

# Practica Spark y Kafka Streaming

## 1. Código productor:

Los comandos necesarios para ejecutar bin/kafka-console-producer.sh dentro de kafka se han recopilado en el archivo de comandos bdp\_producer.sh, cuyo contenido es el que se muestra a continuación:

```
#Nos trasladamos a la ruta donde esta instalado kafka
cd /home/kafka/kafka_2.11-2.4.0/bin
#Crear el topic
./kafka-topics.sh --create --zookeeper localhost:2181 --replication-
factor 1 --partitions 1 --topic practicaBDP
#Listamos los topics para verificar que el topic fue creado
./kafka-topics.sh --list --zookeeper localhost:2181
#Copiamos el archivo a enviar al productor
cp /home/keepcoding/Descargas/personal.json .
ls
cd ..
#Ejecutar el productor en el topic creado
cat bin/personal.json | bin/kafka-console-producer.sh --broker-list
localhost:9092 --topic practicaBDP

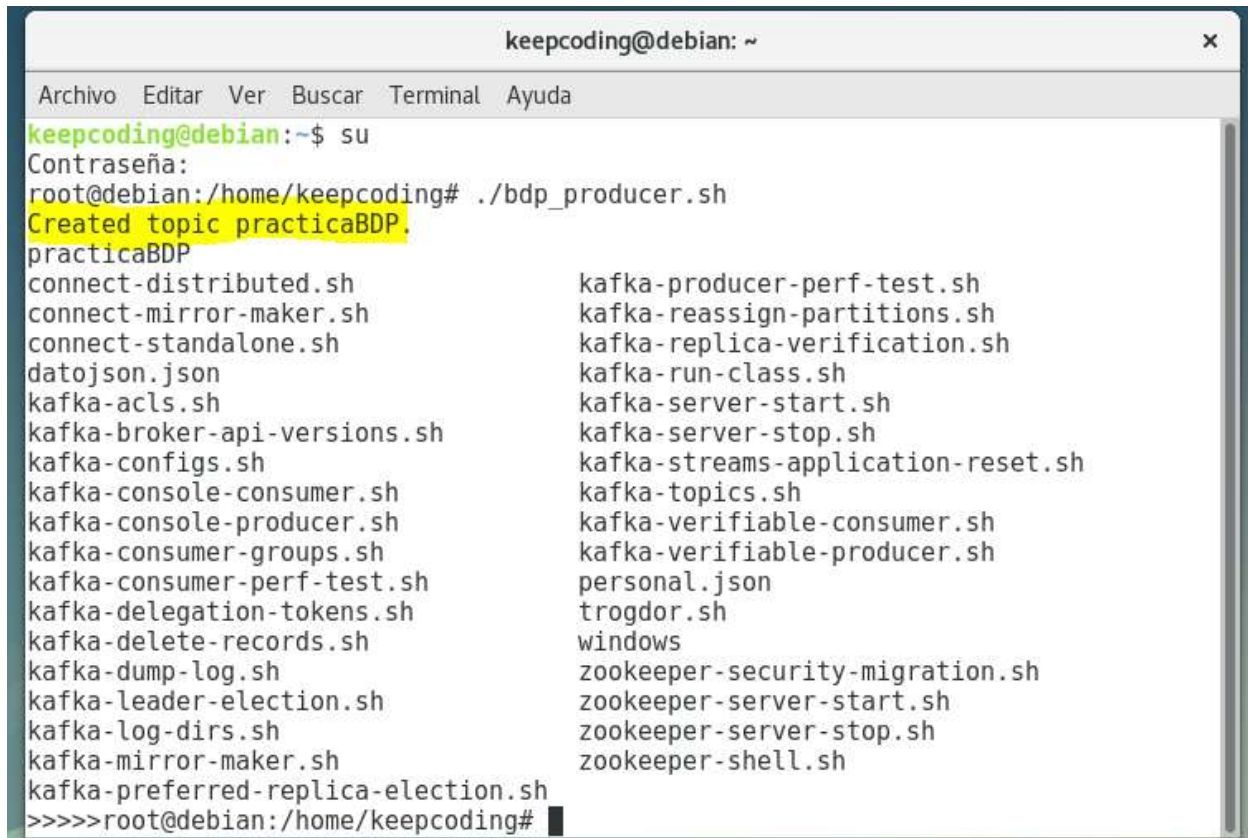
#Codigo para eliminar el topic (no se ejecuta)
#./kafka-topics.sh --zookeeper localhost:2181 --delete --topic
practicaBDP
```

