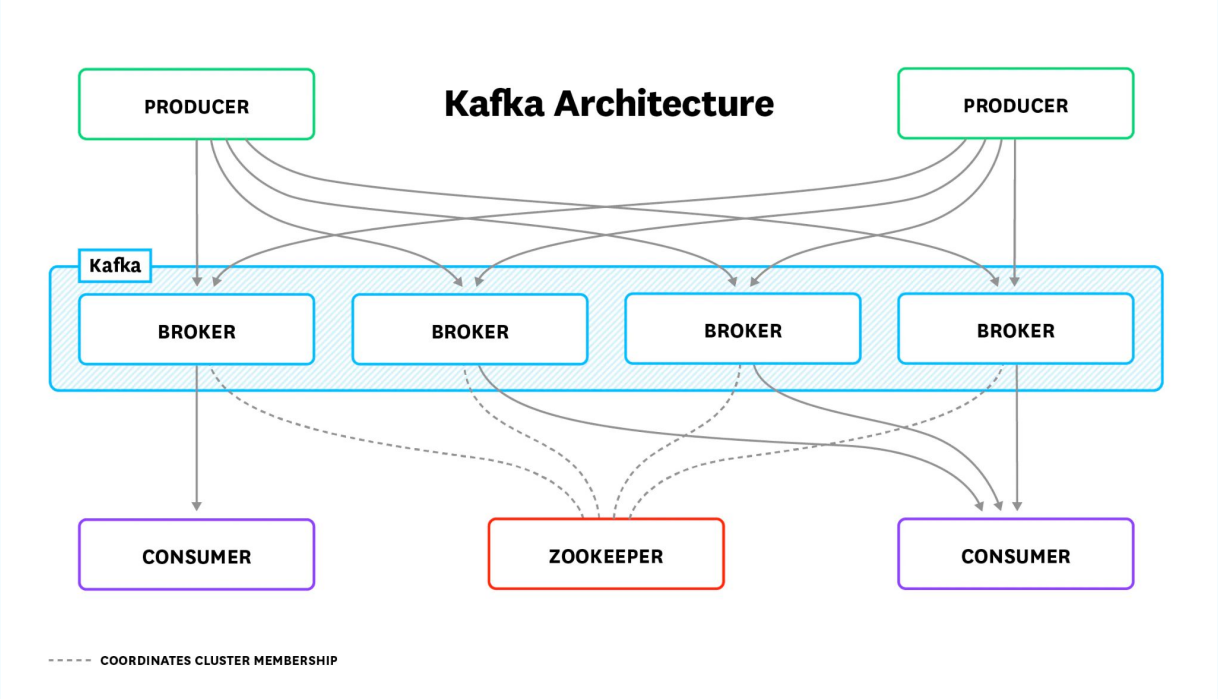
Confluent Basics

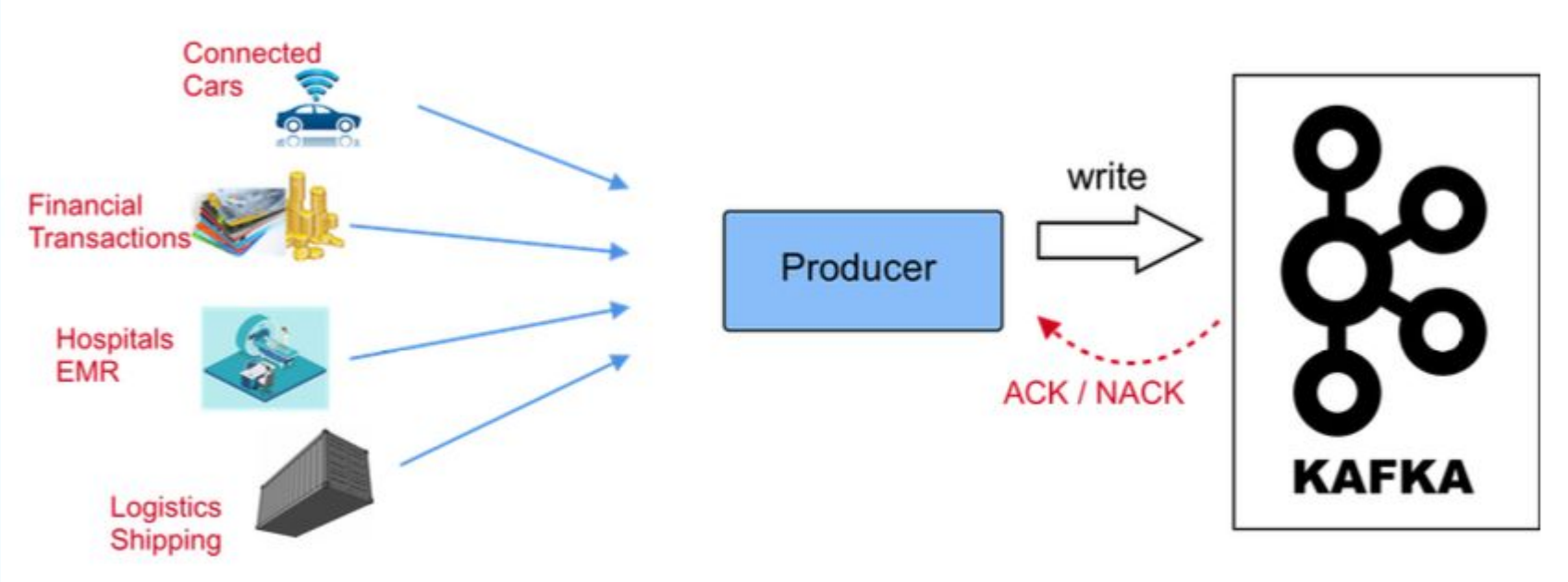
Kafka 101

High Level Architecture



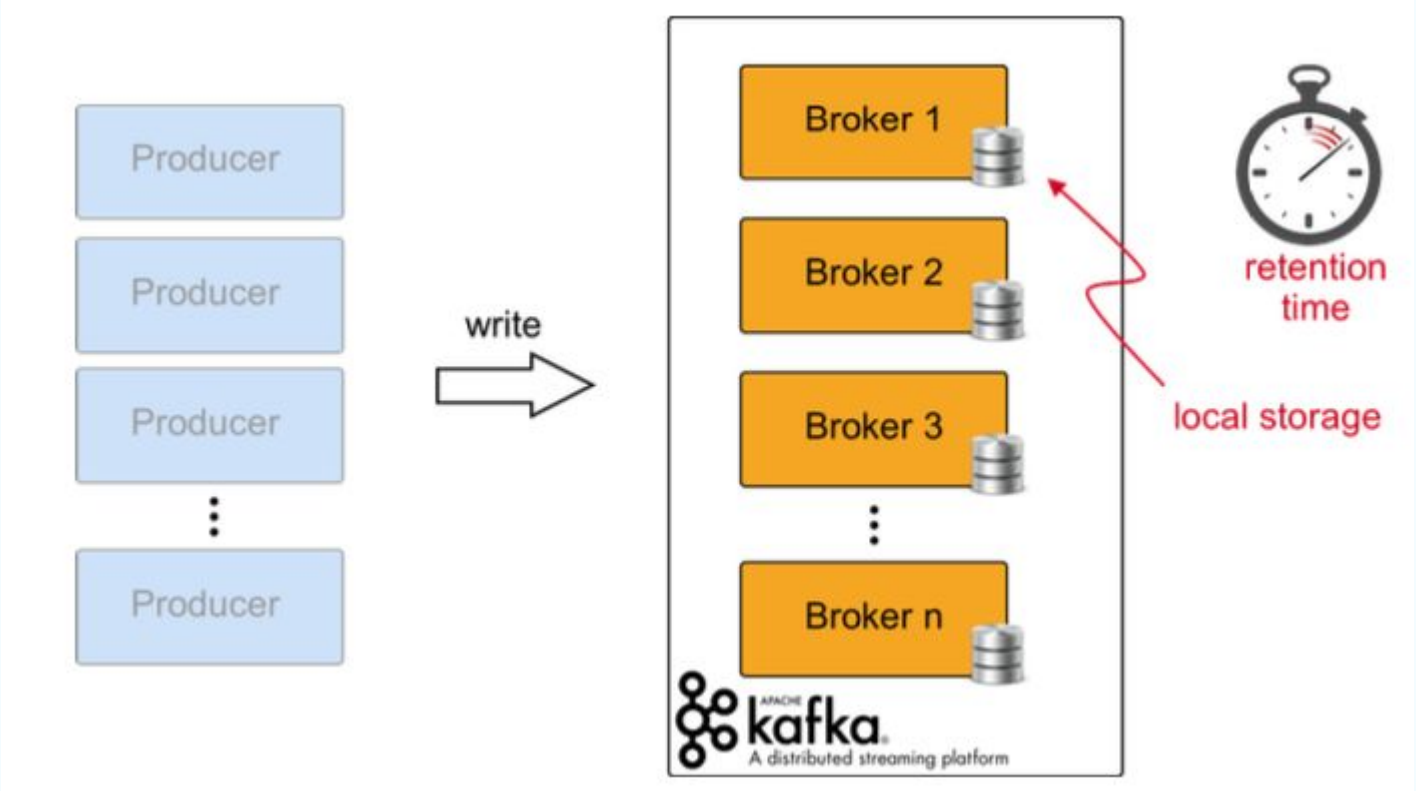
Kafka는 이벤트(메시지/레코드)를 Producers에서 Consumers로 전달

Producer



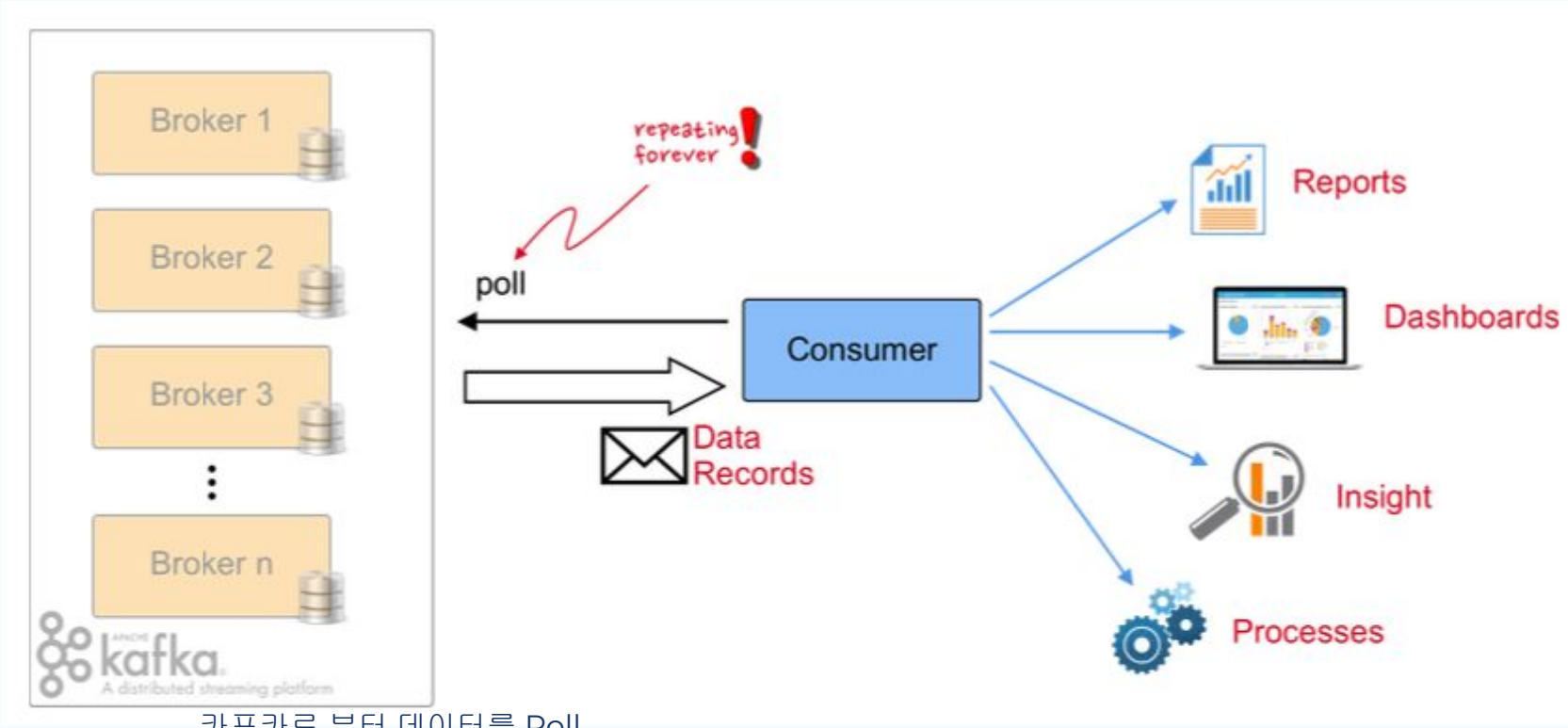
데이터를 카프카로 전송하는 역할을 하는것이 프로듀서
전송한 데이터에 대해 ACK 수신

Kafka Brokers



하나의 카프카 클러스터는 한 개 이상의 브로커로 구성
프로듀서로 부터 데이터를 받고 임시로 페이지 캐시에 저장하거나 영구적으로 스토리지에 저장
브로커가 데이터를 유지하는 시간 : retention time

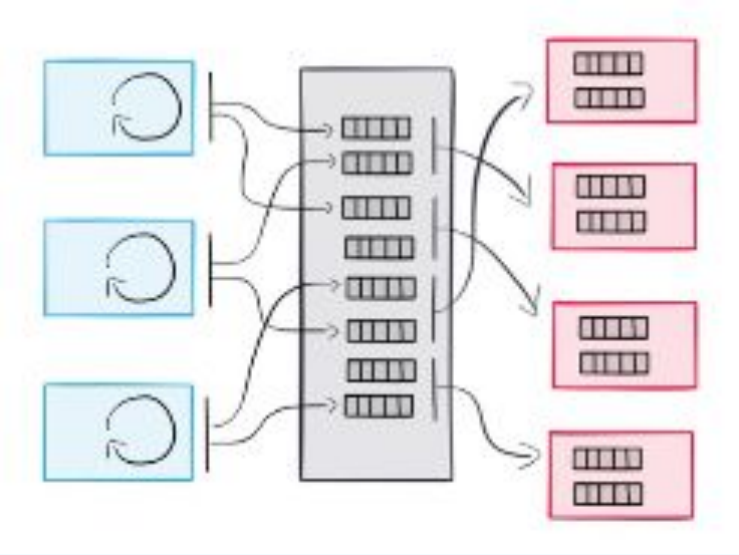
Consumers



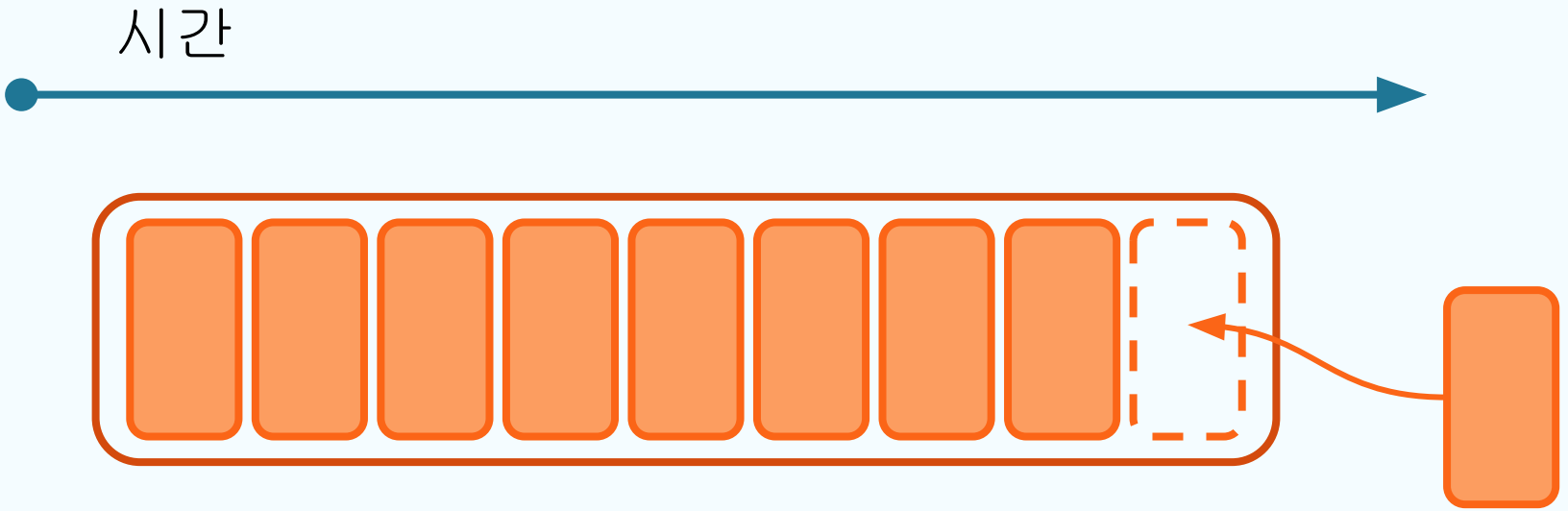
카프카로 부터 데이터를 Poll
각각의 컨슈머는 주기적으로 브로커에 데이터가 있는지 확인
여러개의 컨슈머가 동시에 polling 수행
컨슈머는 컨슈머그룹단위로 처리

