Laboratório 2 sobre Spark/Kafka (com API Gráfica)

A) Objetivo: O objetivo deste experimento é que o aluno compreenda a configurar saída gráfica para processamento Spark/kafka. Para isto, deve criar dois canais kafka, sendo um de entrada e um outro de saída, sendo que este último deve ser associado a um conector ElasticSearch, conforme roteiro a seguir.

B) Roteiro do laboratório

1. Configurando os canais Kafka:

a) Criar o canal kafka de saída (interação com o Kibana) pelo comando abaixo:

```
$ kafka-topics.sh --create --topic student-a19000-saida --bootstrap-server cm3:9092
```

b) Faça um teste do diálogo produtor/consumidor usando o tópico criado

```
$ kafka-console-consumer.sh -topic student-a19000-saida --bootstrap-server cm3:9092
$ kafka-console-producer.sh -topic student-a19000-saida --bootstrap-server cm3:9092
```

c) Na console do produtor, digite algo e veja como as mensagens são ecoadas no consumidor.

2. Configurando a conexão do canal Kafka com o ElasticSearch/Kibana

a) Em uma pasta específica, crie os arquivos connect.properties e elastic.properties, conforme o template a seguir, alterando os campos que estão em itálico/negrito:

```
name=elasticsearch-sink-<usuario>
connector.class=io.confluent.connect.elasticsearch.ElasticsearchSinkConnector
tasks.max=1
topics=student-<usuario>-<indice>
key.ignore=true
schema.ignore=true
connection.username=<usuario>
connection.password=<senha>
connection.url=http://cm1:9200
type.name=kafka-connect
drop.invalid.message=true
behavior.on.null.values=IGNORE
behavior.on.malformed.documents=IGNORE
errors.tolerance=all
```

```
bootstrap.servers=cm3:9092
key.converter=org.apache.kafka.connect.json.JsonConverter
value.converter=org.apache.kafka.connect.json.JsonConverter
key.converter.schemas.enable=false
value.converter.schemas.enable=false
internal.key.converter=org.apache.kafka.connect.json.JsonConverter
internal.value.converter=org.apache.kafka.connect.json.JsonConverter
internal.key.converter.schemas.enable=false
internal.value.converter.schemas.enable=false
offset.storage.file.filename=/tmp/connect.offsets
offset.flush.interval.ms=10000
listeners=http://0.0.0.0:<porta_do_conector>
```

---

b) Após a configuração dos arquivos, faça a seguinte chamada:

```
$ connect-standalone.sh connect.properties elastic.properties
```

3. Monte o script de processamento Spark/Kafka/ElasticSearch/Kibana:

Para compreender essa integração, vamos alterar o experimento anterior, de modo que o consumidor seja um *script* do Spark, usando um notebook

a) Utilizando a interface notebook, crie as seguintes células:

```
from pyspark.sql import SparkSession
```

```
SPARK_MASTER="spark://cm1:7077"
APP_NAME="Exemplo Spark, Kafka e ElasticSearch"

KAFKA_HOST="cm3:9092"

KAFKA_TOPIC_IN="student-a19000-entrada"
KAFKA_TOPIC_OUT="student-a19000-saida"

INTERVAL = "10 seconds"
```

```
spark = SparkSession.builder.master(SPARK_MASTER).appName(APP_NAME).getOrCreate()
```

```
messages = spark.readStream.format("kafka").option("kafka.bootstrap.servers", KAFKA_HOST) \
                .option("subscribe", KAFKA_TOPIC_IN) \
                .option('includeTimestamp', 'true').load()
```

```
from pyspark.sql.functions import explode, split, col, upper, window, to_json, struct, lit
```

```
words = messages \
            .select(
                explode(split(col("value"), "\s+")).alias("word"),
                messages.timestamp
            ).select(
                upper(col("word")).alias('word'),
                col("timestamp")
            )
```

```
wordCounts = words.withWatermark("timestamp", INTERVAL) \
                    .groupBy(
                        window(words.timestamp, INTERVAL, INTERVAL),
                        "word"
                    ).count()
```

```
wordCountsJson = wordCounts.select(
                                lit('1').alias('key'),
                                to_json(struct("word", "count")).alias('value')
                            )
```

```
wck = wordCountsJson.writeStream.outputMode("update").format("kafka").option("kafka.bootstrap.servers", KAFKA_HOST)
        .option('topic', KAFKA_TOPIC_OUT).option('checkpointLocation', '/tmp/spark-a190020377') \
        .trigger(processingTime=INTERVAL) \
        .start()
```

b) Entre no Kibana pelo navegador (http://cm4:5601), configure o Dashboard, os gráficos associados e vincule-os ao canal de saída onde o Spark está jogando os valores contabilizados.

C) Atividades de entrega:

Após a realização do experimento, gerar a documentação na forma de um relatório, contemplando todos os passos executados, especialmente para os tipos de gráficos e saídas utilizados. Este relatório, em formato pdf e o script Python notebook devem ser compactados e postados no Moodle.