Kafka에 대한 이해

기술 세미나

2025.02.17

Team 2 (weAreFoodie)

김대연, 민정인, 어태규, 최다영

목차 a table of contents

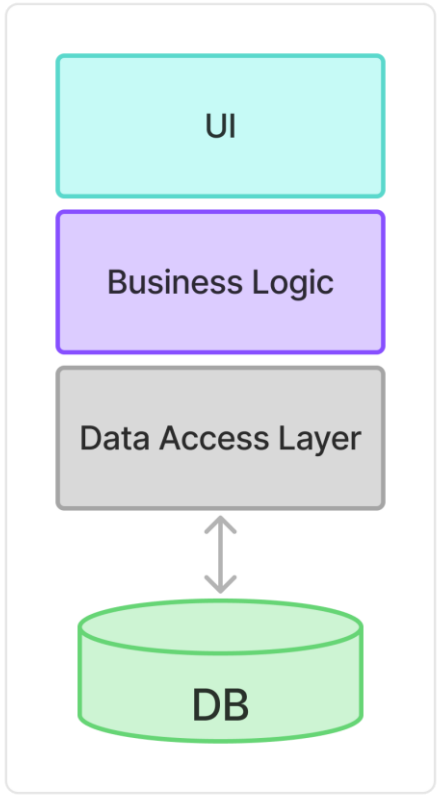
Part 1

도입 배경 및 필요성

Part 1

도입 배경 및 필요성 | 기존 시스템의 문제점

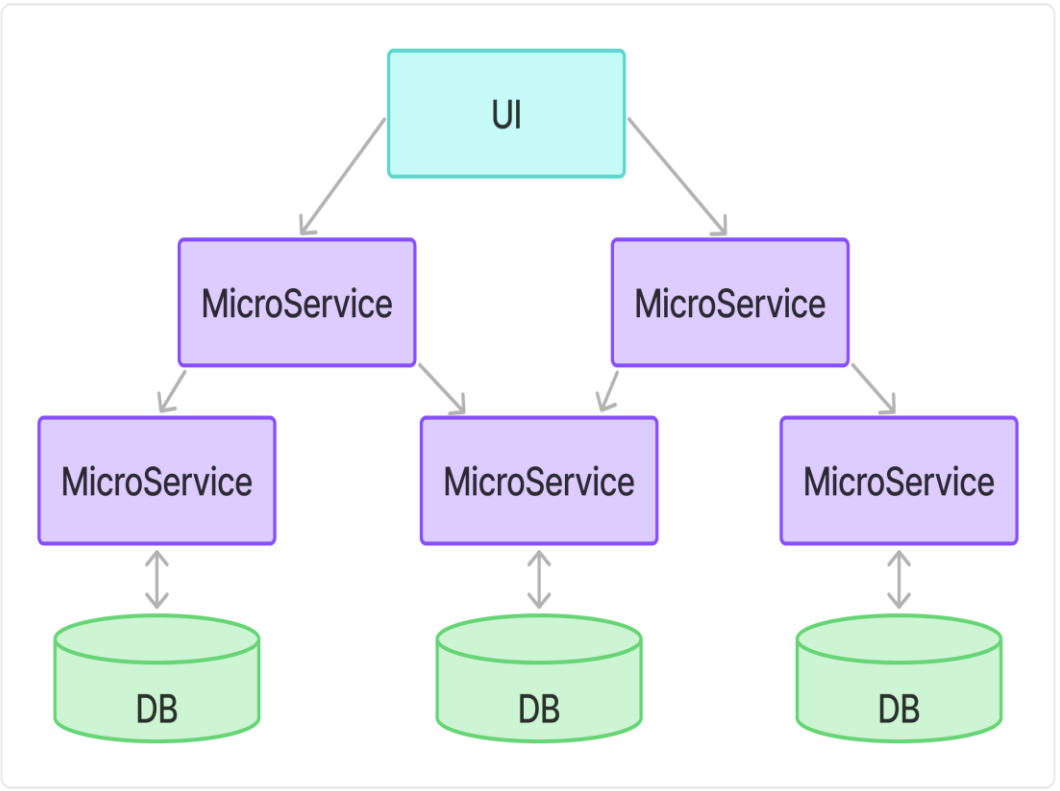
모놀리식 아키텍처



유지 보수와 변경 및 확장의 어려움

MSA 의 확산

마이크로서비스 아키텍처



도입 배경 및 필요성 | 기존 시스템의 문제점

사용자 경험 개선

운영 최적화

비즈니스 인사이트 도출

+ α

사용자 행동

센서 데이터

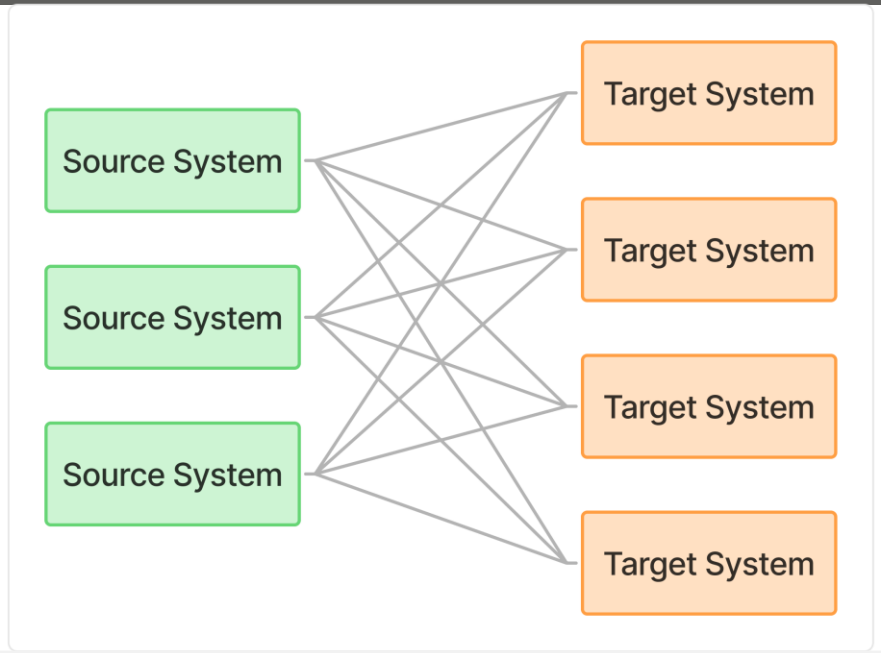
각종 거래 데이터 . . .

