Kafka에 대한 이해

기술 세미나

2025.02.17

Team 2 (weAreFoodie)

김대연, 민정인, 어태규, 최다영

목차 a table of contents

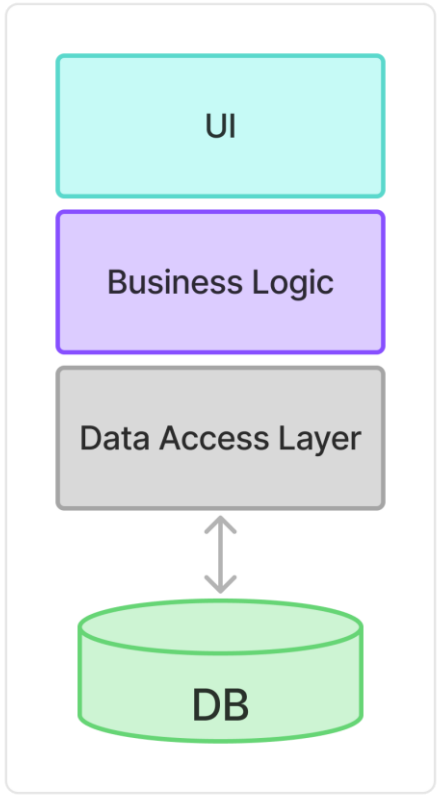
Part 1

도입 배경 및 필요성

Part 1

도입 배경 및 필요성 | 기존 시스템의 문제점

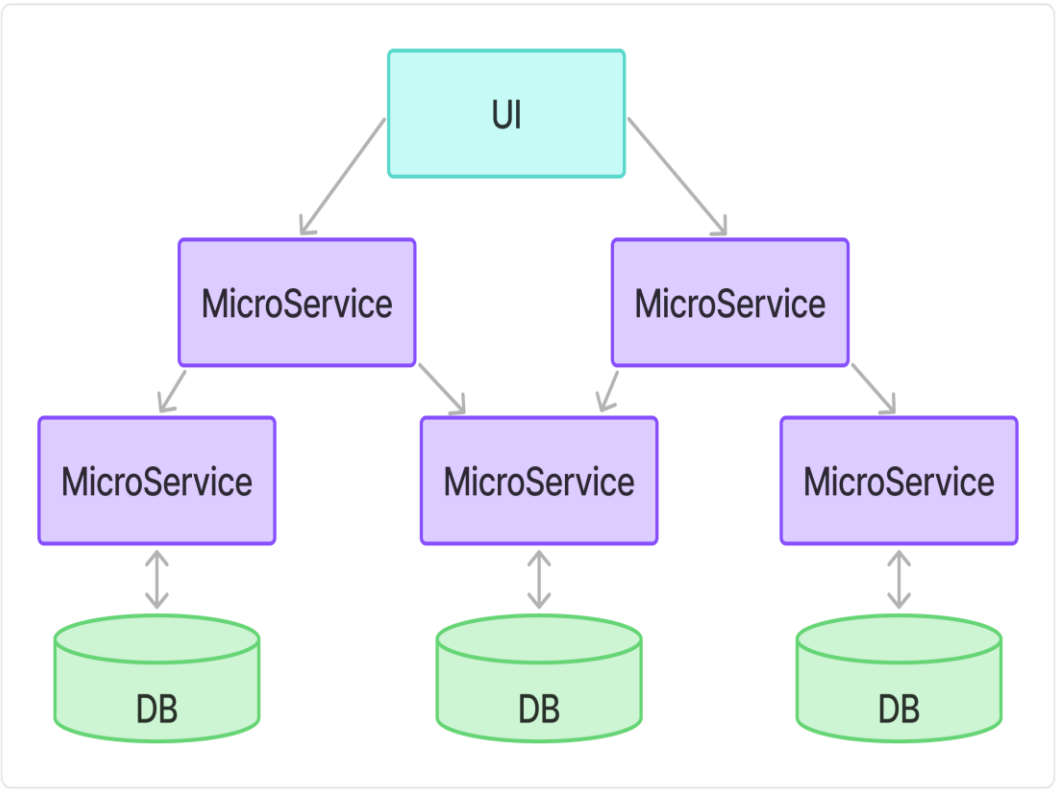
모놀리식 아키텍처



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

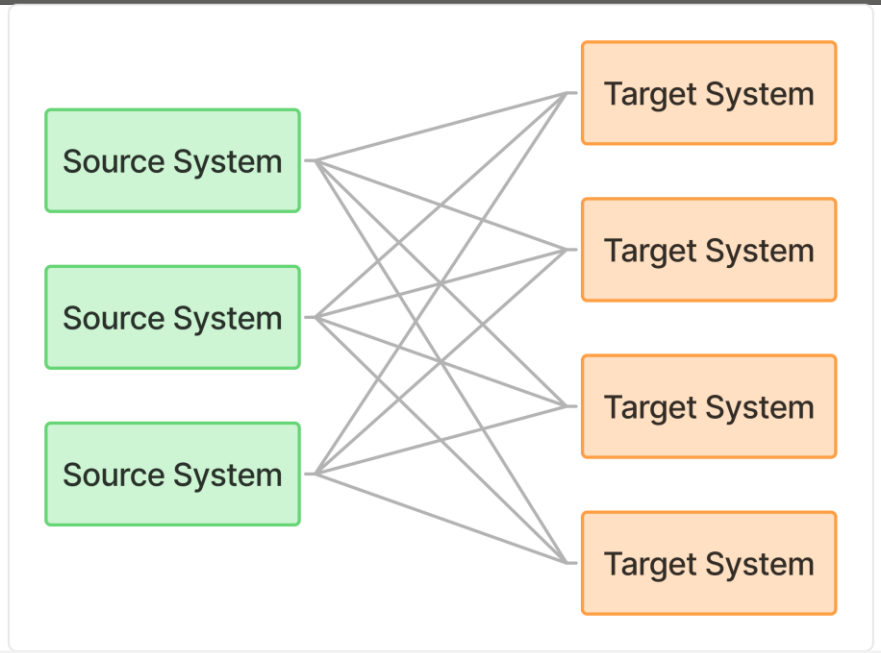
각종 거래 데이터 . . .

