Apache Kakfa

18기 엔지니어링 조은학
02069@NAVER.COM
2022.08.10

CONTENTS

01 카프카란?

02 카프카 구조 및 원리

03 카프카 특징

04 카프카 성능

05 카프카 실습

06 Appendix

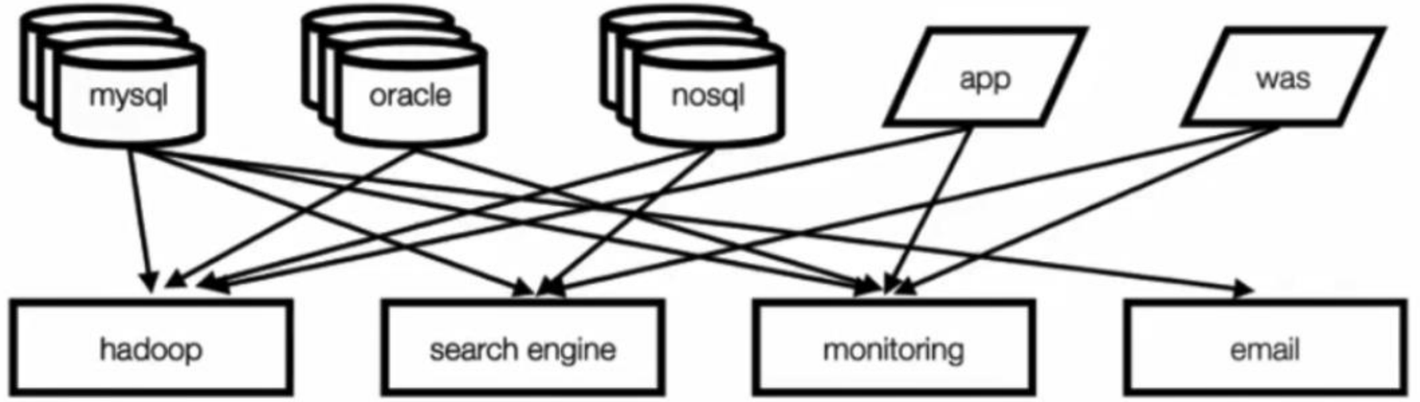
07 과제

01 카프카란?

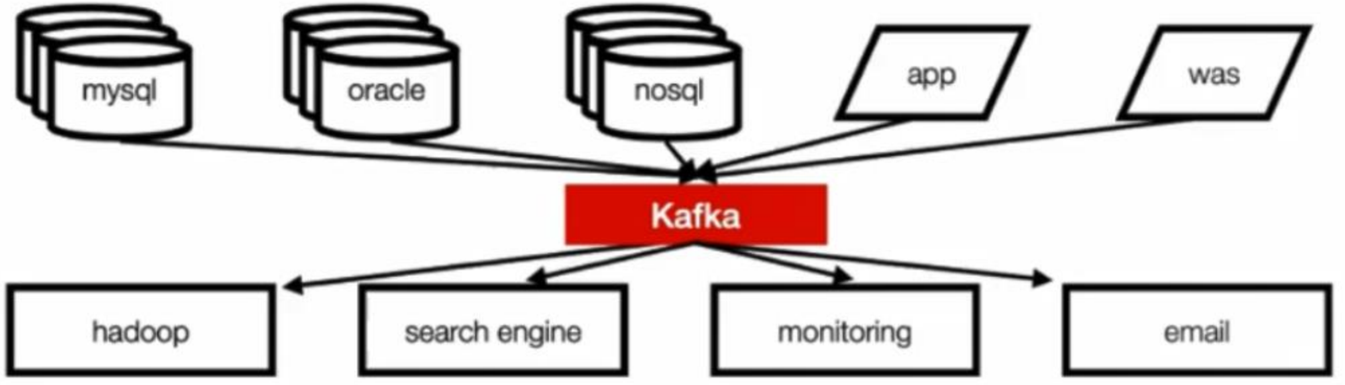
- 링크드인에서 만든 고성능 분산메세징용 큐
- 안정적인 버퍼링(큐잉) / 스트리밍 처리 / 대용량 로그 수집

01 카프카란?

카프카 이전

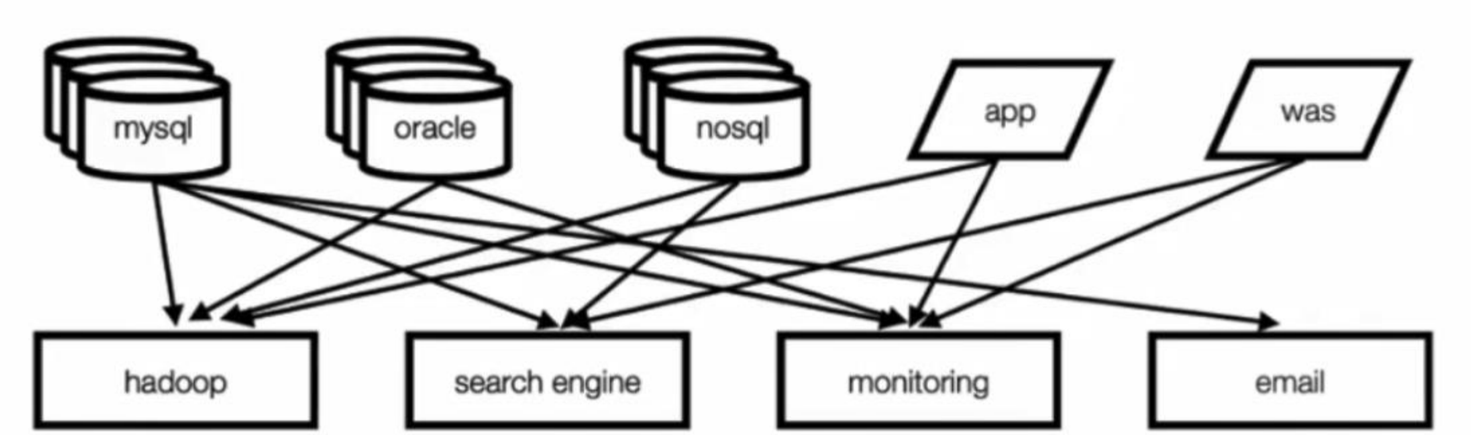


카프카 이후



01 카프카란?

