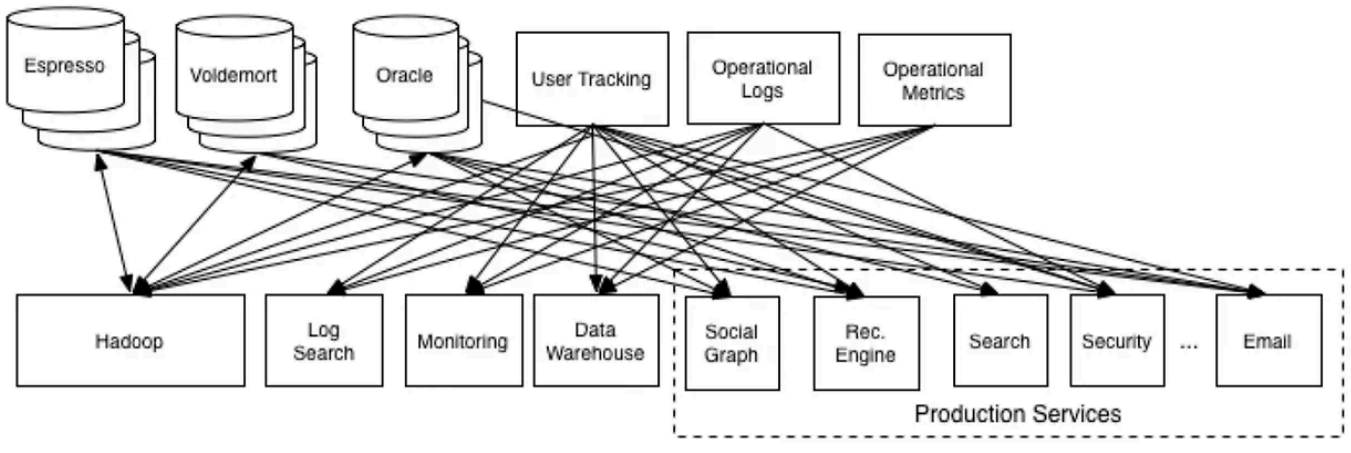
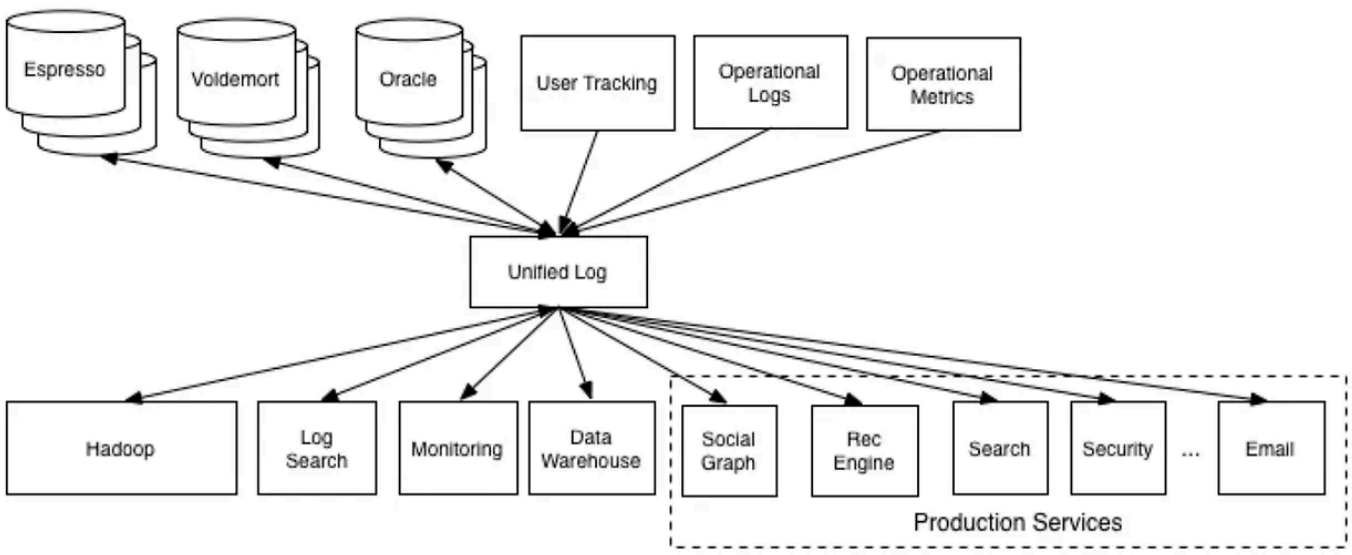
1장

