파일 > 카프카 > 스파크 > 엘라스틱 서치

(참고:
https://www.youtube.com/watch?v=1kMcBH4apao&list=PL589M8KPPT1YP2IN8nXbqfheRwAnzdDLx&index=7)

모듈 설치

이 프로젝트를 하려면 필요한 모듈이 많다.

모듈을 다운받을 루트 홈의 library 폴더로 간다

```
cd ~/library
```

필요한 모듈은 다음과 같다

- spark-streaming
- spark-streaming-kafka
- spark-sql-kafka
- kafka-clients
- kafka-streams
- kafka

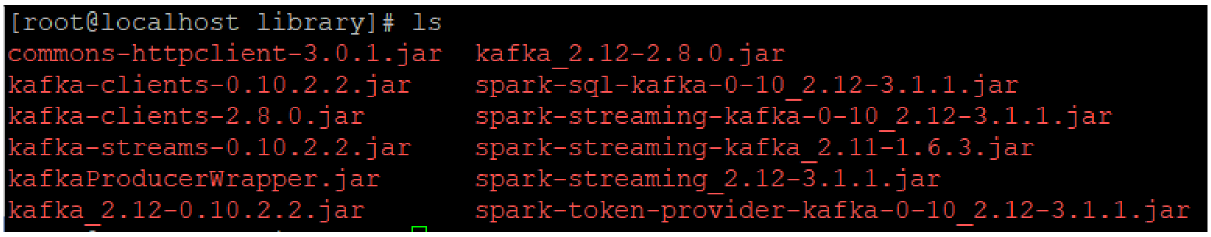
maven repository (https://mvnrepository.com/)에서 검색 후 다운받을 수 있다

아래 명령어로 쭉 다운받자

spark-streaming	wget https://repo1.maven.org/maven2/org/apache/spark/spark-streaming_2.12/3.1.1/spark-streaming_2.12-3.1.1.jar
spark-streaming-kafka	wget https://repo1.maven.org/maven2/org/apache/spark/spark-streaming-kafka-0-10_2.12/3.1.1/spark-streaming-kafka-0-10_2.12-3.1.1.jar

spark-sql-kafka	wget https://repo1.maven.org/maven2/org/apache/spark/spark-sql-kafka-0-10_2.12/3.1.1/spark-sql-kafka-0-10_2.12-3.1.1.jar
kafka-clients	wget https://repo1.maven.org/maven2/org/apache/kafka/kafka-clients/2.8.0/kafka-clients-2.8.0.jar
kafka-streams	wget https://repo1.maven.org/maven2/org/apache/kafka/kafka-streams/0.10.2.2/kafka-streams-0.10.2.2.jar
kafka	wget https://repo1.maven.org/maven2/org/apache/kafka/kafka_2.12/0.10.2.2/kafka_2.12-0.10.2.2.jar
spark-token-provider-kafka	wget https://repo1.maven.org/maven2/org/apache/spark/spark-token-provider-kafka-0-10_2.12/3.1.1/spark-token-provider-kafka-0-10_2.12-3.1.1.jar

제대로 다운받아졌는지 ls 명령어로 확인한다



```
[root@localhost library]# ls
commons-httpclient-3.0.1.jar  kafka_2.12-2.8.0.jar
kafka-clients-0.10.2.2.jar    spark-sql-kafka-0-10_2.12-3.1.1.jar
kafka-clients-2.8.0.jar       spark-streaming-kafka-0-10_2.12-3.1.1.jar
kafka-streams-0.10.2.2.jar    spark-streaming-kafka_2.11-1.6.3.jar
kafkaProducerWrapper.jar      spark-streaming_2.12-3.1.1.jar
kafka_2.12-0.10.2.2.jar       spark-token-provider-kafka-0-10_2.12-3.1.1.jar
```

(다운받은것만 제대로 있으면 되고 다른건 없어도 괜찮음)

ZOOKEEPER & KAFKA 서버 켜기

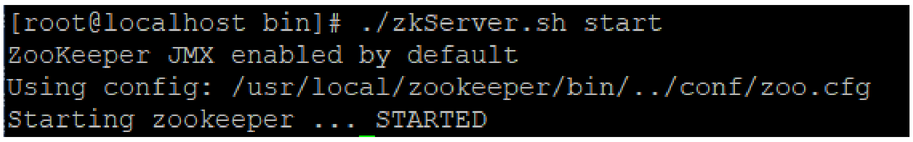
이제 주키퍼랑 카프카 서버를 켜자

주키퍼 실행파일 페이지로 이동

```
cd /usr/local/zookeeper/bin
```

주키퍼 서버열기

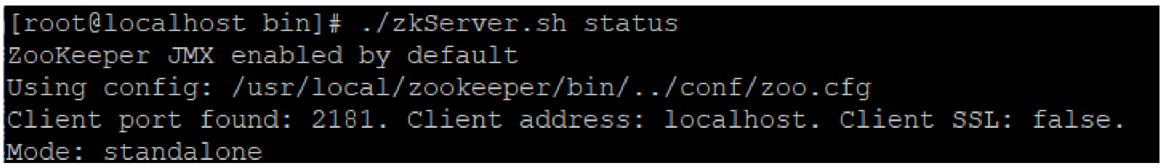
```
./zkServer.sh start
```



```
[root@localhost bin]# ./zkServer.sh start
ZooKeeper JMX enabled by default
Using config: /usr/local/zookeeper/bin/../conf/zoo.cfg
Starting zookeeper ... STARTED
```

열렸는지 확인하기

```
./zkServer.sh status
```



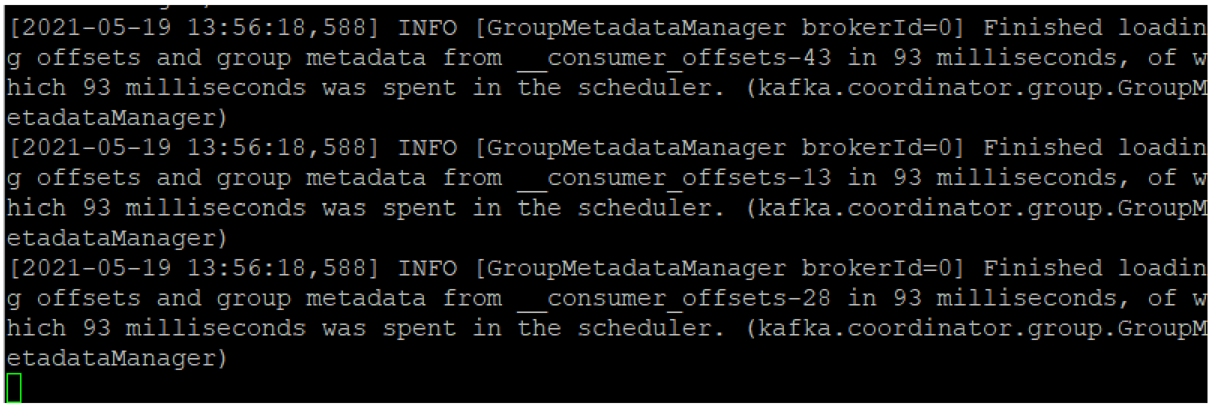
```
[root@localhost bin]# ./zkServer.sh status
ZooKeeper JMX enabled by default
Using config: /usr/local/zookeeper/bin/../conf/zoo.cfg
Client port found: 2181. Client address: localhost. Client SSL: false.
Mode: standalone
```

