UNIVERSIDADE DE COIMBRA FACULDADE DE CIÊNCIAS E TECNOLOGIA ***Departamento de Engenharia Informática***	**Project #3** **Integração de Sistemas/** **Enterprise Application Integration** **2019/20 – 1st Semester** MEI, MEIG **Deadline: 2019-12-13**

Nota: A fraude denota uma grave falta de ética e constitui um comportamento não admissível num estudante do ensino superior e futuro profissional. Qualquer tentativa de fraude pode levar à reprovação na disciplina tanto do facilitador como do prevaricador.

MUITO IMPORTANTE: o código entregue pelos alunos vai ser submetido a um sistema de deteção de fraudes.

VERY IMPORTANT: the code delivered by students will be submitted to a fraud detection system.

Message Oriented Middleware (MOM) and Kafka Streams

Objectives

- Learn how to create simple asynchronous and message-oriented applications.
- Learn to use Kafka Streams.

Resources

Apache Kafka Introduction: https://kafka.apache.org

Kafka Streams: https://kafka.apache.org/documentation/streams/

Apache Kafka Tutorial:
https://www.tutorialspoint.com/apache_kafka/apache_kafka_simple_producer_example.htm

Look for the Kafka and Kafka Streams materials available on Inforestudante.

Make sure you understand the following concepts:

- Producer
- Consumer
- Topic

- Partition and partition offset
- Broker
- Zookeper

Kafka Training (doesn't count for evaluation)

1. You should start by installing Kafka.

2. You may look at the reading material suggested before and follow that material to have a basic example running with a producer and consumer.

3. What happens to the messages that arrive at the topic, before the subscriber makes the subscription?

4. How do you change the type of data of the keys and values that you send?

5. How do you configure different partitions inside a topic?

6. What is the point of Consumer Groups? How do they work?

7. Now, read the Kafka Streams tutorial and run the example that counts the words that go through a stream.

8. Refer to the following blog message for this and the following exercises:

   https://eai-course.blogspot.com/2018/11/playing-with-kafka-streams.html

   Use Kafka Streams to count the events that happened for a given key. Display the result as 'k->v' in the final stream.

9. Now sum all the values that showed up for each key.

10. Use a materialized view to do range queries on the sums on a separate service.

11. Now, control the time considered in the stream to be 1 minute.

12. How could you send complex data instead of primitive types?

13. Use the Kafka Connect application to periodically fetch data from a database table (source). You can find help for this operation in the following blog message:

    https://eai-course.blogspot.com/2019/11/how-to-configure-kafka-connectors.html

14. Now do the inverse operation: send from a topic to a database table (sink).

Project Description (for evaluation)

