경력기술서

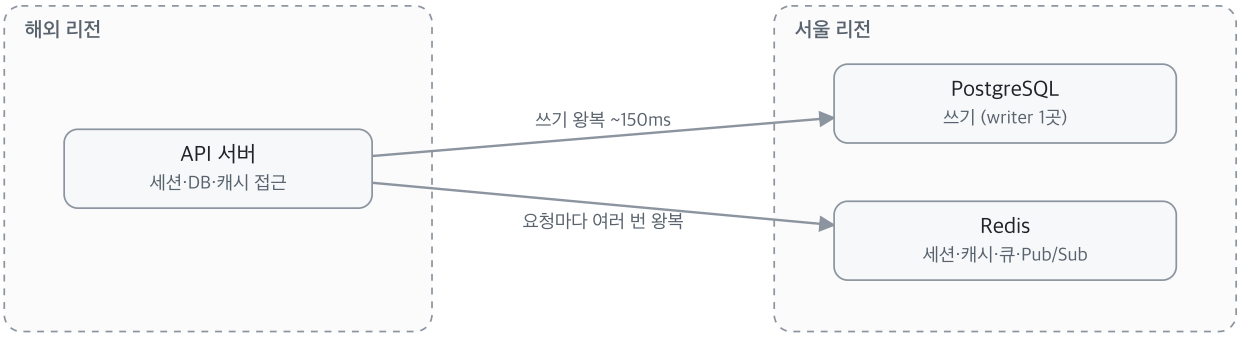
그림이 본문, 글은 보조 설명입니다. 회사 내부 정보(서비스명·주소·고객 데이터)는 일반화했습니다.

