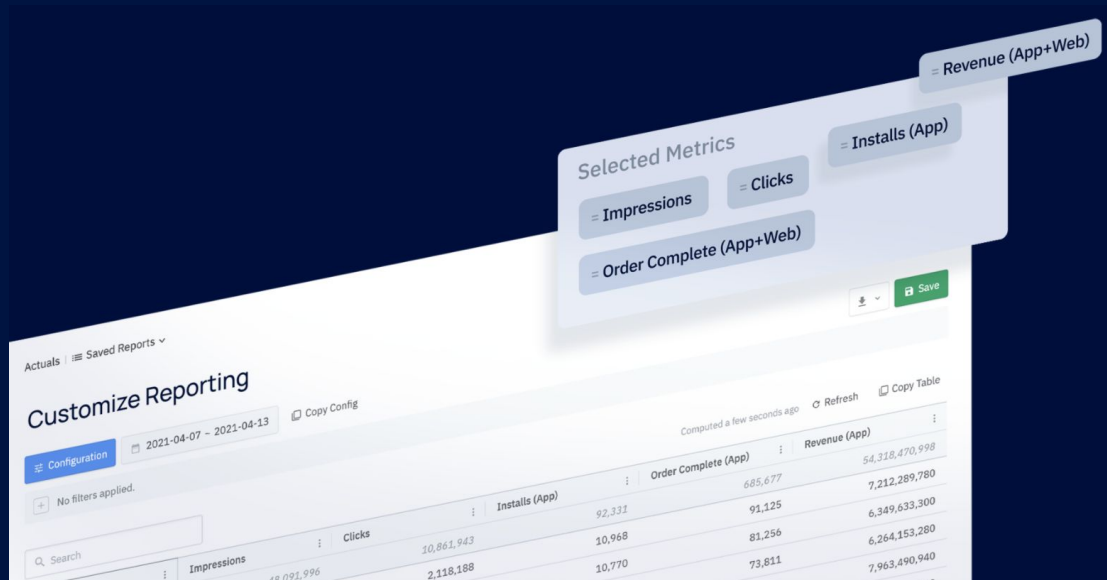


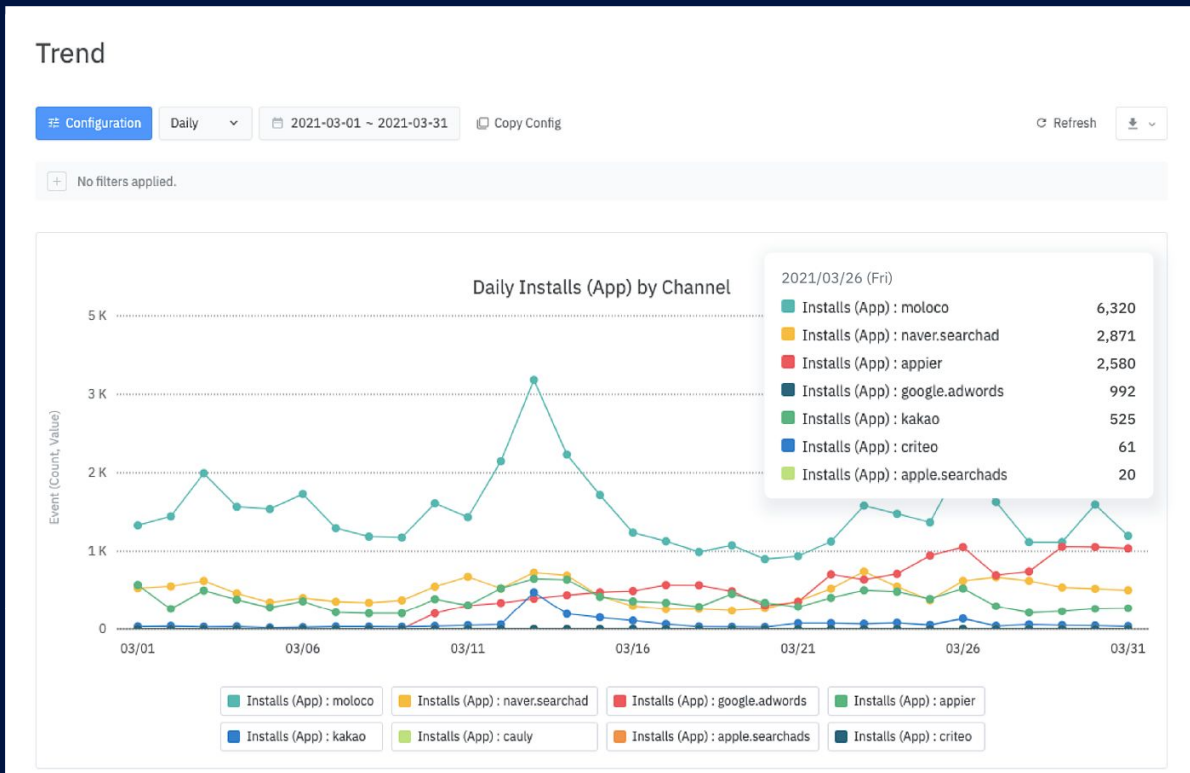
Lag 없는 실시간 데이터 파이프라인을 위한 아키텍처 개선기

Overview

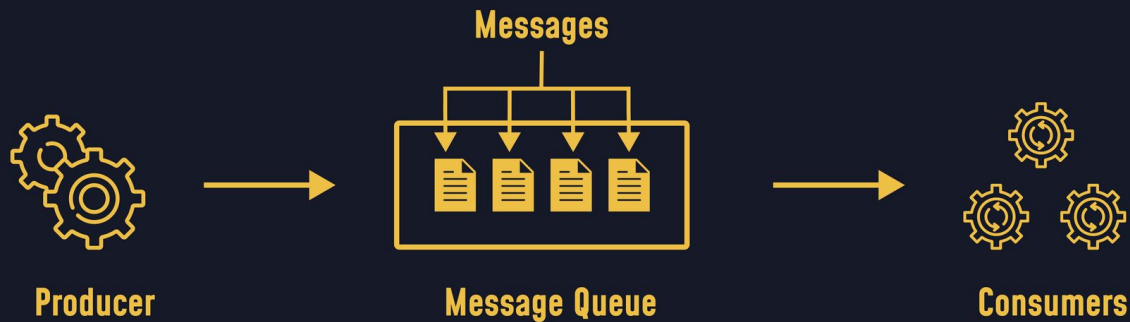
Daily 1B 이벤트를 수집, 처리, 제공하는 데이터 파이프라인 운영



Daily 1B 이벤트를 실시간 수집, 처리, 제공하는 데이터 파이프라인 운영



수집, 처리 사이에 Message Queue를 뒀야 한다는 것은 당연한데
그 다음은 뭘까요?





