Dead Letter Queue

Implement `DeserializationExceptionHandler` in class

- Override `configure` method and create a producer
- Override `handle` method, send corrupted record to `quarantine` topic and return `DeserializationHandlerResponse.CONTINUE` to continue processing of messages.

Set the `default.deserialization.exception.handler` class in StreamsConfig

```
props.put(StreamsConfig.DEFAULT_DESERIALIZATION_EXCEPTION_HANDLER_CLASS_CONFIG, "com.admatic.DLQ");
```

DLQ.java

```
vim src/main/java/com/admatic/DLQ.java
```

```
package com.admatic;
```

```
import org.apache.kafka.clients.consumer.ConsumerRecord;
import org.apache.kafka.clients.producer.KafkaProducer;
import org.apache.kafka.clients.producer.ProducerRecord;
import org.apache.kafka.common.serialization.Serdes;
import org.apache.kafka.streams.KafkaStreams;
import org.apache.kafka.streams.StreamsBuilder;
import org.apache.kafka.streams.StreamsConfig;
import org.apache.kafka.streams.errors.DeserializationException
Handler;
import org.apache.kafka.streams.processor.ProcessorContext;

import java.util.HashMap;
import java.util.Map;
import java.util.Properties;

public class DLQ implements DeserializationExceptionHandler {
    private KafkaProducer<byte[], byte[]> dlqProducer;

    @Override
    public DeserializationHandlerResponse handle(final Processo
rContext context,
                                                 final Consumer
Record<byte[], byte[]> record,
                                                 final Exceptio
n exception) {

        System.out.println("Exception caught during Deserializa
tion, sending to the dead queue topic; " +
                "taskId: {" + context.taskId() + "}, " +
                "topic: {" + record.topic() + "}, " +
                "partition: {" + record.partition() + "}, " +
                "offset: {" + record.offset() + "}\n\n"
                + exception);

        String dlqTopic = "quarantine";
        dlqProducer.send(new ProducerRecord<>(dlqTopic, record.
key(), record.value()));

        return DeserializationHandlerResponse.CONTINUE;
    }
    @Override
    public void configure(final Map<String, ?> configs) {
        Properties props = new Properties();
        props.put("bootstrap.servers", "localhost:9092");
        props.put("acks", "all");
        props.put("retries", 0);
        props.put("batch.size", 16384);
        props.put("linger.ms", 1);
        props.put("buffer.memory", 33554432);
        props.put("max.request.size", "99999999");
        props.put("key.serializer", "org.apache.kafka.common.se
rialization.ByteArraySerializer");
        props.put("value.serializer", "org.apache.kafka.common.
serialization.ByteArraySerializer");

        dlqProducer = new KafkaProducer<>(props);
    }

    public static void main(String[] args) {
        if (args.length != 2) {
            System.out.println("Usage: <input_topic> <output_to
pic>");
            return;
        }

        Map<String, Object> props = new HashMap<>();
        props.put(StreamsConfig.APPLICATION_ID_CONFIG, "my-stre
am-processing-application");
        props.put(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "loca
lhost:9092");
        props.put(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG,
 Serdes.Integer().getClass());
        props.put(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFI
G, Serdes.String().getClass());
        props.put(StreamsConfig.DEFAULT_DESERIALIZATION_EXCEPTI
ON_HANDLER_CLASS_CONFIG, "com.admatic.DLQ");
        StreamsConfig config = new StreamsConfig(props);

        StreamsBuilder builder = new StreamsBuilder();
        builder.<Integer, String>stream(args[0]).mapValues(valu
e -> value.length() + "").to(args[1]);
```

```
        KafkaStreams streams = new KafkaStreams(builder.build()
, config);
        streams.start();
    }
}
```

Create input, output and quarantine topics

```
kafka-topics.sh --delete --zookeeper localhost:2181 --topic inp
ut-topic
kafka-topics.sh --delete --zookeeper localhost:2181 --topic out
put-topic
kafka-topics.sh --delete --zookeeper localhost:2181 --topic qua
rantine

kafka-topics.sh --create --zookeeper localhost:2181 --topic inp
ut-topic --partitions 1 --replication-factor 1
kafka-topics.sh --create --zookeeper localhost:2181 --topic out
put-topic --partitions 1 --replication-factor 1
kafka-topics.sh --create --zookeeper localhost:2181 --topic qua
rantine --partitions 1 --replication-factor 1
```

Compile and Run the App

```
mvn compile
mvn exec:java -Dexec.mainClass="com.admatic.DLQ" -Dexec.args="i
nput-topic output-topic"
```

Verification

Normal Record

- Sending record with key as `null` when the app expects `Integer` type key

```
kafka-console-producer.sh --broker-list localhost:9092 --topic
input-topic
admatic
```

```
kafka-console-consumer.sh --bootstrap-server localhost:9092 --t
opic output-topic
7
```

Corrupted Record

- Sending record with `String` type key when the app expects `Integer` type key

```
kafka-console-producer.sh --broker-list localhost:9092 --topic
input-topic --property "parse.key=true" --property "key.separat
or=:"
a:admatic
```

```
org.apache.kafka.common.errors.SerializationException: Size of
data received by IntegerDeserializer is not 4
Exception caught during Deserialization, sending to the dead qu
eue topic; taskId: {0_0}, topic: {input-topic}, partition: {0},
 offset: {2}

org.apache.kafka.common.errors.SerializationException: Size of
data received by IntegerDeserializer is not 4
```

- View the corrupted record in `quarantine` topic

```
kafka-console-consumer.sh --bootstrap-server localhost:9092 --t
opic quarantine --from-beginning
admatic
```