Viết NOTES:
Lưu dưới 2 dạng:

1. File docx dùng để chỉnh sửa
2. File .pdf dùng để preview trên Git

Demo một đoạn text dài

Xử lý dữ liệu Twitter trong thời gian thực

Trong bài tutorial này, chúng ta sẽ viết chương trình sử dụng TwitterStreaming API để thu thập dữ liệu trong thời gian thực và truyền về Kafka. Sau đó ta sẽ sử dụng Spark như một Consumer của Kafka để tiến hành xử dữ liệu đổ về. Để hiểu rõ thêm về bài tutorial này, các bạn có thể tham khảo hai bài tutorial trước là: Sử dụng Kafka với Twitter và Tích hợp Kafka với Spark sử dụng Structured Streaming.

Trong bài này, chúng ta sẽ sử dụng kết hợp cả hai ngôn ngữ Java và Scala để viết hai chương trình:

– KafkaTwitterStreaming.java: sử dụng Java để viết chương trình thu thập dữ liệu Twitter trong thời gian thực (Twitter Streaming Data) và truyền về Kafka

– KafkaSparkSS.scala: sử dụng Scala để viết chương trình Spark lấy dữ liệu từ Kafka broker.

Tương tự như trong bài tutorial Sử dụng Kafka với Twitter, chúng ta sẽ sử dụng thư viện twitter4j nhưng là twitter4j-stream thay vì twitter4j-core. Do đó ta thêm dependency sau vào file pom.xml:

```
<!-- https://mvnrepository.com/artifact/org.twitter4j/twitter4j-stream -->
<dependency>
  <groupId>org.twitter4j</groupId>
  <artifactId>twitter4j-stream</artifactId>
  <version>4.0.7</version>
</dependency>
```

Ta tiến hành cấu hình thuộc tính của Kafka và thực hiện kết nối với tài khoản Twitter tương tự như bài Sử dụng Kafka với Twitter với hai hàm sau (ta thêm .setJSONStoreEnabled(true) để truyền dữ liệu về Kafka dưới dạng Json):

```
private static Properties getKafkaProp()
{
  // create instance for properties to access producer configs
  Properties props = new Properties();
  //Assign localhost id
  props.put("bootstrap.servers", "localhost:9092");
  //Set acknowledgements for producer requests.
  props.put("acks", "all");
  //If the request fails, the producer can automatically retry,
  props.put("retries", 0);
  //Specify buffer size in config
  props.put("batch.size", 16384);
  //Reduce the no of requests less than 0
  props.put("linger.ms", 1);
  //The buffer.memory controls the total amount of memory available to the producer for buffering.
  props.put("buffer.memory", 33554432);
  props.put("key.serializer",
      "org.apache.kafka.common.serialization.StringSerializer");
  props.put("value.serializer",
```

```
            "org.apache.kafka.common.serialization.StringSerializer");

    return  props;
  }

  private static Configuration getTwitterConf()
  {
    //Config Twitter API key to access Twitter API
    //The String keys here are only examples and not valid.
    //You need to use your own keys
    ConfigurationBuilder cb = new ConfigurationBuilder();
        cb.setDebugEnabled(true)
          .setJSONStoreEnabled(true)
          .setOAuthConsumerKey("Fljmu9Wp1YVNXhqfmDHDyEAz9")
          .setOAuthConsumerSecret("7CZDMiqhaeV7FOsUTYLgi9utt4eYEVaxqVuKZj5VGHLYqO0mLU")
          .setOAuthAccessToken("1060702756430729216-1L9lL05TdEbanhGDFETkKMknmbw70w")
          .setOAuthAccessTokenSecret("Qu41ydcAzTxClfVW4BMU6UjziS6Lv9Kkwz1zBXKh3JWrx");

    return cb.build();
  }
```

Tiếp theo ta viết hàm getStreamTweets() để thu thập dữ liệu Twitter trong thời gian thực như sau:

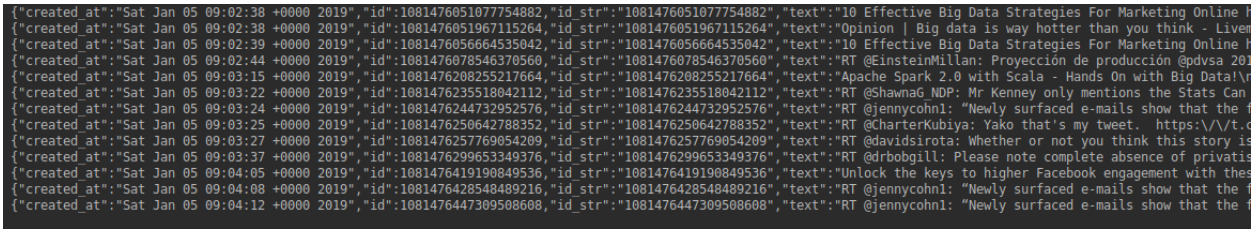
```
   public static void getStreamTweets(final Producer<String, String> producer,String topicName) {
      TwitterStream twitterStream = new TwitterStreamFactory(getTwitterConf()).getInstance();
      StatusListener listener = new StatusListener(){

        public void onStatus(Status status) {
          //Status To JSON String
          String statusJson = DataObjectFactory.getRawJSON(status);
          ProducerRecord data = new ProducerRecord("TwitterStreaming", statusJson);
```

```
            System.out.println(statusJson);

            //Send data
            producer.send(data);

        }

        public void onException(Exception ex) {
            ex.printStackTrace();
        }

        public void onDeletionNotice(StatusDeletionNotice statusDeletionNotice) {

        }

        public void onTrackLimitationNotice(int numberOfLimitedStatuses) {

        }

        public void onScrubGeo(long userId, long upToStatusId) {

        }

        public void onStallWarning(StallWarning warning) {

        }
      };

      twitterStream.addListener(listener);
      twitterStream.filter(topicName);
    }
```

Chạy chương trình KafkaTwitterStreaming với chủ đề tìm kiếm là "Big Data" ta được kết quả sau (các bạn tham khảo thêm về Kafka cũng như cách khởi chạy Kafka trong loạt bài tutorials về Kafka):

```
1  public static void main(String[] args) {
2      //The Kafka Topic
3      String topicName="Big Data";
4
5      //Define a Kafka Producer
6      Producer<String, String> producer = new KafkaProducer<String, String>(getKafkaProp());
7      getStreamTweets(producer,topicName);
8
9  }
```

Kết quả chạy trong IntelliJ (dữ liệu hiển thị là dữ liệu dưới dạng Json, chứa đầy đủ thông tin về bài đăng như ID, nội dung(text), thời gian post(created_at), …):



```
{"created_at":"Sat Jan 05 09:02:38 +0000 2019","id":1081476051077754882,"id_str":"1081476051077754882","text":"10 Effective Big Data Strategies For Marketing Online
{"created_at":"Sat Jan 05 09:02:38 +0000 2019","id":1081476051967115264,"id_str":"1081476051967115264","text":"Opinion | Big data is way hotter than you think - Liver
{"created_at":"Sat Jan 05 09:02:39 +0000 2019","id":1081476056664535042,"id_str":"1081476056664535042","text":"10 Effective Big Data Strategies For Marketing Online
{"created_at":"Sat Jan 05 09:02:44 +0000 2019","id":1081476078546370560,"id_str":"1081476078546370560","text":"RT @EinsteinMillan: Proyección de producción @pdvsa 201
{"created_at":"Sat Jan 05 09:03:15 +0000 2019","id":1081476208255217664,"id_str":"1081476208255217664","text":"Apache Spark 2.0 with Scala - Hands On with Big Data!\r
{"created_at":"Sat Jan 05 09:03:22 +0000 2019","id":1081476235518042112,"id_str":"1081476235518042112","text":"RT @ShawnaG_NDP: Mr Kenney only mentions the Stats Can
{"created_at":"Sat Jan 05 09:03:24 +0000 2019","id":1081476244732952576,"id_str":"1081476244732952576","text":"RT @jennycohn1: “Newly surfaced e-mails show that the
{"created_at":"Sat Jan 05 09:03:25 +0000 2019","id":1081476250642788352,"id_str":"1081476250642788352","text":"RT @CharterKubiya: Yako that's my tweet.  https:\/\/t.c
{"created_at":"Sat Jan 05 09:03:27 +0000 2019","id":1081476257760954209,"id_str":"1081476257760954209","text":"RT @davidsirota: Whether or not you think this story is
{"created_at":"Sat Jan 05 09:03:37 +0000 2019","id":1081476299653349376,"id_str":"1081476299653349376","text":"RT @drbobgill: Please note complete absence of privatis
{"created_at":"Sat Jan 05 09:04:05 +0000 2019","id":1081476419190849536,"id_str":"1081476419190849536","text":"Unlock the keys to higher Facebook engagement with thes
{"created_at":"Sat Jan 05 09:04:08 +0000 2019","id":1081476428548489216,"id_str":"1081476428548489216","text":"RT @jennycohn1: “Newly surfaced e-mails show that the
{"created_at":"Sat Jan 05 09:04:12 +0000 2019","id":1081476447309508608,"id_str":"1081476447309508608","text":"RT @jennycohn1: “Newly surfaced e-mails show that the
```