실시간 데이터 처리의 필요성 증가

Part 1

도입 배경 및 필요성 | 기존 시스템의 문제점

마이크로서비스 아키텍처(MSA)의 확산



- NxM 개별 통합(통신)
- 프로토콜 문제
- 데이터 포맷 문제
- 스키마 변경 문제
- 성능 저하
- 장애 전파

실시간 데이터 처리의 중요성 증가

데이터베이스 중심 설계

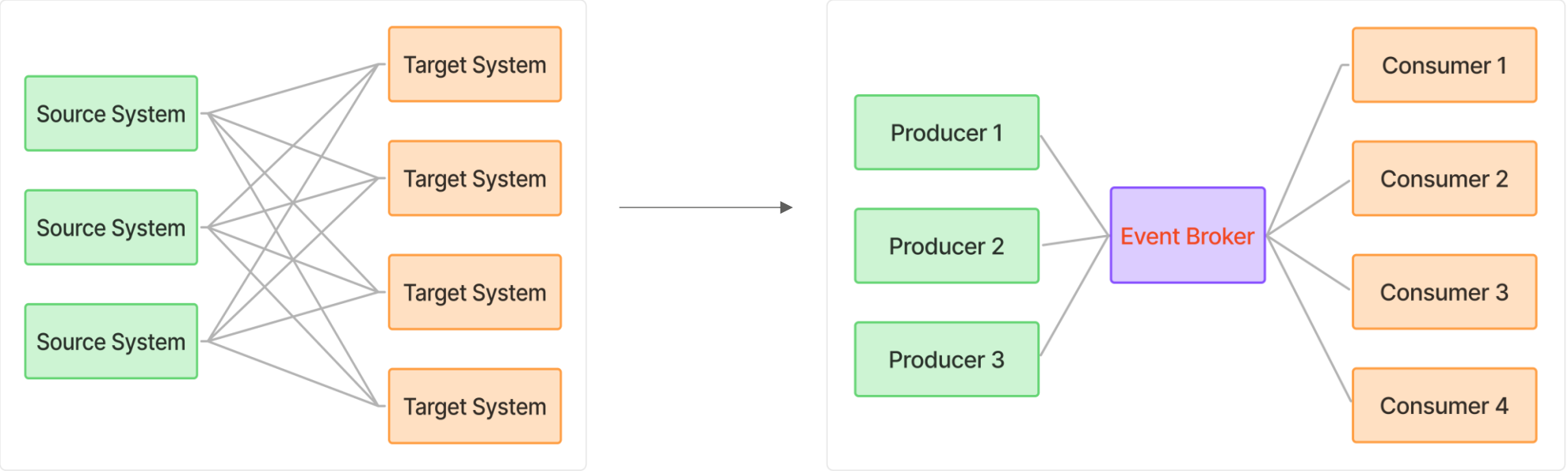
↓

실시간 데이터 처리의 어려움

- 서비스 증가 시 DB 부하 증가 문제
- 트래픽 급증 시 DB 병목 발생 문제
- 데이터가 저장된 후 처리 가능

이벤트 중심(Event-Driven) 아키텍처

비동기적 통신과 **이벤트 기반 데이터 처리**를 중심으로 작동함
서비스 간의 결합도를 낮추고 확장성을 높임



도입 배경 및 필요성 | Kafka

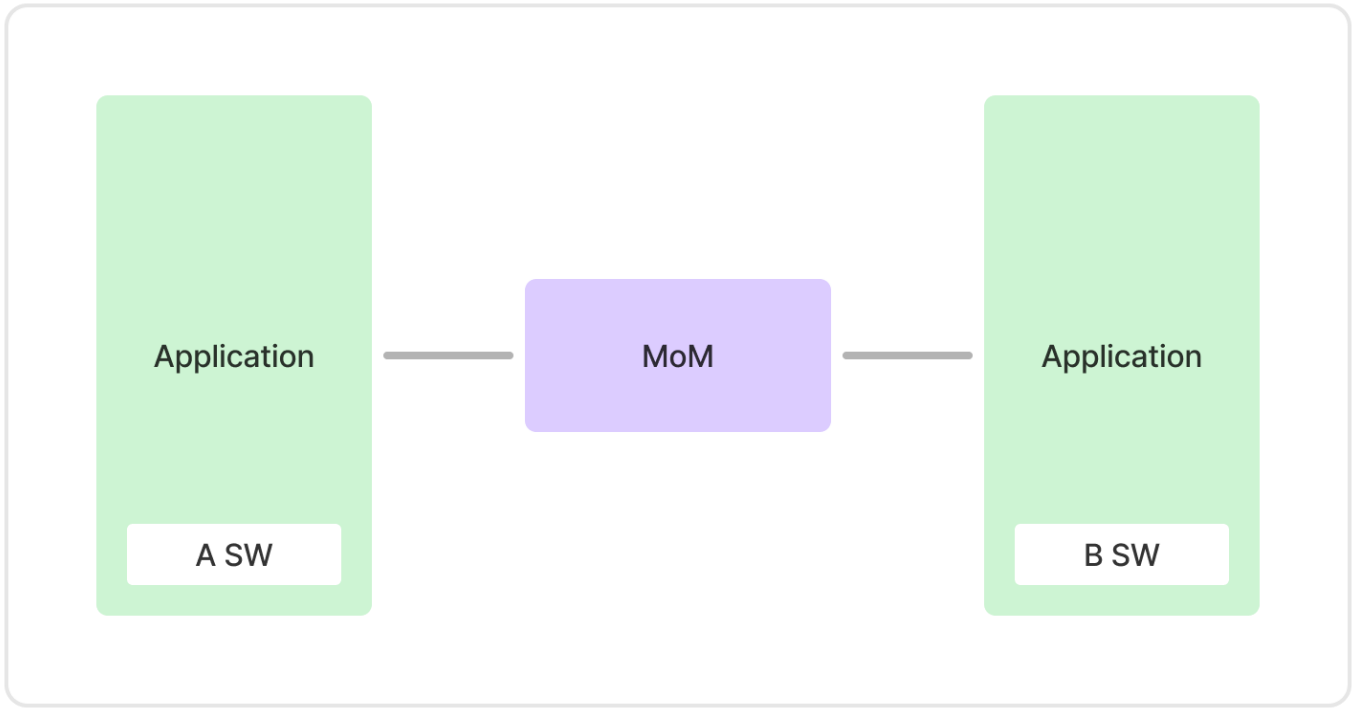
Part 2

카프카의 핵심 요소

Part 2

카프카의 핵심 요소 | MoM

MOM(Message-Oriented Middleware)

