PORTFOLIO

Server Programmer

Backend , Tools , Server

이준혁

문제를 흐름으로 보고,
병목은 계측하고 반복은 자동화합니다.

FastAPI & Spring Boot API, 오픈소스 CLI, 클라우드 운영 자동화를 통해 사용자, 개발자, 운영자의 흐름을 개선해왔습니다.

Email	bbbong9@gmail.com
Mobile	010-7339-8266
GitHub	github.com/bnbong
Blog	bnbong.github.io

RESUME

이준혁 / JunHyeok Lee

Server Programmer / Backend Engineer / Tools

EDUCATION

2016.03 - 2019.02 이매고등학교 졸업

2019.03 - 2026.08 한양대학교 ERICA 컴퓨터학부(졸업 예정)

WORK EXPERIENCE

2024.07 - 2024.08 카카오엔터프라이즈 IaaS 기술기획 인턴

2023.09 - 2024.06 K-Buddy 창업팀 Backend & DevOps

2021.04 - 2023.01 공군 정보체계관리 / 병장 만기전역

2020.09 - 2021.04 GiftMusic 창업팀 Backend Engineer

CERTIFICATE / AWARDS

2024.11 창업우수상

2023.11 창업우수상

2023.08 제4회 예술데이터가 바꾸는 세상 아이디어톤 우수상

2023.06 정보처리산업기사 취득

2020.12 정보처리기능사 취득

ETC

2020 – 2025 통기타동아리 JOY 통기타 강사

2023 SW 학회 JARAM 학회원

2023 통기타동아리 JOY 회장

2020 통기타동아리 JOY 부회장

2019 알고리즘 연구 학회 0&1 학회원

SKILLS

01

Language

서버, 시스템, 도구 구현 언어

- Python
- Java
- C
- JavaScript

02

Server

API, 비동기, 실시간 통신

- FastAPI
- Spring Boot
- WebSocket
- PyTorch

03

Data Store

관계형 DB, 캐시, NoSQL

- PostgreSQL
- Redis
- MongoDB
- MySQL

04

Infra and Deployment

컨테이너, CI/CD, 클라우드 운영

- Docker
- Kubernetes
- Actions
- Jenkins

05

Public Cloud

인스턴스 배치, 네트워크, 인프라 구성, DevOps 확장

- AWS
- OCI
- Azure
- GCP

Learning TypeScript, Go, Terraform, Helm

#JavaScript #FastAPI #PyTorch #PostgreSQL #MongoDB #Redis

PROJECT 01

Wegis

멀티모달 AI 기반 피싱 사이트 탐지 서비스

URL과 웹 사이트 HTML body를 함께 분석하는 **멀티모달 AI** 모델이 브라우저 확장 프로그램으로 현재 페이지의 모든 URL의 피싱 사이트 여부를 **실시간으로 판별하는 피싱 탐지 서비스**를 구현했습니다.
초기에는 직접 구축한 Kubernetes 인프라로 운영하다가 단일 컨테이너 구조로 재설계한 뒤 **계측 기반 성능 최적화**를 통해 응답 지연까지 개선했습니다.