## 2. Código consumidor:

Para verificar que el script del productor ha funcionado, ejecutamos via consola el script bin/kafka-console-consumer.sh, y al igual que se creo un script combinado del productor, tambien tenemos el archivo de comandos bdp\_consumer.sh con las siguientes intrucciones:

```
#Nos trasladamos a la ruta donde esta instalado kafka
cd /home/kafka/kafka_2.11-2.4.0
#Ejecutar el consumidor para leer el topic creado
bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --
topic practicaBDP --from-beginning
```

Al ejecutar los scripts tanto del productor como del consumidor desde el terminal tenemos estos resultados, los cuales son satisfactorios:

A terminal window titled 'keepcoding@debian: ~' with a menu bar (Archivo, Editar, Ver, Buscar, Terminal, Ayuda). The user runs 'su' and then './bdp\_producer.sh'. The output lists various Kafka scripts and files. The text 'Created topic practicaBDP.' is highlighted in yellow. The prompt changes from root@debian to root@debian:/home/keepcoding.

```
keepcoding@debian:~$ su
Contraseña:
root@debian:/home/keepcoding# ./bdp_producer.sh
Created topic practicaBDP.
practicaBDP
connect-distributed.sh          kafka-producer-perf-test.sh
connect-mirror-maker.sh       kafka-reassign-partitions.sh
connect-standalone.sh         kafka-replica-verification.sh
datojson.json                 kafka-run-class.sh
kafka-acls.sh                 kafka-server-start.sh
kafka-broker-api-versions.sh  kafka-server-stop.sh
kafka-configs.sh             kafka-streams-application-reset.sh
kafka-console-consumer.sh     kafka-topics.sh
kafka-console-producer.sh     kafka-verifiable-consumer.sh
kafka-consumer-groups.sh     kafka-verifiable-producer.sh
kafka-consumer-perf-test.sh   personal.json
kafka-delegation-tokens.sh    trogdor.sh
kafka-delete-records.sh       windows
kafka-dump-log.sh             zookeeper-security-migration.sh
kafka-leader-election.sh      zookeeper-server-start.sh
kafka-log-dirs.sh             zookeeper-server-stop.sh
kafka-mirror-maker.sh         zookeeper-shell.sh
kafka-preferred-replica-election.sh
>>>>root@debian:/home/keepcoding#
```

```
keepcoding@debian: ~  
Archivo Editar Ver Buscar Terminal Ayuda  
keepcoding@debian:~$ su  
Contraseña:  
root@debian:/home/keepcoding# ./bdp_consumer.sh  
{ "id":1, "first_name": "Jeanette", "last_name": "Penddreth", "email": "jpenddreth0@census.gov", "gender": "Female", "ip_address": "26.58.193.2" },  
{ "id":2, "first_name": "Giavani", "last_name": "Frediani", "email": "gfredianil@senate.gov", "gender": "Male", "ip_address": "229.179.4.212" },  
{ "id":3, "first_name": "Noell", "last_name": "Bea", "email": "nbea2@imageshack.us", "gender": "Female", "ip_address": "180.66.162.255" },  
{ "id":4, "first_name": "Willard", "last_name": "Valek", "email": "wvalek3@vk.com", "gender": "Male", "ip_address": "67.76.188.26" }
```

Posteriormente, consumimos el archivo desde kafka con el objeto scala

Practica\_ConsumidorJSON.scala:

```
import org.apache.spark.sql.SparkSession  
import org.apache.spark.sql.types.{IntegerType, StringType, StructType}  
import org.apache.spark.sql.functions.{from_json,col}  
  
object Practica_ConsumidorJSON {  
  def main(args: Array[String]): Unit = {  
    //declarar la app  
    val  
    spark=SparkSession.builder().appName("JsonKafkaConsumer").master("local").getOrCreate()  
  
    //Configurar stream de entrada  
    val df=spark.readStream  
      .format("kafka")  
      .option("kafka.bootstrap.servers", "localhost.localdomain:9092")  
      .option("subscribe", "practicaBDP")  
      .option("startingOffsets", "earliest")  
      .load()
```

```

// castear los datos leídos en formato kafka para convertirlos en
strings
val res=df.selectExpr("CAST(value AS STRING) as value")

//Declarar la estructura de los datos
val schema= new StructType()
    .add("id", IntegerType)
    .add("first_name", StringType)
    .add("last_name", StringType)
    .add("email", StringType)
    .add("gender", StringType)
    .add("ip_address", StringType)

//Seleccionar y filtrar dos los rows
val persona=res.select(from_json(col("value") , schema).as ("data"))
    .select ("data.*")
    .filter("first_name not like \"Giavani\"")
    .filter("first_name not like \"Noell\"")

//Mostrar los datos ledios del streaming
println("mostrar los datos por consola")

persona.writeStream
    .format("console")
    .outputMode("append")
    .start()
    .awaitTermination(60000)
}
}

```

Al ejecutar el script sin las sentencias filter podemos ver el contenido del archivo personal.json enviado por el productor

BDProcessing [~/IdeaProjects/BDProcessing] - .../src/main/scala/Practica\_ConsumidorJSON.scala [BDProcessing] x

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

BDProcessing > src > main > scala > Prac > Practica\_ConsumidorJSON

Practica\_ConsumidorJSON.scala

```

30 //Seleccionar y filtrar dos los rows
31 val persona=res.select(from_json(col( colName = "value" ), schema).as ( alias = "data"))
32   .select ( col = "data.*")
33
34 //Mostrar los datos ledios del streaming
35 println("mostrar los datos por consola")
36
37 persona.writeStream
38   .format( source = "console")
39   .outputMode( outputMode = "append")
40   .start()

```

Practica\_ConsumidorJSON > main(args: Array[String])

Run: Practica\_ConsumidorJSON

Batch: 0

```

20/02/08 23:21:06 INFO CodeGenerator: Code generated in 56.134647 ms
20/02/08 23:21:07 INFO CodeGenerator: Code generated in 67.602987 ms
+-----+-----+-----+-----+-----+-----+
| id|first_name|last_name|email|gender|ip_address|
+-----+-----+-----+-----+-----+-----+
| 1| Jeanette|Penddreth|jpenddreth0@censu...|Female|26.58.193.2|
| 2| Giavani|Frediani|gfrediani1@senate...|Male|229.179.4.212|
| 3| Noell|Bea|nbea2@imageshack.us|Female|180.66.162.255|
| 4| Willard|Valek|wvalek3@vk.com|Male|67.76.188.26|
+-----+-----+-----+-----+-----+-----+

```

```

20/02/08 23:21:07 INFO WriteToDataSourceV2Exec: Data source writer org.apache.spark.sql.execution.str
20/02/08 23:21:07 INFO SparkContext: Starting job: start at Practica_ConsumidorJSON.scala:40
20/02/08 23:21:07 INFO DAGScheduler: Job 1 finished: start at Practica_ConsumidorJSON.scala:40, took
20/02/08 23:21:07 INFO CheckpointFileManager: Writing atomically to file:/tmp/temporary-3306c066-7b88
20/02/08 23:21:07 INFO CheckpointFileManager: Renamed temp file file:/tmp/temporary-3306c066-7b88-461
20/02/08 23:21:07 INFO MicroBatchExecution: Streaming query made progress: {
  "id" : "9f04c91c-ef29-481f-b385-0189ad3bef03",
  "runId" : "1cd7d169-01f6-4560-bed7-6f185714d48f",
  "name" : null,
  "timestamp" : "2020-02-08T22:21:02.581Z",
  "batchId" : 0,
  "numInputRows" : 4,

```

4: Run 6: TODO sbt shell Terminal Event Log



Y finalmente al ejecutar con las sentencias filter excluimos dos registros diferentes segun las indicaciones de la practica:

The screenshot shows an IDE window titled "BDProcessing [~/IdeaProjects/BDProcessing] - .../src/main/scala/Practica\_ConsumidorJSON.scala [BDProcessing]". The code in the editor is as follows:

```

27 .add( name = "gender",StringType)
28 .add( name = "ip_address",StringType)
29
30 //Seleccionar y filtrar dos los rows
31 val persona=res.select(from_json(col( colName = "value" ),schema).as ( alias = "data"))
32 .select ( col = "data.*")
33 .filter( conditionExpr = "first_name not like \"Giavani\"" )
34 .filter( conditionExpr = "first_name not like \"Noell\"" )
35
36 //Mostrar los datos ledios del streaming
37 println("mostrar los datos por consola")

```

The Run console shows the following output:

```

Run: Practica_ConsumidorJSON x
20/02/08 23:23:32 INFO DAGScheduler: Job 0 finished: start at Practica_ConsumidorJSON.scala:42, took
20/02/08 23:23:32 INFO WriteToDataSourceV2Exec: Data source writer org.apache.spark.sql.execution.str
-----
Batch: 0
-----
20/02/08 23:23:32 INFO CodeGenerator: Code generated in 32.265734 ms
20/02/08 23:23:32 INFO CodeGenerator: Code generated in 32.892089 ms
+-----+-----+-----+-----+-----+-----+
| id|first_name|last_name|          email|gender| ip_address|
+-----+-----+-----+-----+-----+-----+
| 1| Jeanette|Penddreth|jpenddreth0@censu...|Female| 26.58.193.2|
| 4| Willard| Valek|      wvalek3@vk.com| Male|67.76.188.26|
+-----+-----+-----+-----+-----+-----+

```

The bottom of the console shows additional logs from SparkContext and DAGScheduler, indicating the job is finished and the streaming query progress is being tracked.