플리토 · 실시간 AI 통번역

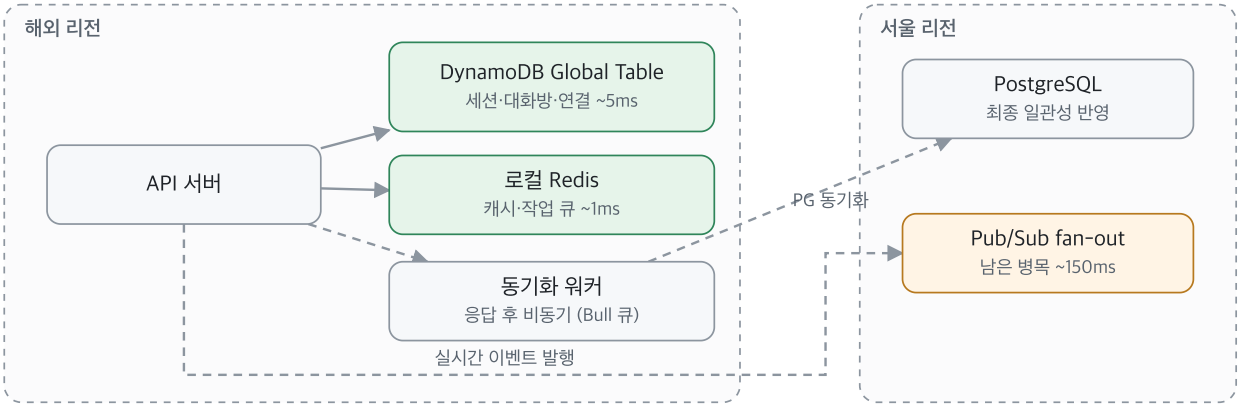
글로벌 리전 레이턴시 개선

2025.11 – 2026.05 | 설계 · 구현 주도 | DynamoDB · Redis · PostgreSQL | 재현 실험 코드 →

Before — 모든 쓰기가 서울 리전으로 왕복



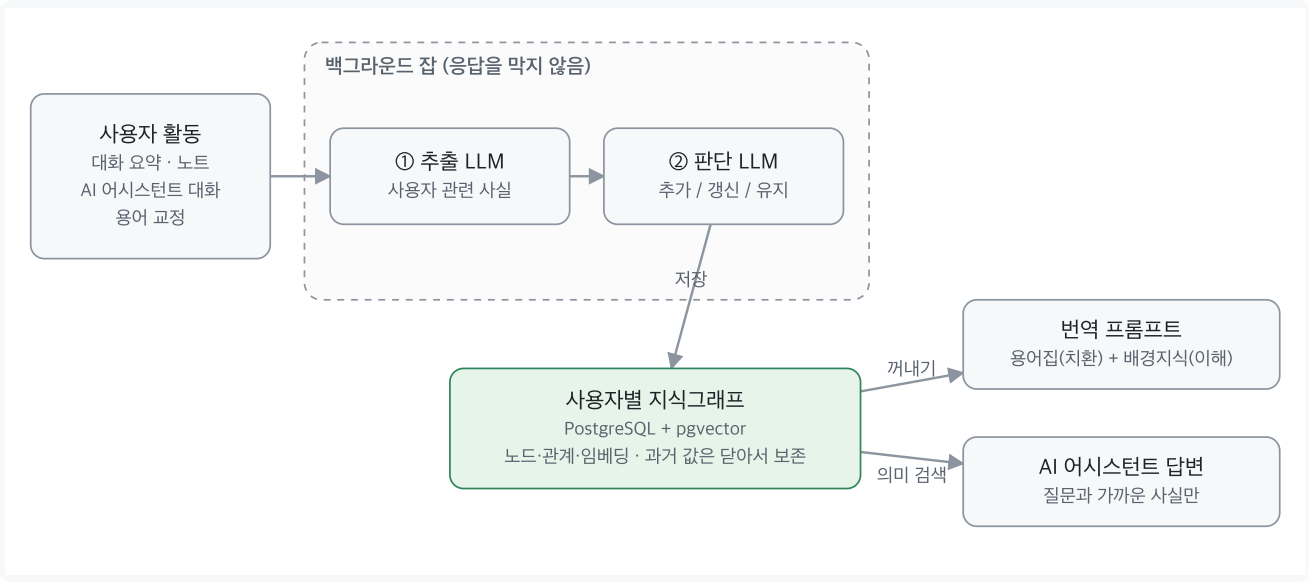
After — 리전 로컬 쓰기 + 비동기 동기화



문제	해외 리전에서 쓰면 국내보다 4~5배 느리다는 피드백. 유력한 원인은 서울 리전에 묶인 DB·Redis 쓰기 왕복(1회 약 150ms)이 요청마다 여러 번 발생하는 구조.
판단	Aurora Global Database·Aurora DSQL 검토 — 쓰기는 여전히 한 리전이거나 대륙 간 미지원이고 비용도 더 높아 보류. 실시간성이 강한 데이터만 리전 로컬 쓰기로 옮기기로 결정.
해결	대화방·세션·연결 추적을 DynamoDB Global Tables로, 캐시·작업 큐를 리전별 Redis로 전환. PostgreSQL은 큐로 비동기 동기화하고, 기능 플래그로 언제든 기존 구조로 되돌릴 수 있게 함.
결과	Close API **1,673 → 327ms (-80%)**, Create API **2,630 → 90ms (-97%)** — 해외 상황을 재현한 개발 환경에서 15회 측정. 남은 병목은 Pub/Sub fan-out.