카프카 이전

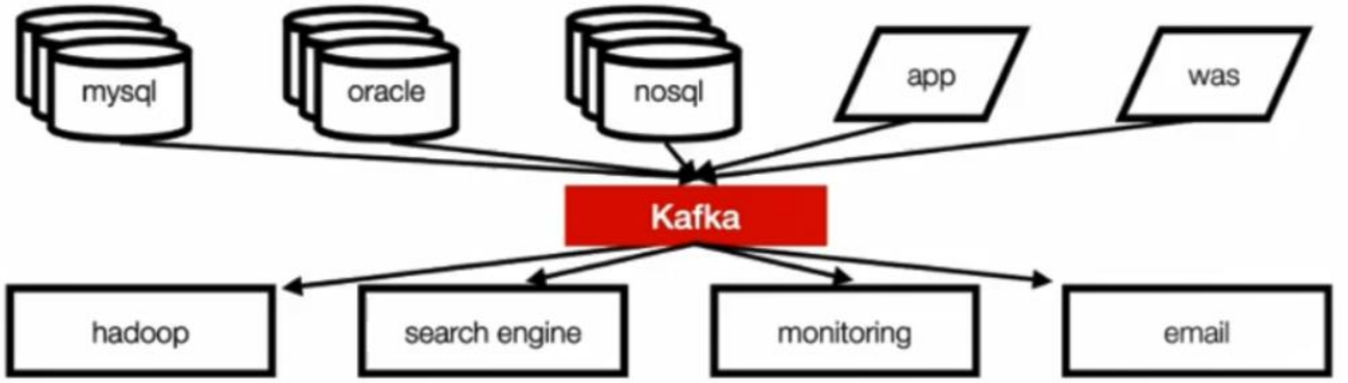


문제점

1. 데이터 연동의 복잡성
2. 데이터의 양이 증가할 경우 확장이 용이하지 않음

01 카프카란?