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

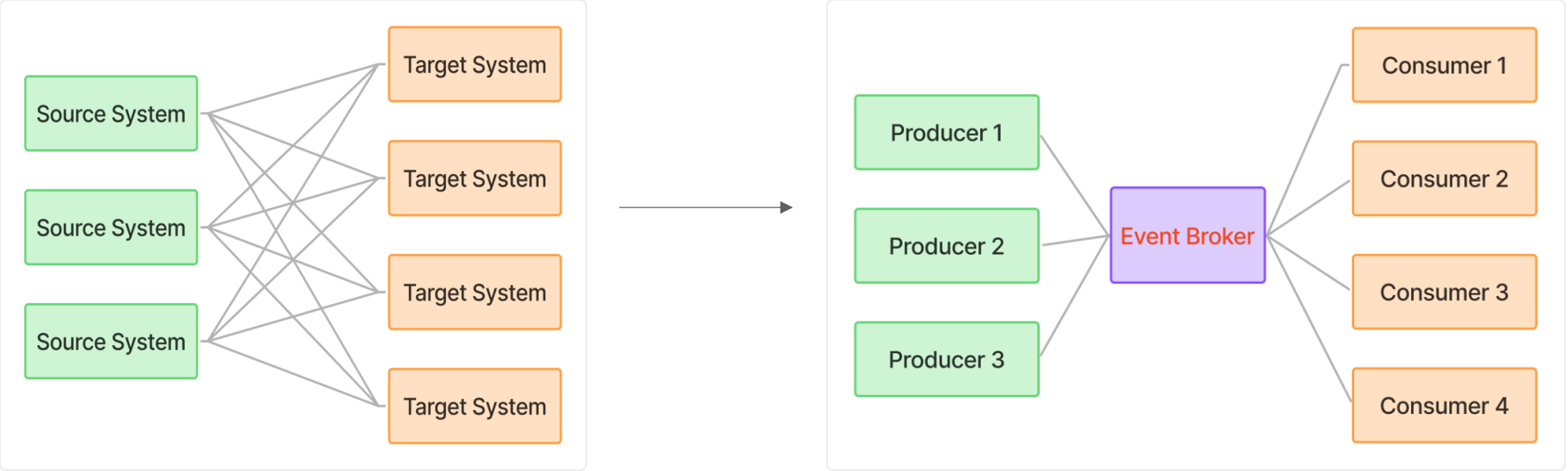
↓

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

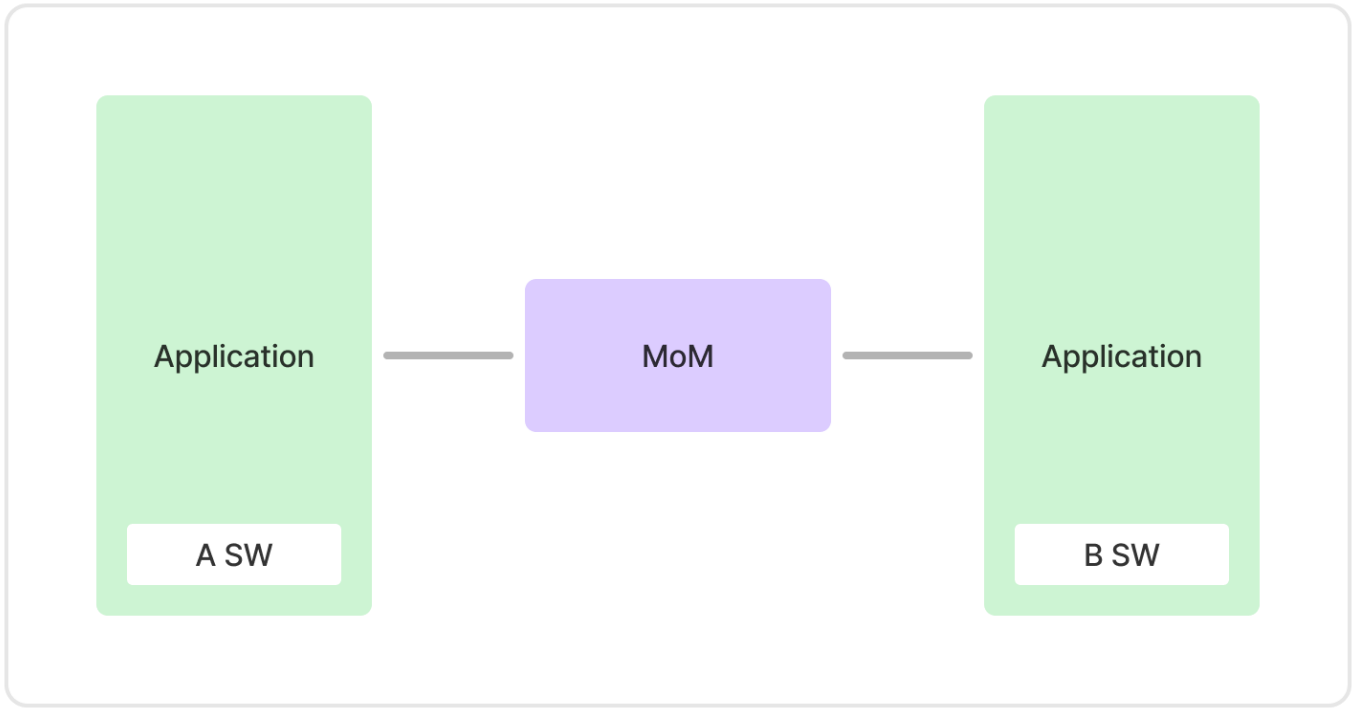
Part 2

카프카의 핵심 요소

Part 2

카프카의 핵심 요소 | MoM

MOM(Message-Oriented Middleware)