Decoupling Producers and Consumers

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

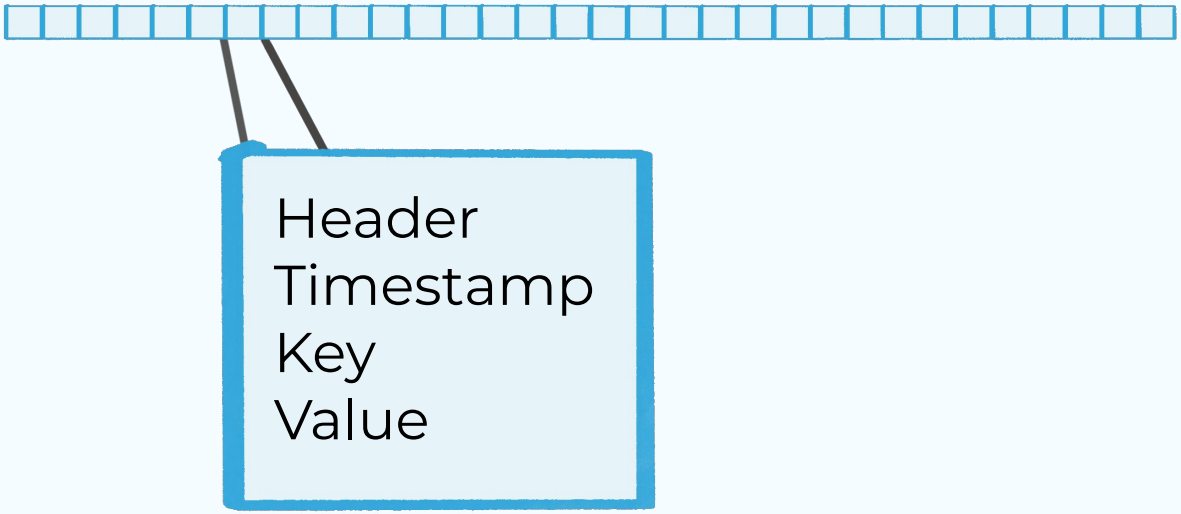


로그



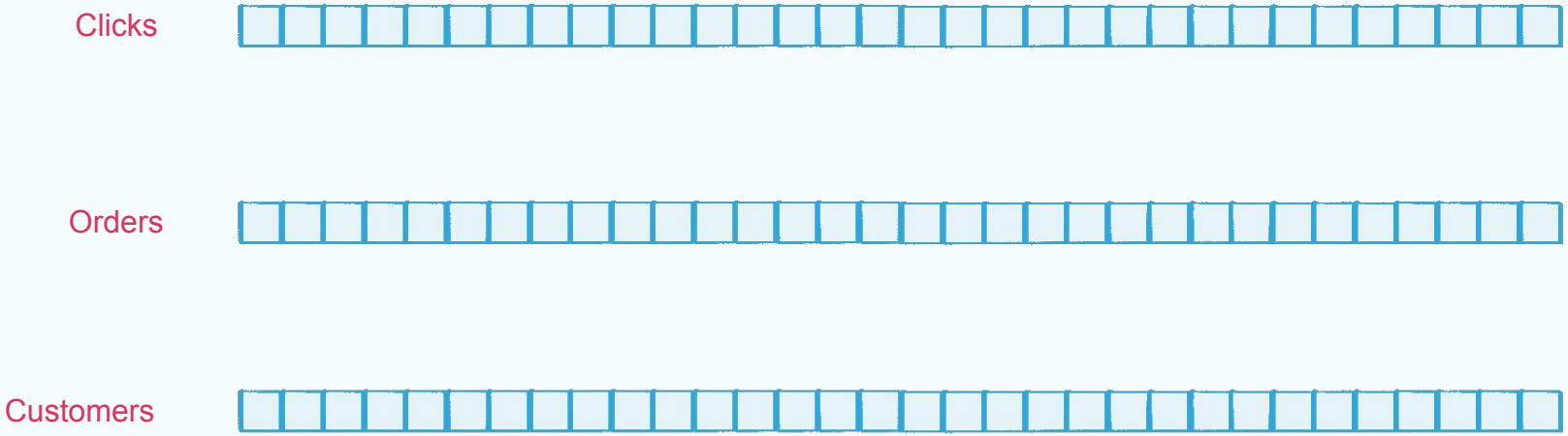
이벤트는 항상 로그의 맨 뒤에 추가됨

메시지 - 단지 키와 값을 가진 **Bytes** + 메타데이터



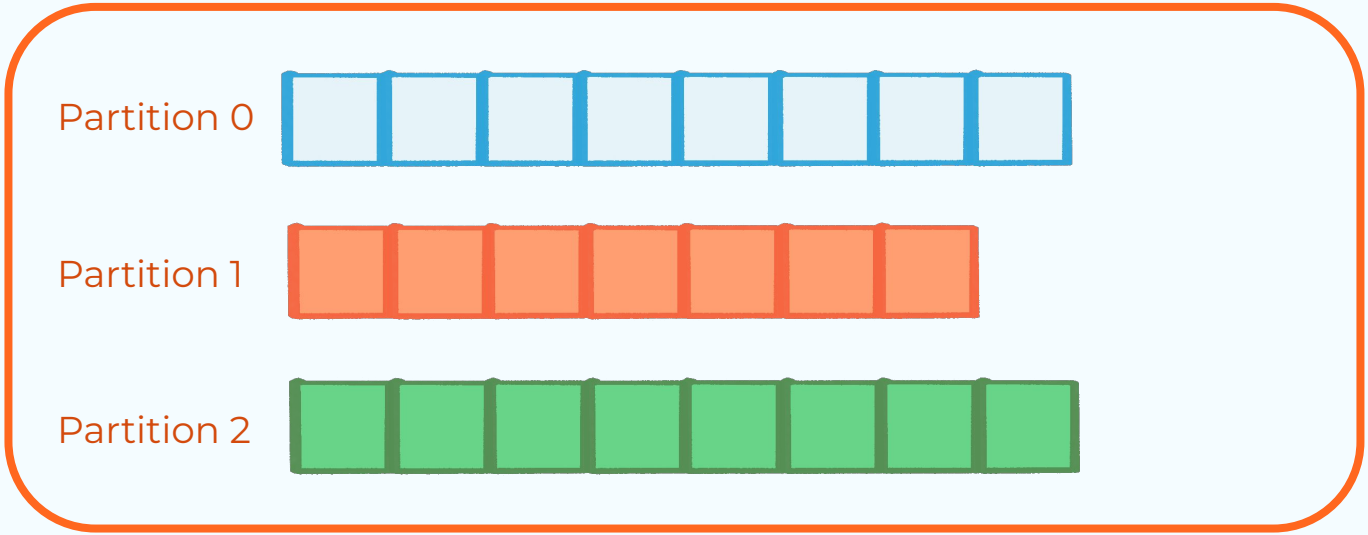
이벤트(메시지/레코드)는 메타데이터를 가진 키:값 쌍의 Byte Array

토픽



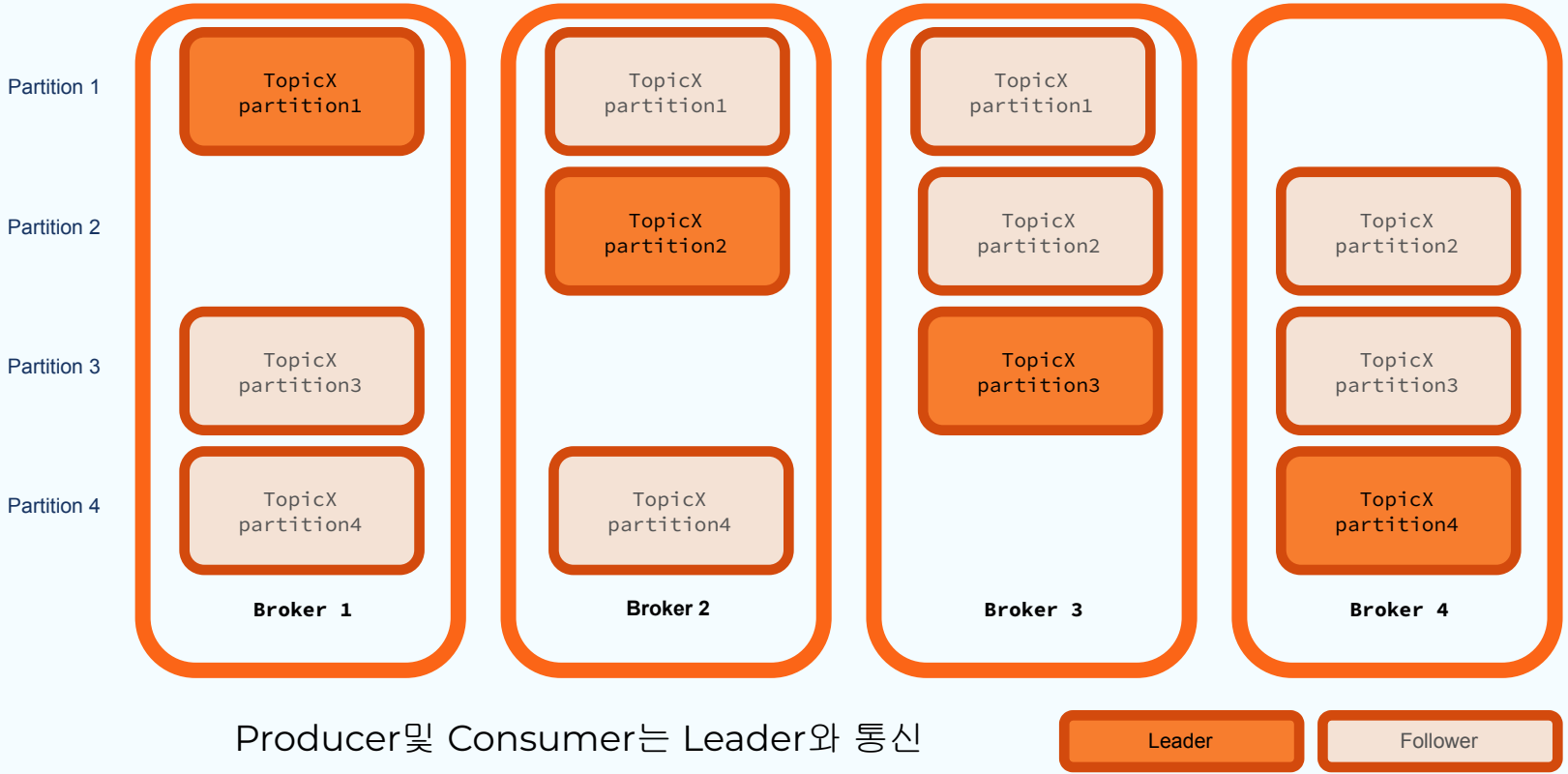
토픽 - 메시지의 논리적인 집합
(테이블이나 큐와 유사함)