지식그래프 기반 초개인화

2026.01 – 2026.10 | 백엔드 설계 · 구현 · 상용 반영 | PostgreSQL · pgvector · LLM



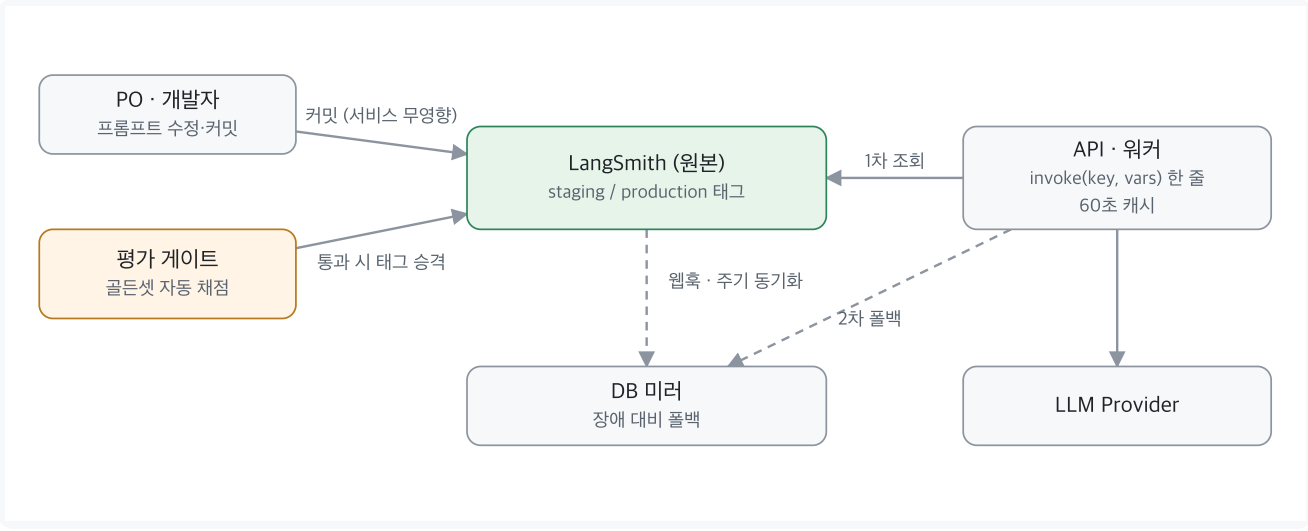
문제 사용자가 여러 번 고친 표기나 자신의 분야 맥락을 번역·AI 답변이 기억하지 못함.

해결 사용자 활동에서 LLM이 사실을 추출하고, 다시 LLM이 기존 지식과 비교해 추가/갱신/유지를 판단해 사용자별 그래프로 저장. 번역에는 용어집·배경지식으로, AI 답변에는 질문과 의미가 가까운 사실만 넣음. 모든 쌓기는 백그라운드 잡.

개선 상용 데이터를 보니 노트 발표자의 이야기가 사용자 사실로 섞이는 문제가 있어 "사용자 중심" 관계 구조로 재설계. 고정 기준선 게이트로 회귀 없이 채택한 뒤 상용 사용자 이관 완료.

LLM 프롬프트 관리 체계

2026.06 | 설계 · 구현 | LangSmith · 평가 자동화



문제 프롬프트가 코드에 있어 문장 하나를 고쳐도 배포가 필요했고, 바뀐 버전의 품질을 확인할 방법이 없음.

해결 LangSmith를 원본으로 두고 서버가 직접 조회(60초 캐시, 장애 시 DB 미러 폴백). 개발/운영은 태그로 분리하고, 골든셋 자동 채점을 통과한 버전만 운영 태그를 받게 함.

결과 코드 배포 없이 프롬프트를 고쳐 약 1분 안에 반영 — PO·PM용 사용 가이드를 함께 제공.

