손민기

Backend Developer & Tech Lead

smk2692@naver.com | 010-7420-0799 | GitHub | Blog

소개

"익숙함에 머무르기보다, 더 나은 방향을 만드는 개발자"가 되고자 합니다.

현재에 안주하는 순간 성장은 멈춘다고 믿기에, 늘 새로운 도전과 변화를 즐기는 개발자입니다. 문제가 보이면 먼저 움직이고, 더 나은 답을 찾기 위해 동료들과 치열하게 토론하는 과정을 좋아합니다.

빠르게 변하는 기술 흐름 속에서도 책과 강의로 꾸준히 배우며, ADR과 실제 구축 사례를 근거로 새로운 기술을 팀에 자연스럽게 스며들게 해왔습니다.

경력 요약 총 경력 8년 3개월

(주)펫프렌즈

2021.03 - 현재 (5년 3개월)

백엔드 개발자

물류/발주/판매자/정산/WMS/광고 플랫폼 개발 및 인프라 관리

- 2023.07 팀장 승진 / 인터널서비스팀 승격
- 2022.03 파트장 승진

주식회사 동그라미소프트

2018.10 - 2021.02 (2년 5개월)

백엔드 / 프론트 개발자

삼성전자 B2B 프로젝트 (GEHS, IRP, ERP)

- 2020.10 대리 승진

기술 스택

Languages	Java 17/21, Kotlin, Python, TypeScript, Node.js, JavaScript
Frameworks	Spring Boot 3.x, Spring WebFlux, Spring Batch, JPA, MyBatis, Hibernate
Streaming	Kafka, Kafka Streams, Schema Registry, MSK Connect, Debezium CDC
Databases	MySQL, MongoDB/DocumentDB, OpenSearch, Redis, Oracle, DB2, PostgreSQL, DynamoDB
Infrastructure	AWS (ECS, EKS, MSK, Batch, RDS, S3, DMS, Lambda, SNS), Terraform, Docker, Linux
DevOps	GitHub Actions, Jenkins, CircleCI, Blue/Green Deployment, GitOps Bridge
Monitoring	Prometheus, Grafana, Sentry, CloudWatch, OpenSearch, Kafka-UI
Frontend	Vue.js, React, HTML/CSS, JavaScript

경력 기술

(주)펫프렌즈 총 8명 팀 리드

2021.03 - 현재 (5년 3개월)

담당: 배송 관리, WMS 창고 시스템, 발주서비스, 판매자서비스, 정산 서비스, 광고 플랫폼, 인프라 관리

운영업무

상시

팀이 담당하는 배송, 물류, 발주, 판매자, 정산, 광고, 인프라 도메인을 상시 운영하고 관리하는 업무

- 배송 관리: 당일배송(심쿵배송), 새벽배송, 택배배송 전체 프로세스 관리
- WMS 창고 시스템 관리: CJ, 팀프레시, JD 등 3PL 연동 및 재고 동기화
- 발주서비스: 매입 상품에 대한 관리자 및 거래처 발주 시스템 관리
- 판매자서비스: 위탁 & 업체 관리자 및 거래처 관리 시스템 관리
- 정산 서비스 관리: 정산 대시보드, 세금계산서 발행, 정산 자동화
- 인프라 관리: AWS 인프라 운영, Terraform IaC, 모니터링 체계 구축
- **AI 코드리뷰 도입**: GitHub Actions 기반 AI 코드리뷰 자동화로 코드 품질 향상 (2024.07)

광고 플랫폼 내재화 프로젝트

2025.07 - 2026.05

외부 광고 플랫폼(CitrusAD) 의존도를 제거하고 자체 광고 시스템 구축

- Engine, Admin, Worker, Batch, Streams 5개 멀티모듈 MSA 서비스 설계
- Co-routine, R2DBC, Reactive Programming 적용
- Kafka Streams 기반 실시간 이벤트 파이프라인 구축
- **Gumbel Top-k Distribution** 알고리즘 기반 확률적 광고 랭킹 설계 (선정 API **설계 목표 ~200ms**, 실측 단건 90~500ms, 인덱스 레플리카 0→2로 **60→40ms(-30%)**)
- OpenSearch 읽기/쓰기 인덱스 분리 설계, 15분 간격 Phase Switching
- CitrusAD → 내재화 플랫폼 Migration Simulation 진행 (15분/1시간 광고 선정 비교)
- OOM 장애 대응: R2DBC Connection Pool 이슈
- CitrusAD **월 700만원** 외부 플랫폼 수수료 **100% 절감**