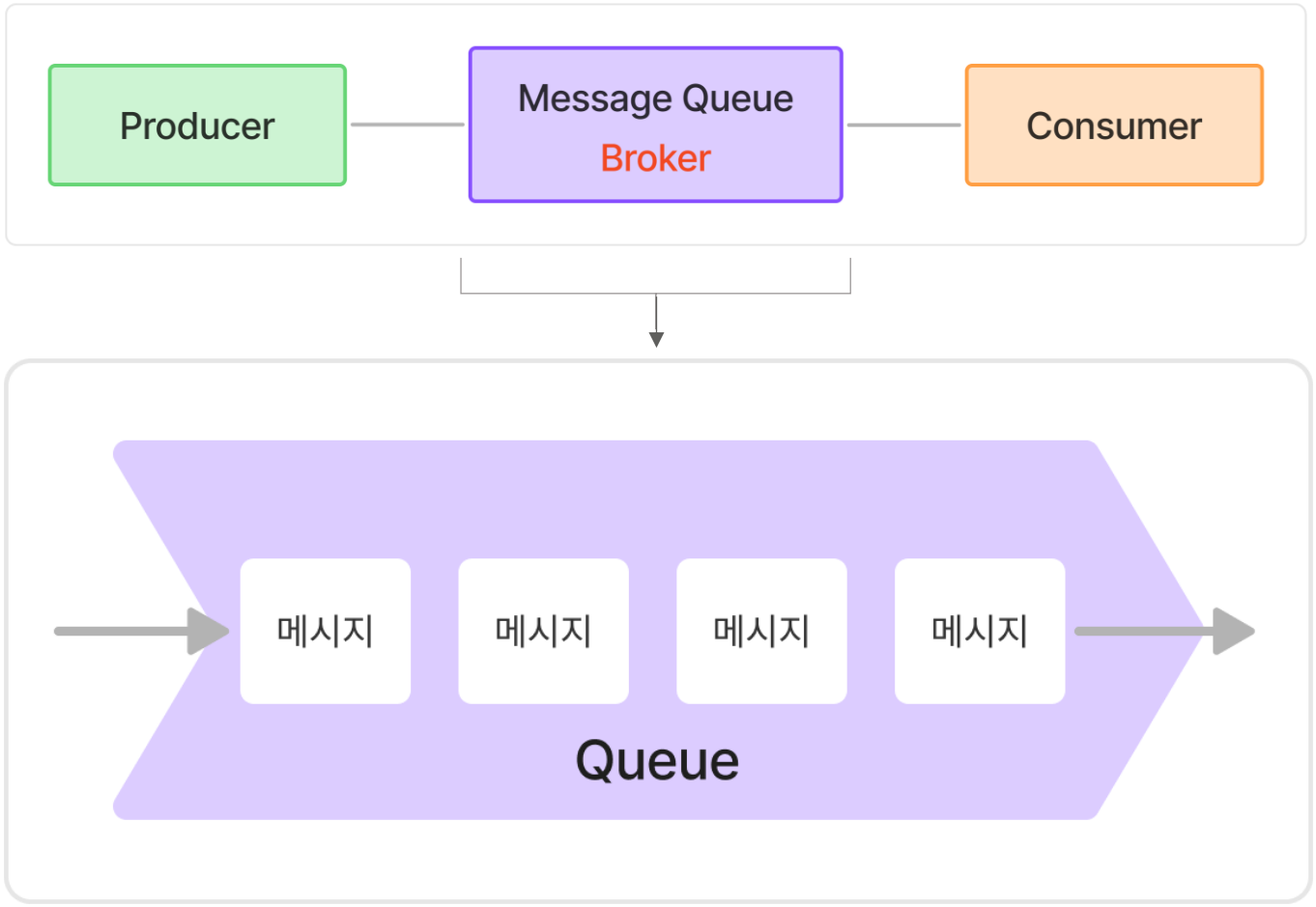


응용 SW 간 비동기적 데이터 통신

비동기 메시징

느슨한 결합

Part 2 카프카의 핵심 요소 | 메시징 큐



큐 구조를 이용해서 구현한 MoM

비동기 메세징

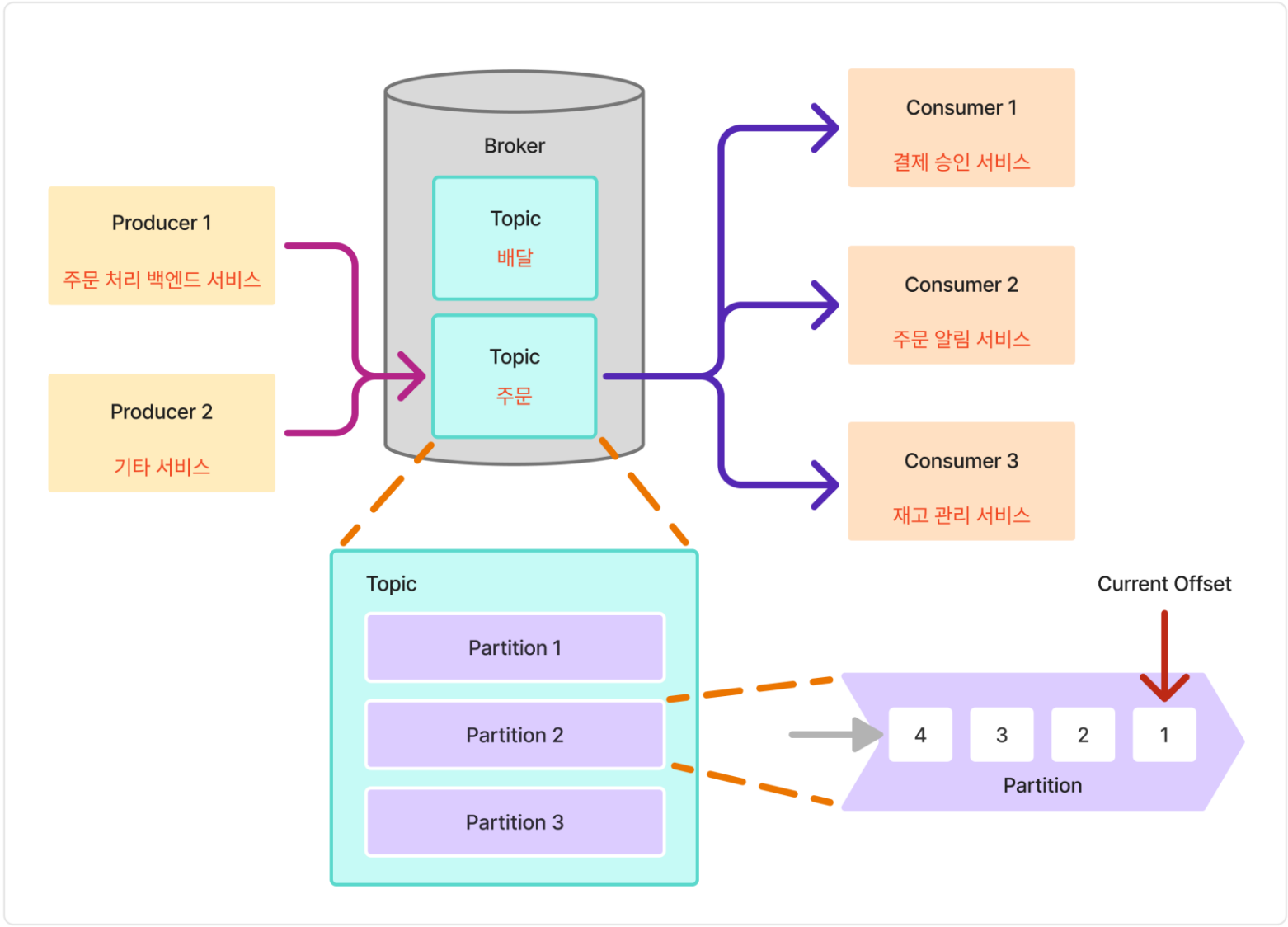
메시지 손실 위험 및 재처리 어려움