Mathias Verraes

@mathiasverraes



There are only two hard problems in distributed systems: 2. Exactly-once delivery 1. Guaranteed order of messages 2. Exactly-once delivery

3:40 AM · Aug 15, 2015 · Twitter for Android

Airbridge 서비스와 Workload 소개
기존 아키텍처의 문제점
새로운 아키텍처 아이디어
Consumer Decoupled Architecture
경험한 어려움
새 아키텍처 적용 후 결과
앞으로 더 시도해봐야 할 것


Airbridge 서비스와 Workload 소개

Attribution?

Search: 시디즈 T800HLDA

All Images Videos News Maps Shopping


Ads · Shop 시디즈 T800HLDA



T800HLDA/TN8
메쉬의자 2type

₩519,000
Free shipping
오늘의집

[View more](#)



시디즈 T80 시리즈 T800HLDA
메쉬의자 그레이

₩553,000
Free shipping
쿠팡

[View more](#)

Google is not a party to the product sale

쿠팡 Coupang

Sponsored ·

이 할인을 무엇..? 놓치면 후회할 역대급 할인
드디어 시작된 쿠팡 WOW DAY 최대 75% 빅 세일!
무료 배송과 무료 반품이 가능한
로켓배송 상품은 오늘 주문하면 내일 도착!



델 80 cm UHD 울트라 사파 모니터

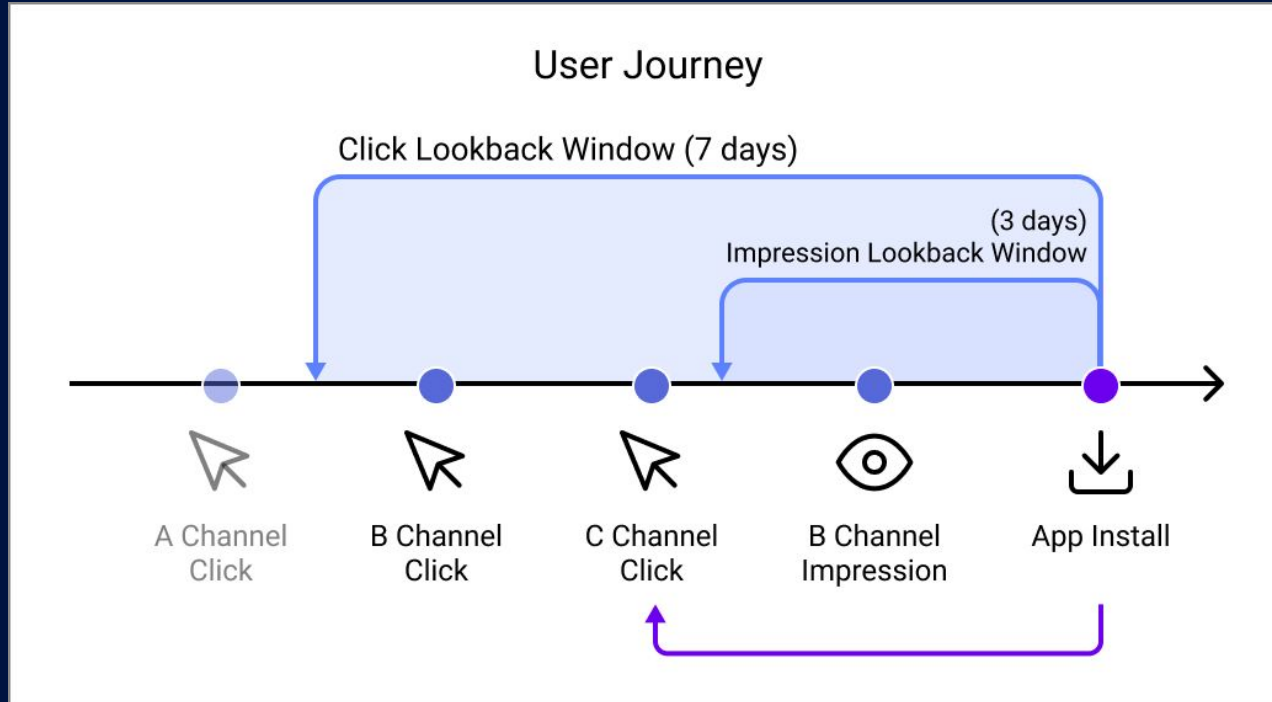
[Shop now](#)



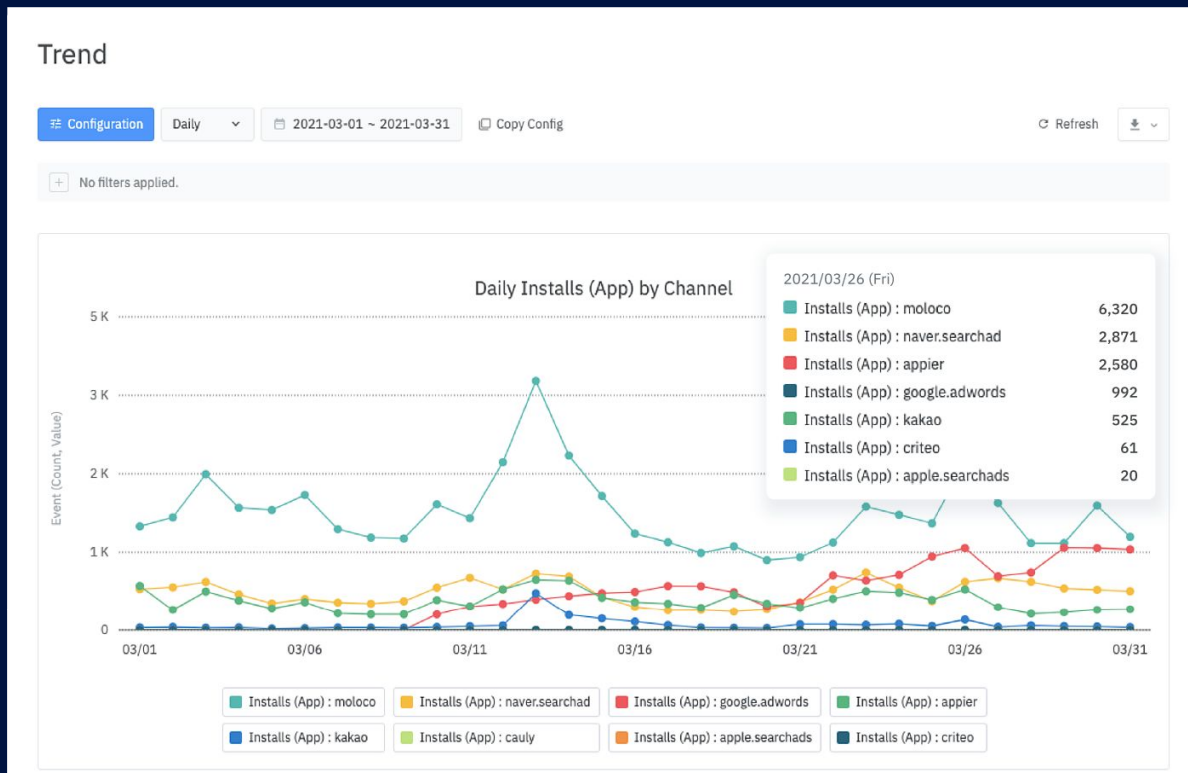
지이크 남성용 핸드메이드 울 투버튼 체크 코트

[Like](#) [Comment](#) [Share](#)

