Jinho Bae

Backend Engineer · DevOps Engineer

(+82)10-4028-0249 onionring4028@gmail.com Baejinho4028

KOIN (대학 생활 지원 앱) - DAU 1,300+, 점유율 85.7%

Spring Boot · Java · MySQL · Nginx · Datadog / 2024.03 - Present

- 데이터 중심 의사결정 ⇒ **기능 사용률 2배 증가** 2p
- VPC 내부 통신 전환 ⇒ **운영 비용 73% 절감** 3p
- 운영 비용을 고려한 서비스 수준 목표 ⇒ **가용률 99.9% 유지** 4p
- 팀 업무 투명화 ⇒ **커뮤니케이션 비용 93% 감소** 5p

AI Voice Call (통신사 AI 전화 상담 솔루션) - 월 15만 건 상담 자동화

Spring Boot · Java · Asterisk PBX · MySQL · Docker / 2025.12 - 2026.02

- AI 전화 상담 자동화 ⇒ **상담원 의존도 85% 감소** 6p

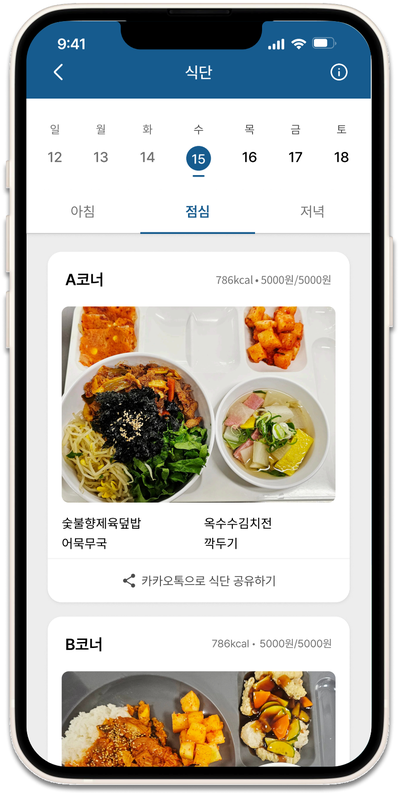
“Operate systems based on data and operational costs.”

데이터 중심 의사결정 ⇒ 기능 사용률 2배 증가

역할 : Webp 작업 분배, HTTP/2 Multiplexing 과 gzip 적용 (기여도 40%)

대학 생활 지원 앱
DAU 1,300+ / 점유율 91.6%

Summary



이미지 로딩 시간

4s → **0.25s**

↓ 94% 감소

기능 DAU

556 → **1,173**

↑ 111% 증가 (약 2배)

점유율

85.7%

대학 필수 앱

Issue 1

Webp 변환 및 압축으로 전송 속도 개선

APM에서 해당 기능의 일일 사용자 수가 732 → 556명으로 감소하는 것을 식별했습니다.
기존에 제공하던 학식 메뉴 조회 기능에서 메뉴 사진을 추가 제공했지만, 이미지 로딩이 느렸습니다.
이미지 전송 속도를 높이기 위해 Webp 변환 및 압축하여 용량을 2.8MB → 200KB로 줄였습니다.
Orientation 메타데이터로 회전값을 고정하고, 반응형 UI 대응으로 여러 width 저장했습니다.
초기 배치 처리는 과도한 리소스 낭비로 Lambda@Edge 기반 on-demand 방식을 적용했습니다.

Issue 2

HTTP/2 Multiplexing 및 gzip 압축으로 외부 네트워크 병목 제거

이미지 용량 문제 외에도 리소스 다운로드 시간으로 인하여 TCP 연결 초과 병목이 발생했습니다.
gzip 압축으로 다른 리소스의 다운로드 시간을 줄여 지연을 제거하여 병목을 완화했습니다.
또한, HTTP/2 Multiplexing을 적용하여 단일에서 병렬 연결로 전환했습니다.
Queueing Time 434ms → 22ms, Content Download Time 255ms → 23ms