Java 21 | Kotlin | Spring Boot 3.x | Kafka Streams | OpenSearch | R2DBC | AWS ECS | Terraform

PF-WMS 내재화 (팀프레시 WMS 이관)

2025.10 - 2026.03

팀프레시(TWMS) WMS 코드를 구매해 펫프렌즈 인프라로 이관하여 은봉리(여주) 창고에서 직접 운영하는 메인 창고 시스템. 레거시 현대화와 CJ 택배 직접 연동 교체를 병행 (직접 만든 mini-WMS는 물류 창고 연동용 자체 WMS로 역할 분리)

- **Terraform 기반 AWS 인프라**: 8개 서비스 stage/prod IaC (ECS, RDS, S3, Lambda, Route53)
- **CI/CD 파이프라인 구축**: GitHub Actions 기반 배포 자동화
- **Java 8/11 → 17** (서비스별 상이), Spring Boot 3.x 마이그레이션 145개 파일
- **Maven → Gradle** 멀티모듈 전환으로 **빌드 35초 → 8초(77% 단축)**
- **PF-WMS DB 스키마 107개 테이블** 설계, 시크릿 AWS Secrets Manager 이관

Java 17 | Gradle | AWS ECS | RDS | S3 | Lambda | Route53 | GitHub Actions

MSK Public → Private 서브넷 이관 (ISMS 보안 대응)

2025.11 - 2026.01

ISMS 인증 요구사항 충족을 위해 Public 서브넷에 배포된 MSK 클러스터를 Private 서브넷으로 무중단 이관

- **Terraform 모듈 4종 개발**: msk_cluster, msk_replicator, msk_connector, RDS CDC SecretManager
- **MSK Replicator** 활용 무중단 이관 전략 수립 (기존 클러스터 → Replicator → 신규 클러스터)
- Consumer/Producer 순차 전환으로 **30개 이상 서비스 무중단 이관**
- Kafka 버전 업그레이드 (2.8.2 → **3.8**)
- Migration Playbook 작성 및 Kafka-ui 기반 모니터링 설정

AWS MSK | Kafka 3.8 | MSK Replicator | Terraform | Debezium CDC | Private Subnet

Confluent PoC
2025.06 - 2025.07

Self-managed Kafka(MSK) → Confluent Cloud 전환 검토를 위한 PoC 진행

- Confluent Cloud 계정 생성 및 환경 구성
- 기존 MSK 대비 Confluent Cloud 기능 비교 분석
- Schema Registry, ksqlDB, Kafka Connect 등 관리형 서비스 검토
- 비용 시뮬레이션 및 마이그레이션 전략 수립
- 팀 내 업무 분배 및 PoC 결과 공유

Confluent Cloud | Kafka | Schema Registry | ksqlDB

물류 대행사(JD) 이관 프로젝트
2024.09 - 2025.02

기존 물류 시스템에서 JD 물류 대행사로 전환하는 대규모 프로젝트

- 한진택배(JD) 배송조회 전략 추가 및 송장 트래킹 구현
- WMS-BATCH 입고 조회 배치 개발 및 입고 전송 배치 다건 리팩터링
- 카카오 모빌리티 파일럿 제거로 레거시 시스템 경량화
- 운송사 자체배송 입력부 변경 및 출고완료토픽 연동
- 물류 대행사 전환 프로젝트 안정적 완료

Java | Python | Spring Batch | Kafka

판매자어드민 대시보드
2024.06 - 2024.08

판매자(거래처)를 위한 실시간 대시보드 설계 및 개발

- 피그마 기반 요구사항 분석 및 테이블 설계
- **Server-Sent Events(SSE)** 기반 실시간 데이터 푸시 아키텍처 설계
- SSE vs WebSocket 비교 ADR(Architecture Decision Record) 작성
- 판매 현황, 정산 현황, 재고 현황 실시간 모니터링 기능 구현