대량의 실시간 데이터 처리 한계

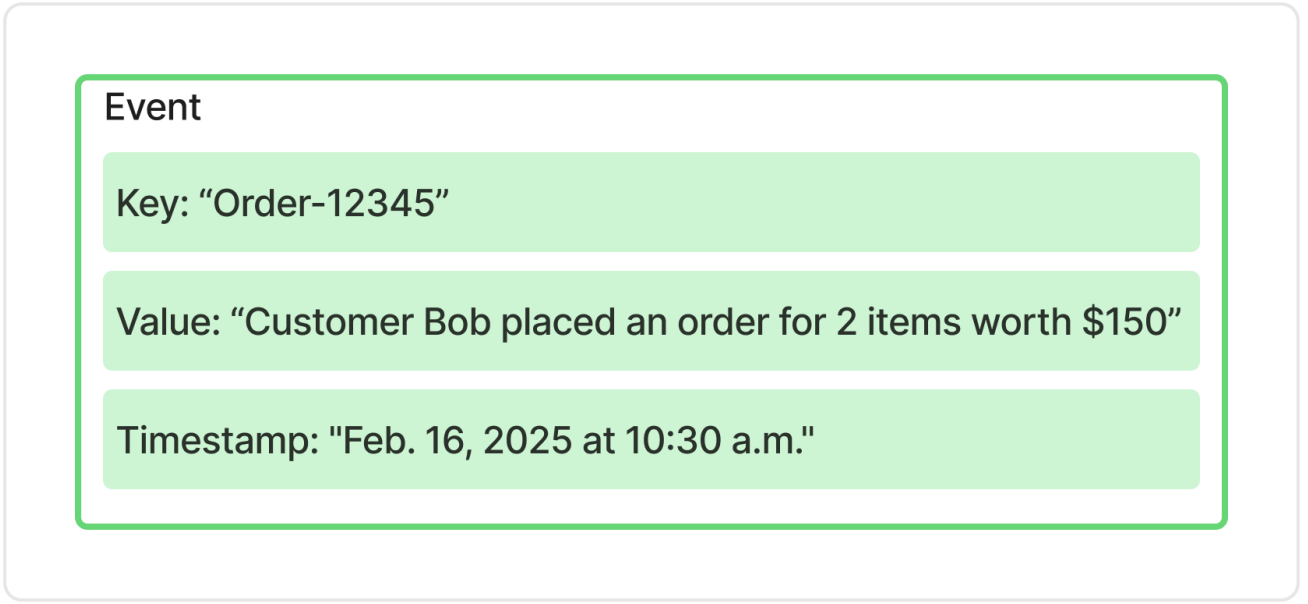
카프카의 핵심 요소 | 카프카란?

고성능 분산형 이벤트 스트리밍 플랫폼

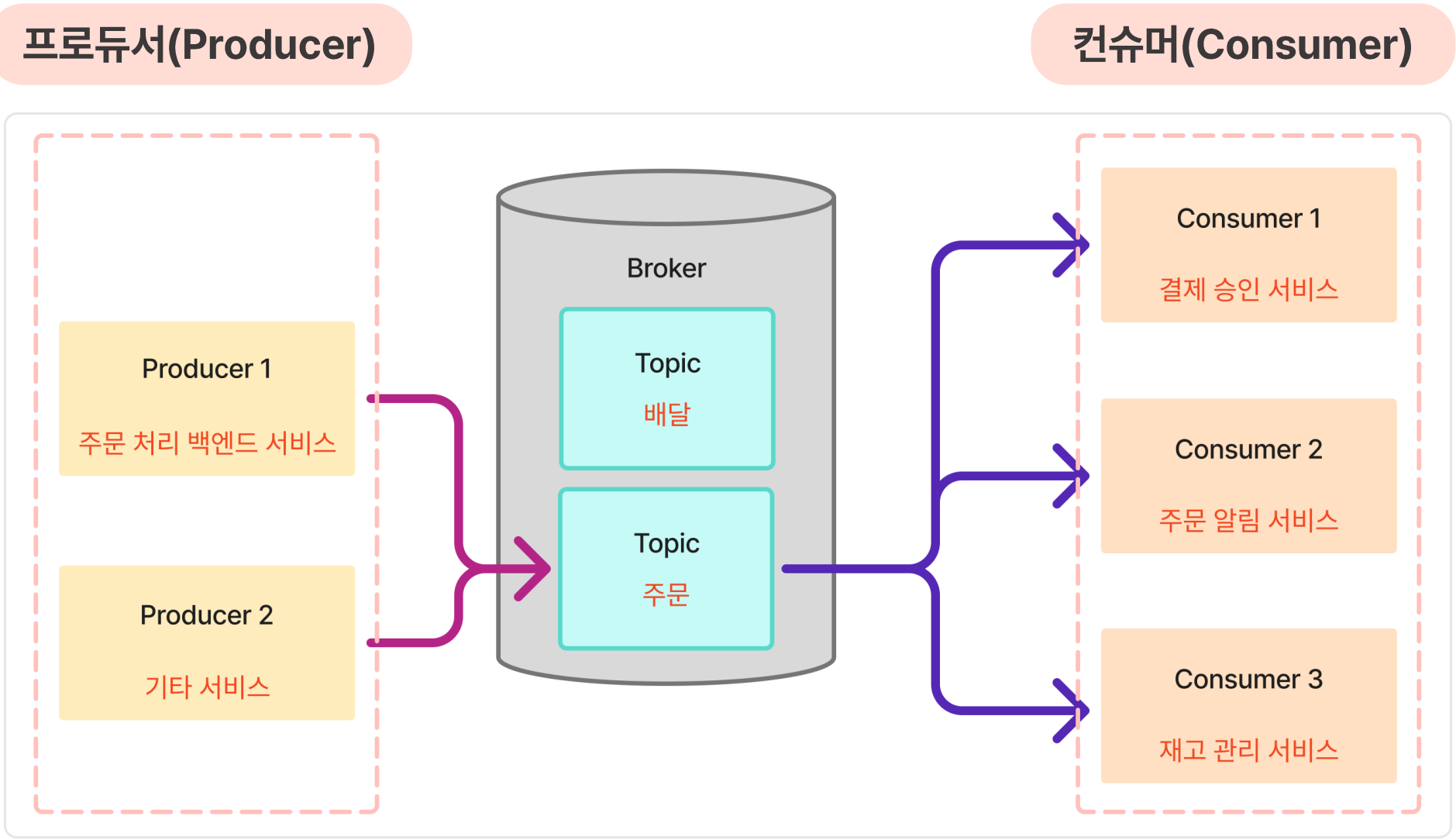
예 : 이커머스 주문 시스템


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)

