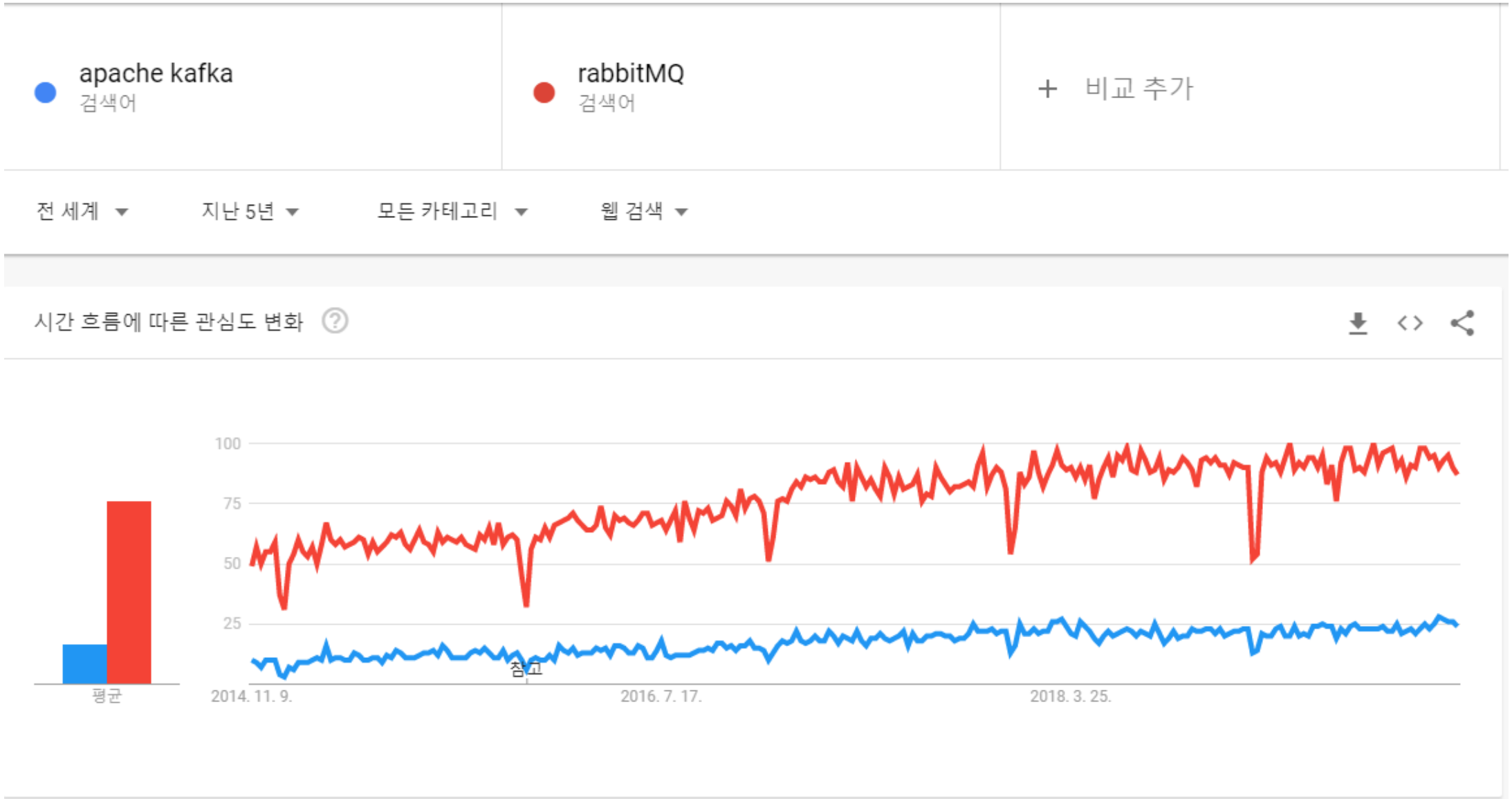
데이터 스트림

Apache kafka를 이용한 데이터 스트림

kafka는 무엇일까?

- 대용량 대규모 메시지 데이터를 빠르게 처리하기 위해 개발
- 메세징/스트리밍 플랫폼
- 링크드인 내부의 문제를 해결하고자 개발
- 2011년 초, 아파치 공식 오픈소스로 공개
- 빅데이터를 이용하는 기업이 늘면서 카프카에 많은 관심

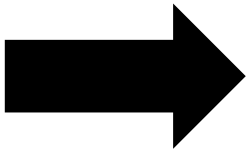


출처 : https://trends.google.co.kr/trends/explore?date=today%205-y&q=apache%20kafka,rabbitMQ

kafka는 어떻게 탄생하였나?

링크드인

- 비즈니스 인맥 소셜네트워크 서비스
- 회원 5억명 이상(2018년 기준)
- 실 사용자 1억 6000명 이상(2017년 4월 기준)
- 2016년 6월 13일 262억달러(약 31조원) 에 MS에 인수

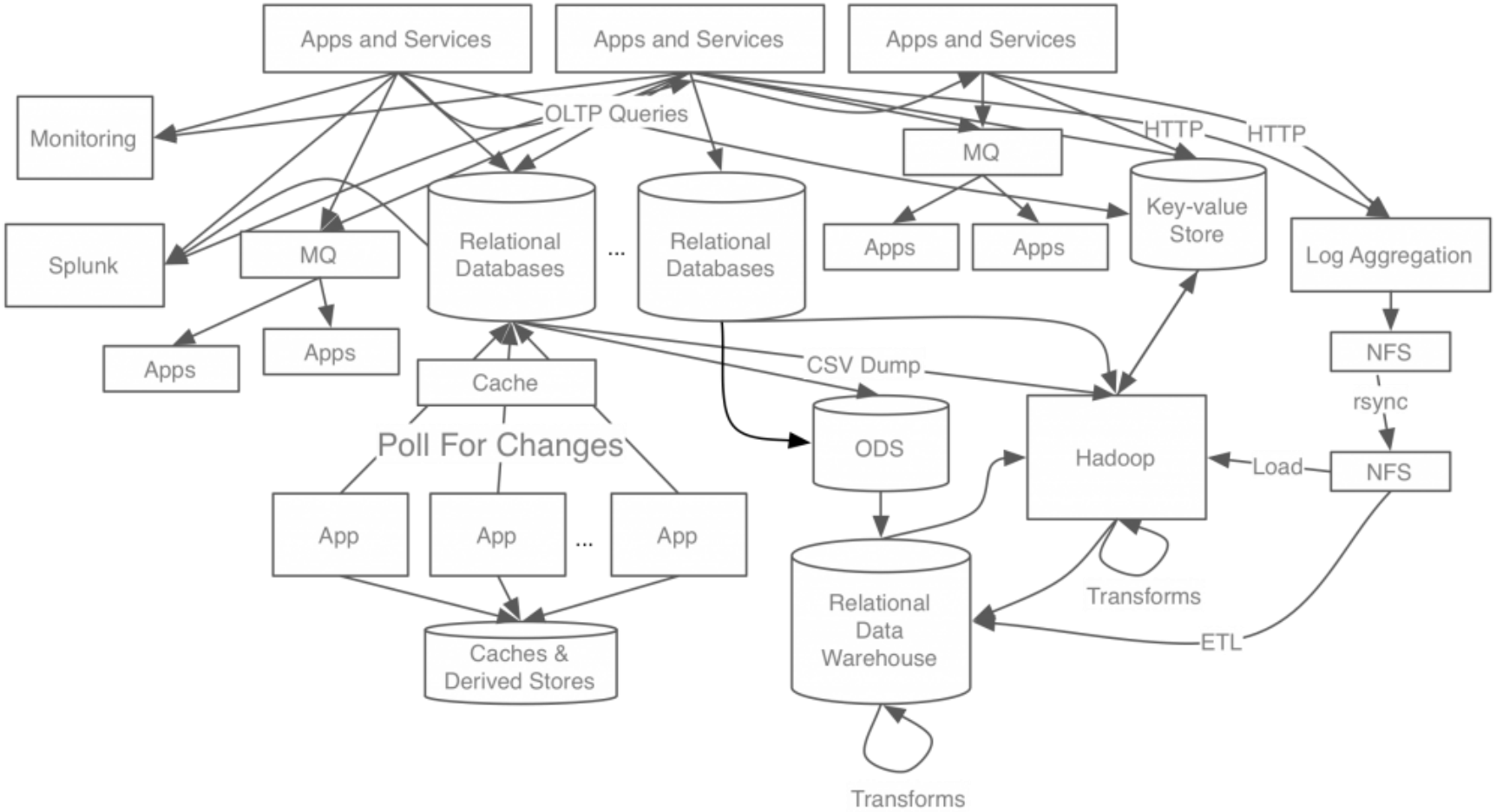


"이 정도 규모 서비스에서는
데이터 처리를 도대체 어떻게 할까?"