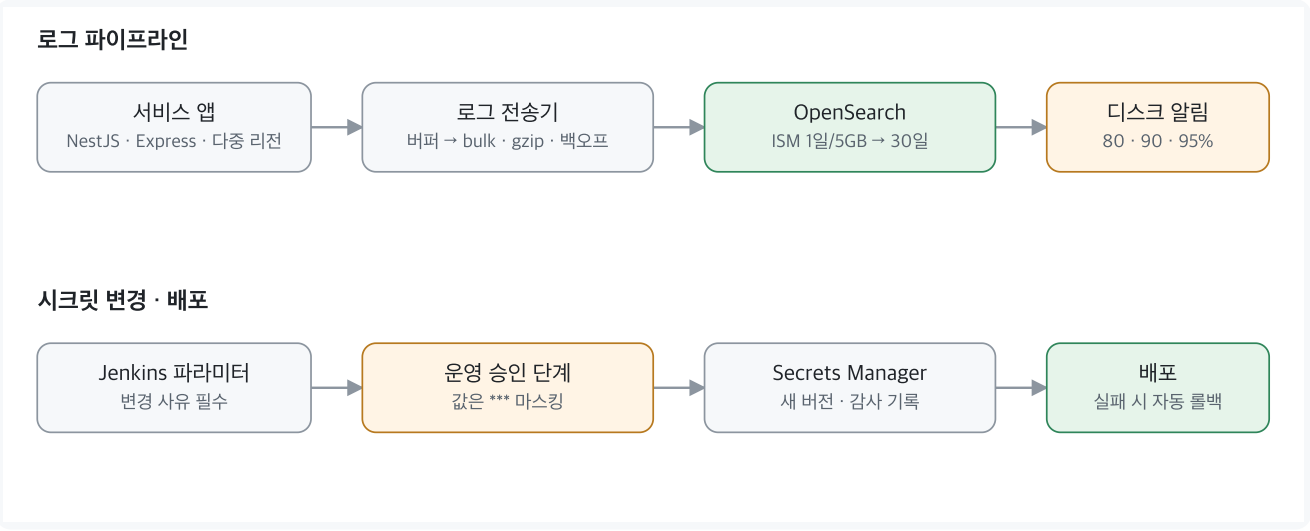
외부 AI 에이전트 연결(MCP) — 기능 오너

2026.10 – 진행 중 · PRD · Figma · 서버 명세

목표	고객이 쓰는 AI 에이전트(Claude · ChatGPT · Cursor 등)가 대화·노트 기록을 읽고 용어집을 미리 준비하게 함.
한 일	타사 연결 방식·사용 한도를 비교 조사해 PRD를 쓰고, Figma 시안(데스크톱·모바일 웹)·서버 명세·아키텍처 그림을 한 벌로 작성. 이 문서 방식을 이후 기능(음성 파일 업로드)에도 적용 중.
상태	출시 전 — 선행 작업이 끝난 뒤 구현을 이어감.

로그 · 배포 운영 안정화

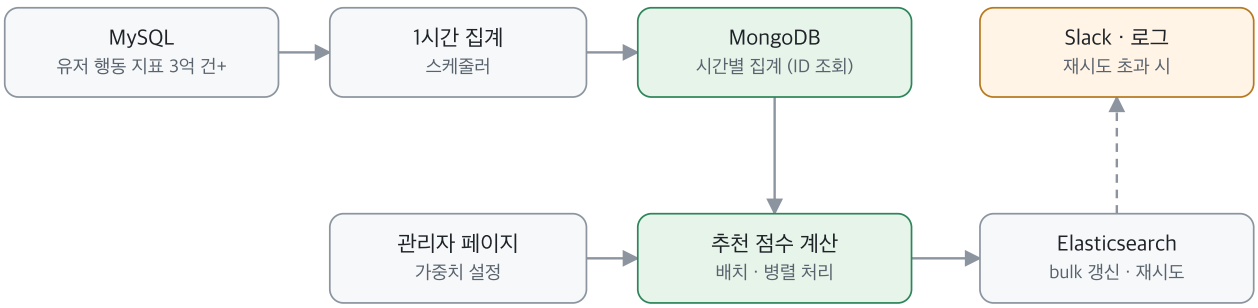
2025.10 – 2026.05 · 구축 · 장애 복구 · OpenSearch · Jenkins



로그	여러 리전의 서비스 로그를 OpenSearch로 수집(전송 실패가 서버에 영향 없게 설계). 이후 디스크 풀 장애는 debug 로그 유입이 원인이라 유입을 막고 보관 정책·압축·디스크 알림을 기본값으로 고정.
배포	시크릿 변경을 Jenkins 파이프라인으로 일원화(변경 사유 필수 · 값 마스킹 · 운영 승인). 서버 CPU 포화 장애 뒤 불필요한 로그를 정리하고, 배포가 실패하면 이전 버전으로 되돌리는 배포·로그 보관 정책을 도입.

