iscte TECNOLOGIAS E ARQUITETURA

Algoritmos para Big Data
Lab class handout #5

Apache Kafka as a messaging system, and data profiling **2024/2025**

Learning objectives

This lab class aims to introduce Apache Kafka as a messaging system that may play an important role in data streaming. Additionally, profiling of data to be used is considered, based upon reports generated by a specific tool – YData Profiling.

Note: It is acceptable that the work to be undertaken may require some time beyond the class. If that is the case, work should continue outside of class. And if needed, assistance will be provided during the office hours established or to be agreed with the lecturer.

Supporting information

- Course slides
- Apache Kafka
- Apache Kafka quickstart
- Python client kafka-python
- YData Profiling

Introduction

Apache Kafka is an open-source, distributed event streaming platform designed for high-throughput, low-latency data processing. It is mainly used for real-time data streaming, enabling applications to process and respond to data as it arrives. It provides scalability, reliability and fault tolerance through data replication across multiple nodes. Also, it decouples systems, i.e., it decouples data producers and consumers, which simplifies system architecture and improves flexibility.

Regarding the dataset to be used in this lab class, it is needed an archive file located at:

`https://bigdata.iscte-iul.eu/datasets/books-amazon.zip`

Alongside this handout describing the tasks to accomplish, it is provided: (i) a notebook to deal with data preparation and (ii) a set of Python files related to the workings of Apache Kafka. Whenever is needed, both notebook and Python files are expected to be adjusted/completed in order to accomplish the tasks.

Tasks to accomplish

A. Data ingestion

In this initial task, execute the following operations:

1. Download the archive file already mentioned and save it in a proper directory, within the working directory of your Pyspark instalation.

2. After creating a *SparkSession* to work with, read the data into a DataFrame, let us call it `df`, and then check it.

B. Data profiling

In this task, execute the following operations

1. Use Spark instructions to check critical aspects of `df` previously set: duplicated rows and existence of NULLs in the columns.

2. With tool YData Profiling, generate a profile report concerning the above `df`. At this point, and in order to speed up computations, do not require information regarding correlations among columns. Save the report as an html file and then, through a browser, open the file and analyze the generated profile.

C. Data cleaning

In this task, the goal is to carry out operations over `df` so we may guarantee that clean data is available to be used at later stages. Hence:

1. Analyze in detail the profile report that was generated.

2. If needed and/or advisable, carry out cleaning operations over `df`.

3. Next, make sure that the new and transformed DataFrame – the outcome of the cleaning operations just mentioned – is saved into files, in two formats: parquet and csv.

D. Apache Kafka installation

In this task, the goal is to set up Apache Kakfa. There are many routes available, as pointed out in the link provided in the supporting information, Apache Kafka quickstart. Let us opt for installing Apache Kakfa in the local machine. Hence, follow the steps:

1. In terminal inside VS Code, download the latest release and then explode the archive:

```
$ wget https://dlcdn.apache.org/kafka/4.0.0/kafka_2.13-4.0.0.tgz
$ tar -xzf kafka_2.13-4.0.0.tgz
```

2. In the directory that was created, start the Kafka environment using the downloaded files. (See also steps 2, 3, 4 and 5 of the instructions in *Apache Kafka quickstart*).

```
$ KAFKA_CLUSTER_ID="$(bin/kafka-storage.sh random-uuid)"
$ bin/kafka-storage.sh format --standalone -t $KAFKA_CLUSTER_ID \
                  -c config/server.properties
$ bin/kafka-server-start.sh config/server.properties
```

E. From data producer to data consumer manually

In this task we will test sending and receiving messages manually. But first let us connect to the docker environment in a local terminal (outside VS Code)

1. Check the running docker image ID:

```
$ docker ps
```

2. Connect to the running docker (substitute $< container\_id$ with correct ID):

```
docker exec -w /home/jovyan/code -it <container\_id> bash
```

3. Repeat the last step to have two working terminals.

   In one of these windows, create a topic to store events. Do not forget to get into the kafka directory. Run the following commands:

   ```
   $ bin/kafka-topics.sh --create --topic mytest --bootstrap-server localhost:9092
   $ bin/kafka-topics.sh --describe --topic mytest --bootstrap-server localhost:9092
   ```

   Start sending/receiving data with the following commands, one in each window:

   ```
   $ bin/kafka-console-producer.sh --topic mytest --bootstrap-server localhost:9092
   $ bin/kafka-console-consumer.sh --topic mytest --from-beginning \
   --bootstrap-server localhost:9092
   ```

F. From data producer to data consumer

In this task, we are going to send the cleaned data mentioned above, that was stored into a file, (see task C) to a local Kafka cluster. Conversely, such data will be consumed from the Kafka cluster. Notice that there is no need to use Apache Spark in this task.

Hence, execute the following operations:

1. Create a directory, let us call it *Kafka*, where we will be working on. Then, in that directory, install the Python client kafka-python:

   ```
   $ pip install kafka-python
   ```

2. Copy into the Kafka directory mentioned above the files provided with Python code:

   `kafka-cluster-info.py, kafka-cluster-create-topic.py`
   `kafka-producer.py, kafka-consumer.py`

   Check the code and complete/adjust if needed. Its execution will allow sending data from the data file to a Kafka cluster, and then to be consumed from the Kafka cluster.

3. Make sure that a Kafka cluster running by running:

   ```
   $ python3 kafka-cluster-info.py
   ```

   Alternatively, in the directory where Kafka has been installed, execute:

   ```
   $ ./bin/kafka-topics.sh --list --bootstrap-server localhost:9092
   ```

4. Create a new topic: by running:

   ```
   $ python3 kafka-cluster-create-topic.py
   ```

5. Open two terminals, each one of them positioned in the directory `Kafka`. Then, in one terminal execute the command:

   ```
   $ python3 kafka-producer.py
   ```

   Likewise, in the other terminal, execute the command:

   ```
   $ python3 kafka-consumer.py
   ```

   Check the outcome. You can stop the running processes by executing **Crtl-C** in both terminals.

6. Finally, terminate the Kafka environment. To do so, stop the Kafka broker with **Crtl-C**.

   Furthermore, in order to delete any data of the local Kafka environment, e.g. events that were created, run the command:

   ```
   $ rm -rf /tmp/kafka-logs /tmp/kraft-combined-logs
   ```

As a note, use this last command if you are getting problems starting Kafka.