Partitions



파티션은 토픽을 위한 물리적인 저장장소
각각의 토픽은 1개 혹은 그 이상의 파티션을 가질 수 있음
메시지의 순서는 파티션 내에서 보장됨

레플리카 - 파티션 리더 및 팔로워

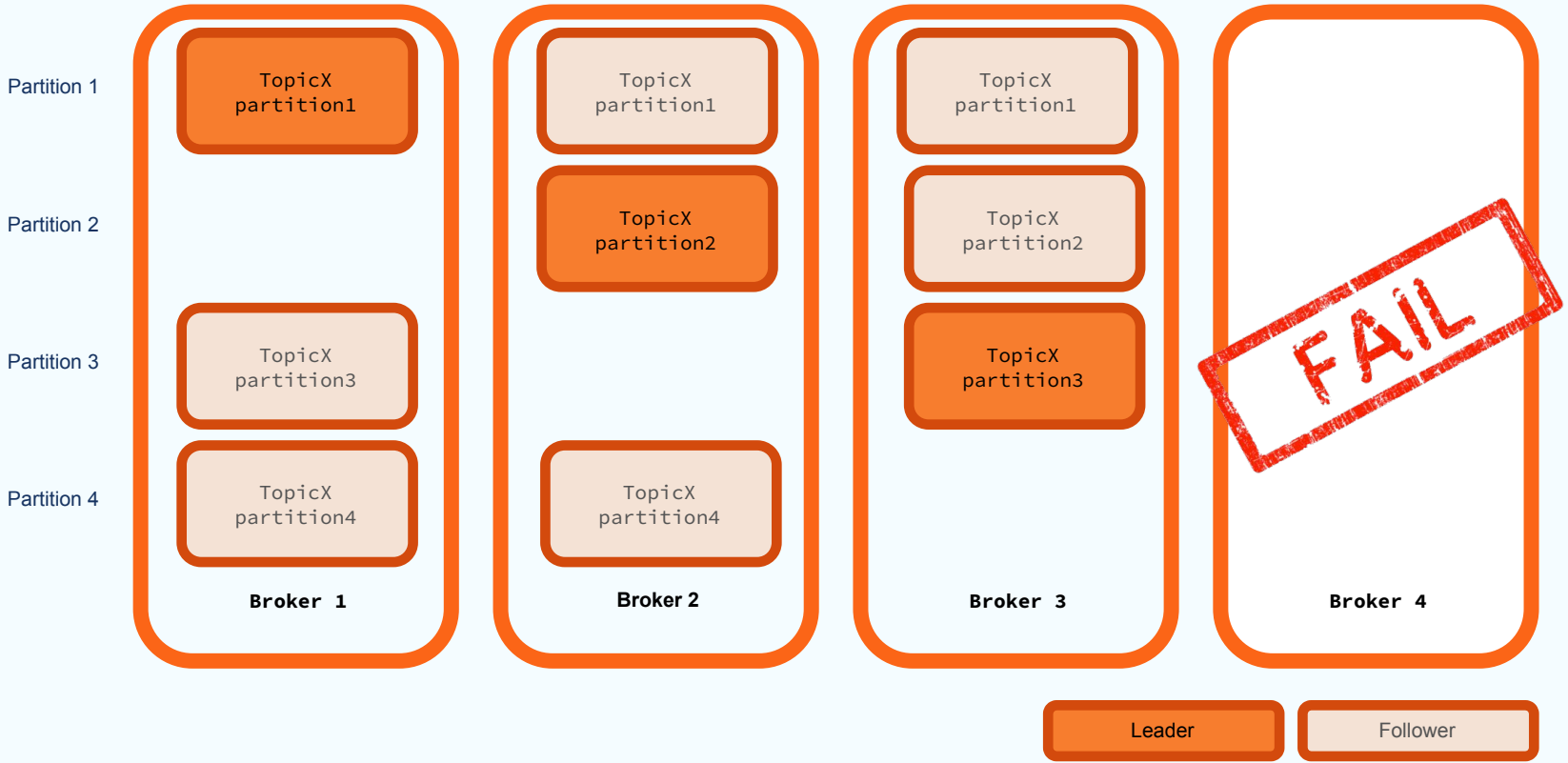


Producer및 Consumer는 Leader와 통신

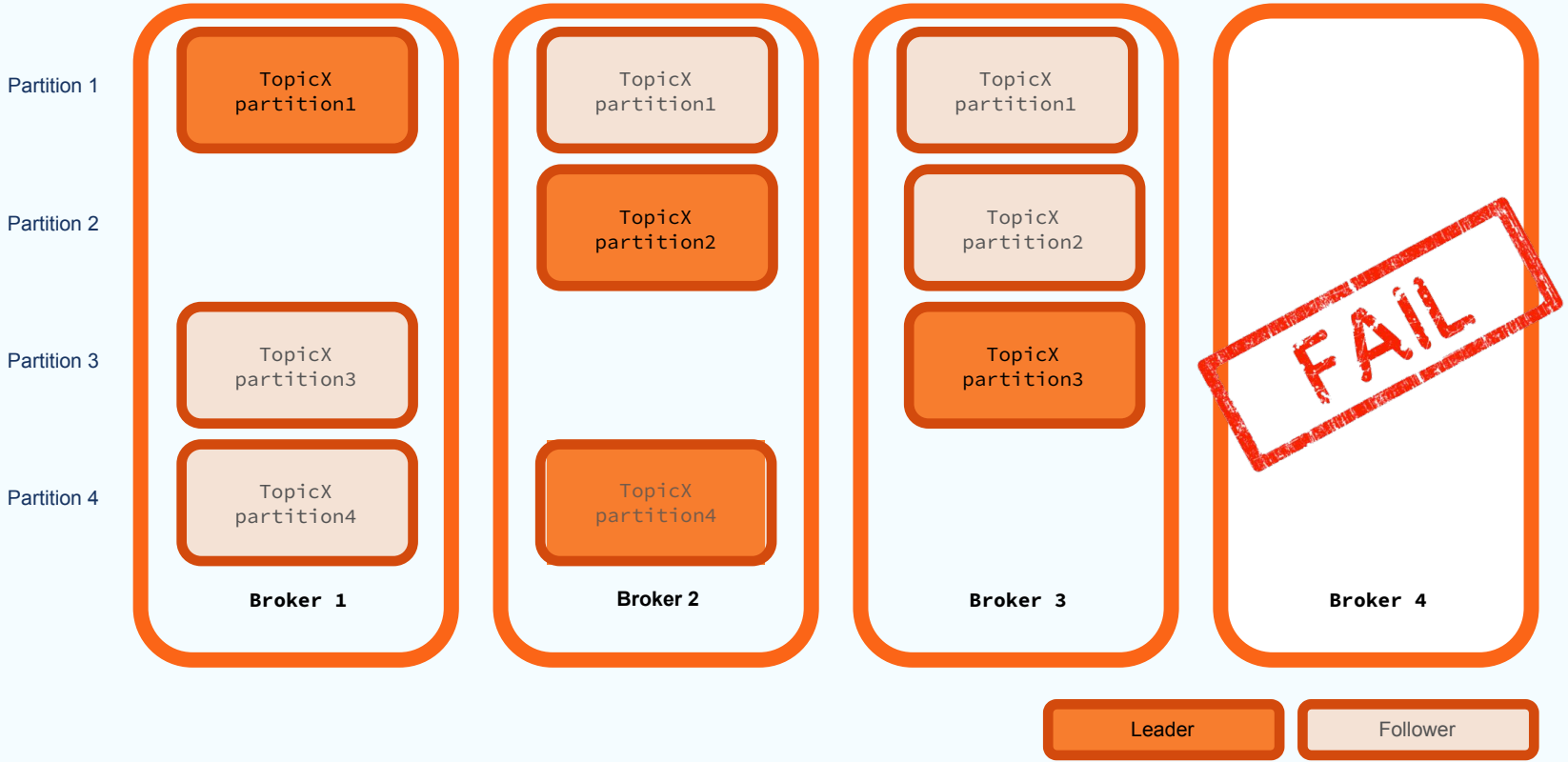
Leader

Follower

레플리카 - Broker 장애



레플리카 - 파티션 리더 & 복제



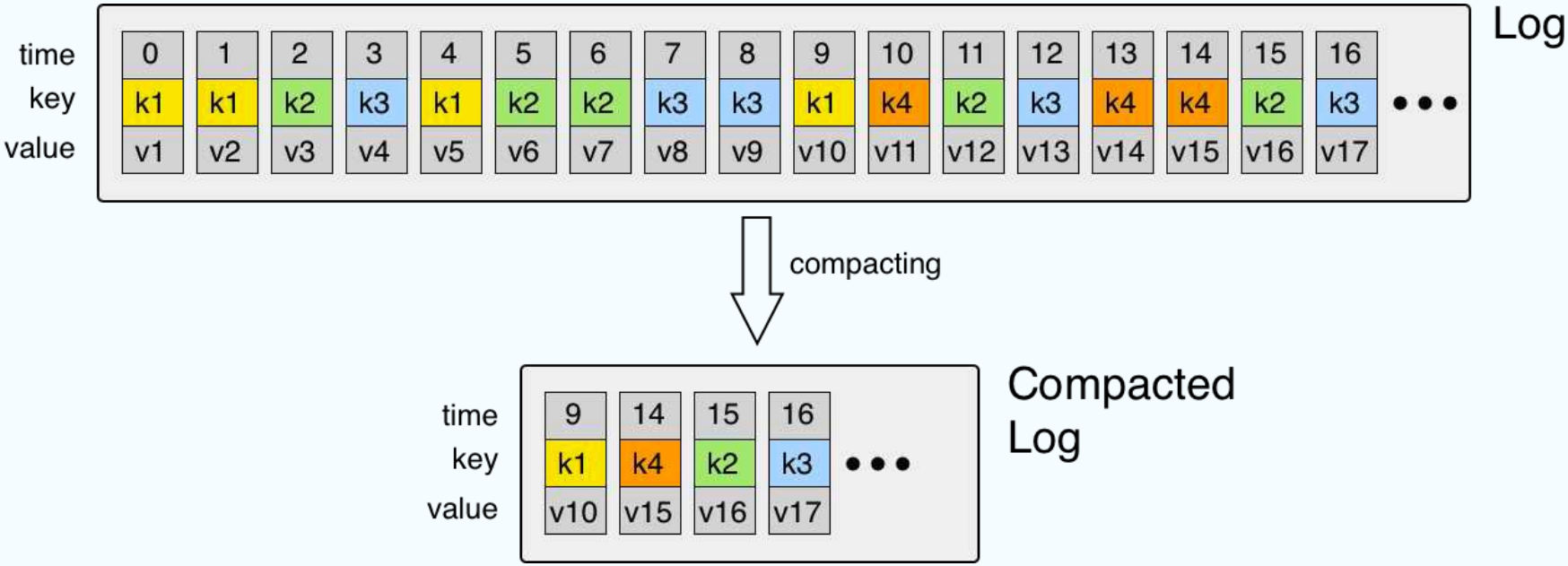
Retention 옵션

- Delete - 시간 혹은 공간기준

- Compact

retention.ms = 604800000ms(7일)
업무의 성격에 맞게 글로벌 혹은 토픽 단위로 설정. 비용과 관련됨
Compliance factor -> GDPR

Compacted Topics - 키별로 가장 최근의 값만을 유지

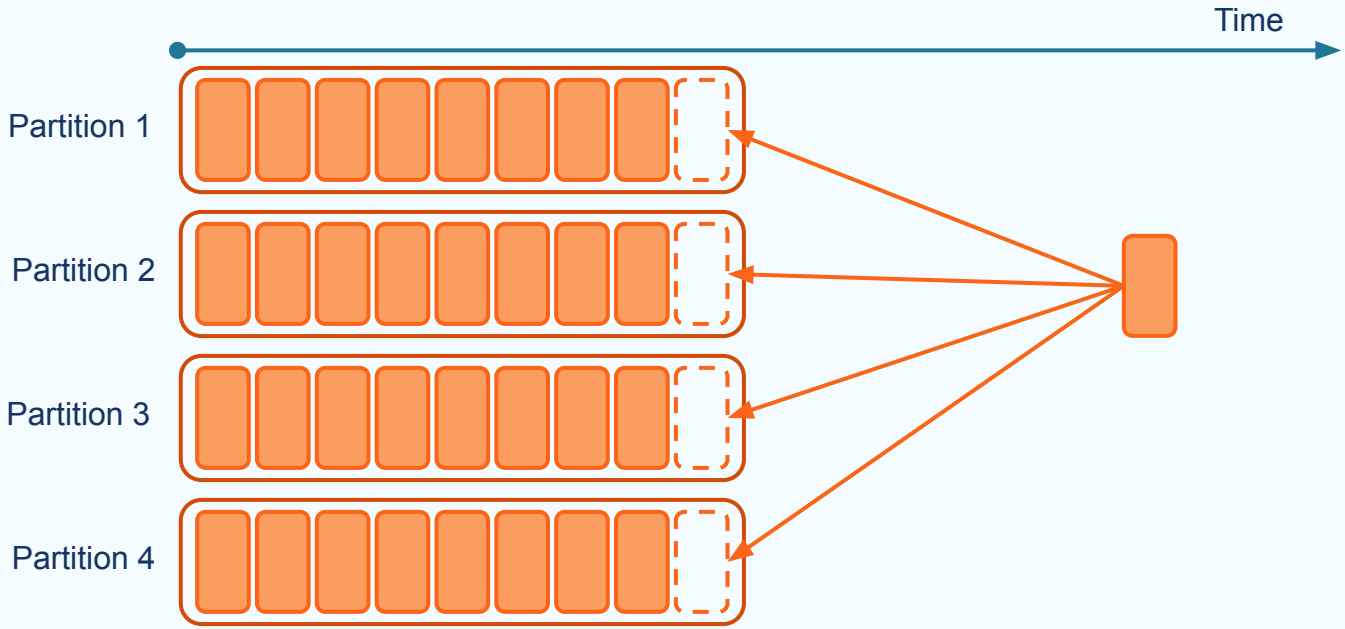


Producer

Producer

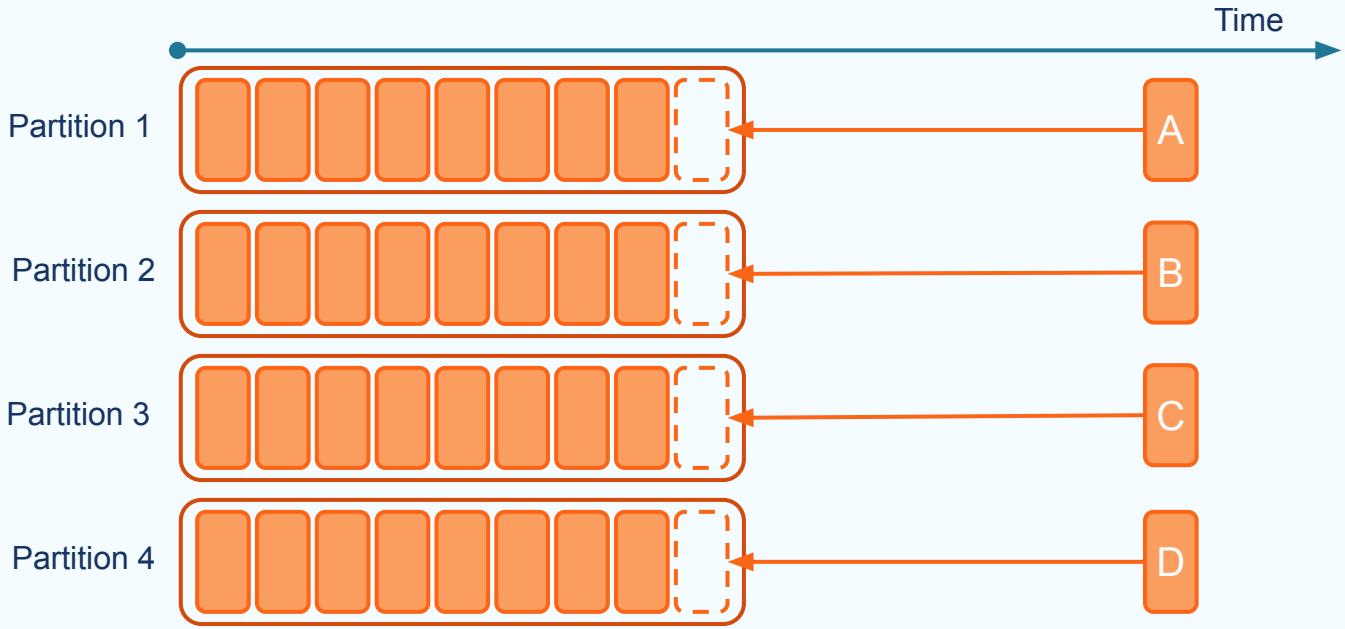
- 프로듀서가 데이터를 메시지로 작성
- 모든 언어로 작성 가능
    - 네이티브: Java, C/C++, Python, Go, .NET, JMS
    - 커뮤니티별 더 많은 언어 지원
    - 지원되지 않는 언어에 대한 REST 프록시 지원
- Command line 프로듀서 도구 제공