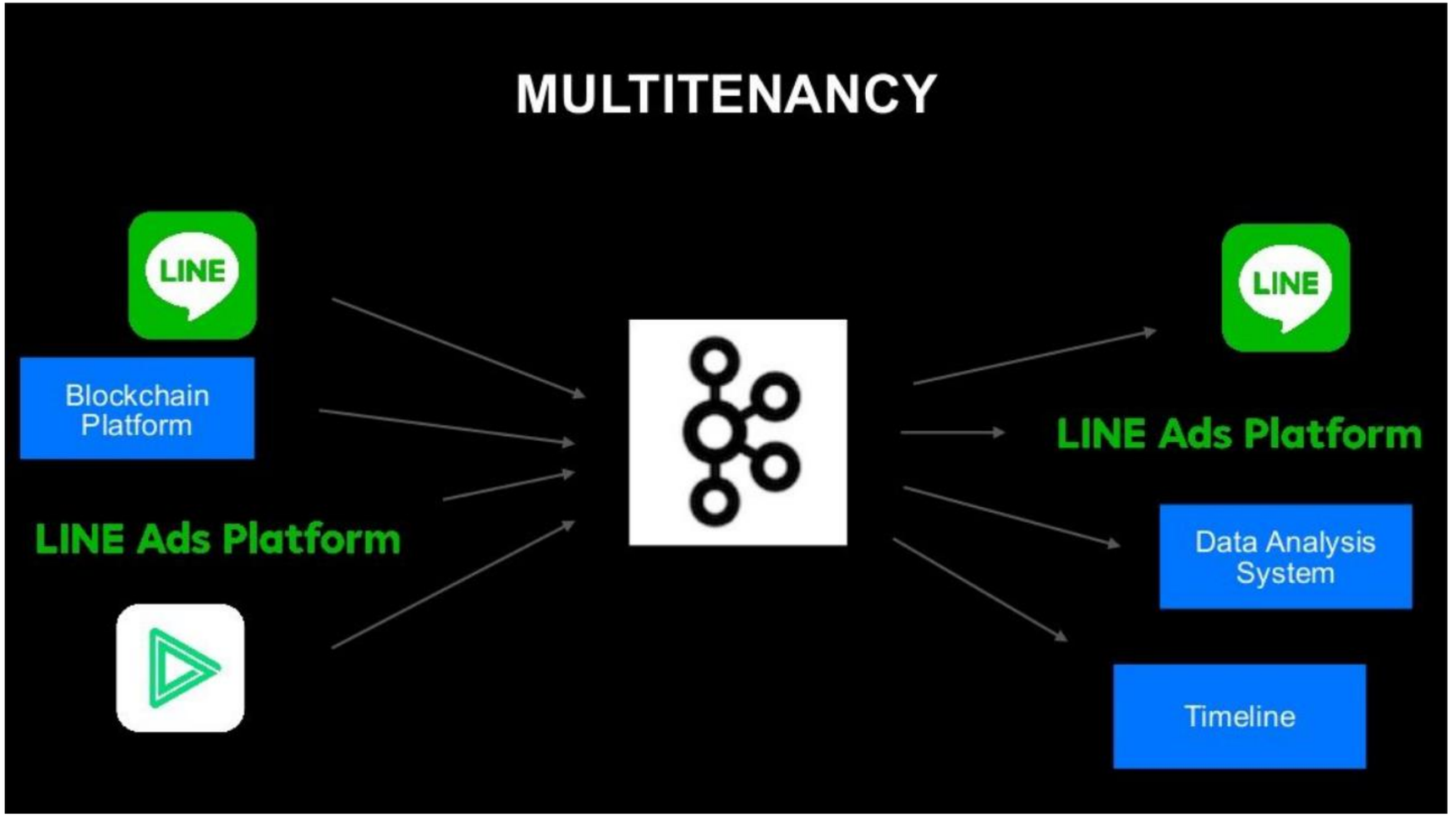
카프카 이후



해결 방법

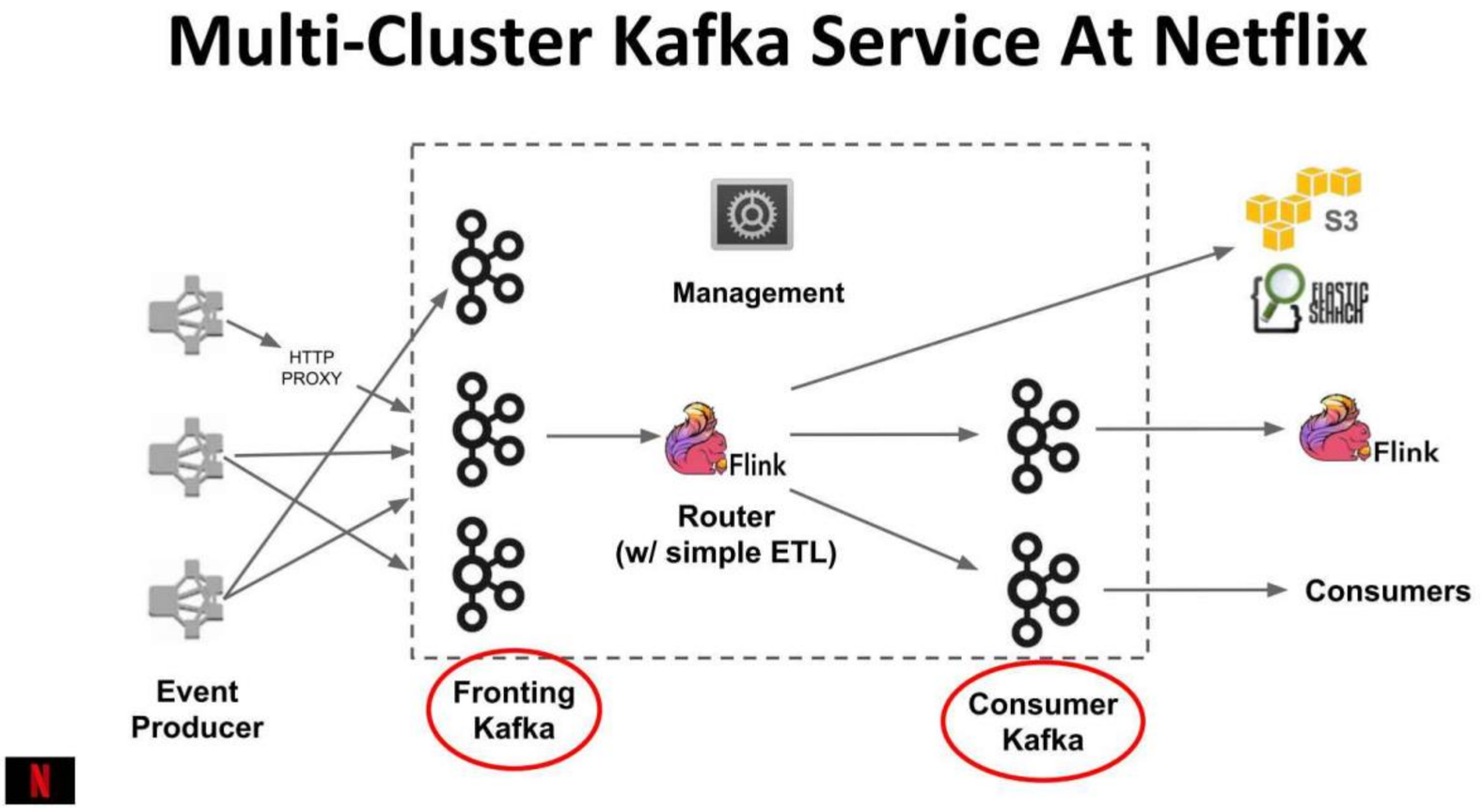
1. 데이터를 보내는 쪽과 받는 쪽으로 구분하여 단순화
2. 데이터의 양이 증가하면 카프카만 고려하면 됨

01 카프카란? - 실제 예시(라인)



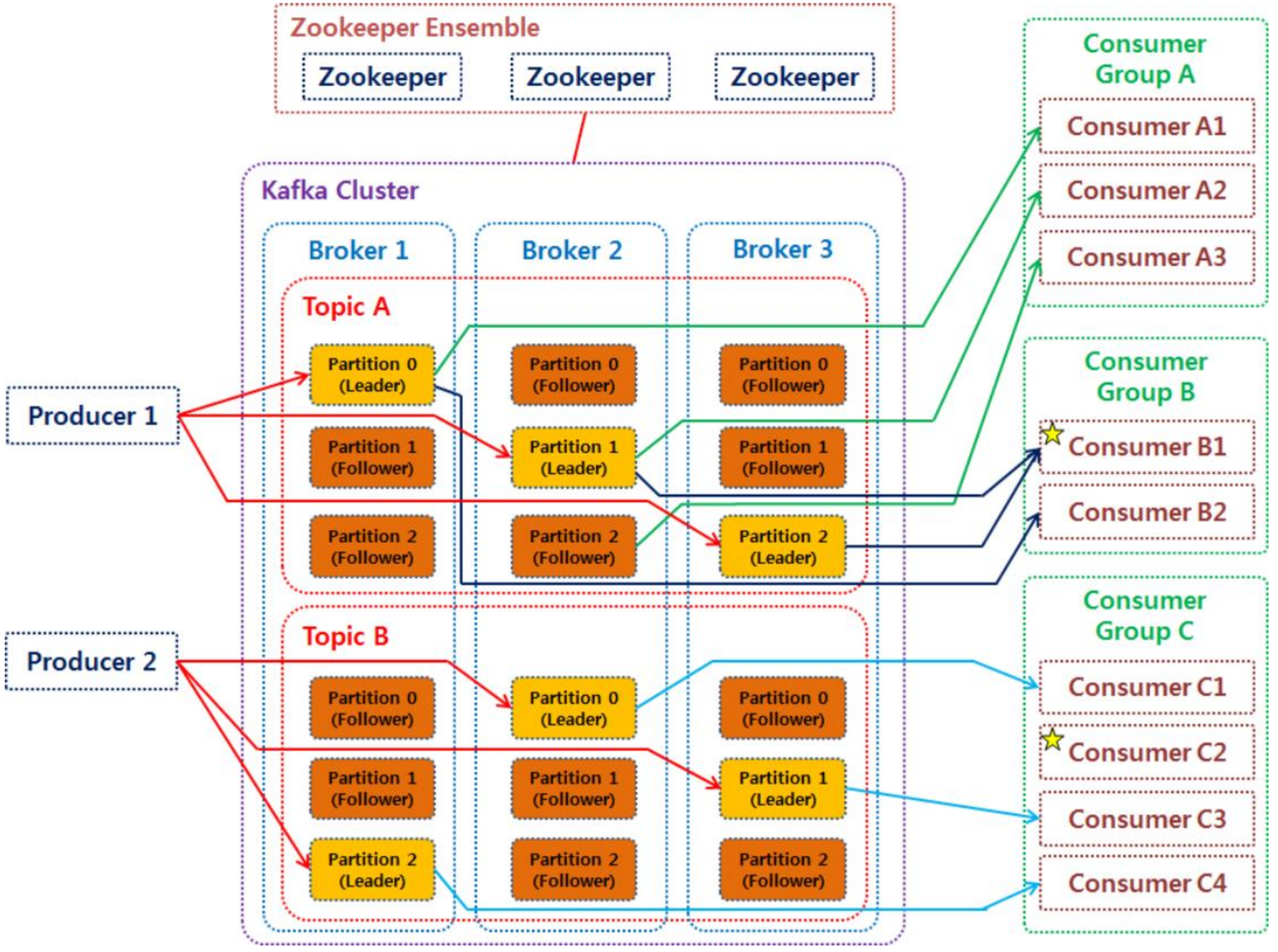
https://engineering.linecorp.com/ko/blog/how-to-use-kafka-in-line-1/