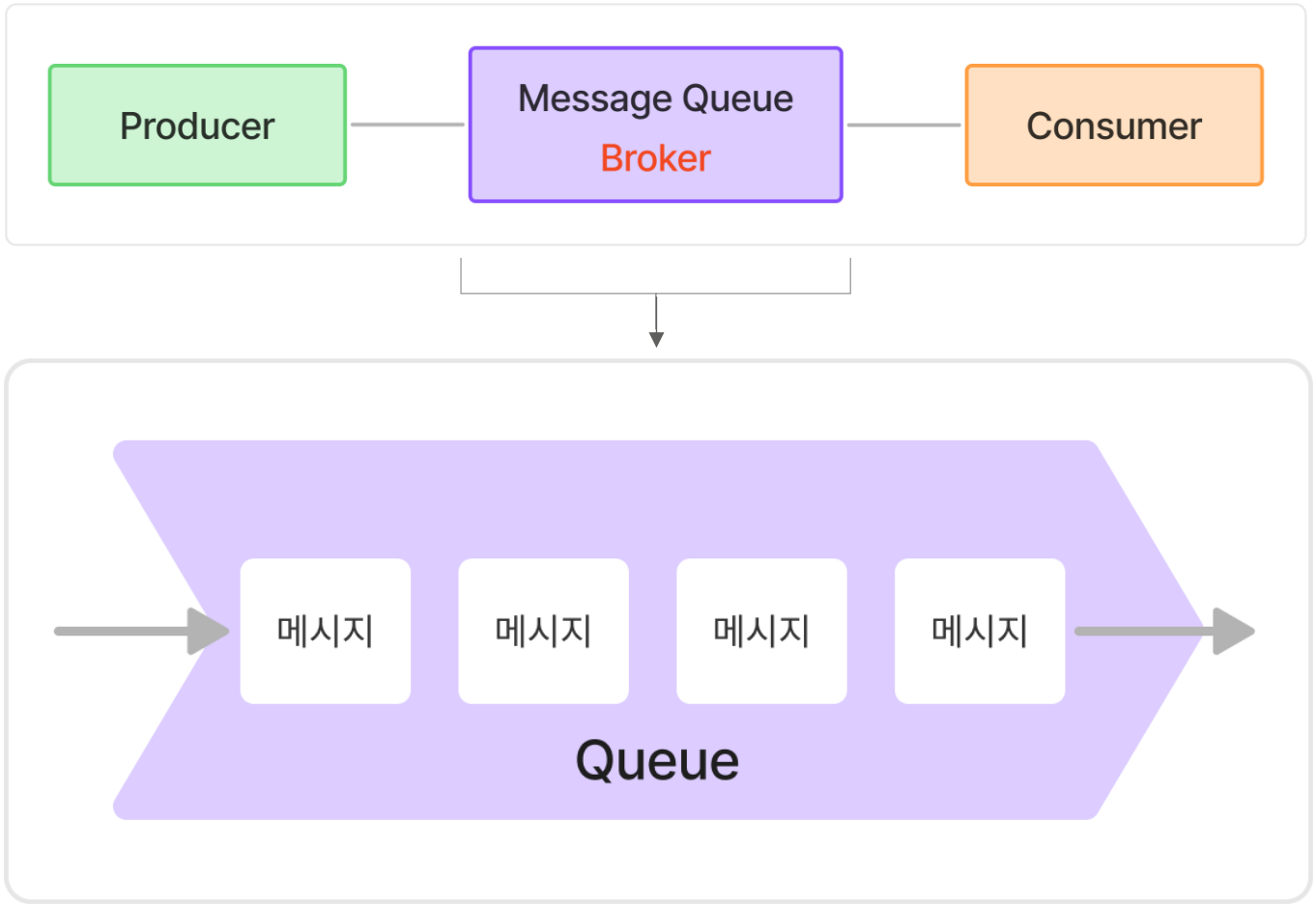


응용 SW 간 비동기적 데이터 통신

비동기 메시징

느슨한 결합

Part 2 카프카의 핵심 요소 | 메시징 큐



큐 구조를 이용해서 구현한 MoM

비동기 메세징

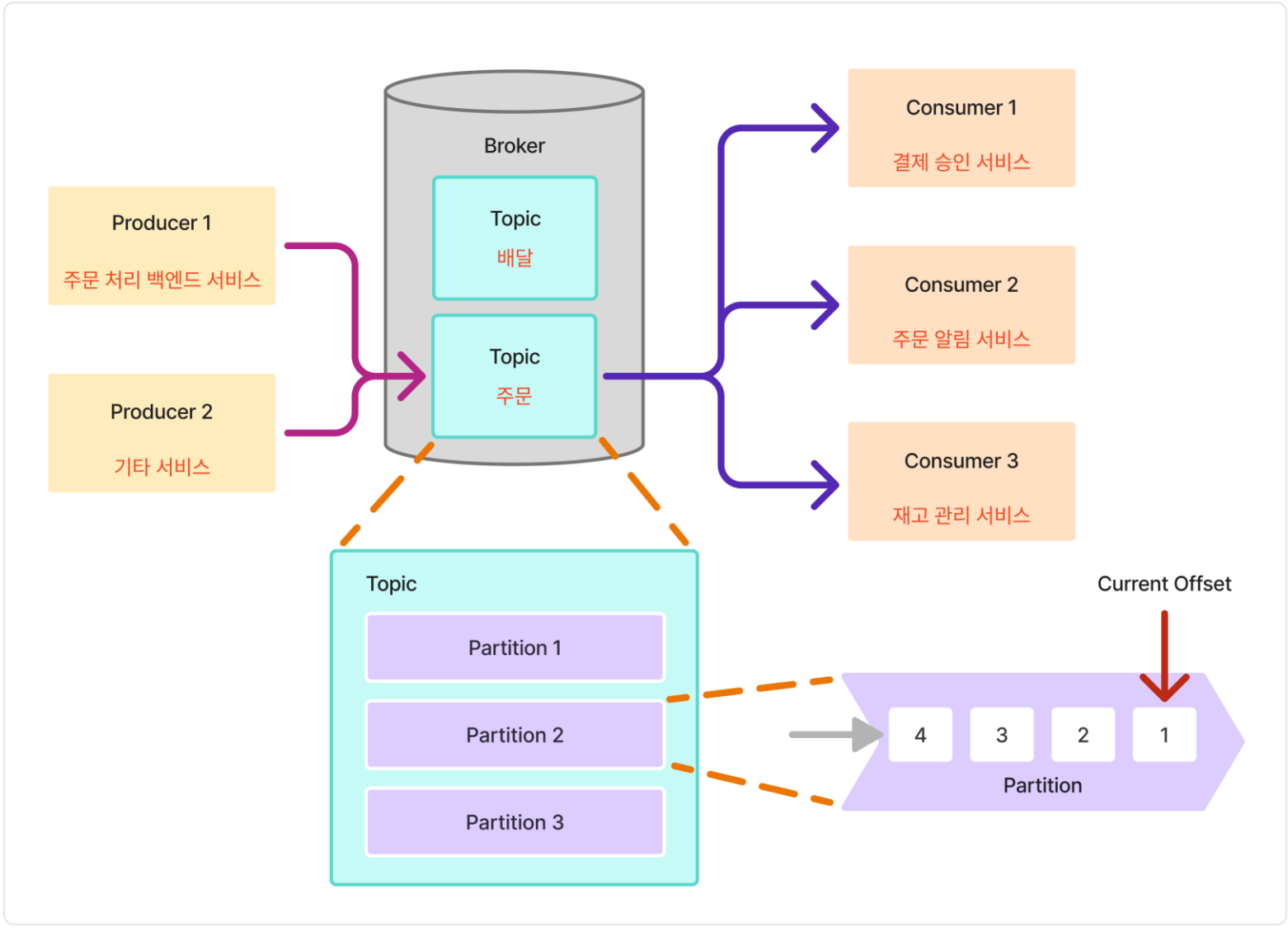
메시지 손실 위험 및 재처리 어려움

대량의 실시간 데이터 처리 한계

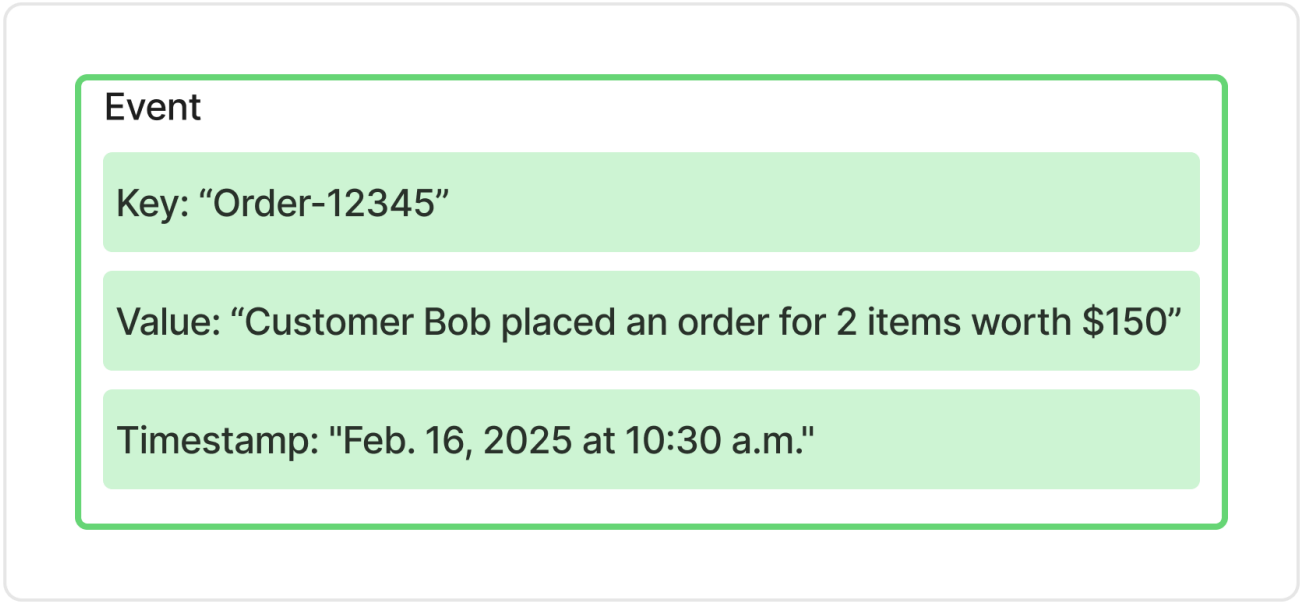
카프카의 핵심 요소 | 카프카란?

고성능 분산형 이벤트 스트리밍 플랫폼

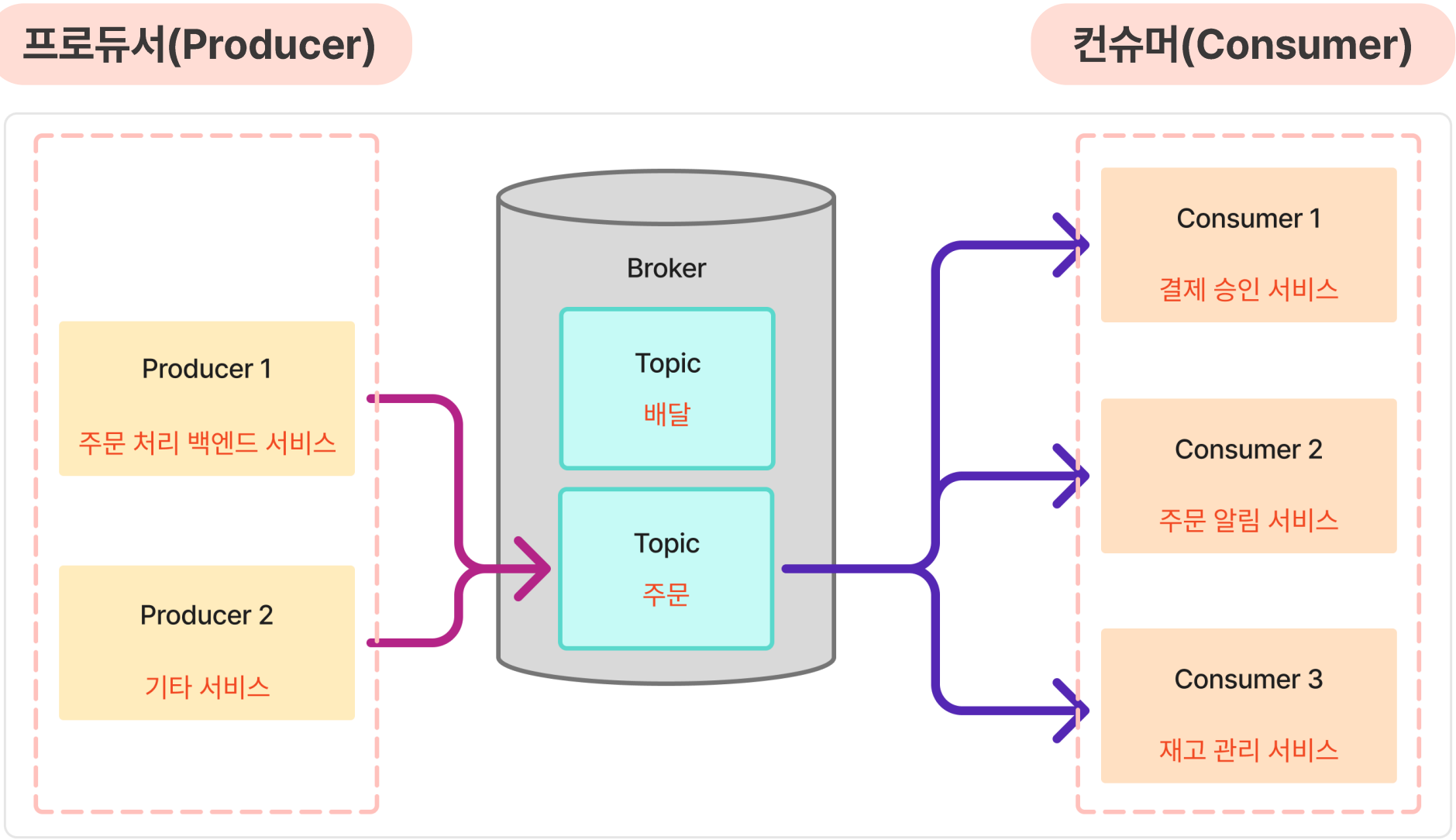
예 : 이커머스 주문 시스템


Broker
Topic
배달
Topic
주문
Producer 1
주문 처리 백엔드 서비스
Producer 2
기타 서비스
Consumer 1
결제 승인 서비스
Consumer 2
주문 알림 서비스
Consumer 3
재고 관리 서비스
Topic
Partition 1
Partition 2
Partition 3
Current Offset
4
3
2
1
Partition

이벤트(Event)

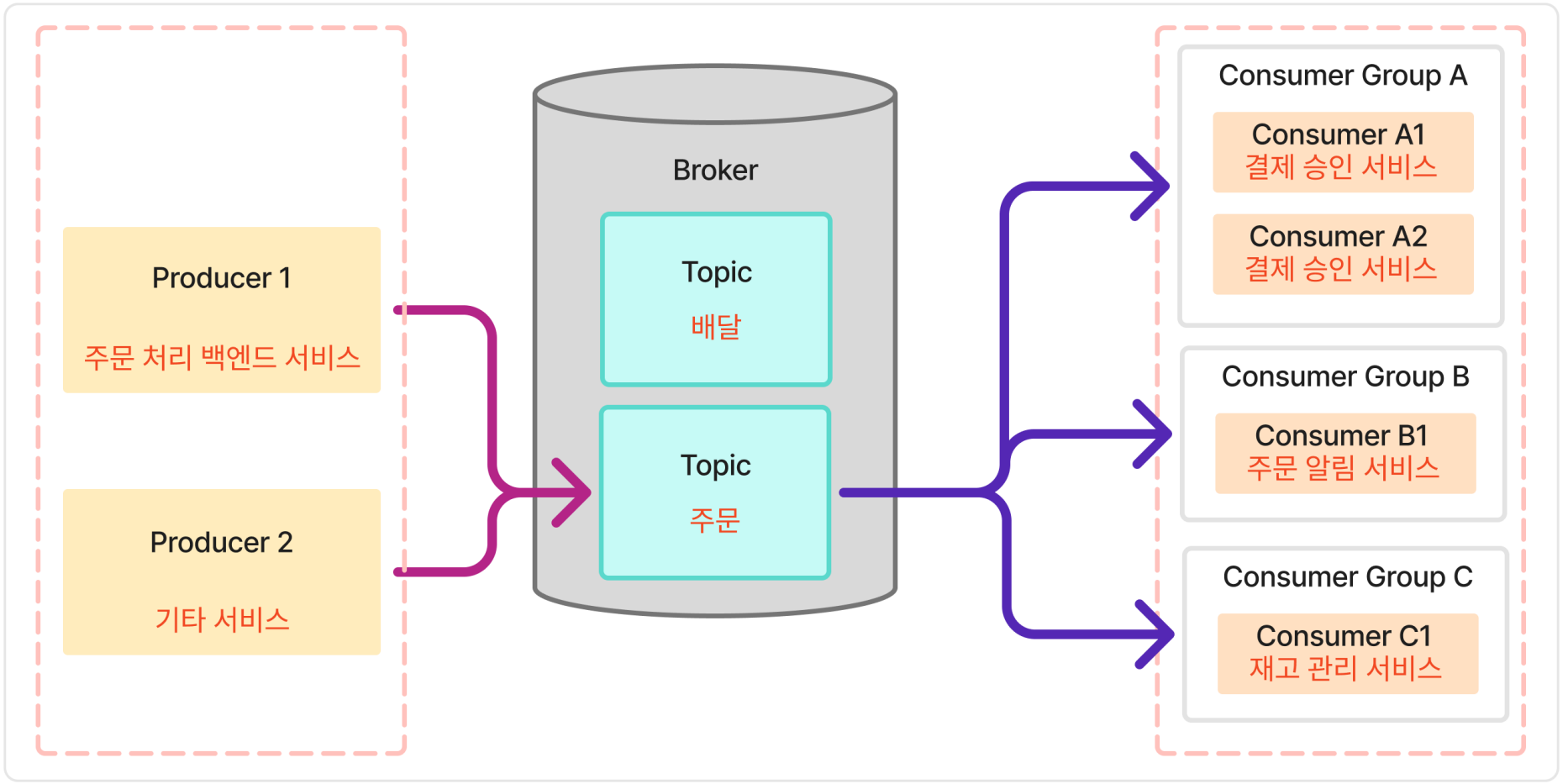


- Kafka에서 데이터를 저장하고 처리하는 기본 단위
- “무언가 발생했다”
- 이벤트(event), 레코드(record), 또는 메시지(message)


