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

Top of Form

Bottom of Form

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](http://itechseeker.com/tutorials/apache-kafka/lap-trinh-voi-apache-kafka/su-dung-kafka-voi-twitter/) và [Tích hợp Kafka với Spark sử dụng Structured Streaming](http://itechseeker.com/tutorials/apache-kafka/lap-trinh-voi-apache-kafka/tich-hop-kafka-voi-spark-su-dung-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](http://itechseeker.com/tutorials/apache-kafka/lap-trinh-voi-apache-kafka/su-dung-kafka-voi-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:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | <!-- 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](http://itechseeker.com/tutorials/apache-kafka/lap-trinh-voi-apache-kafka/su-dung-kafka-voi-twitter/) với hai hàm sau (ta thêm .setJSONStoreEnabled(true) để truyền dữ liệu về Kafka dưới dạng Json):

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39 | 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();      } |

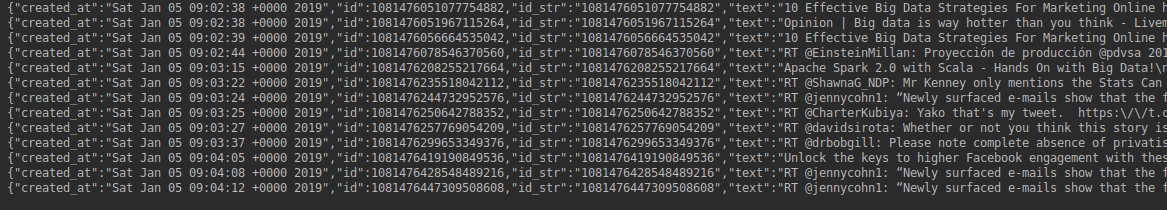
Tiếp theo ta viết hàm getStreamTweets() để thu thập dữ liệu Twitter trong thời gian thực như sau:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | 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](http://itechseeker.com/tutorials/apache-kafka/)):

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

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), …):



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):



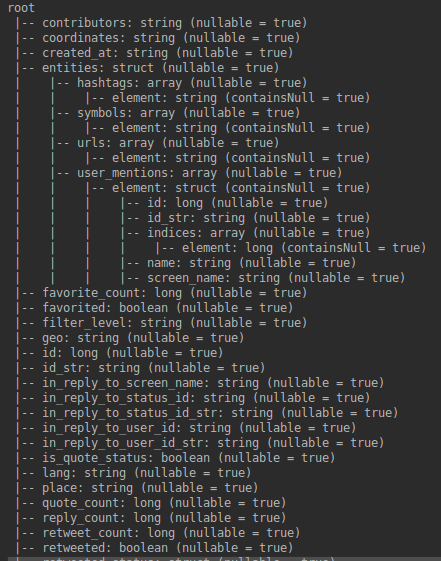
Như vậy ta đã hoàn thành việc viết chương trình KafkaTwitterStreaming.java để thu thập dữ liệu Twitter trong thời gian thực và truyền về Kafka (số lượng dữ liệu sẽ tăng theo thời gian do chương trình sẽ liên tục thu thập thông tin liên quan đến chủ đề cần tìm kiếm)

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](http://itechseeker.com/tutorials/apache-kafka/lap-trinh-voi-apache-kafka/tich-hop-kafka-voi-spark-su-dung-structured-streaming/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | //Define a Spark session      val spark=SparkSession.builder().appName("Spark Kafka Integration using Structured Streaming")                                      .master("local")                                      .getOrCreate()        //Set the Log file level      spark.sparkContext.setLogLevel("WARN")        //Implicit methods available in Scala for converting common Scala objects into DataFrames      import spark.implicits.\_        //Subscribe Spark to topic 'TwitterStreaming'      val df=spark.readStream.format("kafka")        .option("kafka.bootstrap.servers","localhost:9092")        .option("subscribe","TwitterStreaming")        .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](http://itechseeker.com/tutorials/apache-spark/lap-trinh-spark-voi-intellij-idea/spark-sql-dataset-va-dataframes/phan-tich-va-thong-ke-du-lieu-su-dung-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  2  3 | //Extract the schema from a sample of Twitter Data      val twitterData=spark.read.json("src/main/resources/data\_source/twitter.json").toDF()      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  2  3  4 | //Reading the streaming json data with its schema      val twitterStreamData=df.selectExpr( "CAST(value AS STRING) as jsonData")        .select(from\_json($"jsonData",schema = twitterDataScheme).as("data"))        .select("data.\*") |

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

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

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

