Kafka producer + consumer config 설정 및 실습

▼ producer config

▼ Kafka producer config?

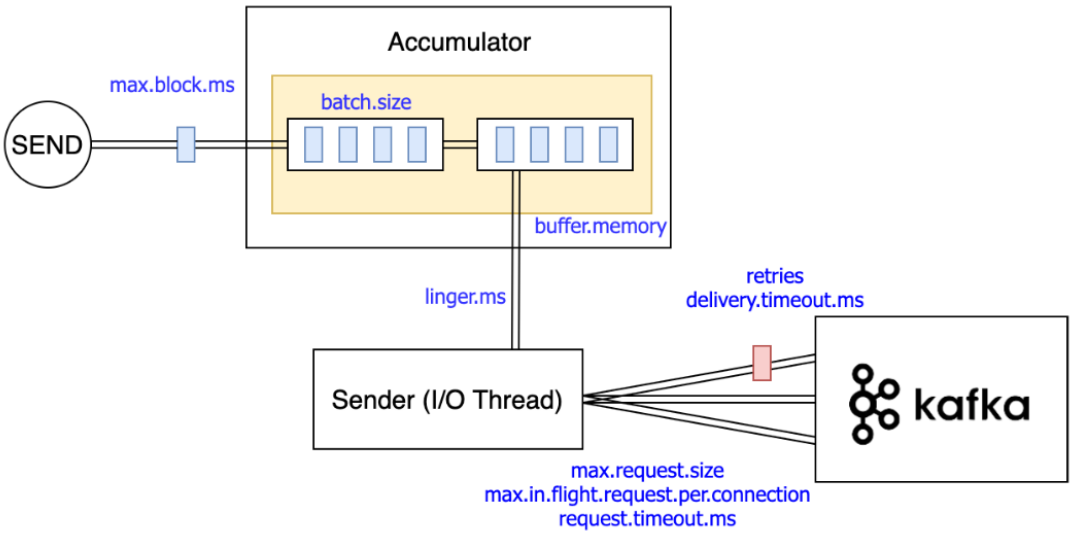
⇒ 데이터를 생산하는 역할

producer의 설정값들은 데이터를 브로커에 발송할 때, 발송하는 데이터의 양/ 주기 및 데이터를 받는 브로커와의 네트워크 연결 등을 조절하는데 사용

producer는 비동기로 브로커에 데이터를 발송하기 때문에 발송하는 과정에서의 설정을 확인

▼ 프로듀서 아키텍처

프로듀서에서 레코드를 발송할 때의 구조를 그린 그림(Accumulator 와 Sender가 포함)



▼ Accumulator

레코드를 축적하는 역할

buffer.memory	프로듀서가 브로커에 보낼 레코드를 모아두는 버퍼의 전체 메모리

	바이트 수 프로듀서가 사용하는 전체 메모리 크기는 buffer.memory와 대략 비슷하다. 브로커로 레코드를 전달하는 것보다 쌓이는 양이 더 많다 → 버퍼 부족 이때, max.block.ms 설정이 필요함
max.block.ms	버퍼가 가득 찼을 때 send(), partitionsFor() 메서드를 블록하는 시간
batch.size	레코드를 묶어서 발송함으로써 성능 향상 레코드를 묶는 byte 단위의 사이즈 배치가 가득찰때까지 프로듀서가 기다리는 것은 X

▼ **Sender**

레코드를 브로커에 전달하는 역할, Accumulator가 쌓아놓은 레코드를 비동기로 브로커에 계속 발송

▼ **linger.ms**

브로커로 발송하기 전에 Accumulator에 축적된 데이터를 지연시키며 가져오는 것
→ 부하 상황에 브로커에 요청수를 줄이기 위해서

한 파티션에 batch.size 만큼 레코드가 있으면 lingers.ms 값을 무시하고 발송
기본값 0 : 대기하지 않는다.

브로커와 통신할 때 사용하는 설정

retries	레코드 발송에 실패할 경우 재시도 하는 횟수 기본값 2147486347 → **횟수가 매우 커 retries를 다 채우지 않고 재시도를 멈추는 설정이 존재 ⇒ delivery.timeout.ms**
delivery.timeout.ms	send() 메서드를 호출하고 성공과 실패를 결정하는 상한시간 브로커로부터 ack를 받기 위해 대기하는 시간 실패시 재전송에 허용된 시간
max.request.size	요청할 수 있는 최대 bytes 사이즈 프로듀서에 의한 한 배치에 보내는 사이즈 제한 서버에서는 별도의 배치 사이즈 설정을 가지고 있음
max.in.flight.request.per.connection	블록되기 전에 클라이언트가 보내고 받지 못한 요청의 최대 개수 1보다 높은 값을 설정하면 재발송 과정에서 순서가 변경될 수 O
request.timeout.ms	요청 응답에 대한 클라이언트의 최대 대기 시간, 타임아웃 시간 동안 응답을 받지 못하

	면 요청을 다시 보낸다 불필요한 재시도를 줄이기 위해 브로커 설정 **replica.lag.time.max.ms** 보다 큰 값이어야 한다
replica.lag.time.max.ms	브로커 팔로워가 복제를 위한 패치 요청을 하지 않을 경우 ISR에서 제외하는 시간

▼ **Importance high config**

▼ **acks**

프로듀서는 전송을 보장하기 위해 acks 라는 설정을 사용합니다.