카프카 실행 폴더로 이동한다

```
cd /usr/local/kafka/bin
```

실행시킨다

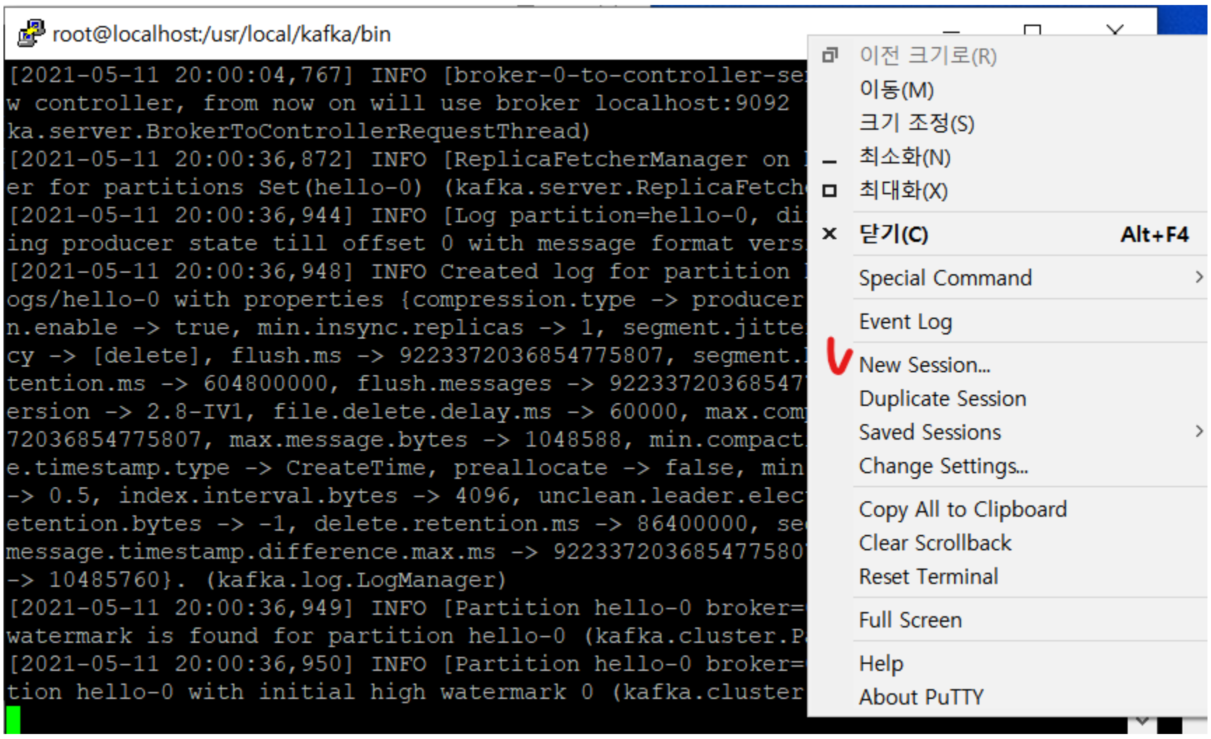
```
./kafka-server-start.sh ../config/server.properties
```



```
[2021-05-19 13:56:18,588] INFO [GroupMetadataManager brokerId=0] Finished loadin
g offsets and group metadata from __consumer_offsets-43 in 93 milliseconds, of w
hich 93 milliseconds was spent in the scheduler. (kafka.coordinator.group.GroupM
etadataManager)
[2021-05-19 13:56:18,588] INFO [GroupMetadataManager brokerId=0] Finished loadin
g offsets and group metadata from __consumer_offsets-13 in 93 milliseconds, of w
hich 93 milliseconds was spent in the scheduler. (kafka.coordinator.group.GroupM
etadataManager)
[2021-05-19 13:56:18,588] INFO [GroupMetadataManager brokerId=0] Finished loadin
g offsets and group metadata from __consumer_offsets-28 in 93 milliseconds, of w
hich 93 milliseconds was spent in the scheduler. (kafka.coordinator.group.GroupM
etadataManager)
```

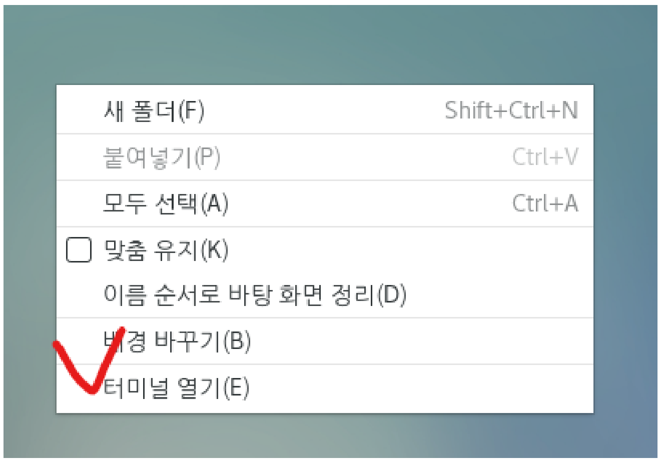
서버가 잘 켜지면 커맨드 창을 새로 하나 띄워서 kafka 콘솔을 열자

putty 라면 위 하얀 바에 우클릭하고 new session 을 클릭해서 창을 하나 더 연다



```
[2021-05-11 20:00:04,767] INFO [broker-0-to-controller-se
w controller, from now on will use broker localhost:9092
ka.server.BrokerToControllerRequestThread)
[2021-05-11 20:00:36,872] INFO [ReplicaFetcherManager on
er for partitions Set(hello-0) (kafka.server.ReplicaFetch
[2021-05-11 20:00:36,944] INFO [Log partition=hello-0, di
ing producer state till offset 0 with message format vers
[2021-05-11 20:00:36,948] INFO Created log for partition
ogs/hello-0 with properties {compression.type -> producer
n.enable -> true, min.insync.replicas -> 1, segment.jitte
cy -> [delete], flush.ms -> 9223372036854775807, segment.
tention.ms -> 604800000, flush.messages -> 92233720368547
ersion -> 2.8-IV1, file.delete.delay.ms -> 60000, max.com
72036854775807, max.message.bytes -> 1048588, min.compact
e.timestamp.type -> CreateTime, preallocate -> false, min
-> 0.5, index.interval.bytes -> 4096, unclean.leader.elec
etention.bytes -> -1, delete.retention.ms -> 86400000, se
message.timestamp.difference.max.ms -> 922337203685477580
-> 10485760}. (kafka.log.LogManager)
[2021-05-11 20:00:36,949] INFO [Partition hello-0 broker=
watermark is found for partition hello-0 (kafka.cluster.P
[2021-05-11 20:00:36,950] INFO [Partition hello-0 broker=
tion hello-0 with initial high watermark 0 (kafka.cluster
```