Limitations

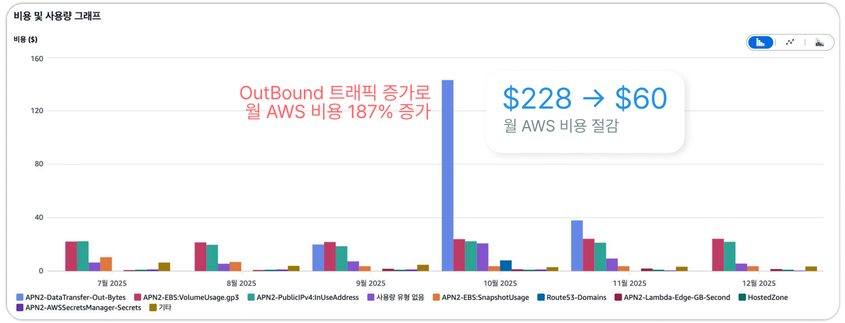
이미지 로딩이 이탈의 원인인지 검증하기 위해 A/B 테스트를 구축했습니다.
그러나 사용자 수 감소 이후, 유의미한 결과를 도출하기에는 한계가 있었습니다.
이를 통해 실험 기반 검증이 가능한 사용자를 갖추는 것이 중요하다는 것을 느꼈습니다.

VPC 내부 통신 전환 ⇒ 운영 비용 73% 절감

역할 : 문제 식별, Private IP 작업 지시, 접근 통제 (기여도 30%)

대학 생활 지원 앱
DAU 1,300+ / 점유율 91.6%

Summary



운영 비용

$228 → **$60**

↓ 73% 감소

아웃바운드 트래픽

26.9MB → **0.3MB**

↓ 99% 감소

Issue 1

Private IP 기반 통신으로 네트워크 비용 절감

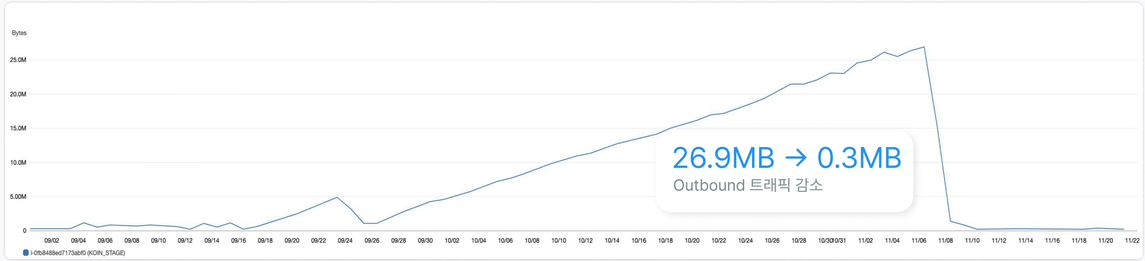
AWS 운영 비용이 $79 → $228로 증가했고, 서비스의 확장에 비해 높은 비용이 발생했습니다.
모니터링 지표를 확인한 결과, Outbound 트래픽이 점차 증가하는 것을 식별했습니다.
레거시 서비스 간 통신이 Public IP로 이루어져 잦은 배포로 인해 트래픽이 증가했습니다.
서비스 간 통신과 배포 경로를 VPC Private Network 기반으로 일원화했습니다.
이렇게 불필요한 Outbound 트래픽으로 인한 네트워크 비용을 줄이고, 보안도 강화했습니다.

Issue 2

접근 통제 및 보안 강화

auth.log 분석 결과 DA팀의 무분별한 초당 5건 이상의 접근이 확인됐습니다.
그래서 DB는 Public endpoint 대신 Private IP + SSH 터널링으로 접근 가능하도록 했습니다.
AWS IAM, 서버 계정, DB 계정을 역할별로 분리하여 최소 권한 원칙을 적용했습니다.
또한, 비정상적인 접근을 차단하기 위해 Fail2Ban을 적용했습니다.