추천 상품 스케줄러 재구축

MySQL · MongoDB · Elasticsearch



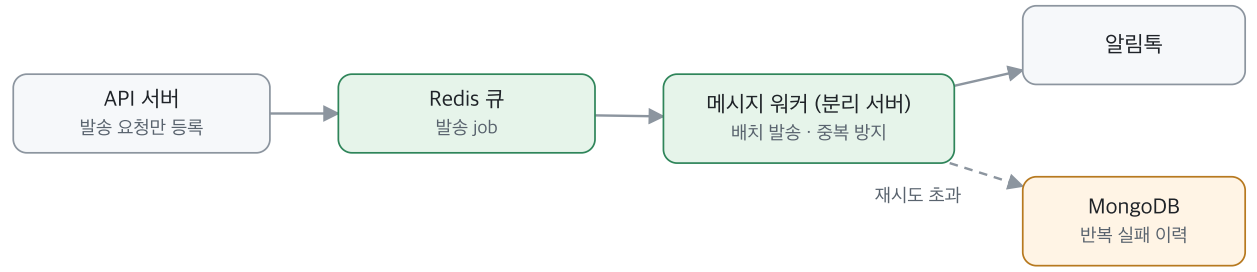
문제 3억 건 이상의 행동 지표를 매번 시간 범위로 조인·정렬해 60분 이상 걸리고, 추천 기준이 쿼리에 고정돼 3년간 손대지 못함.

해결 행동 지표를 1시간 단위로 MongoDB에 미리 집계해 ID 조회로 바꾸고, 점수 계산·반영을 배치 병렬 처리. 가중치는 관리자 페이지에서 설정, Elasticsearch 반영 실패는 재시도 후 알림.

결과 스케줄러 **60분 → 15분**.

알림 메시지 워커

Redis 큐 · MongoDB

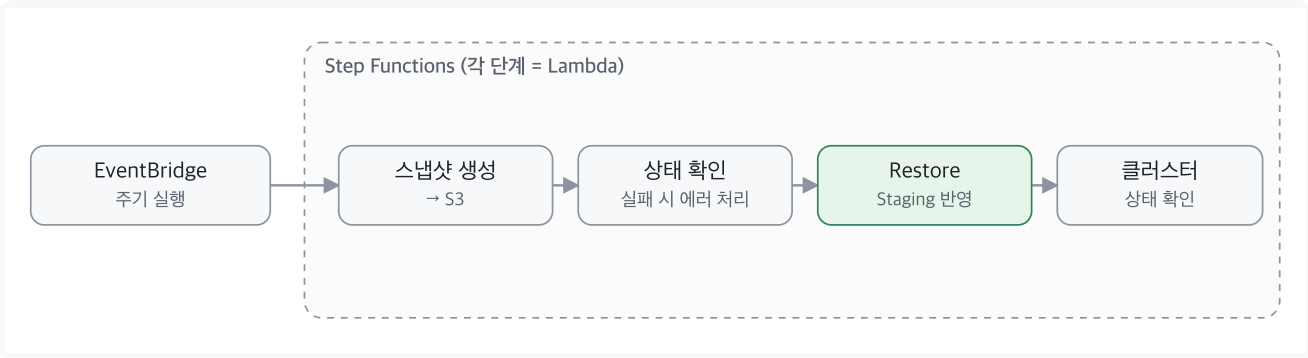


문제 100만 사용자 대상 주기 알림톡 발송이 API 서버에 부담을 줌.

해결 발송을 Redis 큐 + 별도 워커 서버로 분리하고 배치 발송·중복 방지 적용. 실패는 재시도, 반복 실패는 MongoDB에 이력 저장.

Elasticsearch 스냅샷 마이그레이션 자동화

Step Functions · Lambda · S3



해결 스테이징 검색 데이터를 운영과 맞추기 위해 스냅샷 생성 → 상태 확인 → 복원 → 클러스터 확인을 Step Functions 단계로 자동화하고 EventBridge로 주기 실행.