카프카는 어떻게 탄생하였나?

- 링크드인 데이터 플랫폼 요구사항
  ① Metric monitoring data system : 서비스에서 일어나는 미터링(사용량, 응답시간, 에러카운트) 저장용 시계열 데이터 처리 시스템
  ② Log monitoring data system : 앱/서비스에서 발생하는 로그를 저장하고 이것을 기반으로 stream/batch로 분석하도록 데이터를 저장하는 시스템
  ③ Service contents, Main data(user정보 등) system : 대부분의 앱에서 보낸 OLTP용 쿼리 (주로 데이터 갱신)를 실행
  ④ 추천/장바구니 같이 트렌젝션 처리는 필요 없으나, 실시간 처리가 요구되는 key/value 형태의 저장소
  ⑤ 서비스와 플랫폼 전체에서 발생되는 데이터를 모아서 일간/주간/월간/연간 데이터를 제공하는 데이터마켓 또는 이를 batch 분석하는 data warehouse
  ⑥ 빅데이터를 저장/처리하기 위한 대용량 분산 저장소(하둡) : 빅데이터를 처리해서 batch로 warehouse에 보냄

카프카는 어떻게 탄생하였나?



kafka 개발 이전 링크드인 데이터 처리 시스템

출처 : https://www.confluent.io/blog/event-streaming-platform-1

카프카는 어떻게 탄생하였나?

- end to end arcitecture의 문제 – 복잡도의 증가
  - 실시간 OLTP 처리와 비동기 처리가 동시에 이루어짐
  - 하지만 통합된 전송 영역이 없음
  - 문제 발생 시, 여러 데이터 시스템을 다 뒤져야 함
  - 트러블 슈팅, 업데이트, 하드웨어 증설 등의 작업 시에도 많은 시간 소요

카프카는 어떻게 탄생하였나?

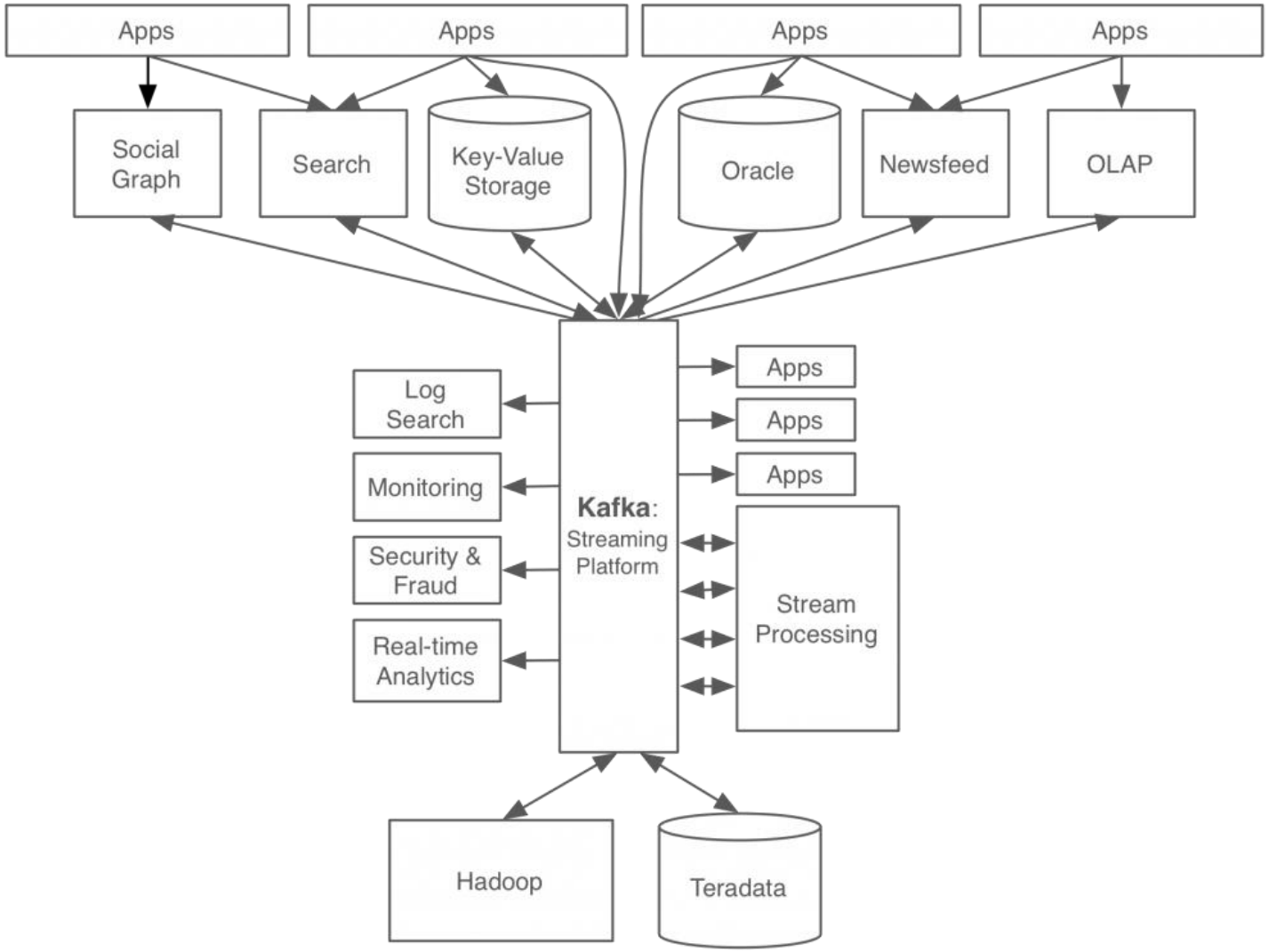
- end to end arcitecture의 문제 – 데이터 파이프라인 관리 어려움
  - 수 많은 데이터 시스템을 담당하는 각 팀마다 입맛대로 다른 파이프라인
  - 동일한 데이터도 각각 다른 파이프라인
  - 통합 데이터 분석을 하려면 서로의 파이프라인을 필연적으로 연결해야 함
  - 통합도 어렵고, 확장도 어렵고, 운영도 어렵다
  - 복잡도로 서로 다른 두 시스템 간의 데이터가 서로 달라져서 신뢰도 문제 발생

kafka의 탄생

- 데이터 시스템의 복잡도와 파편화 때문에 링크드인에 문제
- 제이 크랩스, 니하 나케티, 준 라오 등의 팀 구성해 새 시스템 개발
- 링크드인의 새 데이터 처리 시스템 목표
  - 프로듀서와 컨슈머의 분리
  - 메시징 시스템과 같이 영구 메시지 데이터를 여러 컨슈머에게 허용
  - 높은 처리량을 위한 메시지 최적화
  - 데이터가 증가함에 따라 스케일아웃이 가능한 시스템