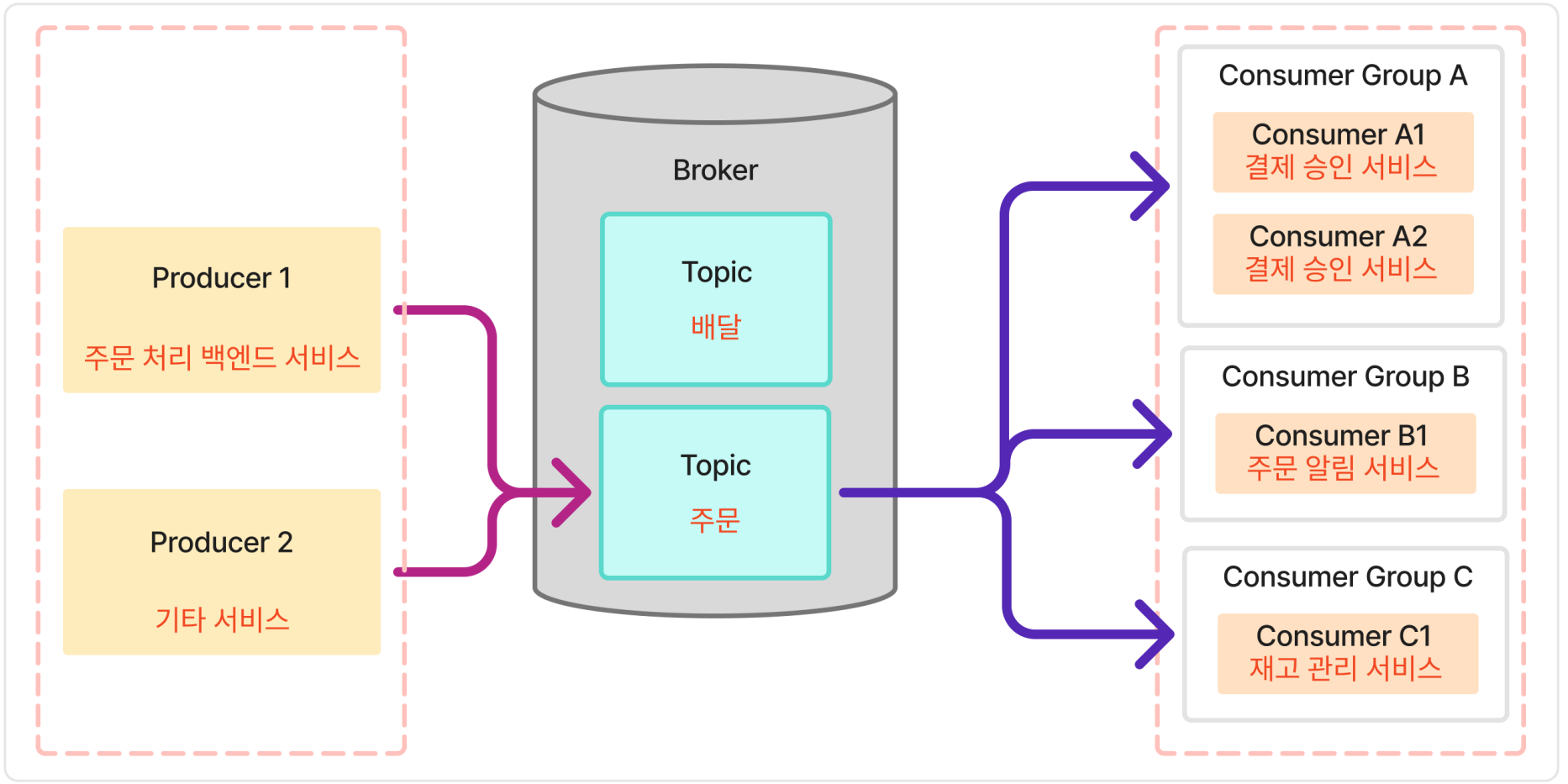


- Kafka에서 데이터를 저장하고 처리하는 기본 단위
- “무언가 발생했다”
- 이벤트(event), 레코드(record), 또는 메시지(message)


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

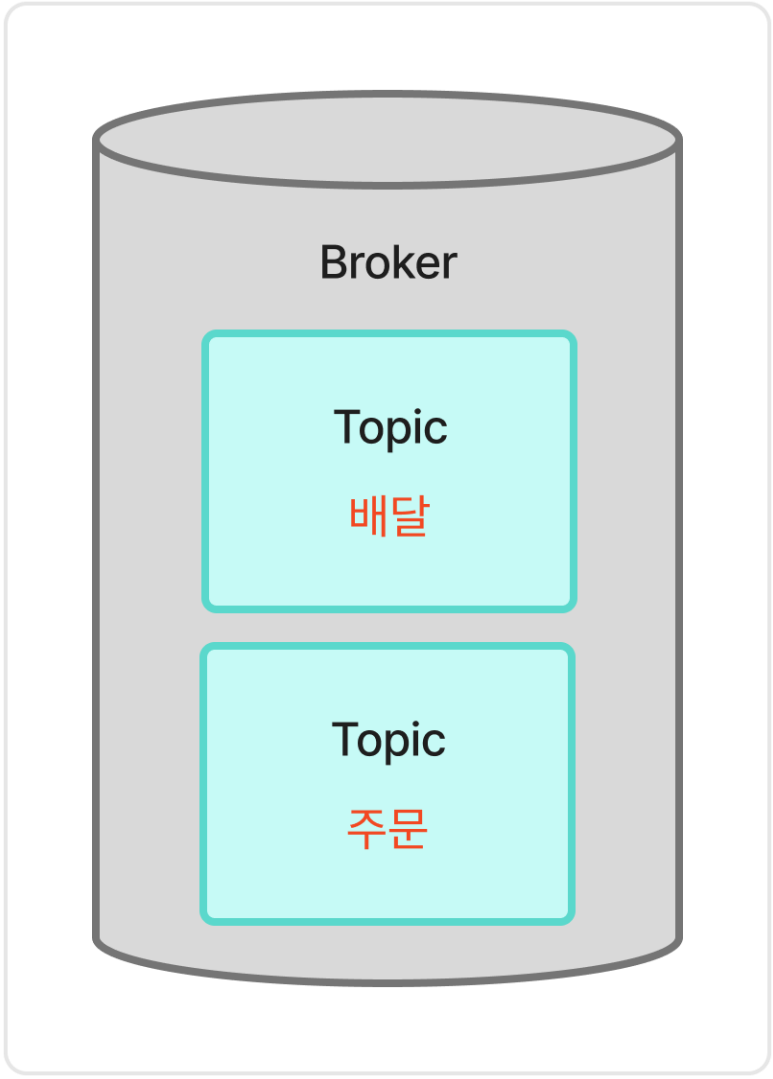
프로듀서(Producer)