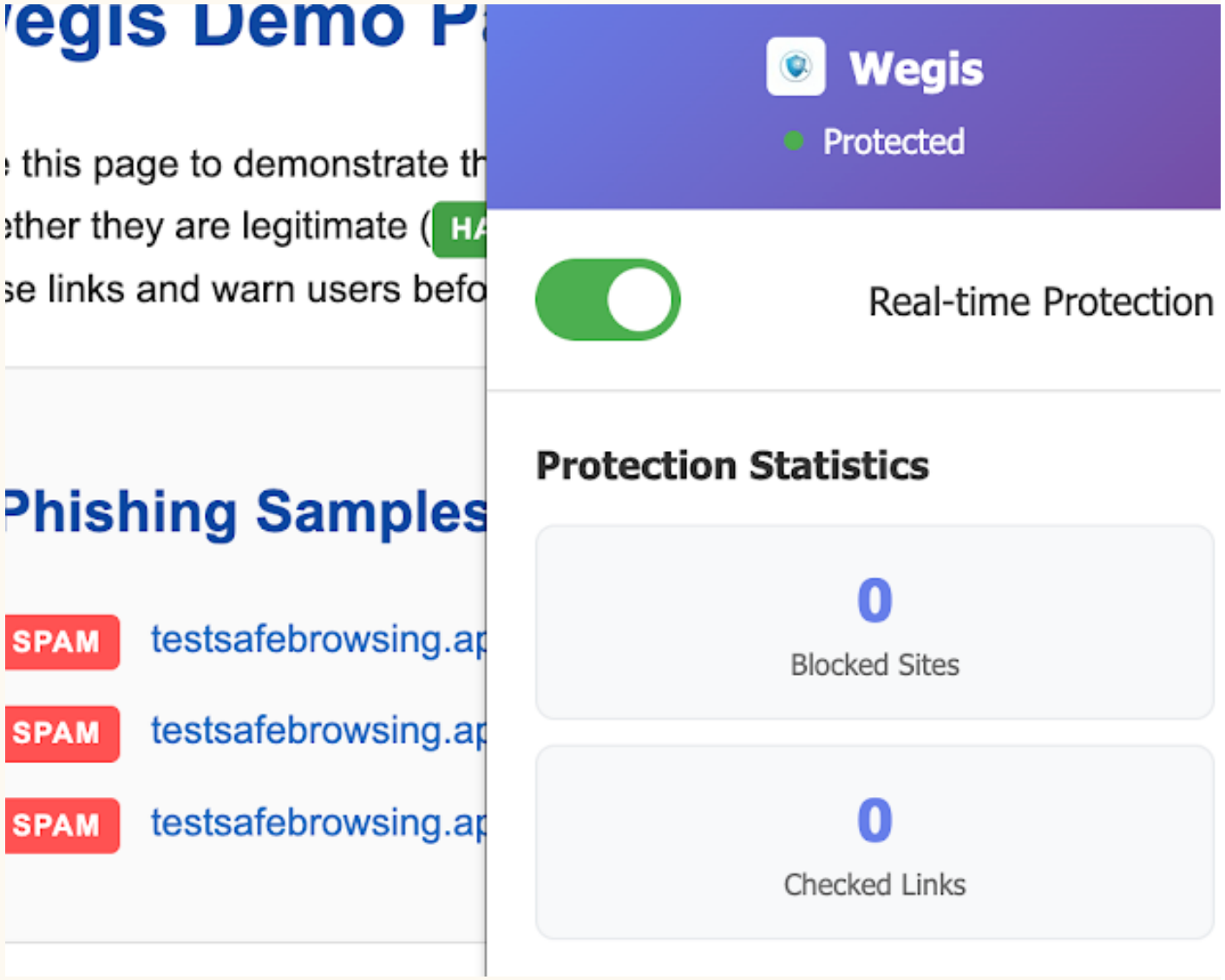
진행 기간| 2025.09 – 2025.10

팀 규모| 개인 프로젝트

주요 역할| AI 모델 학습 참여 / 서버 추론 파이프라인 구현 / 인프라 구축 및 재설계 / 브라우저 확장 프로그램 개발 / 성능 최적화

프로젝트 링크 (클라이언트)| https://github.com/bnbong/Wegis

프로젝트 링크 (서버)| https://github.com/bnbong/Wegis_server



PROJECT 01 Overview

Wegis 프로젝트 개요

프로젝트 배경 및 개요

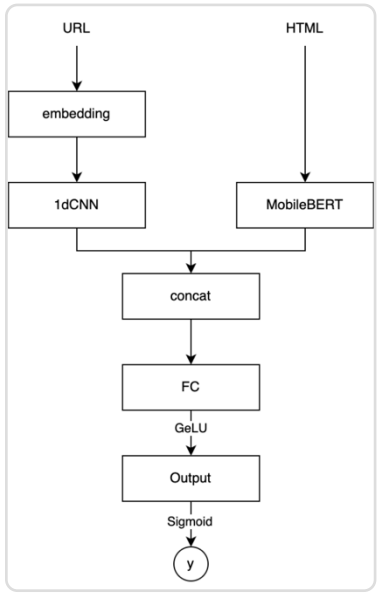
- URL과 HTML body content를 함께 분석하는 **멀티모달 AI 피싱 탐지 서비스**입니다.
- 졸업 프로젝트의 QR/URL 피싱 탐지 연구를 **브라우저 보호 도구로 확장**했습니다.
- 사용자가 링크와 QR URL을 **실시간으로 확인**할 수 있는 제품형 구조가 필요했습니다.

주요 역할

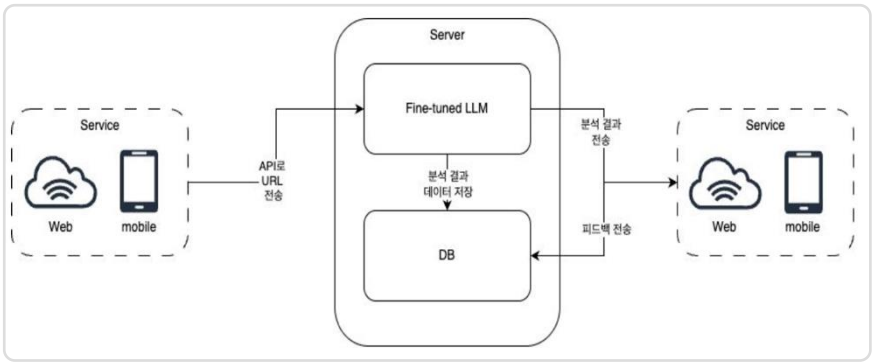
- CNN + MobileBERT 멀티모달 **모델 실험**과 FastAPI **추론 서버 구현**에 참여했습니다.
- Selenium 기반 **HTML 수집**, **전처리**와 단건/배치/피드백 **API를 구현**했습니다.
- Redis 캐시, 저장소 구성, 브라우저 확장 연동, 배포 구조 재설계를 담당했습니다.

결과 및 성과

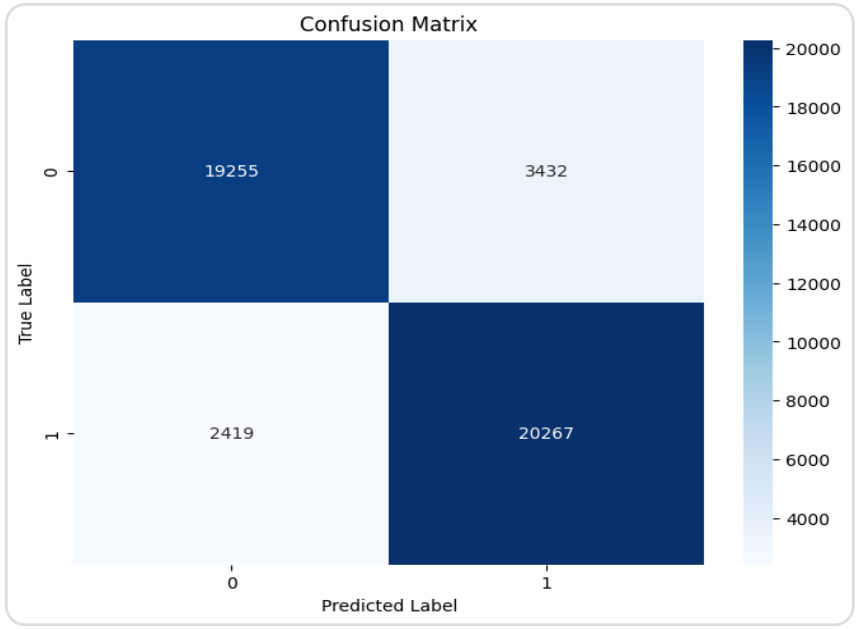
- **F1 0.8739, Accuracy 0.8710** 수준의 모델 성능을 확인했습니다.
- 단일 컨테이너 구조로 재설계해 운영비를 **월 26,000원에서 0원**으로 절감했습니다.
- avg latency 1,092.9ms → **715.3ms**, p95 1,297.1ms → **731.3ms**로 개선했습니다.



▲ 멀티모달 모델 구조



▲ 피싱사이트 판별 전체 흐름



▲ Confusion Matrix

PROJECT 01 Cost & Latency Optimization

복잡한 인프라를 걷어내고, 병목을 계측해 응답을 줄였습니다

문제

Kubernetes 기반 졸업 프로젝트를 개인 서비스로 옮기자 운영 비용과 관리 복잡도가 커졌고, **HTML fetch**와 **추론 파이프라인 중복**으로 응답이 느려졌습니다.