※ 스케일 업 : 리소스의 수요가 증가하여 하드웨어의 크기를 키우는 것
※ 스케일 아웃 : 리소스의 수요가 증가하여 하드웨어의 개수를 늘리는 것

kafka의 탄생


Apps
Apps
Apps
Apps
Social Graph
Search
Key-Value Storage
Oracle
Newsfeed
OLAP
Log Search
Monitoring
Security & Fraud
Real-time Analytics
Kafka: Streaming Platform
Apps
Apps
Apps
Stream Processing
Hadoop
Teradata

kafka의 탄생

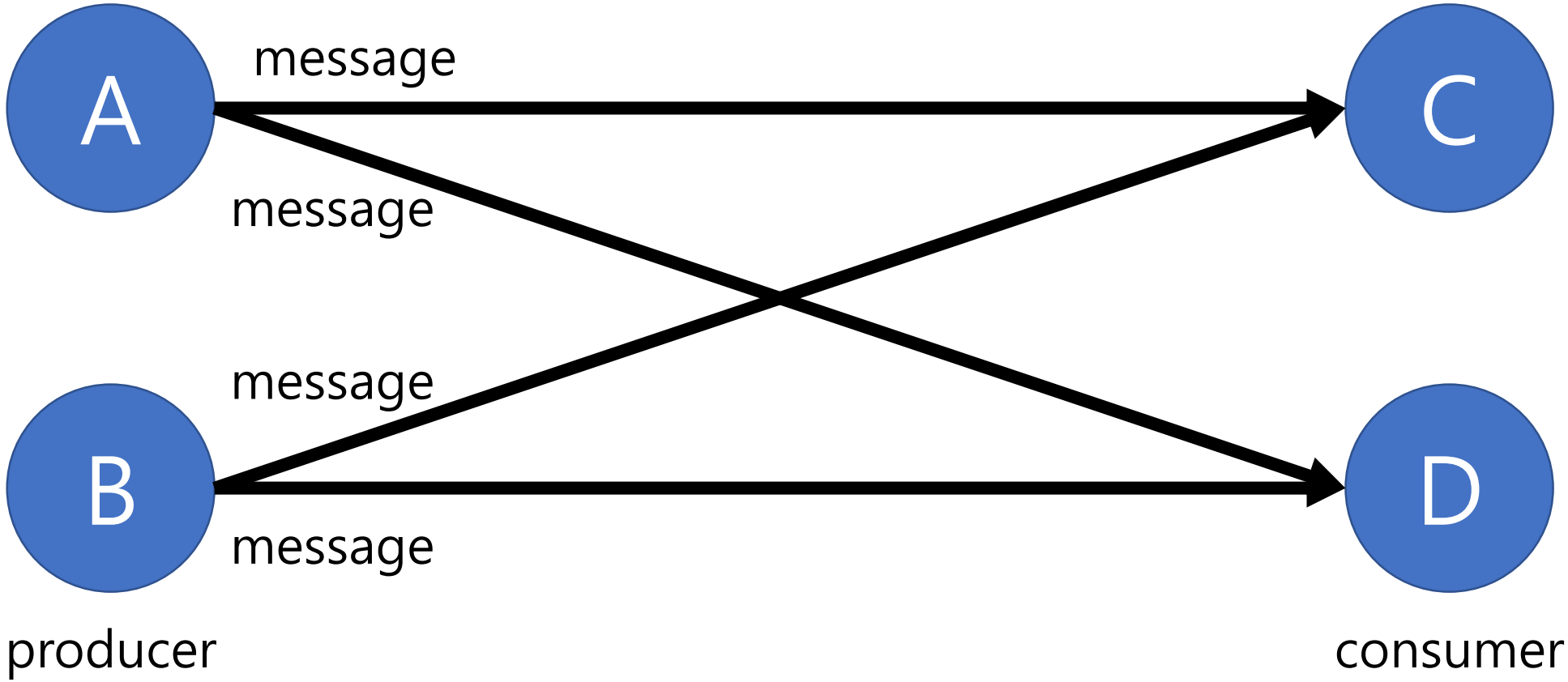
- 링크드인이 kafka 적용 후
  - 전사 데이터 파이프라인 -> 모든 데이터스토어와 데이터/이벤트 연결
  - 새로운 데이터스토어 추가되더라도 kafka 표준포멧으로 부담없이 연결
  - 이전에 불가능 했던 다양한 분석(보안, 실시간분석, 스트림처리) 가능
    -> 높은 신뢰도의 데이터 분석/실시간 분석 가능하여 서비스 품질 상승
  - 여러 데이터서비스와의 데이터 전달 포멧, 파이프라인 등을 별도로 고려하거나 개발하지 않아도 되므로 본인의 메인 업무에만 집중가능

