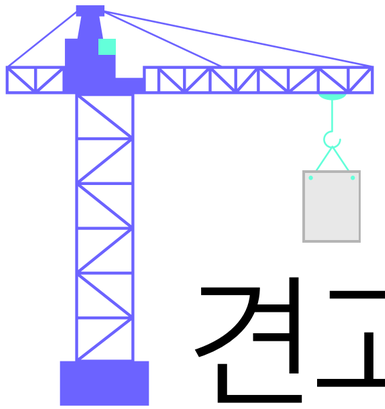
PORTFOLIO

2025

FOUNDATION · CONSISTENCY · RELIABILITY



견고한 기초에서 시작해
높은 확장성으로 나아가다

CONTACT

PHONE : 010 6483 5892 | MAIL : rkddkwl059@naver.com

송진우

송진우 (1997.04.19) | https://github.com/openSongce

PORTFOLIO

INTRODUCTION

기본기가 **튼튼**해야 **서비스**가 무너지지 않는다.

저는 **기반**을 다지고, 그 **위**에 **확장** 가능한 **서비스**를 올리는 **개발자** 입니다.

송진우 Song-Jin-Woo

/ 1997.04.19

010-6483-5892

rkddkwl059@naver.com

https://github.com/openSongce

학력 Education	2013.03 - 2016.02	경북고등학교 졸업
	2016.03 - 2025.02	영남대학교 기계공학부 기계설계 졸업
		복수전공 컴퓨터 공학 졸업
이력 Background	2023.03 - 2024.12	YEUNGNAM UNIVERSITY CONERGENCE S/W COURSE
	2025.01 - 현재	삼성청년SW·AI아카데미(SSAFY) 13기
자격 Certifications	2025.03.15	OPIC 영어 INTERMEDIATE MID 1
	2025.06.13	정보처리기사

어떤 역량을 가지고 있을까

| 스킬 및 역량

Java & Kotlin (Spring / Android) 80%

MySQL · PostgreSQL 85%

협업 및 코드리뷰 80%

Jenkins · Docker · k8s · AWS 85%

Prometheus · Grafana 75%

CI/CD 구축 및 운영

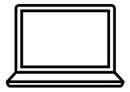
Jenkins + Kaniko 기반 **CI/CD** 파이프라인 구축 경험

Docker 이미지를 자동 빌드·배포하고, **k3s** 클러스터에 **Helm**으로 롤링 업데이트 운영

NHCafe(AWS EC2에 수동 빌드·재실행 배포 경험),

BookgleBookgle(Compose 배포, AWS EC2), **HelloWorld**(k3s 이전/관찰성, AWS EC2)

백엔드 개발 경험



Spring Boot + JPA, REST/gRPC API 설계 및 구현

AWS S3 + Presigned URL 기반 미디어 업로드/다운로드 API

OpenAI DALL·E-3 연동, 캐리커처 생성 후 이미지 저장

사용자 일기(텍스트·이미지) 및 미디어 저장소 설계

BookgleBookgle(gRPC 동기화), **HelloWorld**(S3 + DALL·E-3 API, 일기/사진 저장, MSA)

Observability 구축

Prometheus + Grafana 기반 지표 모니터링

Loki + Promtail 로그 수집 파이프라인

알람 규칙 기반 자동 탐지 및 대응 프로세스 단축

BookgleBookgle (기본 대시보드 구성 및 성능 지표 추적),

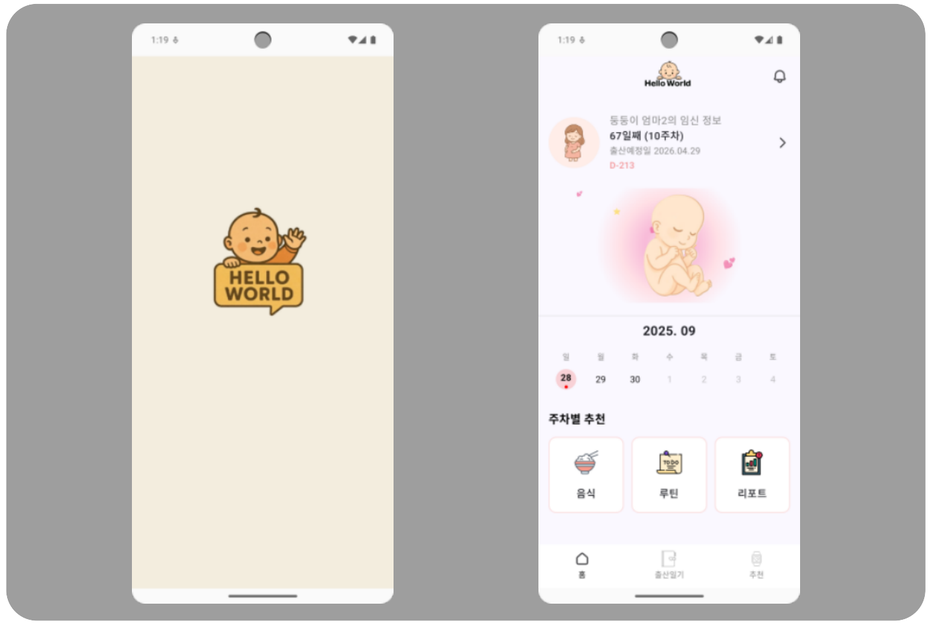
HelloWorld (Prometheus/Grafana 모니터링, Loki 로그 수집)

