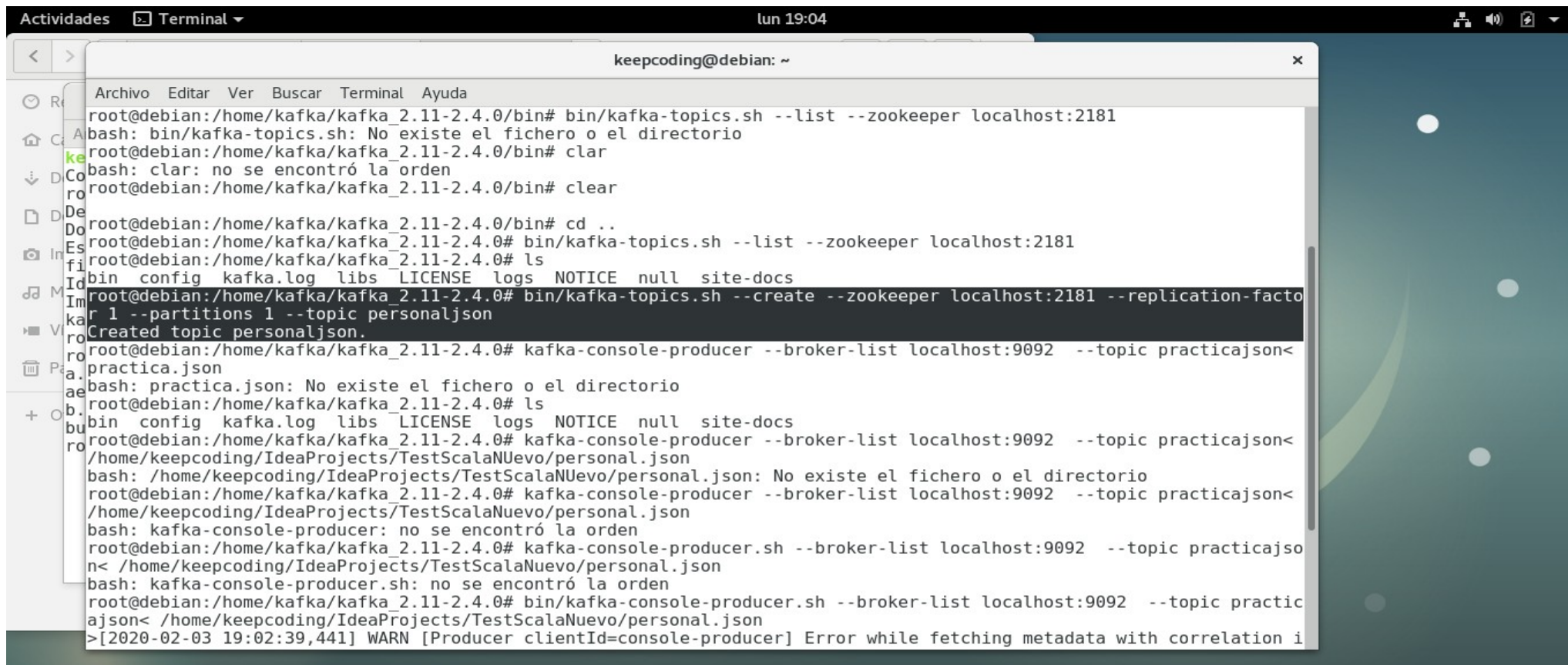


En este pantallazo vemos la creación del topic “**personaljson**” en el cual guardaremos nuestro archivo “personal.json”

A terminal window titled 'keepcoding@debian: ~' with a menu bar (Archivo, Editar, Ver, Buscar, Terminal, Ayuda) and a status bar (lun 19:04). The terminal shows the following commands and output:

```
root@debian:/home/kafka/kafka_2.11-2.4.0/bin# bin/kafka-topics.sh --list --zookeeper localhost:2181
bash: bin/kafka-topics.sh: No existe el fichero o el directorio
root@debian:/home/kafka/kafka_2.11-2.4.0/bin# clear
bash: clear: no se encontró la orden
root@debian:/home/kafka/kafka_2.11-2.4.0/bin# cd ..
root@debian:/home/kafka/kafka_2.11-2.4.0# bin/kafka-topics.sh --list --zookeeper localhost:2181
root@debian:/home/kafka/kafka_2.11-2.4.0# ls
bin  config  kafka.log  libs  LICENSE  logs  NOTICE  null  site-docs
root@debian:/home/kafka/kafka_2.11-2.4.0# bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic personaljson
Created topic personaljson.
root@debian:/home/kafka/kafka_2.11-2.4.0# kafka-console-producer --broker-list localhost:9092 --topic practicajson<
practica.json
bash: practica.json: No existe el fichero o el directorio
root@debian:/home/kafka/kafka_2.11-2.4.0# ls
bin  config  kafka.log  libs  LICENSE  logs  NOTICE  null  site-docs
root@debian:/home/kafka/kafka_2.11-2.4.0# kafka-console-producer --broker-list localhost:9092 --topic practicajson<
/home/keepcoding/IdeaProjects/TestScalaNuevo/personal.json
bash: /home/keepcoding/IdeaProjects/TestScalaNuevo/personal.json: No existe el fichero o el directorio
root@debian:/home/kafka/kafka_2.11-2.4.0# kafka-console-producer --broker-list localhost:9092 --topic practicajson<
/home/keepcoding/IdeaProjects/TestScalaNuevo/personal.json
bash: kafka-console-producer: no se encontró la orden
root@debian:/home/kafka/kafka_2.11-2.4.0# kafka-console-producer.sh --broker-list localhost:9092 --topic practicajson<
/home/keepcoding/IdeaProjects/TestScalaNuevo/personal.json
bash: kafka-console-producer.sh: no se encontró la orden
root@debian:/home/kafka/kafka_2.11-2.4.0# bin/kafka-console-producer.sh --broker-list localhost:9092 --topic practicajson<
/home/keepcoding/IdeaProjects/TestScalaNuevo/personal.json
>[2020-02-03 19:02:39,441] WARN [Producer clientId=console-producer] Error while fetching metadata with correlation id 1: [personaljson: -1]
```