acks=0 : 프로듀서는 서버로부터 어떤 승인도 기다리지 않는다. 레코드를 소켓 버퍼에 담는 즉시 전송한 것으로 간주 → 서버가 레코드를 받았다는 보장 X & retries 설정 효력 X
레코드를 보내고 받은 오프셋은 항상 **-1**로 세팅

acks=1 : 리더는 로컬 로그에 레코드 기록, 모든 팔로워의 승인을 기다리지 않고 응답
리더가 레코드를 승인한 직후, 팔로워가 복제하기 전에 리더가 실패하면, 레코드는 유실 됨

acks=all : 리더는 in-sync 레플리카 셋 전체가 레코드를 승인할 때까지 기다린다. in-sync 레플리카가 최소 하나라도 살아있으면 레코드를 유실하지 않음을 보장(acks=-1과 동일)

▼ consumer config

Consumer는 Partition에 저장된 데이터를 Polling 해오는 역할

> 💡 **폴링**(polling)이란?
> 하나의 장치(또는 프로그램)가 충돌 회피 또는 동기화 처리 등을 목적으로 다른 장치(또는 프로그램)의 상태를 주기적으로 검사하여 일정한 조건을 만족할 때 송수신 등의 자료처리를 하는 방식

이때 Partition offset 위치를 기록(Commit) 할수도 있음

Consumer가 여러개 일때, Consumer group을 통해 여러개의 분할(Partion)에 대한 병렬처리가 가능

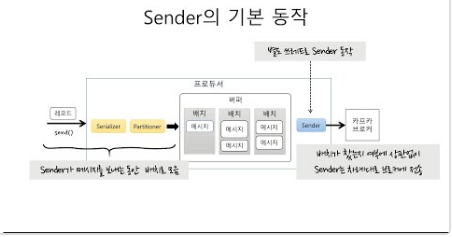
▼ **Importance high**

boostrap.servers	카프카 클러스터에 대한 초기 커넥션을 구축하는 데 사용할 호스트/ 포트 쌍리스트
fetch.min.bytes	패치 요청에서 서버가 반환해야 하는 최소한의 데이터 양 사용 가능한 데이터가 충분하지 않으면, 데이터가 그만큼 쌓이길 기다렸다가 응답함 기본 1바이트 커질수록 서버 처리량을 늘릴 수 있음(대기 시간도 늘어남)
group.id	컨슈머가 속한 컨슈머 그룹을 식별하는 유니크 문자열
heartbeat.interval.ms	컨슈머 코디네이터에 보내는 하트비트의 예상 시간 간격 컨슈머의 세션이 활성 상태인지 확인 session.timeout.ms 보다 낮게 설정해야 함 기본 3초
max.partition.fetch.byte	서버가 파티션 당 반환할 최대 데이터 크기
session.timeout.ms	카프카 그룹 관리 기능에서 클라이언트 실패를 감지할 때 사용할 타임아웃 기본 10초

▼ reference

kafka 조금 아는 척하기 2 (개발자용) - 프로듀서

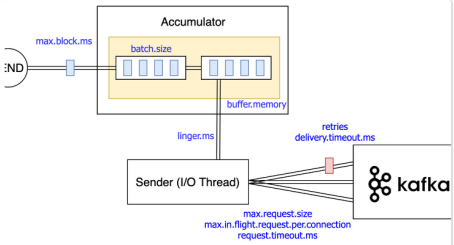
https://www.youtube.com/watch?v=geMtm17ofPY



[Kafka] Producer config 정리

이번 글에서는 카프카 Producer(이하 프로듀서)의 주요 설정 값이 프로듀서의 아키텍처에서 어떤 역할을 하는지 정리한다. 카프카 문서에서는 각 설정값이 설명으로만 나열되어 있어서 이해하기 어

https://devidea.tistory.com/90



Producer Configuration

아파치 카프카 공식 레퍼런스 를 한글로 번역한 문서입니다. 전체 목차는 여기 에 있습니다. 다음은 프로듀서의 설정이다: key.serializer

https://godekdls.github.io/Apache%20Kafka/producer-configuration/

Consumer Configuration

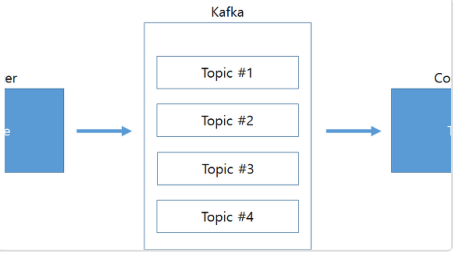
다음은 컨슈머의 설정이다: key.deserializer org.apache.kafka.common.serialization.Deserializer 인터페이스를 구현한 키의 디시리얼라이저 클래스. value.deserializer

https://godekdls.github.io/Apache%20Kafka/consumer-configuration/

Apache Kafka 정리

하이 .. 최근에 프로젝트를 진행하면서 대용량 실시간 데이터를 Apache Kafka를 이용하여 사용해 보았다. 간단하게 Kafka는 링크드인에서 2011년 오픈소스로 공개된 실시간 분산 데이터 처리

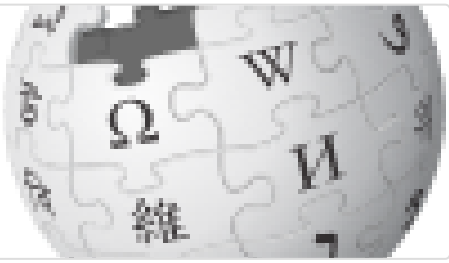
https://devkingdom.tistory.com/273



폴링 (컴퓨터 과학) - 위키백과, 우리 모두의 백과사전

폴링(polling)이란 하나의 장치(또는 프로그램)가 충돌 회피 또는 동기화 처리 등을 목적으로 다른 장치(또는 프로그램)의 상태를 주기적으로 검사하여 일정한 조건을 만족할 때 송수신 등의 자료처

https://ko.wikipedia.org/wiki/%ED%8F%B4%EB%A7%81_(%EC%BB%B4%ED%93%A8%ED%84%B0_%EA%B3%BC%ED%95%99)



🐢 Kafka producer + consumer 실습