Java | Spring Boot | SSE | JPA | MySQL

개발 Kafka 인프라 구축
2024.07 - 2024.08

개발 환경 전용 Kafka 클러스터 및 스트림 처리 인프라 구축

- 개발 환경 Kafka EC2 클러스터 구성
- **ksqlDB** 구성 및 스트림 처리 환경 구축
- Kafka-UI 기반 모니터링 도구 연동
- CloudMap 서비스 디스커버리 추가

Kafka | ksqlDB | Kafka-UI | AWS CloudMap | EC2

AWS Batch 아키텍처 설계 및 전사 도입
2024.03 - 2024.05

EC2 기반 배치 처리를 AWS Batch로 전환하여 비용 최적화 및 리소스 동적 활용 구현

- **AWS Batch 아키텍처 설계**
  - 작업 정의(Job Definition), 작업 큐(Job Queue), 컴퓨팅 환경 구성
  - Fargate Spot 기반 구성으로 온디맨드 대비 비용 절감 (**Spot 최대 70% 할인** 기반)
  - ADR(Architecture Decision Record) 작성 및 전사 공유
- **Terraform 모듈 개발**
  - aws_batch 모듈 개발 (infra/app 분리 구조)
  - WMS, Supply, 검색 등 다중 도메인 적용
  - env-dev.tfvars, env-prod.tfvars 환경별 설정 분리
- **CI/CD 파이프라인 구축**
  - GitHub Actions + ECR + Jenkins 스케줄링 연동
  - Docker Multi-stage Build 적용
  - 검색엔진 배포 시 자동 전체색인 실행 자동화
- **Python 배치 프레임워크 개발**
  - petfriends-aws-batch-plugin (Jenkins 플러그인) 및 aws_batch Terraform 모듈 개발
  - 팀/파트별 AWS Batch 구축 가이드 문서 작성
  - 공통 라이브러리 패키지화 (DB, SecretManager, Kafka, Slack 등)
- **기술블로그 작성**: 서버가 속삭였다 나 그만 괴롭혀 — 펫프렌즈 Batch 인프라 이야기 (펫프렌즈 기술블로그)

AWS Batch | Fargate Spot | Terraform | GitHub Actions | Jenkins | Python | Docker

판매자 심쿵배송 (제트배송)
2023.07 - 2023.11

셀러가 위탁 판매하는 품목을 펫프렌즈 창고에서 풀필먼트 대행하여 추가 수익을 창출한 프로젝트

- 전체적인 테이블 및 업무 설계
- 기획팀 및 물류팀 요구사항 정리
- 팀원 일정 관리 및 코드 리뷰 주도
- 정산 시스템 구축
- 프로젝트 오픈 완료

Java | Spring Boot | JPA | MySQL

Node 배치 선셋
2023.05 - 2023.07

펫프렌즈 Node 레거시 배치 → Python or Java 배치로 이관 (관리되지 않는 레거시 배치 정리)

- Node 배치 분석 및 이관

- 데이터 증가로 인한 인덱싱 문제 개선, 약 **2,000줄 쿼리 로직 분리 및 최적화**
- min(id), max(id) 범위 조건으로 모수 쿼리 최적화
- MySQL In절 slowQuery 해결: In 건수 분할 실행 방식 적용
- 이관 완료

Python Java MySQL Node.js

창고다원화 프로젝트
2023.01 - 2023.06

메인 CJ 물류 창고를 팀프레시 창고로 이관하여 두 개의 물류 창고 사용

- 창고별 재고 관리 및 주문 분배를 위한 **Redis 메타데이터 연동**
- 여러 센터 확장 가능한 Interface 테이블 및 파이프라인 코드 설계
- **Spring Batch 도입**: CI/CD 구축, Jenkins 스케줄링
- 기존 Jenkins 이용하여 스케줄링, SSH 사용하여 다른 서버에서 실행하는 방식 구성
- Jenkins version 너무 낮아 플러그인 업데이트가 안되서 EFS 이용한 버전 업데이트
- 단위 TestCode 샘플 작성
- 확장성 있는 Job 생성을 위한 패턴화 및 패키지 구조 설계
- 오픈 완료