En este siguiente pantallazo de la terminal, vemos como hemos creado un bróker e introducido nuestro archivo, para ello tuvimos que direccionarlo:
</home/keepcoding/IdeaProjects/TestScalaNuevo/personal.json

```
Actividades Terminal lun 19:05
keepcoding@debian: ~
Archivo Editar Ver Buscar Terminal Ayuda
Created topic personaljson.
root@debian:/home/kafka/kafka_2.11-2.4.0# kafka-console-producer --broker-list localhost:9092 --topic practicajson<
practica.json
bash: practica.json: No existe el fichero o el directorio
root@debian:/home/kafka/kafka_2.11-2.4.0# ls
bin config kafka.log libs LICENSE logs NOTICE null site-docs
root@debian:/home/kafka/kafka_2.11-2.4.0# kafka-console-producer --broker-list localhost:9092 --topic practicajson<
/home/keepcoding/IdeaProjects/TestScalaNuevo/personal.json
bash: /home/keepcoding/IdeaProjects/TestScalaNuevo/personal.json: No existe el fichero o el directorio
root@debian:/home/kafka/kafka_2.11-2.4.0# kafka-console-producer --broker-list localhost:9092 --topic practicajson<
/home/keepcoding/IdeaProjects/TestScalaNuevo/personal.json
bash: kafka-console-producer: no se encontró la orden
root@debian:/home/kafka/kafka_2.11-2.4.0# kafka-console-producer.sh --broker-list localhost:9092 --topic practicajson<
/home/keepcoding/IdeaProjects/TestScalaNuevo/personal.json
bash: kafka-console-producer.sh: no se encontró la orden
root@debian:/home/kafka/kafka_2.11-2.4.0# bin/kafka-console-producer.sh --broker-list localhost:9092 --topic practicajson<
/home/keepcoding/IdeaProjects/TestScalaNuevo/personal.json
[2020-02-03 19:02:39,441] WARN [Producer clientId=console-producer] Error while fetching metadata with correlation id 1 : {practicajson=LEADER_NOT_AVAILABLE} (org.apache.kafka.clients.NetworkClient)
root@debian:/home/kafka/kafka_2.11-2.4.0# cd bin
root@debian:/home/kafka/kafka_2.11-2.4.0/bin# kafka-console-producer.sh --broker-list localhost:9092 --topic practicajson<
/home/keepcoding/IdeaProjects/TestScalaNuevo/personal.json
bash: kafka-console-producer.sh: no se encontró la orden
root@debian:/home/kafka/kafka_2.11-2.4.0/bin# bin/kafka-console-producer.sh --broker-list localhost:9092 --topic practicajson<
/home/keepcoding/IdeaProjects/TestScalaNuevo/personal.json
bash: bin/kafka-console-producer.sh: No existe el fichero o el directorio
root@debian:/home/kafka/kafka_2.11-2.4.0/bin# cd ..
root@debian:/home/kafka/kafka_2.11-2.4.0# bin/kafka-console-producer.sh --broker-list localhost:9092 --topic practicajson<
/home/keepcoding/IdeaProjects/TestScalaNuevo/personal.json
root@debian:/home/kafka/kafka_2.11-2.4.0#
```

Aquí podemos ver como lanzamos en kafka-console-producer (terminal de la izquierda), y el resultado en el kafka-console-consumer (terminal de la derecha)

The image displays two terminal windows from a desktop environment. The top bar indicates the user is logged in as 'keepcoding' on a 'debian' machine at 'lun 19:27'.