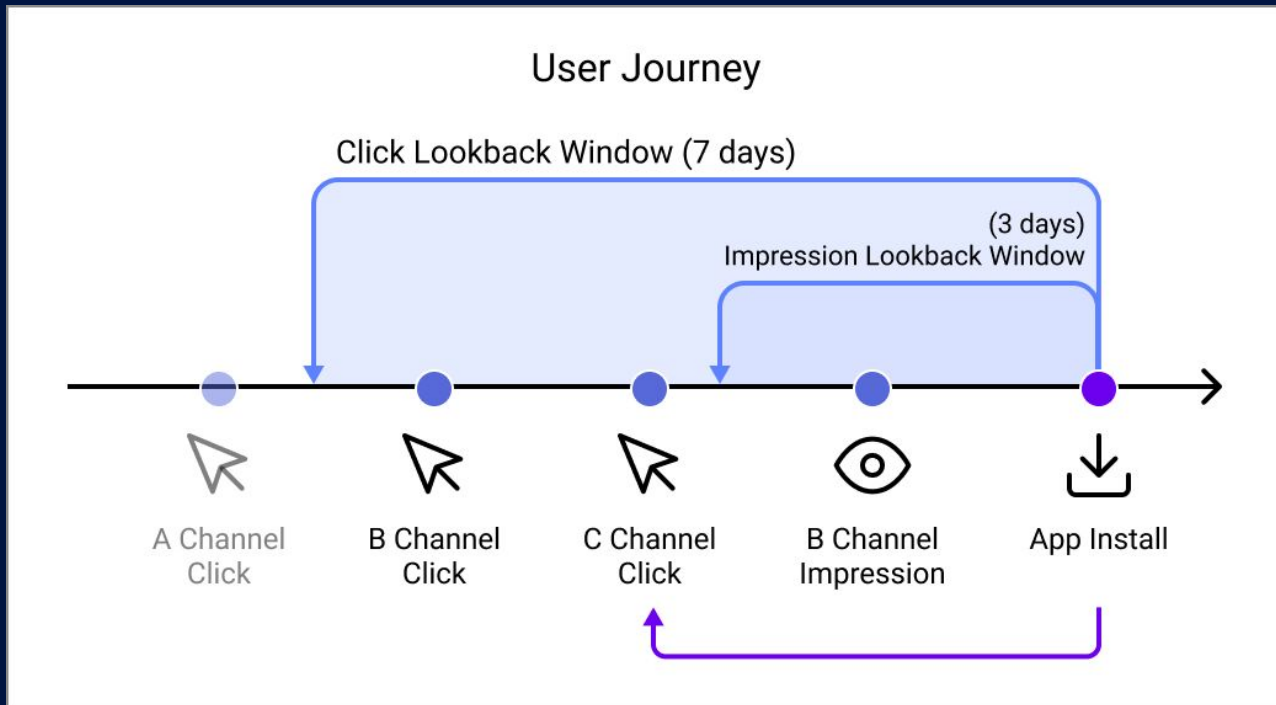
Attribution?



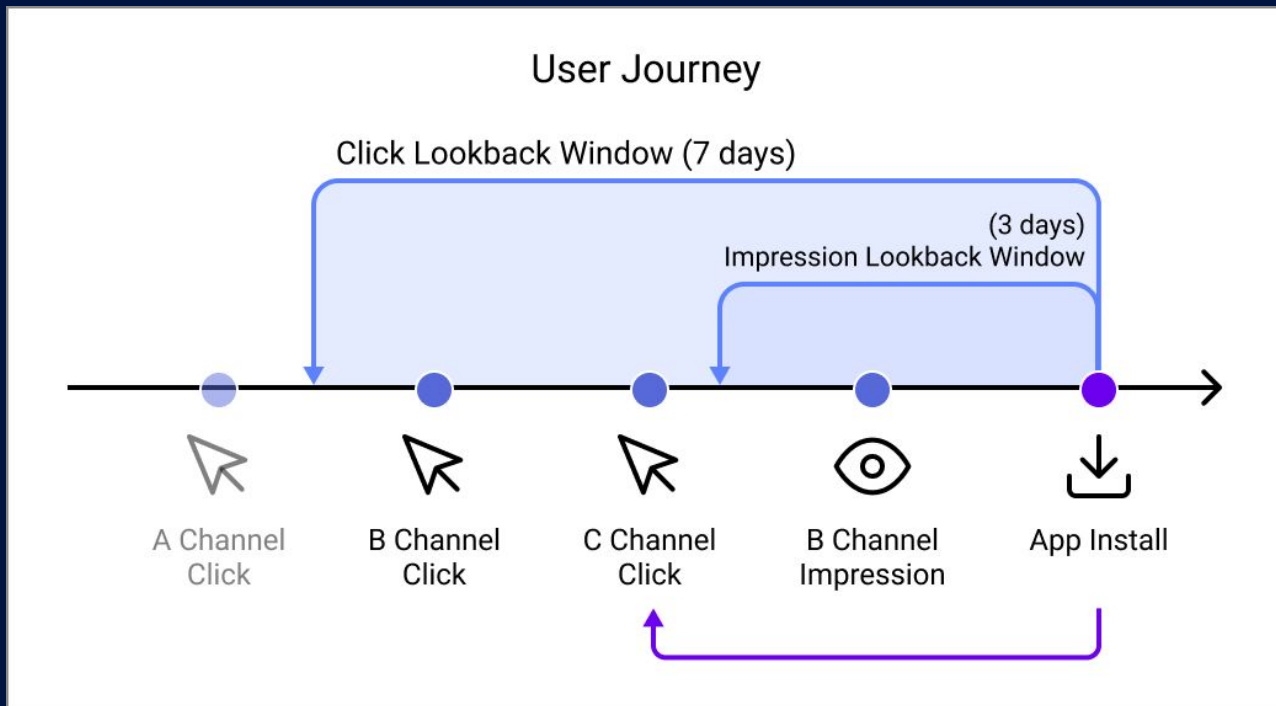
Attribution?