컨슈머 그룹(Consumer Group)



토픽

- 이벤트가 저장되는 논리적인 단위

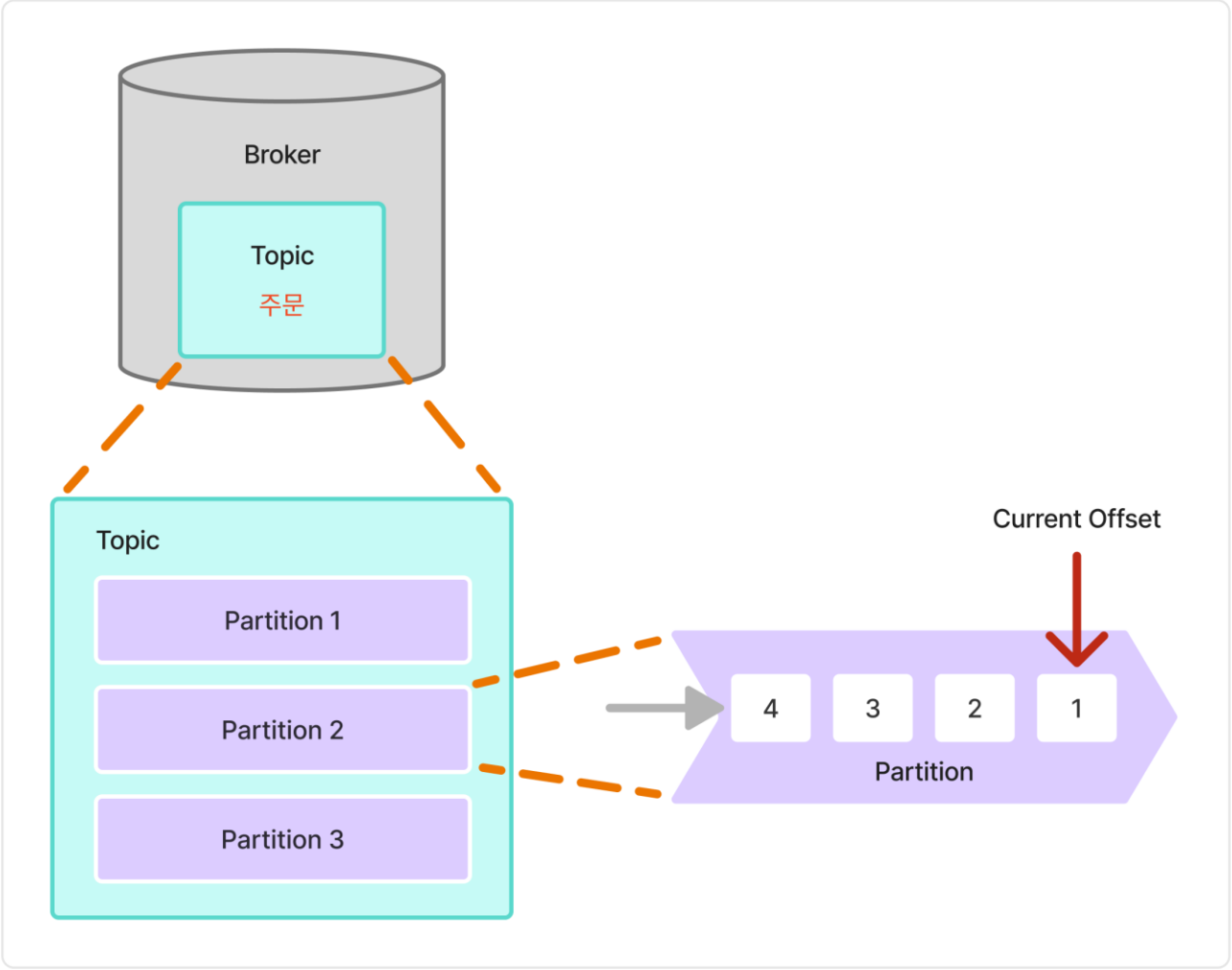


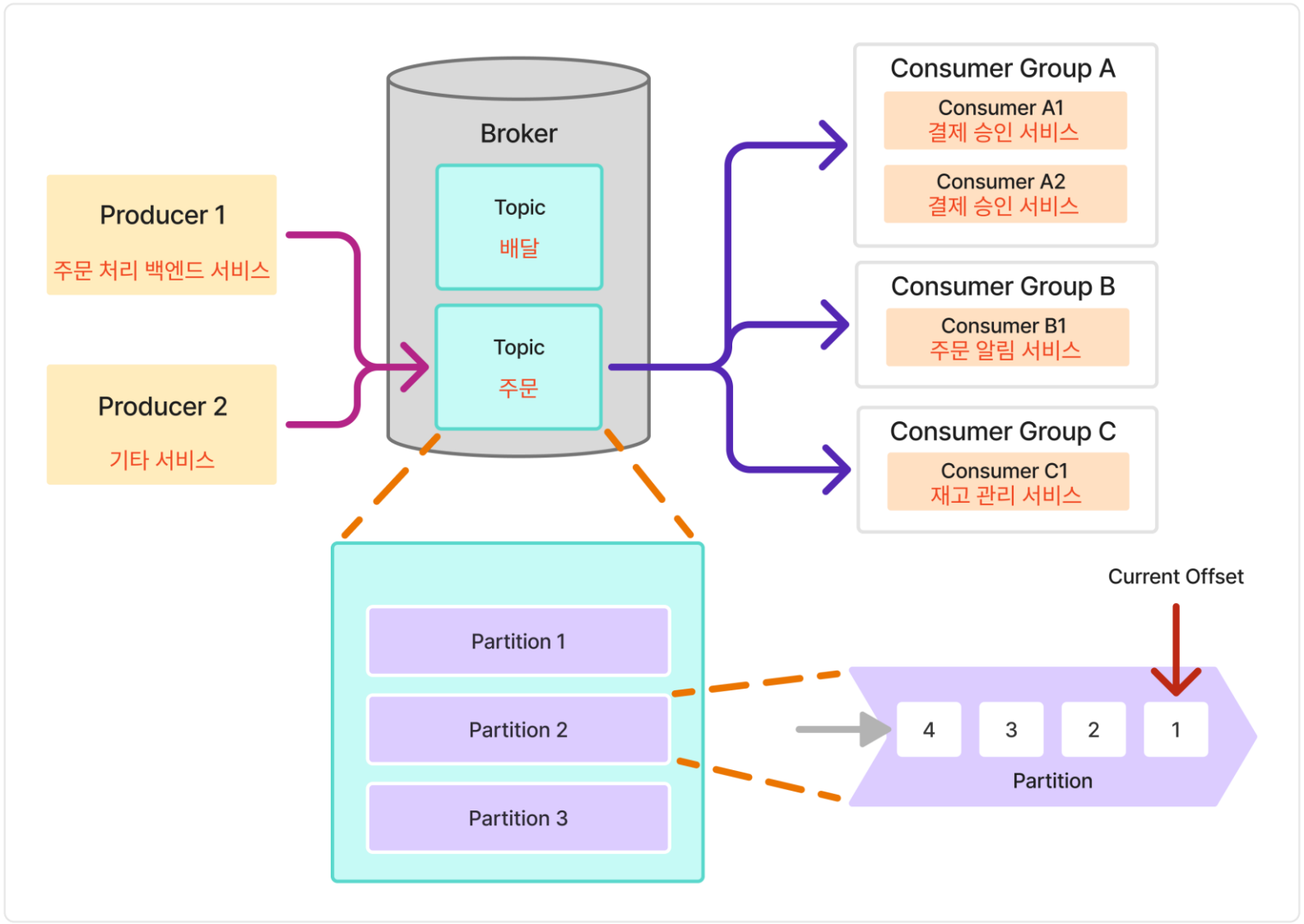
파티션

- 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

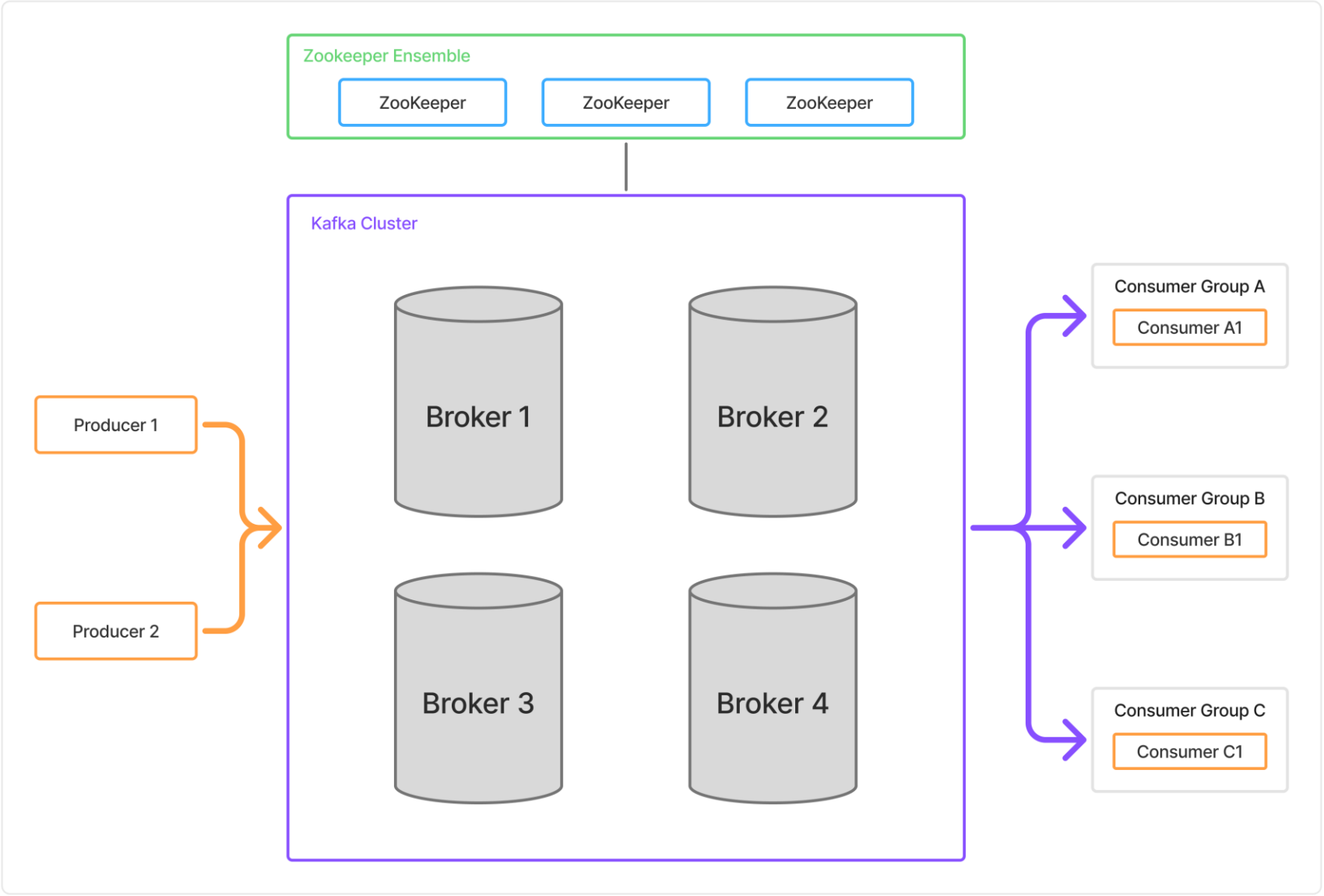