링크드인의 kafka이용한 데이터 처리 시나리오

하루에 1조개의 메세지 처리

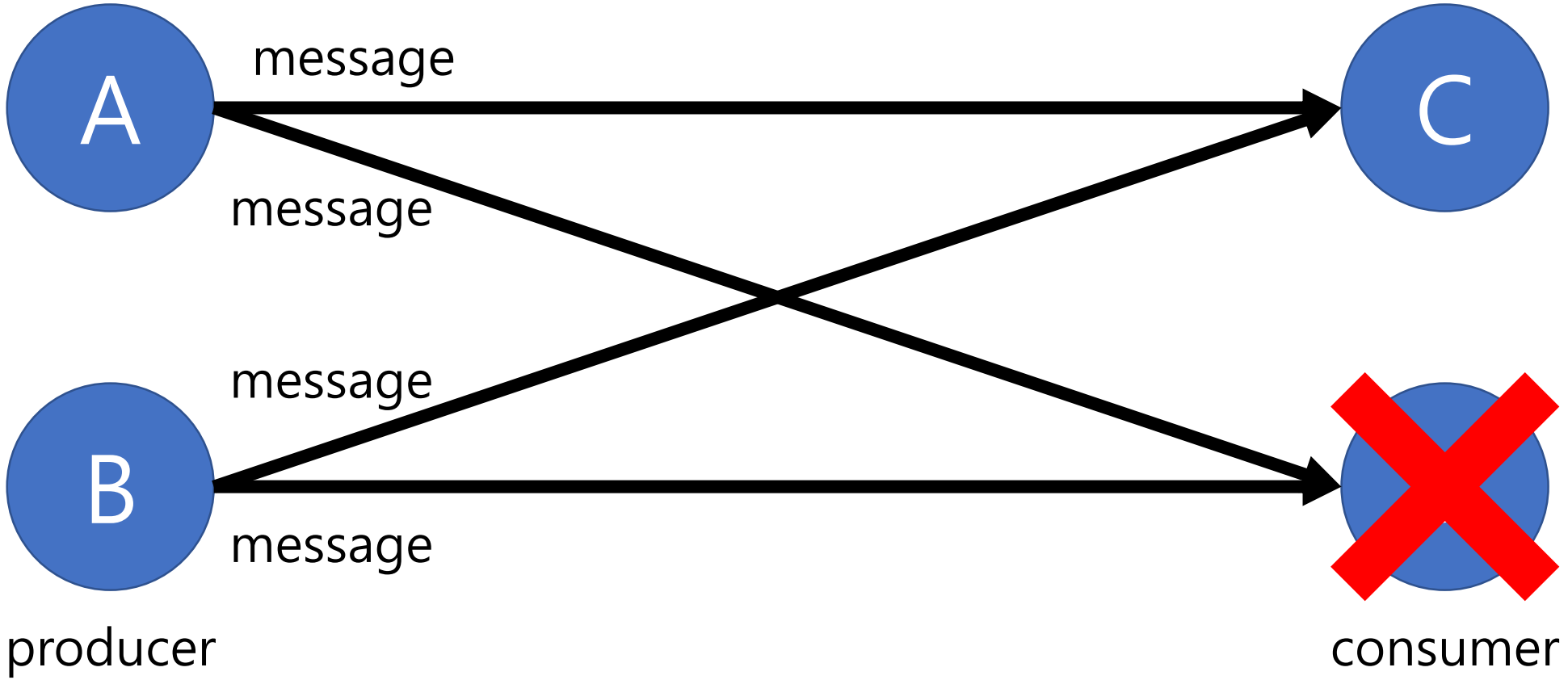
kafka 동작방식과 원리

- 일반적인 형태의 네트워크 구조



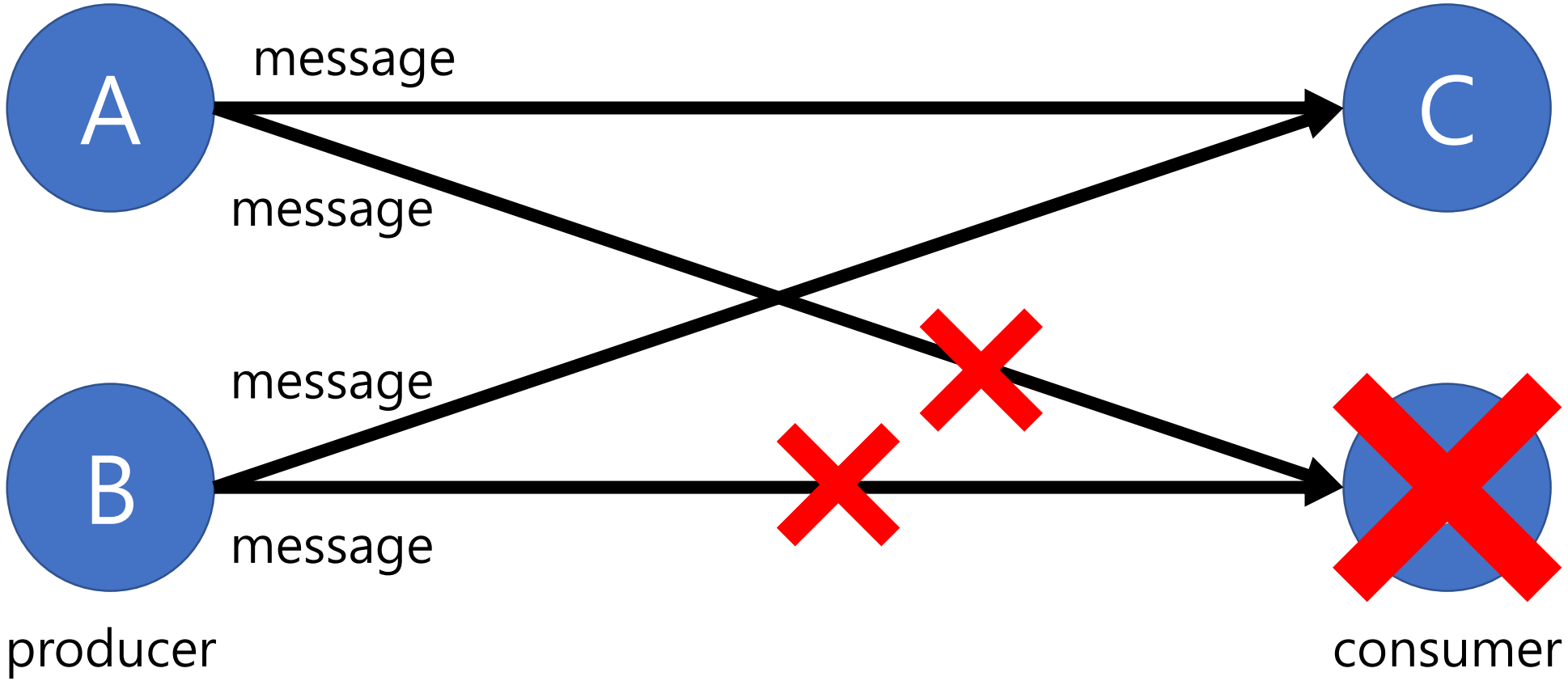
kafka 동작방식과 원리

- 일반적인 형태의 네트워크 구조



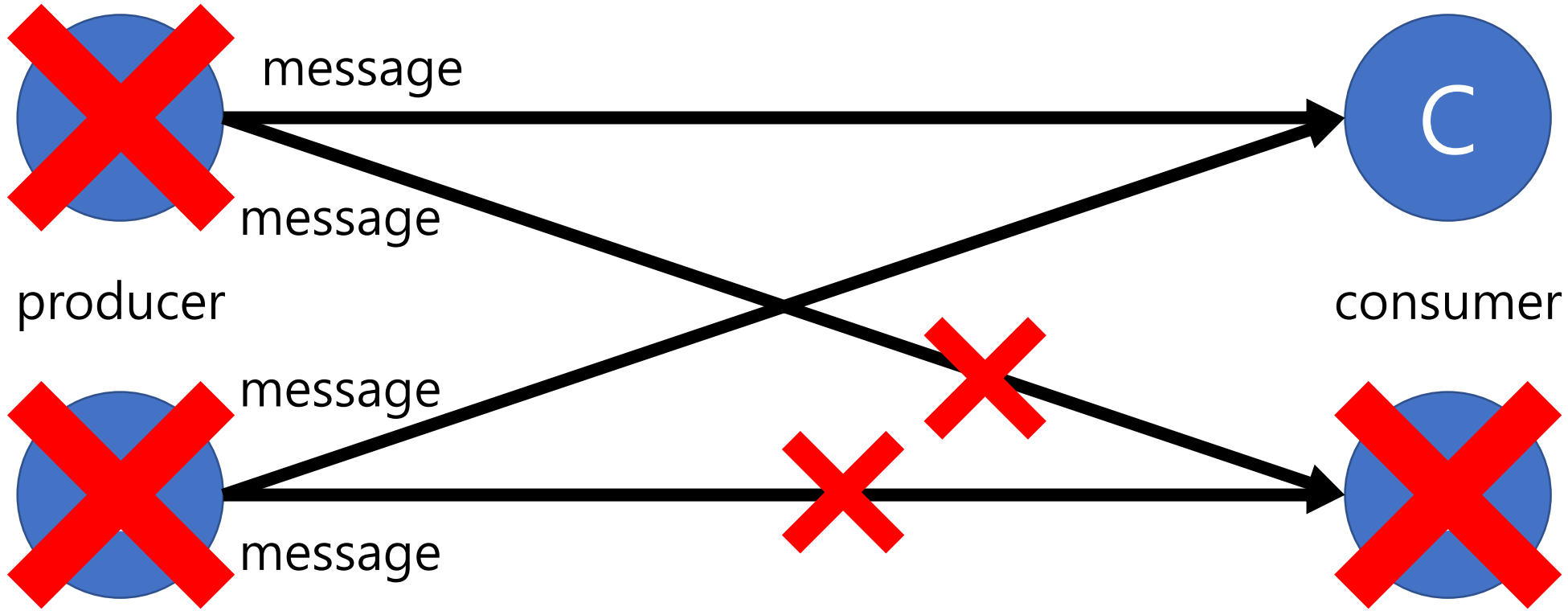
kafka 동작방식과 원리

- 일반적인 형태의 네트워크 구조



kafka 동작방식과 원리

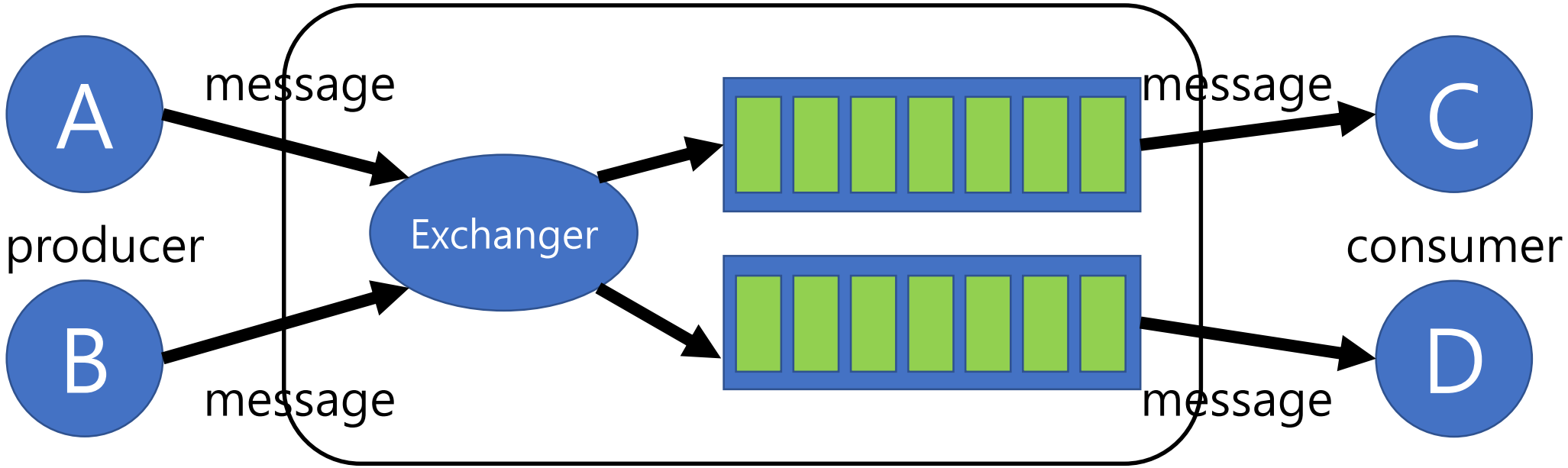
- 일반적인 형태의 네트워크 구조



메시지 대기와 장애처리를 하지 않으면 장애가 전파되어버린다.

