Round 8 Kafka 라이브세션

1. 카프카 기본 개념 & 설치
   - 목표: 카프카가 브로커, 토픽, 파티션, 컨슈머 그룹 개념으로 동작한다는 것 이해
   - 실습 포인트: Zookeeper 없이 Kafka 2.8+ 단일 브로커 설치 → 토픽 생성 → 메시지 생산/소비 테스트
   - 실행 가이드
   - Docker로 카프카 단일 브로커 실행
   - kafka-topics.sh --create 로 토픽 생성
   - kafka-console-producer.sh, kafka-console-consumer.sh 로 메시지 송수신

---

2. 파티션 & 오프셋 이해
   - 목표: 파티션 단위 순서 보장, 오프셋 개념 학습
   - 실습 포인트: 토픽 파티션 수=1, 파티션 수=3 으로 나눠서 메시지 순서 비교
   - 실행 가이드
   - 파티션 1개 토픽에서 메시지 순서 확인
   - 파티션 3개 토픽에서 메시지 순서와 오프셋 관찰
   - 동일 키(Partition Key) 설정 시 순서 보장되는지 확인

---

3. 프로듀서 acks & 내구성 테스트
   - 목표: acks 설정(0, 1, all)에 따른 내구성·성능 트레이드오프 이해
   - 실습 포인트: 브로커 중지 시 메시지 유실 여부 관찰
   - 실행 가이드
   - acks=0 → 빠르지만 유실 가능성
   - acks=1 → 리더만 확인
   - acks=all → 모든 ISR 확인, 브로커 하나 죽어도 유실 없음 확인

---

4. 컨슈머 그룹 & 리밸런싱
   - 목표: 컨슈머 그룹, 파티션 재할당(Rebalancing) 동작 학습
   - 실습 포인트: 컨슈머 수 증감 시 파티션 재할당 확인

- 실행 가이드
- 컨슈머 1개 → 2개 → 3개 순서대로 실행 후 파티션 할당 변화 확인
- 한 컨슈머 종료 시 리밸런스 로그 관찰

---

5. 오프셋 커밋 & 재처리
   - 목표: 오프셋 자동 커밋 vs 수동 커밋 차이 이해
   - 실습 포인트: 컨슈머 프로세스 강제 종료 후 재시작 시 메시지 재처리 여부
   - 실행 가이드
   - enable.auto.commit=true 로 실행 후 중간 종료
   - enable.auto.commit=false + 수동 커밋으로 실행해 재처리 차이 비교

---

6. 파티션 키 & 순서 보장
   - 목표: 파티션 키를 통한 순서 보장 학습
   - 실습 포인트: 동일 키와 랜덤 키로 전송했을 때 순서 차이 확인
   - 실행 가이드
   - key=주문ID 로 보내 순서 유지 확인
   - 랜덤 키로 보내 순서 깨지는지 확인

---

7. 리플리케이션 & ISR/OSR
   - 목표: 복제 팩터(RF), ISR(In-Sync Replica) 개념 이해
   - 실습 포인트: 브로커 장애 시 ISR/OSR 변화를 관찰
   - 실행 가이드
   - RF=3 토픽 생성, 브로커 1대 중지 후 ISR 감소 확인
   - min.insync.replicas=2 설정 후 2대 중지 시 쓰기 실패 확인

---

8. 리밸런싱 전략 & 세션 타임아웃
   - 목표: 리밸런싱 트리거 조건 이해
   - 실습 포인트: 컨슈머 세션 타임아웃 설정 변경 후 장애 상황 테스트
   - 실행 가이드

- session.timeout.ms 짧게 설정 후 컨슈머 일시 중지 → 리밸런스 발생 확인
- CooperativeSticky 전략으로 리밸런스 영향 줄이기

---

9. 로그 보존 & 압축 정책
   - 목표: 로그 세그먼트 삭제, Compact 정책 이해
   - 실습 포인트: retention.ms, cleanup.policy=compact 비교
   - 실행 가이드
   - 1분 보존 설정 후 메시지 자동 삭제 확인
   - Compact 정책 적용 후 키별 최신 값만 유지되는지 확인

---

10. 장애 복구 & Exactly Once
    - 목표: Idempotent Producer, Transactional Producer 학습
    - 실습 포인트: 장애 상황에서 중복/유실 없는 처리 확인
    - 실행 가이드
    - enable.idempotence=true 로 중복 제거 확인
    - transactional.id 로 Exactly-Once 세미나 구현

---

11. 성능 튜닝 실험
    - 목표: linger.ms, batch.size, 압축 알고리즘 성능 영향 관찰
    - 실습 포인트: Throughput vs Latency 비교
    - 실행 가이드
    - linger.ms=0/20, compression.type=snappy/lz4/zstd 각각 측정
    - 처리량·지연 모니터링 후 결과 비교

---

12. 모니터링 & 운영 지표
    - 목표: Kafka 주요 메트릭 파악
    - 실습 포인트: kafka-consumer-groups.sh 와 JMX 지표 관찰
    - 실행 가이드
    - Lag, 처리량, 리밸런스 빈도 지표 확인
    - UnderReplicatedPartitions 감시 테스트

기존 컨테이너 정리

```
docker compose down -v
```

재기동

```
docker compose up -d
```

kafka-ui 로그 확인 (연결 실패 메시지 없어야 정상)

```
docker logs -f kafka-ui
```

토픽 생성

```
docker exec -it kafka
kafka-topics --create --topic test-topic
--partitions 1 --replication-factor 1
--bootstrap-server localhost:9092
```

컨슈머 설정

```
docker exec -it understanding-kafka-kafka-1
kafka-console-consumer
--topic test-topic
--bootstrap-server localhost:9092
--group my-group
--from-beginning
```

메시지 전송

```
docker exec -it understanding-kafka-kafka-1
kafka-console-producer
--bootstrap-server localhost:9092
--topic test-topic
--property parse.key=true
--property key.separator=:
```

파티션 1개 - Comsumer 2개

```
curl -X POST --location "http://localhost:8081/api/kafka/send-with-key"
-H "Content-Type: application/json"
-d '{
"key": "1",
```

"message": "foo"
}'

curl -X POST --location "http://localhost:8081/api/kafka/send/loopers"
-H "Content-Type: application/json"
-d '{
"message": "foo"
}'

// TODO
캐시 무효화 : StockAdjusted, LikeChanged 발생 시 상황에 따라 Redis 캐시 삭제