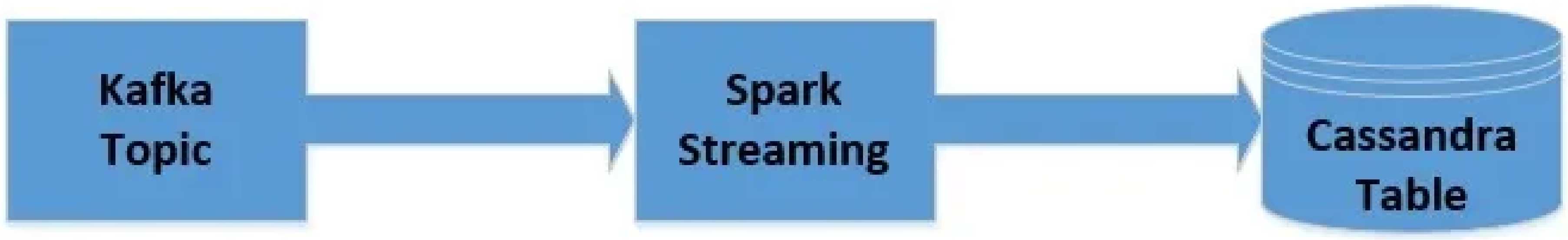


# Building an End-to-End Streaming Project with Spark, Kafka, and Cassandra

In this article we will walk through the process of setting up and running an end-to-end streaming project using Spark, Kafka, and Cassandra. This project enables the consumption of messages from a Kafka topic, processing the data using Spark, and storing the results in Cassandra. Let's get started!

## 1. Running Servers

First, we need to start the required servers. Follow the instructions below to start Spark, Kafka, and Cassandra.

### Start Spark

Navigate to the Spark directory and run the following command:

```
sbin/start-all.sh
```

Use the `jps` command to verify that Spark has started successfully.

### Start Kafka

Navigate to the Kafka directory.
Start **Zookeeper** by running the following command:

```
nohup bin/zookeeper-server-start.sh config/zookeeper.properties >> nohup.out &
```

Start **Kafka** by running the following command:

```
nohup bin/kafka-server-start.sh config/server.properties >> nohup.out &
```

Create a **Kafka topic** by running the following commands:

```
bin/kafka-topics.sh --create --bootstrap-server localhost:9092 --replication-factor 1 --partitions 1 --topic mytopic

bin/kafka-topics.sh --list --zookeeper localhost:2181
```

### Start Cassandra

Navigate to the Cassandra directory.
Start **Cassandra** by running the following command:

```
nohup bin/cassandra -f &
```

Open the **Cassandra shell** by running the following command:

```
bin/cqlsh
```

Create a keyspace and table by executing the following commands:

```
create keyspace sparkdata with replication ={'class':'SimpleStrategy','replication_factor':1};
use sparkdata;

CREATE TABLE cust_data (fname text, lname text, url text, product text, cnt counter, primary key (fname, lname, url, product));
select * from cust_data;
```

## 2. Spark Kafka Cassandra Streaming Code

Configure Spark to connect with Cassandra and Kafka:

- Go to the Spark installation directory and navigate to the `conf` folder.
- Copy the `spark-env.sh.template` file and rename it as `spark-env.sh`.
- Open the `spark-env.sh` file using a text editor.
- Add the following lines at the end of the file to configure the necessary dependencies:

```
export SPARK_CLASSPATH="/home/fayssal/apache-cassandra-4.0.10/lib/*:/home/fayssal/kafka_2.12-3.5.0/*"
export SPARK_EXTRA_CLASSPATH="/home/fayssal/apache-cassandra-4.0.10/lib/*:/home/fayssal/kafka_2.12-3.5.0/*"
```

Now, Navigate to the Spark directory and run the following command to start the Spark Shell:

```
bin/spark-shell --packages "com.datastax.spark:spark-cassandra-connector_2.12:3.3.0","org.apache.spark:spark-sql-kafka-0-10_2.12:3.4.0"
```