01 카프카란? - 실제 예시(넷플릭스)



https://www.infoq.com/presentations/cloud-native-kafka-netflix/

01 카프카 구조 및 원리 - 카프카 용어



Broker - 카프카의 서버

Topic - 데이터의 구분 단위, 한 개 또는 여러 개의 파티션으로 이루어짐

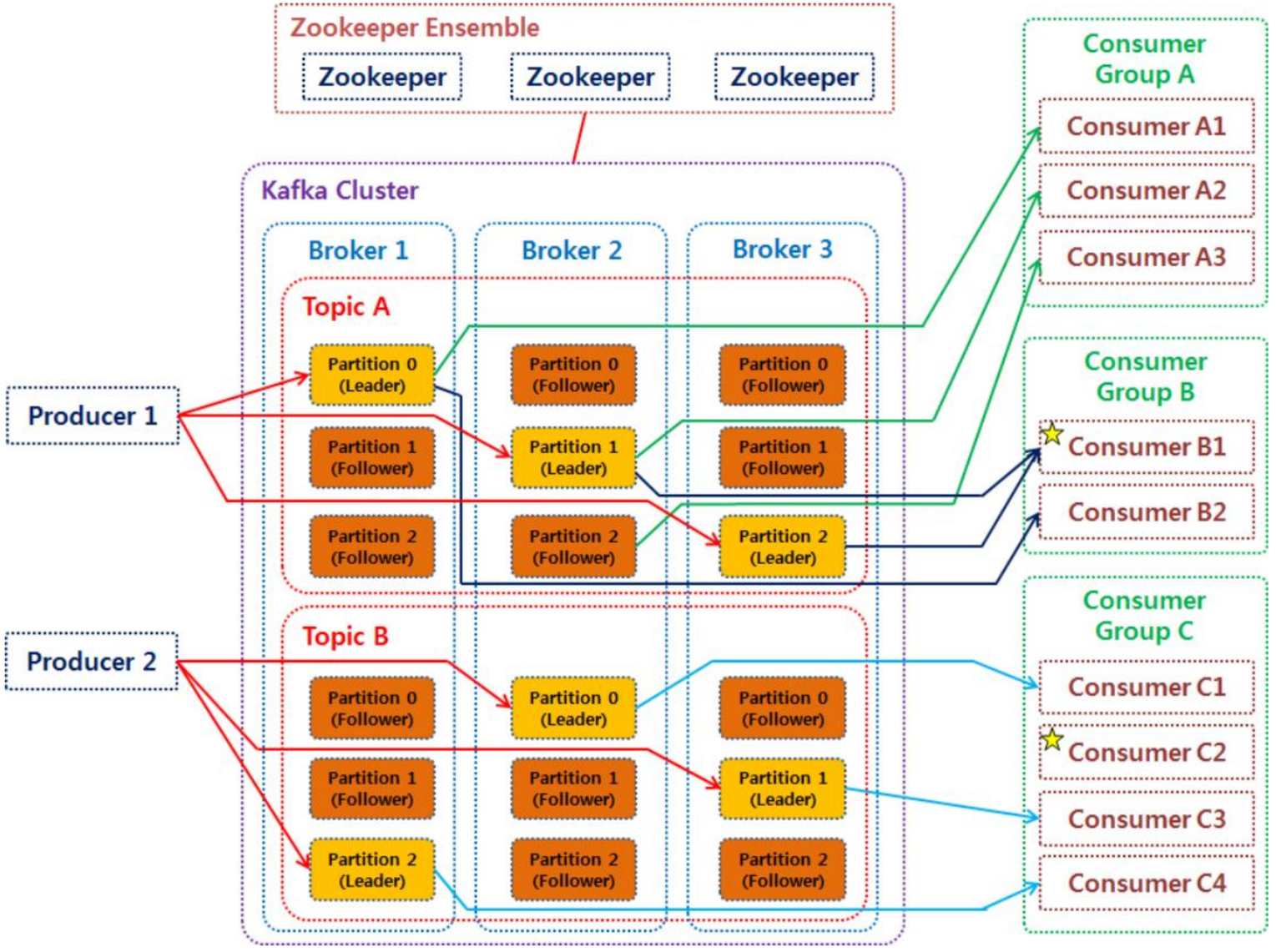
Partition - 토픽 안에서의 데이터 구분 단위

Producer - 이름 그대로 생산자, 데이터를 생산해서 브로커로 데이터를 보냄

Consumer - 이름 그대로 소비자, 브로커에 적재된 데이터를 가져옴

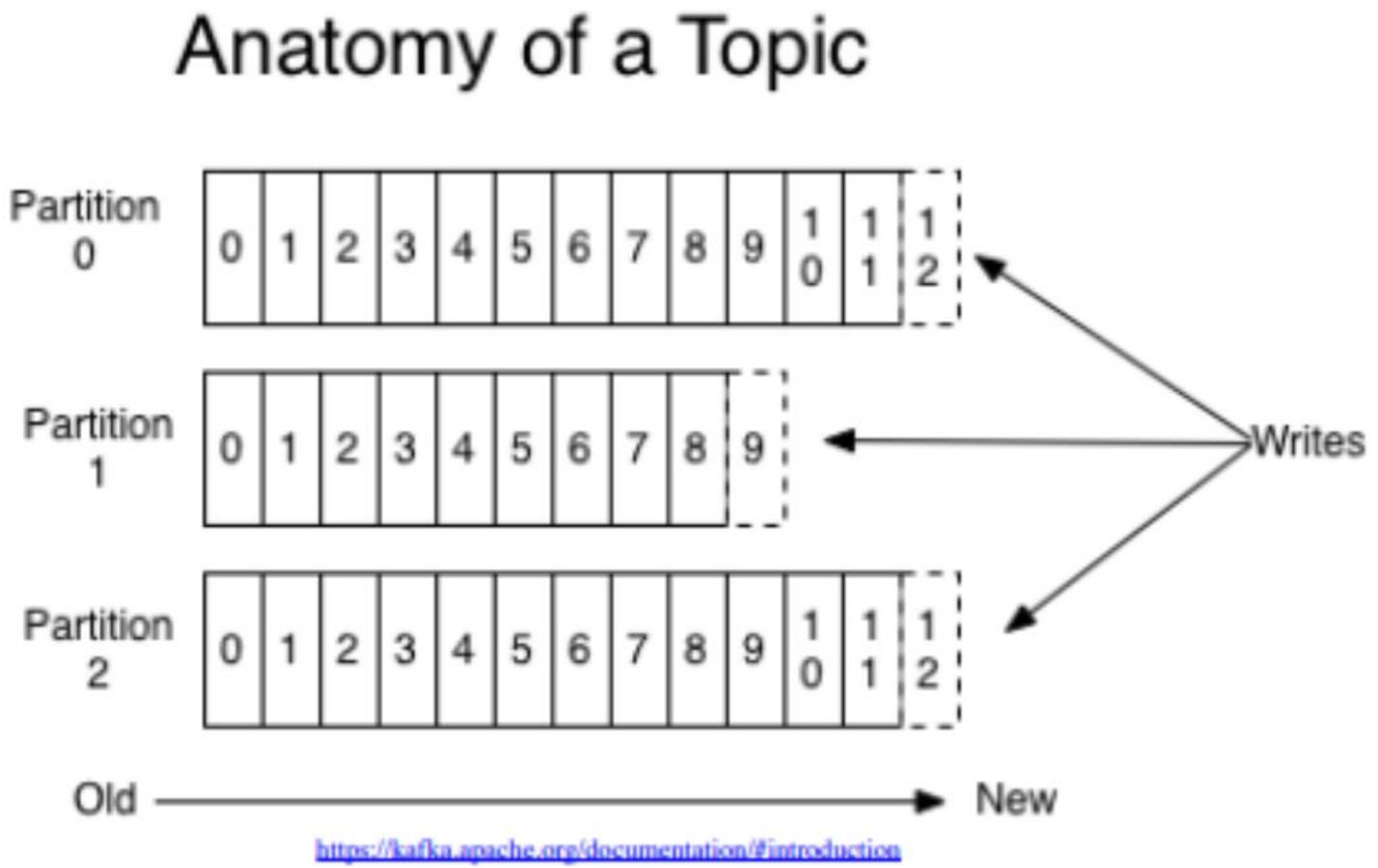
Zookeeper - 카프카의 메타데이터 관리 도구, 원래 카프카에 필수였는데 점차 카프카에서 제거중(22.05)