Produce to Kafka - Key가 없는 경우



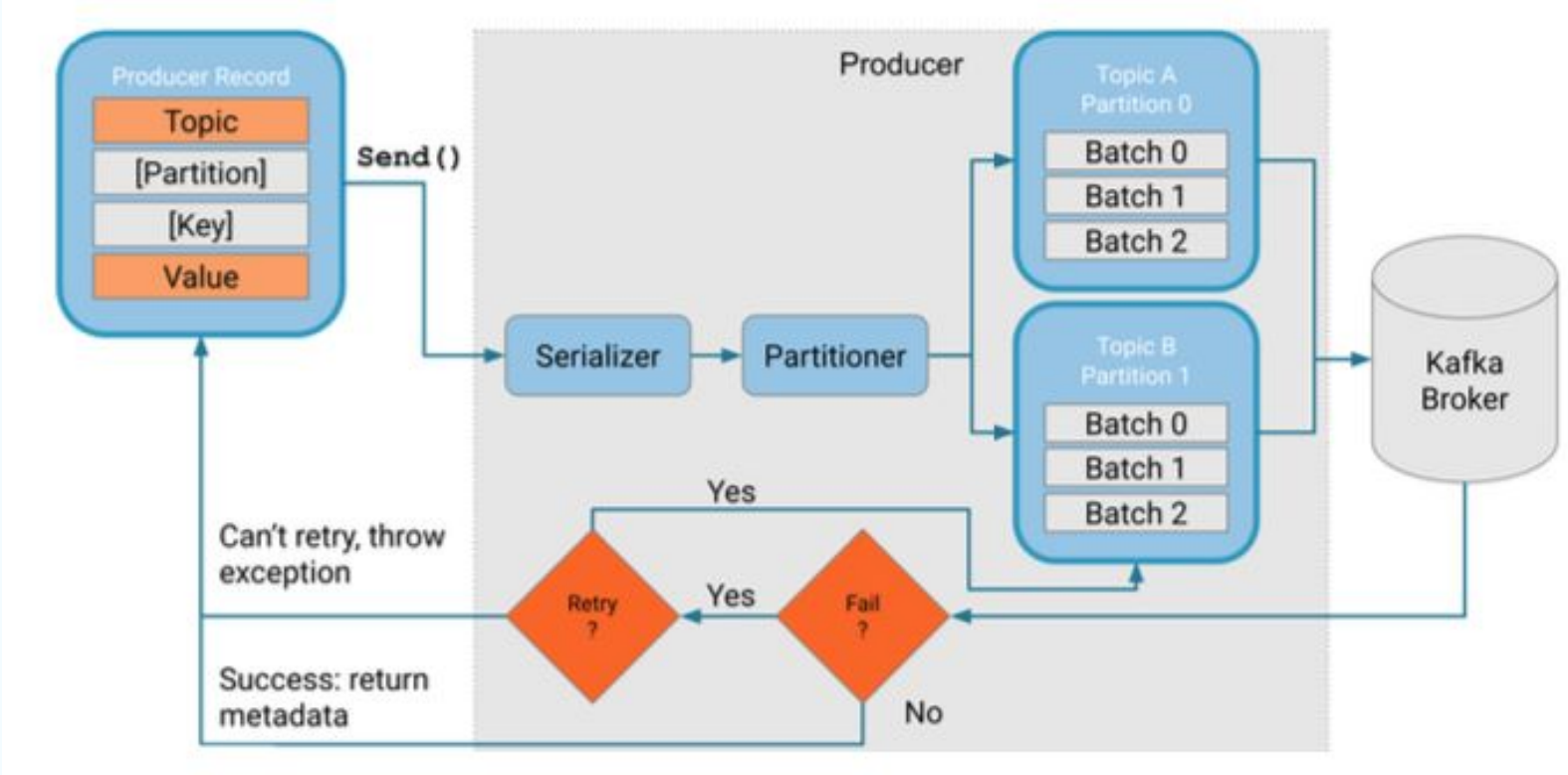
키가 없는 경우 메시지는 라운드 로빈 방식으로 파티션이 배정되어 생산됨

Produce to Kafka - 키가 있는 경우

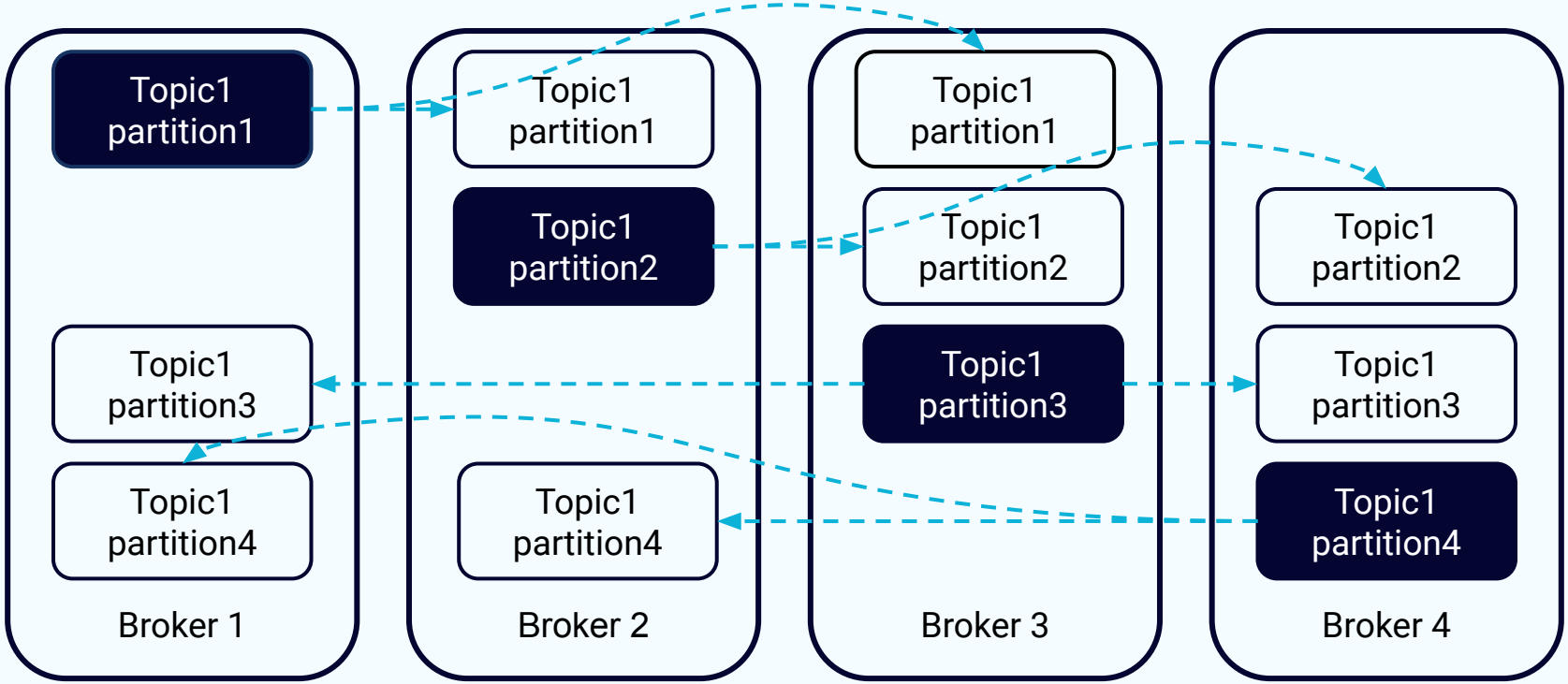


기본 파티셔너 : partition 위치 = hash(key) % 파티션 수

Producer Design



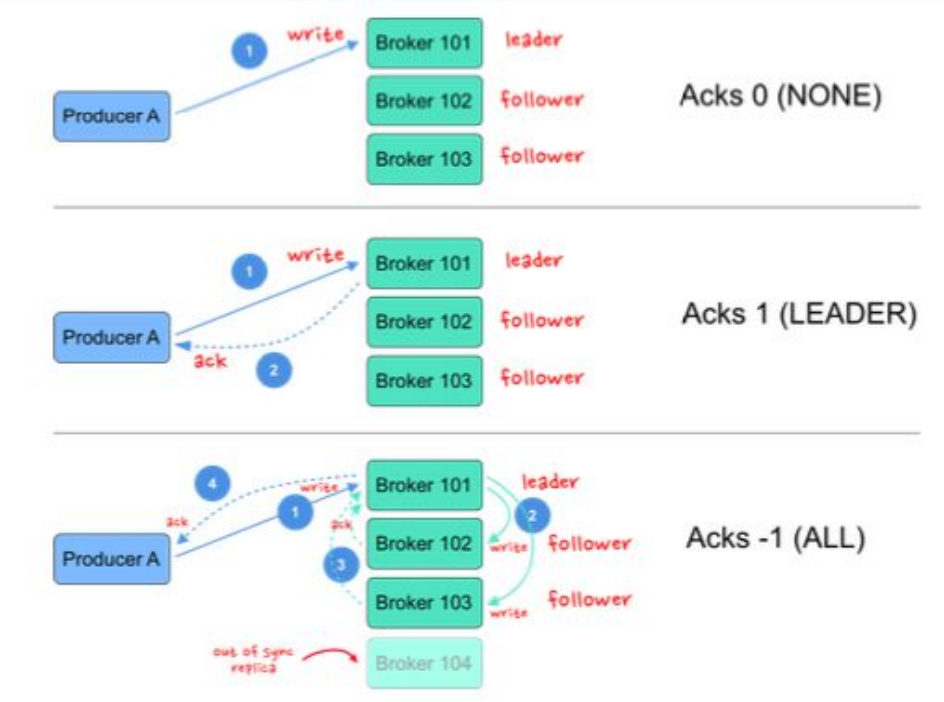
Apache Kafka - scale out and failover



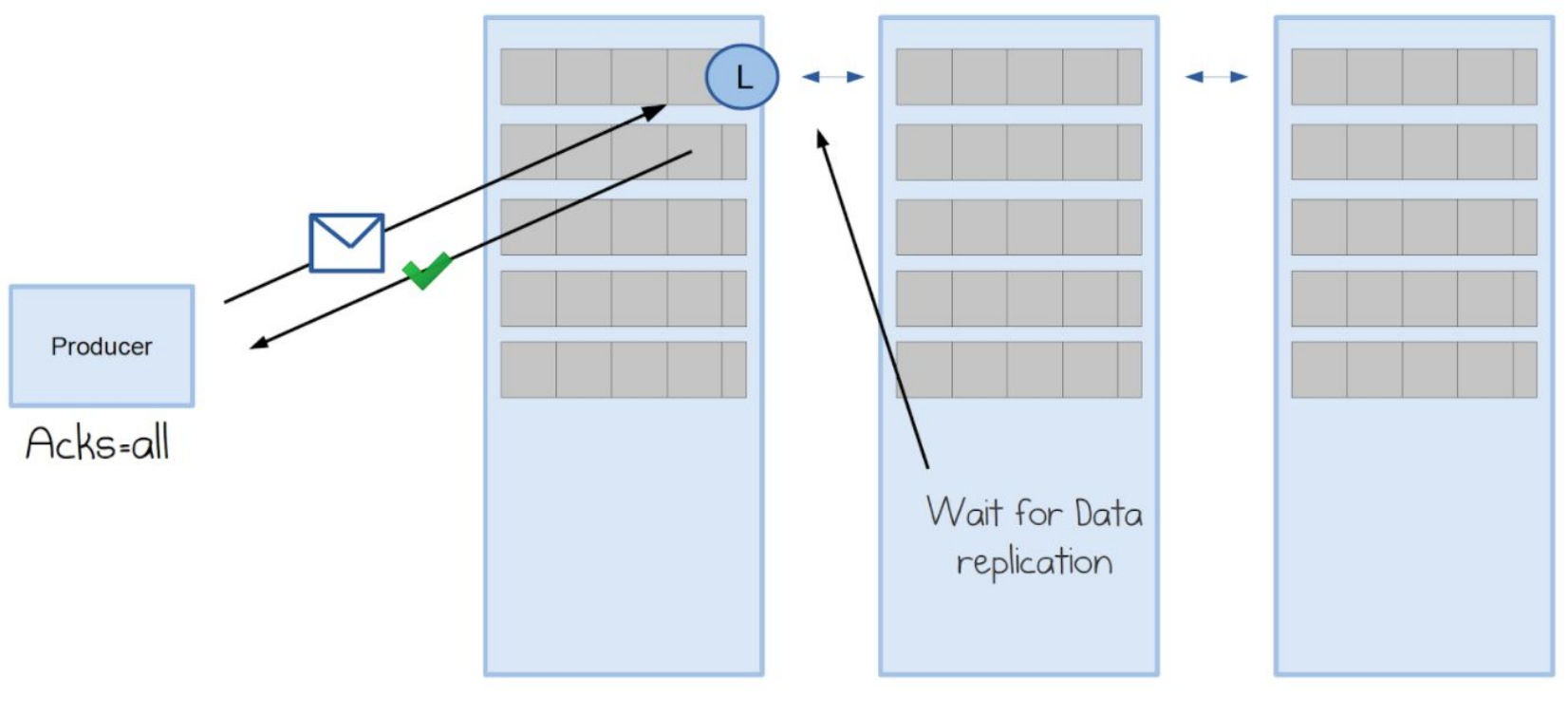
Data Durability

acks=all

- “The leader will wait for the full set of in-sync replicas to acknowledge the record.”
- Details in docs



Replication before acknowledgement



Retries

Recommendation

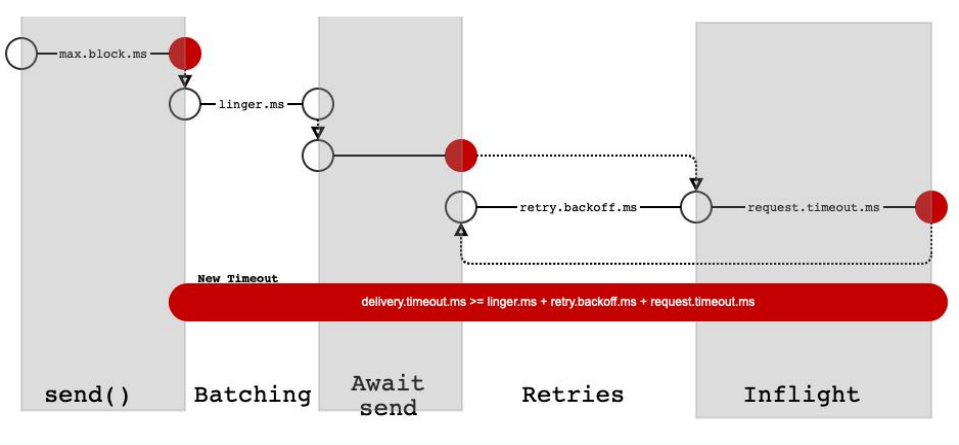
- Use built-in retries

Consider

- Using a high value for retries (e.g. the modern defaults of infinity)

Ensure

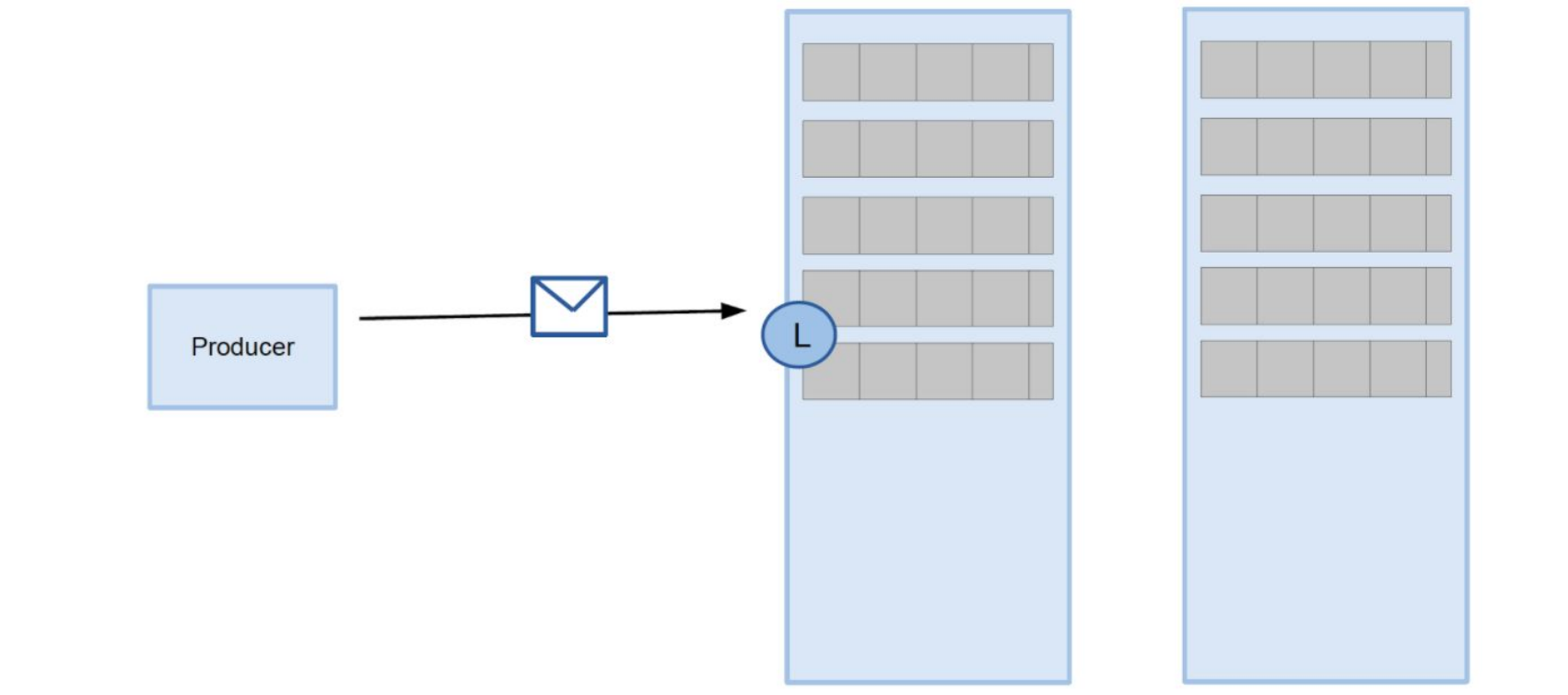
- Broker failure can be detected and resolved



delivery.timeout.ms

- Guaranteed upper bound
- Records either get sent, fail or expire from the point when send() returns

What happens when there is an issue?



enable.idempotence

Business event duplicates?

- enable.idempotence does not protect against the app sending a message twice
- No deduplication of business events

When set to 'true'

- Producer will ‘ensure’ that exactly one copy of each message is written.

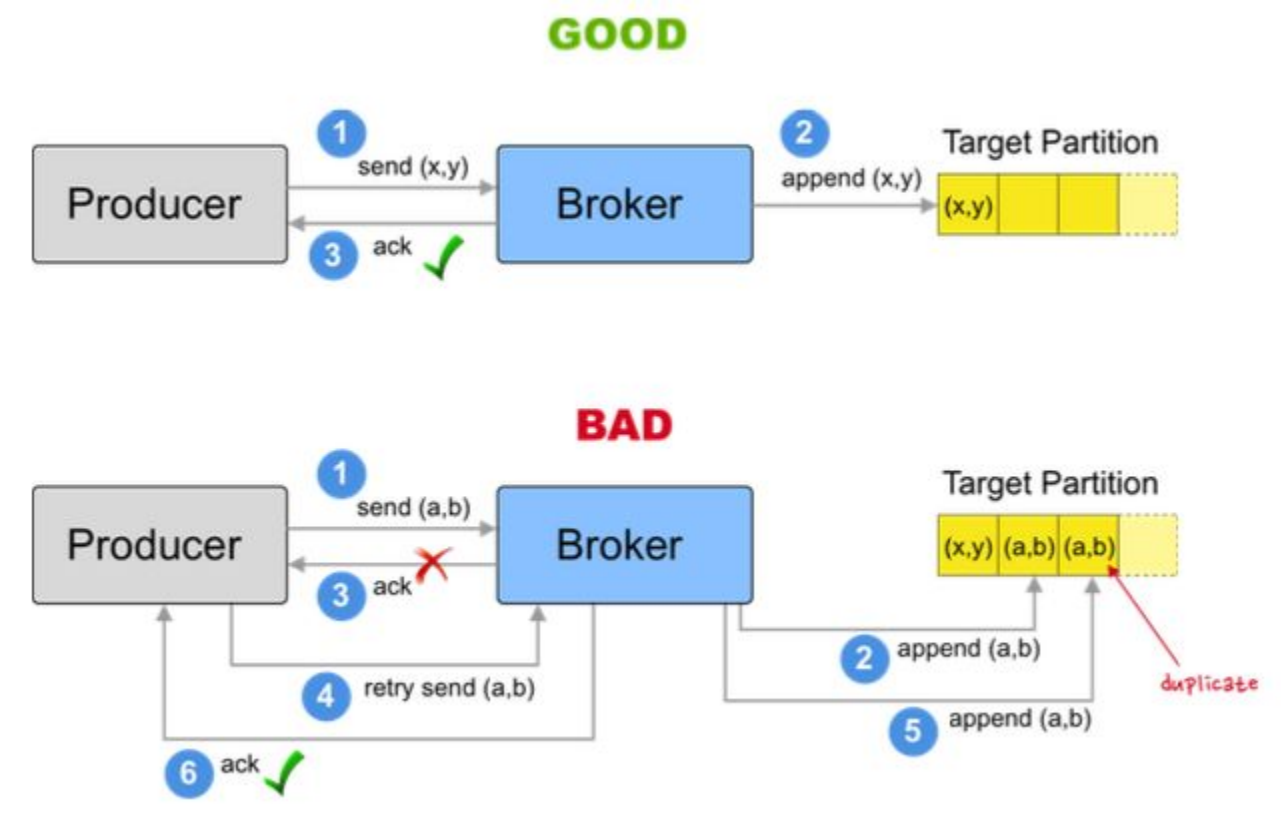
When set to 'false'

- Producer retries due to broker failures etc. may write duplicates of the retried message

Protects against duplicates from retires

- Default: true
- Details producer configurations
- Limits applicable values for other config parameters

Idempotent Producer



Consider configuring

Configuration is dependent on
- use case
- upstream setup and retry specification

Example
- Producer buffer limited to 32 MB (default)

If duplicates & ordering are not important
- retries=Integer.MAX_VALUE
- delivery.timeout.ms=Integer.MAX_VALUE
- max.inflight.requests.per.connection=5 (default)

If ordering is important, but duplicates don't matter
- retries=Integer.MAX_VALUE
- delivery.timeout.ms=Integer.MAX_VALUE
- max.inflight.requests.per.connection=1