프로듀서(Producer)
컨슈머(Consumer)
Producer 1
주문 처리 백엔드 서비스
Producer 2
기타 서비스
Broker
Topic
배달
Topic
주문
Consumer 1
결제 승인 서비스
Consumer 2
주문 알림 서비스
Consumer 3
재고 관리 서비스

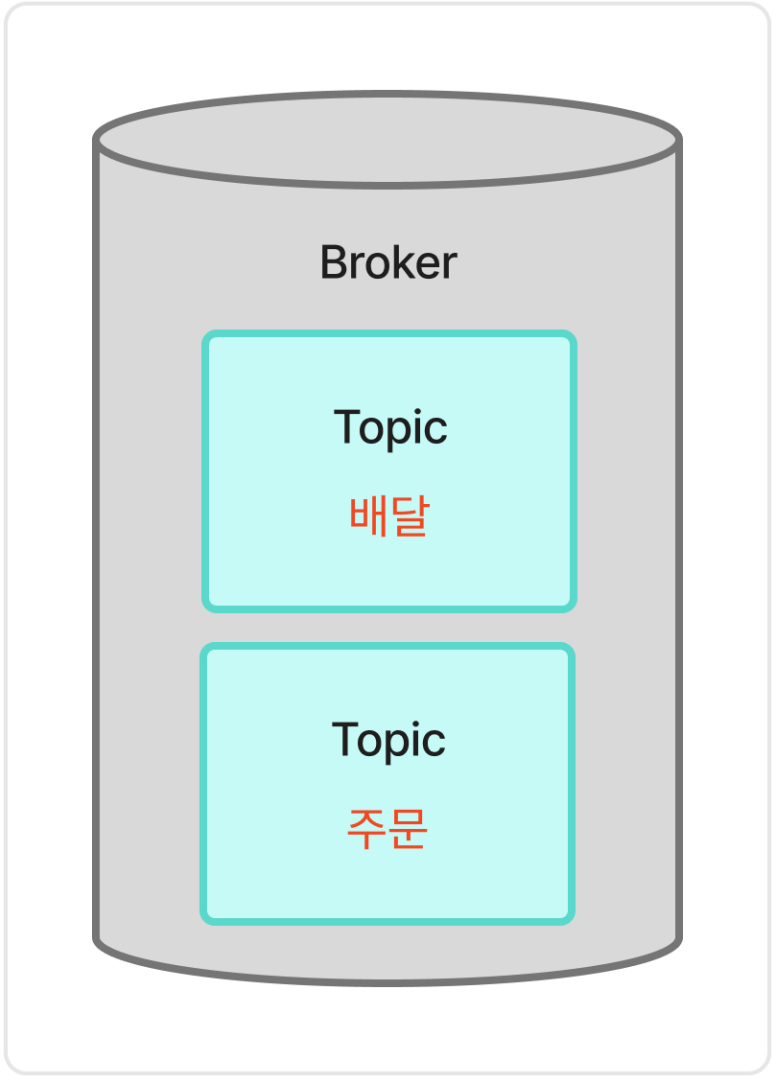
프로듀서(Producer)

컨슈머 그룹(Consumer Group)



토픽

- 이벤트가 저장되는 논리적인 단위

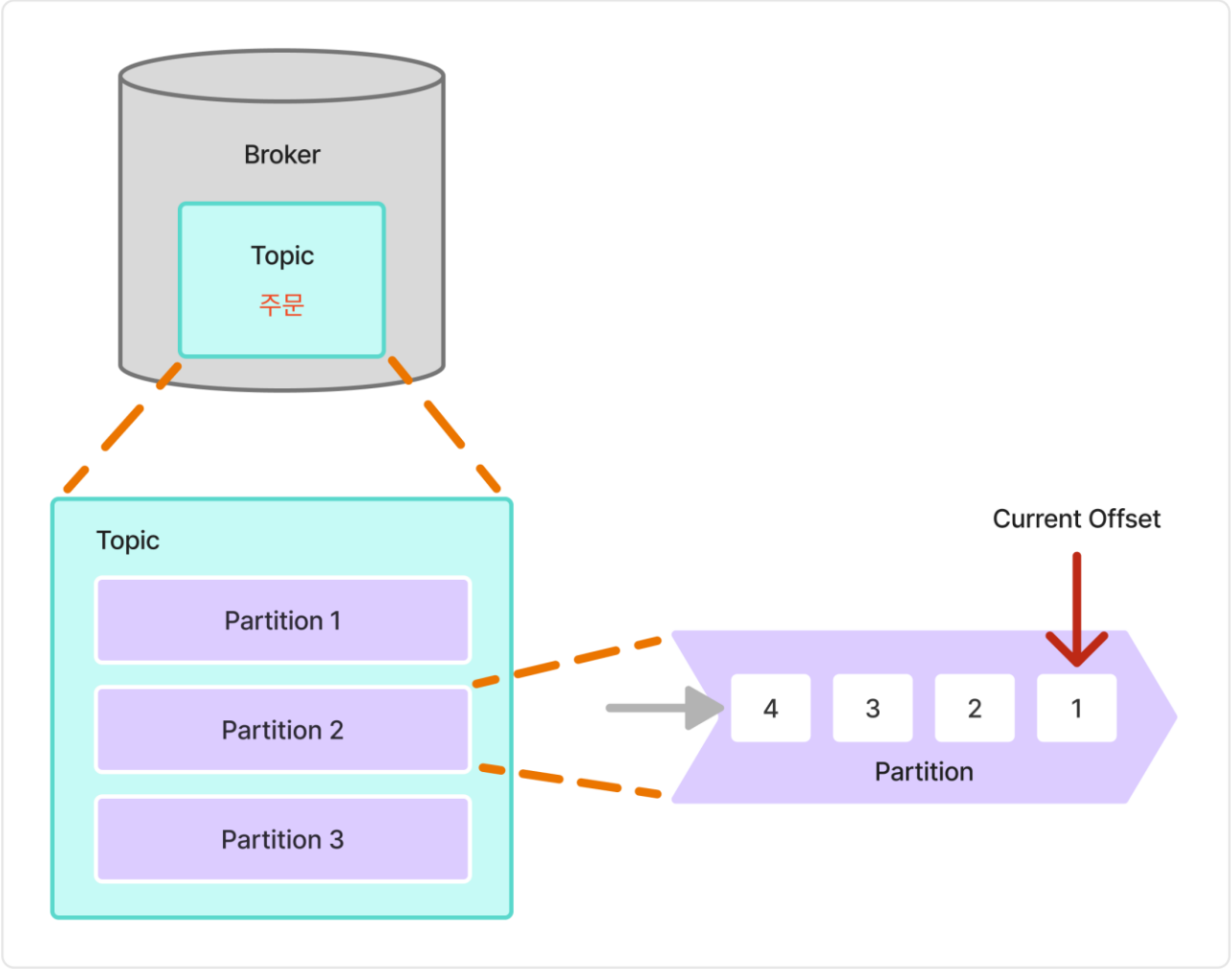


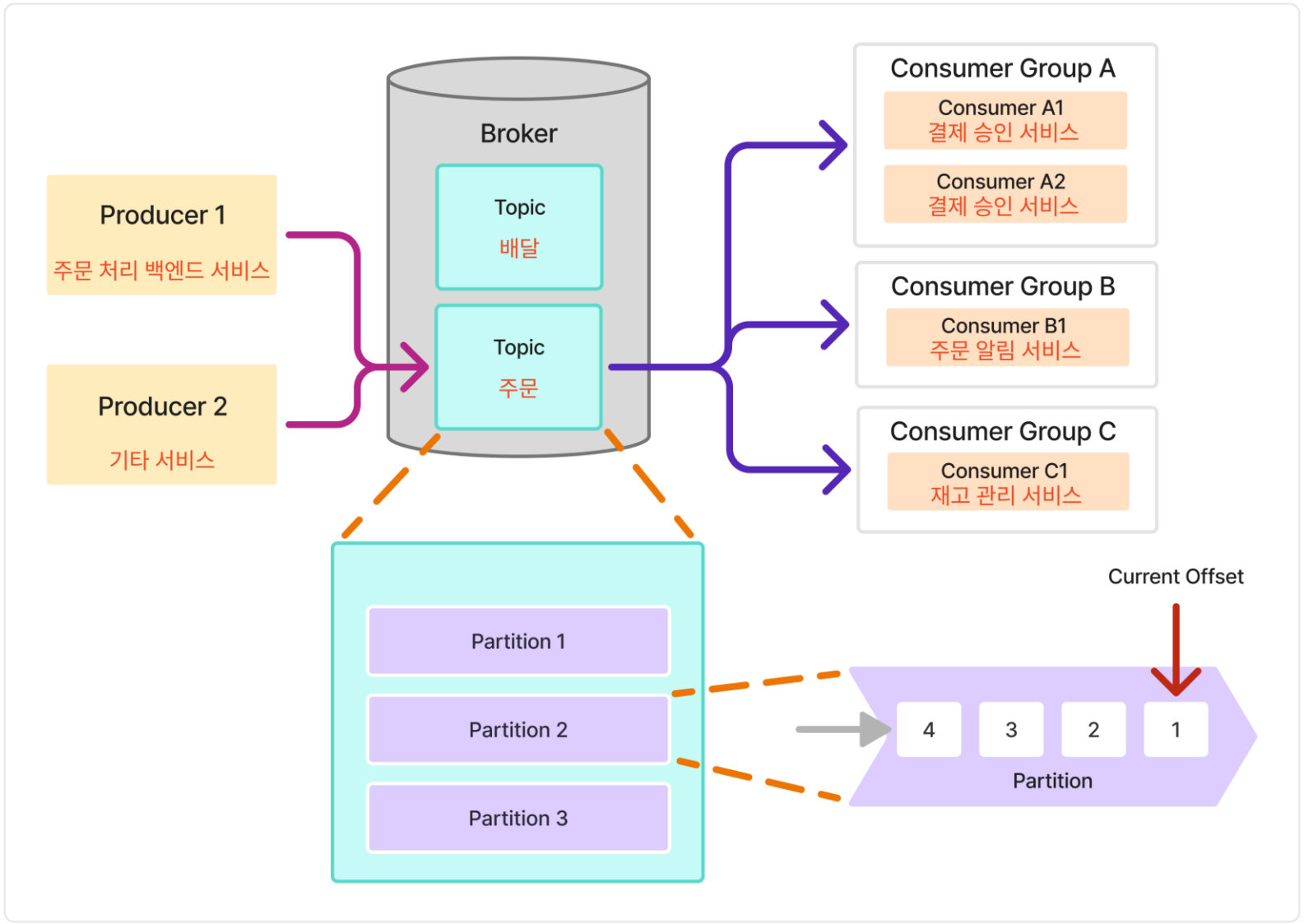
파티션

- 토픽을 분류하는 물리적 단위
- 이벤트가 실제로 저장되는 단위

Offset

- 파티션 내에서 이벤트의 고유한 위치
- Consumer는 Offset을 기반으로 기억 (어디까지 데이터를 읽었는가)





