01. Kafka 개요

장태영(XW - JANG TAE YOUNG)님이 작성, 9월 24, 2024에 최종 변경

1. Kafka란?

1.1. 등장 History

- Apache Kafka는 2010년에 LinkedIn 엔지니어들에 의해 처음 개발된 분산 스트리밍 플랫폼
- 2011년 초에 오픈 소스로 공개된 후 2012년 Apache Incubator에서 졸업
- Kafka의 이름은 공동개발자 중의 한사람인 Jay Kreps가 프란츠 카프카(Franz Kafka)의 이름에서 따왔으며 "쓰기에 최적화된 시스템"의 의미를 가짐

1.2. 용도 및 특징

- Apache Kafka는 분산 이벤트의 저장 및 스트림을 처리하는 메시징툴
- 실시간 데이터 피드를 처리하기 위한 고처리량, 저지연 플랫폼
- 메시지를 디스크에 지속적으로 저장하여 내구성을 보장
- 클러스트 간 복제를 통해 데이터를 안전하게 보관
- 많은 기업에서 Async 방식 메시징 플랫폼으로 확장하여 활용 중

1.3. 메시징 솔루션 종류

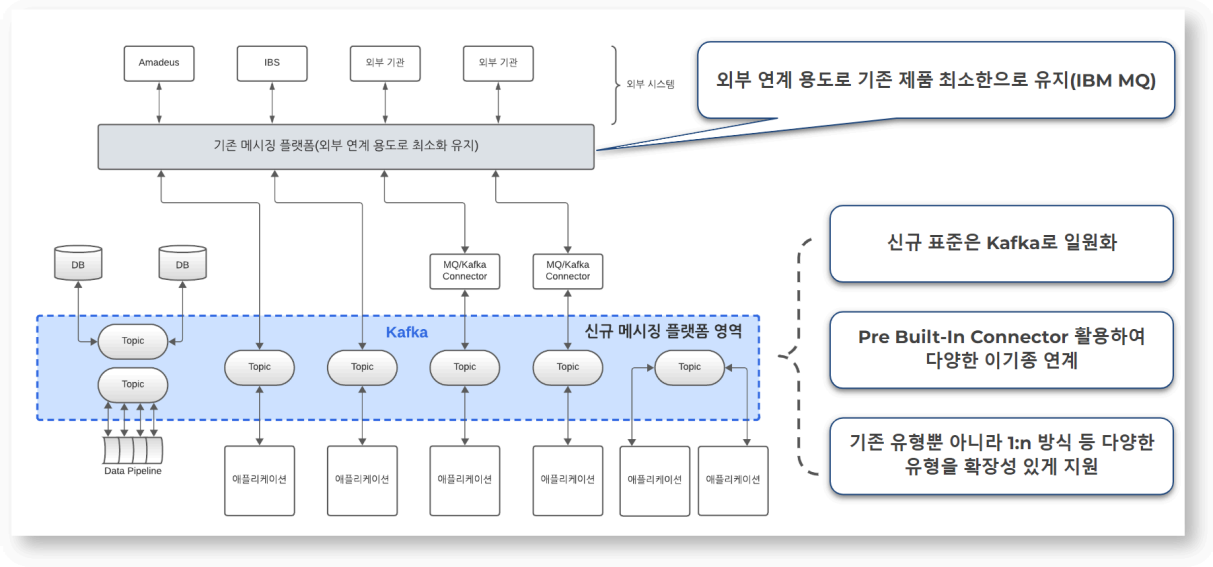
- Kafka는 다른 메시징 솔루션에 비하여 비교적 늦게 등장하였으나 최근 MSA 및 EDA에 적합한 메시징툴로 많이 사용됨