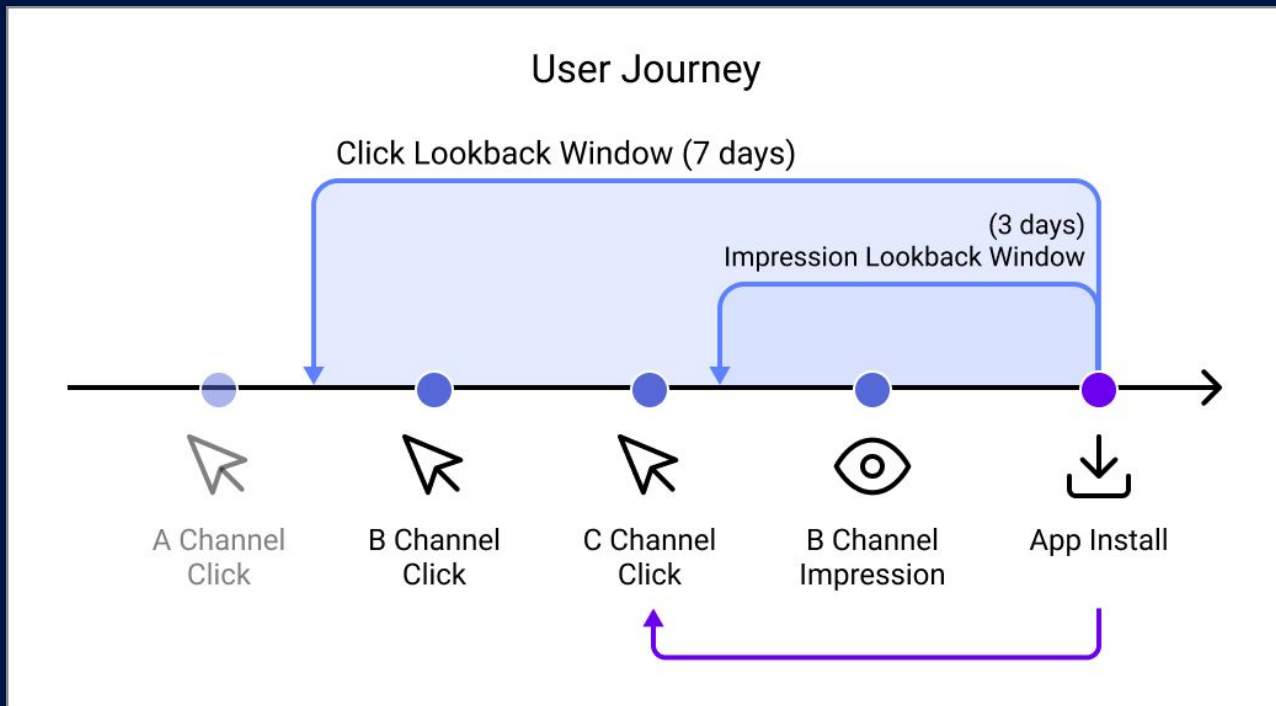
실시간으로 이벤트를 처리



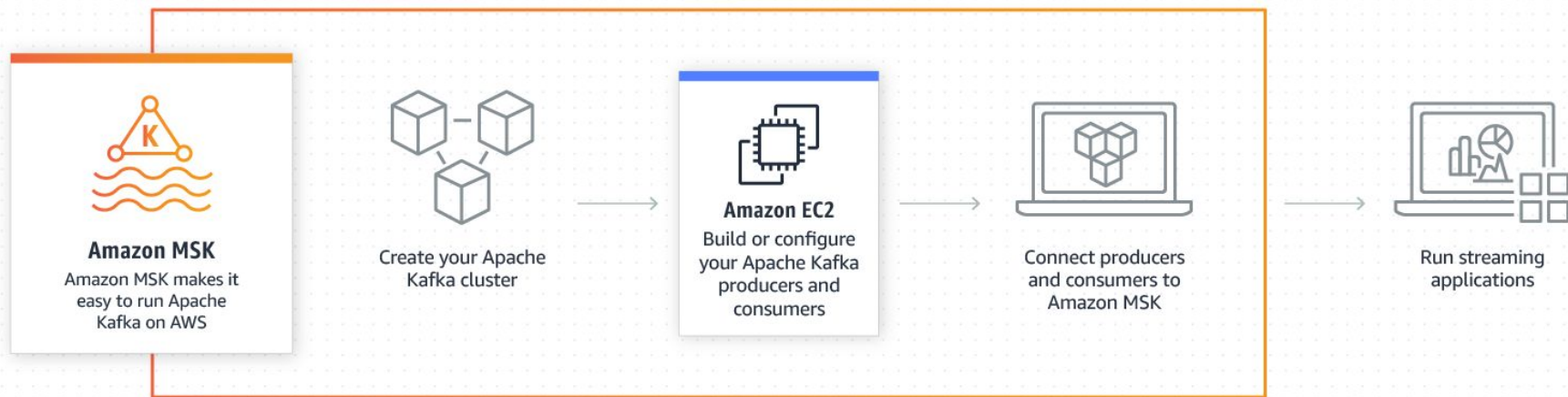
실시간으로 순서에 맞게 이벤트를 처리



실시간으로 순서에 맞게 한 번만 이벤트를 처리

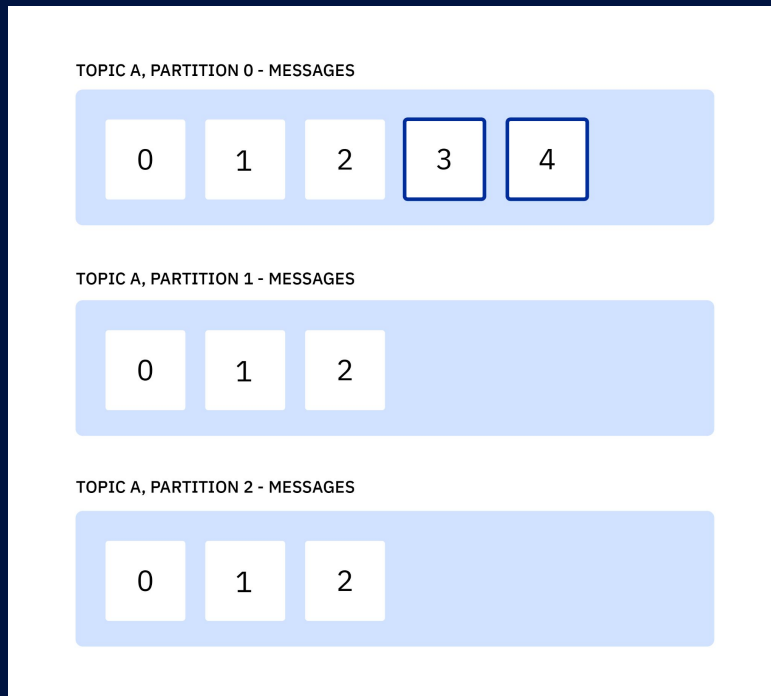


오래 전부터 **Kafka** 적극 활용

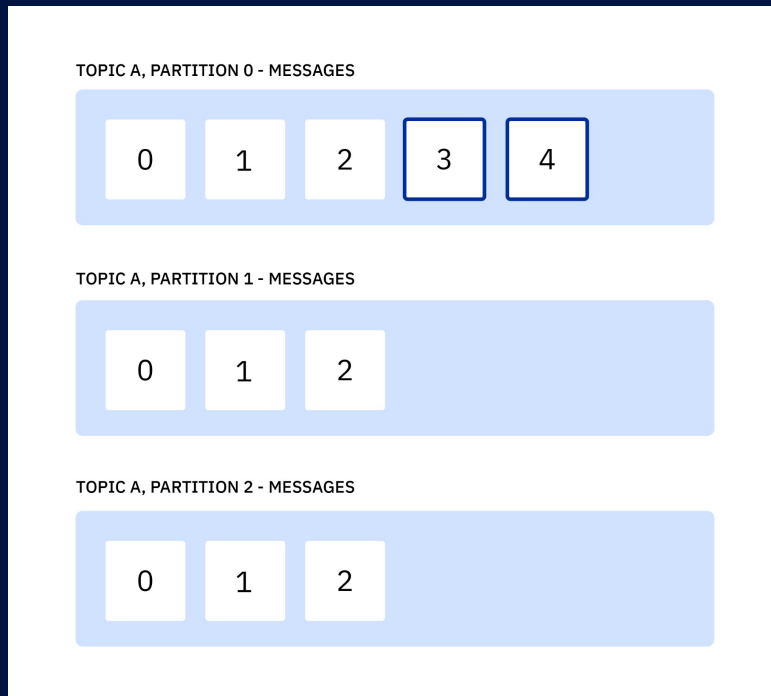


기존 아키텍처의 문제점

순서대로 이벤트를 처리하기 위해 Partition key 활용

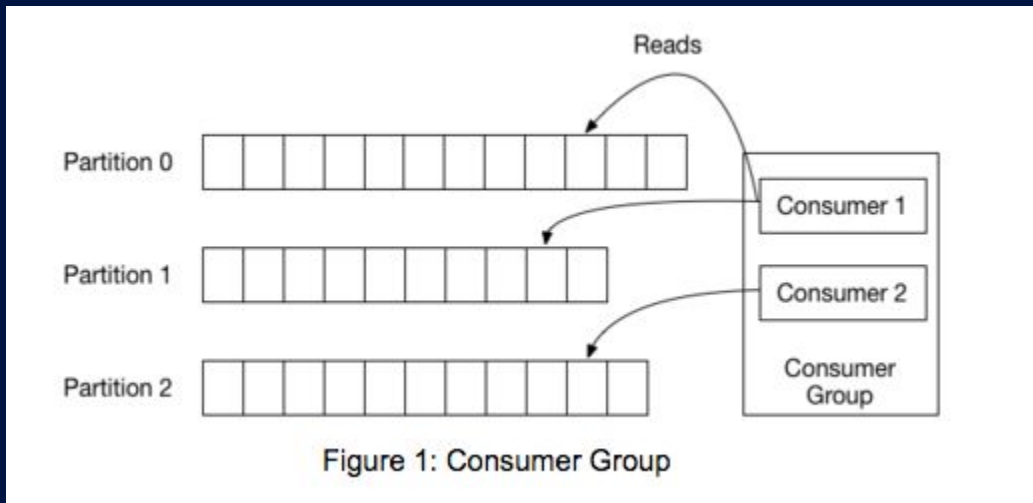


비정상적으로 많은 이벤트를 발생시키는 Fraud 발생시 Skew 발생