Step 1

프로듀서가 이벤트 발행

Step 2

원하는 토픽에 이벤트 전송

Step 3

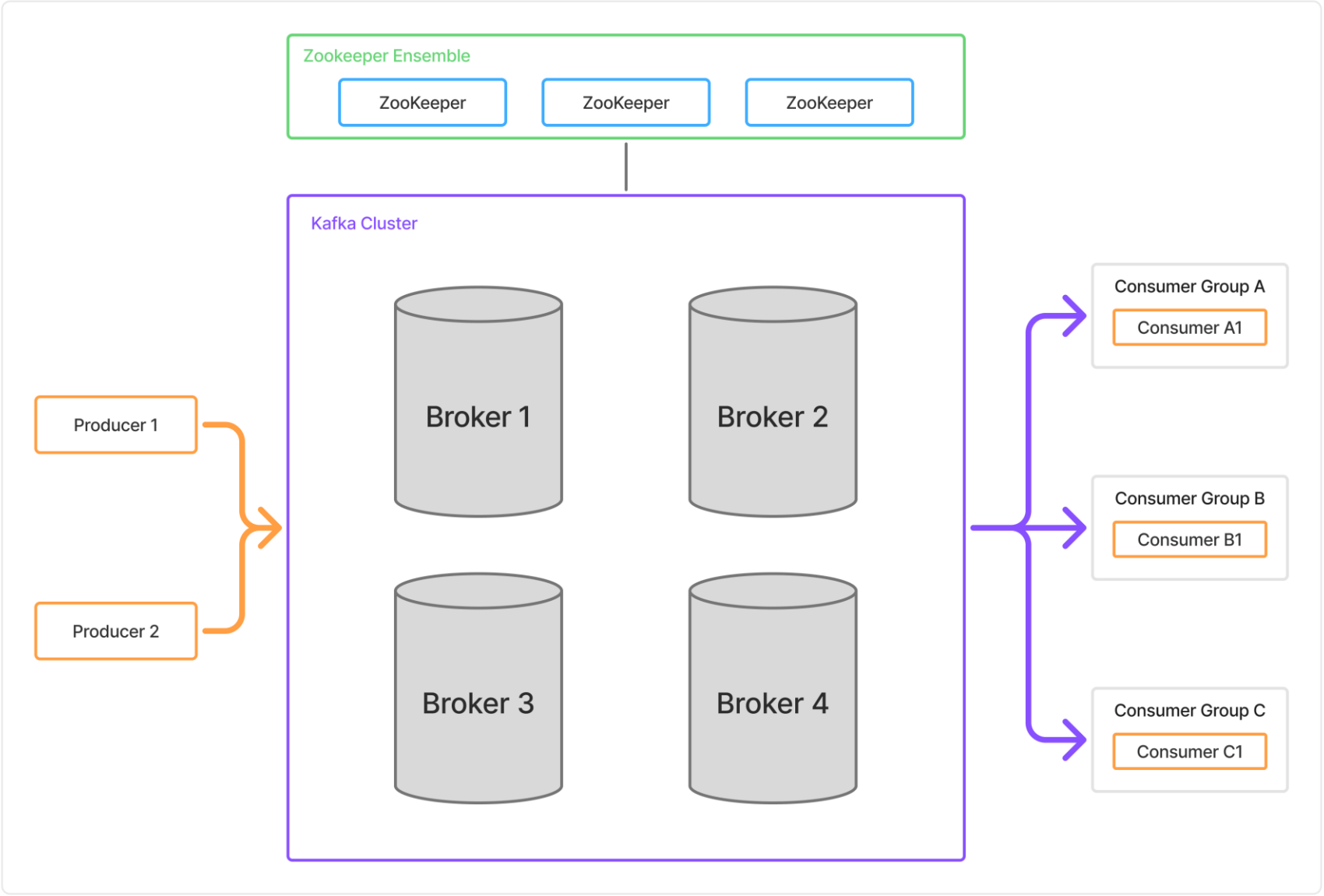
파티션에 분산 저장

Step 4

오프셋을 이용해
컨슈머가 이벤트 소비

Part 2

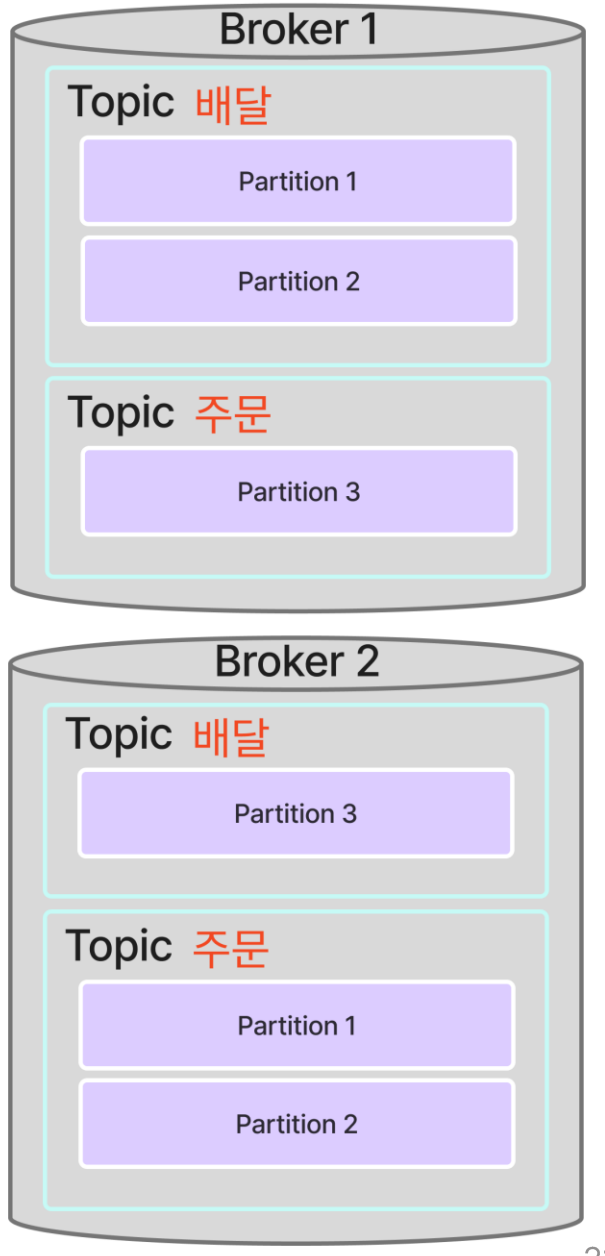
카프카의 핵심 요소 | 카프카 클러스터

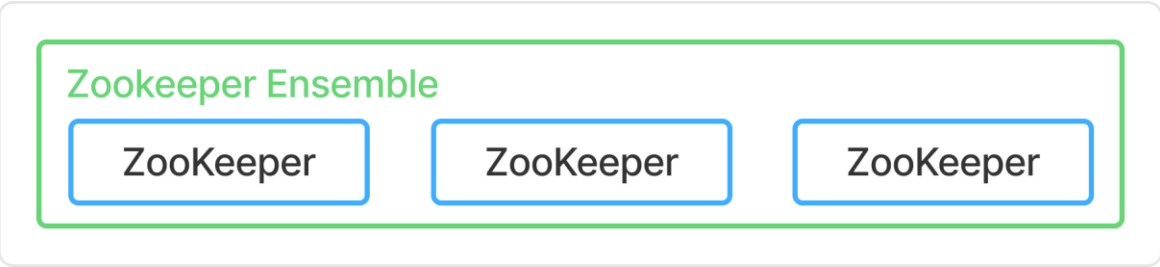


Part 2 **카프카의 핵심 요소 | 카프카 클러스터**

브로커(Broker)

- 카프카 핵심 서버
- 디스크에 이벤트(메시지) 저장
- 컨슈머의 요청에 따라 데이터 전달(Pull 방식)
- 컨슈머가 소비한 메시지는 브로커에서 바로 삭제 되지 않고 유지
- 클러스터 내에서 토픽과 파티션을 저장 및 관리
- 각 브로커는 특정 토픽의 일부 파티션을 저장 및 관리