kafka 동작방식과 원리

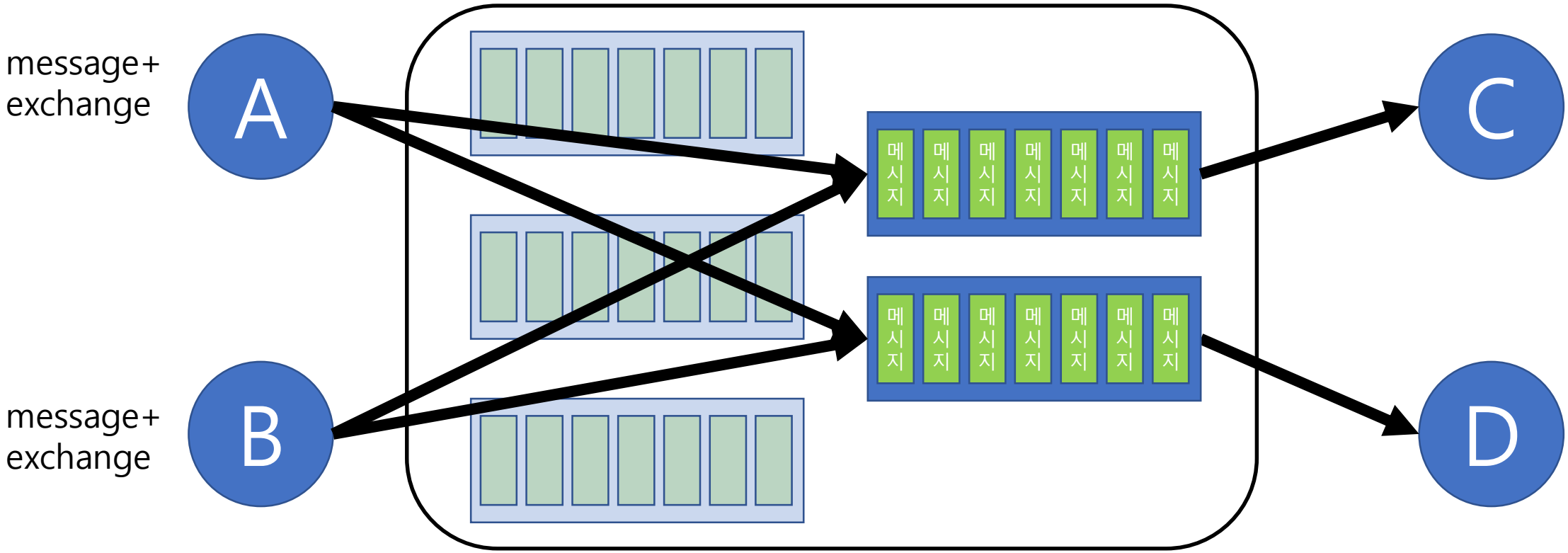
- pub-sub 형태의 네트워크 구조



- 큐 관리, 메시지 정합성, 전달 결과를 정확히 관리하기 위해 내부 프로세스가 복잡
- 메시지 보관, 교환, 전달 과정의 신뢰성을 보장하는 것이 목적이므로 속도/용량 떨어짐

kafka 동작방식과 원리

- 카프카 아키텍처

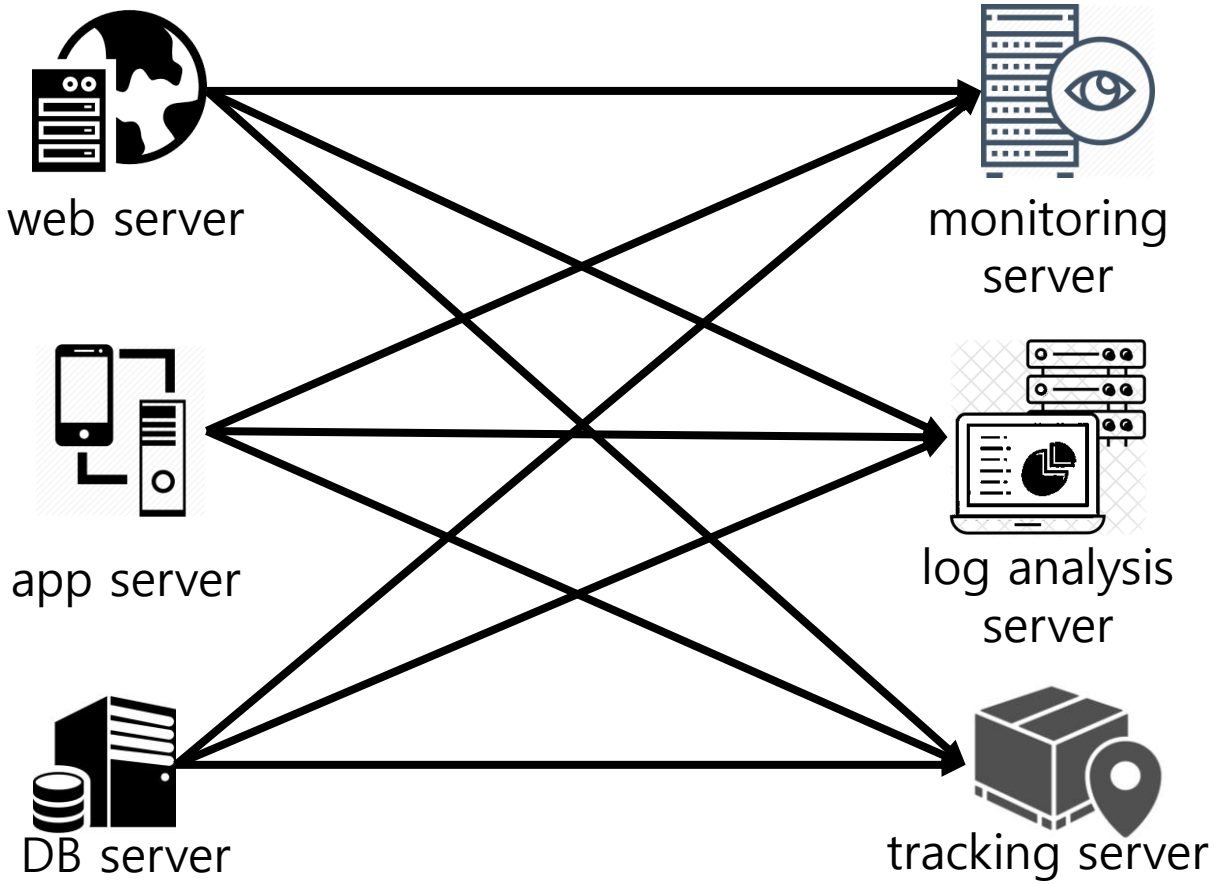


kafka의 특징

- producer와 consumer의 분리
- multi-producer / multi-consumer
- disk에 메시지 저장
- 확장성
- 높은 성능

