# Homework 4

Aim of this homework is to get familiar with Apache Kafka components, Kafka’s native client API and obtain basic knowledge about Kafka and Spark integration.

## Part 1

The first part is to install **Apache Kafka** and start the very first Kafka broker (Kafka server).

1. Make sure that you have installed the latest version of **Java 11**
2. Download **Apache Kafka 2.4.1**: [Download](http://kafka.apache.org/downloads) page, use the binary download for **Scala 2.11**
3. Follow [Quick Start](http://kafka.apache.org/documentation/#quickstart) document from the [offical documentation of Apache Kafka](http://kafka.apache.org/documentation/). Do the steps up to and including the **Step 5: Start a consumer**

Or you can use Docker:

1. Use [kafka-docker](https://github.com/wurstmeister/kafka-docker) Docker image
2. Follow the steps at [Pre-Requisites](https://github.com/wurstmeister/kafka-docker#pre-requisites) and [Usage](https://github.com/wurstmeister/kafka-docker#usage)
3. Send messages from within a Docker container

## Part 2

The seconds part is to create Producer/Consumer applications for a specific dataset:

1. Create a new Kafka topic called **clickstream** with multiple partitions.
2. Create custom data generator using [Kafka Producer API](https://kafka.apache.org/documentation/#producerapi).   
   You need to create a standalone Java/Scala application that continuously generates Clickstream events to the topic.

Please download sample data from here: <https://s3-eu-west-1.amazonaws.com/yc-rdata/yoochoose-data.7z>

Given a sequence of click events performed by some user during a typical session in an e-commerce website. The file **yoochoose-clicks.dat** (CSV file with coma delimiter) comprising the clicks of the users over the items. Each record/line in the file has the following fields/format: Session ID, Timestamp, Item ID, Category

* **Session ID** – the id of the session. In one session there are one or many clicks. Could be represented as an integer number.
* **Timestamp** – the time when the click occurred. Format of YYYY-MM-DDThh:mm:ss.SSSZ
* **Item ID** – the unique identifier of the item that has been clicked. Could be represented as an integer number.
* **Category** – the context of the click. The value "S" indicates a special offer, "0" indicates a missing value, a number between 1 to 12 indicates a real category identifier, any other number indicates a brand. E.g. if an item has been clicked in the context of a promotion or special offer then the value will be "S", if the context was a brand i.e. BOSCH, then the value will be an 8-10 digits number. If the item has been clicked under regular category, i.e. sport, then the value will be a number between 1 to 12.

Read data from **yoochoose-clicks.dat** file and store it to the topic, using Session ID as the key and all other fields as value.

Ensure that data related to the same **Session ID** is written to the same Kafka partition in the right occurrence order (**Timestamp**)

1. Create custom data reader using [Kafka Consumer API](https://kafka.apache.org/documentation/#consumerapi).   
   You need to build a standalone Java/Scala application that continuously consumes data from the topic:
   1. enrich the data with an additional field ‘**Country’**. Please implement an util function that returns a random country code (US, GB, DE etc.) based on the session id. Events with the same session id have to get the same ‘Country’ field.
   2. create two additional topics: **rich\_clickstream**, **errorstream**. Write enriched data to **rich\_clickstream** topic if **Category** is a valid category (it can be ‘s’, 1-12 or 8-10 digits number), all other data rows should be written to **errorstream** topic.
   3. you may want to write message in JSON to **rich\_clickstream** or in any other format you find appropriate.
2. Implement custom topic offset management. For example, you can store offsets in a local file or an external DB. You need to implement exactly once semantic, so we won’t get any data duplicates in **rich\_clickstream** topic if we restart the consumer application. If you won’t be able to achieve exactly once semantic, write in comments in code why so and which semantic does your program have.

**Optional**: implement data manipulations using [Kafka Streams API](https://kafka.apache.org/documentation/streams/)

## Part 3

The third part is to create a Spark application to consume data from the topic:

1. Read data from **rich\_clickstream** topic using [Spark Structured Streaming](https://spark.apache.org/docs/2.4.7/structured-streaming-programming-guide.html)
2. Write data to HDFS as json files partitioned by **Date** in yyyy-MM-dd format and by **Country**, so the data should be stored in different folder based on timestamp date + country code (/2014-08-01/US/file-part-\*)