개발 방식 · 권한

Feature Flag 기반 Trunk-Based 개발

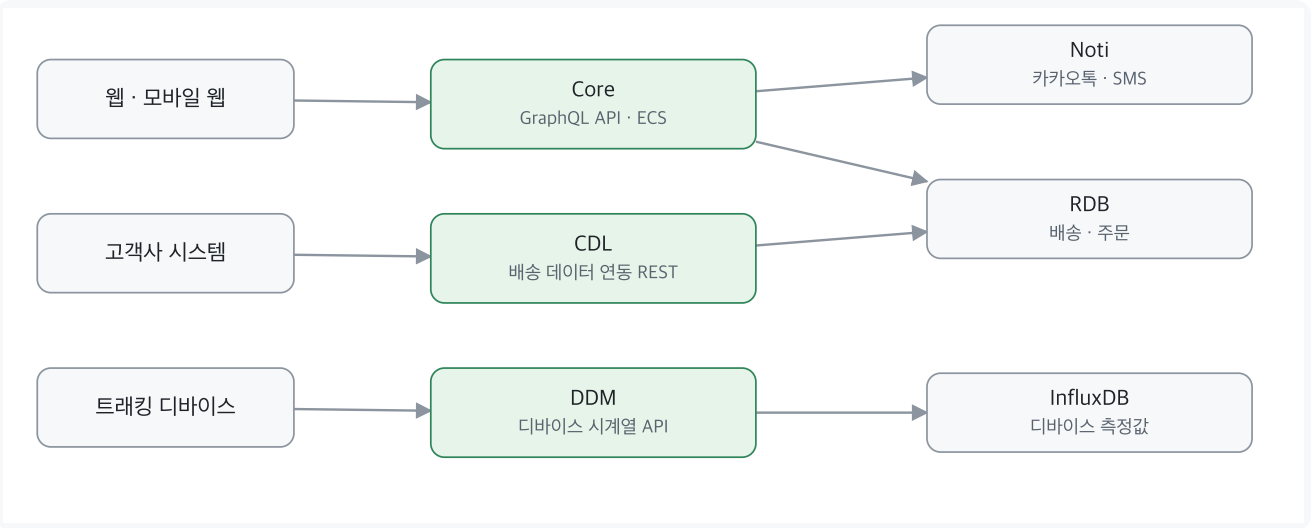
환경별 브랜치 충돌과 배포 지연 문제 → 기능 플래그로 활성화를 제어하고 main 하나로 배포. API 배포 상태를 문서(Swagger) 라벨로 공유.

AWS 권한 정책

모든 권한을 가진 단일 그룹 → 신규 입사자는 읽기 전용, 업무별 권한 그룹, 민감 작업은 별도 승인으로 분리.

기능 기준 서버 분리 · 통합 관제 플랫폼

NestJS · GraphQL · ECS · Lambda · InfluxDB



문제	고객사 연동 API 트래픽이 몰리면 서비스 전체가 영향을 받았고, 산업별 서비스가 따로 있어 운영 리소스가 늘어남.
해결	클라이언트 API(Core) · 디바이스 시계열(DDM) · 고객사 연동(CDL) · 알림(Noti)으로 서버를 분리하고 일부는 Lambda로 구성. 산업별 서비스는 출고·배송·검수·도착 공통 구조의 통합 플랫폼으로 이관.
결과	운영 중인 고객사 서비스를 멈추지 않고 배송 약 **200만 건** · 디바이스 시계열 약 **5억 건** 이관.

문제 해결 · 협업

고객사 연동 API 지연

10분마다 1만 건 이상 들어오는 연동 요청의 중복 검사가 느린 쿼리를 만듦 → 메모리 조회로 쿼리 수를 줄이고 1,000건 단위로 나눠 병렬 처리 (**20초 → 5초**), 이후 Serverless로 분리.

시계열 데이터 분리

콤마로 이어 붙인 문자열로 저장하던 측정값을 InfluxDB로 분리, 실시간 변경은 GraphQL Subscription으로 전달.

VOC time

니즈가 정리되지 않은 고객사에 MVP를 먼저 제공하고 한 달에 한 번 현장 실무자를 만나 기능을 맞춤 개발.