Inside the Spark Shell, execute the following code:

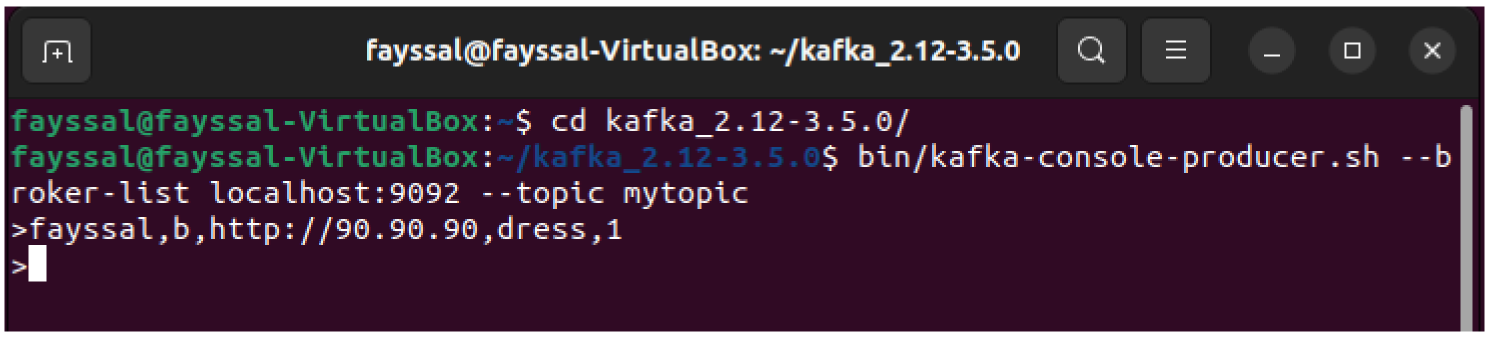
```
import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.functions._
import org.apache.spark.sql.streaming.Trigger

val spark = SparkSession
  .builder()
  .appName("KafkaStructuredStreaming")
  .config("spark.cassandra.connection.host", "127.0.0.1")
  .getOrCreate()

import spark.implicits._

val kafkaBrokers = "localhost:9092"
val kafkaTopic = "mytopic"

val kafkaDF = spark
  .readStream
  .format("kafka")
  .option("kafka.bootstrap.servers", kafkaBrokers)
  .option("subscribe", kafkaTopic)
  .load()
  .selectExpr("CAST(value AS STRING)")
  .as[String]

val dataDF = kafkaDF
  .map(line => {
    val arr = line.split(",")
    (arr(0), arr(1), arr(2), arr(3), arr(4))
  })
  .toDF("fname", "lname", "url", "product", "cnt")

dataDF
  .writeStream
  .format("org.apache.spark.sql.cassandra")
  .option("keyspace", "sparkdata")
  .option("table", "cust_data")
  .option("checkpointLocation", "/home/fayssal/spark-3.4.0-bin-hadoop3/checkpoint")
  .start()

spark.streams.awaitAnyTermination()
```

## 3. Testing

Now that we have set up the streaming pipeline, let's test it by creating a Kafka consumer and producer.

