Kafka Message Aggregation using Camel and Spring Boot

# Introduction

[Apache Camel](https://camel.apache.org/) is a popular open source integration framework that can work with almost any message brokers like Kafka, ActiveMQ, RabbitMQ etc. It provides [out of the box support](https://camel.apache.org/manual/latest/enterprise-integration-patterns.html) for the most popular EIPs ([Enterprise Integration Patterns](https://www.enterpriseintegrationpatterns.com/patterns/messaging/toc.html)). Camel can also work seamlessly with Spring Boot, and that makes it a killer combination. In this example, we will see how to use the [Aggregate EIP](https://camel.apache.org/manual/latest/aggregate-eip.html) provided by Camel to do message aggregation on Kafka.

# Problem Statement

We are building a microservice. It reads BankDetail messages in JSON format from the Kafka Topic *bank-details*. A BankDetail message has the below attributes:

private int id;

private int age;

private String job;

private String marital;

private String education;

private String defaulted;

private BigDecimal balance;

private String housing;

private String loan;

private String contact;

private int day;

private String month;

private int duration;

private int campaign;

private int pdays;

private int previous;

private String poutcome;

private String y;

BankDetail messages come in batches, each message of the same batch, has the same *kafka.key*. The microservice then aggregates all the messages of the same batch based on the *job* and finds out the count of various job categories. It would then publish the result of the aggregation on the Kafka Topic *bank-details-aggregated*. A typical aggregate message would look like:

{

"adminCount": 478,

"blueCollarCount": 946,

"entrepreneurCount": 168,

"houseMaidCount": 112,

"managementCount": 969,

"retiredCount": 230,

"selfEmployedCount": 183,

"servicesCount": 417,

"studentCount": 84,

"technicianCount": 768,

"unemployedCount": 128,

"unknownCount": 38

}

# Solution

## Project Setup using Maven

Its a standard Spring Boot project. We will define the Camel BOM as below:

<dependencyManagement>

<dependencies>

<!-- Camel BOM -->

<dependency>

<groupId>org.apache.camel</groupId>

<artifactId>camel-spring-boot-dependencies</artifactId>

<version>${spring.camel-version}</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

Then we will define the dependencies for Camel:

<!-- START :: Camel -->

<dependency>

<groupId>org.apache.camel</groupId>

<artifactId>camel-spring-boot-starter</artifactId>

</dependency>

<dependency>

<groupId>org.apache.camel</groupId>

<artifactId>camel-stream-starter</artifactId>

</dependency>

<dependency>

<groupId>org.apache.camel</groupId>

<artifactId>camel-kafka</artifactId>

</dependency>

<dependency>

<groupId>org.apache.camel</groupId>

<artifactId>camel-kafka-starter</artifactId>

</dependency>

<dependency>

<groupId>org.apache.camel</groupId>

<artifactId>camel-jackson-starter</artifactId>

</dependency>

<!-- END :: Camel -->

## Contract with the message publisher

We have the below contract with the BankDetail message publisher:

1. The messages will be published in JSON format on the Kafka Topic *bank-details*.
2. The *kafka.key* of all messages of the same group or batch would be identical.
3. There is no expectation as to the type of the key, as long as the keys are fairly unique across different batches. In this example, I am using a random UUID.
4. After all messages in a batch are published, a CompletionSignal is published on Kafka.

A simple BankDetail publisher that respects the above contract can be found [here](https://github.com/paawak/spring-boot-demo/tree/master/kafka-spring/kafka-simple-publisher).

## Implementation details

### Camel Route

### Aggregation Strategy

### Aggregation Implementation

# Discussion