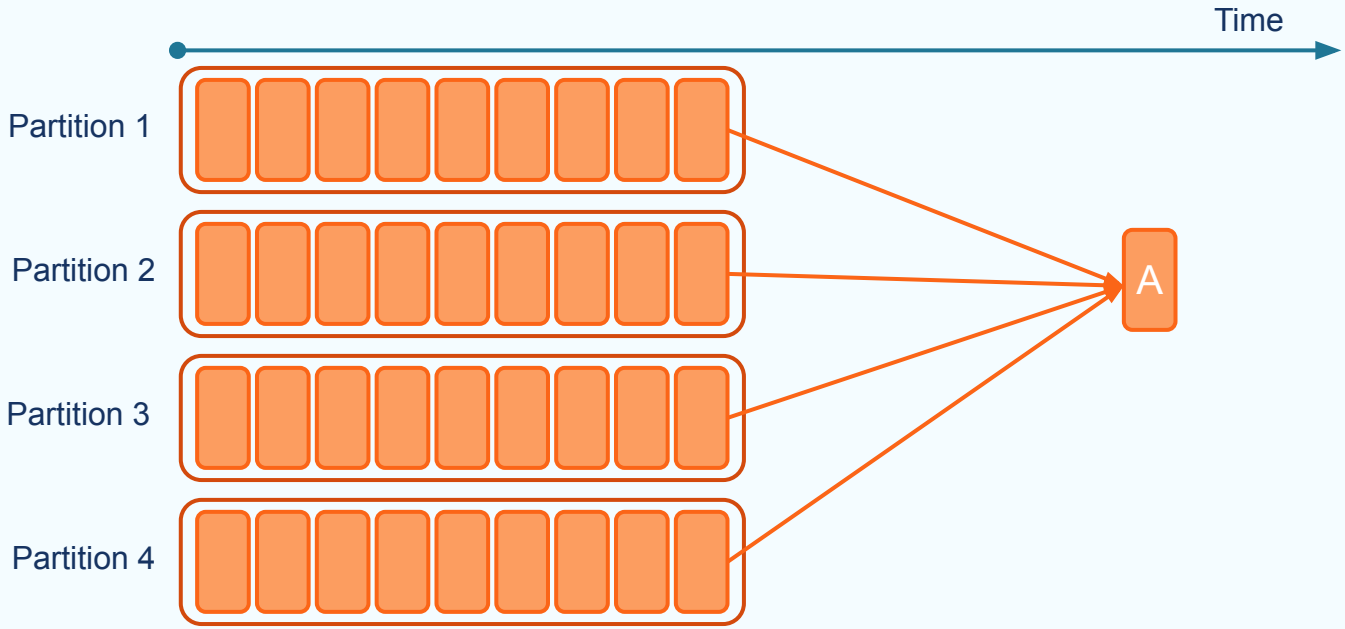
If duplicates & ordering are important
- retries=Integer.MAX_VALUE
- delivery.timeout.ms=Integer.MAX_VALUE
- enable.idempotence=true
- max.inflight.requests.per.connection=5 (required)

Consumer

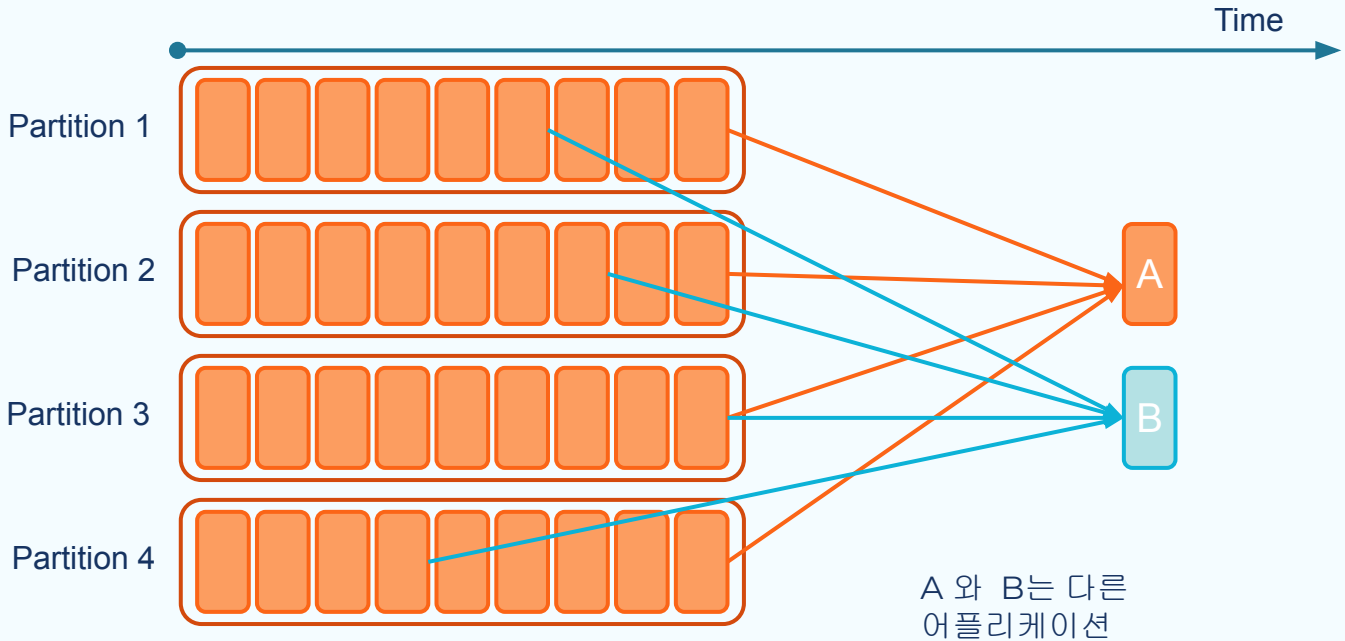
Consumer

- 컨슈머는 1개 이상의 토픽에서 메시지를 가져와 처리
- 새로 유입되는 메시지가 자동으로 검색됩니다.
- Consumer Offset
  - 마지막으로 읽은 메시지 추적
  - 내부의 특수 토픽에 저장됨 (__consumer_offsets)
- 클러스터에서 읽기 위한 CLI 도구 존재