The left terminal window has a title bar 'keepcoding@debian: ~' and a menu bar with options like Archivo, Editar, Ver, Buscar, Terminal, and Ayuda. It shows the following sequence of actions:
1. A file named 'personal.json' is opened in the editor.
2. The user runs a Kafka console producer command: `bin/kafka-console-producer.sh --broker-list localhost:9092 --topic practicajson < /home/keepcoding/IdeaProjects/TestScalaNuevo/personal.json`.
3. The user enters the message 'hola que tal' and presses Enter.
4. A warning message appears: `[2020-02-03 19:21:25,674] WARN [Producer clientId=console-producer] Error while fetching metadata with correlation id 1 : {topicpruebal=LEADER_NOT_AVAILABLE} (org.apache.kafka.clients.NetworkClient)`.
5. The user enters another 'hola que tal' message.
6. Another identical warning message appears.
7. The user enters a third 'hola que tal' message.
8. A third identical warning message appears.
9. The user enters the message 'bien' and presses Enter.
10. The terminal prompt returns to `root@debian:/home/kafka/kafka_2.11-2.4.0#`.

The right terminal window also has a title bar 'keepcoding@debian: ~' and a similar menu bar. It shows the execution of the Kafka console consumer command: `bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic practicajson --from-beginning`. The output consists of four JSON objects representing messages consumed from the topic:
1. {"id": 1, "first_name": "Jeanette", "last_name": "Penddreth", "email": "jpenddreth@census.gov", "gender": "Female", "ip_address": "26.58.193.2"},
2. {"id": 2, "first_name": "Giavani", "last_name": "Frediani", "email": "gfredianil@senate.gov", "gender": "Male", "ip_address": "229.179.4.212"},
3. {"id": 3, "first_name": "Noell", "last_name": "Bea", "email": "nbea2@imageshack.us", "gender": "Female", "ip_address": "180.66.162.255"},
4. {"id": 4, "first_name": "Willard", "last_name": "Valek", "email": "wvalek3@vk.com", "gender": "Male", "ip_address": "67.76.188.26"}.

Tras ejecutar el código en scala, podemos observar la tabla de datos con los filtros realizados:

The screenshot shows the IntelliJ IDEA Community Edition interface. The main editor displays a Scala file named `pruebapracticaconsumerkafka.scala` with the following code:

```
//hacemos un schema para estructurar nuestros datos en una tabla

val schema= new StructType().add( name = "id",IntegerType).add( name = "first_name",StringType)
.add( name = "last_name",StringType).add( name = "email",StringType)
.add( name = "gender",StringType).add( name = "ip_address",StringType)
//hacemos un select similar a sql con los filtros para que no aparezcan Jeanette y Female
val personDF=res.select(from_json(col( colName = "value"),schema).as ( alias = "data"))
.select ( col = "data.*")
.filter( conditionExpr = "data.first_name != 'Jeanette'")
.filter( conditionExpr = "data.gender != 'Female'")
print("mostrar los datos por consola")

//generamos el Stream de escritura con salida por consola
personDF.writeStream
.format( source = "console")
```

The Run tool window at the bottom shows the output of the program, displaying a table of data with columns: id, first_name, last_name, email, gender, and ip_address. The data is filtered to exclude Jeanette and Female.

id	first_name	last_name	email	gender	ip_address
2	Giavani	Frediani	gfrediani@senate...	Male	229.179.4.212
4	Willard	Valek	wvalek3@vk.com	Male	67.76.188.26
2	Giavani	Frediani	gfrediani@senate...	Male	229.179.4.212
4	Willard	Valek	wvalek3@vk.com	Male	67.76.188.26

The status bar at the bottom indicates that the build completed successfully in 11 s 282 ms (moments ago).

Y éste es el código utilizado:

```
import org.apache.spark.sql.SparkSession

import org.apache.spark.sql.types.{IntegerType, StringType, StructType}

import org.apache.spark.sql.functions.{col, from_json}

import sun.text.normalizer.UCharacter.NumericType

object Kafkajson {

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

    val spark=SparkSession.builder().appName("kafkajson").master("local[2]").getOrCreate()

    val dfStream=spark.readStream

      .format("kafka")

      .option("kafka.bootstrap.servers","localhost:9092")

      .option("subscribe","practicajson")

      .option("startingOffsets","earliest")

      .load()

    // castear los datos leídos en formato kafka para convertirlos en strings

    val res=dfStream.selectExpr("CAST(value AS STRING)")

    //hacemos un schema para estructurar nuestros datos en una tabla

    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)
```

//hacemos un select simliar a sql con los filtros para que no aparezcan Jeanette y Female

```
val personDF=res.select(from_json(col("value"),schema).as ("data"))
```

```
.select ("data.*")
```

```
.filter("data.first_name != 'Jeanette'")
```

```
.filter("data.gender != 'Female'")
```

```
print("mostrar los datos por consola")
```

//generamos el Stream de escritura con salida por consola

```
personDF.writeStream
```

```
.format("console")
```

```
.outputMode("append")
```

```
.start()
```

```
.awaitTermination()
```

```
}
```

```
}
```