02 카프카 구조 및 원리-브로커



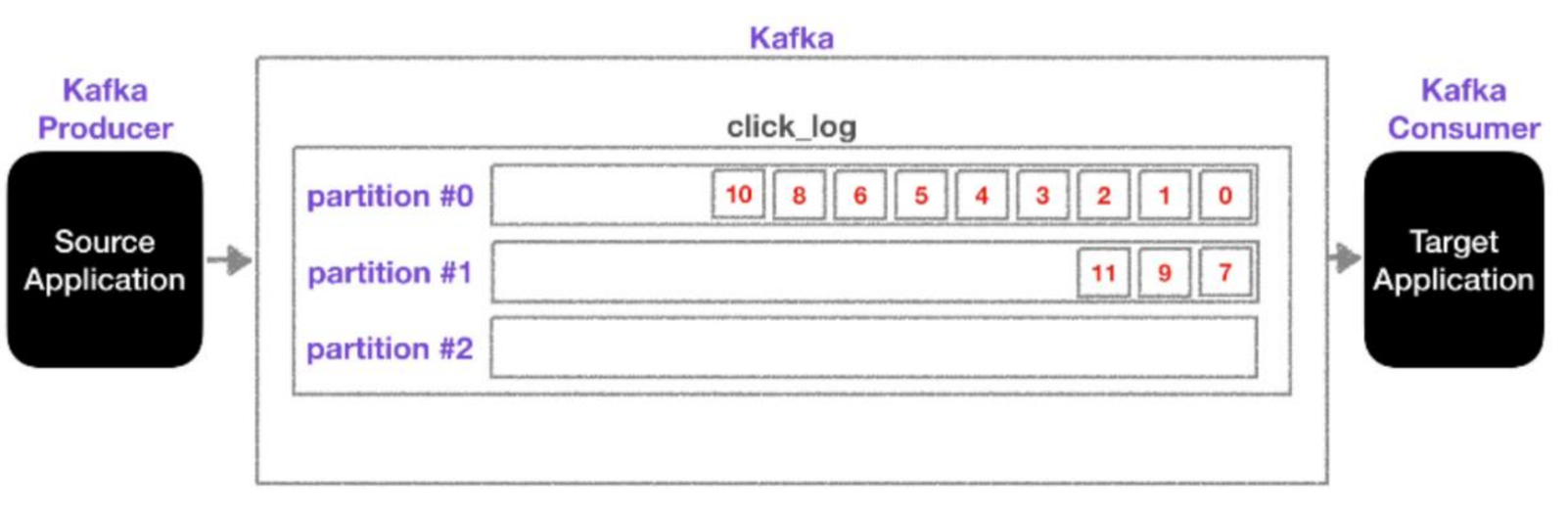
broker
- 카프카의 서버
- 일반적으로 3대 이상의 브로커로 구성
- 브로커 중 한 대는 controller
(controller는 브로커가 장애 발생으로 사용할 수 없는 경우, 브로커에게 담당 파티션을 할당해줌)

02 카프카 구조 및 원리-토픽



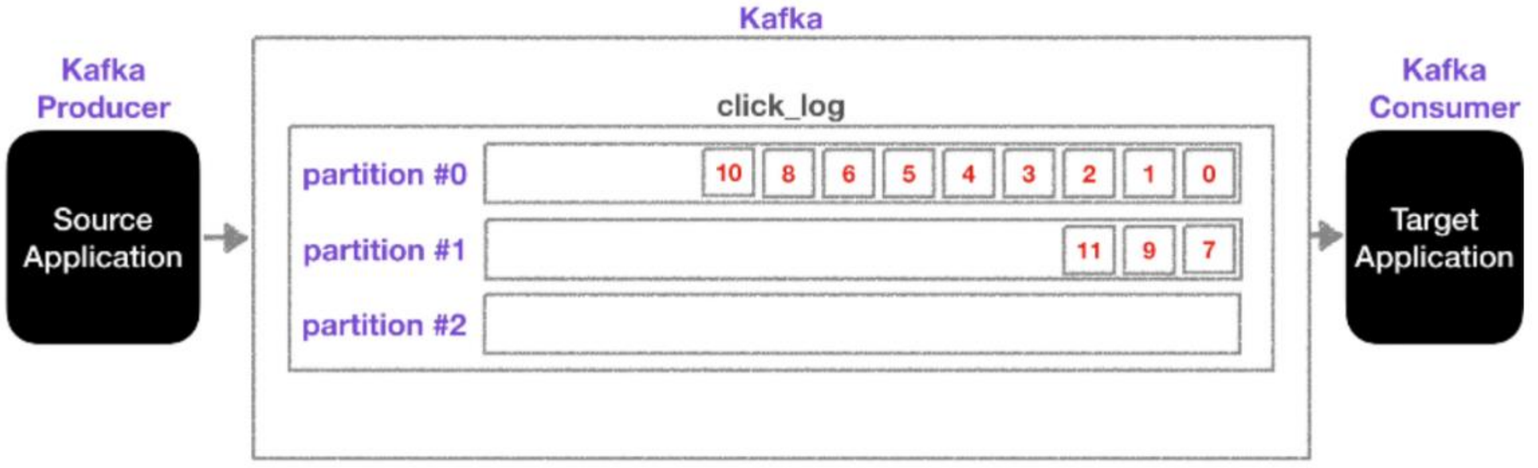
topic
- 메세지 구분 단위
- 1~n개의 파티션으로 구분됨
- 각 파티션마다 고유한 offset을 가짐
(offset은 파티션 내에서 데이터의 위치를 표시하는 숫자)