링크드인 : 카프카 도입 이전 아키텍처



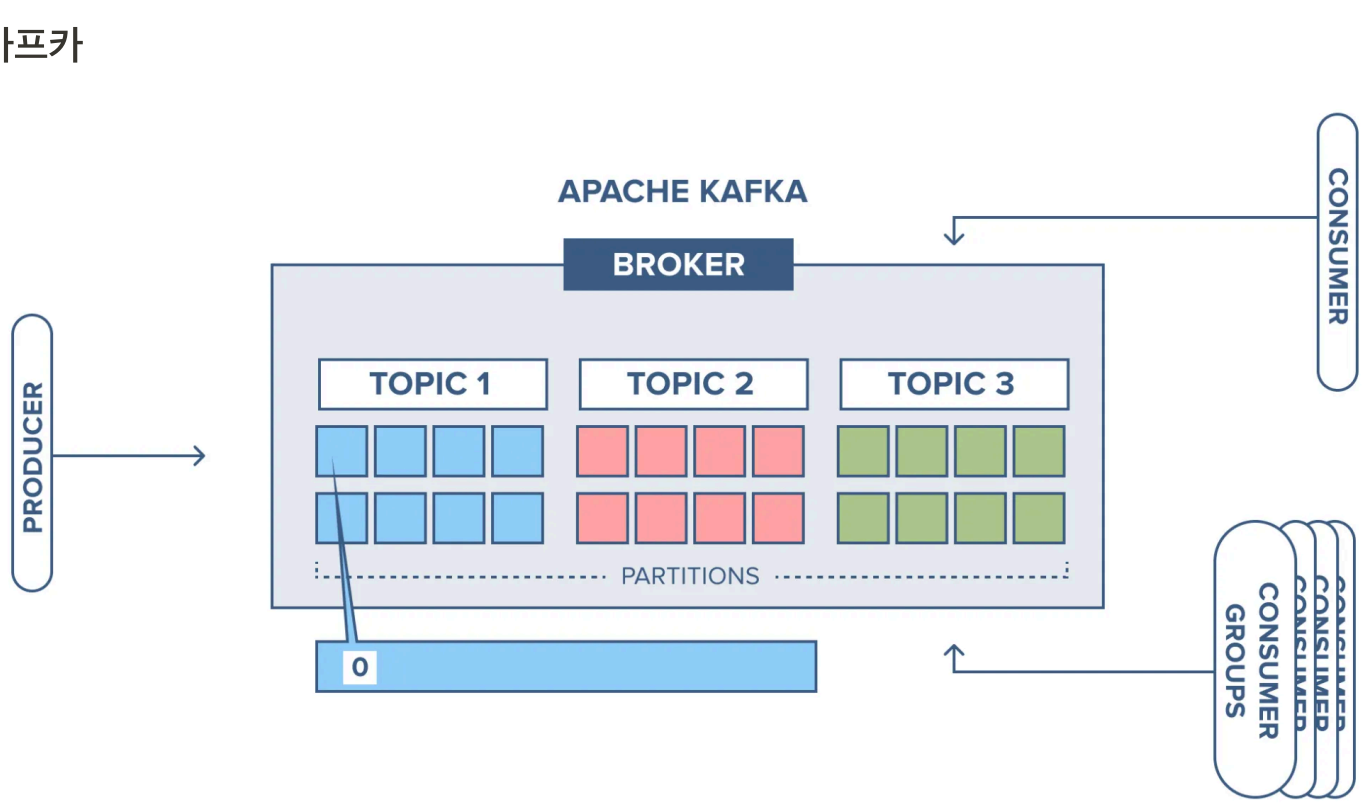
- 링크드인에서 파편화된 데이터 수집 및 분배 아키텍처를 운영하는 데에 어려움을 겪음 : 소스 애플리케이션과 타킷 애플리케이션이 단방향으로 직접 연결된 형태 → 신규 시스템 개발 : 카프카

