Инструкция по обработке данных

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

Указываем необходимые пути:

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

export HIVE_HOME=”/home/hadoop/apache-hive-4.0.0-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=”/home/hadoop/spark-3.5.3-bin-hadoop3/”

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

python3 –m venv .venv -- Создаем виртуальное окружение

source .venv/bin/activate -- Активируем виртуальное окружение

pip install -U pip -- обновляем pip

pip install ipython

ipython -- запускаем интерактивную оболочку

%logstart

```
from pyspark.sql import SparkSession
from onetl.connection import SparkHDFS
from onetl.file import FileDFReader
from onetl.connection import Hive
from onetl.file.format import CSV
from onetl.db import DBWriter
from pyspark.sql import function as F
spark = SparkSession.builder \
        .master("yarn") \
        .appName("spark-with-yarn") \
        .config("spark.sql.warehouse.dir", "/user/hive/warehouse") \
        .config("spark.hive.metastore.uris", "thrift://jn:9083") \
        .enableHiveSupport() \
        .getOrCreate()
hdfs = SparkHDFS(host="nn", port=9000, spark=spark, cluster="test")
reader = FileDFReader(connection=hdfs, format=CSV(delimiter="\t", header=True), source_path="/input")
df = reader.run(["for_spark.csv"])
df.count() – считаем количество строчек
df.printSchema() – смотрим список полей и иипов данных
dt = df.select("registration date")
dt.show()
df = df.withColumn("reg_year", F.col("registration date").substr(0, 4))
dt = df.select("reg_year")
dt.show() – получили столбец в котором содержится год
df.rdd.getNumPartitions() – смотрим сколько в датафрейме партиций
hive = Hive(spark=spark, cluster="test")
```

```
hive.check()

Hive(cluster='test')

writer = DBWriter(connection=hive, table="test.spark_parts", options={"if_exists": "replace_entire_table"})

writer.run(df)

writer = DBWriter(connection=hive, table="test.one_parts", options={"if_exists": "replace_entire_table"})

writer.run(df.coalesce(1)) – делаем из всех партиций одну

writer = DBWriter(connection=hive, table="test.hive_parts", options={"if_exists": "replace_entire_table", "partitionBy":"reg_year"})

writer.run(df)
```

%logstop

spark.stop()

quit()

beeline –u jdbc:hive2://jn:5432 –n scott –p tiger

pip install prefect

vim ipython_log.py -- превращаем файл в скрипт prefect:

```
from pyspark.sql import SparkSession

from onetl.connection import SparkHDFS

from onetl.file import FileDFReader

from onetl.connection import Hive

from onetl.file.format import CSV

from onetl.db import DBWriter

from pyspark.sql import function as F


from prefect import flow, task

@task

def get_spark():

        spark = SparkSession.builder \
```

```
        .master("yarn") \
        .appName("spark-with-yarn") \
        .config("spark.sql.warehouse.dir", "/user/hive/warehouse") \
        .config("spark.hive.metastore.uris", "thrift://jn:9083") \
        .enableHiveSupport() \
        .getOrCreate()
    return spark


@task
def stop_spark(spark):
    spark.stop()


@task
def extract(spark):
    hdfs = SparkHDFS(host="nn", port=9000, spark=spark, cluster="test")
    reader = FileDFReader(connection=hdfs, format=CSV(delimiter="\t", header=True), source_path="/input")
    df = reader.run(["for_spark.csv"])
    return df


@task
def transform(df):
    df = df.withColumn("reg_year", F.col("registration date").substr(0, 4))
    return df


@task
def load(spark, df):
    hive = Hive(spark=spark, cluster="test")
    writer = DBWriter(connection=hive, table="test.hive_parts", options={"if_exists": "replace_entire_table", "partitionBy":"reg_year"})
```

```
        writer.run(df)


@flow
def process_data():
        spark = get_spark()
        df = extract(spark)
        df = transform(df)
        load(spark, df)
        stop_spark(spark)
if __name__ == “__main__”:
        process_data()
```

Сохраняем файл и выходим из редактора.

python ipython_log.py