운영 복잡성 증가

Zookeeper 장애 시 문제

확장성 제한

Zookeeper

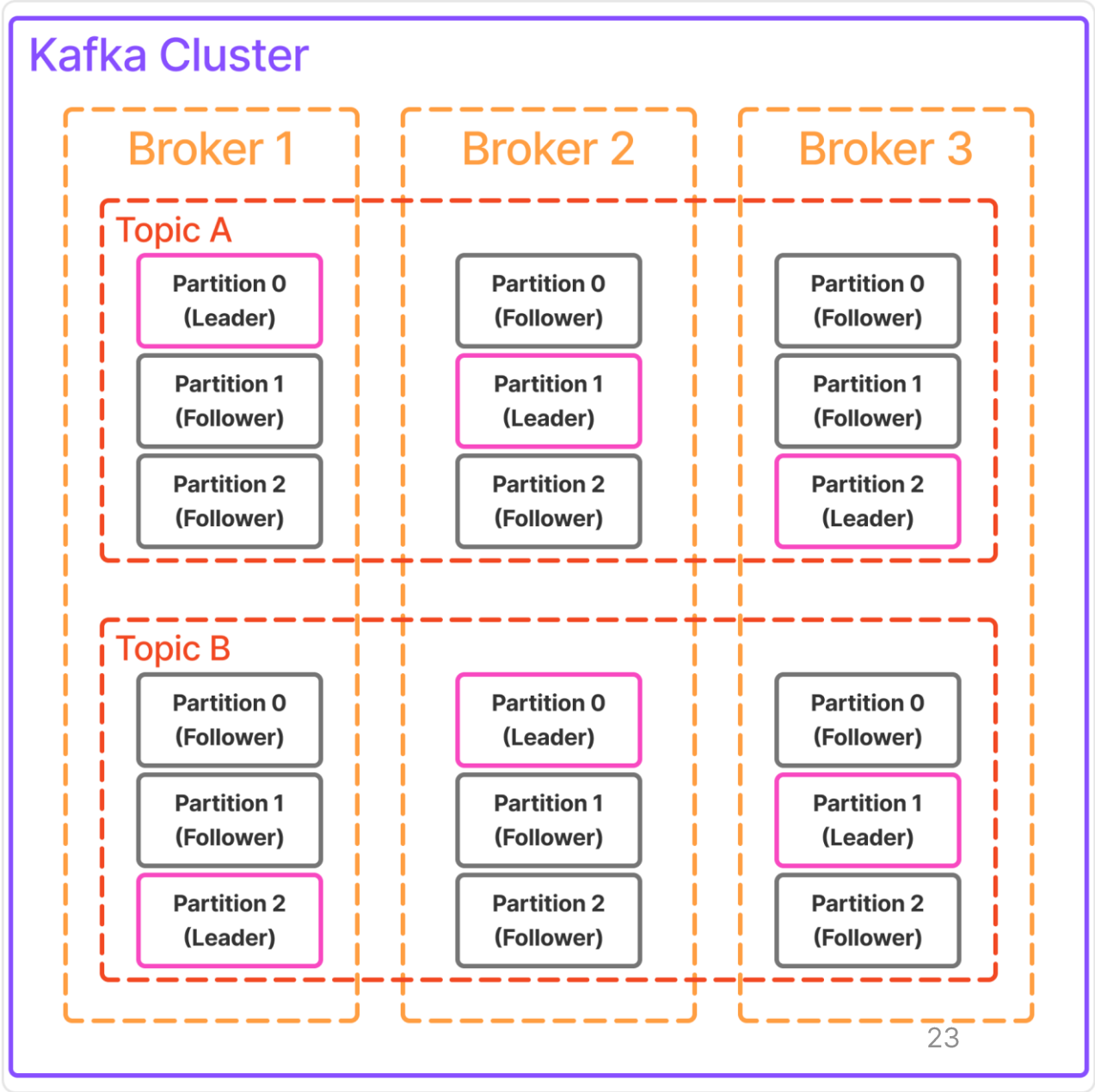
- 클러스터 내 브로커 관리
- 컨트롤러 브로커 선출
- 클러스터의 메타데이터 저장 및 관리
- Kafka 클러스터의 중앙 조정자 역할

Kraft(Kafka Raft)

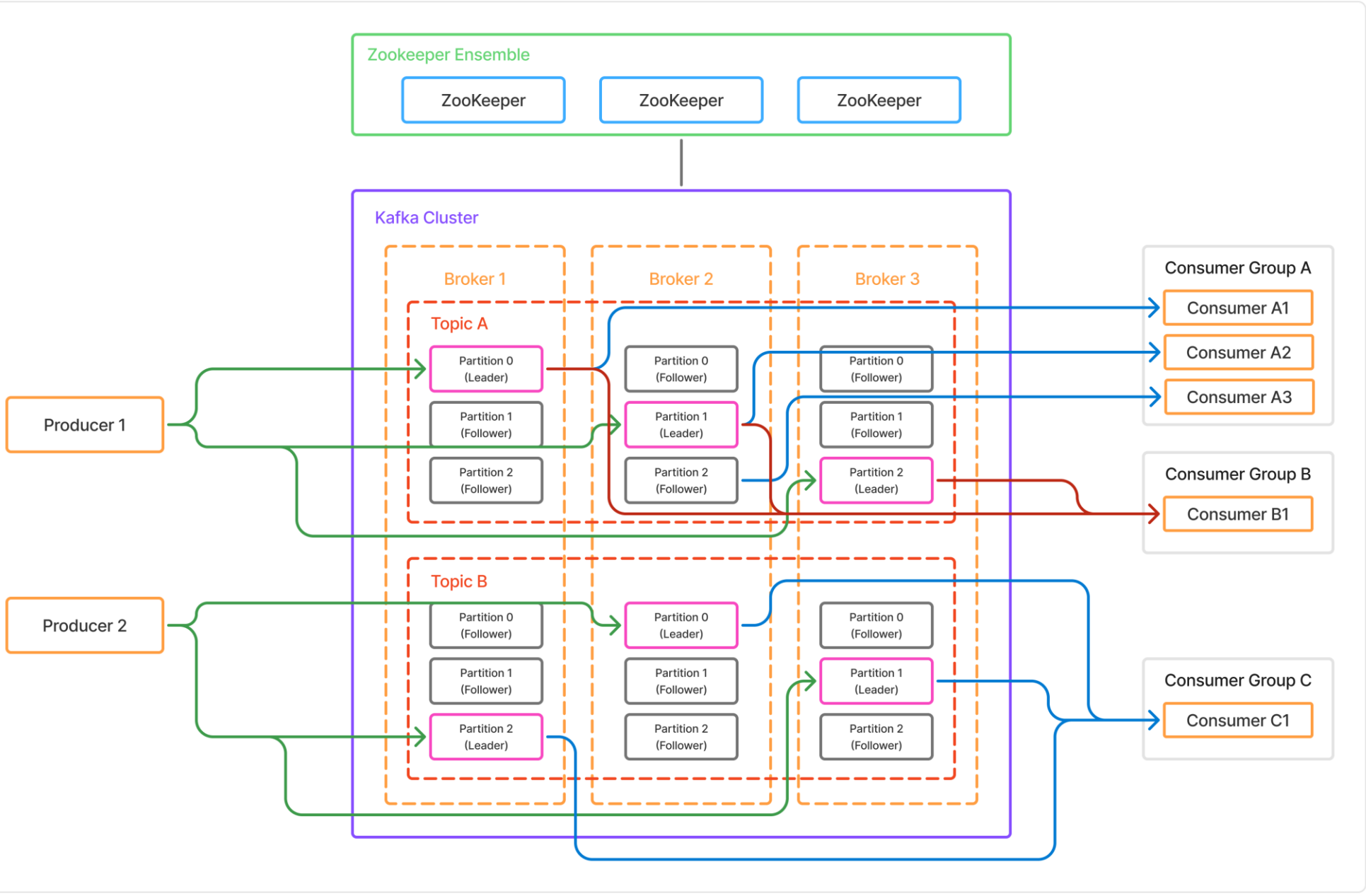
- Zookeeper 없이 동작하는 모드
- Kafka가 자체적으로 클러스터 및 메타데이터 관리
- Raft 알고리즘 기반

토픽 복제(Topic Replication)

- 토픽의 복제계수(Replication Factor) : N
  -> N개의 브로커에 동일한 데이터를 저장
- 각 파티션을 여러 개의 브로커에 복제
  -> 1 = Leader, 나머지 = Follower



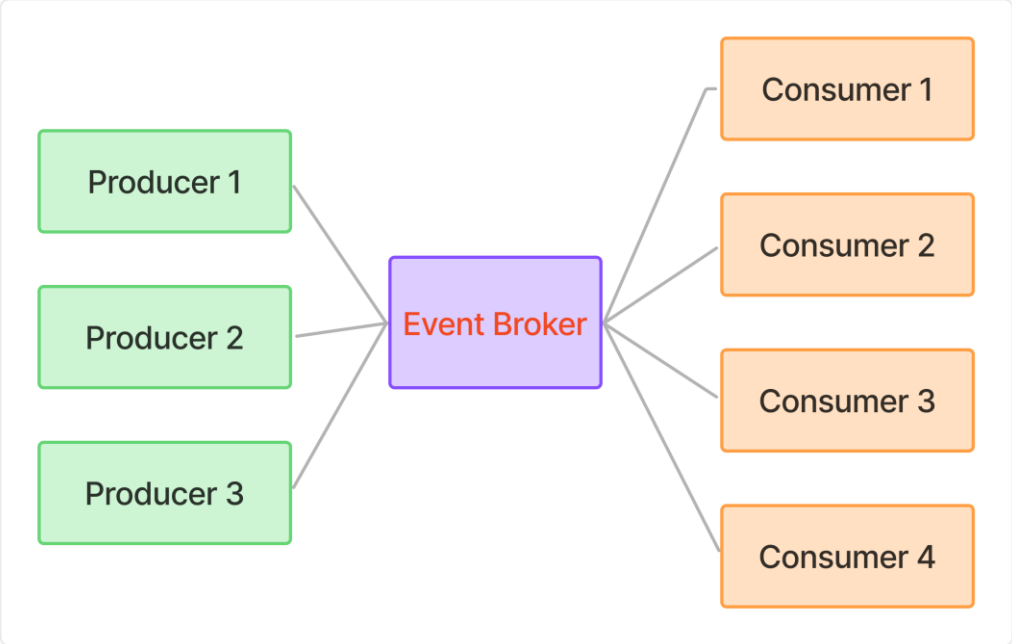
카프카의 핵심 요소 | 카프카 클러스터



Part 3

카프카의 특장점

카프카의 특장점 | 중앙 집중형 이벤트 허브 제공



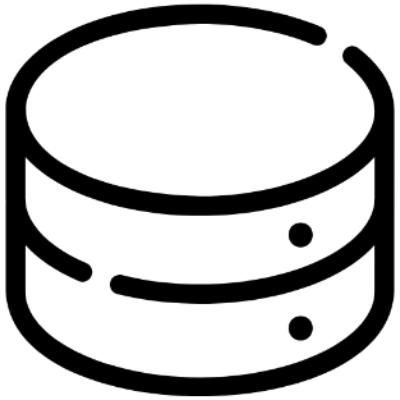
N:M 통합을 위한 단일 플랫폼

- **중앙** 허브
- **Pub-Sub** 모델

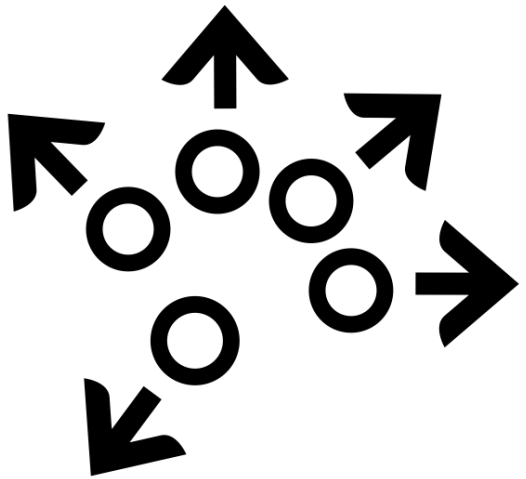
시스템간 느슨한 결합(Decoupling)