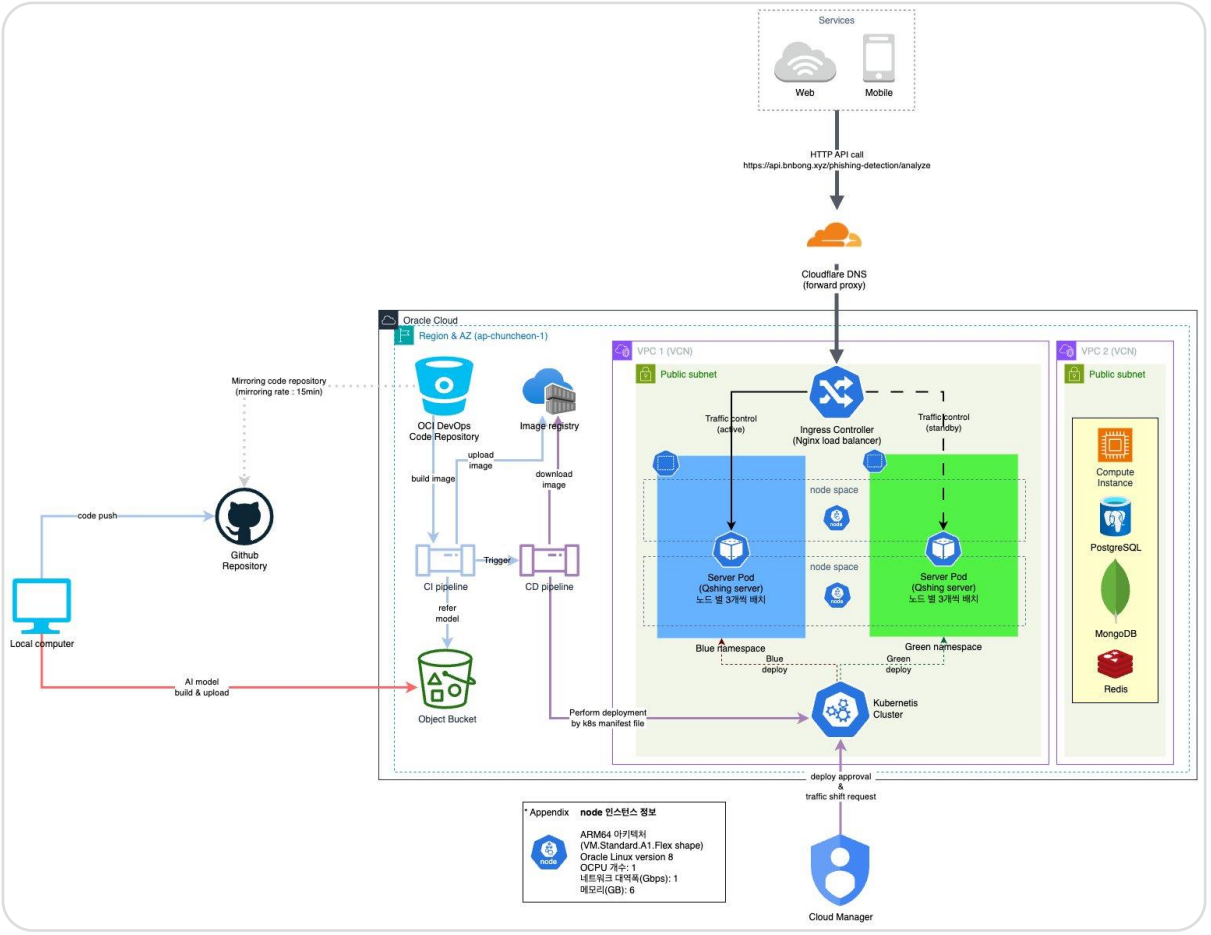
해결

단일 VM, **컨테이너** 구조로 재설계하고 HTML fetch 재사용, tokenizer cache, in-flight dedup, semaphore 기반 **병렬 처리 제어**를 적용했습니다.

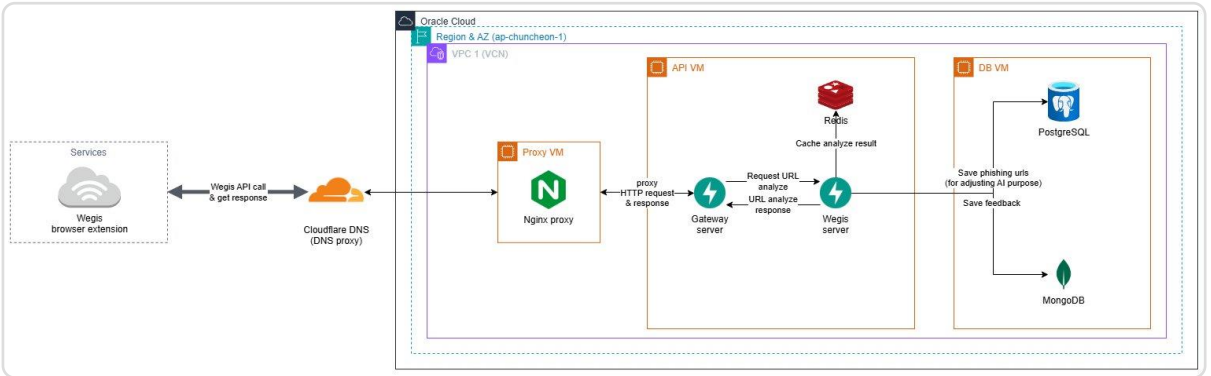
결과

월 운영비를 26,000원에서 **0원**으로 줄였고, 평균 latency 1,092.9ms → **715.3ms**, p95 1,297.1ms → **731.3ms**로 개선했습니다.

0원	34.6%	43.6%
월 운영비	평균 latency 개선율	p95 latency 개선율



▲ 이관 전 - Kubernetes 기반 인프라 (blue-green 무중단 배포)



▲ 이관 후 - 단일 컨테이너 구조

#Spring Boot #FastAPI #Jenkins #PostgreSQL #Redis #Docker

PROJECT 02

K-Buddy

외국인 생활 장벽을 낮추는 커뮤니티 서비스의 서버 & 인프라 구축

국내 거주 외국인이 장소, 이벤트, 커뮤니티 정보를 더 쉽게 접할 수 있도록 **Spring Boot** 기반 메인 서버와 **FastAPI** Mock API, **OCI** 기반 인프라, **CDN & 로깅 & 배포 체계**를 함께 설계한 프로젝트입니다.