02 카프카 구조 및 원리-파티션



partition
- 토픽을 나누는 단위
- 레코드가 모여 파티션이 됨
(레코드는 메시지 하나를 의미)
- 파티션은 한 번 늘리면 줄일 수 없기 때문에 주의해서 늘려야 함

02 카프카 구조 및 원리-파티션 구분방법



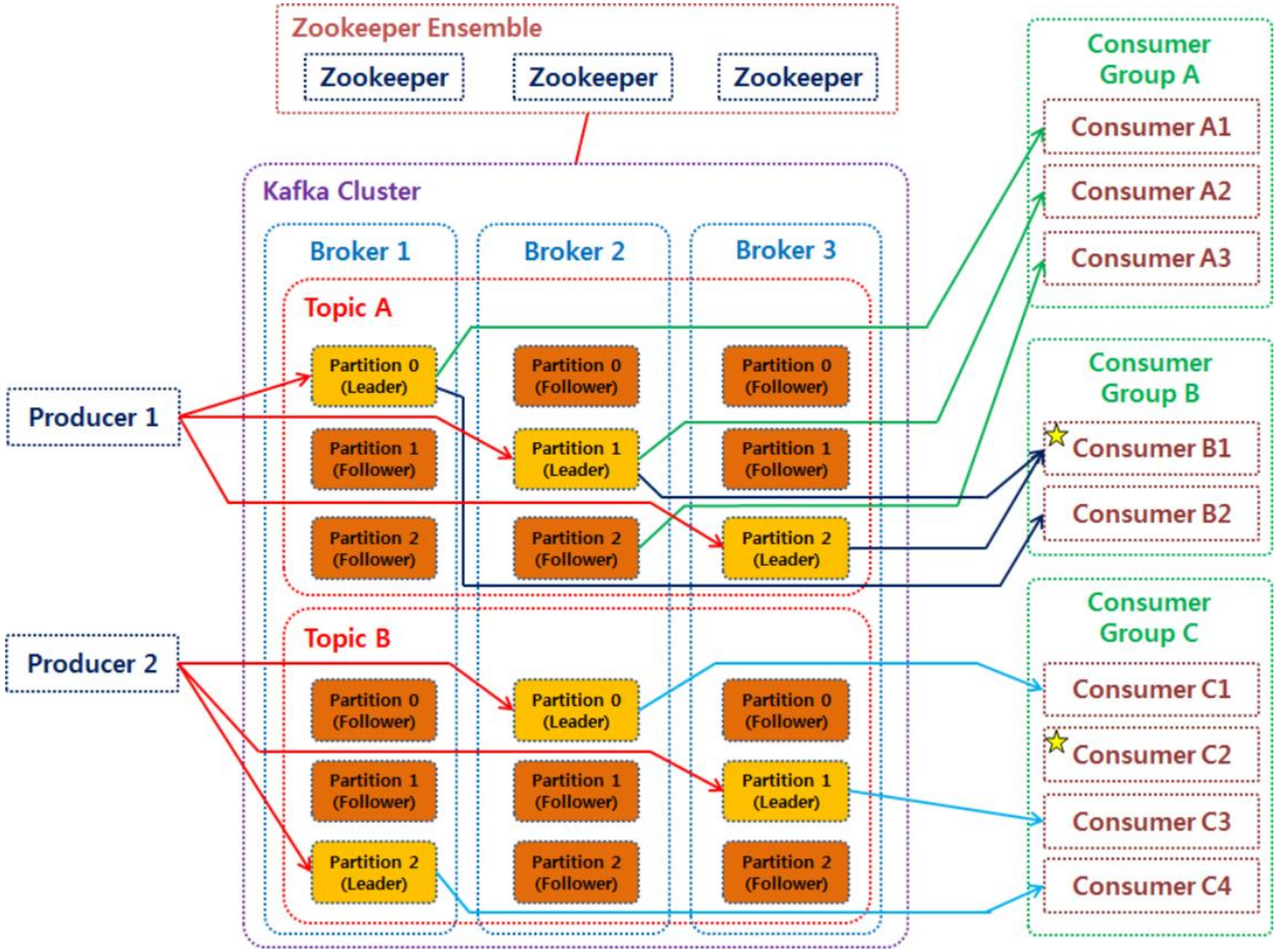
partitioner

partitioner에 의해 레코드는 어느 파티션으로 갈지 정해짐
지정하지 않으면 defaultpartitioner로 동작

defaultpartitioner 동작 방식

Key 값 존재 -> Key 값의 Hash 값을 이용해서 할당
Key 값 존재X -> Round-Robin 방식으로 할당

02 카프카 구조 및 원리-리더 파티션과 팔로워 파티션



leader partition

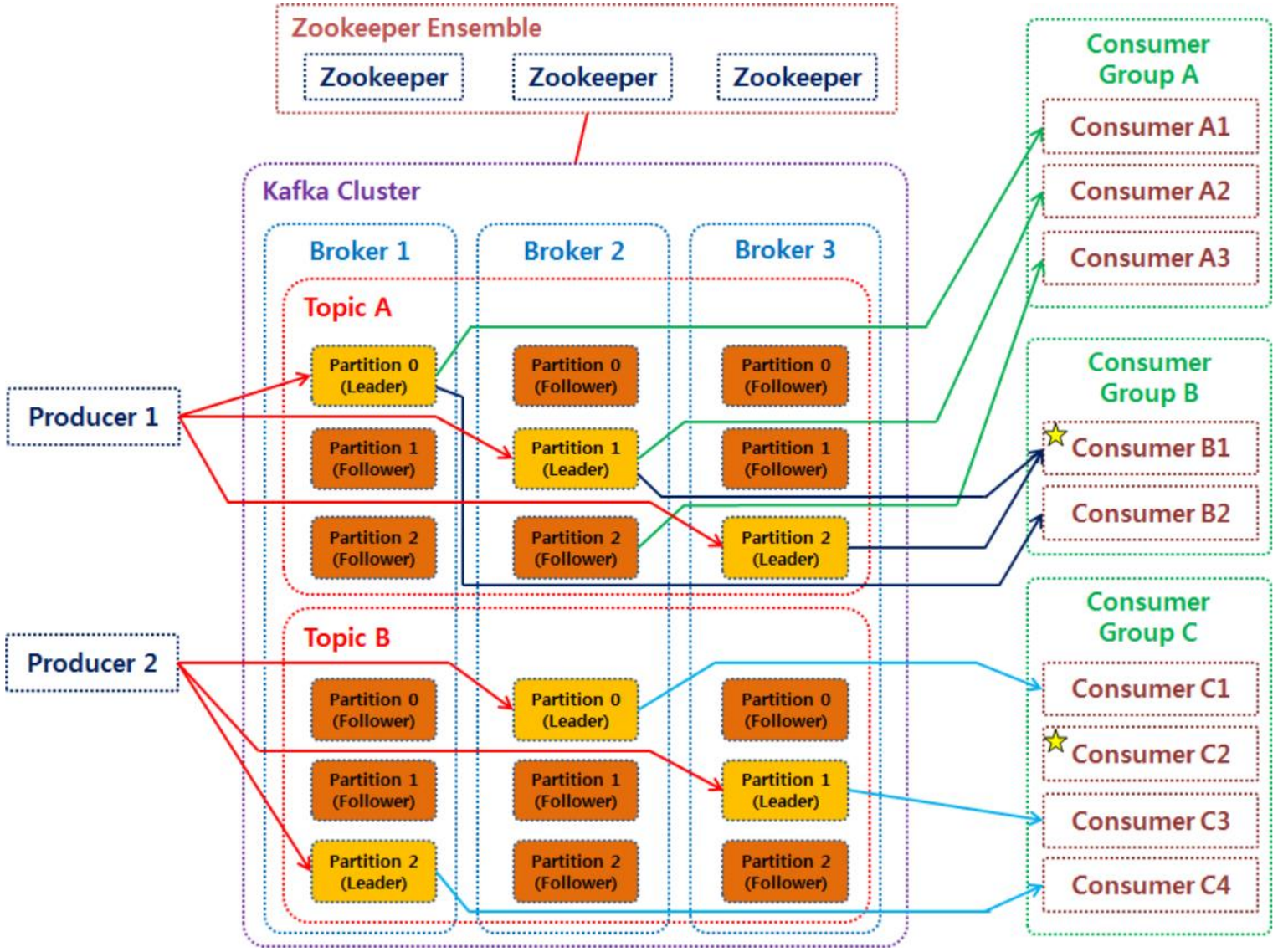
- 프로듀서 또는 컨슈머와 직접 통신하는 파티션
- (원본 데이터라고 생각하면 됨)

follower partition