링크드인 : 카프카 도입 이후 아키텍처



- 각각의 애플리케이션끼리 연결하여 데이터를 처리하는 것이 아니라 한 곳에 모아 처리할 수 있도록 중앙 집중화
- 소스 애플리케이션과 타깃 애플리케이션 사이의 의존도를 최소화하여 커플링을 완화한 형태

카프카



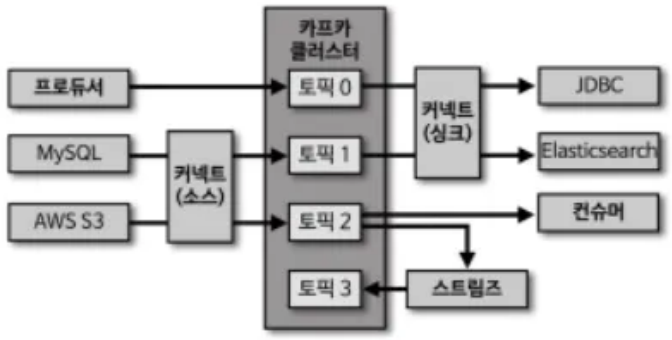
- 카프카는 직렬화 및 역직렬화를 통해 ByteArray로 통신 → 전달할 수 있는 데이터 포맷 제한 x
- 파티션 : 데이터가 저장되는 곳, FIFO 방식으로 동작

- 상용 환경에서는 최소 3대 이상의 브로커(서버)로 클러스터를 구성 → 가용성
- 데이터를 묶음 단위로 처리하는 배치 전송 → 낮은 지연, 높은 처리량

데이터 레이크, 데이터 파이프라인

- 수십 테라바이트가 넘어가는 빅데이터를 기존의 데이터베이스로 관리하는 것은 사실상 불가능 → 일단 생성되는 데이터를 모두 모으는 것이 중요 → 데이터 레이크
- **데이터 레이크** : 데이터 웨어하우스와 달리 필터링되거나 패키지화되지 않은 비가공 데이터가 저장되는 공간 (즉, 서비스로부터 수집 가능한 모든 데이터를 모음)
- **데이터 파이프라인** : 엔드 투 엔드 방식의 데이터 수집 및 적재를 개선하고 안정성을 추구하며, 유연하면서도 확장 가능하게 자동화 한 것 (데이터 레이크의 구현 방식)