PORTFOLIO

INDEX

2090-2095

배포·운영 자동화로 안정성 지표를 개선



기획배경

한국 산후우울증 유병률 **24%** 이상, 관리·지원 서비스 **부족**

실시간 모니터링 과 **예방 중심 관리** 필요성 **대두**

웨어러블·AI·모바일을 결합한 **맞춤형 케어 서비스** 로 문제 해결을 **목표**

임산부 케어 & 빅데이터 추천 서비스 # HelloWorld

출산 이후 산모와 가족이 겪는 정신적 부담(산후우울증, 스트레스 등)을 완화하기 위해,

웨어러블 데이터 기반 맞춤형 케어 와 **가족 연동형 기록 서비스** 를 제공하는 앱

빅데이터 기반 추천과 AI 연동으로 **개인화** 된 케어 경험을 지원

내 역할 기간 : 25.08-25.09 | 기여도 : 20%

#Jenkins #Kaniko #k3s #Helm #Traefik #cert-manager #Prometheus #Grafana #Loki #S3 #DALL-E-3

팀 구성: Android 2 · Backend 2 · Big Data 1 · **나(DevOps/Backend)**

CI/CD: Jenkins + **Kaniko** 자동 빌드/배포, 롤백

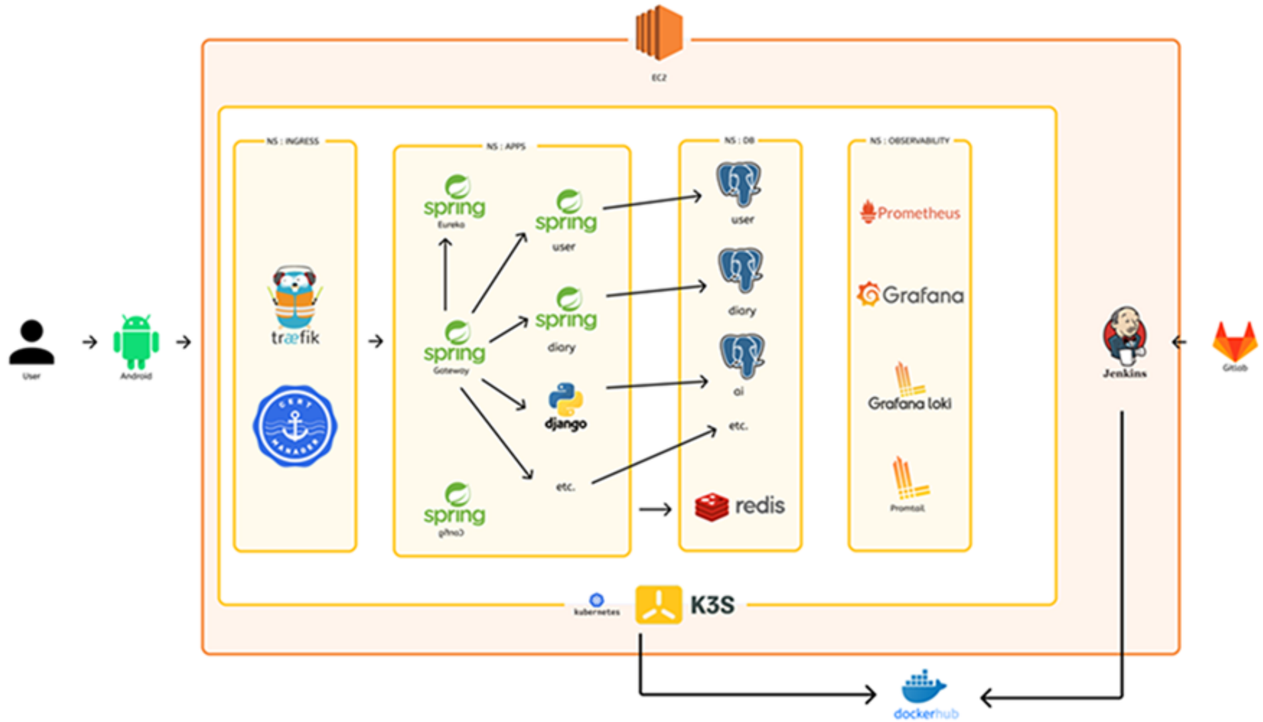
Cluster: k3s on **EC2**, **Helm**, **Traefik** + **cert-manager**

