Portfolio

운영 개선으로 이어진 기술 조사, 인프라 의사결정, 시스템 개선 사례입니다.

기술 사례

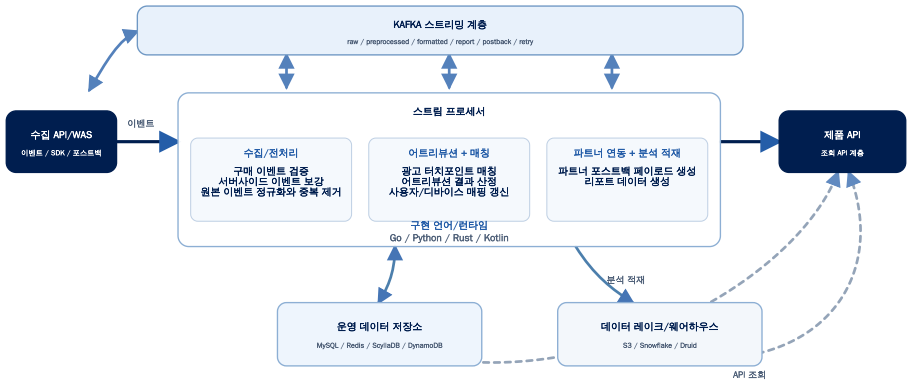
서비스 아키텍처 / 2026

[Service Architecture] Airbridge Datapipeline

Airbridge 데이터 파이프라인은 유입 이벤트를 어트리뷰션 결과, 파트너 포스트백, 리포팅 데이터, 제품 조회 상태로 바꾸는 핵심 처리 계층입니다.

AIRBRIDGE 실시간 데이터 파이프라인 아키텍처

주요 처리 경계와 데이터 이동을 압축해 표현했습니다.



이벤트 처리 흐름

수집 API는 SDK 이벤트, 서버사이드 이벤트, 리다이렉트 요청을 받아 Kafka에 원본 이벤트를 저장합니다. 이후 이벤트 검증, 이벤트 보강, 정규화, 중복 제거를 거쳐 전환 이벤트를 발생시킨 광고 이벤트를 특정하고, 각 처리 결과는 리포트, 고객사 API, 파트너 포스트백 전송 등에 사용됩니다.

저장소와 조회 경계

각 단계는 Kafka topic을 경계로 둔 domain worker로 나뉩니다. 구매 검증, 서버사이드 이벤트 보강, 전처리, 어트리뷰션/터치포인트 처리, 사용자/디바이스 매핑 갱신, 파트너 포스트백 전송, 분석 export, report materialization이 서로 다른 실패 지점과 확장 단위를 갖습니다. 운영 저장소는 설정, 체크포인트, 사용자/디바이스 매핑, 저지연 조회 상태를 담당하고, S3/Snowflake/Druid 계열 저장소는 원본 이벤트, 포맷된 이벤트, 리포트, 전송 로그를 분석과 리포팅 목적으로 보관합니다.

- 한 이벤트가 수집된 뒤 어떤 worker가 어떤 상태를 만들고 어떤 후속 결과를 남기는지 추적할 수 있도록, 수집/전처리, 어트리뷰션, 터치포인트 처리, 파트너 포스트백 전송, 분석 적재 책임을 나눠 운영합니다.
- Kafka는 원본, 전처리, 포맷, 리포트, 포스트백, 재시도 흐름 전반에서 처리 단계와 재처리 경계를 나누는 공통 스트리밍 계층입니다.
- 정규화 이후 어트리뷰션 계산과 터치포인트 처리를 별도 워커에 할당하여, 각 처리에 맞는 스택과 스케일링 정책을 이용할 수 있습니다.
- 파트너 포스트백 전송은 별도 작업 큐, 재시도, failover 경로로 처리해 외부 API 지연이나 장애가 메인 이벤트 처리와 리포트 생성 경로를 직접 막지 않습니다.

데이터베이스 운영 최적화 / 2024

[Digging Deep] ScyllaDB Materialized View 최적화

ScyllaDB에서 일부 Materialized View가 base table보다 훨씬 큰 디스크를 차지하면서 클러스터 증설이 필요한 것처럼 보였던 문제를 분석했습니다. 실제 조회 경로를 확인해 View 의존을 제거하고 디스크 사용량을 30%(약 9TiB) 줄였습니다.

MATERIALIZED VIEW 제거 판단

디스크 사용량 이상치에서 출발해 ScyllaDB 내부 동작으로 원인을 확인하고, Materialized View 없이도 필요한 조회를 처리할 수 있는지 검증했습니다.

스토리지 원인 분석

1. 디스크 사용량 이상치
   특정 Materialized View가 base table보다 훨씬 큰 공간을 차지했습니다.
2. View 갱신 방식
   View key가 바뀌면 기존 view row 삭제와 새 view row 삽입이 함께 발생합니다.
3. tombstone 정리 지연
   gc_grace_seconds가 지나기 전에는 compaction 이후에도 tombstone이 남을 수 있습니다.
4. 증설 압박
   활성 데이터보다 정리 대기 데이터가 클러스터 용량을 더 크게 압박했습니다.

