**Kafka 수동 커밋**

**Kafka 커밋모드**

* Kafka default 커밋모드는 autoCommit 이다.
* Kafka 커밋모드가 auto(default) 일 때 Partition이 증가해 Rebalancing이 발생하면 커밋되지 않은 Message들은 자칫 컨슈머가 다시 Subscribe하여 중복처리할 수도 있다.

**Kafka 커밋모드 변경**

* autoCommit 설정을 false로 변경하여 수동커밋 모드로 변경한다.
* Product 마이크로서비스 application.yml 화일의 cloud.stream.kafka 하위의 설정을 주석해제하고 저장한다.

**bindings:**

**event-in:**

**consumer:**

**autoCommitOffset: false**

* Order와 Product 마이크로서비스를 기동한다.

**cd order**

**mvn spring-boot:run**

**cd product**

**mvn spring-boot:run**

현재, kafkatest 토픽의 파티션이 2개이므로, 실행한 Product 서비스 Console에 2개의 파티션이 할당되었음을 볼 수 있다.  
partitions assigned: [kafkatest-0, kafkatest-1]

**Lag 확인**

* Order 서비스에 포스팅하여 Kafka Event를 발행한다.

**http POST :8081/orders message=1st-Order**

**http POST :8081/orders message=2nd-Order**

* Product 마이크로서비스는 메시지를 소모(처리)했음에도 불구하고 Partition에서의 OffSet이 증가하지 않아 Lagging이 발생하고 있다.
* Partition Lagging 확인

**$kafka\_home/bin/kafka-consumer-groups.sh --bootstrap-server localhost:9092 --group product --describe**

**Manual Commit**

* Product 서비스에서 수동으로 ACK를 날려 Manual Commit을 해준다.
* Product 서비스의 PolicyHandler.java에서 아래 메서드의 블럭주석을 해제하고 기존 메서드를 블럭주석 처리한다.

**@StreamListener(KafkaProcessor.INPUT)**

**public void wheneverOrderPlaced\_PrintMessage(@Payload OrderPlaced orderPlaced, @Header(KafkaHeaders.ACKNOWLEDGMENT) Acknowledgment acknowledgment) {**

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

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

**acknowledgment.acknowledge();**

**}**

* Product 마이크로서비스를 재시작한다.
* Console 로그를 조회하면 메시지가 재처리한 것이 확인된다.
* Order 서비스에 포스팅하여 Kafka Event를 추가 발행한다.

**http POST :8081/orders message=3rd-Order**

**http POST :8081/orders message=4th-Order**

**Lag 확인**

* + Partition Lagging을 재확인하면 이제는 Lagging이 확인되지 않는다.

**$kafka\_home/bin/kafka-consumer-groups.sh --bootstrap-server localhost:9092 --group product --describe**

Checkpoints

1. 모든 요구사항을 만족하는가

Hints