만약 centos 의 커맨드 창을 쓰고 있다면 바탕화면 우클릭해서 커맨드창 열기를 클릭해서 창을 하나 더 연다



새로 연 커맨드 창에서 kafka 실행파일이 모여있는 폴더로 이동

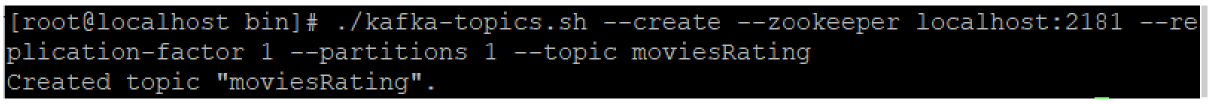
```
cd /usr/local/kafka/bin
```

토픽 실행 명령 입력하기

제일 뒤의 토픽명은 moviesRating 으로 해주자

```
./kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic moviesRating
```

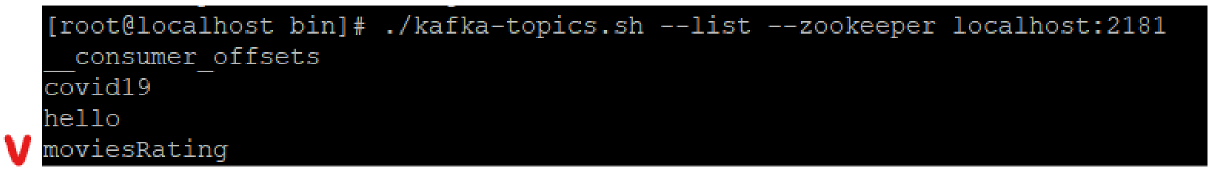
성공화면



```
[root@localhost bin]# ./kafka-topics.sh --create --zookeeper localhost:2181 --re
plication-factor 1 --partitions 1 --topic moviesRating
Created topic "moviesRating".
```

토픽이 잘 생성되었는지 확인하자

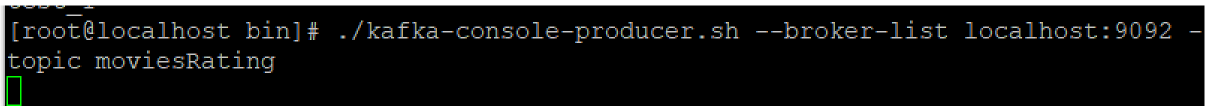
```
./kafka-topics.sh --list --zookeeper localhost:2181
```



```
[root@localhost bin]# ./kafka-topics.sh --list --zookeeper localhost:2181
__consumer_offsets
covid19
hello
moviesRating
```

이제 카프카 프로듀서 콘솔을 열자

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



```
[root@localhost bin]# ./kafka-console-producer.sh --broker-list localhost:9092 -
topic moviesRating
```

SPARK 에서 SCALA 코딩하기

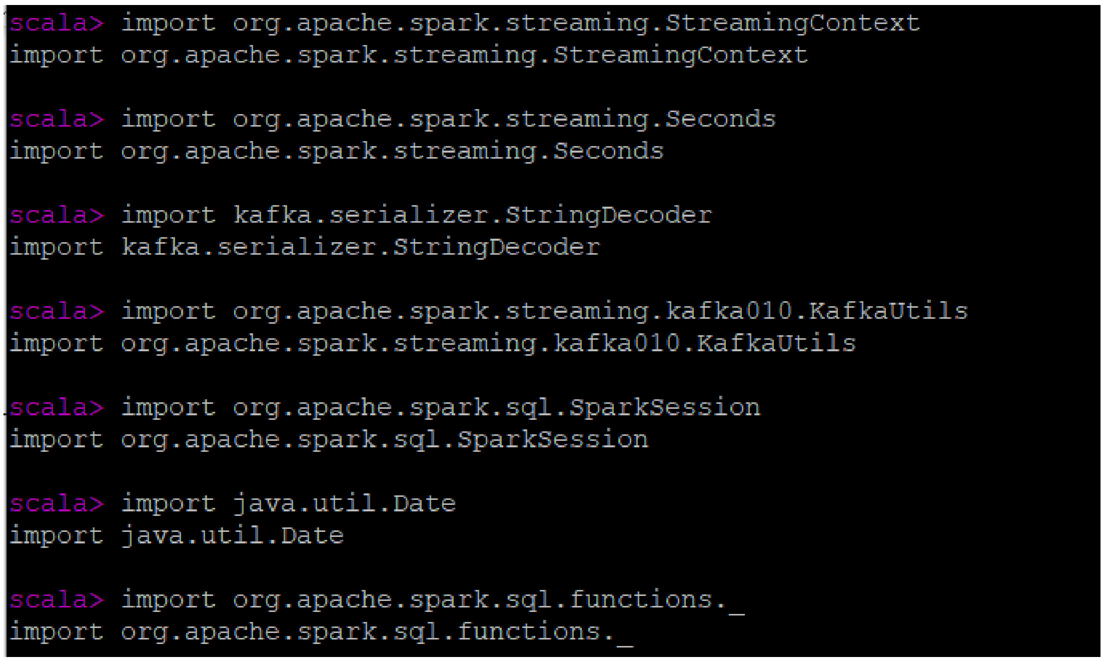
이제 아까 위에서 했던 것 처럼 커맨드 창을 하나 더열고 스파크를 실행시킨다

```
spark-shell --master=local[1] --packages="org.elasticsearch:elasticsearch-spark-30_2.12:7.12.0" --jars /root/library/commons-httpclient-3.0.1.jar,/root/library/spark-streaming_2.12-3.1.1.jar,/root/library/spark-streaming-kafka-0-10_2.12-3.1.1.jar,/root/library/spark-sql-kafka-0-
```

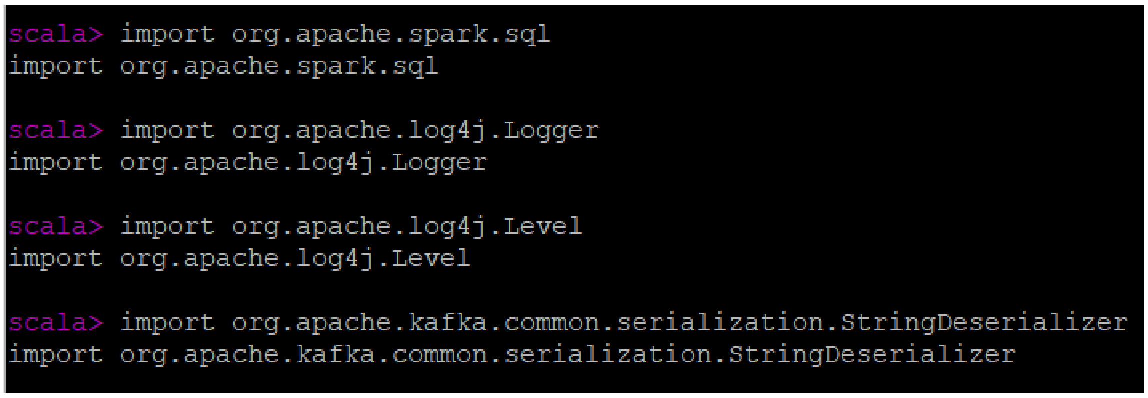
```
10_2.12-3.1.1.jar,/root/library/kafka-clients-2.8.0.jar,/root/library/kafka-streams-
0.10.2.2.jar,/root/library/kafka_2.12-0.10.2.2.jar,/root/library/spark-token-provider-kafka-0-
10_2.12-3.1.1.jar
```

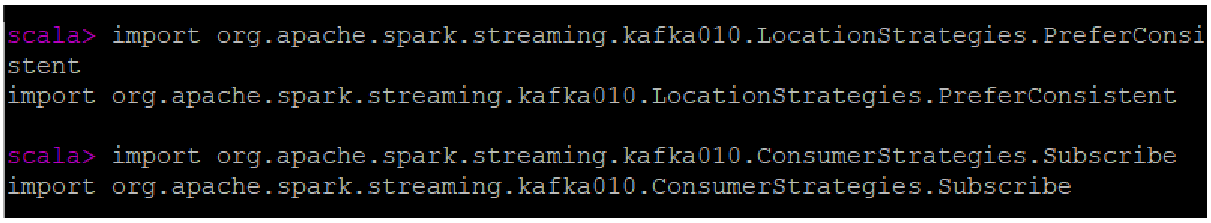
일단 모듈을 임포트한다

```
import org.apache.spark.streaming.StreamingContext
import org.apache.spark.streaming.Seconds
import kafka.serializer.StringDecoder
import org.apache.spark.streaming.kafka010.KafkaUtils
import org.apache.spark.sql.SparkSession
import java.util.Date
import org.apache.spark.sql.functions._
import org.apache.spark.sql
import org.apache.log4j.Logger
import org.apache.log4j.Level
import org.apache.kafka.common.serialization.StringDeserializer
import org.apache.spark.streaming.kafka010.LocationStrategies.PreferConsistent
import org.apache.spark.streaming.kafka010.ConsumerStrategies.Subscribe
import org.apache.kafka.clients.consumer.ConsumerRecord
```



```
scala> import org.apache.spark.streaming.StreamingContext
import org.apache.spark.streaming.StreamingContext

scala> import org.apache.spark.streaming.Seconds
import org.apache.spark.streaming.Seconds

scala> import kafka.serializer.StringDecoder
import kafka.serializer.StringDecoder

scala> import org.apache.spark.streaming.kafka010.KafkaUtils
import org.apache.spark.streaming.kafka010.KafkaUtils

scala> import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.SparkSession

scala> import java.util.Date
import java.util.Date

scala> import org.apache.spark.sql.functions._
import org.apache.spark.sql.functions._
```

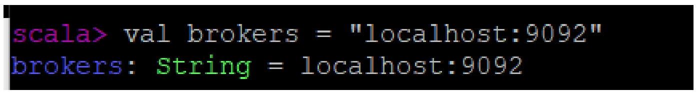

```
scala> import org.apache.spark.sql
import org.apache.spark.sql

scala> import org.apache.log4j.Logger
import org.apache.log4j.Logger

scala> import org.apache.log4j.Level
import org.apache.log4j.Level

scala> import org.apache.kafka.common.serialization.StringDeserializer
import org.apache.kafka.common.serialization.StringDeserializer
```


```
scala> import org.apache.spark.streaming.kafka010.LocationStrategies.PreferConsi
stent
import org.apache.spark.streaming.kafka010.LocationStrategies.PreferConsistent

scala> import org.apache.spark.streaming.kafka010.ConsumerStrategies.Subscribe
import org.apache.spark.streaming.kafka010.ConsumerStrategies.Subscribe
```

이제 필요한 객체들을 만들어준다

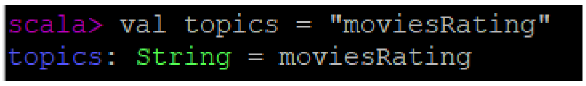
우선 브로커 객체를 만든다

```
val brokers = "localhost:9092"
```


```
scala> val brokers = "localhost:9092"
brokers: String = localhost:9092
```

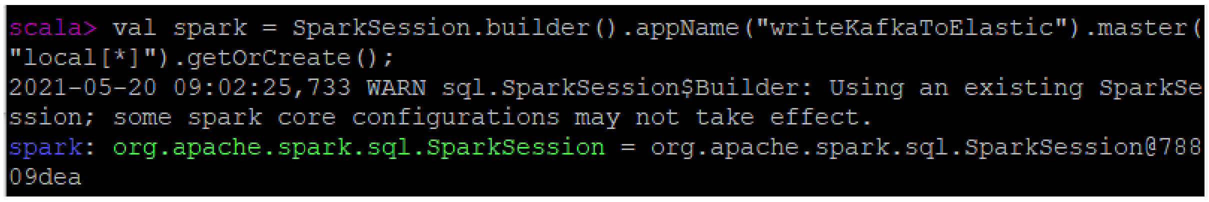
토픽 객체를 만든다

```
val topics = "moviesRating"
```


```
scala> val topics = "moviesRating"
topics: String = moviesRating
```

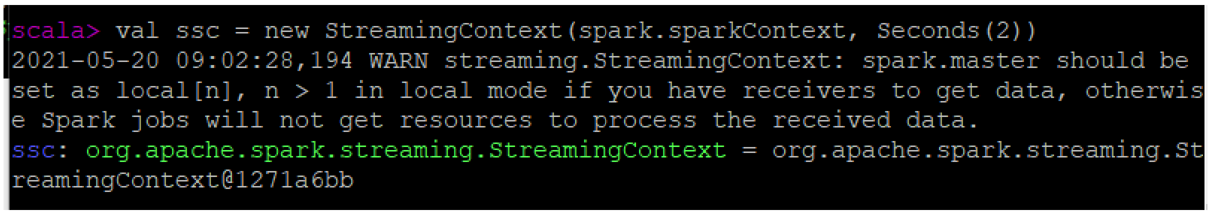
스파크 연결 객체를 만든다

```
val spark = SparkSession.builder().appName("writeKafkaToElastic").master("local[*]").getOrCreate();
```


```
scala> val spark = SparkSession.builder().appName("writeKafkaToElastic").master(
"local[*]").getOrCreate();
2021-05-20 09:02:25,733 WARN sql.SparkSession$Builder: Using an existing SparkSe
ssion; some spark core configurations may not take effect.
spark: org.apache.spark.sql.SparkSession = org.apache.spark.sql.SparkSession@788
09dea
```