Spring Batch SpringBatchTest Redis Jenkins AWS EFS

데이터 판매를 위한 데이터 이관 (ETL - DBMS)
2023.04 - 2023.05

펫프렌즈의 데이터를 외부로 판매하기 위해 특정 개인정보를 제외한 신규 DB에 데이터 이관 및 실시간 동기화 제공

- **AWS DMS** 이용 데이터 이관 후 CDC 연동 (데이터하우스)
- 특정 개인 정보 및 필드 제외 작업
- DMS Terraform 모듈화 진행
- DMS 실패 시 AWS SNS + Lambda 연동 Slack 알람 생성
- 부하테스트 및 검증, 트러블 슈팅 정리
- 이관 완료

AWS DMS Terraform AWS SNS Lambda Slack

발주어드민, 판매자어드민 고도화 및 리팩터링
2022.11 - 2023.03

발주어드민: 각 거래처에 입고 요청을 하여 입고 확정, 정산까지 진행하는 발주 원스톱 솔루션
판매자어드민: 펫프렌즈에서 거래처 물건을 대신 판매하는 서비스를 위한 시스템 솔루션

- JPA 제대로 사용할 수 있게 코드 개선 및 Service Layer → Domain 구조로 변경
- 전체적인 아키텍처 Package 구조 정의
- Java 버전 및 Gradle 의존성 관리 업데이트 (Java 11 → 17)
- 데이터 수천만 건 증가에 따른 쿼리 튜닝 (**1분 → 5초, 12배 향상**)
- Index 관리 및 서비스 튜닝
- 정산 대시보드 API 개발 (발주/요청/선지급금/거래처 대시보드)
- 150개 거래처로 확대

Java 17 Spring Boot JPA Gradle MySQL

mini-WMS 구축 및 MSA 전환

2022.03 - 2022.10

펫프렌즈 자체 WMS 시스템 구축 / 3PL 연동 및 MSA 아키텍처 전환 / 창고별 자재 및 입/출고 관리

- **업무 프로세스 및 데이터 flow 설계**
  - MSA 위한 신규 Database 도입 및 WMS 도메인 DB 설계 (이벤트 스토밍 DDD)
  - WMS 도입으로 변경된 프로세스 Data Flow 도식화
  - CJ I/F 연동 및 재고 동기화 업무 정리
  - Timf 3PL 시스템과 **9개 센터** 연동
- **CI/CD 전체적으로 수정**
  - GitHub Actions 사용
  - 컨테이너 기반으로 변경 ECS 도입
  - 배포 프로세스 발표 및 도식화
- **DDD Architecture, Test Code 도입**
  - Event Storming 도입 (전사적으로 소통하여 진행)
  - 각각의 연동 서비스들이 들어와도 업무 코드는 변하지 않게 전략 패턴으로 틀을 잡고 진행
- **Spring Event 도입**
  - 각 바운더리 컨텍스트의 의존관계 제거하기 위해 사용
  - MessageQueue, Slack 업무 서비스 영향 없게 진행 용도로 사용
- **MSK Producer, Consumer 설계** (Node, Python, Java)
  - Message Format 공통화 작업
  - Producer, Consumer 실패 전략 수립하여 자동화되도록 개발
  - Prometheus, Grafana, Kafka KOWL 모니터링 도입
  - MSK (Amazon Kafka) 설계 및 개발하여 공통으로 도입할 수 있게 레퍼런스 발표 진행
  - MSK Connect 사용한 CDC (Debezium) 구축
- **Terraform 사용**
  - 서버 관리 및 생성 용도
  - 수동으로 직접 EC2, ALB, AutoScaling 등 만들던 것을 Terraform 사용하여 ECS 모듈화
- **모니터링 도입**
  - Prometheus, Grafana, Kafka-UI, SonarQube 오픈소스 사용하여 새로운 기술에 대한 안정성 및 지표 확보
  - OpenSearch 로그 중앙화
- Swagger 도입
- Spring Boot 2 → 3.2.3, Java 11 → 17, Maven → Gradle Migration