- **Producer**와 **Consumer**가 서로를 **모름**
- 새로운 시스템 **확장** 시 **기존 시스템**에 영향을 주지 **않음**

디스크 기반 메시지 저장 (Log-based Storage)



- **장기간** 데이터 보관
- **Retention Policy**
  - 설정된 **보관 기간** 동안 메시지 유지
- **로그 압축**(Log Compaction)
  - **오래된 데이터 정리** 후 최신 데이터 유지
- **과거 데이터 재처리**(Replay)
  - **장애 발생** 시 특정 시점의 데이터 **다시 처리**

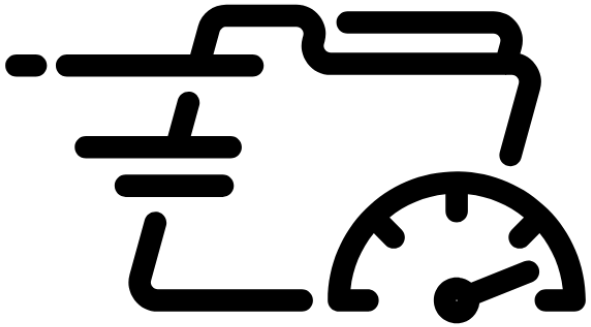


높은 확장성

- **Cluster** (Broker) 확장성
  - **선형적** 성능 확장 가능
- **Partition** 단위 **분산 처리**
  - 데이터 **저장** 및 **처리량 조절** 가능
- **Consumer** 확장성
  - **Consumer 그룹**
  - 다수의 Consumer가 **병렬**로 메시지 소비 가능

분산 시스템 기반 고가용성 보장

- **Follower**들이 **Leader**의 데이터를 동기화
- 분산 시스템으로 **높은 안정성**
- **자동 장애 복구**(Failover)
- 장애 발생 시 새로운 Leader를 **자동 선출**하여 **서비스 중단 최소화** 및 **고가용성** 유지



메시지 처리 보장

- **At-least-once**
  - 메시지 **유실 방지**
- Producer **acks**
  - 데이터 **손실 최소화**
- Consumer **offset** 관리
  - 메시지 **순서 보장** 및 **재처리** 지원