Kết quả chạy Kafka Consummer (nhận kết quả tương tự như kết quả hiển thị trong IntelliJ, kết quả dưới đây chính là bài đăng thứ 4 có số ID là 1081476078546370560 ở trên):


```
{"created_at":"Sat Jan 05 09:02:44 +0000 2019","id":1081476078546370560,"id_str":"1081476078546370560","text":"RT @EinsteinMillan: P
royección de producción @pdvsa 2019 para fuentes #OPEP direct\/secondary basados en big data. Nota: modelo presenta ine…","source":"
<a href=\"https:\/\/mobile.twitter.com\" rel=\"nofollow\">Twitter Lite<\/a>","truncated":false,"in_reply_to_status_id":null,"in_repl
y_to_status_id_str":null,"in_reply_to_user_id":null,"in_reply_to_user_id_str":null,"in_reply_to_screen_name":null,"user":{"id":17806
1315,"id_str":"178061315","name":"Miriam Millan","screen_name":"MiriamMercedesa","location":". Edo. Anzoategui,  Venezuela","url":nu
ll,"description":"Economista, trabajo independiente,amo la paz ,odio la hipocresia, la mentira y creo en Dios por sobre todas las co
sas del mundo.","translator_type":"regular","protected":false,"verified":false,"followers_count":31,"friends_count":11,"listed_count
":0,"favourites_count":1352,"statuses_count":1661,"created_at":"Fri Aug 13 19:32:13 +0000 2010","utc_offset":null,"time_zone":null,"
geo_enabled":true,"lang":"es","contributors_enabled":false,"is_translator":false,"profile_background_color":"352726","profile_backgr
ound_image_url":"http:\/\/abs.twimg.com\/images\/themes\/theme5\/bg.gif","profile_background_image_url_https":"https:\/\/abs.twimg.c
om\/images\/themes\/theme5\/bg.gif","profile_background_tile":false,"profile_link_color":"D02B55","profile_sidebar_border_color":"82
9D5E","profile_sidebar_fill_color":"99CC33","profile_text_color":"3E4415","profile_use_background_image":true,"profile_image_url":"h
ttp:\/\/pbs.twimg.com\/profile_images\/864951396088590339\/KQhofzyc_normal.jpg","profile_image_url_https":"https:\/\/pbs.twimg.com\/
profile_images\/864951396088590339\/KQhofzyc_normal.jpg","profile_banner_url":"https:\/\/pbs.twimg.com\/profile_banners\/178061315\/
1438734526","default_profile":false,"default_profile_image":false,"following":null,"follow_request_sent":null,"notifications":null},
"geo":null,"coordinates":null,"place":null,"contributors":null,"retweeted_status":{"created_at":"Sat Dec 15 20:41:30 +0000 2018","id
":1074041782558707713,"id_str":"1074041782558707713","text":"Proyección de producción @pdvsa 2019 para fuentes #OPEP direct\/seconda
ry basados en big data. Nota: modelo presenta… https:\/\/t.co\/YoGz7V3nzN","display_text_range":[0,140],"source":"<a href=\"http:\/\
/twitter.com\/download\/iphone\" rel=\"nofollow\">Twitter for iPhone<\/a>","truncated":true,"in_reply_to_status_id":null,"in_reply_t
o_status_id_str":null,"in_reply_to_user_id":null,"in_reply_to_user_id_str":null,"in_reply_to_screen_name":null,"user":{"id":14391275
9,"id_str":"143912759","name":"E. Millan A.","screen_name":"EinsteinMillan","location":"The Sea Front, The Capital","url":null,"desc
ription":".:Petróleo y Gas. Mercados Energéticos. Energy Markets. Upstream Oil & Gas SME & Sr. Advisor:.","translator_type":"none","
protected":false,"verified":false,"followers_count":2399,"friends_count":990,"listed_count":8,"favourites_count":163,"statuses_count
":2403,"created_at":"Fri May 14 19:34:43 +0000 2010","utc_offset":null,"time_zone":null,"geo_enabled":true,"lang":"en","contributors
_enabled":false,"is_translator":false,"profile_background_color":"C0DEED","profile_background_image_url":"http:\/\/abs.twimg.com\/im
ages\/themes\/theme1\/bg.png","profile_background_image_url_https":"https:\/\/abs.twimg.com\/images\/themes\/theme1\/bg.png","profil
e_background_tile":false,"profile_link_color":"1DA1F2","profile_sidebar_border_color":"C0DEED","profile_sidebar_fill_color":"DDEEF6"
,"profile_text_color":"333333","profile_use_background_image":true,"profile_image_url":"http:\/\/pbs.twimg.com\/profile_images\/1079
92076052380057 7\/E42fNDAV_normal.jpg","profile_image_url_https":"https:\/\/pbs.twimg.com\/profile_images\/1079920760523800577\/E42fN
```