Java 17 | Spring Boot 3.2.3 | Spring Event | JPA | MSK (Kafka) | Terraform | Docker | DynamoDB | MySQL

개발 서버 VPC 변경 작업

2022.01 - 2022.03

레거시 서버 인프라 VPC 이관 작업

- 기존 수기로 만든 인프라 제거
- **작업 순위 1**
  - 기존 EC2 → ECS 이관 Terraform 코드 작성
  - dev vpc용 보안그룹 세팅
  - 로드밸런서 및 오토스케일링 그룹 세팅
  - 업무 서비스 GitHub Actions CI 세팅
  - ECR, ECS 생성 및 배포 테스트
  - 로그 중앙화 수집기(사이드카) 추가
- **작업 순위 2**: 배포, 통신 등 정상 동작 확인
- **작업 순위 3**: 기존 서버 인스턴스, 로드밸런서 등 제거, Route53 → 신규 로드밸런서 변경

- **무중단 이관 완료**

AWS Terraform GitHub Actions Shell Script

발주어드민 오픈 및 유지 보수
2021.07 - 2022.01

기존에는 물류팀과 MD팀이 카카오톡, 메일 등으로 시스템 없이 수기로 발주를 진행하고 있었습니다. 이를 개선해 파트너 상품을 제외한 자체 관리 상품의 발주 등록부터 정산까지 시스템에서 자동으로 처리하도록 구축한 프로젝트입니다.

- 개발팀 스케줄 관리 및 업무 정의
- 전체적인 테이블 설계
- Server Setting
- 발주 등록 및 관리 프로세스 개발
- JWT Token 기반 로그인 프로세스 개발
- Enum 사용하여 공통 코드 관리 진행
- @Transaction 이용하여 Master DB / Slave DB 분리하여 사용 가능하도록 개발
- 엑셀 다운로드/업로드 공통 모듈 개발
- 전체적인 업무 개발자들이 편하게 코드를 짤 수 있게 공통 모듈 개발
- CJ I/F Log DynamoDB 적재
- 오픈 완료

Java Spring Boot JPA Gradle JWT DynamoDB Jenkins Python

펫프렌즈 모놀리틱 → 마이크로서비스 전환
2021.05 - 2021.11

기존에는 mobile, admin, rider, partner, worker 5개 서비스가 하나의 파이프라인으로 묶여 있어, 코드 한 줄만 수정해도 5개 서비스를 모두 배포해야 했고 상용 배포에 20분 이상이 소요됐습니다. 각 서비스를 별도 repository로 분리해 변경된 서비스만 배포하는 방식으로 전환했습니다.

- 5개 서비스 (mobile, admin, rider, partner, worker) 각각 Repository 분리
- 상용 배포 시간 **20분 → 4분으로 단축**
- 기존 배포 방식 정리 및 사이드이펙트, 코드 배포 레퍼런스 확인
- 새로운 배포 방식에 맞는 AWS 이용한 서버 환경 구성
- Docker 설정 및 CircleCI 배포 설정 파일 수정
- AWS CodeDeploy 연동
- 5개 중 1개 (Admin) 분리 완료, 사이드 이펙트 없음

Node.js (Koa) CircleCI CodeDeploy Docker GitHub

펫프렌즈 바디챌린지 (입사 프로젝트)
2021.03 - 2021.05

자체 라이더 → 심쿵배송(당일 배송) 전체적인 프로세스 변경 / CJ I/F 물류 연동

- 라이더 앱 API, 어드민 페이지 API, CJ 연동 개발 및 유지보수
- 어드민 페이지 (Node.js) 개발
- 운송사 관리, 라이더 관리, 배차화면 등 관리 툴 페이지 개발
- ERD 및 업무 프로세스 도식화 진행
- 히스토리가 하나도 남은 게 없어서 Notion 문서 정리

Node.js (Koa) Python CircleCI Jenkins MySQL AWS

주식회사 동그라미소프트

2018.10 - 2021.02 (2년 5개월)