Observability: Prom / **Grafana** / **Loki** / **Promtail**

API: S3 Presigned URL, **DALL·E-3** 생성/저장

Impact: 배포시간 △%, 실패율 ▽%, MTTR △__%

	Declarative: Checkout SCM	Checkout	Kube debug	Detect changed (code/k8s/ingress)	Docker build & push (Kaniko)	Ensure namespace (optional)	Deploy to k3s (services)	Deploy to k3s (ingress)	Declarative: Post Actions
Average stage times: (full run time: ~4min 52s)	16s	1s	1s	2s	2min 52s	2s	1min 19s	0ms	815ms



배포 파이프라인 한눈 요약

GitLab 푸시 → **Ephemeral K8s Agent** 생성

변경 감지(코드/매니페스트/Ingress) → 빌드·배포 타겟 최소화

Kaniko 로 **도커이미지** 빌드/푸시 (<svc>-<gitSha>, <svc>-latest)

k3s(EC2)로 서비스 RollingUpdate + rollout status 확인

Ingress 변경 분만 별도 apply

결과는 **GitLab Commit Status** 로 즉시 피드백

설계 이유

MSA: 서비스가 많아 → 변경된 서비스만 빌드/배포해야 시간·리스크↓

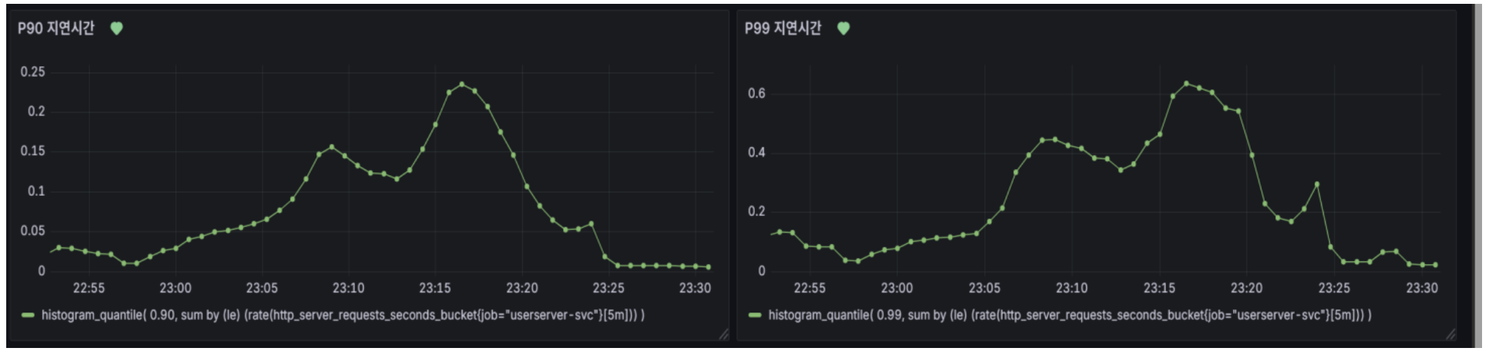
Kaniko: K8s 내부 daemonless 빌드(Docker 데몬 없이 안전/빠름)