Như vậy ta đã hoàn thành việc viết chương trình KafkaTwitterStreaming.java để thu thập dữ liệu Twitter trong thời gian thực và truyền về Kafka (số lượng dữ liệu sẽ tăng theo thời gian do chương trình sẽ liên tục thu thập thông tin liên quan đến chủ đề cần tìm kiếm)

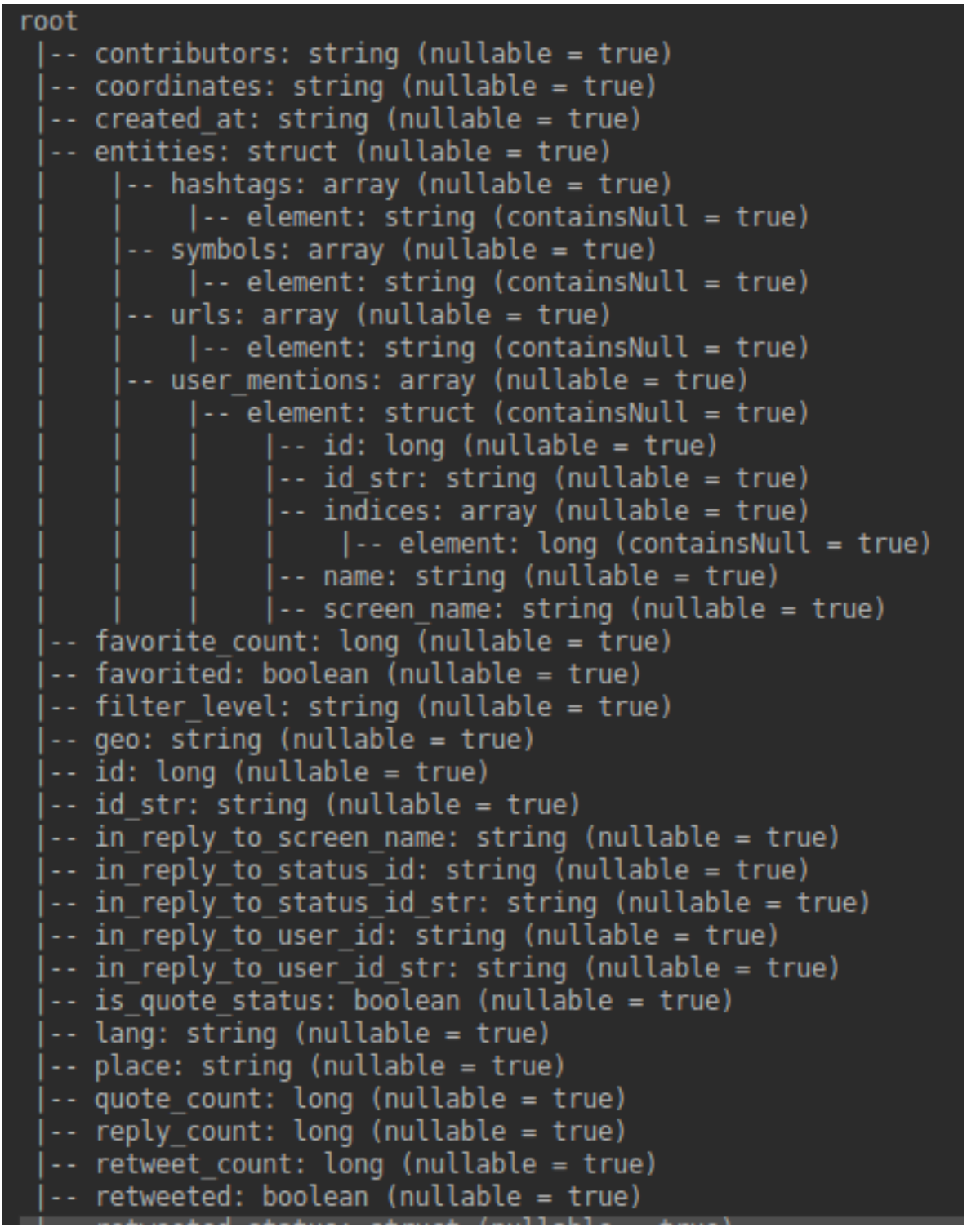
Bước tiếp theo ta viết chương trình KafkaSparkSS.scala để lấy dữ liệu về từ Kafka broker. Ta thực hiện việc kết nối Spark với Kafka tương tư như trong bài tutorial Tích hợp Kafka với Spark sử dụng Structured Streaming

```
1  //Define a Spark session
2  val spark=SparkSession.builder().appName("Spark Kafka Integration using Structured Streaming")
3                  .master("local")
4                  .getOrCreate()
5
6  //Set the Log file level
7  spark.sparkContext.setLogLevel("WARN")
```

```
8
9   //Implicit methods available in Scala for converting common Scala objects into DataFrames
10  import spark.implicits._
11
12  //Subscribe Spark to topic 'TwitterStreaming'
13  val df=spark.readStream.format("kafka")
14    .option("kafka.bootstrap.servers","localhost:9092")
15    .option("subscribe","TwitterStreaming")
16    .load()
```