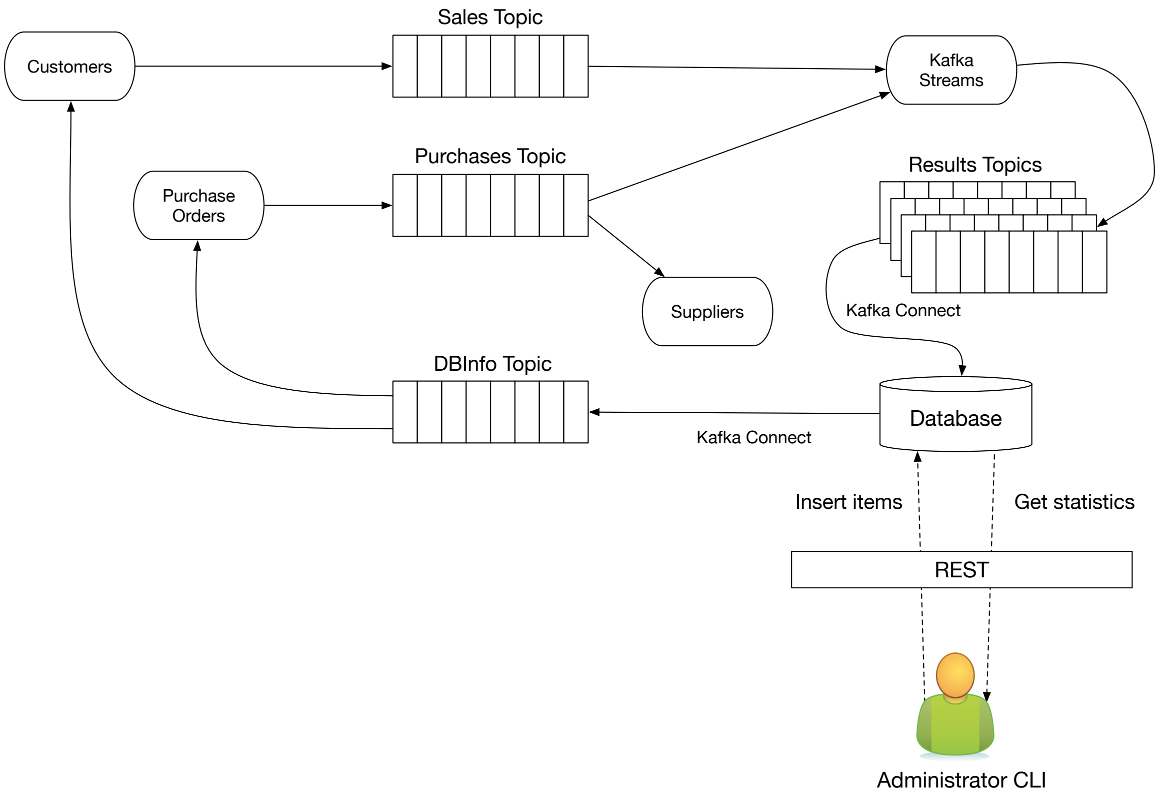

Figure 1 - Overview of the Shop

Refer to Figure 1 summarizing a shop in Kafka. The shop has the following Kafka topics:

- **DBInfo**, where one or more source connectors write a list of countries and a list of items for sale from the database.
- **Purchases** topic, where a process simulating purchases to suppliers keeps writing orders of items listed in the DBInfo topic. Each purchase item includes a price and a number of units. Students should randomly select the item to purchase, its price and the units.
- **Sales** topic, where a process simulating customers keeps writing. Each sale has an item, a price, a number of units and a country of origin. All of these are randomly generated, but the items and countries must be listed in the DBInfo topic. Students need not to be concerned about whether the stock for each item is positive or if the shop is profitable.
- One or more applications using Kafka streams will listen to these topics and compute a number of metrics, as enumerated in the requirements, including revenues, expenses, profit, etc. This or these applications will then write their

computations to another topic, the **Results** topics, from where a Kafka connector will write the results back to the database.

A server-side application will read data from the database and expose the data via REST. The final piece of the application is a command line application that allows the administrator to enter items, countries and read statistics from the REST services. No authentication is necessary.

Components to Develop

Students need to develop the following applications of Figure 1: Customers, Purchase Orders, Kafka Streams, the administrator CLI and the REST implementation. From this list, all the former are stand-alone applications, while the latter is a server-side application that must be deployed on a server. They must also configure Kafka connectors to automatically extract and write data from/to the database.

With the exception of the data that was mentioned before, the precise definition of the communication format is left for the students to determine. Clean solutions that serialize complex data structures as JSON strings are encouraged.

Requirements

As an administrator I can perform the following actions:

1. Add countries to the database. To simplify countries cannot be deleted and optionally not changed.
2. List countries from the database.
3. Add items for sale in the shop to the database. Again, these cannot be deleted, but may be changed if students wish.
4. List items from the database.
5. Get the revenue per item.
6. Get the expenses per item.
7. Get the profit per item.
8. Get the total revenues.
9. Get the total expenses.
10. Get the total profit.
11. Get the average amount spent in each purchase (separated by item).
12. Get the average amount spent in each purchase (aggregated for all items).
13. Get the item with the highest profit of all (only one if there is a tie).
14. Get the total revenue in the last hour[1] (use a tumbling time window).
15. Get the total expenses in the last hour (use a tumbling time window)
16. Get the total profits in the last hour (use a tumbling time window).
17. Get the name of the country with the highest sales per item. Include the value of such sales.

Students should include means in their application to enable a fast verification of the results they are displaying. Solutions should maximize the computations they do with Kafka streams and reduce the database queries to their simplest possible form (for

[1] Students may change this interval freely for testing and demonstration purposes.

example, students should use Kafka Stream to compute profits, instead of computing them on the database).

Final Delivery

- The project should be made in groups of 2 students. I do expect all the members of the group to be fully aware of all the parts of the code that is submitted. Work together!
- To automate the build of the project, students should use Maven.
- Students should submit the project part that is for evaluation in a zip file with the **source** of a complete Maven project, using Inforestudante. Do not forget to associate your work colleague during the submission process.
- Grading is performed according to a grid that will be published later and according to individual student defenses.
- No report is necessary.

Good Work!