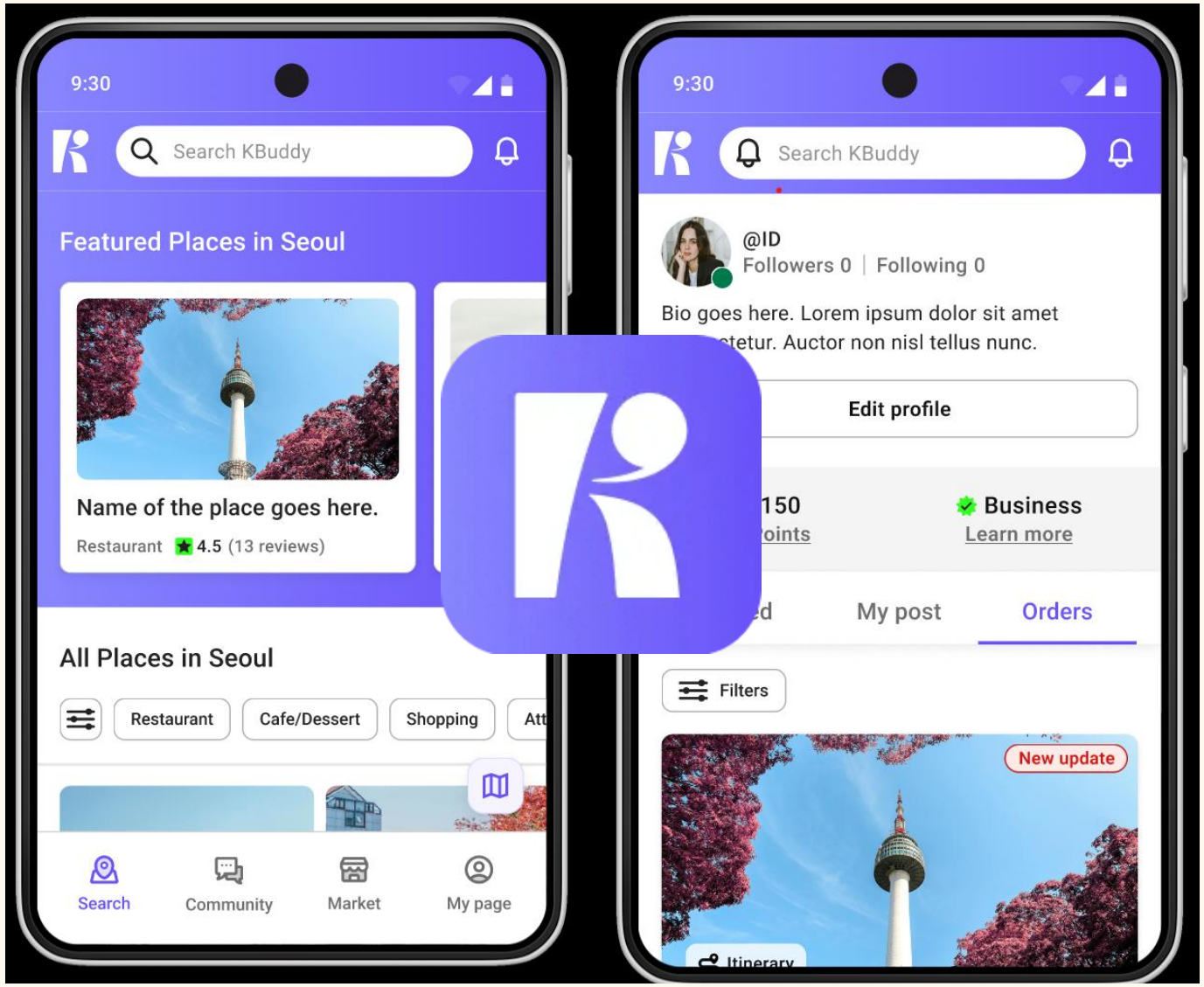
진행 기간| 2023.09 – 2024.06

팀 규모| 백엔드 2명, 프론트엔드 3명, 디자이너 1명

주요 역할| 메인 서버 개발 / Mock API 개발 / 클라우드 인프라 구축 / 위키 및 소통 구조 등 협업 기반 구축

프로젝트 링크 (메인)| https://github.com/bnbong/KBuddy-Server

프로젝트 링크 (Mock API)| https://github.com/bnbong/KBudddy-MockAPI



PROJECT 02 Overview

K-Buddy 프로젝트 개요

프로젝트 배경 및 개요

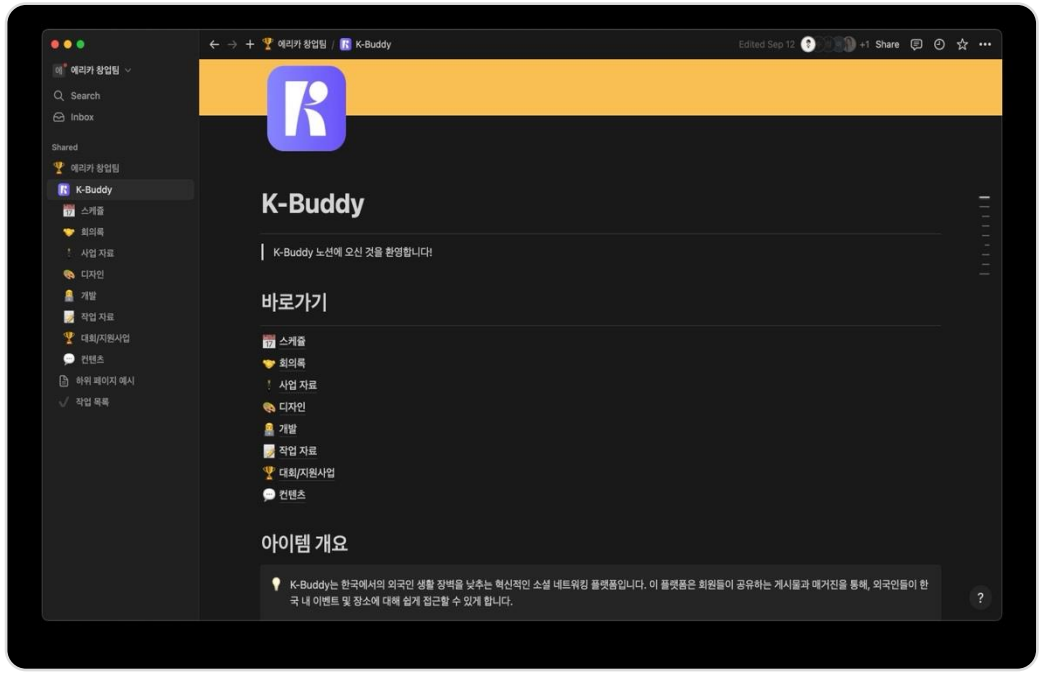
- **국내 거주 외국인**에게 장소, 이벤트, 커뮤니티 정보를 제공하는 **생활 정보 플랫폼**입니다.
- **모바일 앱** 개발과 **서버** 개발이 동시에 진행되어 API 계약, 테스트 환경이 먼저 필요했습니다.
- 소규모 팀이 운영하는 서비스라 **복잡도는 낮추고 배포, 장애 확인 비용을 줄이는 것**이 목표였습니다.

