Multi Node Spark Cluster Set up

1. Create a multi node fyre cluster with 1 master and 2 slaves

OS -redhat8.5

Graphical user interface, website

Description automatically generated

1. Add entries in hosts file (master and slaves)

vi /etc/hosts/

Graphical user interface, text

Description automatically generated

1. Install java 8 on master and slaves

jdk1.8.0\_202

check java version

Text

Description automatically generated

1. Install scala on master and slaves

yum install scala

Text

Description automatically generated

5 - Check by SSH to all the slaves from master( it should be loggedin without asking password)

ssh slave1

ssh slave2

6 – Download apache spark 3.2.1 in master and slaves

wget <https://dlcdn.apache.org/spark/spark-3.2.1/spark-3.2.1-bin-hadoop3.2.tgz>

untar the spark-3.2.1-bin-hadoop3.2.tgz

tar -zxvf spark-3.2.1-bin-hadoop3.2.tgz

Use the following command to move the spark software files to respective directory

mv spark-3.2.1-bin-hadoop3.2.tgz /root/spark

**Set up the environment for Spark in master and slaves**

Edit *bash\_profile* file and add env variable for SPARK\_HOME ,JAVA\_HOME,HADOOP\_HOME

vi ~/.bash\_profile

Text

Description automatically generated

Use the following command for sourcing the *~/.bash\_profile* file.

source ~/.bash\_profile

7- Download Hadoop 3.3.2 in Master and Slaves

wget https://www.apache.org/dyn/closer.cgi/hadoop/common/hadoop-3.3.2/hadoop-3.3.2.tar.gz

tar -zxvf hadoop-3.3.2.tar.gz

8- **Spark Master Configuration**

Do the following procedures only in master.

Move to spark *conf* (root/spark/conf)folder and create a copy of template(spark-env.sh.template) to  *spark-env.sh* and rename it.

cp spark-env.sh.template spark-env.sh

Now edit the configuration file *spark-env.sh.*

$ vi spark-env.sh

Add below given entries in spark-env.sh

export SPARK\_MASTER\_HOST='<MASTER-IP>'

export JAVA\_HOME=<Path\_of\_JAVA\_installation>

Graphical user interface, text

Description automatically generated

Create log4j.properties file in */root/spark/con*f/ by copying log4j.properties.template

cp log4j.properties.template log4j.properties

and edit log path as given below

log4j.appender.file.File=/root/spark/logs/job.log

Text

Description automatically generated

## 9-Add Workers on Slave Nodes

Edit the configuration file on Master & *slaves* in (*/root/spark/con*f)

Create file slaves by copying slaves.template

cp slaves.template slaves

ON MASTER

Text

Description automatically generated

ON SLAVES

Text

Description automatically generated

**Start Spark Cluster**

To start the spark cluster, run the following command on master.

$ cd /root/spark/sbin/  
$ ./start-all.sh

**Check whether services have been started**

To check daemons on master and slaves, use the following command.

$ jps

A picture containing text

Description automatically generated

To stop the spark cluster, run the following command on master.

$ cd /root/spark/sbin  
$ ./stop-all.sh

**Spark Web UI**

Browse the Spark UI to know about worker nodes, running application, cluster resources.

http://<MASTER-IP>:8080/

Graphical user interface

Description automatically generated with medium confidence

**Spark Application UI**

http://<MASTER\_IP>:4040/

Table

Description automatically generated

TO Run spark application

spark-shell --master spark://9.21.106.140:7077 ( masterip:7077)

Text

Description automatically generated

To check spark-shell is working

Text

Description automatically generated

**To check connector use case follow below steps**

Clone <https://github.ibm.com/dataconnect/wdp-connect-library-download>

Build this project by updating wdp.connect.library.version in gradle.propeties

Graphical user interface, text, application, email

Description automatically generated

Copy the builded jar to Master and Slaves Nodes in root folder

Edit spark-defaults.conf in /root/spark/conf/ in Master and slaves and add below entries for referring latest wdp-connect-library-download jars

spark.jars /root/wdp-connect-library-download/\*.jar

Text

Description automatically generated

Also download and add below given jars to wdp-connect-library-download folder or spark/jars/ to fix some runtime errors

spark-sql\_2.12-3.2.1.jar

spark-core\_2.11-1.5.2.logging.jar

spark-core\_2.11-2.4.8.jar

gson-2.8.9.jar

Removed gson-2.2.4jar to avoid duplicate

To Test Connector use case

val readOptions=Map(

"type"->"match360",

"connection.crn"->"crn:v1:staging:public:mdm-oc:us-south:a/1495b2f71a6f42ca8e13e92090495609:9b670d47-def6-4c54-8f8d-5da2ca707c55::",

"connection.iam\_url"->"[https://iam.test.cloud.ibm.com](https://iam.test.cloud.ibm.com/)",

"connection.gateway\_url"->"[https://api.dataplatform.dev.cloud.ibm.com](https://api.dataplatform.dev.cloud.ibm.com/)",

"connection.api\_key"->"oytgdOJ68IKS9FToHLqpTOEqFYu3wWMCi\_7KoJ7qYz\_x",

"source.row\_limit"->"100",

"source.record\_type\_name"->"person",

"source.record\_sub\_type\_name"->"person\_record",

"num\_partitions"->"1",

"metrics\_ref"->"test");

val dfReader = spark.sqlContext.read.format("com.ibm.connect.spark").options(readOptions)

val df = dfReader.load

df.printSchema

df.show

df.count

df.rdd.getNumPartitions

Logs can be found at /root/spark/logs/

Reference:

<https://medium.com/ymedialabs-innovation/apache-spark-on-a-multi-node-cluster-b75967c8cb2b>

<https://github.ibm.com/dataconnect/wdp-connect-service/wiki/Interactively-test-connectors-with-a-local-Spark-cluster>