메시징 솔루션 종류 및 주요 용도	대표적 제품
▣ Extract, Transform, Load(ETL) Tools ☞ 다양한 소스의 데이터를 불출 및 변환하여 타겟에 로드하며 DW업무에 주로 사용	- Informatica, AWS Glue, IBM DataStage
▣ Message Queue ☞ 비동기 Decoupled 큐방식 데이터 통합툴이며 안정성이 장기간에 걸쳐 입증됨	- IBM MQ, AWS SQS, Apache ActiveMQ
▣ Enterprise Application Integration(EAI) ☞ MSA 및 Event Driven 기술 발달전 1990 ~ 2000대 등장한 APPL 통합 기술	- Oracle SOA Suite, Tipco, Red Hat Fuse
▣ Application Programming Interface Management(APIM) ☞ EAI에 비하여 가볍고 Loosely Coupled로 표준화된 API 표준 기술	- Google Apigee, MS Azure APIM
▣ Data Streaming Solution ☞ 실시간 Streaming 처리 솔루션으로서 최근의 MSA 및 EDA에 적합한 메시징툴	- Confluent Kafka, AWS MSK, Apache Kafka
▣ Data Virtualization ☞ 가상화된 Data Layer를 통하여 데이터 복제 및 이동 없이 통합된 View를 제공	- Denodo Platform, TIBCO Data Virtualization

2. 당사 도입

2.1. 배경

- 기존 IBM MQ 제품은 비용 및 기술 측면에서 유연하지 못한 점이 있어 클라우드 기반 오픈소스 메시징 솔루션을 추가로 도입
- IBM MQ 제품은 외부 연계 용도로 지속 사용



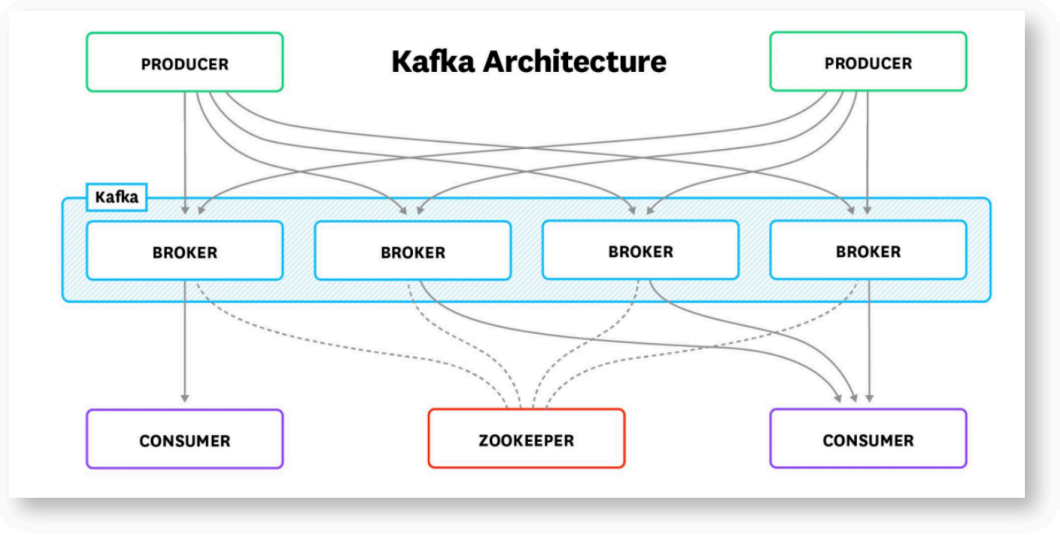
2.2. 도입 제품

- On-premise형 Apache Kafka, SaaS형 AWS MSK 및 Confluent Kafka 등을 비교 검토하여 Confluent Cloud를 도입
- 클라우드 확장성을 지원하며 다양한 Built-in Connector 제공 등을 통하여 엔터프라이즈 적용 사례가 많은 점이 고려됨
- Kafka 솔루션은 Confluent사에서 SaaS 방식으로 운영되나, Connector는 비용 등을 고려하여 당사 AWS EKS에 설치 운영