주요 역할

- Spring Boot + JPA 기반 **메인 API 서버를 설계**하고 주요 **도메인 API**를 구현했습니다.
- 프론트엔드 테스트용 FastAPI Mock API를 별도로 구축해 **병렬 개발을 지원**했습니다.
- OCI, Docker, Jenkins, GitHub Actions, Nginx, EFK, Uptime Kuma 기반 **운영 환경을 구성**했습니다.

결과 및 성과

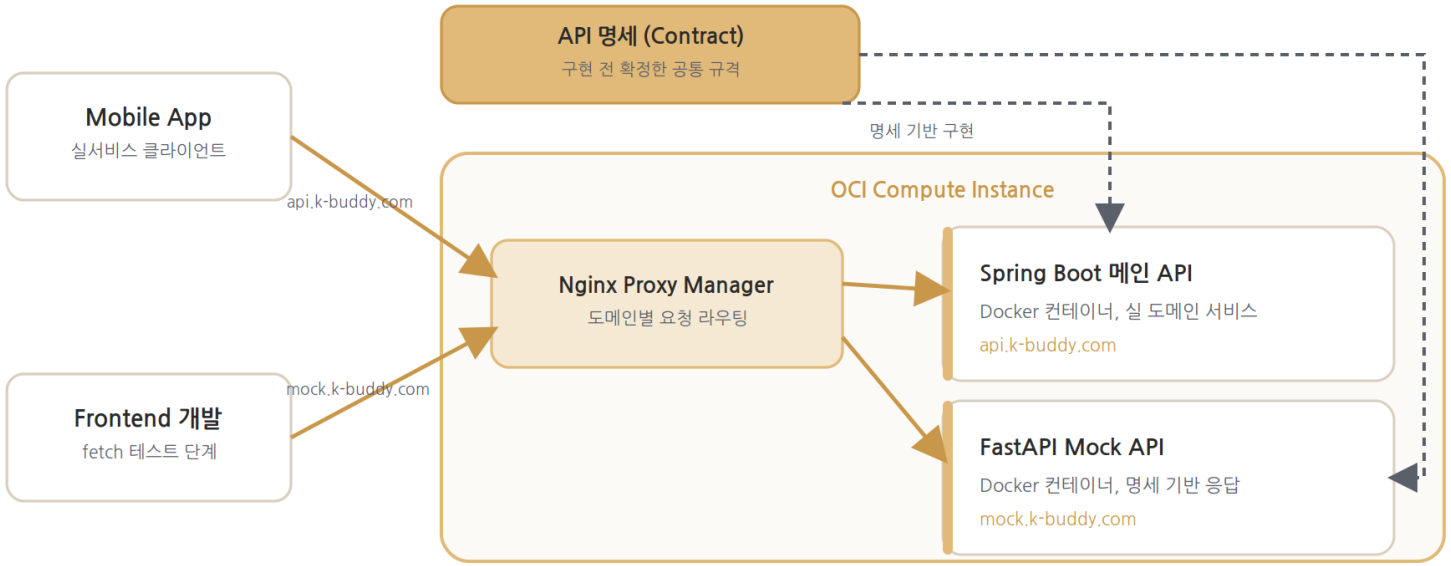
- 프론트엔드 연동 테스트와 서버 개발을 병렬화해 팀의 대기 시간을 줄였습니다.
- 배포, 로그, 헬스체크 확인 경로를 만들어 장애 대응과 운영 확인 비용을 낮췄습니다.
- Confluence/Notion/Slack을 구성하여 **API 문서와 의사결정 흐름을 공유**했습니다.



▲ 구성했던 K-Buddy 협업 Notion 워크스페이스

PROJECT 02 API Contract & Operations

팀의 대기 시간을 줄이기 위해 서버와 Mock API를 분리했습니다



▲ API 명세 기반 메인 API / Mock API 분리 구조

문제 클라이언트 개발은 실제 API 완성을 기다렸고, 팀이 배포, 로그, 헬스체크를 **수작업으로 반복**했습니다.

해결 확정한 **API 명세 기반**으로 Spring Boot 메인 서버와 FastAPI Mock API를 분리 배치하고 **도메인 라우팅**을 구성했습니다.

결과 프론트엔드 fetch 테스트와 메인 API 개발을 **병렬화**하고, 배포, 로그, 헬스체크 **진입 비용**을 함께 낮췄습니다.

#Python #FastAPI #Click #Rich #Mkdocs #GitHub Actions

PROJECT 04

FastAPI-fastkit

FastAPI 입문자를 위한 오픈소스 CLI & 문서 플랫폼

Python/FastAPI 입문자가 초기 서버 환경 구성과 프로젝트 구조 설계에서 겪는 진입 장벽을 줄이기 위해, CLI로 프로젝트를 생성하고 튜토리얼 문서로 학습까지 이어지는 개발자 친화형 오픈소스를 운영한 프로젝트입니다.