삼성전자 B2B 프로젝트 전문 SI 기업에서 MES/ERP 시스템 개발

삼성전자 GEHS 2단계 (Global Environment, Health and Safety)

2020.03 - 2021.02

공통, 화학물질, 안전방재, 국내보건, 해외보건 4개 모듈 클라우드 서비스

- Front window.SessionStorage 사용하여 검색조건 유지 개발
- Vue.js 공통 컴포넌트 개발
- ExcelDownLoad, ExcelUpload Java 공통 서비스 개발
- 업무에서 OzReport Tool 사용 시 data log BackUp 개발
- Service 트랜잭션 발생 시 application 레벨에서 실행 전 query 파싱 후 이전 데이터 백업 개발 (MyBatis Plugin 사용)
- SA 서버 관리 보조 (git Branch를 개발, QA, SPD(운영 전 서버), 운영으로 나눠서 관리)
- 화학물질, 안전방재 (업무): Front 화면 개발, Back-end MVC 패턴 서비스 로직 개발
- 배치 및 메일링 개발
- ERD 및 DB 설계 및 수정
- 기존 시스템에서 신규 시스템으로 **최대 3천만 건** 데이터 이관
- 삼성 내부 결재(Knox) 및 후처리 개발

Vue.js | Velocity | Node.js | Java | Spring Boot | Spring Batch | MyBatis | DB2 | Jeus | Jenkins

삼성전자 제품환경서비스 프로젝트

2020.02

새로운 Login Process 개발

- 기존 Login Logic (Knox Login) + New Login Logic (AD Login) 개발
- IE와 Chrome 구별하여 각각의 예전 로그인, 신규 로그인 탈 수 있게 개발
- API 호출 후 CallBack 데이터를 받아와서 로그인 로직 처리

애니프레임워크 | Java | XML | Retrofit2 | DB2

삼성전자 GEHS 1단계 (Global Environment, Health and Safety)

2019.04 - 2020.01

공통, BQMS, FLMS (ERP 시스템) 패키지화

- FLMS (Facility LifeCycle Management System) 설비 수명 관리 시스템 개발
- Front 화면 개발, Back-end MVC 패턴 서비스 로직 개발
- 배치 및 메일링 개발
- 삼성 내부 결재(Knox) 및 후처리 개발

Vue.js | Velocity | Node.js | Java | Spring Boot | Spring Batch | MyBatis | Oracle | DB2 | PostgreSQL

삼성바이오로직스 ERP 관리 시스템

2019.03

삼성 바이오로직스 ERP 시스템 개발

- 넥사크로(Front Tool) 개발

- 애니프레임워크, Java, XML 기반 개발
- 사양서 작성

넥사크로 Java 8 Spring 애니프레임워크 MyBatis DB2

삼성전자 IRP (IT투자 / 운영비용 관리체계 개선)
2018.10 - 2019.03

IT투자 / 운영비용 관리체계 개선

- Velocity 이용한 화면 개발
- Hibernate 사용하는 서비스 로직 수정

Java Spring Hibernate Velocity MyBatis DB2

주요 트러블슈팅 & 아키텍처

운영 환경에서 마주친 장애와 병목을 어떻게 분석하고 해결했는지, 그 과정에서 만든 아키텍처를 장애 대응, 성능 최적화, 아키텍처로 나누어 정리했습니다.

장애 대응

장애 RCA, 장애 격리

노출 API OOM이 반복돼도, 광고 선정은 멈추지 않은 이유

노출 트래픽이 2배로 튀자 노출 API에 OOM이 발생해 광고 엔진 태스크가 **50~62분 주기로 반복 재시작**됐습니다. 노출 이벤트는 지연되고 일부 누락됐지만, 광고 선정은 OpenSearch와 Redis만으로 돌도록 DB 의존을 끊어 둔 덕분에 선정과 응답은 정상이었습니다.
장애 RCA를 주도해 OOM의 원인을 규명했습니다. 직접 원인은 노출 수집 경로의 R2DBC 커넥션 풀이었습니다.

