* Kafka 2번째 교육

1. 카프카와 메시지큐

래빗MQ – 메세지 큐의 한계점 <- 메모리에 메시지를 적재하고 컨슈머로 푸쉬하는 방식 / 고성능/ 메세지 유실 가능성 존재 / 순서를 보장

카프카 – 메세지 큐의 구조에서 컨슈머 단이 더 추가됐고 디스크에 메시지를 적재하는 분산형 구조에 초점이 맞춰진 방식 / 메세지 큐에 비해 성능이 떨어짐 / 안정적 / 순서를 보장하지 않음

카프카는 MQ에 비해 로그 처리에 특화되어 TPS가 압도적으로 매우 우수하고 컨슈머가 메세지에 순차적으로 접근하기 때문에 디스크 I/O의 한계를 어느 정도 극복

1. 카프카 구성 요소

프로듀서 – 브로커(카프카) – 컨슈머(ELK,하둡)

주키퍼 – 클러스터 관리용 애플리케이션

토픽 안에서 파티션의 offset으로 순서를 가리키고 있음

카프카에서 offset은 컨슈머가 가져간 순서를 나타냄

컨슈머는 offset을 설정해서 가져갈 수 있음

컨슈머 스레드 하나는 하나의 파티션만 바라볼 수 있음(파티션의 개수 : 컨슈머의 개수 = 1:1인 경우가 베스트)

들어오는 메시지와 나가는 메시지의 차이를 봤을 때 컨슈머로 가져가는 곳에서 병목이 생긴 경우 카프카에 렉이 쌓인다고 함(렉이 항상 0인 경우가 좋음)

1. 카프카의 데이터 안전성

토픽은 파티션 수 만큼 리플리케이션을 만들 수 있음(보통 replication 옵션 1로하여 백업함)

원본 토픽은 leader라고 하고 리플리케이션은 follower(백업)라고 함

리더와 팔로워가 같은 노드에 들어가지 않도록 하여 리더가 죽은 경우를 방지하는 것이 목적

리더가 죽어 팔로워로 복구하는 경우 완전히 복사될 때까지는 팔로워가 리더 역할을 하고 완전히 복사가 되면 역할을 변경

메시지가 오면 리더한테 쓰고 리더가 팔로워한테 전달하여 복원 구조를 가짐

ACK 옵션 0 : 메시지가 오면 카프카가 정상적으로 메시지를 받았다고 프로듀서에게 바로 ok를 전달하는 것

ACK 옵션 1 : 카프카에서 리더가 쌓았을 때 ok를 프로듀서에게 전달

ACK 옵션 -1 or All : 리플리케이션까지 받았을 때 프로듀서에게 ok 전달

1. 카프카와 주키퍼

주키퍼는 클러스터의 메타데이터을 저장하여 클러스터를 관리하는 애플리케이션

주키퍼로 카프카를 관리하고 사용자는 주키퍼를 보고 토픽과 메시지와 노드를 결정하고 함

클러스터 – 보통 노드 30개 정도로 셋팅하고 주키퍼로 이를 관리

주키퍼 클러스터의 노드도 홀수개로 책정하여 짝수개가 되면 에러 상황으로 인지

주키퍼 3대-medium / 카프카 3대-Large 설치 화요일까지