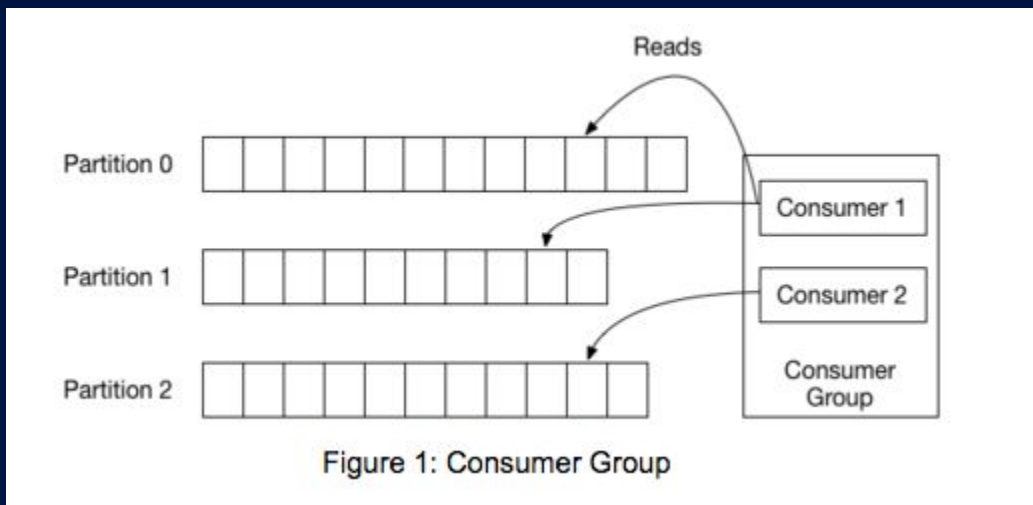


Kafka Design:

하나의 Partition에는 한 consumer만 consume 가능



Partition을 계속 늘릴 수록 kafka broker의 부담이 커짐
처리해야할 partition 수 증가, batch request 성능 저하 등



Consumer python 구현체에서는 multiprocessing 활용:
IPC 등 때문에 성능에 좋지 않음

앞친데 덮친 격:

ECS에서는 Task당 최대 10 vCPU 사용 가능

옆친데 덮친 격:

ECS에서는 Task당 최대 10 vCPU 사용 가능

⇒ 최대 Partition 수만큼만 task 실행 가능

새로운 아키텍처 아이디어

1안:

Spark streaming과 같은 driver, executor model

1안:

Spark streaming과 같은 driver, executor model

문제점:

Spark 실행 환경을 구축하고 운영하기 위한 어려움이 존재
코드 레벨과 운영 모두 큰 migration 필요

2안:

Kafka consumer와 application server decouple

2안:

Kafka consumer와 application server decouple

채택 이유:

기존 운영 환경 최대한 유지 가능

코드 레벨 수정 적음

2안:

Kafka consumer와 application server decouple

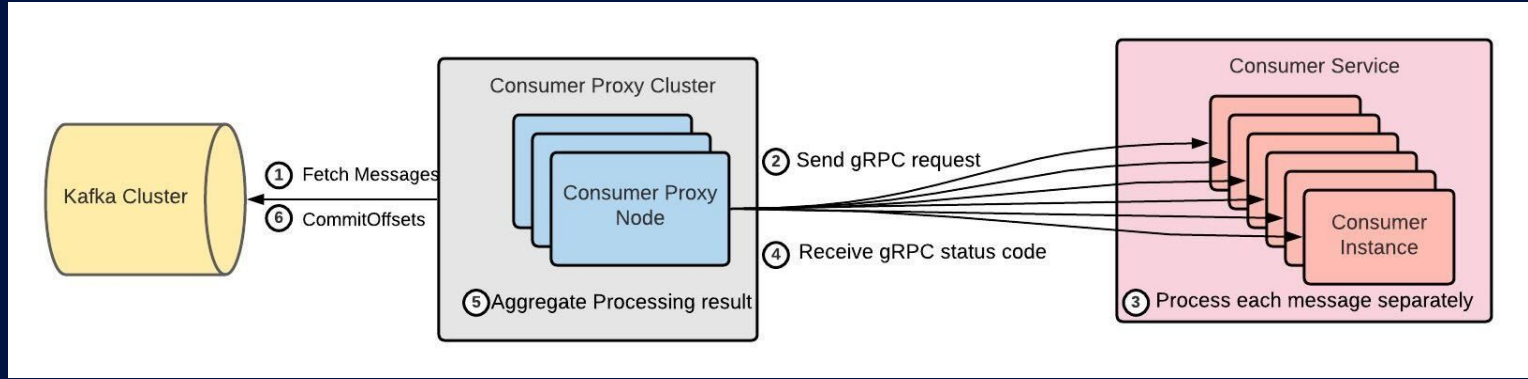
채택 이유:

기존 운영 환경 최대한 유지 가능

코드 레벨 수정 적음

당장 어떻게 하면 될 지 눈에 보임

Consumer Decoupled Architecture



Consumer application의 비즈니스 로직을 application server로 분리

Consumer application의 비즈니스 로직을 application server로 분리
Consumer application과 application server는 따로 배포 운영

Consumer application의 비즈니스 로직을 application server로 분리
Consumer application과 application server는 따로 배포 운영
Consumer application은 application server에게 처리 요청

Consumer application의 비즈니스 로직을 application server로 분리
Consumer application과 application server는 따로 배포 운영
Consumer application은 application server에게 처리 요청
Consumer application과 server간 통신에는 gRPC 활용

무중단 운영:

Application server를 무중단으로 배포할 수 있도록 설계 필요

Application server는 비즈니스 로직 변경이 잦기 때문

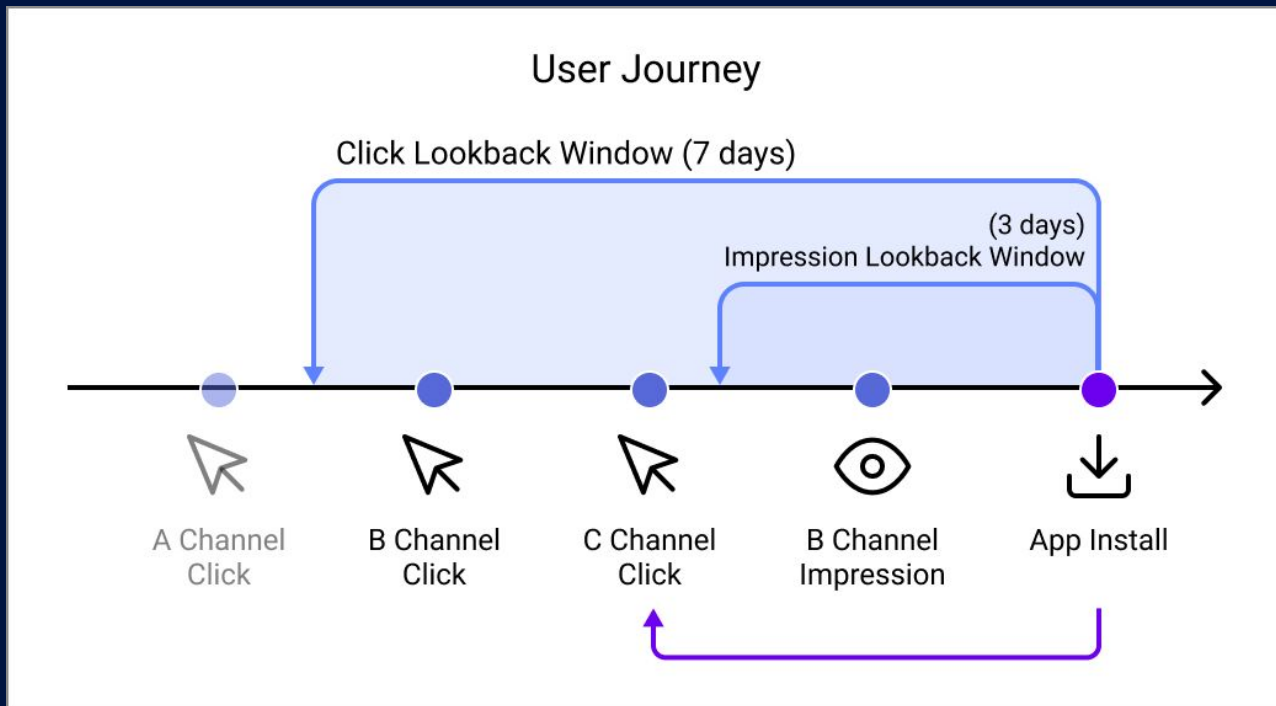
네트워크 비용 최소화:

네트워크 비용이 과대해지지 않도록 설계 필요

Load balancer를 따로 둘 경우 네트워크 비용이 중복되므로
Cloud Map(service discovery)를 활용

데이터 처리 순서:

데이터 처리 순서를 고려하여 비즈니스 로직 구현



데이터 처리 순서:

데이터 처리 순서를 고려하여 비즈니스 로직 구현

Consumer가 batch window 내에서 순서를 고려하여
application server 호출

데이터 처리 순서:

데이터 처리 순서를 고려하여 비즈니스 로직 구현

1. consumer가 batch size만큼 kafka에서 consume
2. consumer에서 batch 내의 touchpoint event를 먼저 처리 요청
 - a. touchpoint event들 DB에 저장
3. touchpoint event들이 모두 처리 완료된 뒤 consumer가 batch 내의 install event 처리 요청
 - a. touchpoint를 먼저 로드
 - b. event 시간을 고려하여 attribution 로직 실행

재시도 로직:

Application server가 실패한 경우 재시도 로직

Consumer: At-least-once delivery

Application server: Exactly-once delivery

재시도 로직:

Application server가 실패한 경우 재시도 로직

Application server로부터 실패 응답을 받을 경우 재시도

Max retry 초과시 실패한 메시지를 DLQ에 저장

Consumer는 Kafka에서 받아온 메시지를 모두 처리한 뒤에 commit

재시도 로직:

Application server가 실패한 경우 재시도 로직

Consumer가 모종의 이유로 데이터를 중복으로 consume 하더라도
Application server가 Exactly-once 처리하므로 문제 없음

경험한 어려움

A record는 IP level까지만 명시 가능한데 ECS에서는 port level이 필요
grpc-go에서 기본 DNS load balancer가 grpclb 라는 이름만 지원

해결:

Service Discovery SRV record 활용

AWS Cloud Map을 사용하고 있어서 grpc-go의 resolver interface에
맞춰서 Cloud Map API를 사용하는 [resolver 구현](#)

Python에서 직접 gRPC server를 실행할 경우 multi thread 환경이 되어 multi core 활용이 잘 되지 않음

해결:

Envoy를 sidecar로 붙여서 해결

새 아키텍처 적용 후 결과

Kafka cluster 부하 감소:

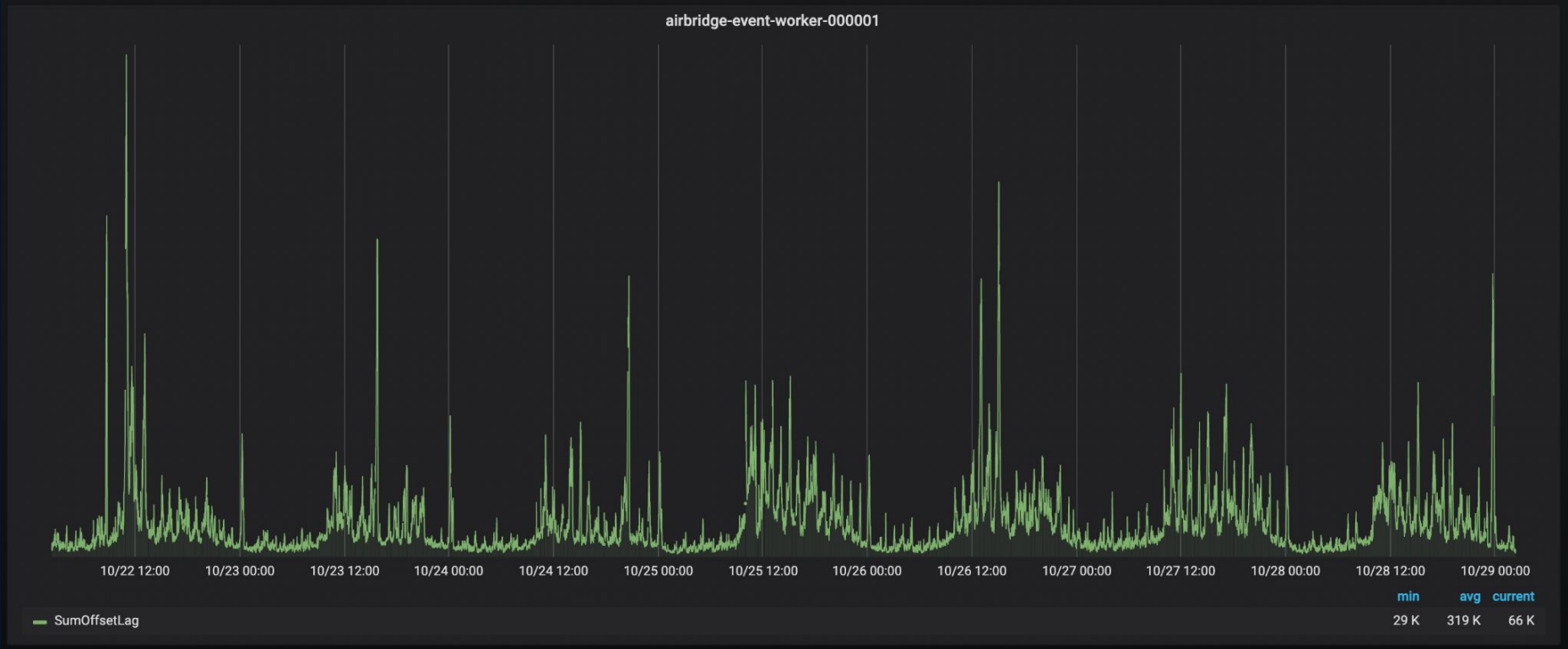
Partition 수를 기존 대비 20% 정도로 적게 유지할 수 있음
그만큼 적은 broker들로 서비스 가능