3. 주요 구성

3.1. High Level 아키텍처

- Kafka는 이벤트, 즉 메시지나 레코드를 Producers에서 Consumers로 전달
- Kafka의 Cluster 및 Broker 등의 Instance는 Confluent사에서 책임 운영



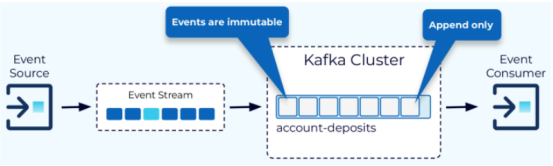
3.2. Kafka Topic과 파티션

- **Topic이란 ?**
  - 특정 이벤트(데이터)에 대한 명명된 저장공간
    - **데이터를 구분하여 최종적으로 저장하기 위한 저장소**
    - 시스템에는 많은|Topic이 포함되어 있음
    - Topic 은 **N 개의 Partition으로 구성**
  - 이벤트의 지속 가능한 **로그**
    - 추가만 가능
    - Offset 으로만 검색 가능, 인덱싱되지 않음
  - 이벤트는 **변경 불가능**

    → 지속가능

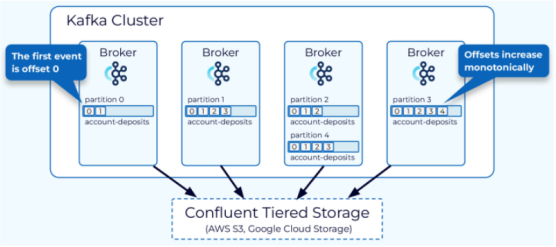
    → 보존기간 설정 가능

- 대용량 병렬 처리가 필요할 경우 Multi-Partition을 운영하여 처리 속도를높일 수 있으나 대부분의 당사 요건은 Single Partition으로 대응 가능



Topic(Kafka의 핵심 개념)

- Kafka 내부의 이벤트(데이터) 저장소
- 추가만 가능하며 변경 불가한 이벤트 로그
- 토픽에서 이벤트의 처리를 분산하기 위하여 파티션 개념 사용
- 이론상 용량 제한은 없으며 Retention Time과 Size에 의해 제한될 수 있음



Partition

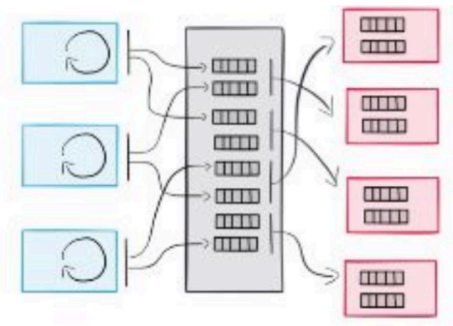
- Kafka 이벤트의 기본 저장 단위이며 병렬처리의 기본단위
- 하나의 Topic은 하나 이상의 Partition으로 구성됨
- 병렬처리를 위해서 다수의 Partition을 사용
- 다수의 Client가 여러 브로커에 데이터를 Read/Write 가능
- 분산 클러스터 구조로 고가용성과 확장성 제공
- Multi 파티션에서 Key 기반 Hash Algorithm을 이용한 Ordering을 지원

3.3. Producer와 Consumer

Producer	Consumer

• 프로듀서가 데이터를 메시지로 작성
• 모든 언어로 작성 가능
  - Java, C#, Python, .Net 등 모두 지원 가능
  - 단, 당사의 Application Modernization의 표준 언어가 Java/Spring Boot이므로 이를 권장
  - 지원되지 않는 언어에 대한 REST 프록시 지원
• Command Line 프로듀서 도구 사용 가능
	• 컨슈머는 1개 이상의 토픽에서 메시지를 가져와 처리
	• 새로 유입되는 메시는 자동으로 검색
	• Consumer Offset
	  - 마지막으로 읽은 메시지 추적
	  - 내부의 특수 토픽에 저장됨(_consumer_offsets)
	• 클러스터에서 읽기 위한 CLI 도구 존재