producer와 consumer의 분리

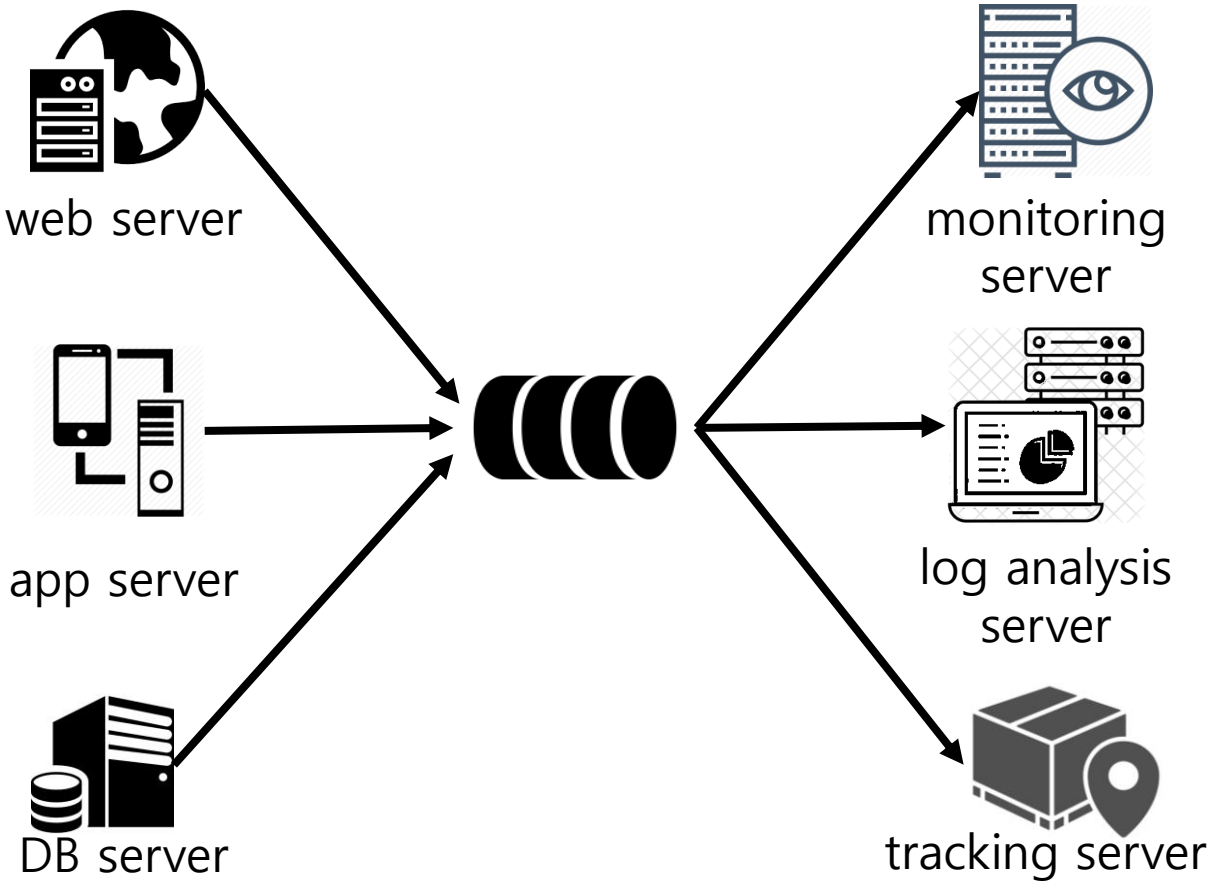
- 풀링 방식의 분석시스템



- 서버가 추가될 때마다 연결할 시스템이 많기 때문에 필연적으로 추가적인 작업이 필요
- 모니티링 서버에 문제가 발생 시, 연결된 서버들 또한 지연이슈가 발생

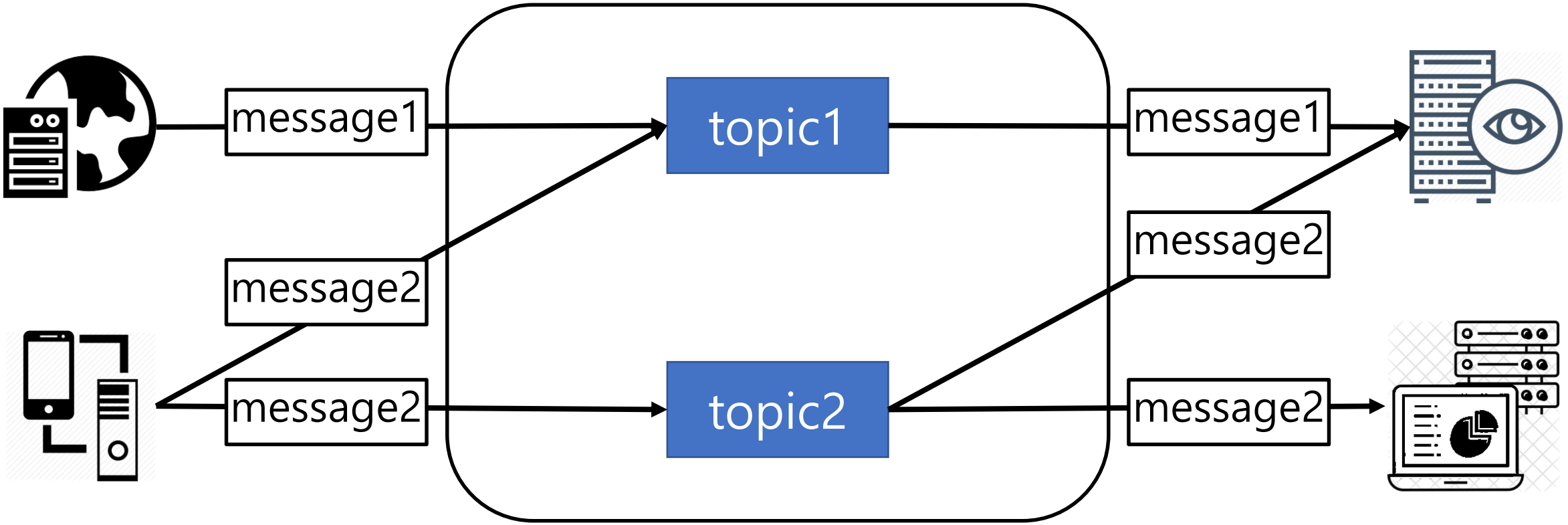
producer와 consumer의 분리

- kafka를 이용한 pub-sub 방식

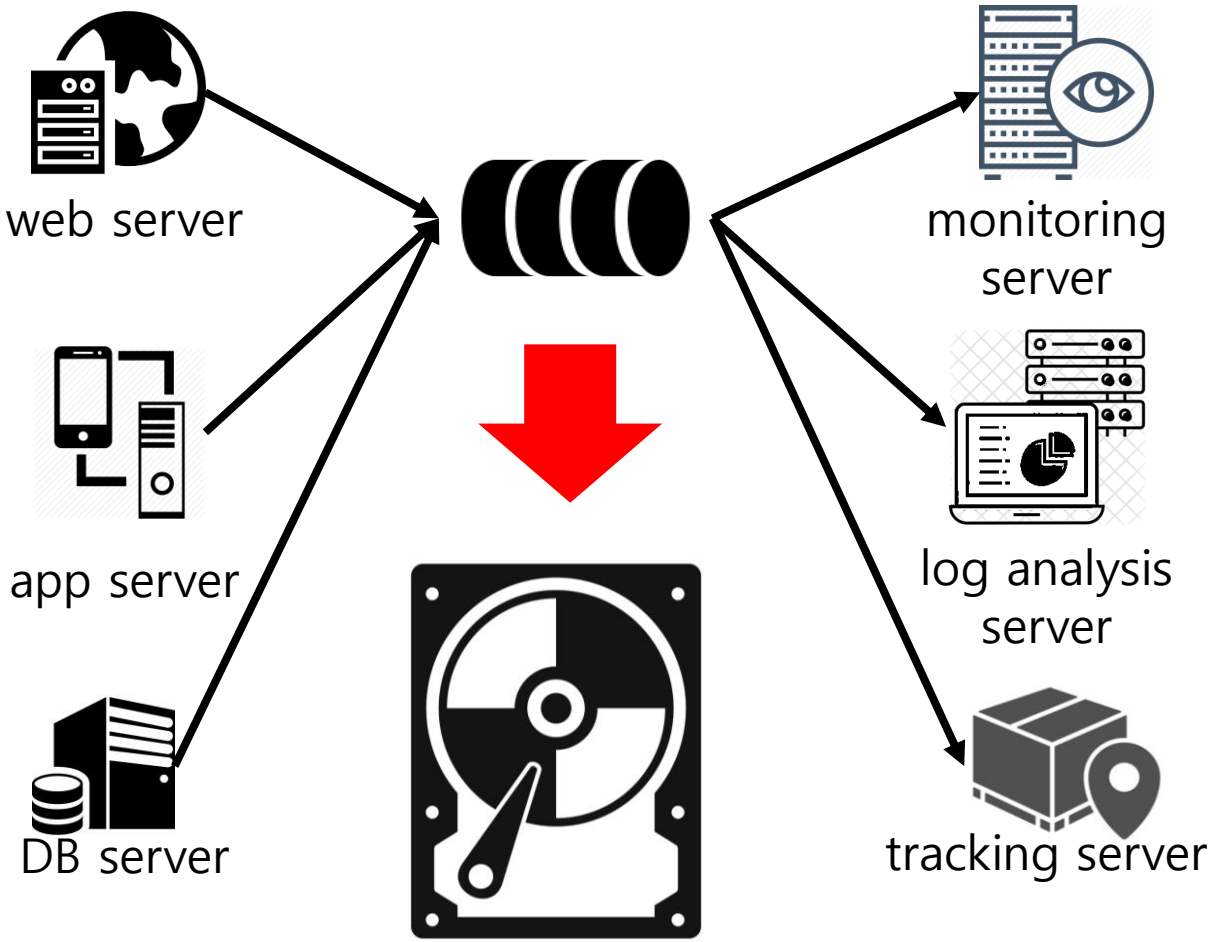


- producer와 consumer를 분리하여 서로의 상태 의존도가 떨어짐
- 서버가 추가되더라도 메시지를 카프카로만 전송하면 되므로 추가작업에 대한 부담이 줄어듬

multi-producer / multi-consumer



디스크에 메시지 저장



- multi-consumer 구현 가능
- 버그 발생 시, consumer 중간 후, 재실행 가능
- 메시지 유실없이 작업이 가능

확장성

- 카프카 클러스터는 3대의 브로커로 시작해서 수십대까지 확장 가능
- 무중단 온라인 상태로 확장가능
- 트레픽 및 사용량 증가 시, 리소스 범위에서 elastic하게 확장이 가능