빅데이터 파이프라인으로서의 카프카



[그림 1.2-1] 카프카 생태계 모식도

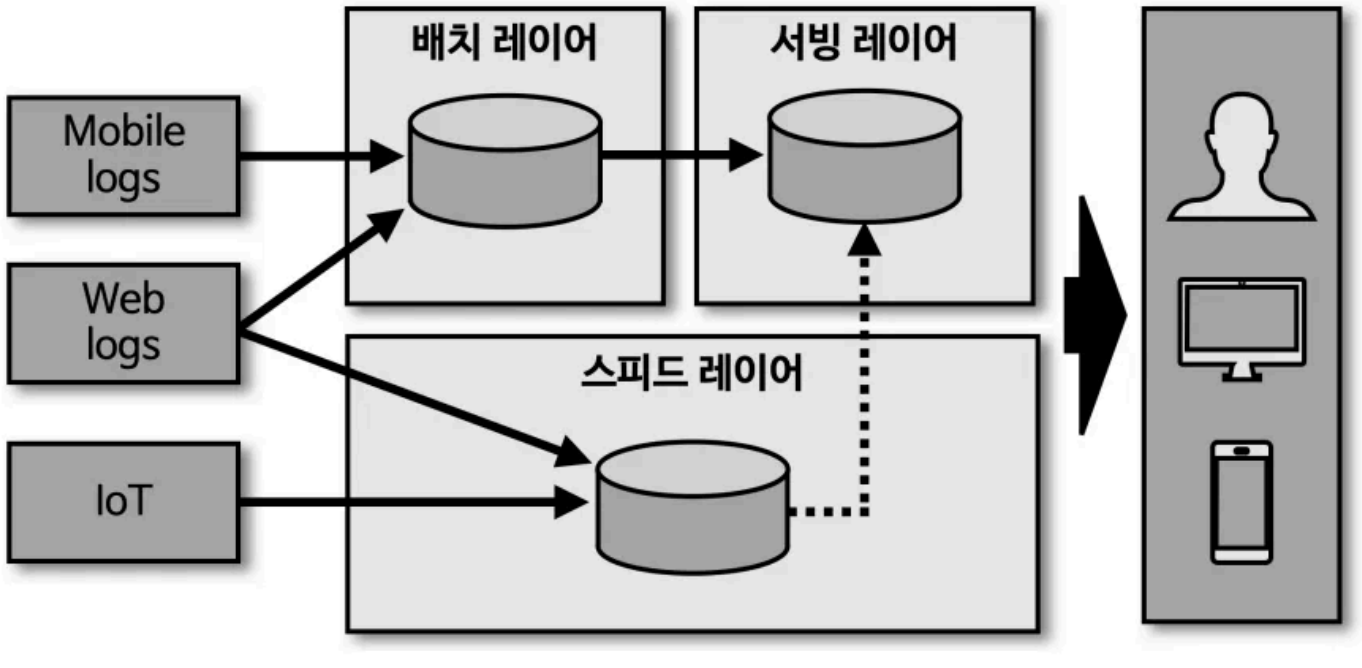
- **높은 처리량** : 배치 전송을 통해 네트워크 통신 횟수를 최소화하고, 동일 목적의 데이터를 여러 파티션에 분산 저장함으로써 데이터를 병렬로 처리
- **확장성** : 가변적인 환경에서 안정적인 스케일 인/아웃이 가능하도록 설계됨, 스케일링 과정은 클러스터의 무중단 운영을 지원 → 365일 24시간 비즈니스 모델에서도 안정적인 운영이 가능
- **영속성** : 다른 메시징 플랫폼과 달리 데이터를 파일 시스템에 저장 (운영체제의 페이지 캐시를 활용하여 처리량을 향상) → 디스크 기반의 파일 시스템을 활용한 덕분에 장애 발생 시 안정적인 데이터 재처리가 가능
- **고가용성** : 3개 이상의 브로커로 운영되는 카프카 클러스터는 복제(replication)를 통해 고가용성을 확보

안정적인 카프카 운영에 3대 이상의 클러스터 구성이 필요한 이유

- 1대 : 브로커의 장애가 서비스의 장애로 이어지므로 테스트 목적으로만 사용 (SPOF)
- 2대 : 브로커 간 데이터가 복제되는 시간 차이로 인해 일부 데이터가 유실될 가능성이 존재
- 유실을 막기 위해 min.insync.replicas 옵션을 2로 설정하면 최소 2개 이상의 브로커에 데이터가 완전히 복제됨을 보장하며, 이 때 브로커를 3대 이상으로 운영해야만 한다. 3개 중 1개의 브로커에 장애가 나더라도 지속적으로 데이터를 처리할 수 있기 때문이다.