파티션에 분산 저장

Step 4

오프셋을 이용해
컨슈머가 이벤트 소비

Part 2

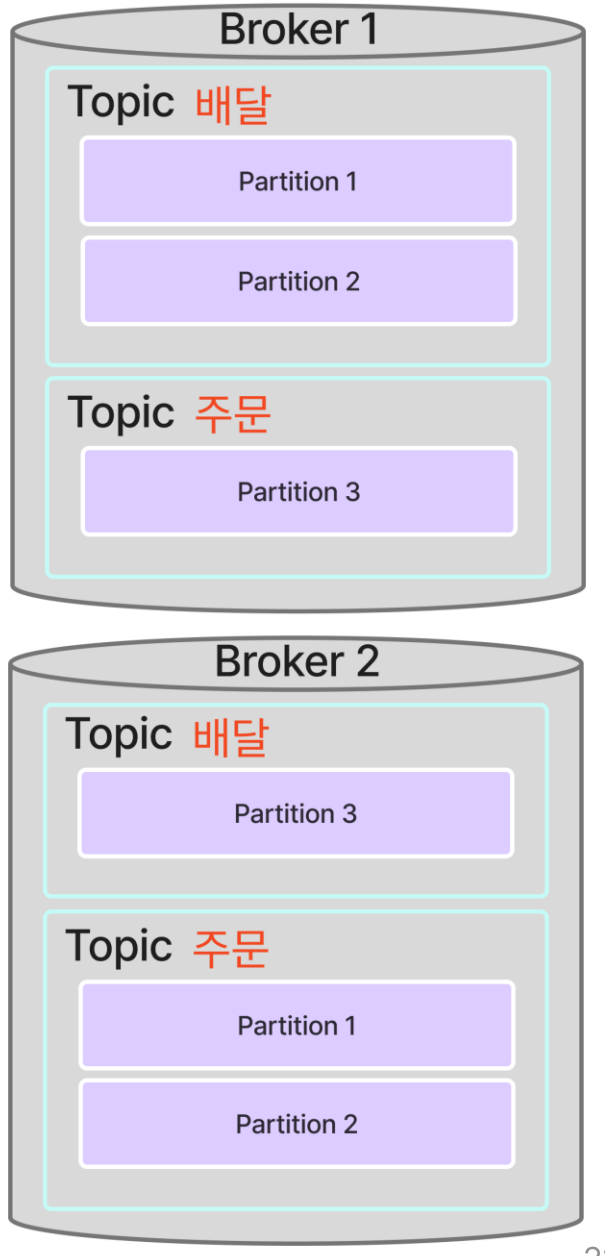
카프카의 핵심 요소 | 카프카 클러스터

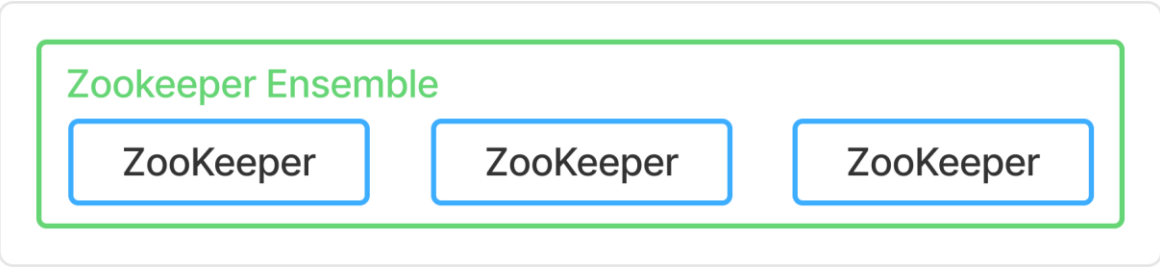


Part 2 **카프카의 핵심 요소 | 카프카 클러스터**