높은 성능

- 높은 성능을 목적으로 pub-sub 모델을 개선한 시스템
- 내부적으로 분산처리, 배치처리 등 다양한 기법 사용
- 링크드인 사례 : 하루에 1조개 메시지 처리, 1페타바이트 처리

Kafka의 확장과 발전

- ESB(Enterprise Service Bus) 구현
- 실시간 처리 및 분석을 위한 데이터 파이프라인
- 블록체인 개발에 활용

• ESB의 특징

ESB(Enterprise Service Bus) 구현

고성능 pub-sub 모델

ESB
쉽게 구현

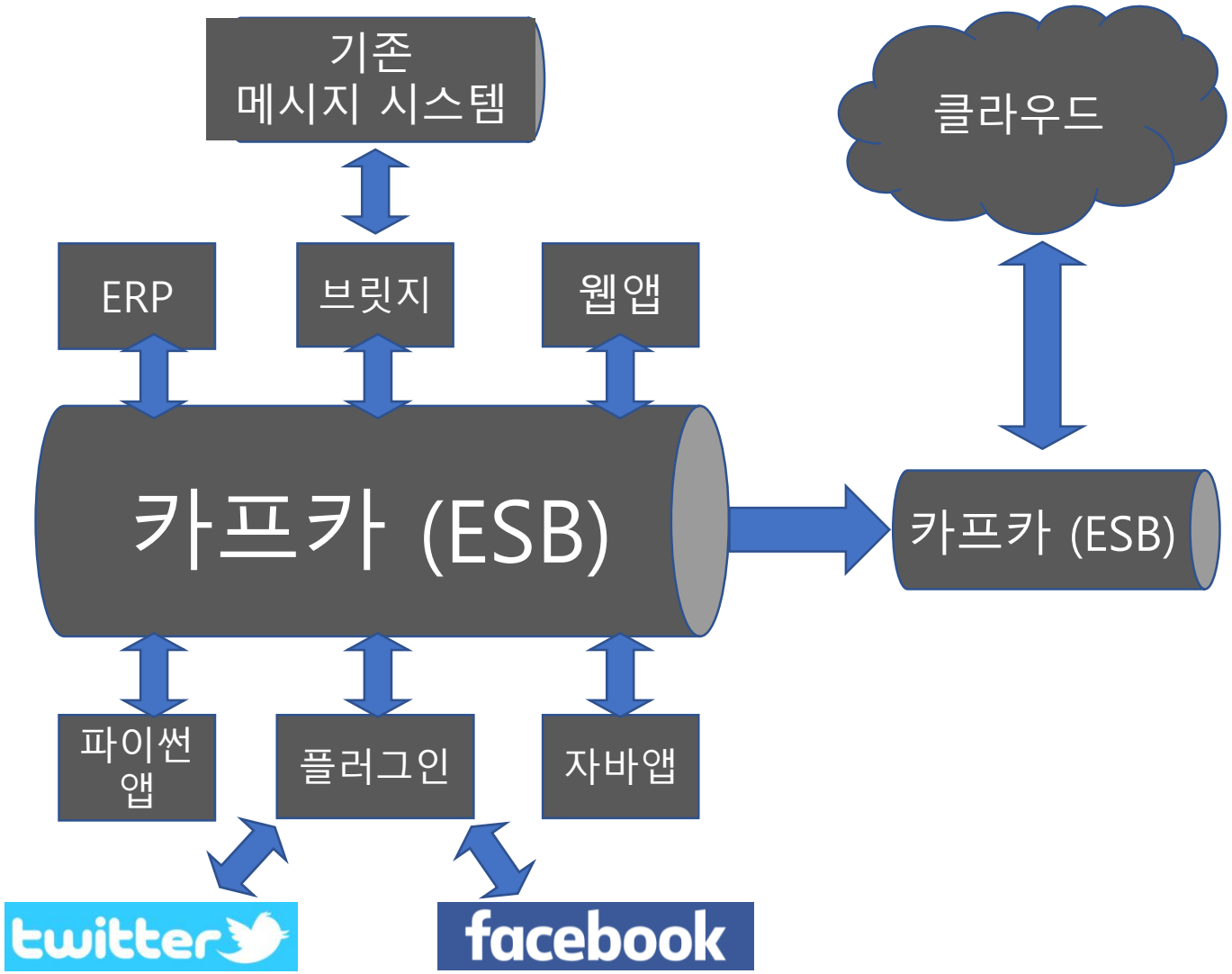
SOA(service oriented architecture)의 핵심구성요소