Outbound Traffic (Bytes/days)

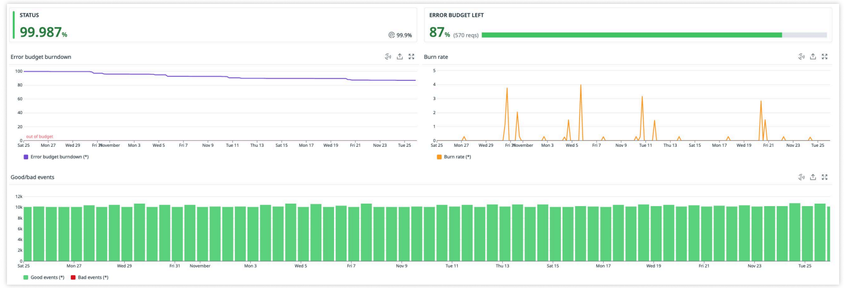


운영 비용을 고려한 서비스 수준 목표 ⇒ 가용률 99.9% 유지

역할 : SLO 도입, 무중단 배포 적용, Presigned URL 작업 할당 (기여도 70%)

대학 생활 지원 앱
DAU 1,300+ / 점유율 91.6%

Summary



Datadog SLO

가용률

95.6% → **99.98%**

99.9%↑ 유지

배포 다운타임

120초 → **0초**

무중단 서비스

Issue 1

SLO ≥ 99% 으로 정량적 목표 설정

서비스가 성장하며 모든 시간에 사용자가 존재하는 서비스가 되었습니다.
이로 인해 안정성이 중요해졌고 서비스 수준 목표 같은 정량적 기준이 필요했습니다.
무작정 고가용성을 추구하면 운영 비용과 복잡도가 증가할 수 있어 SLO ≥ 99% 를 설정했습니다.
Error Budget Alerts보다 필요할 때만 조치를 취하기 위해서 Burn Rate Alerts를 선택했습니다.

Issue 2

무중단 배포 적용으로 다운타임 제거

단일 WAS 환경에서 배포마다 120초간의 다운타임이 발생했습니다.
사용자 신뢰성을 위해 사용자가 적은 새벽에 배포를 진행하면서, 팀원들의 피로가 누적되었습니다.
무중단 배포 중 서비스 규모와 비용을 고려하여, Blue/Green 배포를 적용했습니다.
트래픽을 전환하는 Graceful Shutdown과, 파일 교체 과정에서의 원자성 문제도 보완했습니다.

Issue 3

Presigned URL로 OOM 방지

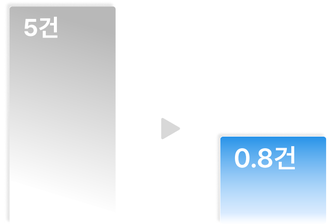
이미지 업로드를 서버에서 처리하며, 피크 타임마다 Out Of Memory가 발생했습니다.
Serverless 기반의 Presigned URL 업로드를 적용하여 부하를 제거했습니다.
또한, 파일 위변조를 방지하기 위해 이미지 서명을 검증하기로 했습니다.
Presigned URL 생성 시 Content-Length와 Content-Type을 포함해 서명하도록 구현했습니다.

팀 업무 투명화 ➔ 커뮤니케이션 비용 93% 감소

역할 : 문제 식별, 이슈트래커 및 Figjam 도입 (기여도 100%)

대학 생활 지원 앱
DAU 1,300+ / 점유율 91.6%

Summary



QA 누락

5건 → **0.8건**

↓ 93% 감소

스프린트 달성률

78% → 92%

14%p ↑

Issue 1

이슈트래커 도입으로 티켓 중앙 집중 관리

이슈나 작업 진행 상황 내용이 여러 채널에 파편화 되어있었습니다.
서로의 작업을 확인하는 과정이 반복적으로 발생했고, 이는 똑같이 노이즈가 되었습니다.
Jira는 너무 무겁고, 비용이 많이 발생하여 Linear 이슈트래킹 도구를 도입했습니다.
작업을 티켓 단위로 분리하고 우선순위와 진행 상태를 중앙 집중형으로 관리했습니다.

