Retry & Dead Letter Queue

메시지 처리 중 오류가 발생하면, 일정 횟수동안 재시도 하거나, 그래도 처리하지 못하는 메시지는 별도의 Queue 에 큐잉하여 관리자가 백오피스를 통해 확인 및 처리하도록 해야 하는데 이번 랩에서는 이를 학습한다.

10분 이내 완료

INTO THE LAB

Instruction

Retry & DLQ

Kafka Retry

- **Consumer** 가 **message** 를 처리하던 중 오류가 발생하면 해당 **Message** 를 다시 **Polling** 하여 처리해야 한다.
- 이를 **Retry** 라고 하며, 간단하게 **Kafka** 설정으로 동작할 수 있다.
- **Product** 마이크로서비스 **application.yml** 화일의 **cloud.stream.bindings.event-in** 하위의 설정을 주석해제하고 저장한다.

```
bindings:
  event-in:
    group: product
    destination: kafkatest
    contentType: application/json
    consumer:
      max-attempts: 5
      back-off-initial-interval: 1000
      back-off-max-interval: 1000
      back-off-multiplier: 1.0
      defaultRetryable: false
```

- **3 번의 retry 를 수행하는데 Retry 시 백오프 초기간격이 1 초, 이후 최대 1 초 간격으로 retry 를 실행한다.**
- **Product 서비스의 PolicyHandler.java 에서 아래 코드의 블럭주석을 해제하고 기존 메서드를 블럭주석 처리한다.**

```
@StreamListener(KafkaProcessor.INPUT)
    public void wheneverOrderPlaced_PrintMessage(@Payload
OrderPlaced orderPlaced) {
```

```
        System.out.println("Entering listener: " +
orderPlaced.getId());
        System.out.println("Entering listener: " +
orderPlaced.getMessage());
        throw new RuntimeException();

    }
```

- Order 와 Product 마이크로서비스를 기동한다.

```
cd order
mvn spring-boot:run
cd product
mvn spring-boot:run
```

- **Order** 서비스에 포스팅하여 **Kafka Event** 를 발행한다.
- **Product** 에서 **Message** 를 **subscribe** 하여 내용을 출력한다.
- **throw new RuntimeException** 에 의해 **Kafka retry** 가 수행되는지 **Console** 의 **log** 로 확인한다.
- 허나,
- 해당 메시지는 처리될 수 없으므로 파티션 **Lag** 가 항상 잔존하게 된다.

```
$kafka_home/bin/kafka-consumer-groups.sh --bootstrap-server
localhost:9092 --group product --describe
```

- 이는 별도의 Topic 에 저장한 후 백오피스에서 처리해야 할 대상인 것이다.

Kafka Dead Letter Queue(DLQ)

- Kafka 에서 retry 를 통해서도 처리하지 못하는 message 를 Posion pill 이라고 한다.
- Kafka 에서 Posion pill 은 별도의 메시지 저장소인 DLQ 로 보내지게 된다.
- DLQ 는 또 하나의 topic 이며 Consumer 에서 정상적으로 처리되지 못한 message 들이 쌓여있다.
- DLQ 를 설정하기 위해서 아래와 같이 Product 의 application.yml 를 변경한다.
- cloud.stream.kafka 아래에 있는 아래 설정을 주석해제 한다.

```
bindings:
  event-in:
    consumer:
      enableDlq: true
      dlqName: dlq-kafkatest
      dlqPartitions: 1
```

- 저장 후 Product 마이크로서비스를 재기동한다.

```
cd product
mvn spring-boot:run
```

Product 서비스가 기동되면서 Retry 를 반복하게 되고, 그래도 처리하지 못한 메시지를 DLQ 로 보내는 것이 Console 에 확인된다.
Sent to DLQ a message with key='null' and payload='{123, 34, 101, 118, 101, 110, 116, 84, 121, 112, 1...' received from 0

- 설정에서 지정한 DLQ 토픽이 생성되었는지 확인한다.

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

Kafka DLQ Test

- Order 서비스에 포스팅하여 Kafka Event 를 추가 발행한다.

```
http POST :8081/orders message=5th-Order
```

- Product 에서 retry 3 번 시도 후, 자동으로 DLQ 로 보낸다.
- 아래 명령어를 통해 DLQ 에 해당 message 가 쌓였는지 확인한다.

```
$kafka_home/bin/kafka-console-consumer.sh --bootstrap-server
http://localhost:9092 --topic dlq-kafkatest --from-beginning
```

- 커밋모드가 '자동'일때 Dlq 에 처리되지 않은 메세지를 보낸 후, 자동으로 Offset 을 증가시켜 Lag 가 쌓이지 않게 된다.

```
$kafka_home/bin/kafka-consumer-groups.sh --bootstrap-server
localhost:9092 --group product --describe
```

Kafka 수동모드에서의 Retry, DLQ

- 커밋모드가 수동일때에도 동일하게 동작하는지 확인한다.
- Product 서비스의 커밋모드를 수정한다.

cloud.stream.kafka 아래에 있는 Dlq 설정 위에 'autoCommitOffset: false'를 추가하고 저장한다.

```
bindings:
  event-in:
    consumer:
      autoCommitOffset: false # 이 라인만 추가
      enableDlq: true
      dlqName: dlq-kafkatest
      dlqPartitions: 1
```

- Product 서비스의 PolicyHandler.java 에서 아래 코드를 삽입하고, 기존 메서드를 블럭주석 처리한다.

```
    @StreamListener(KafkaProcessor.INPUT)
    public void wheneverOrderPlaced_PrintMessage2(@Payload
OrderPlaced orderPlaced,
                    @Header(KafkaHeaders.ACKNOWLEDGMENT)
Acknowledgment acknowledgment) {
        String str = null;
        try {
```

```
            System.out.println("Entering listener: " +
orderPlaced.getId());
            System.out.println("Entering listener: " +
orderPlaced.getMessage());

            String idx = str.substring(0, 3);   // raise Exception.
        } catch(Exception e) {
            acknowledgment.acknowledge();
            throw e;
        }
    }
```

- 저장 후 Product 마이크로서비스를 재기동한다.

```
cd product
mvn spring-boot:run
```

- Order 서비스에 포스팅하여 Kafka Event 를 추가 발행한다.

```
http POST :8081/orders message=7th-Order
```

- PolicyHandler 의 명시적 Exception Code 로 인해, Manual Commit 과 Retry, DLQ 전송이 일어난다.
- 수동커밋 모드에서도 Lag 가 남지 않으며 DLQ 에도 미처리된 메시지가 정상적으로 적재됨을 최종 확인한다.

```
$kafka_home/bin/kafka-consumer-groups.sh --bootstrap-server
localhost:9092 --group product --describe
$kafka_home/bin/kafka-console-consumer.sh --bootstrap-server
http://localhost:9092 --topic dlq-kafkatest --from-beginning
```

CheckPoints

1. 모든 요구사항을 만족하는가 ☐