진행 기간| 2024.08 – 진행중

팀 규모| 개인 오픈소스 프로젝트

주요 역할| CLI 설계 / 템플릿 구현 / 문서화 / CI & 품질 자동화

프로젝트 링크| https://github.com/bnbong/FastAPI-fastkit

PROJECT 03 Overview

FastAPI-fastkit 프로젝트 개요

프로젝트 배경 및 개요

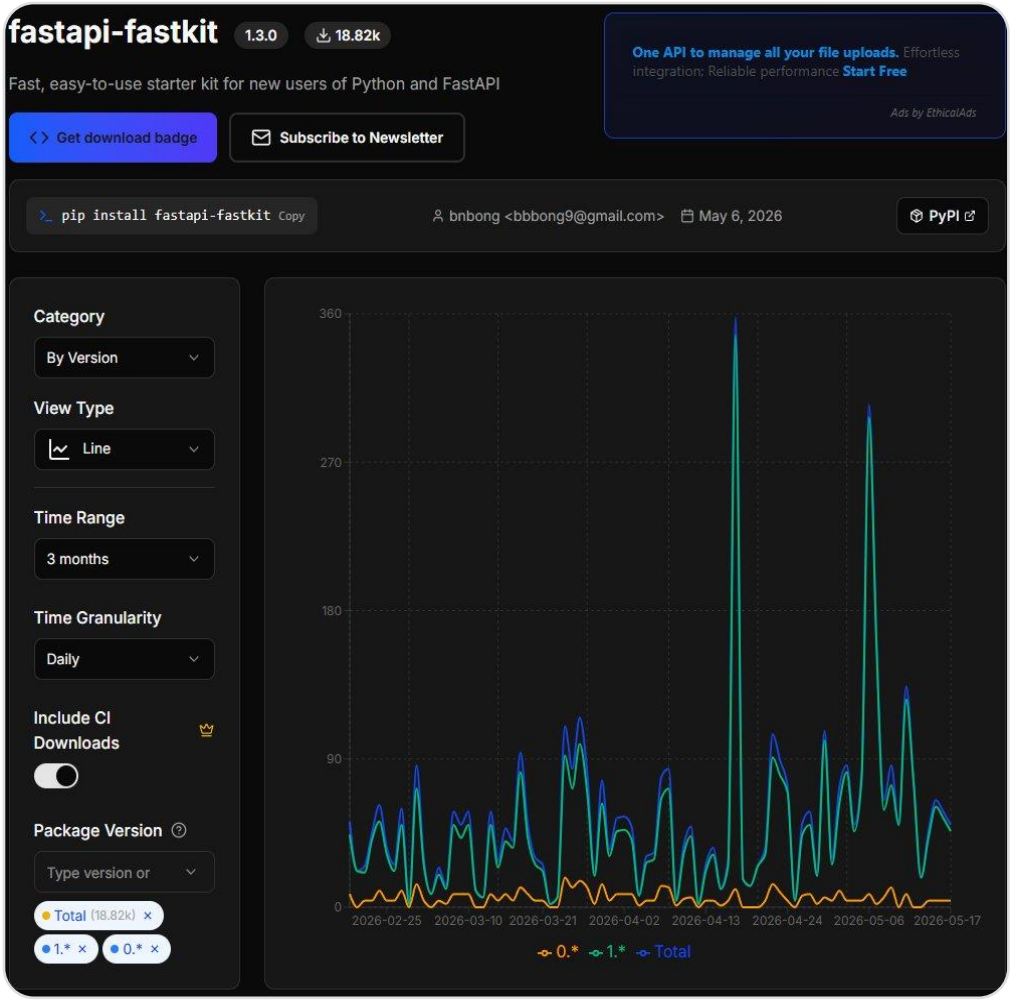
- **FastAPI 입문자**를 위한 오픈소스 CLI & 문서 플랫폼입니다.
- FastAPI는 자유도가 높아 입문자가 구조, DB, 인증, 테스트, Docker **설정을 반복** 선택해야 했습니다.
- 오픈소스는 기능 추가보다 시간이 지나도 **깨지지 않는 구조와 문서** 신뢰성이 중요했습니다.

주요 역할

- init, addroute, runserver, list-templates 등 **CLI command**와 **interactive builder**를 설계했습니다.
- minimal/standard/full 템플릿 및 유즈케이스 템플릿으로 **선택 가능한 scaffolding**을 구현했습니다.
- **GitHub Actions** 기반 matrix test, template QA, release/docs workflow를 구성했습니다.

결과 및 성과

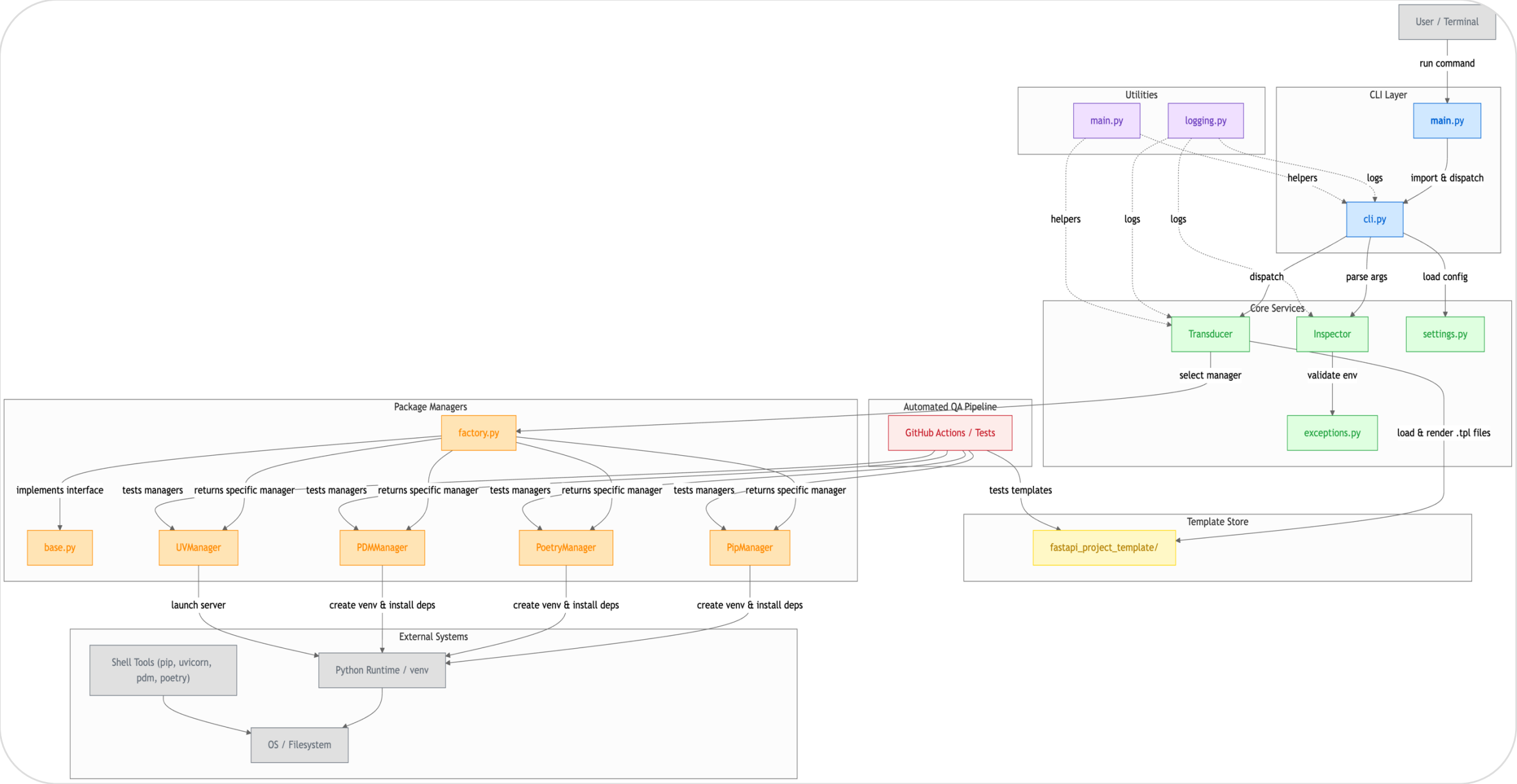
- **PyPI 18k+** 다운로드를 기록한 오픈소스 패키지로 운영 중입니다.
- 코드 변경, 템플릿 검증, 릴리스, 문서 배포를 관리하는 파이프라인을 만들었습니다.
- **7개 국어** 지원 User Guide, Tutorial, CLI Reference 문서로 **사용자 온보딩을 개선**했습니다.