- 노출 중복검증이 Redis(SETNX) 미존재 시 R2DBC로 DB 조회
- 다수 광고 ID를 무제한 병렬 조회 → 동시성 제한 세마포어가 주석 처리로 비활성
- R2DBC 풀에 획득 타임아웃 미설정 → 커넥션 충돌이 무한 대기 → 힙 누적 → OOM

복구와 재발 방지, 그리고 역할 구분.

- 실제 복구 = 엔진 태스크 재시작 (캐시 재투입은 이미 OOM이라 무효) → 공식 MTTR **2시간 48분**, 인지~복구 **10시간 19분**
- 본인 기여 = RCA 주도로 원인 규명 + 풀 사용률 메트릭과 그라파나 대시보드 추가 (풀 설정 재조정은 팀)
- 재발 방지 → 메모리 **2GB → 4GB** 증설 + GC/Heap 모니터링과 알림
- 격리 설계 → 광고 선정은 DB 비의존이라 무영향, 노출 이벤트만 지연

성능 최적화

인프라, 비용

스토리지 등급 변경으로 배치 처리 시간 88% 단축

수백만 건 데이터 인입 배치가 디스크 입출력에서 병목이 걸려 처리 시간 편차가 컸습니다. 스토리지 등급을 **한 단계 상향**하고, 적용 전후를 일별 로그로 직접 비교 측정했습니다.

지표	변경 전	변경 후
평균 소요시간	17.6분	**2분 (-88.7%)**
건당 처리시간	24.9ms	**2.6ms (-89.7%)**
최대 소요시간	1,306초	**152초 (-88.4%)**

쿼리 튜닝, 무중단

복합 인덱스 설계로 느린 정렬 쿼리 개선 (무중단 적용)

지갑 차감 내역 조회가 단일 인덱스로 느렸고, 정렬에서 파일 정렬(filesort)이 발생했습니다. 조회 조건과 정렬 컬럼을 모두 포함한 **복합 인덱스**로 교체해 정렬까지 인덱스로 처리하도록 했습니다.

실행 계획(EXPLAIN)으로 병목을 먼저 확인한 뒤, 조건과 정렬 컬럼 순서를 맞춰 복합 인덱스를 설계했습니다. 인덱스 추가는 운영 트래픽에 영향이 가지 않도록 락 없이 온라인으로 적용했고, 스테이징에서 실행 계획을 다시 확인해 파일 정렬이 사라진 것을 검증한 뒤 상용에 반영했습니다.

아키텍처

무중단 마이그레이션

30개 이상 서비스를 무중단으로 옮기되, CDC만 다르게 다룬 이유

보안 요건을 맞추려고 메시지 브로커(Kafka)를 외부 노출 구간에서 내부 구간으로 옮겨야 했습니다. **30개 이상의 서비스**가 물려 있어 중단이 곧 매출 손실이었고, 같이 걸린 Kafka 메이저 업그레이드(본인 확인 기준 3.8)도 순단 없이 끝내야 했습니다.

사전 검증에서 일반 토픽은 복제로 무중단 전환이 되지만, 변경 데이터 캡처(CDC) 커넥터는 재시작 시 오프셋을 이어받지 못해 **데이터 누락이 구조적으로 발생**한다는 점을 확인했습니다. 그래서 CDC만 분리해 위험을 격리했습니다.

- 30개 이상 서비스 무중단 이관 → 일반 트래픽 다운타임 0
- Kafka 메이저 업그레이드(3.8) 병행, 이관 절차는 Terraform으로 코드화해 재현 가능하게 정리

인프라 설계, 비용 최적화

Jenkins 단일 서버 한계를 AWS Batch 분산으로 넘긴 이유

Jenkins 단일 서버에서 배치를 돌렸는데, 서비스가 커지며 하루 **약 6,400건**(Peak Time 병렬 약 40건)까지 늘었고 언어도 Node에서 Python, Java로 확장됐습니다. 특정 시간대에 **CPU와 메모리가 100%**까지 치솟아 리소스 알림이 빈번했고, 리팩터링이나 서버 증설로는 한 서버에서 수십에서 수백 개 Job을 동시 실행하는 한계를 넘지 못했습니다.