StreamingContext 객체를 만든다

```
val ssc = new StreamingContext(spark.sparkContext, Seconds(10))
```

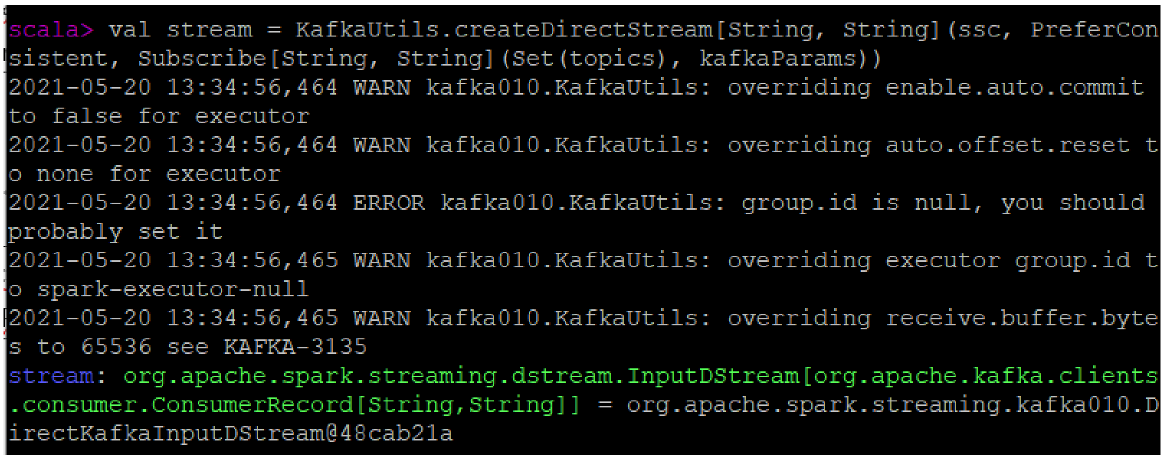

```
scala> val ssc = new StreamingContext(spark.sparkContext, Seconds(2))
2021-05-20 09:02:28,194 WARN streaming.StreamingContext: spark.master should be
set as local[n], n > 1 in local mode if you have receivers to get data, otherwis
e Spark jobs will not get resources to process the received data.
ssc: org.apache.spark.streaming.StreamingContext = org.apache.spark.streaming.St
reamingContext@1271a6bb
```

Kafka 파라메터 객체를 만든다

```
val kafkaParams = Map[String,String](
"zookeeper.connect" -> "localhost:2181",
"bootstrap.servers" -> "localhost:9092",
"metadata.broker.list" -> brokers,
"key.deserializer" -> "org.apache.kafka.common.serialization.StringDeserializer",
"value.deserializer" -> "org.apache.kafka.common.serialization.StringDeserializer",
"auto.offset.reset" -> "latest",
"group.id" -> "use_a_separate_group_id_for_each_stream")
```

Streaming 객체를 만든다

```
val stream = KafkaUtils.createDirectStream[String, String](ssc, PreferConsistent, Subscribe[String,
String](Set(topics), kafkaParams))
```



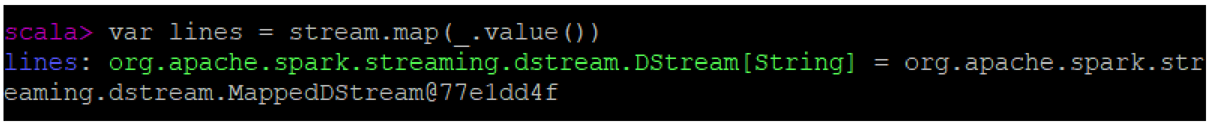
```
scala> val stream = KafkaUtils.createDirectStream[String, String](ssc, PreferCon
sistent, Subscribe[String, String](Set(topics), kafkaParams))
2021-05-20 13:34:56,464 WARN kafka010.KafkaUtils: overriding enable.auto.commit
to false for executor
2021-05-20 13:34:56,464 WARN kafka010.KafkaUtils: overriding auto.offset.reset t
o none for executor
2021-05-20 13:34:56,464 ERROR kafka010.KafkaUtils: group.id is null, you should
probably set it
2021-05-20 13:34:56,465 WARN kafka010.KafkaUtils: overriding executor group.id t
o spark-executor-null
2021-05-20 13:34:56,465 WARN kafka010.KafkaUtils: overriding receive.buffer.byte
s to 65536 see KAFKA-3135
stream: org.apache.spark.streaming.dstream.InputDStream[org.apache.kafka.clients
.consumer.ConsumerRecord[String,String]] = org.apache.spark.streaming.kafka010.D
irectKafkaInputDStream@48cab21a
```

추가로 필요한 모듈을 임포트한다

```
import spark.implicits._
```

스트리밍한 데이터를 한줄한줄 가져올 lines 객체를 생성한다

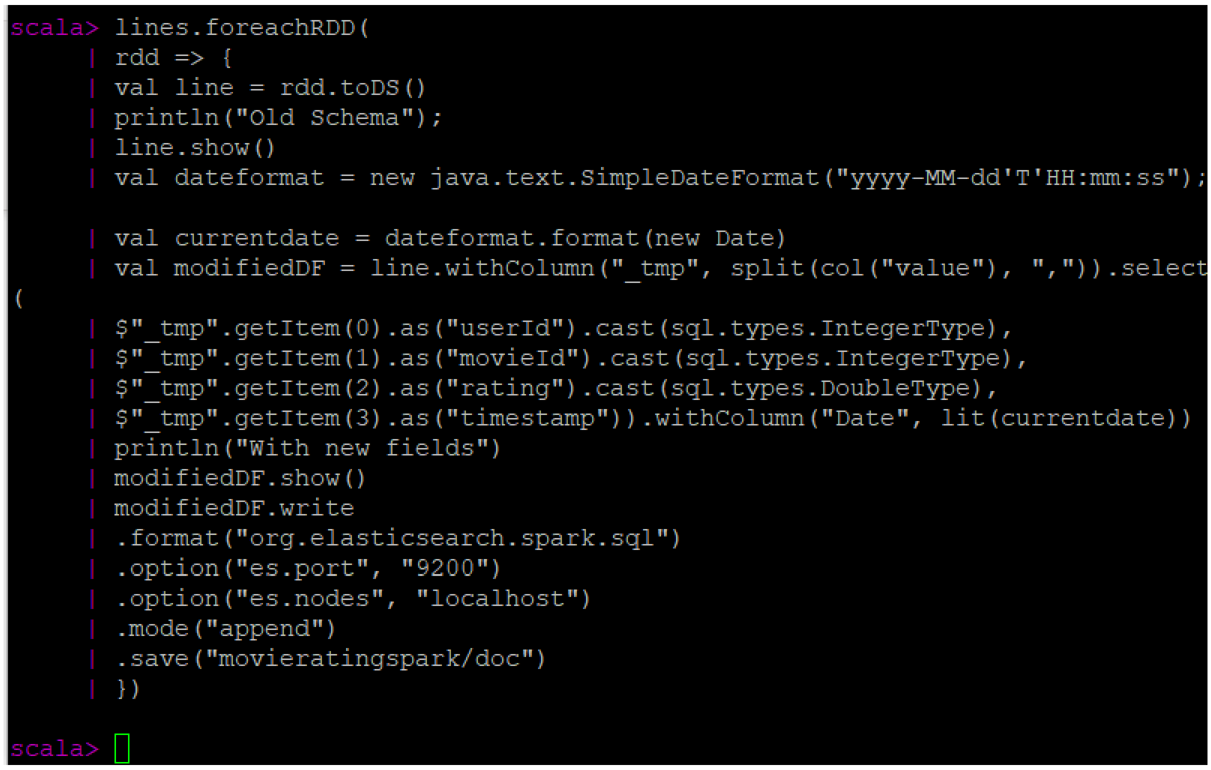
```
var lines = stream.map(_.value())
```



```
scala> var lines = stream.map(_.value())
lines: org.apache.spark.streaming.dstream.DStream[String] = org.apache.spark.str
eaming.dstream.MappedDStream@77e1dd4f
```