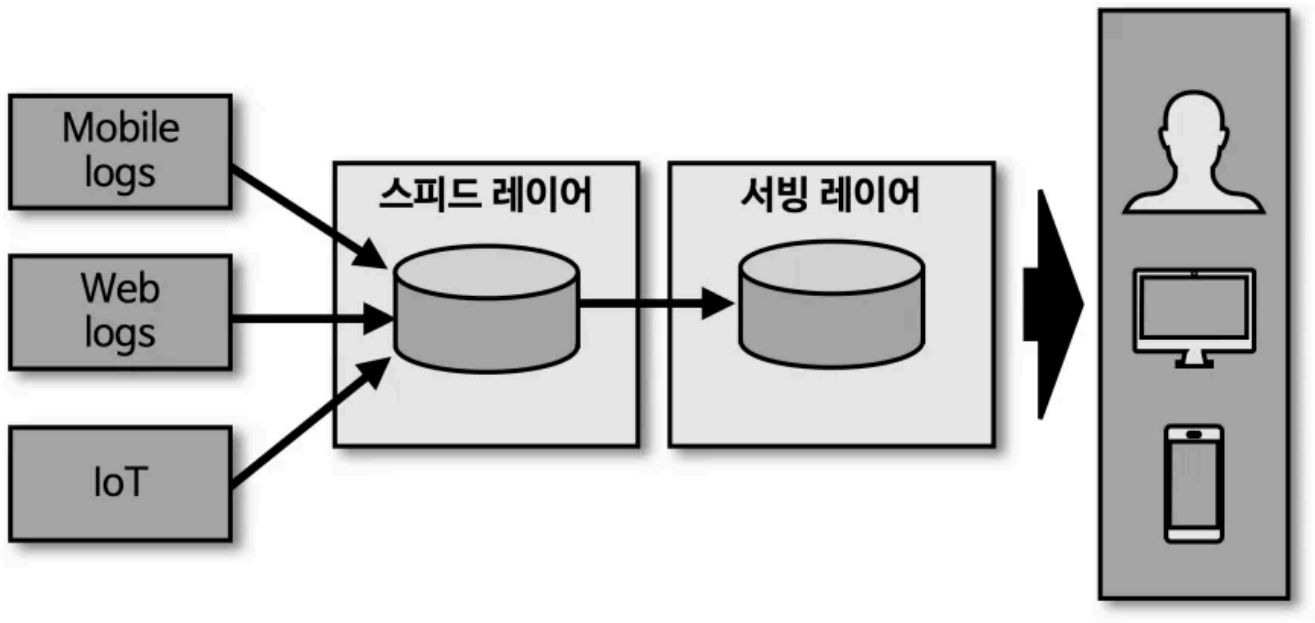
데이터 레이크 아키텍처

[람다 아키텍처]



- 구성
  - **배치 레이어** : 배치 데이터를 모아 특정 타이밍마다 일괄 처리
  - **서빙 레이어** : 데이터가 저장되는 공간
  - **스피드 레이어** : 배치 데이터에 비해 낮은 지연으로 분석이 필요한 경우 원천 데이터를 실시간으로 분석하는 용도로 사용, 카프카는 스피드 레이어에 위치
- 한계 : 데이터를 처리하는데 필요한 로직이 2개의 레이어(배치, 스피드)에 분산되는 **로직 파편화 문제** 존재

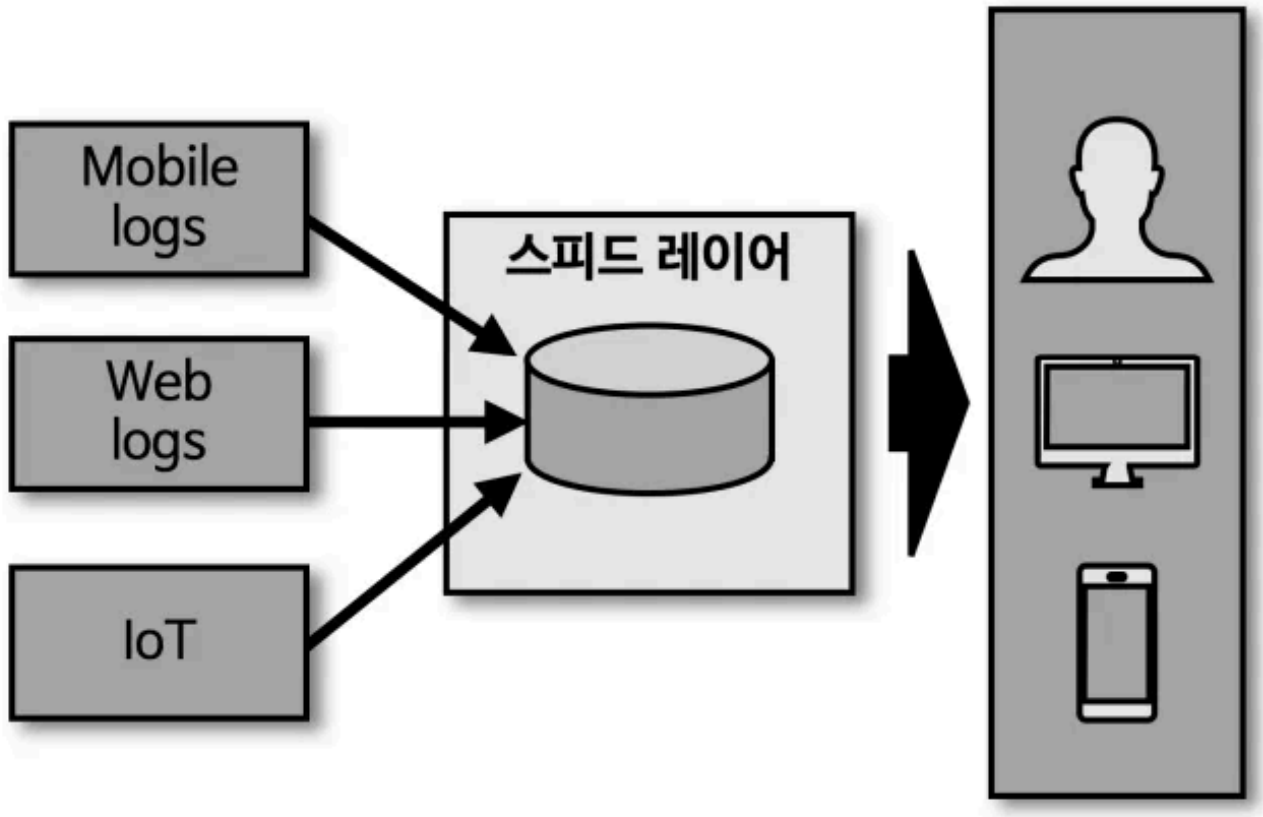
[카파 아키텍처]



