile = "YOUR_SPARK_HOME/README.md" // Should be some file on your system
    val spark = SparkSession.builder.appName("Simple Application").getOrCreate()
    val logData = spark.read.textFile(logFile).cache()
    val numAs = logData.filter(line => line.contains("a")).count()
    val numBs = logData.filter(line => line.contains("b")).count()
    println(s"Lines with a: $numAs, Lines with b: $numBs")
    spark.stop()
  }
}
```

Now, you may trying creating a jar file for your project as shown below
From the /root/Test folder, try the below command

```
sbt package
```

```
If this is the first time you are issuing the above command it may take a while to
download all the dependencies, so wait until it completes.
```

Output

```
[root@localhost Test]# sbt package
[info] welcome to sbt 1.4.5 (Red Hat, Inc. Java 1.8.0_272)
[info] loading project definition from /root/Test/project
[info] loading settings for project test from build.sbt ...
[info] set current project to Simple Project (in build file:/root/Test/)
[success] Total time: 2 s, completed Dec 15, 2020 9:48:38 AM
```

Now you may navigate to /root/Test/target/scala-2.12 to locate simple-project_2.12-1.0.jar file.

Before executing the Scala Spark Driver application, you need to copy some README.md file

```
cp /root/spark-2.4.7-bin-hadoop2.7/README.md .
```

spark-submit --class "SimpleApp" --master local simple-project_2.12-1.0.jar

```
20/12/15 09:58:20 INFO DAGScheduler: ResultStage 3 (count at SimpleApp.scala:10)
finished in 0.024 s
```

```
20/12/15 09:58:20 INFO DAGScheduler: Job 1 is finished. Cancelling potential
speculative or zombie tasks for this job
20/12/15 09:58:20 INFO TaskSchedulerImpl: Killing all running tasks in stage 3: Stage
finished
20/12/15 09:58:20 INFO DAGScheduler: Job 1 finished: count at SimpleApp.scala:10, took
0.074915 s
Lines with a: 64, Lines with b: 32
20/12/15 09:58:20 INFO SparkUI: Stopped Spark web UI at http://172.16.124.213:4040
20/12/15 09:58:20 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint
stopped!
20/12/15 09:58:20 INFO MemoryStore: MemoryStore cleared
20/12/15 09:58:20 INFO BlockManager: BlockManager stopped
20/12/15 09:58:20 INFO BlockManagerMaster: BlockManagerMaster stopped
20/12/15 09:58:20 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint:
OutputCommitCoordinator stopped!
20/12/15 09:58:20 INFO SparkContext: Successfully stopped SparkContext
20/12/15 09:58:20 INFO ShutdownHookManager: Shutdown hook called
20/12/15 09:58:20 INFO ShutdownHookManager: Deleting directory
/tmp/spark-c8bb261e-7573-41e7-a78a-64384a87d51f
20/12/15 09:58:20 INFO ShutdownHookManager: Deleting directory
/tmp/spark-79a2dfa2-6ea1-4061-bf09-1680f7939693
[root@localhost scala-2.12]#
```

The line highlighted above is the expected output.

Now you are all set, Congrats !