높은 처리량

- **단일 브로커**에서 **초당 수십만 개**의 메시지 처리
- **배치 처리** 및 **메시지 압축**으로 네트워크 **오버헤드 감소**

병렬 처리

- **파티션** 단위 **병렬** 처리
- **Consumer Group**을 통한 부하 분산
- **토픽** 별 **파티션 수** 조정 가능

Part 4

데모

Part 4 데모 | 시나리오 설명

암호화폐 거래소

- **현재 체결가** 관련 서비스
- 각각의 서비스는 **MSA**로 구성되어 있는 형태

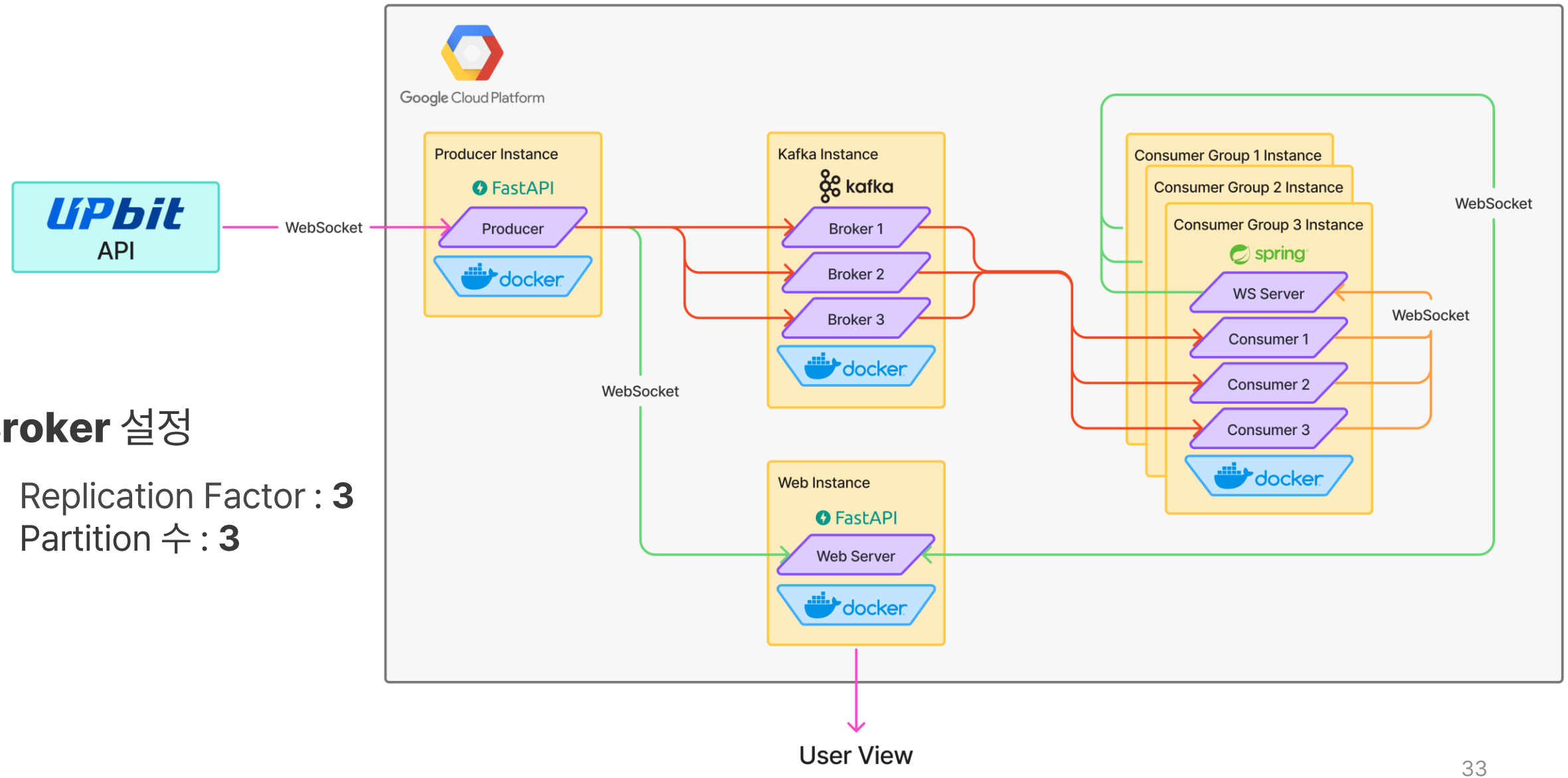
프로듀서

- **거래 체결** 백엔드 서비스

컨슈머

- **실시간 시세** 서비스
- 가격 변동에 따른 **알림** 서비스
- **시장 데이터 분석** 서비스

Part 4 데모 | 아키텍처 설명



Broker 설정

- Replication Factor : **3**
- Partition 수 : **3**

Part 5

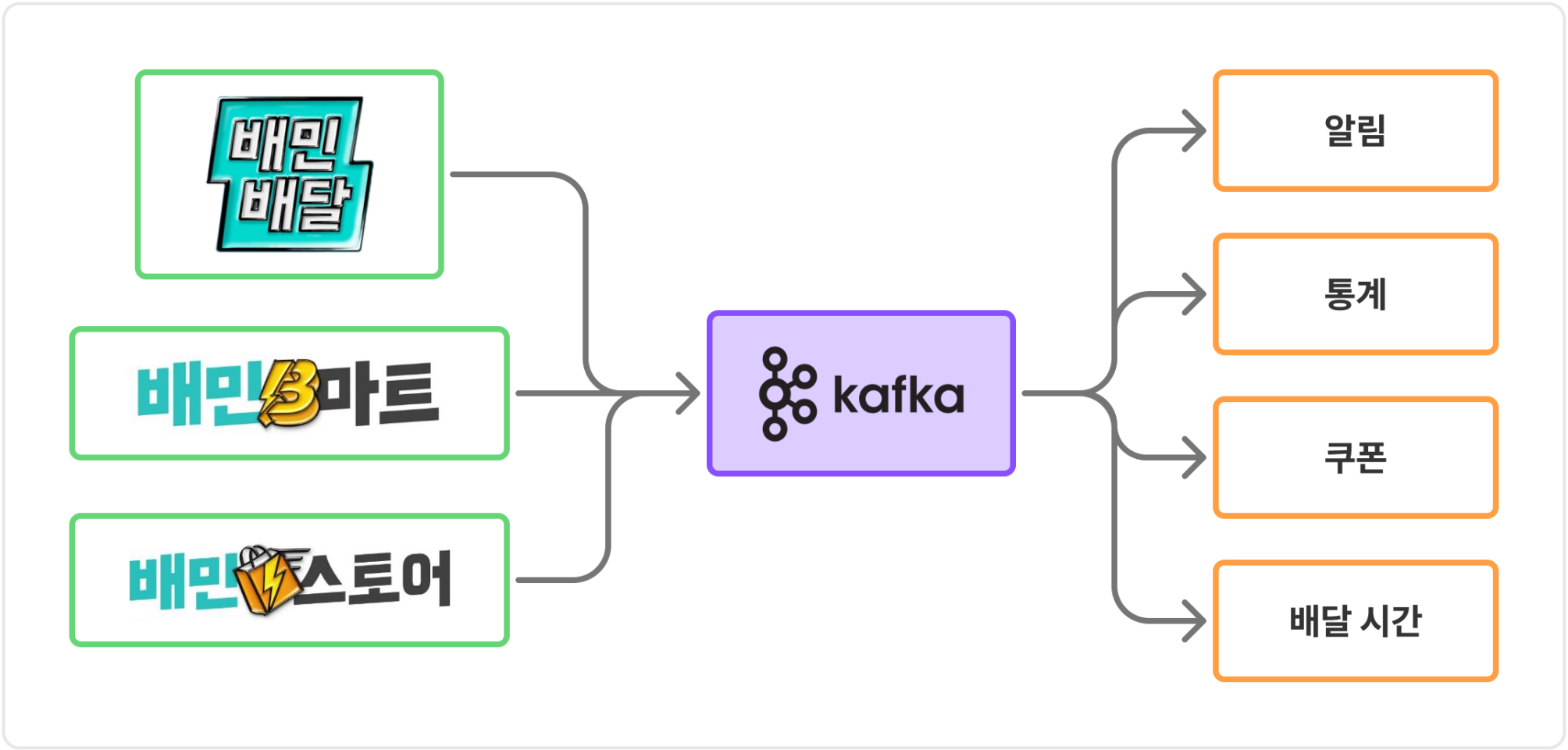
카프카 활용 사례

카프카 활용 사례 | 배달의 민족

첫째	**순서 보장** 이벤트
둘째	이벤트 **버스**
셋째	**실시간 데이터** 분석

Part 5

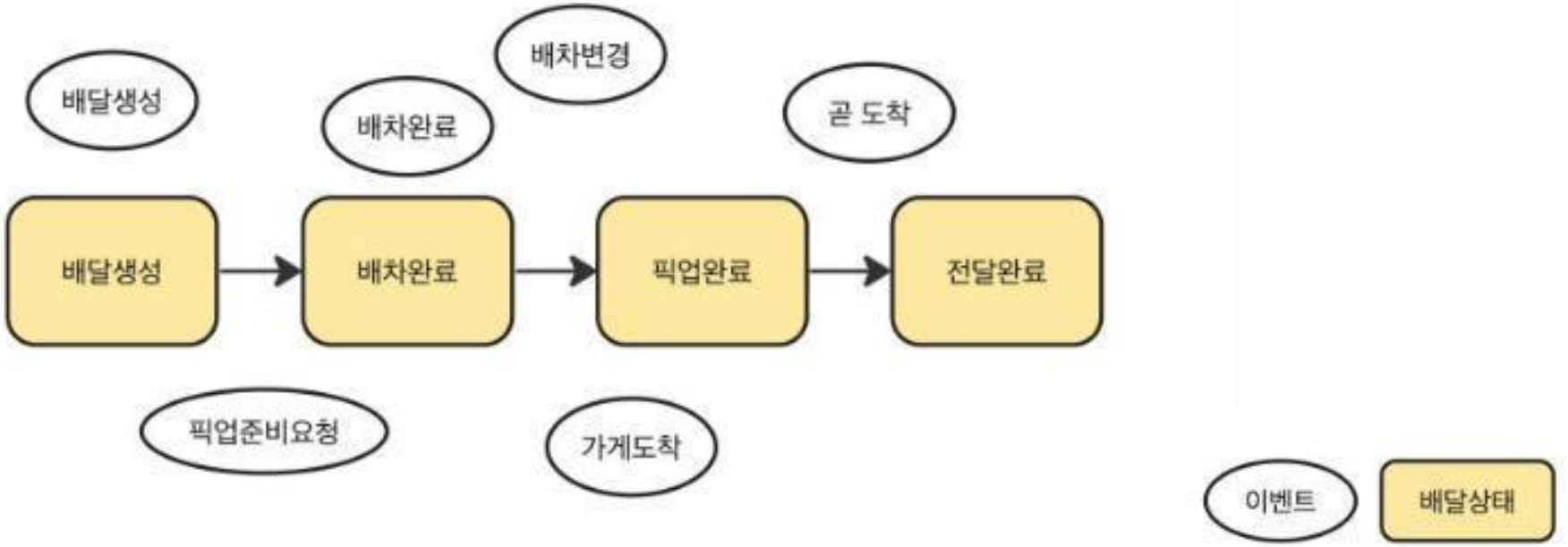
카프카 활용 사례 | 순서 보장 이벤트



주문서비스의 이벤트를 **배달서비스** 중 하나로 분배

- 배달과정을 **중계**하고 **관리**하는 역할

카프카 활용 사례 | 순서 보장 이벤트

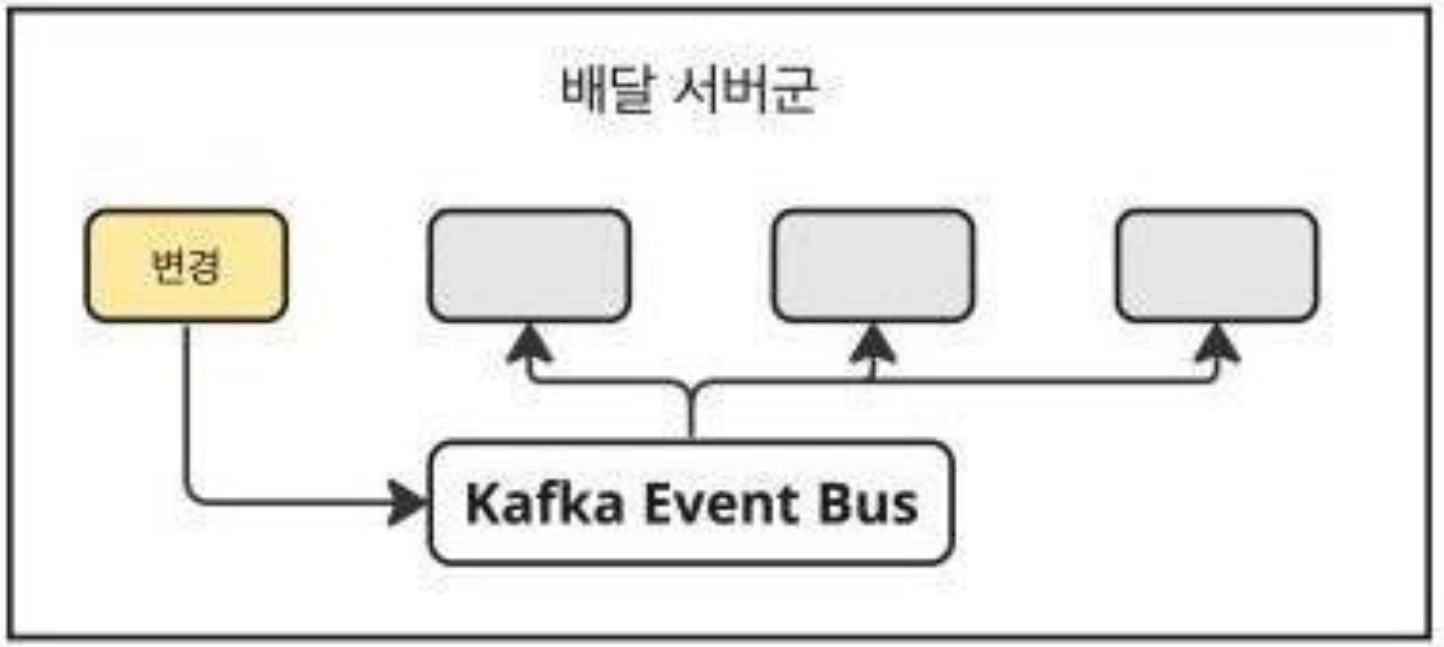


순서 보장 이벤트

- 배달상태는 **순서**가 존재
- **하나의 파티션**에서는 **순서 보장**
  - **같은 키 값**으로 프로듀스 할 시 **하나의 파티션** 보장
- **DBMS**와 연동할 시 **데이터 정합성** 중요

Part 5

카프카 활용 사례 | 이벤트 버스



이벤트 버스

- **분산 시스템**에서 여러 서버의 **값**을 **한번에 변경**해야할 경우
  - **인메모리** 값 **변경 / 초기화**
- 특정 서버에게 **빠르고 동시다발적**으로 값 변경 가능

카프카 활용 사례 | 실시간 데이터 분석

실시간 데이터 분석

- 기존 : **배치**를 이용한 데이터 분석
  - **주기적 실행** → 실시간 데이터 분석 **불가**
- **카프카 스트림즈** 라이브러리를 이용해 **실시간**으로 데이터를 분석

마무리 **결론**

고성능 처리 (High-throughput) **분산** 및 **실시간** 데이터 처리 지원

고성능과 확장성을 갖춘 분산 이벤트 스트리밍 플랫폼

- **마이크로 서비스 아키텍처**(MSA)
- **실시간** 데이터 처리
- **로그** 기반 데이터 저장
- 데이터 **파이프라인** 구축
- **이벤트** 기반 시스템

기타

역할 분배

김대연

- **데모** 담당
- **웹 뷰 / 프로듀서 / 브로커** 개발

민정인

- **발표** 담당
- **특장점, 데모, 사례** 발표

어태규

- **데모** 담당
- **컨슈머** 개발 및 **GCP** 배포

최다영

- **발표** 담당
- **도입 배경, 카프카 개념** 발표

Kafka에 대한 이해

감사합니다

2025.02.17

Team 2 (weAreFoodie)

김대연, 민정인, 어태규, 최다영

기타

자료 출처

[업비트 API]
https://docs.upbit.com/reference/general-info

[FastAPI]
https://fastapi.tiangolo.com/ko/tutorial/

[카프카 클러스터 구성]
https://github.com/apache/kafka/blob/trunk/docker/examples/README.md

[배달의 민족 사용 사례]
https://techblog.woowahan.com/17386/

[카프카 개념]
https://kafka.apache.org/
https://www.youtube.com/watch?v=vHbvbwSEYGo&t=1s

[MSA, Event Driven]
https://www.samsungsds.com/kr/insights/msa_architecture_edm.html

기타

사진 출처

[apache kafka 로고]
https://www.osckorea.com/post/blog-02

[카프카 스트림즈 로고]
https://hevodata.com/learn/kafka-streams/

[픽토그램]
https://thenounproject.com

[비트코인 로고]
https://en.wikipedia.org/wiki/Bitcoin

[업비트 로고]
https://www.freelogovectors.net/upbit-logo/

[Spring 로고]
https://www.logo.wine/logo/Spring_Framework

[FastAPI 로고]
https://tech.osci.kr/fastapi-파이썬으로-간단하게-웹-api-만들기/

[GCP 로고]
https://datasense.be/technology/google-cloud-platform/

[배달의 민족 관련 로고]
https://techblog.woowahan.com/15084/