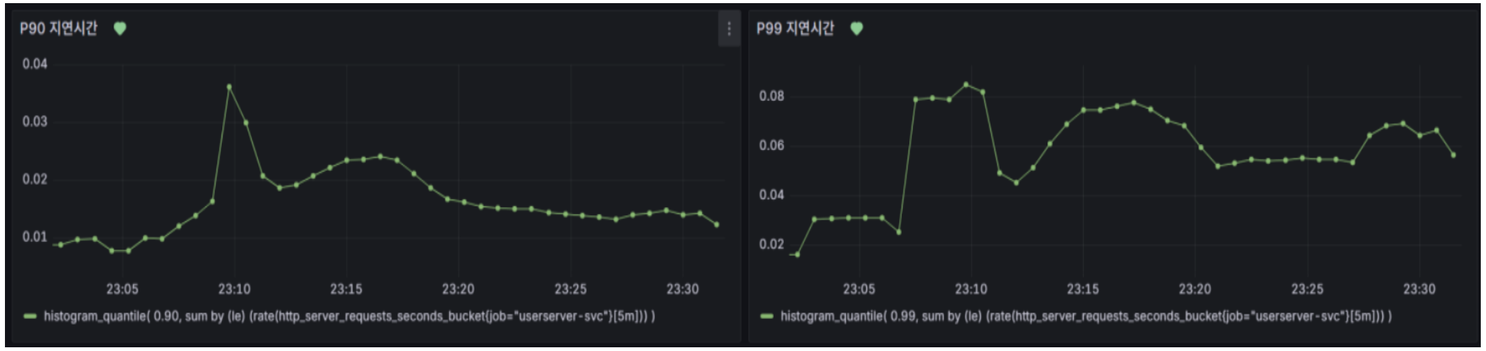
k3s on EC2: 가벼운 K8s로 학습/운영 비용↓ + 표준 배포 경험 유지

Traefik + cert-manager: TLS 자동(Let's Encrypt)·헤더 정합 쉬움

무중단/복원력: Readiness/Liveness, PDB/HPA/Anti-Affinity로 장애 전파 최소화

Observability: Prom/Grafana/Loki로 지표·로그를 하나의 대시보드에서 추적





지연율 단축

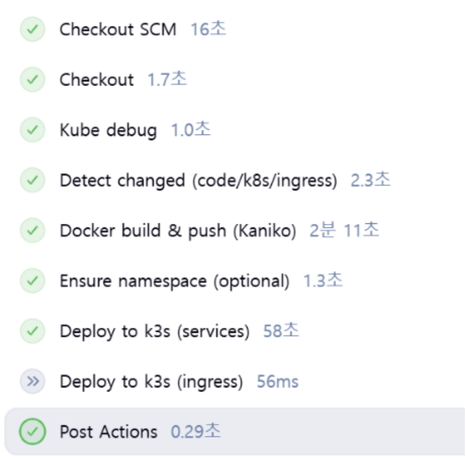
배포 전략을 replicas=3 → 2, maxSurge=1 / maxUnavailable=0로 조정해
롤링 중 메모리 압박 해소
스왑 활성화로 급격한 OOM을 완화, GC 스톨/스케줄링 지연 감소

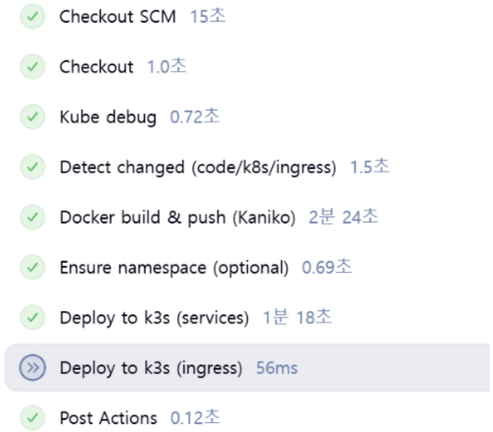
P90: 피크 시 약 0.20~0.24s → 0.01~0.02s 수준으로 하락 (≈ 10배 내외 개선)
P99: 피크 시 약 0.5~0.6s → 0.03~0.04s로 감소 (≈ 12~15배 개선)
롤링 구간 이후 기저(베이스라인) 지연도 안정적으로 낮게 유지

성과 지표(KPI) 및 회고

```
ubuntu@ip-[redacted]:~/yamldir$ free -h
               total        used        free      shared  buff/cache   available
Mem:            15Gi        14Gi       282Mi        99Mi       1.3Gi       1.1Gi
```

```
ubuntu@ip-[redacted]:~/yamldir$ free -h
               total        used        free      shared  buff/cache   available
Mem:            15Gi        12Gi       1.6Gi        99Mi       1.7Gi       2.8Gi
Swap:          2.0Gi       1.3Gi       677Mi
```