Issue 2

Figjam 기반 회의 시각화 및 데일리 스크럼으로 피드백 속도 개선

기능 흐름이 시각화 되지 않아 QA에서 평균 5건 이상의 기능 누락이 발생했습니다.
또한, 주간 회의의 경우 원격으로 진행되어 피드백 주기가 길었고, 이는 곧 병목으로 작용했습니다.
그래서 FigJam으로 요구사항 일치를 위해 기능흐름 및 설계를 시각화했습니다.
또한, 데일리 스크럼으로 막힘과 우선순위를 짧은 주기로 공유하여 피드백 속도를 높였습니다.

Limitations

이슈트래커 도입 전 개발 및 비개발 직군의 협업 지표를 정량적으로 측정하지 못했습니다.
객관적 데이터가 부족해 문제 정의와 해결 방안이 주관에 의존했습니다.
향후 이슈트래커 데이터를 기반으로 협업 지표를 정량적으로 수집·분석할 계획입니다.

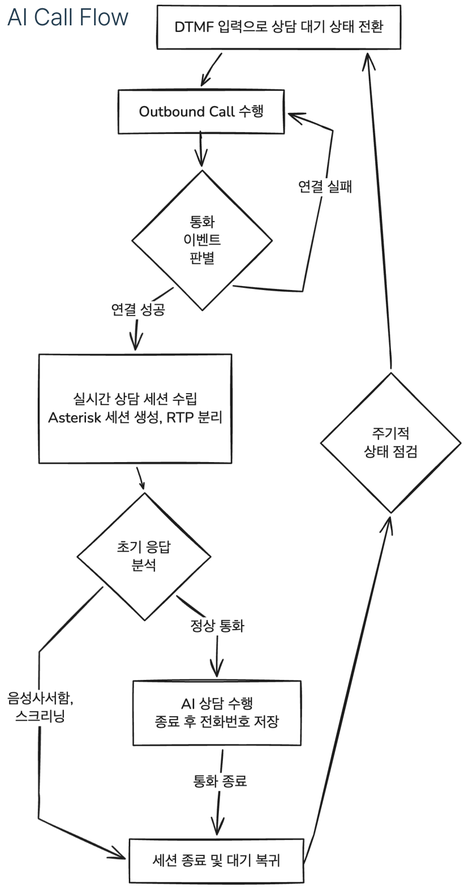
AI 전화 상담 자동화 ⇒ 상담원 의존도 85% 감소

역할 : Asterisk 및 백엔드 구축. RAG와 TTS 최적화 (기여도 60%)

AI Voice Call
월 15만 건 상담 자동화

Summary

AI Call Flow



응답 시간

5.8s → **1.6s**

↓ 72% 감소

상담원 처리량

↓ 85%

15만 → 2만

이탈률

90% → **31%**

동시 30건 처리

Issue 1

AX로 상담원 인력 의존 축소

상담사 인력에 의해 상담 처리량이 의존되어 월 2만 건까지만 상담이 가능했습니다.
인력 증원의 한계를 극복하고자 AI를 중심으로 상담 시스템을 전환했습니다.
통화 제어, SIP 세션제어, Call Routing 을 위해 VoIP 기반 PBX가 필요했습니다.
비용, 레퍼런스, 라우팅 제어 자유도 등을 고려하여 Asterisk PBX로 구축했습니다.

Issue 2

응답 품질 및 속도 보장

영문 중심 TTS 모델의 부자연스러운 한국어 발화로, 2~4초 사용자 이탈률이 70% 발생했습니다.
Voice Cloning과 튜닝으로 발화 품질을 높였지만, 응답 지연 시간이 늘었습니다.
5초 이상 지연 시 이탈률이 90%였기에, 외부 API 의존을 제거하여 Latency를 최소화했습니다.
그리고 RAG 구조에서 Vector DB 대신 애플리케이션 메모리 기반으로 컨텍스트를 관리했습니다.
또한, TTS 문장 단위 분할로 병렬 생성하여 최대 2초 이내 처리를 보장했습니다.

Insights

서비스의 신뢰성은 평균 성능이 아니라 최악의 사용자 경험에서 결정됩니다.
특히 실시간 전화 서비스에서는 이러한 경향이 더욱 크게 나타난다는 것을 배웠습니다.
그래서 사용자 경험을 저해하는 핵심 병목을 최우선으로 개선하여 신뢰성을 높였습니다.