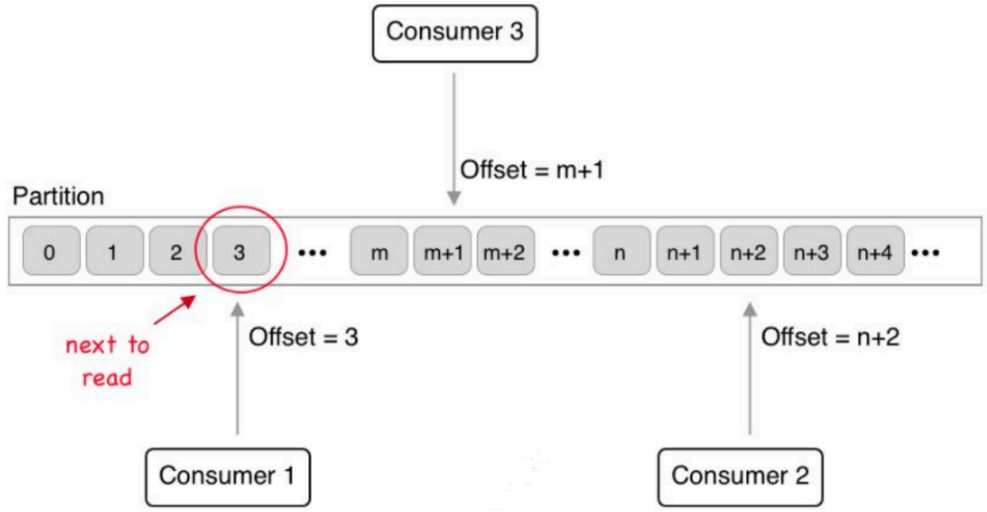
- Decoupling Producer & Consumer
  - 프로듀서와 컨슈머는 분리되어 있음
  - 느린 컨슈머는 프로듀서에 영향을 미치지 않음
  - 프로듀서에게 영향을 주지 않고 컨슈머 추가
  - 컨슈머 장애가 시스템에 영향을 주지 않음



3.4. Consumer Offset

- Kafka는 Consumer Group이 읽은 Offset을 저장
- Consumer Offset은 __consumer_offsets라는 내부 Topic에서 커밋됨
- Consumer Group이 Kafka에서 받은 데이터를 처리하면 Offset을 커밋해야 함
- Consumer가 죽으면 Kafka는 Consumer가 어디에 있었는지 알려주므로 다시 읽을 수 있음

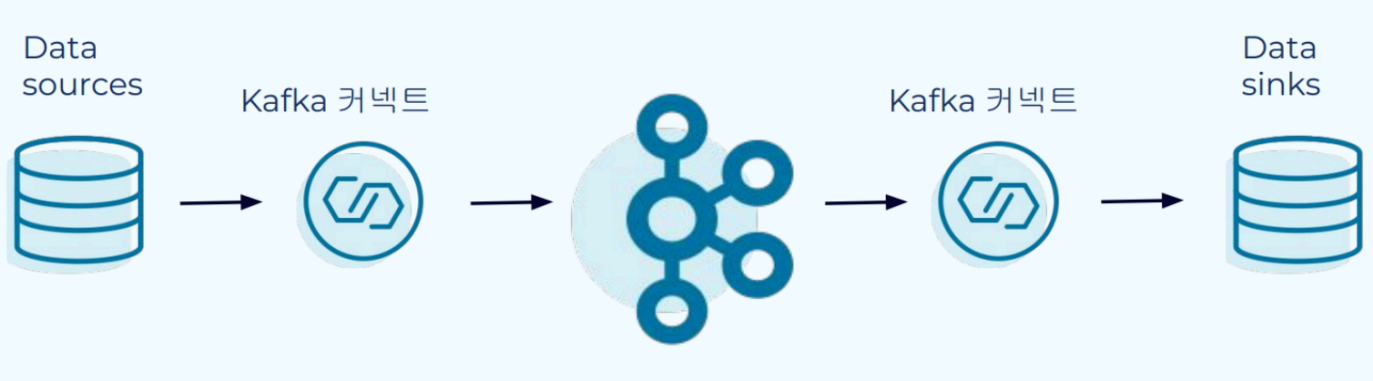
- 동일한 Topic 정보를 여러 Consumer가 각자의 Offset을 사용하여 같이 사용할 수 있음
- 아래의 Consumer 1, Consumer 2, Consumer 3은 다른 애플리케이션