그래서 배치 실행을 AWS Batch로 분산했습니다. 공식 Jenkins 플러그인이 6년간 미업데이트 상태라, AWS Batch 제출과 로그 polling, 재시도, 취소, 알람을 직접 구현한 **자체 Jenkins 플러그인(Java 11)**을 개발했습니다.

- AWS Batch 3요소(Job Definition, Job Queue, Compute Environment)로 구조를 잡고, 컴퓨팅은 Fargate Spot으로 병렬 실행
- aws_batch Terraform 모듈을 infra와 app 분리 구조로 개발하고 dev와 prod tfvars를 나눠 재현 가능하게 표준화
- 런타임 1시간 이상이거나 중단 불가 워크로드는 온디맨드로 분리하고, 30분 이내이면서 중단 허용인 워크로드만 Spot으로 보냄

- 하루 **약 6,400건** 배치 → 단일 Jenkins 서버에서 AWS Batch로 분산 → **CPU 100% 병목 해소**
- 중단 허용 워크로드만 Fargate Spot → 할인 **최대 70%**
- WMS, Supply, 검색 **3개 도메인**에 적용

서버가 속삭였다 "나 그만 괴롭혀" — 펫프렌즈 Batch 인프라 이야기

핵심 성과

비용 최적화

- 광고 플랫폼 내재화로 CitrusAD 외부 수수료 **월 700만원 절감**
- AWS Batch 아키텍처 설계 및 Fargate Spot 도입으로 배치 인프라 비용 절감 (**Spot 최대 70% 할인** 기반)
- Terraform 모듈화로 인프라 프로비저닝 시간 단축 및 휴먼 에러 제거

실시간 데이터 처리

- Kafka Streams 기반 광고 노출/클릭 이벤트 실시간 파이프라인 구축
- Gumbel Top-k Distribution 알고리즘 기반 광고 선정 API **설계 목표 ~200ms** (실측 단건 90~500ms, 인덱스 튜닝 60→40ms)
- mini-WMS에서 **340,000건+** 재고 레코드 실시간 동기화

무중단 마이그레이션

- MSK Public → Private 서브넷 이관 시 MSK Replicator 활용, **30개+ 서비스 다운타임 0** 전환
- Kafka 2.8.2 → 3.8 메이저 버전 업그레이드 병행

- 모놀리틱 → MSA 전환으로 배포 시간 **20분 → 4분 (80% 단축)**

성능 최적화

- 발주어드민 쿼리 튜닝으로 수천만 건 데이터 조회 **60초 → 5초 (12배 개선)**
- 인덱스 재설계 및 N+1 문제 해결
- Node 레거시 배치 **2,000줄 쿼리** 분석 후 Python/Java로 이관, min/max id 범위 조건 최적화

아키텍처 설계 및 표준화

- mini-WMS DDD 기반 **107개 테이블** 설계, **9개 물류센터** 3PL 연동
- Terraform 모듈 표준화 (ECS, MSK, Batch, DMS 등) 및 전사 IaC 가이드 수립
- 자체 Jenkins 플러그인(petfriends-aws-batch-plugin) 개발로 팀별 배치 시스템 자체 구축 지원

팀 리더십

- 인터널서비스팀 팀장으로서 **8명 규모** 팀 리딩
- 광고, WMS, 배송, 발주, 판매자, 정산 **6개 도메인 병렬 관리**
- AI 코드리뷰 도입으로 리뷰 품질 향상 및 개발 생산성 개선

학력

세명대학교

소프트웨어학과, 컴퓨터시스템학과 (복수전공)

2013.03 - 2018.02

- 1학년: 학생회
- 2학년: 과 대표
- 3학년: 과 대표
- 4학년: 취업 및 사회 생활을 준비하기 위해 반 학기 휴학 후 졸업

자격증 및 수상

NCS 국가직무능력표준 (오픈소스 기반 웹 전문가)

- 개인프로젝트 **대상** (2018.05)
- 팀프로젝트 **우수상** (2018.09)

최종 업데이트: 2026-06-19