- 람다 아키텍처의 문제점인 로직 파편화를 해결하기 위해 배치 레이어를 제거한 형태
- 스피드 레이어에서 모든 데이터를 처리하므로 서비스에서 생성되는 모든 종류의 데이터를 스트림 처리해야 함
  - **배치 데이터** : 초, 분, 시간, 일 등 시작 데이터와 끝 데이터가 명확히 정해진 데이터
  - **스트림 데이터** : 시작 데이터와 끝 데이터가 명확히 정해지지 않은 데이터 (사용자의 클릭 로그, 주식 정보, IoT의 센서 데이터)
- **배치 레이어 제거가 가능한 이유** : 로그를 통해 배치 데이터를 스트림 데이터로 표현
  - 전체 데이터를 백업한 스냅샷 데이터가 아닌, 각 시점의 배치 데이터의 변환 기록(change log)을 시간 순서대로 기록하여 배치 데이터를 표현
- 배치 데이터를 스트림 데이터로 표현하기 위해서는 다음 두가지 조건이 만족되어야 함
  1. 로그가 일정 시간동안 삭제되어서는 안되고 지속적으로 추가되어야 하며
  2. 모든 데이터가 스피드 레이어로 들어오기 때문에 SPOF가 될 수 있으므로 내결함성과 장애 허용 특징을 지녀야만 함

  → 이러한 특징에 정확히 부합하는 것이 카프카

[카프카 미래 : 스트리밍 데이터 레이크 아키텍처]



- 스피드 레이어로 사용되는 카프카에 분석과 프로세싱을 완료한 거대한 용량의 데이터를 오랜 기간 저장하고 사용할 수 있다면 서빙 레이어는 제거되어도 됨.
- 서빙 레이어와 스피드 레이어가 이중으로 관리되는 운영 리소스를 줄일 수 있음.
- 이를 위해 자주 사용하는 데이터와 자주 사용하지 않는 데이터 분리 기능을 개발 중
- 자주 접근하지 않는 데이터를 대용량의 오브젝트 스토리지로 저장하면서 자원을 효율적으로 운영 및 관리
- 스피드 레이어에서 데이터를 분석, 프로세싱, 저장하는 SSOT(Single Source Of Truth)
  - 데이터 중복, 비정합성과 같은 문제에서 벗어날 수 있음
- 카프카의 데이터를 쿼리할 수 있는 주변 데이터 플랫폼이 필요 → ksqlDB
  - **ksqlDB** : 카프카의 데이터를 SQL 기반으로 조회할 수 있도록 카프카 스트림즈를 추상화한 것

정리

- 카프카는 대기업 및 은행 뿐만 아니라, 안정적인 운영 및 빠른 확장이 가능하기 때문에 성장 속도가 빠른 스타트업에서도 유용하게 사용 가능
- 카프카 활용 방안은 아키텍처 구성에 따라 무궁무진함 → 아키텍처 구성을 정하려면 카프카의 특징과 동작 방식에 대해 면밀하게 알아야 함 → 알아보자