조회 경로 검증

1. base table 직접 조회
   View 없이도 필요한 조회 패턴을 처리할 수 있는지 확인했습니다.
2. Query tracing
   읽기 경로와 비용이 운영 허용 범위에 들어오는지 확인했습니다.
3. 성능 영향 판단
   운영 적용이 가능한 수준으로 성능 차이가 작았습니다.
4. Materialized View 제거
   파생 테이블 의존을 줄여 디스크 사용량과 증설 압박을 낮췄습니다.

문제 배경

용량 경보를 계기로 테이블별 디스크 사용량을 확인하던 중, 특정 Materialized View 묶음이 base table보다 훨씬 큰 공간을 차지하고 있음을 발견했습니다. 원인 분석 결과 View key가 바뀌면 ScyllaDB는 기존 view row를 지우고 새 row를 쓰며, 이 과정에서 생긴 tombstone은 compaction이 돌아도 gc_grace_seconds 전에는 안전하게 정리될 수 없음을 확인했습니다.

성과

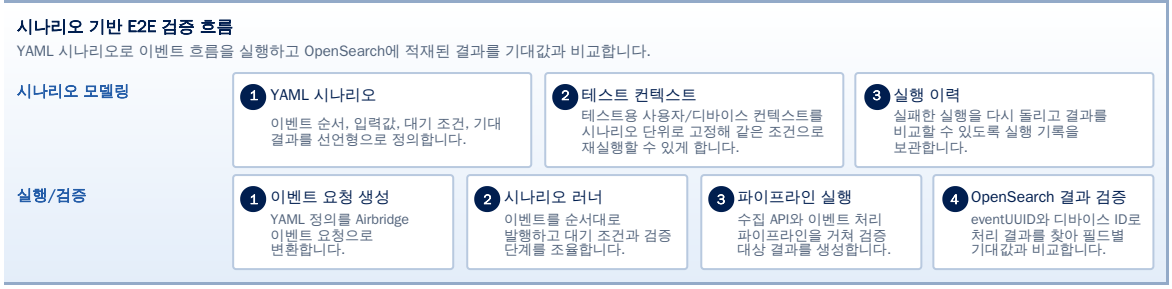
해결책은 노드 추가가 아니라 View 제거였습니다. base table 조회만으로 제품 요구 성능을 만족한다는 것을 확인하고 Materialized View 의존을 없앴습니다. 그 결과 쓰기 증폭과 tombstone 보존으로 계속 커지던 저장 경로를 제거했고, 클러스터 디스크 사용량을 30%(약 9TiB) 줄여 증설 압박을 낮췄습니다.

- Materialized View 조회와 기준 테이블 조회를 비교해, 데이터 모델을 단순화해도 될 만큼 성능 차이가 작다는 점을 확인했습니다.
- 테이블별 디스크 사용량 편차에서 시작해 조회 경로 검증까지 순서대로 좁혔습니다. 스토리지 지표로 과도하게 커진 View를 찾고, tracing으로 기준 테이블 조회 경로를 검증했으며, ScyllaDB 내부 동작을 근거로 삭제 이후에도 디스크 사용량이 즉시 줄지 않는 이유를 설명했습니다.
- 클러스터 증설 전에 해당 조회 패턴이 파생 테이블을 정당화하는지 확인 후 의사결정 하였습니다.

테스트 자동화 플랫폼 / 2024-2026

[Innovation] 선언형 E2E QA 시스템 구현

데이터 파이프라인을 통과하는 사용자 여정을 YAML 시나리오로 정의하고, OpenSearch에 적재된 처리 결과를 조회해 단계별 기대값을 검증하는 E2E 테스트 시스템을 만들었습니다.



문제 배경

기존 QA는 여러 이벤트를 직접 만들고, 비동기 파이프라인이 끝난 뒤 OpenSearch에서 결과를 따로 확인해야 했습니다. 이 시스템은 가입, 설치, 구매, 포스트백처럼 순서가 있는 사용자 여정을 하나의 시나리오로 정의하고, 사용자/디바이스/앱/리전/브랜치 컨텍스트를 고정한 상태로 실제 수집 API와 이벤트 처리 파이프라인을 통과시킵니다.

성과

검증 기준을 UI나 mock 결과가 아니라 파이프라인이 실제로 적재한 OpenSearch 문서에 두었습니다. 실행기는 eventUUID와 디바이스 ID로 처리 결과를 찾고, 필드별 기대값과 downstream 불변 조건을 비교합니다. 그래서 엔지니어는 같은 시나리오를 재실행하고, 실패한 단계와 요청 trace를 확인하며, 변경 사항이 E2E 처리 결과를 깨뜨렸는지 빠르게 판단할 수 있습니다.

- YAML 또는 내부 에디터 입력을 실행 가능한 E2E 시나리오로 변환해, 사람이 매번 이벤트를 직접 만들던 검증 흐름을 재현 가능한 테스트로 바꿨습니다.
- 이벤트 요청 생성, 순차 실행, 비동기 처리 대기, OpenSearch 조회, 필드별 assertion을 하나의 runner 안에서 처리했습니다.
- 사용자 데이터, 디바이스 메타데이터, 앱 설정, 데이터 소스, 리전, feature branch를 실행 컨텍스트로 고정해 같은 조건의 재실행과 회귀 검증이 가능하게 했습니다.
- 실패한 단계, 요청 trace, OpenSearch 조회 결과를 함께 남겨 어느 입력이 어떤 파이프라인 결과를 만들지 못했는지 추적할 수 있게 했습니다.