Như đã đề cập ở trên, dữ liệu truyền về là dưới dạng Json và Spark Structured Streaming lại không hỗ trợ việc tự động trích xuất Schema của dữ liệu Kafka. Do đó ta phải tiến hành định nghĩa Schema này trong chương trình KafkaSparkSS.scala. Chúng ta có thể định nghĩa Schema theo cách thủ công như trong bài Phân tích và thống kê dữ liệu sử dụng Spark DataFrame, tuy nhiên Schema của Twiter tương đối phức tạp nên ta sẽ sử dụng phương pháp dùng Spark để tự động trích xuất Schema của một mẫu dữ liệu Twitter cho trước (Ở đây chúng ta copy một mẫu dữ liệu trong phần chạy KafkaTwitterStreaming.java và lưu thành file twitter.json rồi sử dụng đoạn code sau để trích xuất Schema)

```
1   //Extract the schema from a sample of Twitter Data
2   val twitterData=spark.read.json("src/main/resources/data_source/twitter.json").toDF()
3   val twitterDataScheme=twitterData.schema
```



Sau khi đã xác định được Schema của dữ liệu Twitter, ta tiến hành đọc dữ liệu nhận về với Schema này như sau:

```
1  //Reading the streaming json data with its schema
2  val twitterStreamData=df.selectExpr( "CAST(value AS STRING) as jsonData")
```

```
3     .select(from_json($"jsonData",schema = twitterDataScheme).as("data"))
4     .select("data.*")
```

Sau đó ta tiến hành hiển thị dữ liệu Spark lấy về từ Kafka với đoạn code sau:

```
1     // Display output (all columns)
2     val query = twitterStreamData
3       .writeStream
4       .outputMode("append")
5       .format("console")
6       .start()
7
8     // Display output (only few columns)
9     val query2 = twitterStreamData.select("created_at","user.name","text","user.lang")
10      .writeStream
11      .outputMode("append")
12      .format("console")
13      .start()
14
15    query.awaitTermination()
16    query2.awaitTermination()
```

Chạy chương trình KafkaSparkSS.scala ta được kết quả sau: