import java.util.Properties

import com.bingocloud.{ClientConfiguration, Protocol}

import com.bingocloud.auth.BasicAWSCredentials

import com.bingocloud.services.s3.AmazonS3Client

import org.apache.kafka.clients.producer.{KafkaProducer, ProducerRecord}

import org.nlpcn.commons.lang.util.IOUtil

import java.sql.{Connection, DriverManager, ResultSet}

import com.bingocloud.util.json.JSONException

import org.json.JSONArray

import org.json.JSONObject

import java.sql.ResultSet

import java.sql.SQLException

import com.google.gson.{JsonArray, JsonObject}

object Main3 {

//s3参数

val accessKey = "B295DE50D353418FD1F6"

val secretKey = "Wzg1NkVEQjRGMEIwQTRGRTIxM0NDQzgxQjAwQjFGNDg4M0I0NjU5MkVd"

val endpoint = "scuts3.depts.bingosoft.net:29999"

val bucket = "chenchaoyu"

//要读取的文件

val key = "demo.txt"

//kafka参数

val topic = "chenchaoyu"

val bootstrapServers = "bigdata28.depts.bingosoft.net:23307"

val username = "root"

val password = "ccy123"

//val drive = "com.mysql.jdbc.Driver"

//val url = "jdbc:mysql://localhost:3306/mysql"

var driver="com.mysql.cj.jdbc.Driver"

var url="jdbc:mysql://localhost:3306/mysql?serverTimezone=UTC"

var connection: Connection = null

def main(args: Array[String]): Unit = {

//val s3Content = readFile()

var mysqlContent=getDataFromMysql()

produceToKafka(mysqlContent)

}

@throws[SQLException]

@throws[JSONException]

def resultSetToJson(rs: ResultSet): String = { // json数组

val array = new JSONArray()

// 获取列数

val metaData = rs.getMetaData

val columnCount = metaData.getColumnCount

// 遍历ResultSet中的每条数据

while ( {

rs.next

}) {

val jsonObj = new JSONObject()

// 遍历每一列

for (i <- 1 to columnCount) {

val columnName = metaData.getColumnLabel(i)

val value = rs.getString(columnName)

jsonObj.put(columnName, value)

}

array.put(jsonObj)

}

array.toString

}

def getDataFromMysql():String={

try {

//在spark中如果不写会出错

classOf[com.mysql.jdbc.Driver]

connection = DriverManager.getConnection(url, username, password)

val statement = connection.createStatement()

statement.executeQuery("use test")

val resultSet = statement.executeQuery("select \* from table1")

var jsonResult=resultSetToJson(resultSet)

return jsonResult

} catch {

case e: Exception=> e.printStackTrace()

return ""

} finally {

connection.close()

}

}

/\*\*

\* 从s3中读取文件内容

\*

\* @return s3的文件内容

\*/

def readFile(): String = {

val credentials = new BasicAWSCredentials(accessKey, secretKey)

val clientConfig = new ClientConfiguration()

clientConfig.setProtocol(Protocol.HTTP)

val amazonS3 = new AmazonS3Client(credentials, clientConfig)

amazonS3.setEndpoint(endpoint)

val s3Object = amazonS3.getObject(bucket, key)

IOUtil.getContent(s3Object.getObjectContent, "UTF-8")

}

/\*\*

\* 把数据写入到kafka中

\*

\* @param s3Content 要写入的内容

\*/

def produceToKafka(s3Content: String): Unit = {

val props = new Properties

props.put("bootstrap.servers", bootstrapServers)

props.put("acks", "all")

props.put("key.serializer", "org.apache.kafka.common.serialization.StringSerializer")

props.put("value.serializer", "org.apache.kafka.common.serialization.StringSerializer")

val producer = new KafkaProducer[String, String](props)

val dataArr = s3Content.split("\n")

for (s <- dataArr) {

if (!s.trim.isEmpty) {

val record = new ProducerRecord[String, String](topic, null, s)

println("开始生产数据：" + s)

producer.send(record)

}

}

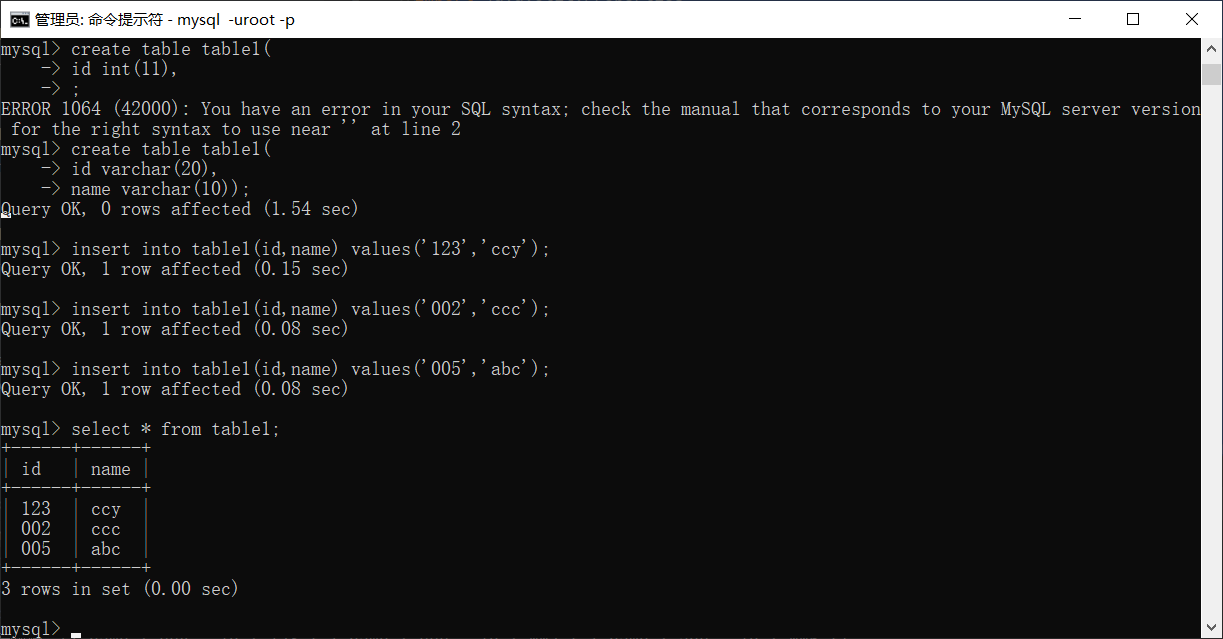
producer.flush()

producer.close()

}

}

Mysql中内容



结果：