RDD 생성후 kafka 에서 값을 불러와 ,를 기준으로 자르고 데이터 프레임으로 만든뒤 엘라스틱서치에 입력하는 코드를 입력해주자

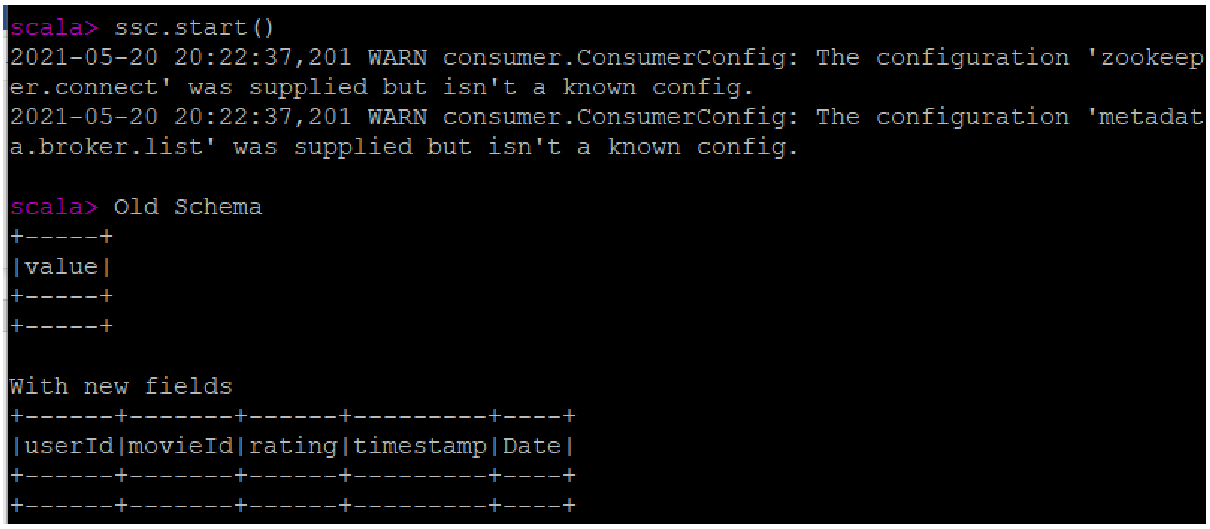
```
lines.foreachRDD(
rdd => {
val line = rdd.toDS()
println("Old Schema");
line.show()
val dateformat = new java.text.SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss");
val currentdate = dateformat.format(new Date)
val modifiedDF = line.withColumn("_tmp", split(col("value"), ",")).select(
$"_tmp".getItem(0).as("userId").cast(sql.types.IntegerType),
$"_tmp".getItem(1).as("movieId").cast(sql.types.IntegerType),
$"_tmp".getItem(2).as("rating").cast(sql.types.DoubleType),
$"_tmp".getItem(3).as("timestamp")).withColumn("Date", lit(currentdate))
println("With new fields")
modifiedDF.show()
modifiedDF.write
.format("org.elasticsearch.spark.sql")
.option("es.port", "9200")
.option("es.nodes", "localhost")
.mode("append")
.save("movieratingspark/doc")
})
```



```
scala> lines.foreachRDD(
     | rdd => {
     | val line = rdd.toDS()
     | println("Old Schema");
     | line.show()
     | val dateformat = new java.text.SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss");

     | val currentdate = dateformat.format(new Date)
     | val modifiedDF = line.withColumn("_tmp", split(col("value"), ",")).select
(
     | $"_tmp".getItem(0).as("userId").cast(sql.types.IntegerType),
     | $"_tmp".getItem(1).as("movieId").cast(sql.types.IntegerType),
     | $"_tmp".getItem(2).as("rating").cast(sql.types.DoubleType),
     | $"_tmp".getItem(3).as("timestamp")).withColumn("Date", lit(currentdate))
     | println("With new fields")
     | modifiedDF.show()
     | modifiedDF.write
     | .format("org.elasticsearch.spark.sql")
     | .option("es.port", "9200")
     | .option("es.nodes", "localhost")
     | .mode("append")
     | .save("movieratingspark/doc")
     | })

scala> 
```

이제 스트리밍을 시작해주자

```
ssc.start()
```

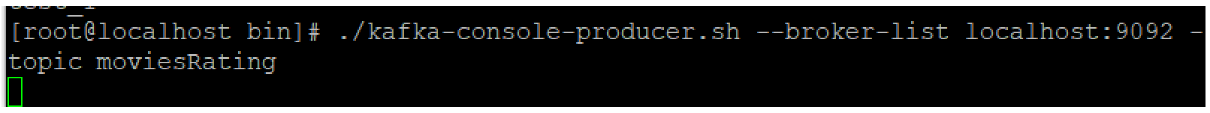


```
scala> ssc.start()
2021-05-20 20:22:37,201 WARN consumer.ConsumerConfig: The configuration 'zookeep
er.connect' was supplied but isn't a known config.
2021-05-20 20:22:37,201 WARN consumer.ConsumerConfig: The configuration 'metadat
a.broker.list' was supplied but isn't a known config.

scala> Old Schema
+-----+
|value|
+-----+
+-----+

With new fields
+------+-------+------+---------+----+
|userId|movieId|rating|timestamp|Date|
+------+-------+------+---------+----+
+------+-------+------+---------+----+
```

카프카에 데이터 입력하기

다시 카프카 컨슈머 콘솔로 돌아간다

가보면 컨슈머 콘솔이 입력을 기다리고 있다



```
[root@localhost bin]# ./kafka-console-producer.sh --broker-list localhost:9092 -
topic moviesRating
```