▲ FastAPI-fastkit PyPI 다운로드 추이(26년 5월 기준)

18k+	9종	Python 3.9~3.14 대응
PyPI 다운로드	제공 템플릿 (유즈케이스 기반)	matrix test

PROJECT 03 Module Diagram



#C #Python #fio

PROJECT 04

FEMU

Hot / Cold Data Separation으로 SSD GC 효율을 개선한 시스템 프로젝트

FEMU SSD emulator의 FTL에 LPN 접근 빈도 기반 hot/cold detection 모듈을 추가하고, write path와 GC path를 분리해 **WAF를 줄이고 IOPS 안정성을 개선**한 시스템 성능 최적화 프로젝트입니다.

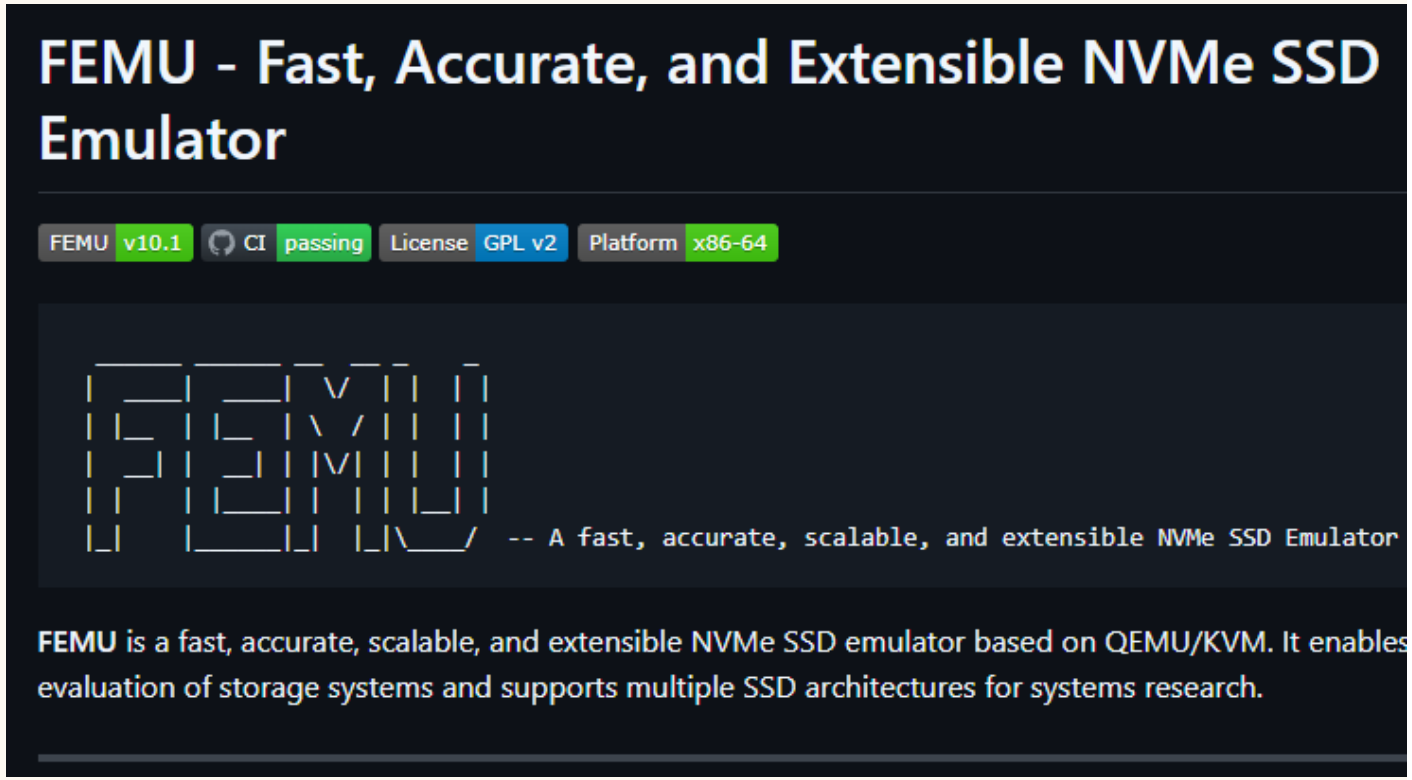
04

진행 기간| 2024.10 – 2024.12

팀 규모| 개인 프로젝트 / 시스템프로그래밍 수업 과제

주요 역할| C 기반 FTL 수정 / benchmark 계측 / python graph analysis

프로젝트 링크| https://github.com/bnbong/FEMU



PROJECT 04 Overview

FEMU 프로젝트 개요

프로젝트 배경 및 개요

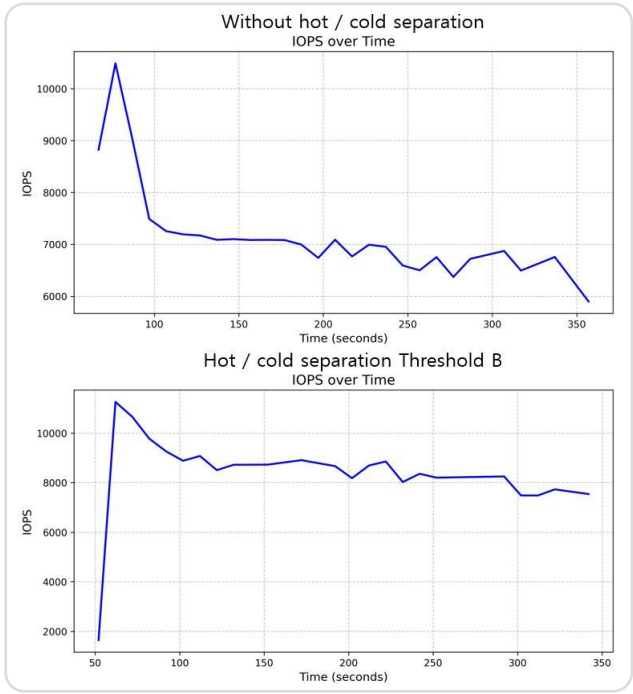
- QEMU 기반 SSD 에뮬레이터 FEMU의 FTL을 수정해 **GC 효율을 개선**한 시스템 프로젝트입니다.
- 기존 FEMU에는 workload별 write hotness를 line에 반영하는 로직이 부족했습니다.
- 기존 FTL 코어 로직을 **최대한 재사용**하면서 **hot/cold branching**을 추가하는 것이 목표였습니다.

주요 역할

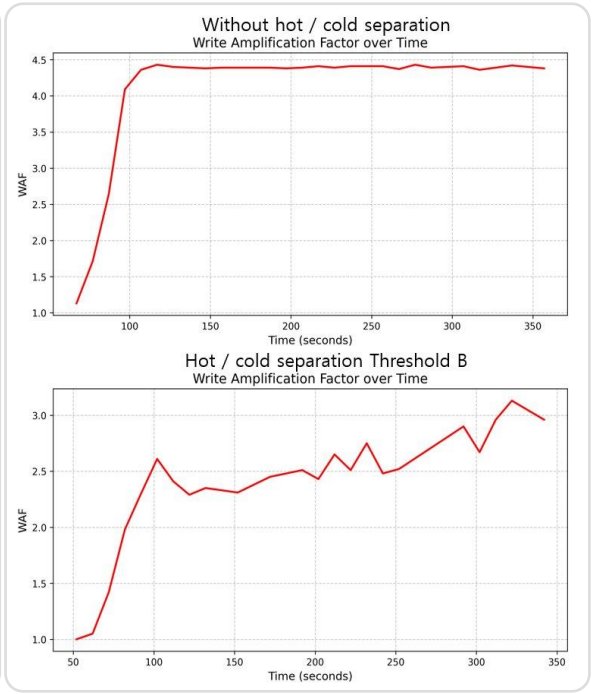
- lpn_access_cnt, write_count, gc_write_count 기반 **통계 수집 로직**을 구현했습니다.
- line_mgmt에 **hot/cold line 구조**를 추가하고 **hot_wp/cold_wp 포인터**를 설계했습니다.
- ssd_write, gc_write_page 등을 수정하고 **fio benchmark**로 효과를 검증했습니다.

결과 및 성과