### Create a Producer

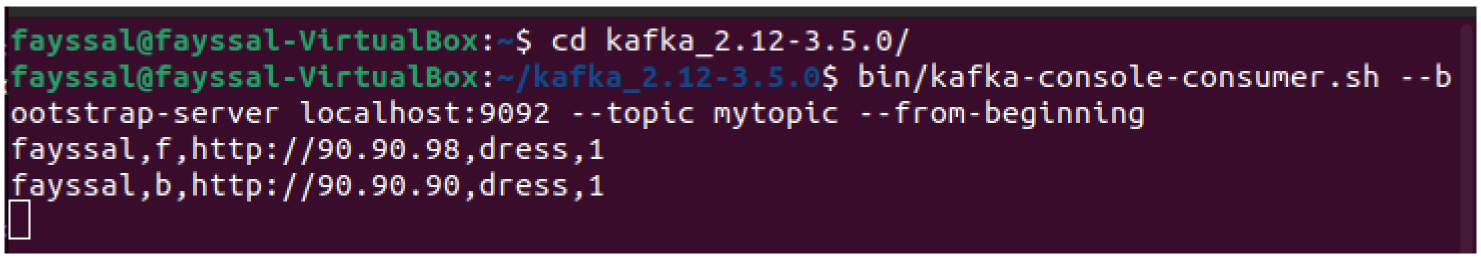
```
bin/kafka-console-producer.sh --broker-list localhost:9092 --topic mytopic
```



```
fayssal@fayssal-VirtualBox:~$ cd kafka_2.12-3.5.0/
fayssal@fayssal-VirtualBox:~/kafka_2.12-3.5.0$ bin/kafka-console-producer.sh --b
roker-list localhost:9092 --topic mytopic
>fayssal,b,http://90.90.90,dress,1
>
```

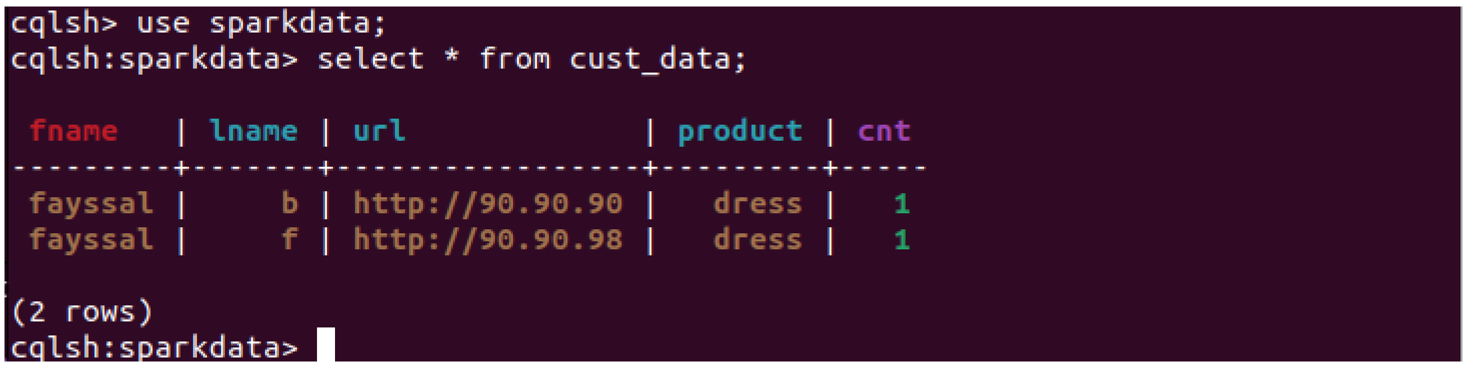
### Create a Consumer

```
bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic mytopic --from-beginning
```



```
fayssal@fayssal-VirtualBox:~$ cd kafka_2.12-3.5.0/
fayssal@fayssal-VirtualBox:~/kafka_2.12-3.5.0$ bin/kafka-console-consumer.sh --b
ootstrap-server localhost:9092 --topic mytopic --from-beginning
fayssal,f,http://90.90.98,dress,1
fayssal,b,http://90.90.90,dress,1
```

### Check Cassandra Keyspace



```
cqlsh> use sparkdata;
cqlsh:sparkdata> select * from cust_data;

 fname   | lname | url             | product | cnt
---------+-------+-----------------+---------+-----
 fayssal |     b | http://90.90.90 |   dress |   1
 fayssal |     f | http://90.90.98 |   dress |   1

(2 rows)
cqlsh:sparkdata>
```