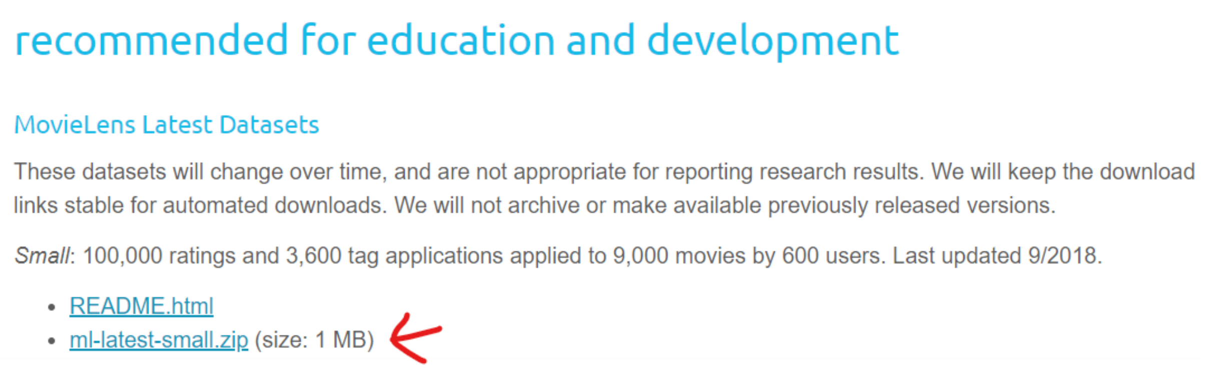
입력할 데이터로는 ratings.csv 파일을 사용할 예정이다

아래 링크에서 다운받을 수 있다

https://grouplens.org/datasets/movielens/

분석용 예제는 250MB 로 너무 크니까 교육용 예제를 사용하자

스크롤을 내리면 아래와 같이 경로를 확인할 수 있다



recommended for education and development

MovieLens Latest Datasets

These datasets will change over time, and are not appropriate for reporting research results. We will keep the download links stable for automated downloads. We will not archive or make available previously released versions.

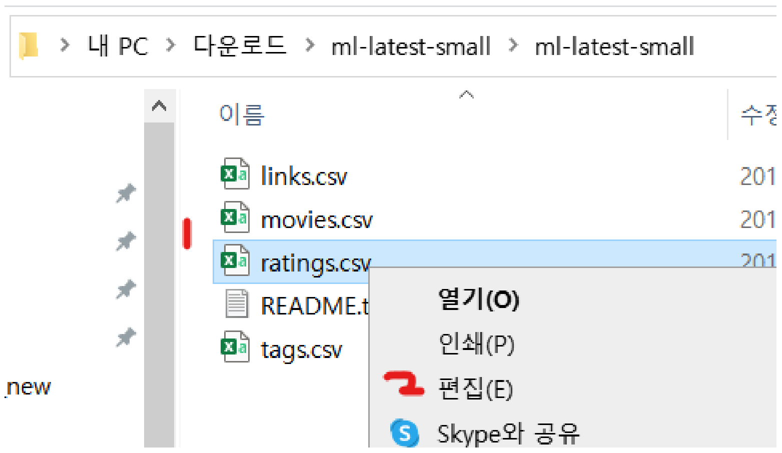
Small: 100,000 ratings and 3,600 tag applications applied to 9,000 movies by 600 users. Last updated 9/2018.

- README.html
- ml-latest-small.zip (size: 1 MB)

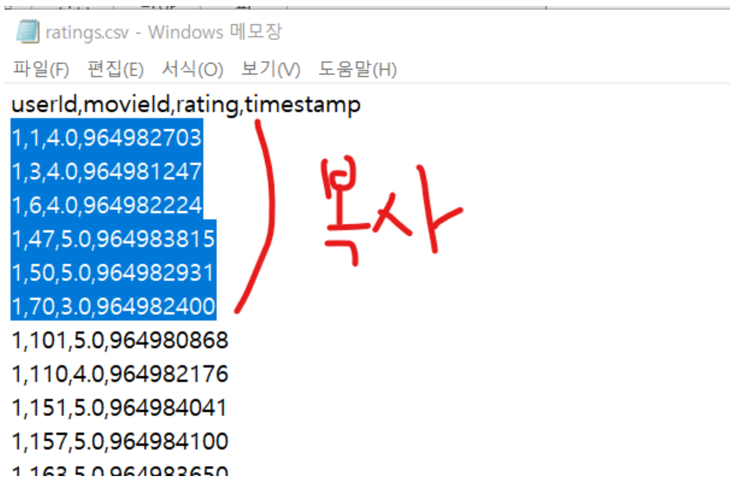
윈도우에서 ml-latest-small.zip 파일을 다운받아 압축 해제하자

압축 해제한 폴더로 들어가면 4 개의 파일을 확인할 수 있다

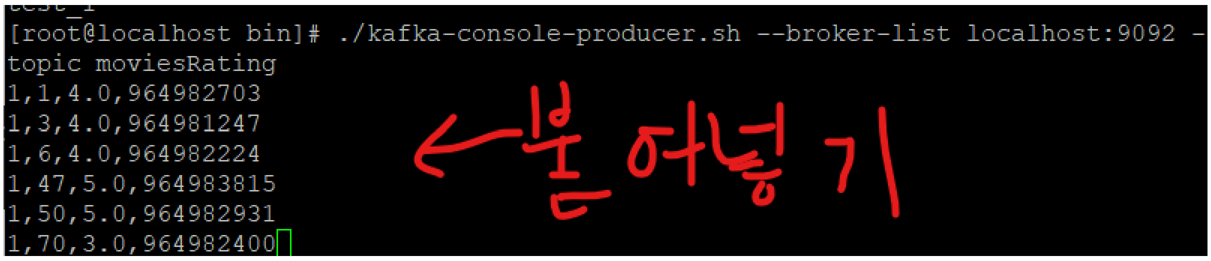
그 중 ratings.csv 파일을 우클릭 후 편집 버튼을 눌러서 메모장 형태로 연다.



