Инструкция по развертыванию Apache Spark под управлением YARN и подключению к кластеру HDFS

Мои данные:

Узел для входа 176.109.91.28

Jump node 192.168.1.118

Name node 192.168.1.119

Data node-00 192.168.1.120

Data node-01 192.168.1.121

ssh team@176.109.91.28 -- Заходим на jump-ноду

sudo -i -u hadoop -- Меняем пользователя на hadoop

ssh nn

sudo apt install python3-venv -- Устанавливаем модуль для создания виртуальных окружений

sudo apt install python3-pip -- Устанавливаем pip

wget https://archive.apache.org/dist/spark/spark-3.5.3/spark-3.5.3-bin-hadoop3.tgz -- Скачиваем Apache Spark

tar -xzvf spark-3.5.3-bin-hadoop3.tgz

Указываем необходимые пути:

export HADOOP_CONF_DIR="/home/hadoop/hadoop-3.4.0/etc/hadoop"

export HIVE_HOME="/home/hadoop/apache-hive-4.0.1-bin"

export HIVE_CONF_DIR=$HIVE_HOME/conf

export HIVE_AUX_JARS_PATH=$HIVE_HOME/lib/*

export PATH=$PATH:$HIVE_HOME/bin

export SPARK_LOCAL_IP=192.168.1.118

$ export SPARK_DIST_CLASSPATH="/home/hadoop/spark-3.5.3-bin-hadoop3/jars/*:/home/hadoop/hadoop-3.4.0/etc/hadoop:/home/hadoop/hadoop-3.4.0/share/hadoop/common/lib/*:/home/hadoop/hadoop3.4.0/share/hadoop/common/*:/home/hadoop/hadoop-3.4.0/share/hadoop/hdfs: /home/hadoop/hadoop-3.4.0/share/hadoop/hdfs/Lib/*:/home/hadoop/hadoop-3.4.0/share/hadoop/hdfs/*:/home/hadoop/hadoop-3.4.0/share/hadoop/mapreduce/*:/home/hadoop/hadoop-3.4.0/share/hadoop/yarn:/home/hadoop/hadoop-3.4.0/share/hadoop/yarn/Lib/*:/home/hadoop/hadoop-3.4.0/share/hadoop/yarn/*:/home/hadoop/apache-hive-4.0.0-alpha-2-bin/*:/home/hadoop/apache-hive-4.0.0-alpha-2-bin/lib/*"

cd spark-3.5.3-bin-hadoop3/

export SPARK_HOME='pwd' -- Устанавливаем переменную SPARK_HOME в текущую директорию

export PYTHONPATH=$(ZIPS=("$SPARK_HOME"/python/lib/*.zip); IFS=:; echo "$(ZIPS[*]}"): $PYTHONPATH -- Добавляем библиотеки PySpark в PYTHONPATH

export PATH=$SPARK_HOME/bin:$PATH -- Добавляем Spark в PATH

cd ../

python3 –m venv venv -- Создаем виртуальное окружение

source venv/bin/activate -- Активируем виртуальное окружение

hive --hiveconf hive.server2.enable.doAs=false --hiveconf hive.security.authorization.enable=false --service metastore -- Запускаем Hive Metastore

pip install –U pip -- Обновляем pip

pip install ipython -- Устанавливаем ipython

pip install onetl[files] -- Устанавливаем библиотеку onetl

ipython -- запускаем оболочку python

```
from pyspark.sql import SparkSession

from pyspark.sql import function as F

from onetl.connection import SparkHDFS

from onetl.connection import Hive

from onetl.file import FileDFReader

from onetl.file.format import CSV

from onetl.db import DBWriter


spark = SparkSession.builder \

        .master("yarn") \

        .appName("spark-with-yarn") \

        .config("spark.sql.warehouse.dir", "/user/hive/warehouse") \

        .config("spark.hive.metastore.uris", "thrift://jn:5432") \

        .enableHiveSupport() \

        .getOrCreate()
```

```
hdfs = SparkHDFS(host="nn", port=9000, spark=spark, cluster="test")

reader = FileDFReader(connection=hdfs, format=CSV(delimiter=",", header=True),
source_path="/input")

df = reader.run(["for_spark.csv"])
```

Применяем трансформации данных

```
df = df.withColumn(df.columns[0]  , F.upper(F.col(df.columns[0] )))  -- Например применяем
```
функцию upper к первой колонке

Применяем партиционирование при сохранении данных

```
df.write \

   .mode("overwrite") \

   .saveAsTable("test.spark_partitions")
```

Проверяем возможность чтения данных стандартным клиентом Hive

```
hive = Hive(spark=spark, cluster="test")

df_check = spark.sql("SELECT * FROM test.spark_partitions")

df_check.show()

spark.stop()  -- Завершаем работу Spark
```