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

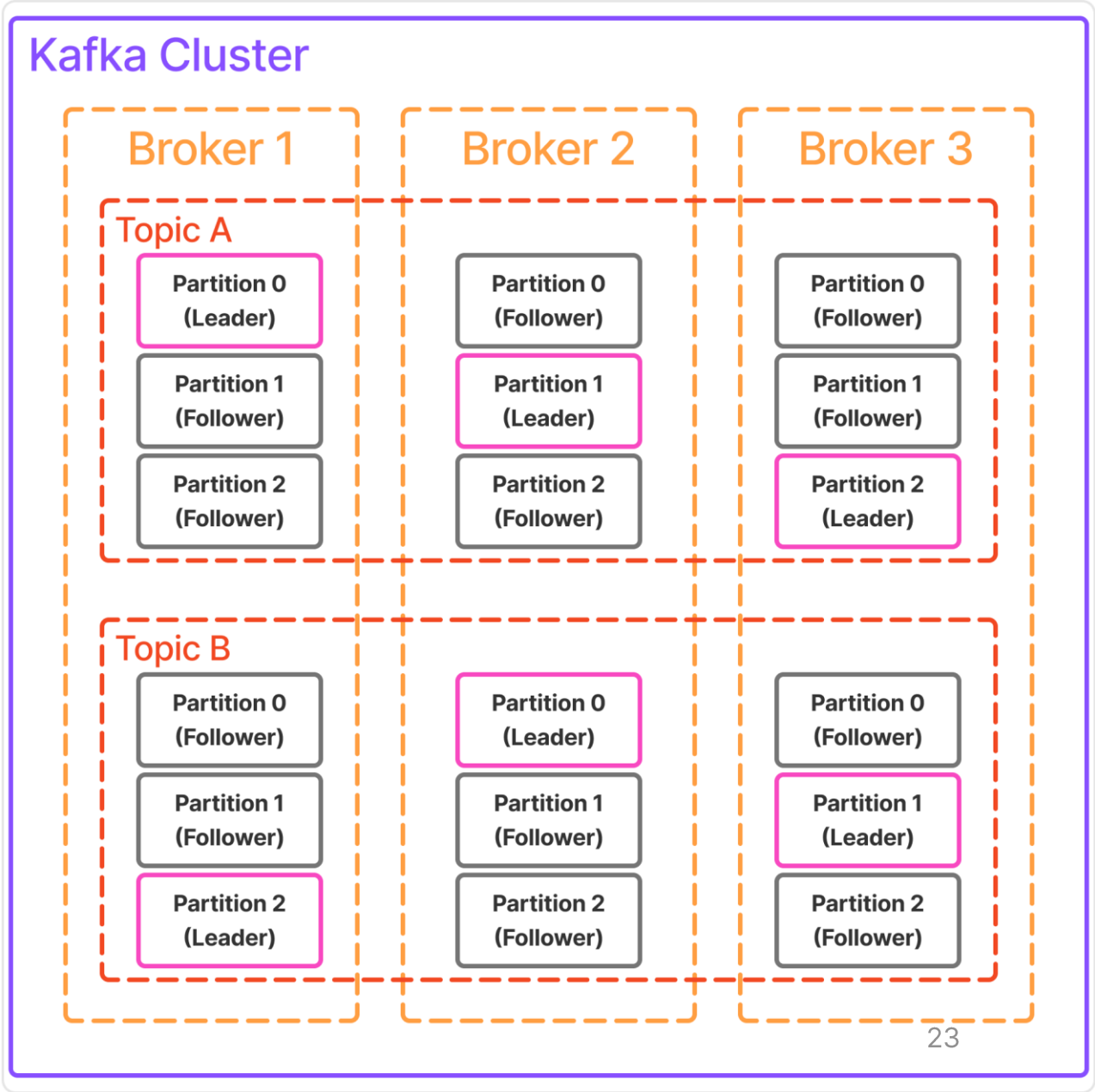
- 클러스터 내 브로커 관리
- 컨트롤러 브로커 선출
- 클러스터의 메타데이터 저장 및 관리
- Kafka 클러스터의 중앙 조정자 역할

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