열린 파일에서 원하는 만큼 데이터를 복사해서 카프카 프로듀서 창에 붙여넣는다

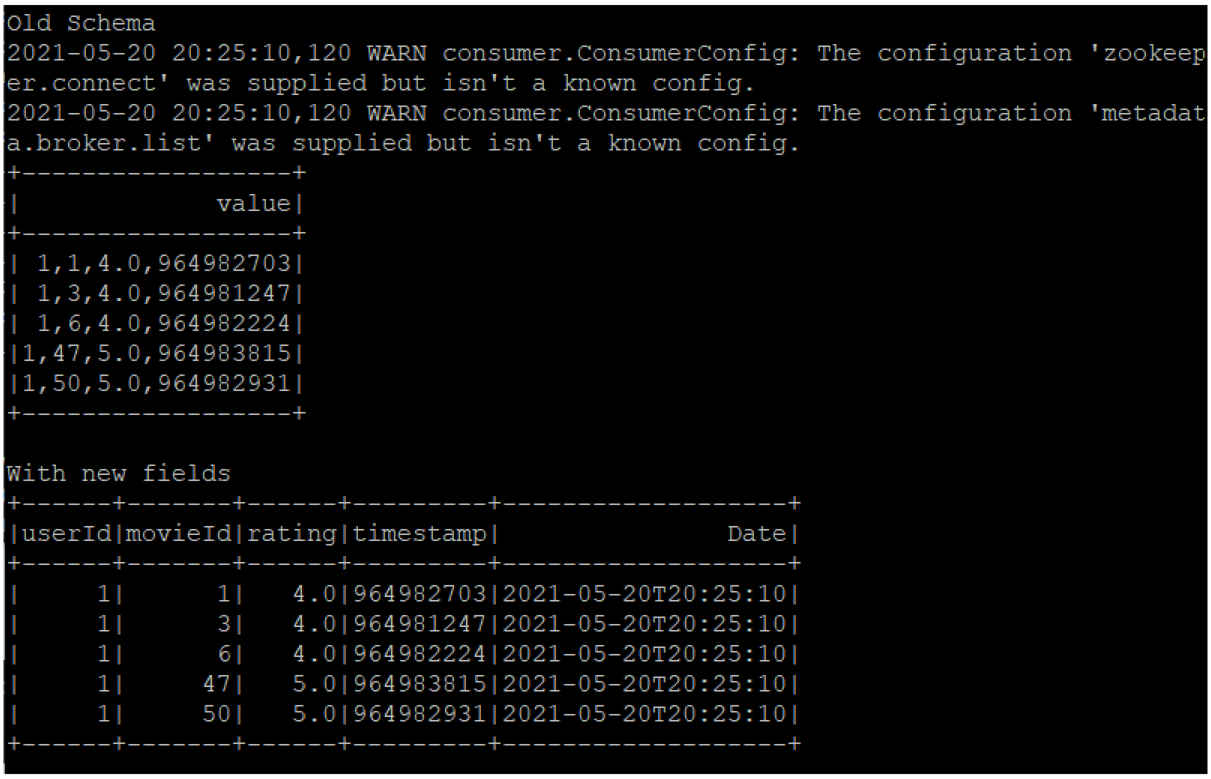

ratings.csv - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
복사

```
userId,movieId,rating,timestamp
1,1,4.0,964982703
1,3,4.0,964981247
1,6,4.0,964982224
1,47,5.0,964983815
1,50,5.0,964982931
1,70,3.0,964982400
1,101,5.0,964980868
1,110,4.0,964982176
1,151,5.0,964984041
1,157,5.0,964984100
```



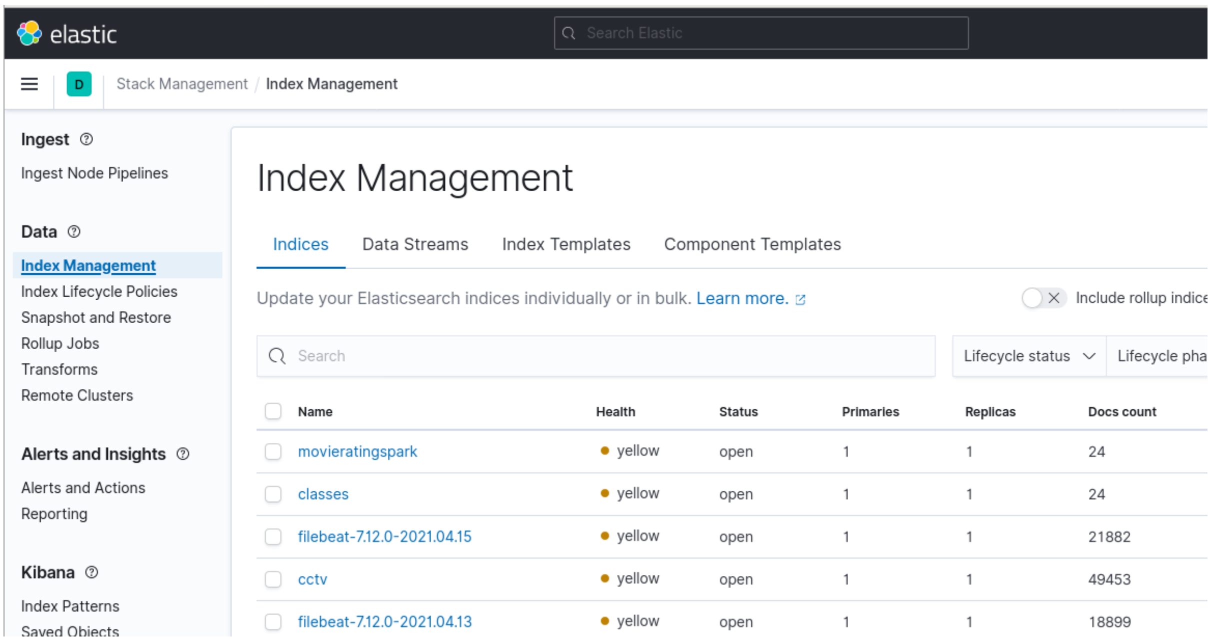
```
[root@localhost bin]# ./kafka-console-producer.sh --broker-list localhost:9092 -
topic moviesRating
1,1,4.0,964982703
1,3,4.0,964981247
1,6,4.0,964982224
1,47,5.0,964983815
1,50,5.0,964982931
1,70,3.0,964982400
```

그런 뒤 실행중인 스칼라 창을 확인하면 데이터가 들어가는 것을 볼 수 있다



```
Old Schema
2021-05-20 20:25:10,120 WARN consumer.ConsumerConfig: The configuration 'zookeep
er.connect' was supplied but isn't a known config.
2021-05-20 20:25:10,120 WARN consumer.ConsumerConfig: The configuration 'metadat
a.broker.list' was supplied but isn't a known config.
+------------------+
|             value|
+------------------+
| 1,1,4.0,964982703|
| 1,3,4.0,964981247|
| 1,6,4.0,964982224|
|1,47,5.0,964983815|
|1,50,5.0,964982931|
+------------------+

With new fields
+------+-------+------+---------+-------------------+
|userId|movieId|rating|timestamp|               Date|
+------+-------+------+---------+-------------------+
|     1|      1|   4.0|964982703|2021-05-20T20:25:10|
|     1|      3|   4.0|964981247|2021-05-20T20:25:10|
|     1|      6|   4.0|964982224|2021-05-20T20:25:10|
|     1|     47|   5.0|964983815|2021-05-20T20:25:10|
|     1|     50|   5.0|964982931|2021-05-20T20:25:10|
+------+-------+------+---------+-------------------+
```

Kibana 에 접속해보면 아래와 같이 인덱스가 잘 생성된 것을 확인할 수 있다



Index Management

Update your Elasticsearch indices individually or in bulk. Learn more.

Name	Health	Status	Primaries	Replicas	Docs count
movieratingspark	● yellow	open	1	1	24
classes	● yellow	open	1	1	24
filebeat-7.12.0-2021.04.15	● yellow	open	1	1	21882
cctv	● yellow	open	1	1	49453
filebeat-7.12.0-2021.04.13	● yellow	open	1	1	18899