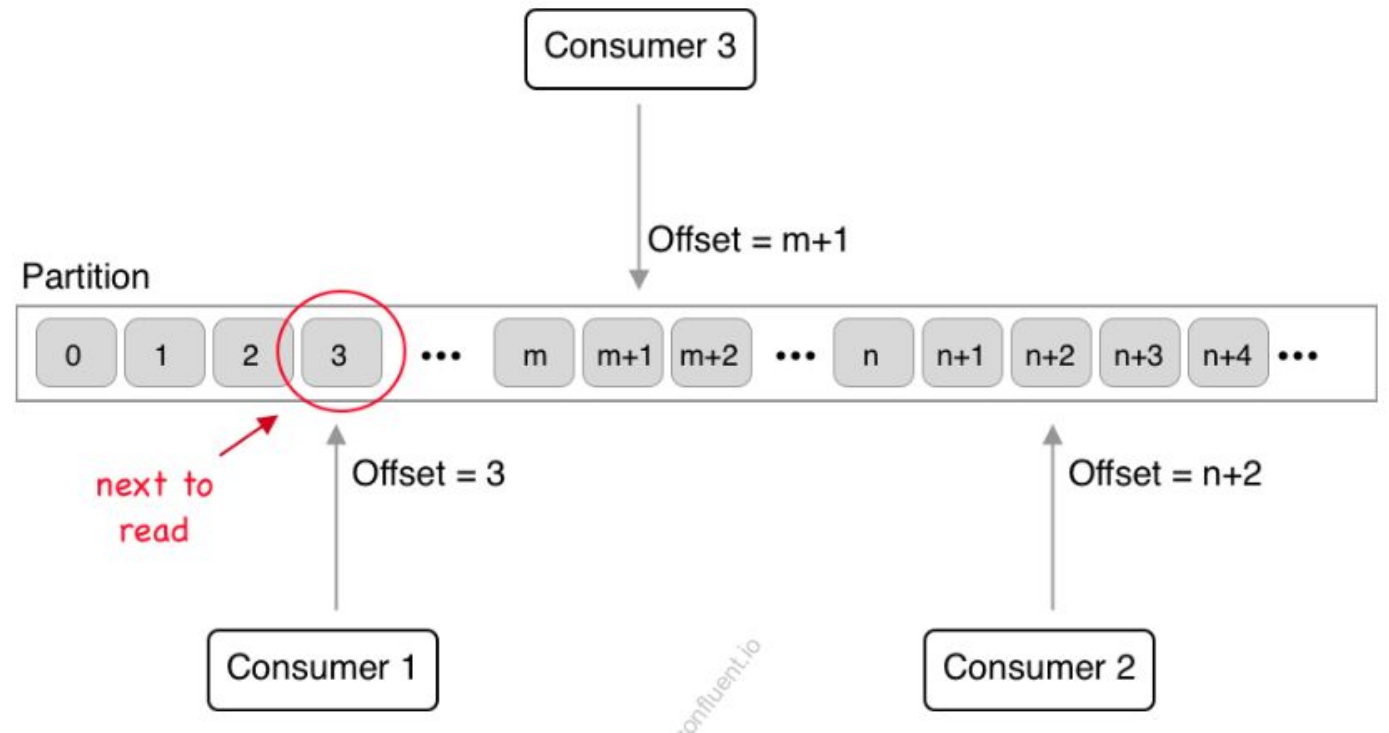
Consume from Kafka - 단일 Consumer



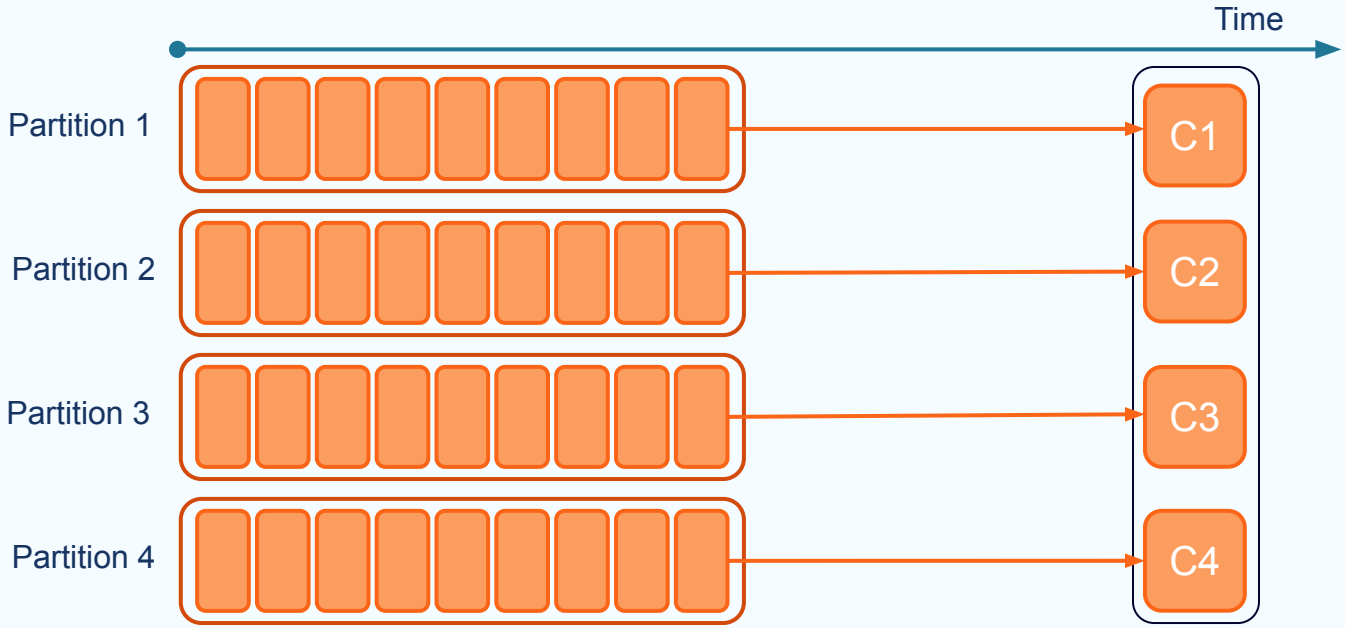
Consume from Kafka - 복수의 Consumers



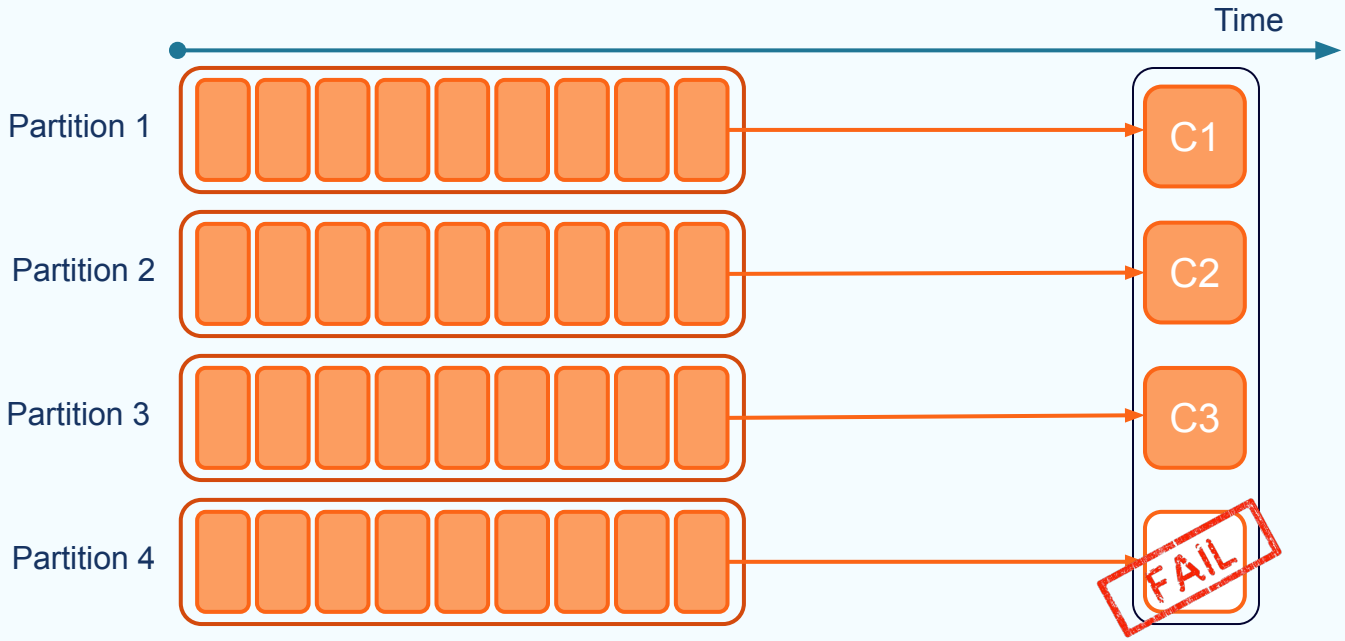
Consumer Offset



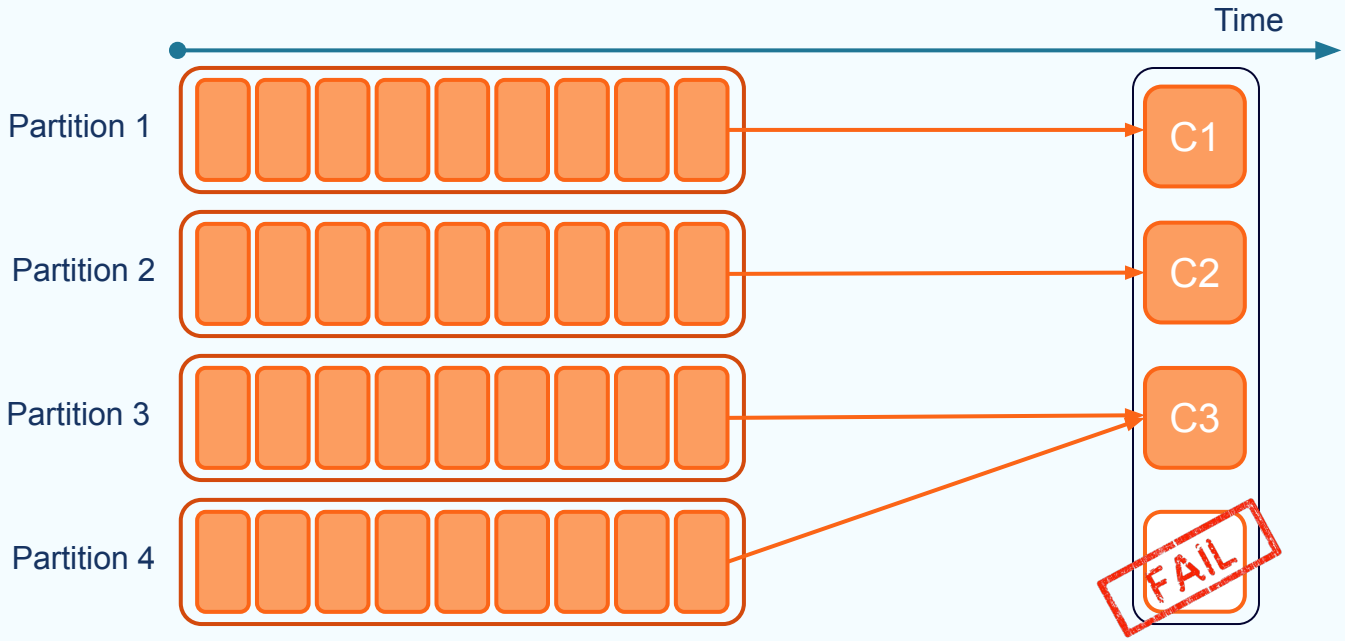
Consume from Kafka - Consumer 그룹



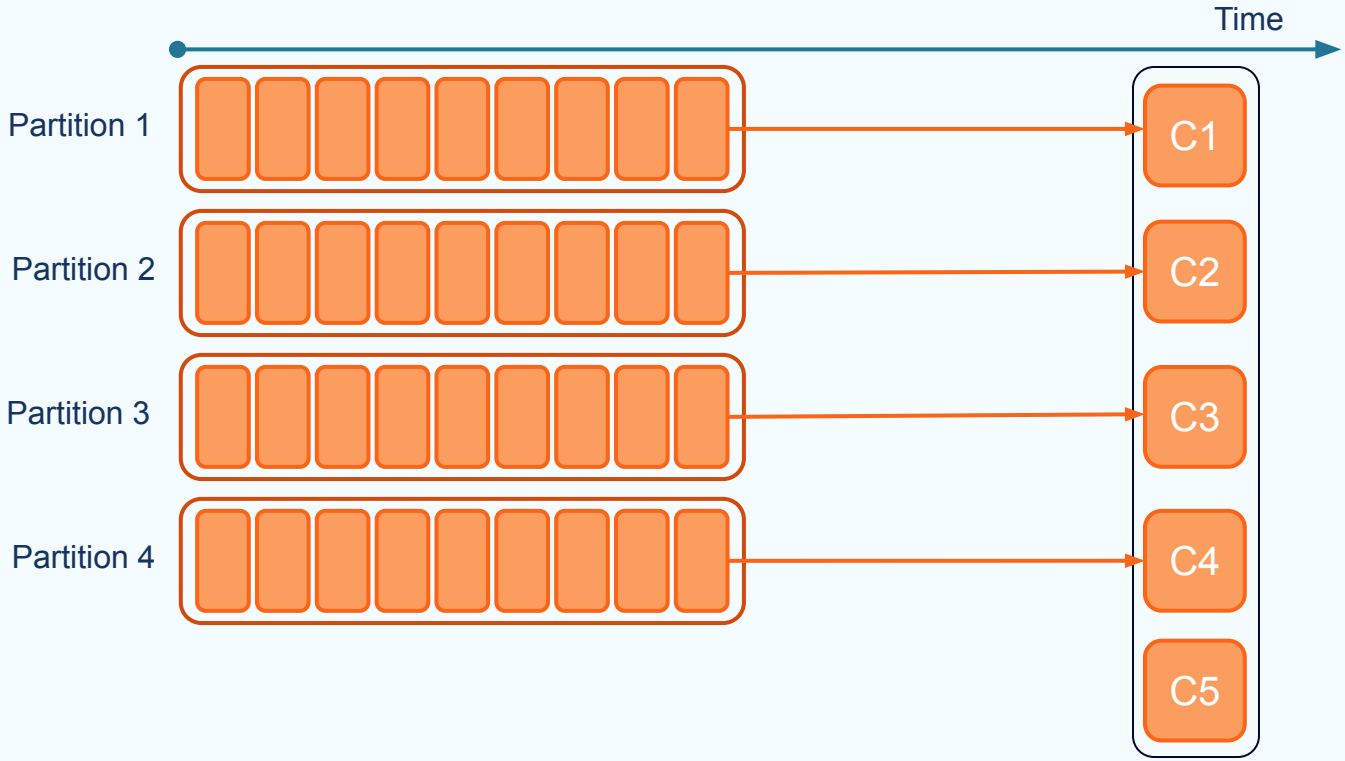
Consume from Kafka - Consumer 그룹 - 장애 발생



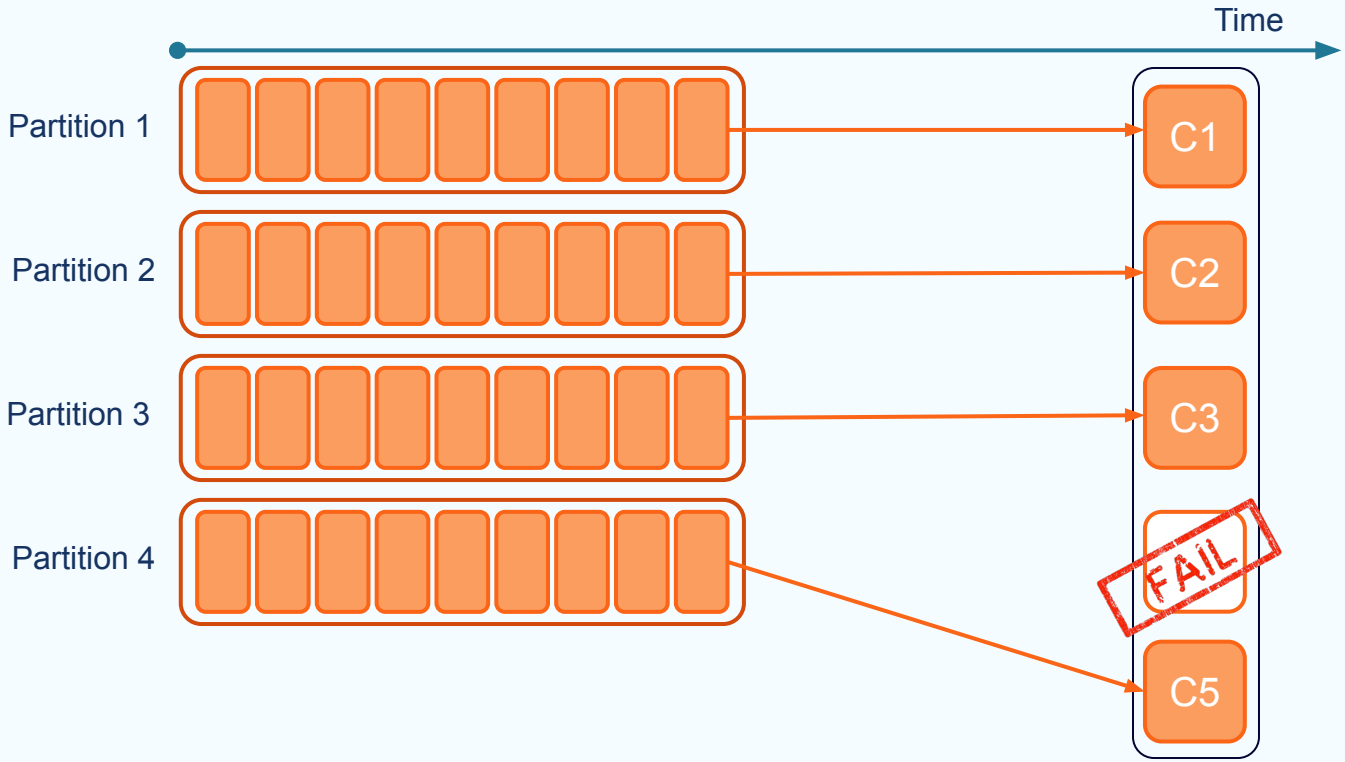
Consume from Kafka - Consumer 그룹 - 장애 발생



Consume from Kafka - Consumer 그룹

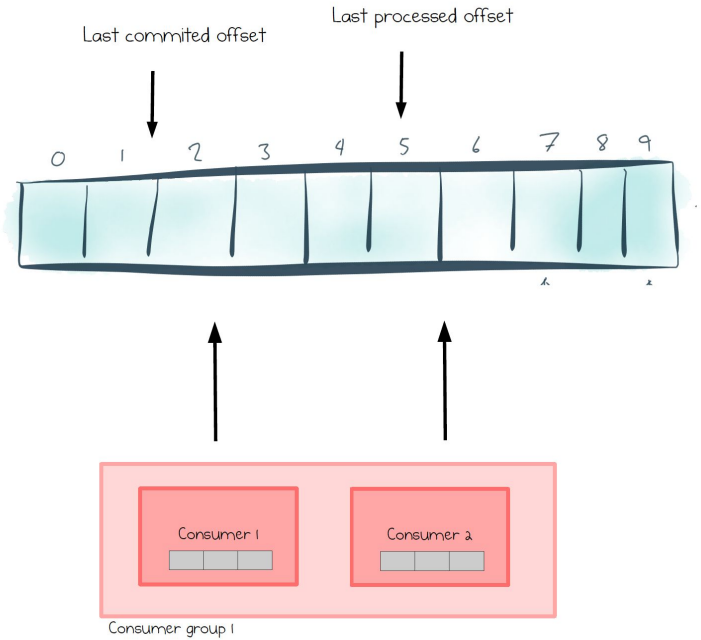


Consume from Kafka - Consumer 그룹



At-least-once semantics

Consumers offer at-least-once semantics



Consumers vs. Partitions

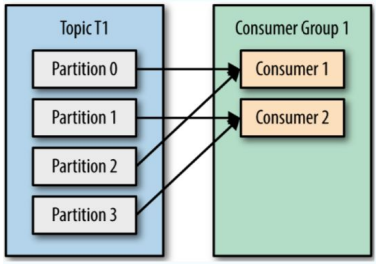
Partitions are the base for scalability, make sure to choose the correct number from beginning! (https://eventsizer.io can help)

No partition will be assigned to more than one consumer in the same group

Consumers pull **batches** from the broker

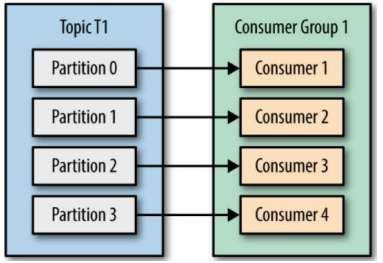
Consumers < Partitions

Consumers can be assigned multiple partitions



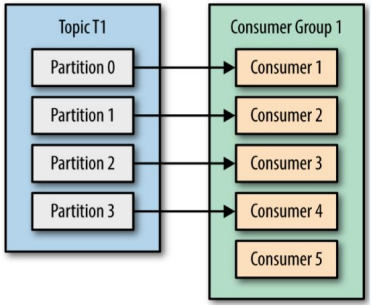
Consumers = Partitions

Consumers are assigned 1 partition each



Consumers > Partitions

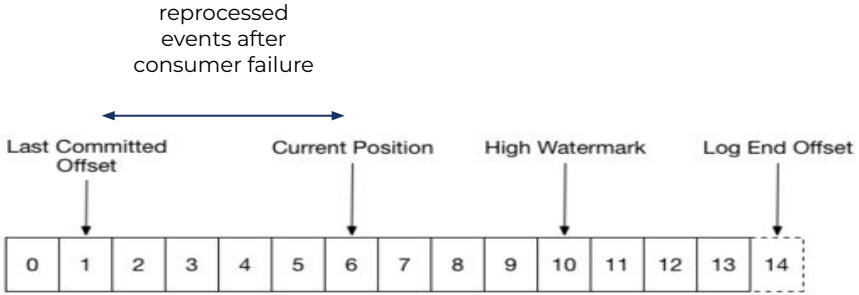
Consumers are assigned 1 partition each. Some consumers are idle.



Consumers Offsets

4 key positions in the partition

- **Log End Offset** - where the producer is (acks may provide back pressure to the producer)
- **High Watermark** - It can be consumed, i.e. leader has replicated the partition to all in sync replicas
- **Current Position** - Current Consumer offset
- **Last Committed Offset** - Last position written by the consumer in the __consumer_offsets topic



Manual Commits

Manually committing aggressively

- Helps to reduce the gap between current position and last committed offset
- But
  - adds a huge workload on Apache Kafka
  - does not provide exactly-once semantics

Consumers offer at-least-once semantics

- Details Consumer Group Offset Management

Commit

```
…
.commitAsync(…)
.commitSync(…)
.committed(…)
.pause(…)
.resume(…)
.seek(…)
…
```

Manually committing can add a huge workload on Apache Kafka

Use manual commit methods judiciously

Best Practice Example:

Custom Retriable consumer and External Service

jeanlouisboudart/retriable-consumer

Kafka for the Enterprise

Complete: Confluent를 통해 Apache Kafka를 완성

개발자	운영자	아키텍트	관리자
제한이 없는 개발자 생산성 제공	**다양한 스케일링 상황에서 효과적인 운영**	**운영 단계를 위한 전제 조건**	**비즈니스 성공을 위한 파트너쉽**
다양한 개발언어를 통한 개발 수행 Non-Java Clients | REST Proxy Admin REST APIs	**관리 & 모니터링** Cloud Data Flow● | Metrics API● Control Center● | Health+●	**엔터프라이즈 레벨의 보안** RBAC | BYOK● | Private Networking● Encryption | Audit Logs	**완전한 참여 모델**
풍부한 Pre-built 에코 시스템 Connectors | Hub | Schema Registry	**유연한 DevOps 자동화 제공** Confluent for K8s● | Ansible Playbooks● Marketplace Availability●	**데이터 호환성** Schema Registry | Schema Validation	**매출 / 비용 / 위험 관리**
스트리밍 데이터베이스 ksqlDB	**동적인 성능 & 유연성 제공** Elastic Scaling● | Infinite Storage● Self-Balancing Clusters● | Tiered Storage●	**Global 탄력성** Multi AZ Clusters● | 99.95% SLA● | Replicator Multi-Region Clusters● | Cluster Linking	**TCO / ROI**

Apache Kafka

Cloud service ●
Software ●

Fully Managed 클라우드 서비스

Availability Everywhere

Self-managed 소프트웨어

Enterprise Support

Professional Services

Committer-driven Expertise

Training

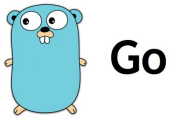
Partners

개발자들이 다양한 방법으로 Kafka를 사용할 수 있도록 지원

개발자들이 가장 생산적으로 사용할 수 있는 언어를 사용하여 개발할수 있도록 하며, 다양한 개발언어가 필요한 아키텍처, 구현 사례 등을 선택하여 구축할 수 있도록 함

Confluent 클라이언트

실전에서 테스트 되고 고성능 지원



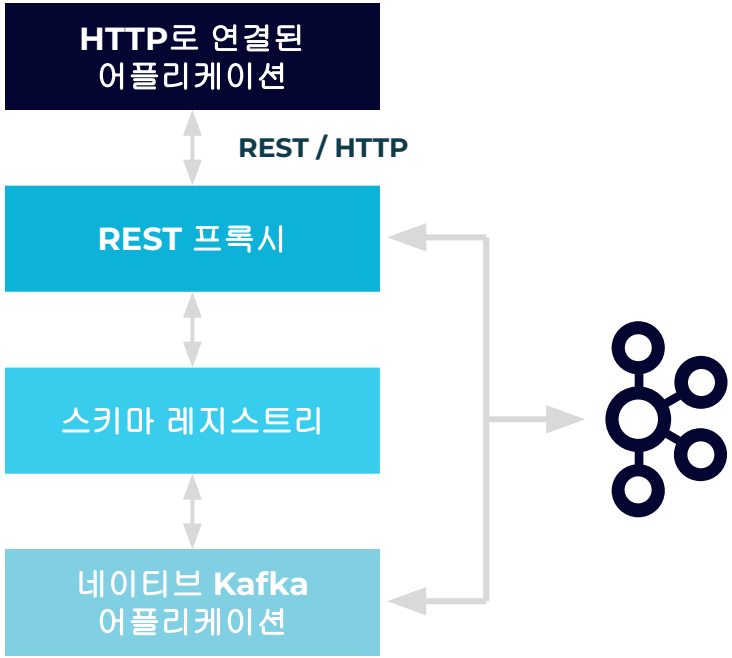
*어떤 유형의 어플리케이션도 **Kafka**와 연결 가능*

어플리케이션들이 Kafka와 연결하기 위한 방식으로 RESTful 인터페이스 제공

HTTP로 연결가능한 어떠한 장치에서도 통신이 가능

third-party 어플리케이션들이 메시지를 produce하고 consume할 수 있도록 함

REST 프록시



Kafka 커넥트

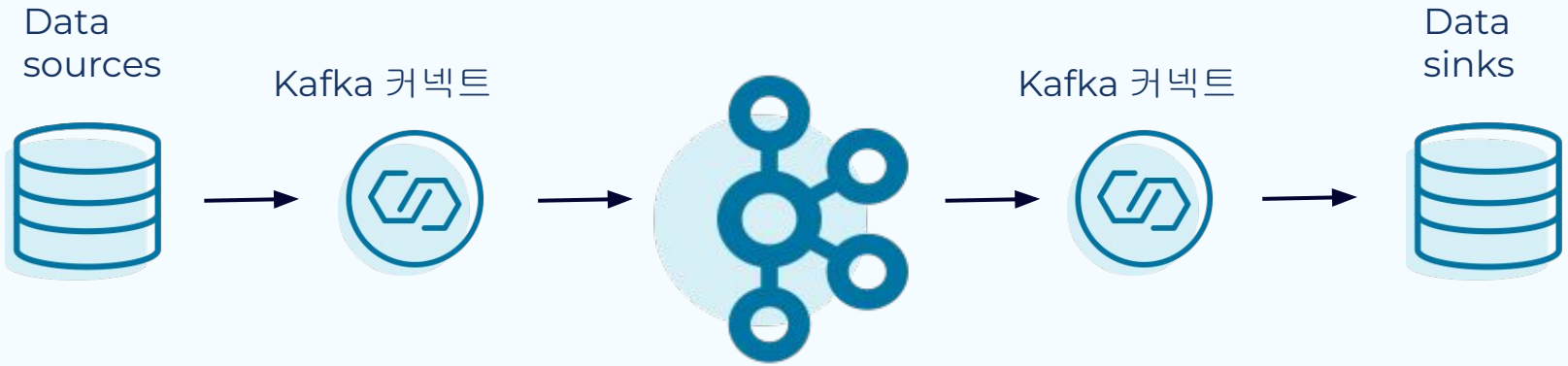
Kafka Connect는 Apache Kafka와 다양한 데이터 시스템 간에 데이터를 확장하고 안정적으로 스트리밍하기 위한 도구입니다. 이를 통해 대규모 데이터를 Kafka 안팎으로 전달하는 커넥터를 빠르게 정의할 수 있습니다. Kafka Connect는 전체 데이터베이스를 수집하거나 모든 애플리케이션 서버의 지표를 Kafka 토픽으로 수집하여 데이터를 빠르게 스트림 처리에 사용할 수 있도록 합니다. Sink 커넥터는 오프라인 분석을 위해 Kafka 토픽의 데이터를 Elasticsearch와 같은 보조 저장소나 Hadoop과 같은 배치 시스템으로 전달할 수 있습니다.