다양한 시스템과 연동하기 위한
멀티 프로토콜과 데이터 타입 지원

Loosed coupled을 위한 메시지큐 지원

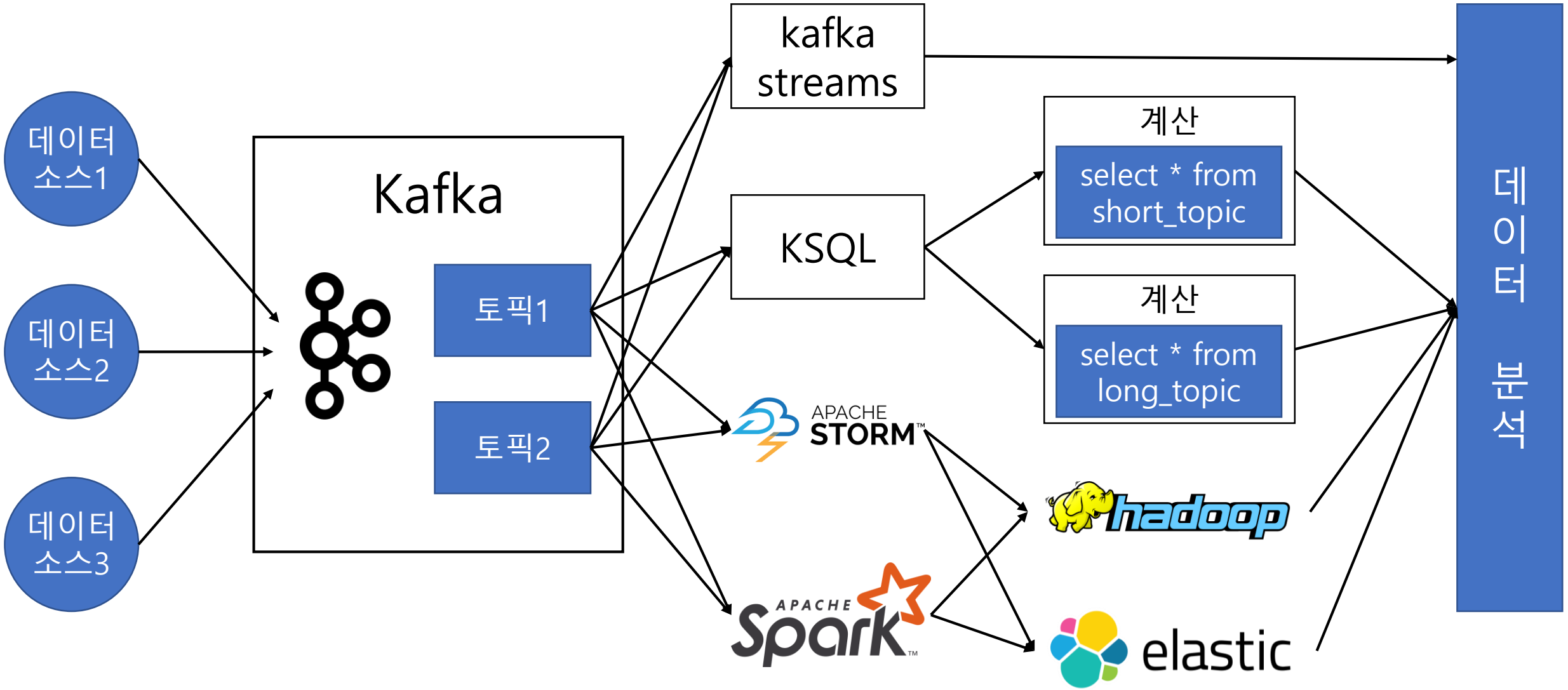
정기적으로 데이터를 가져오는 대신
event-driven 통신 지향

ESB로 활용한 메세지 시스템으로의 kafka

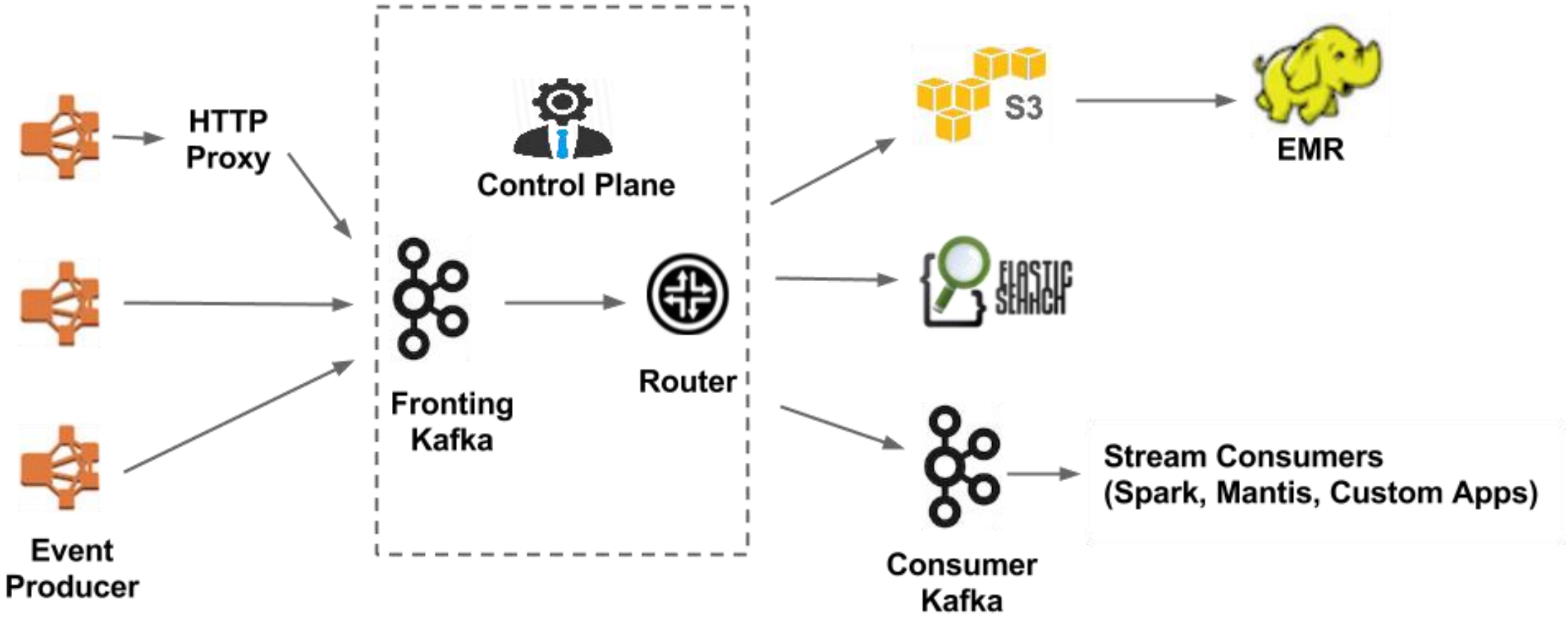


- 기존 메시지 시스템 브릿지로 연결
- 플러그인으로 외부 데이터 또한 가져와서 처리 가능
- 외부 데이터센터의 로깅/미터링/이벤트데이터 연결(퍼블릭클라우드)
- 빅데이터 분석과 머신러닝 플랫폼 만드는데 중요한 요소로 자리잡음

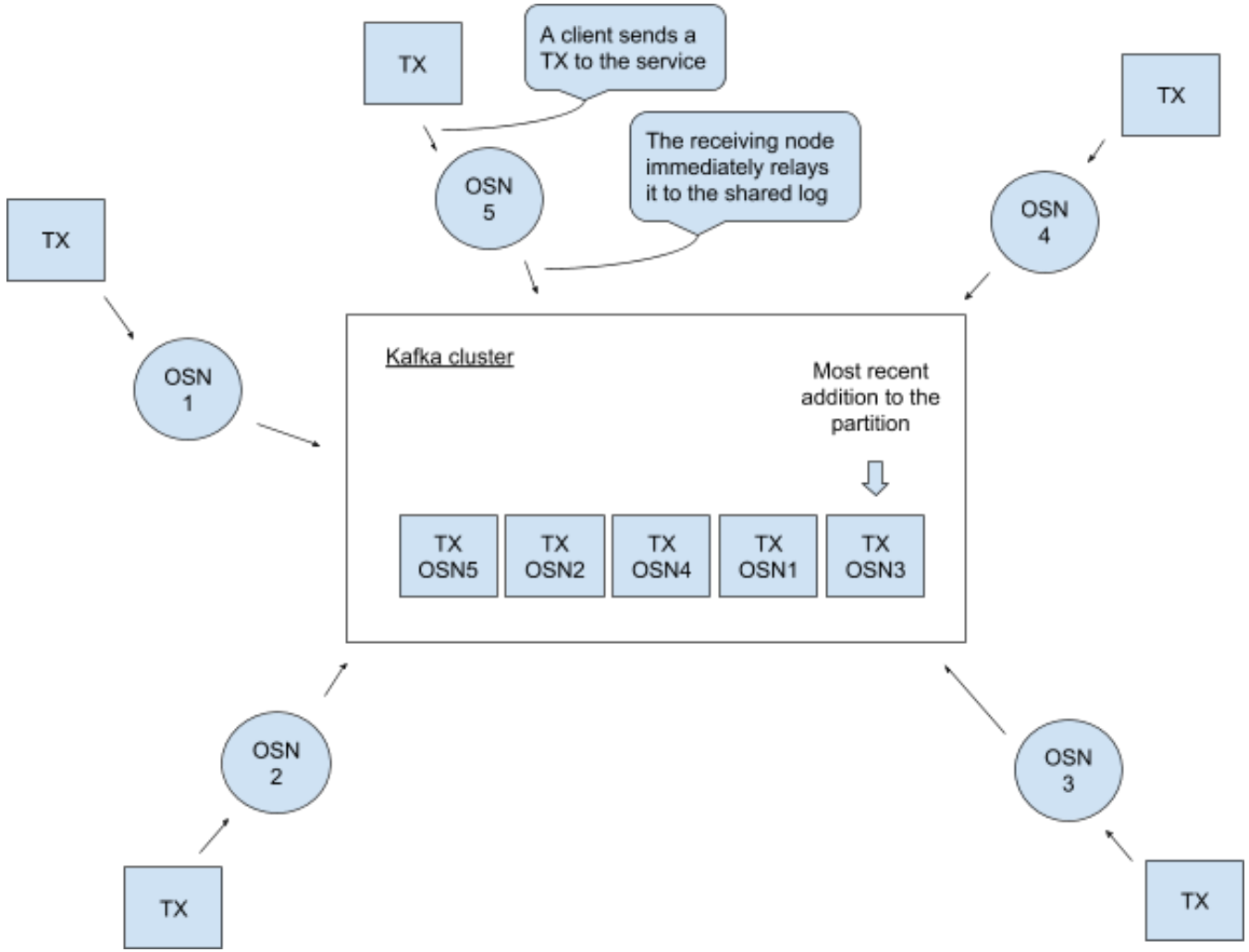
실시간 처리 및 분석을 위한 데이터 파이프라인



데이터 파이프라인 – Netflix 사례



블록체인 – Hyperledger Fabric



kafka의 구성요소