- leader 파티션의 데이터를 복제하여 저장
- leader partition이 장애가 날 시에 follower partition이 leader partition이 될 수 있음

복제 수는 replication factor로 지정할 수 있음

02 카프카 구조 및 원리-프로듀서와 컨슈머

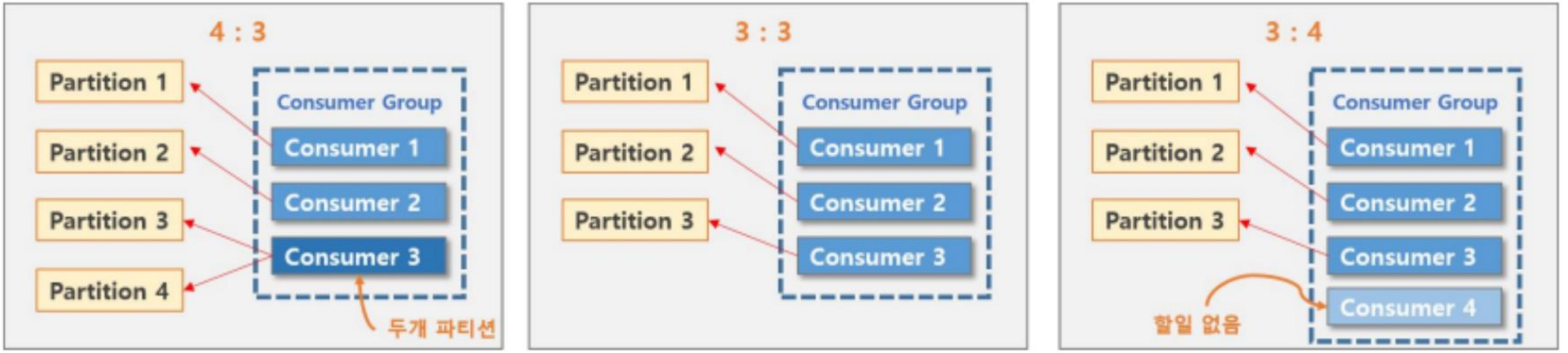


Consumer group

consumer들을 묶어 놓은 그룹

파티션이 여러 개이기 때문에 consumer가 여러 개 있어야 성능에 효율적

02 카프카 구조 및 원리-프로듀서와 컨슈머



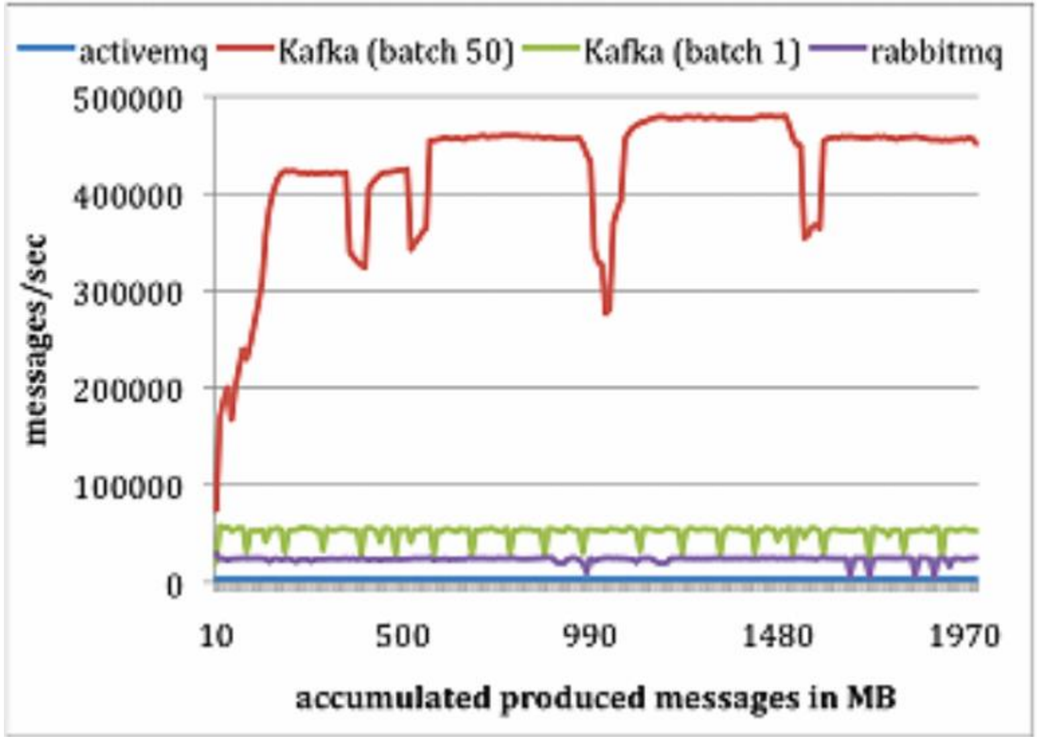
partition >= consumer 로 설정해야함
partition < consumer 면 consumer 하나는 놀게 됨

partition : consumer의 개수를 같게 맞춰주는 것이 보통 제일 이상적
partition >> consumer 면 지연이 생길 수 있음