Kafka Connect는 테스트 및 일회성 작업을 위한 Standalone 프로세스로 실행할 수 있습니다. 또한 전사를 지원하는 확장 가능한 분산형 서비스로 실행할 수 있습니다.
이를 통해 개발 및 테스트를 위한 작은 규모의 환경을 구성하고, 이를통해 진입 장벽 및 운영 오버헤드가 낮은 소규모 운영환경을 구축할 수 있으며 이를 바탕으로 대규모 조직의 데이터 파이프라인을 지원하도록 확장할 수 있습니다.

Kafka 커넥트

코드가 필요없이 알려진 시스템들(데이터베이스, object storage, 큐 등)을 Apache Kafka 연결할 수 있습니다.

커스텀 변환을 위해 약간의 코드를 작성할 필요가 있을 수 있으나, 이미 작성되어 제공되는 SMT(Single Message Transforms) 이나 컨버터를 사용할 수도 있습니다.



데이터 품질 과 연속성 보장

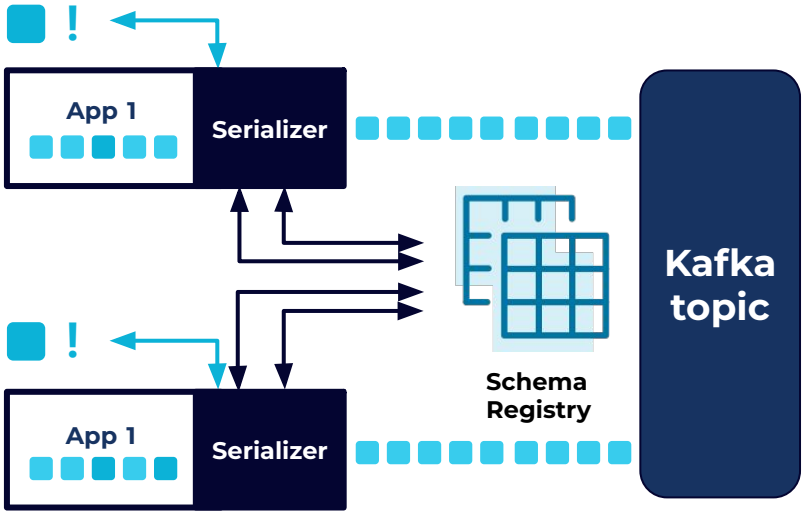
표준 스키마를 사용하여 개발

- **Store and share :** 버전으로 관리하는 모든 표준 스키마를 저장하고 공유
- **Validate data compatibility :** 클라이언트 레벨에서의 데이터 호환성 검증

관리적 복잡성 감소

- **Avoid time-consuming** : 표준 스키마를 모든 개발자들이 함께 사용하여 불필요한 시간 낭비를 줄일 수 있음

스키마 레지스트리



AVRO protobuf Protocol Buffers {JSON}

*데이터 호환성*을 *프로그램을 통해 보장*

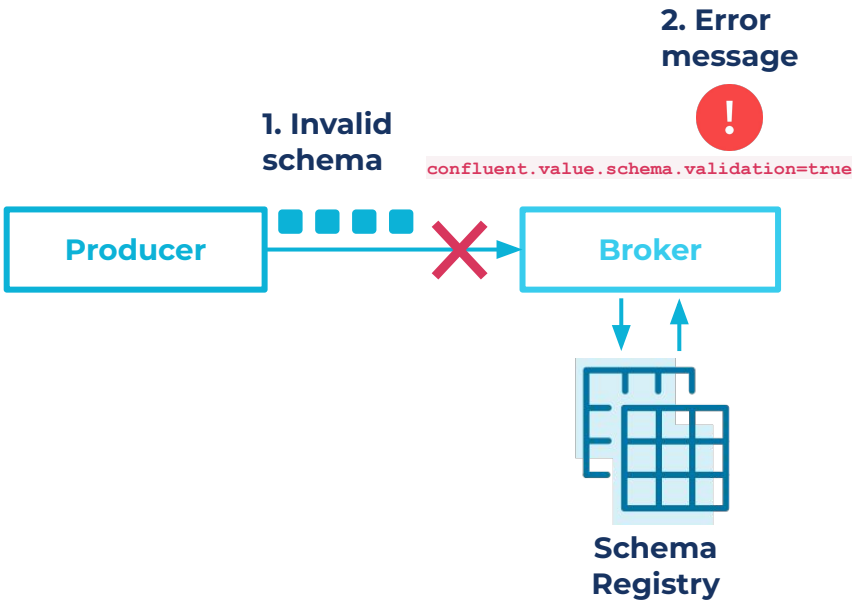
안정적인 스키마 확장

- 자동으로 브로커에서 스키마 검증 시행
- 브로터에서 직접적으로 Confluent 스키마 레지스트리와 연계

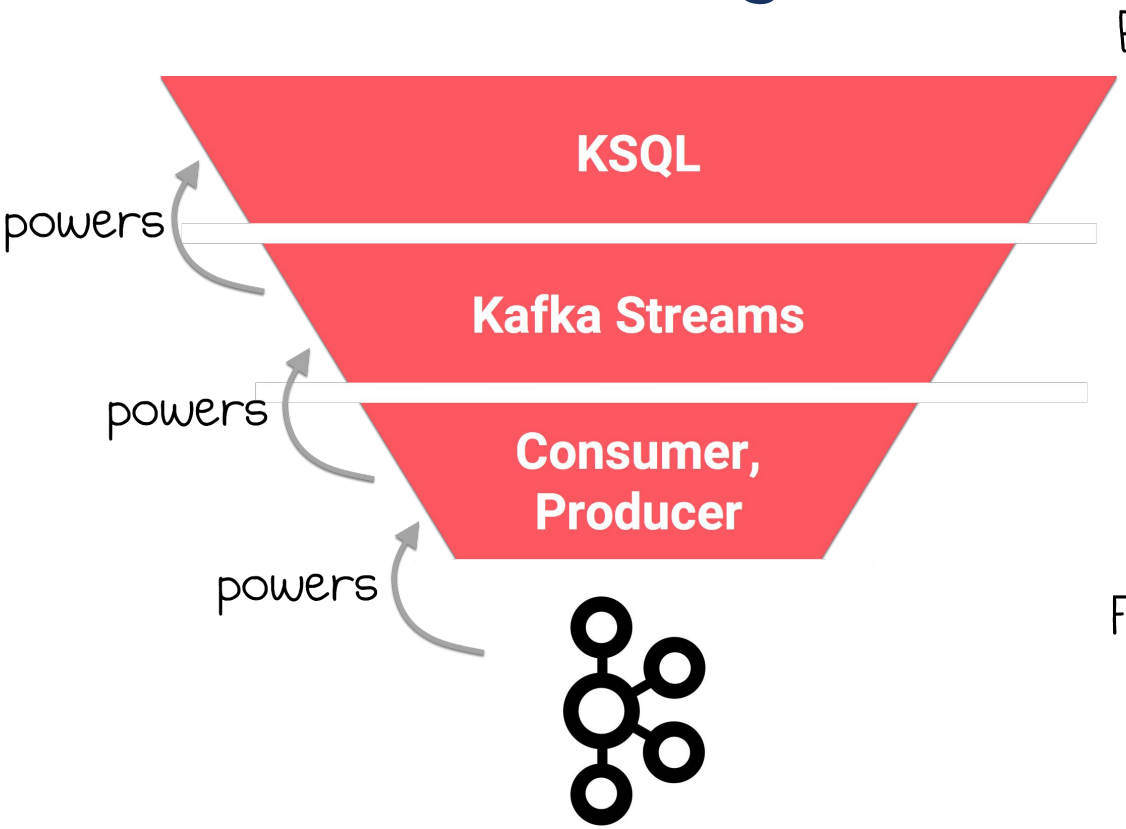
데이터 품질에 대한 개별 제어

- 토픽 레벨의 검증 가능

스키마 검증



Shoulders of Streaming Giants



Ease of Use

CREATE STREAM, CREATE TABLE,
SELECT, JOIN, GROUP BY, SUM, …

KSQL UDFs

KStream, KTable, filter(), map(),
flatMap(), join(), aggregate(),
transform(), …

subscribe(), poll(), send(),
flush(), beginTransaction(), …

Flexibility

Kafka Streams
Scalable Stream Processing

Kafka Streams Java 라이브러리로 확장 가능하고 내구성 있는 스트림 처리 서비스 구축

- **간단한** functional API
- **강력한** Processing API
- **프레임워크 필요없는** 라이브러리, 다른 JVM 라이브러리와 마찬가지로 개발 배포

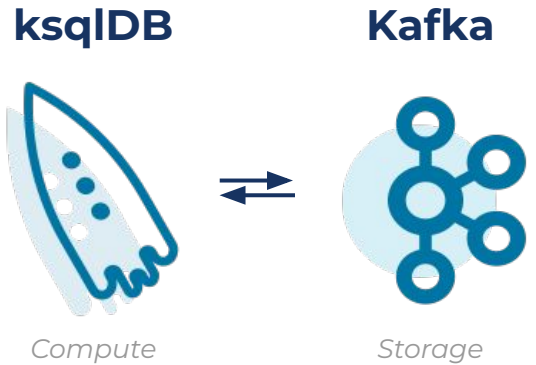
```
builder.stream(inputTopic)
 .map((k, v) ->
    new KeyValue<>(
      (String) v.getAccountId(),
      (Integer) v.getTotalValue())
    )
.groupByKey()
.count()
.toStream().to(outputTopic);
```

Easily Build Real-Time Applications

ksqlDB at a Glance

ksqlDB is a database for building real-time applications that leverage stream processing

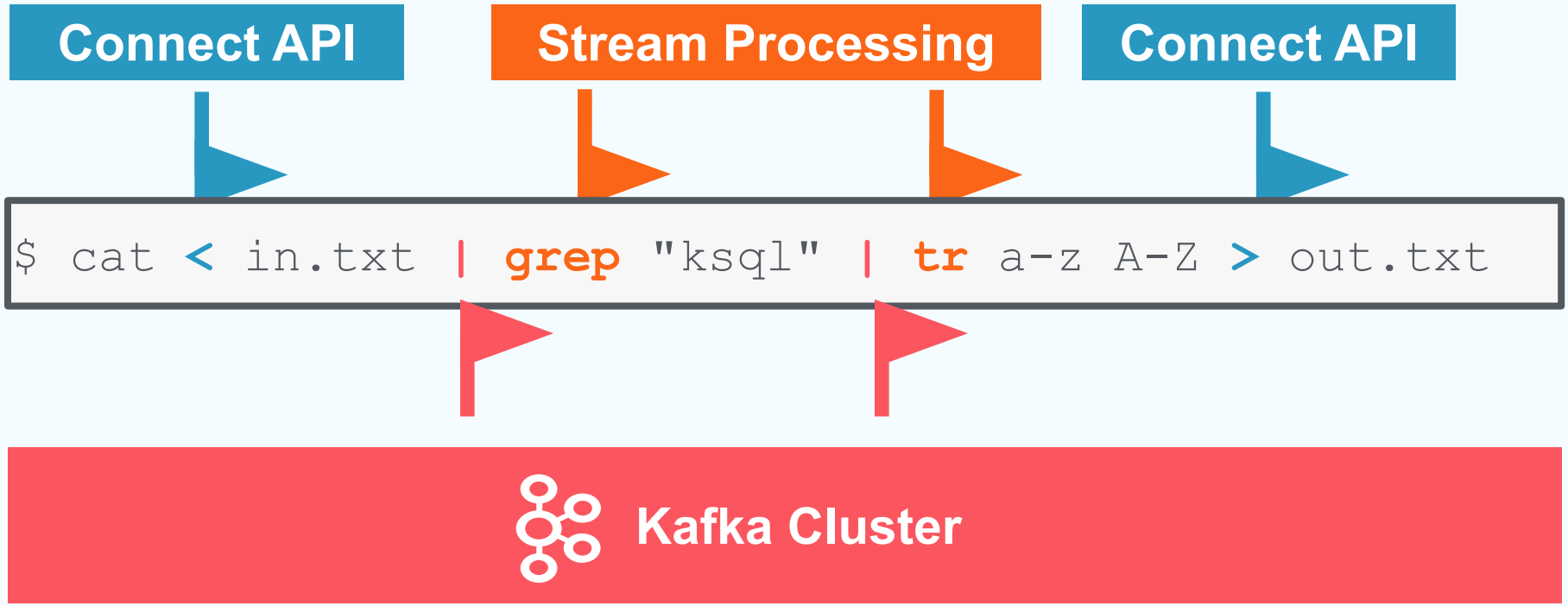
- Aggregations
- Joins
- Filters
- User-Defined Functions
- Push & Pull Query Support
- Embedded Connectors



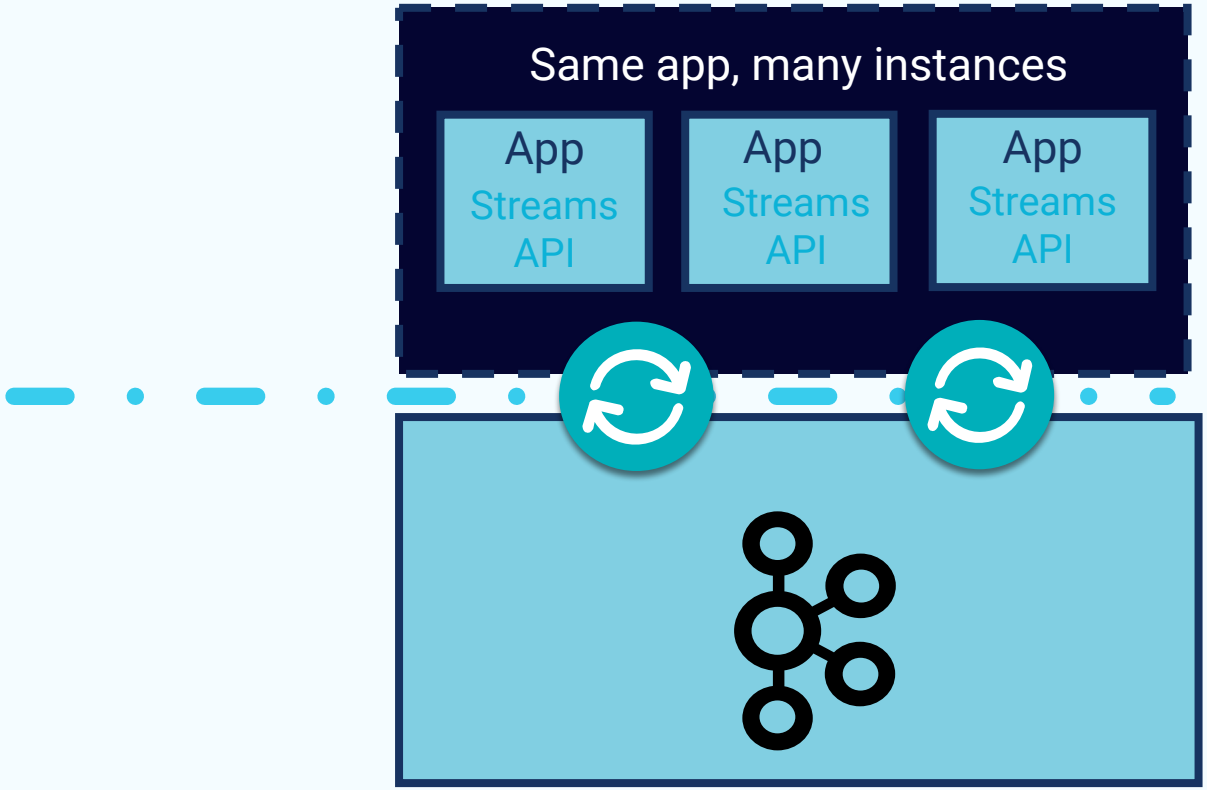

```
CREATE TABLE activePromotions AS
  SELECT rideId,
         qualifyPromotion(distanceToDst) AS promotion
  FROM locations
  GROUP BY rideId
  EMIT CHANGES
```

Build a complete real-time application with just a few SQL statements

Stream Processing by Analogy

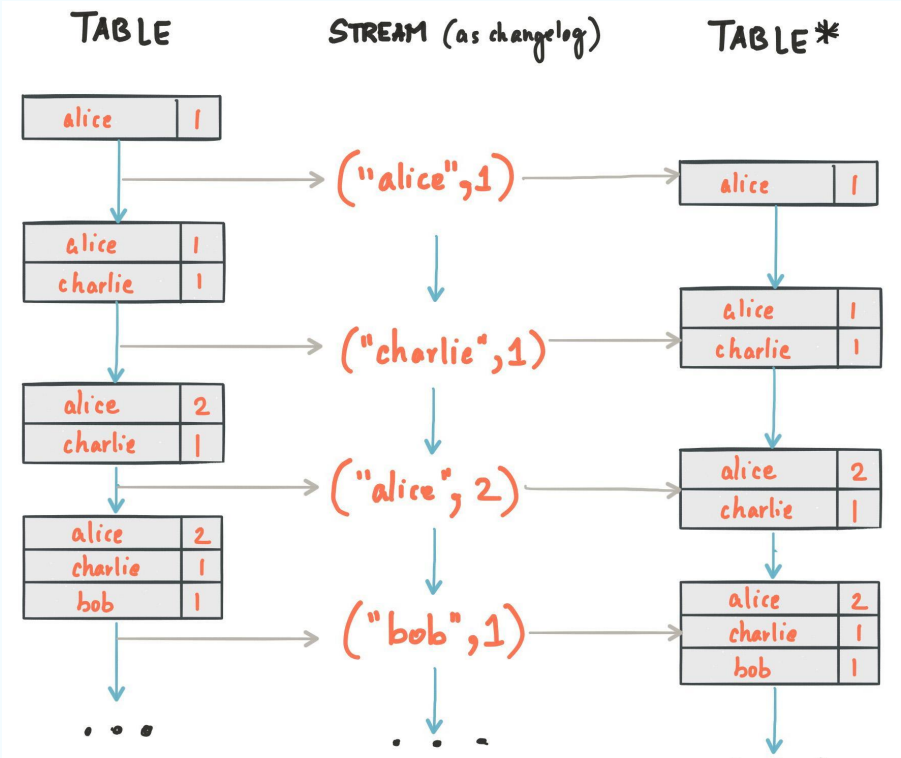


Where does the processing code run?