- Zipfian 1.2 workload 기준 WAF를 4.2~4.3 → **2.7~2.9** 수준으로 낮췄습니다.
- IOPS는 약 7,000~7,500 → **8,500~9,000** 수준으로 개선되었습니다.
- workload 특성에 따라 threshold와 정책 효과가 달라진다는 점을 확인했습니다.



▲ IOPS 개선



▲ WAF 개선

PROJECT 04 Hot / Cold Separation

성능 개선은 구조 변경보다 계측에서 시작했습니다

문제

단순히 hot/cold 구조를 추가하는 것만으로는 개선을 설명할 수 없었습니다.
write skew가 있는 workload에서
WAF와 IOPS가 어떻게 달라지는지 직접 비교해야 했습니다.

해결

host write와 GC write 양쪽에 **hot/cold 분기**를 적용하고,
fio workload별 access count, WAF, IOPS를 수집했습니다.
Python 그래프로 threshold별 차이를 시각화했습니다.

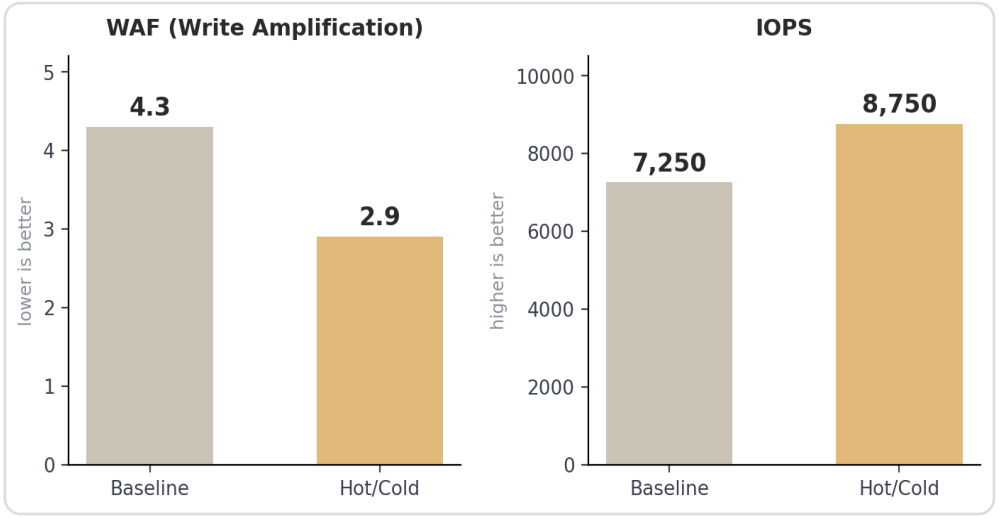
결과

검증한 workload 3종(Normal, pareto, Zipfian distribution) 중 write skew가 강한 **Zipfian** 분포에서 hot/cold separation이 특히 효과적이었습니다.
WAF 감소가 IOPS 안정성으로 이어질 수 있음을 수치로 확인했습니다.

4.3 → 2.9
WAF 감소

7.5k → 9k
IOPS 개선

Zipfian 1.2
검증 workload



▲ Zipfian 1.2 workload 기준 WAF / IOPS 계측 결과

Hot / Cold Write Path

LPN 접근 빈도로 hot/cold line을 분리하고,
write path와 GC path 양쪽에 분기를 적용해 **같은 line에 유사 hotness 데이터가 모이도록** 유도, GC 비용과 WAF를 함께 낮췄습니다.

Thank you

제가 만드는 것은 기능이 아니라, 끊기지 않는 흐름입니다.

이준혁, Server Programmer / Backend Engineer / Tools

Email bbbong9@gmail.com

Mobile 010-7339-8266

GitHub github.com/bnbong

Blog bnbong.github.io