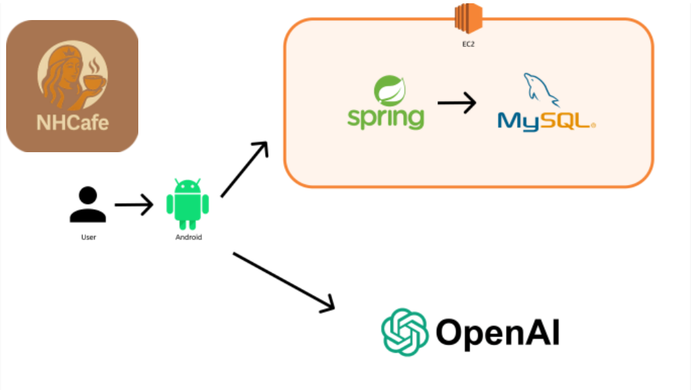




문제해결 액션	• **REPLICA 3 → 2로 조정:** 롤링 중 동시 유지해야 하는 파드가 많아 메모리 압박 → 노드(EC2) 먹통 사례 발생. • REPLICAS=2, MAXSURGE=1 / MAXUNAVAILABLE=0로 바꿔 무중단은 유지하면서 메모리 여유 확보. • **스왑 활성화:** 예기치 않은 피크 시 OOM 대신 완만한 성능 저하로 방어. • **결과**: 사용 가능 메모리 1.1 GIB → 2.8 GIB, 롤링 시간(SERVICES) 1M 18S → 58S.
생산성 증가 액션	• **변경 감지 기반 배포:** 코드/매니페스트/INGRESS 단위로 바뀐 서비스만 빌드·롤링 → 불필요 작업 제거. • **KANIKO IN-CLUSTER 빌드:** 빌드 노드 관리 없이 일관된 속도/보안 확보. • **결과:** 만약 적용하지 않았다면 항상 모든 서버를 재빌드하기 때문에 서버 하나 당 평균 2M의 시간이 증가되어 작업 효율이 낮아짐
핵심 회고	• **가용성은 복제 수만으로 보장되지 않는다.** REPLICA·ROLLING 전략·노드 자원이 함께 설계되어야 안전하다. • **기본이 성능이다.** READINESS/LIVENESS, PDB/HPA, 리소스 한계선을 먼저 정리해야 배포 자동화의 이득이 제대로 나온다. • **MSA는 선택과 집중이 성능.** 변경 감지로 작게/빨리 배포하는 게 전체 체감 속도를 크게 올린다. • **병렬 빌드/롤링 도입(제한형):** 서비스 N개 동시 처리(예: 2~3개) + 동시성 가드(큐/세마포어)로 빌드 시간 추가 단축.

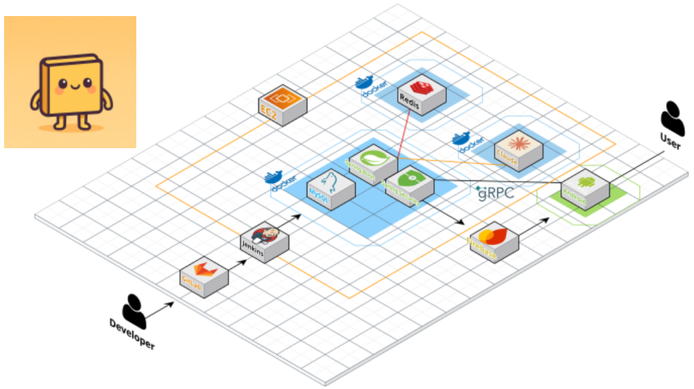
프로젝트 아카이브

수동 배포의 한계를 겪고, 실시간·지연·환경 불일치를 해결하는 과정을 통해 DevOps와 자동화의 가치를 체득했습니다.



NHCafe

초기에 수동 배포(EC2 접속→재빌드→재시작)로 장애/설정 드리프트 발생
STT→GPT→TTS 대화 흐름은 OK, 배포/환경 표준화가 과제 → 자동화 필요 명확



BookgleBookgle

GRPC 양방향 스트리밍으로 저지연 동기화/충돌완화
로컬=서버 DOCKER COMPOSE로 환경 일치·기동시간 단축

RouteMap / Broaf

지도·리스트 상태 동기화, 초기 로딩 지연 개선(지연 로딩·캐시)
KAKAO MAP API 검색/마커/경로선, UI XML + RECYCLERVIEW 표준화

THANK YOU

Song Jin-Woo

끝까지 검토해 주셔서 감사합니다.

M. 010-6483-5892 | E. rkddkwl059@naver.com | W. https://github.com/openSongce