3.5. Kafka Connect

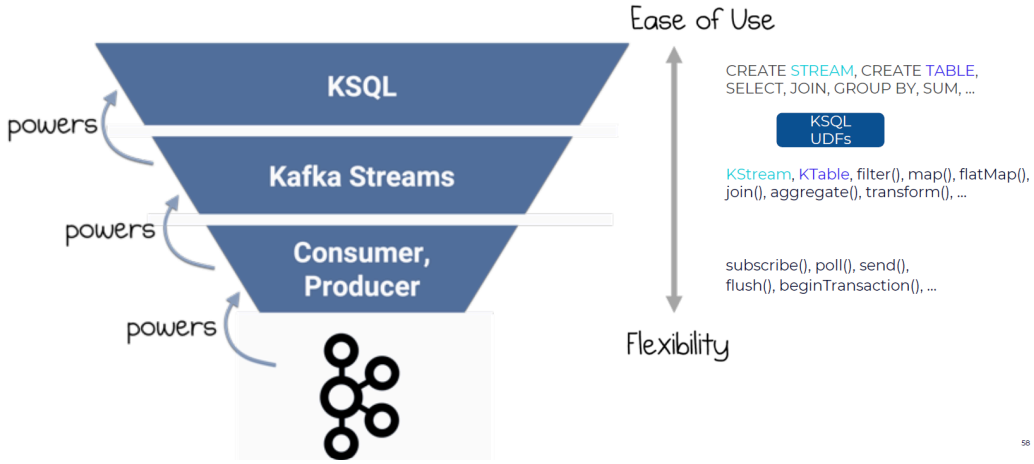
- 알려진 시스템들(데이터베이스, 스토리지, 메시지큐 등)을 코드가 필요 없이 Apache Kafka에 연결 가능
- 현재 IBM MQ Source Connector, IBM MQ Sink Connector, HTTP Sink Connector 등 3개 Connector 사용 중

⭐ IBM MQ Source Connector는 At-Least-Once Delivery를 지원하며 2024년 내 Exactly-Once Delivery 지원 예정



3.6. ksqlDB

- ksqlDB는 Confluent에서 개발한 스트리밍 엔진으로 , Apache Kafka와 통합되어 실시간 데이터 처리를 간편하게 수행할 수 있게함
- ksqlDB를 사용하면 KAFKA 내에서 스트리밍 파이프라인의 구성할 수 있음. 단, Trouble Shooting 시 단순한 메시지 전달 경우보다 복잡함
- 익숙한 SQL 문법을 사용하여 데이터를 처리할 수 있음 (집계, 조인, 윈도잉 등)



3.7. Schema Registry

- Schema Registry는 Confluent에서 개발한 중앙집중식 스키마 관리 도구
- Apache Kafka 메시지 데이터의 스키마를 관리하고 유효성 검사를 수행
- JSON 데이터 타입을 사용할 경우 Schema Registry를 사용하지 않으나 메시지 크기를 줄이기 위해 AVRO타입을 사용할 경우 Schema Registray 등록하여 사용
- Kafka 프로듀서와 컨슈머는 스키마를 사용하여 데이터 일관성과 호환성을 보장할 수 있음

- 스키마가 변경될 때도 데이터 호환성 유지 가능
- 데이터 품질, 표준 준수, 감사 기능 지원

레이블 없음