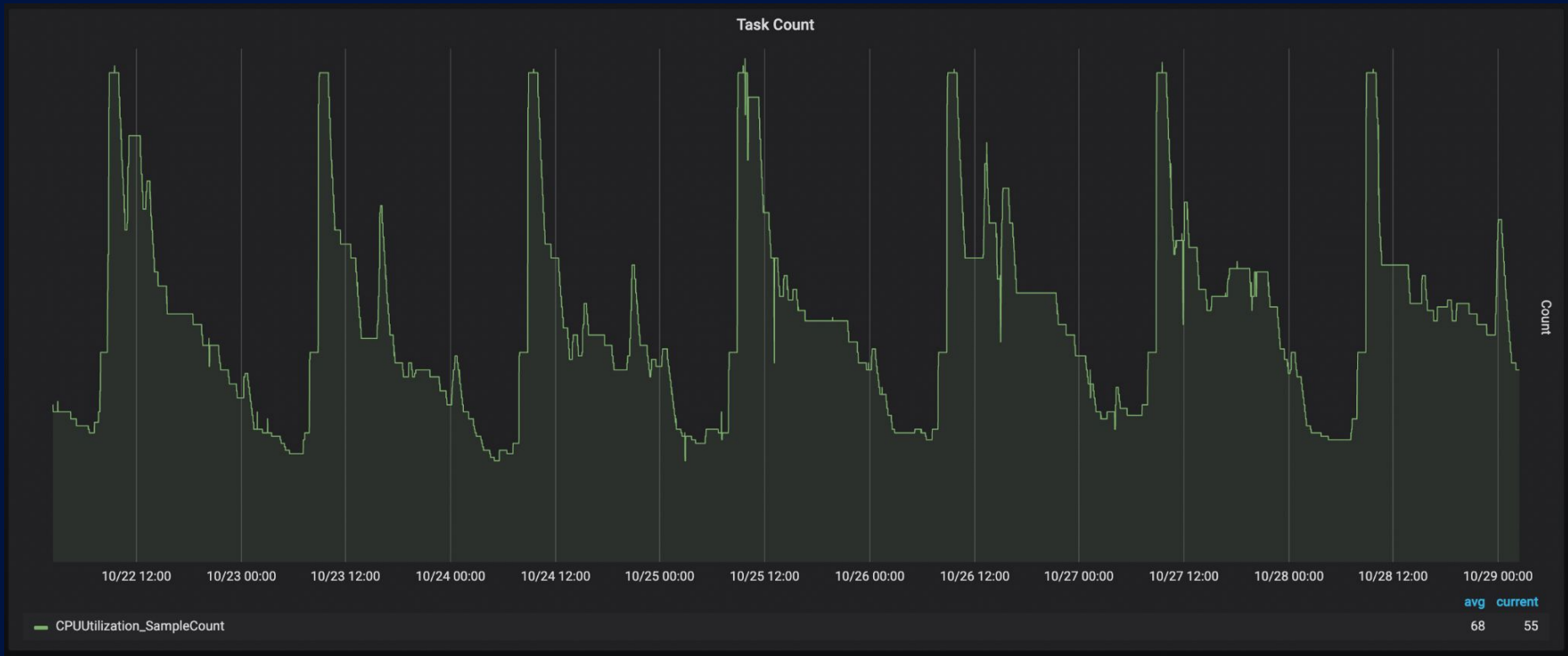
유연한 scaling:

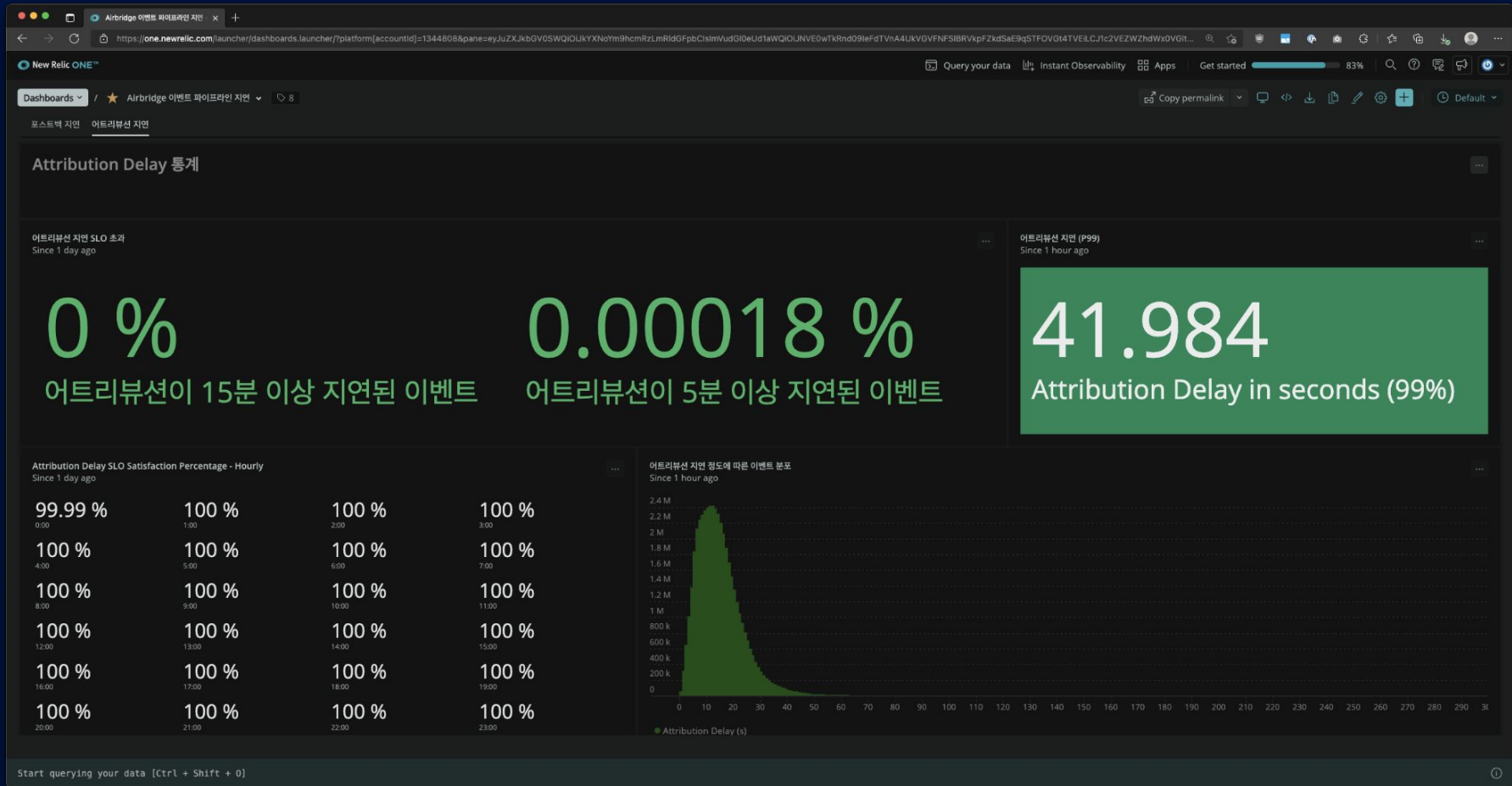
Consumer는 partition 수의 약수만큼 provision

Application server는 traffic에 따라 유연하게 scaling 할 수 있게 됨

Scaling에 대한 고민이 적어짐:
Scale up에 더 이상 목 매지 않아도 되고
Scale out도 자유로움







앞으로 더 시도해봐야 할 것

네트워크 비용 더 최소화:

Zone awareness로 cross zone network 비용 절감

감사합니다.

For inquiries, please reach out to

Juhong Jung, Software Engineer

Email : juhong@ab180.co

AB180 INC.

www.ab180.co