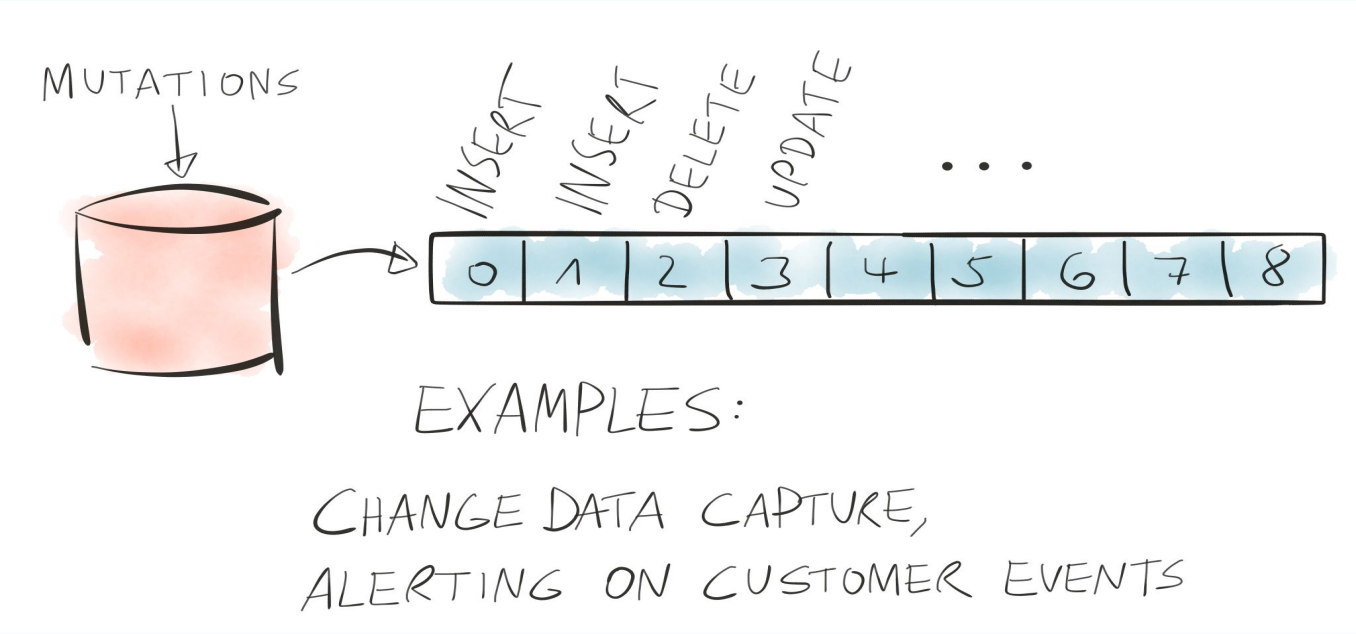
Brokers?
Nope!

Stream <-> Table duality

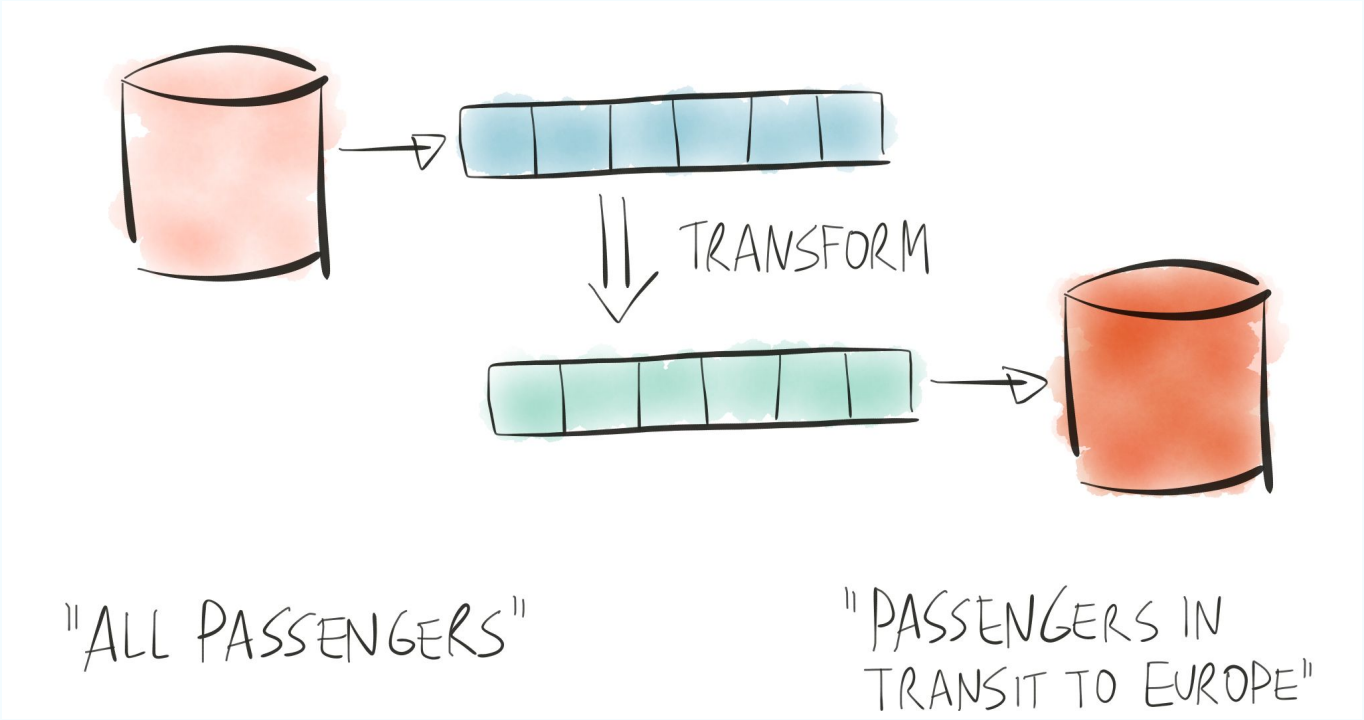


http://www.confluent.io/blog/introducing-kafka-streams-stream-processing-made-simple
http://docs.confluent.io/current/streams/concepts.html#duality-of-streams-and-tables

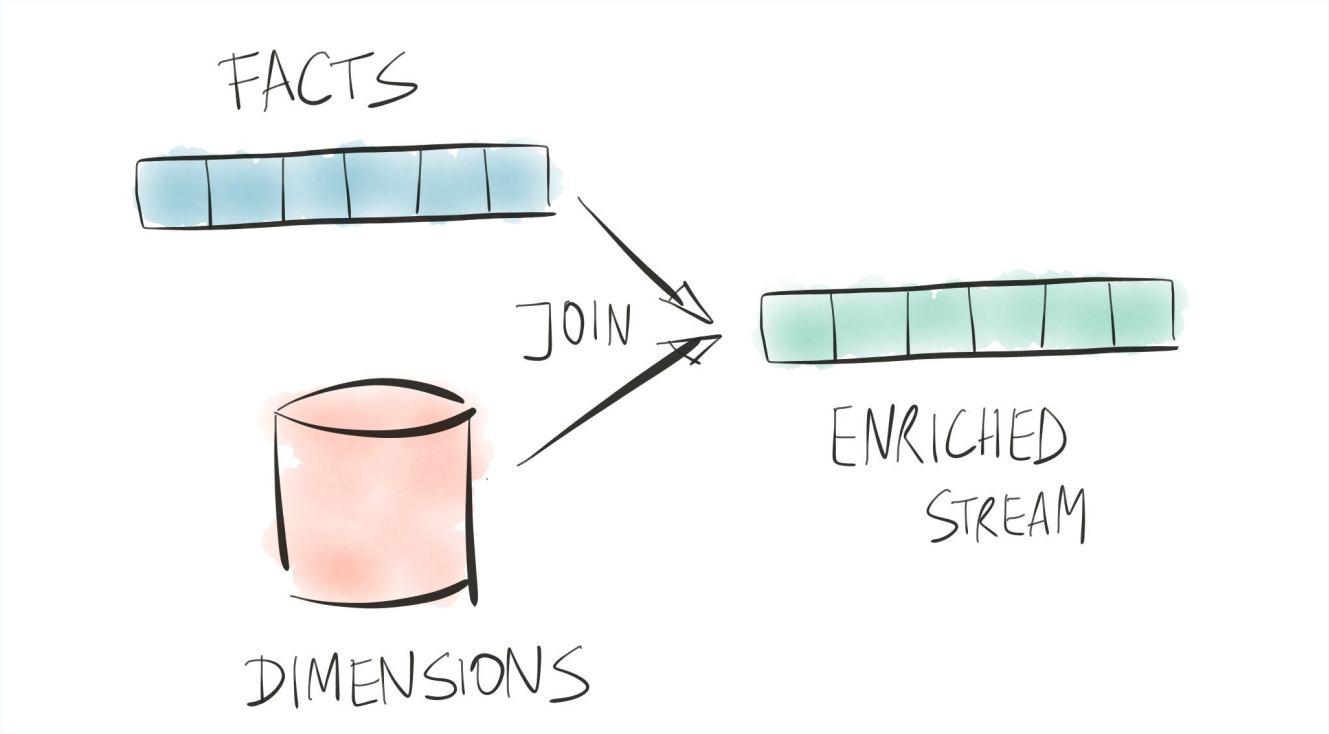
Change Data Capture as an input



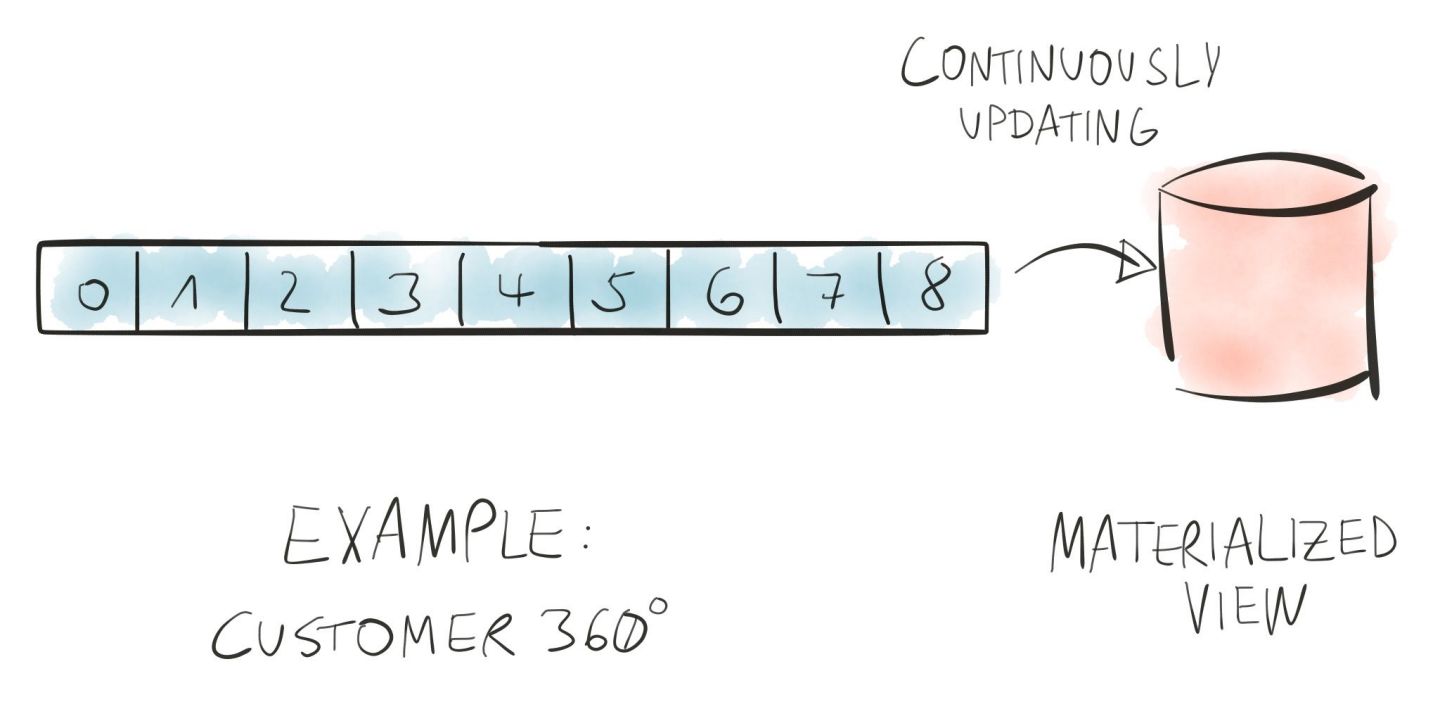
Processing with Kafka Streams



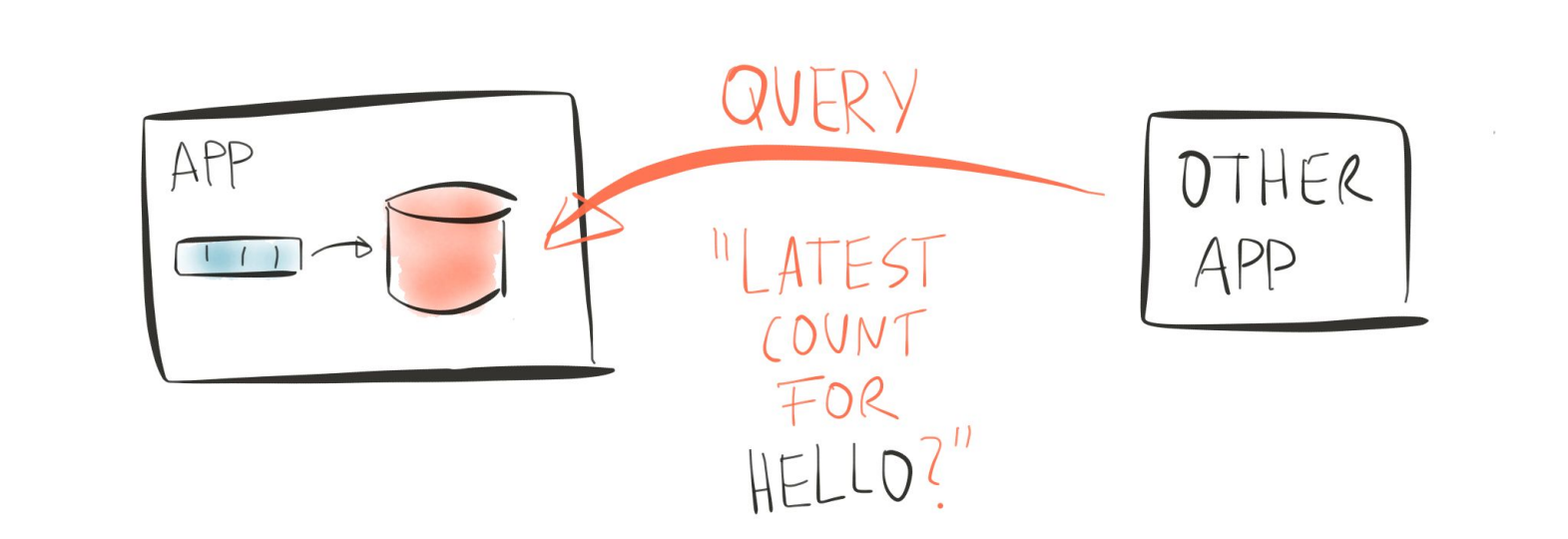
Enrichment with Kafka Streams



Materialized Views as results



Or use Interactive Queries



End to End data pipelines