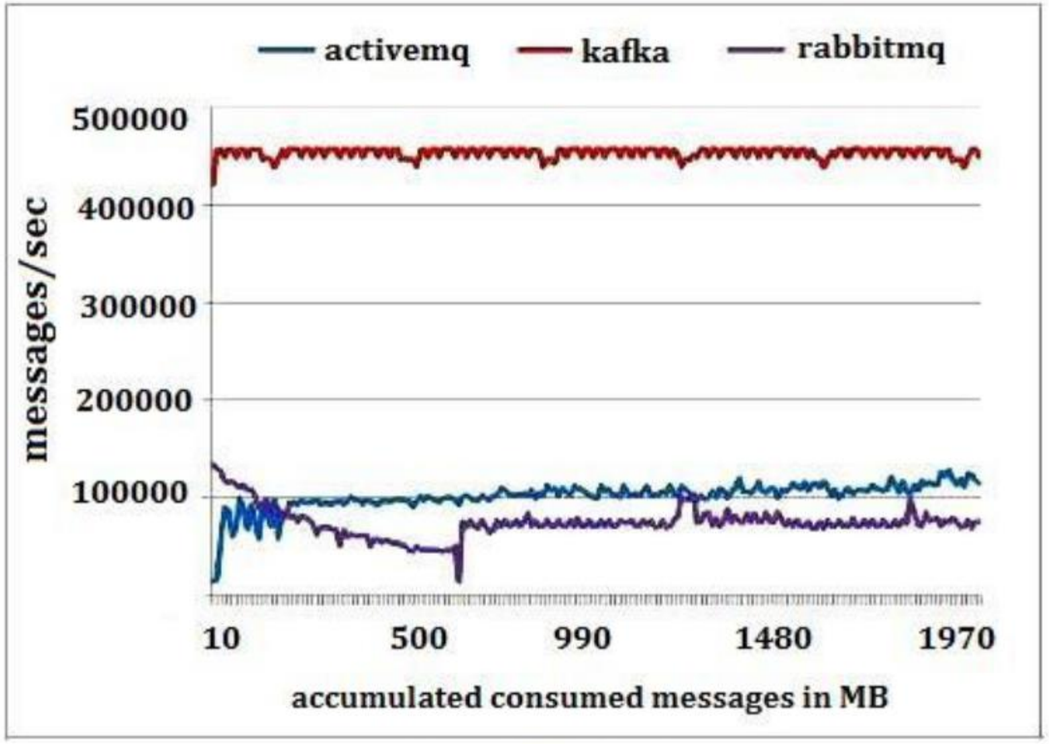
03 카프카 특징

- publish/subcribe 방식 (producer, consumer)
  카프카에 데이터에 전달하면 필요한 곳에서 각자 가져갈 수 있음
- 확장 용이
  카프카 클러스터는 3대의 브로커로 시작하여 수십대의 브로커로 무중단 확장이 가능함
- 고가용성(High Availability)
  토픽은 파티션으로 쪼개져서 클러스터의 각 서버들에 분산 되어 저장되는데 replication 설정으로 데이터 유실을 방지할 수 있음(fault tolerant)

04 카프카 성능



producer 성능



consumer 성능

04 카프카 성능

page cache

페이지 캐시란?

메모리 영역에 어플리케이션이 사용하는 부분을 할당하고 남은 잔여 메모리를 캐시로 전환하여 디스크 접근을 최소화해 I/O 성능을 향상시키는 방법

04 카프카 성능

page cache

Another unconventional choice that we made is to avoid explicitly caching messages in memory at the Kafka layer. Instead, we rely on the underlying file system page cache. This has the main benefit of avoiding double buffering---messages are only cached in the page cache.

Kafka: a Distributed Messaging System for Log Processing 논문 발췌

04 카프카 성능

page cache

Using the filesystem and relying on pagecache is superior to maintaining an in-memory cache or other structure

This suggests a design which is very simple: rather than maintain as much as possible in-memory and flush it all out to the filesystem in a panic when we run out of space, we invert that. All data is immediately written to a persistent log on the filesystem without necessarily flushing to disk. In effect this just means that it is transferred into the kernel's pagecache.

https://kafka.apache.org/documentation/#persistence
카프카 공식문서 발췌

04 카프카 성능

zero copy

일반적인 방식

로컬 파일에서 원격 소켓으로 바이트 단위로 전송하는 일반적인 방식

(1) read data from the storage media to the page cache in an OS
(2) copy data in the page cache to an application buffer
(3) copy application buffer to another kernel buffer
(4) send the kernel buffer to the socket.

4 data copying and 2 system calls

카프카 방식 (zero-copy)

Linux나 Unix의 sendfile API

directly transfer bytes from a file channel to a socket channel

(2)와 (3) 스텝을 없애
2 data copying and 1 system calls

05 카프카 실습

https://capable-fortnight-eae.notion.site/Kafka-c0fba46b8def42c8bdc3e57050ec4797

06 Appendix - 카프카 설정

- broker config (메세지의 최대 크기, 로그 보관 주기, 로그 저장 위치 등)
- producer config (브로커가 잘 받았는지 (acks), 레코드 묶는 배치 사이즈 등)
- topic config, consumer config 등

https://kafka.apache.org/documentation/
#configuration

06 Appendix

- kafka connect
- kafka streams
- ksql
- Burrow (모니터링)
- redpanda (카프카보다 10배 빠르다고 주장)

07 과제

- **broker, topic, producer, consumer 중에 하나를 골라 관련해서 2~3줄 정도로 찾아보고 정리해서 카페에 올리기**

  ex) broker는 카프카에서 producer로부터 전달받은 메시지를 consumer로 전달해주는 중간 역할이다. broker는 한 대 이상의 노드로 이루어져 클러스터로 구성될 수 있는데 내부에는 여러 토픽들이 생성될 수 있다. 이러한 토픽들 안에 있는 파티션은 분산 저장되어 있어 장애 발생 시 안전하게 데이터를 저장할 수 있게 한다.

참고 자료

Apache kafka 기본개념 및 생태계-데브원영
https://www.youtube.com/watch?v=catN_YhV6To&t=1034s

카프카 공식문서
https://kafka.apache.org/documentation/

카프카 논문(2011)
Kafka: a Distributed Messaging System for Log Processing

kafka 조금 아는 척하기 1 (개발자용)-최범균
https://www.youtube.com/watch?v=0Ssx7jJJADI

감사합니다