

Portfolio

AWS

MSA

K8s

김낙균 (KIM NakGyun)

2001.02.21



nacgyun001@email.com



github.com/nacgyun

Certifications & Education



Certifications



AWS Certified Solutions Architect – Associate (SAA)

취득 완료 (2026.05)



정보처리기사

취득 완료 (2025.09)



Education



한경대학교 소프트웨어&서비스컴퓨팅(컴퓨터공학과)

학사 과정 (2020.03 ~ 2026.02) (전공 졸업학점 3.65/4.5)

네트워크(4.5)

컴퓨터 구조(4.5)

정보보호(4.5)

모바일 앱 개발(4.5)



Technical Skills

Cloud & Infra

- AWS(EKS, ECS, Fargate, S3, RDS)
- Kubernetes, Docker, Terraform

Scripting & OS

- Python, Bash, Linux

Monitoring

- Prometheus, Grafana, Loki, Fluent Bit

CI/CD

- Github Actions

Language & Frameworks

- JAVA, JS, Spring Boot, React

Project 1 - UNBOX

AWS 3-Tier 아키텍처 고가용성 신발 거래 플랫폼 인프라 구축

프로젝트 기간

2025.12 ~ 2026.02 (팀 프로젝트, 5인)

담당 역할

인프라 설계 및 구현 담당

Contributions & Impact

장애 전파 차단 및 통신 보안 강화

- 테스트 결과 기반 Linkerd를 도입하여 서킷 브레이커 패턴으로 장애 전파 차단
- mTLS 암호화를 통해 내부 네트워크 보안 강화

인프라 마이그레이션을 통해 총 52% 비용 절감

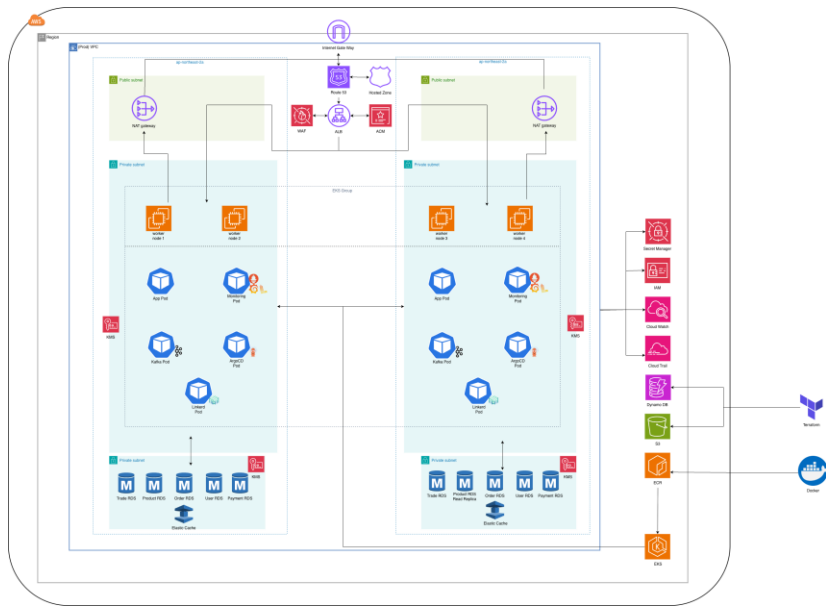
- Cost Explorer 분석을 통해 과도한 비용 지출 탐지 및 인프라 마이그레이션
- 오버 엔지니어링으로 판단되는 MSK 제거
- ECS => EKS 전환으로 운영 비용 최적화

FLG 스택을 통해 모니터링 환경 구축

- FLG + Prometheus를 통해 로그 및 메트릭 모니터링 환경 구축
- Tempo를 도입하여 Log Trace

프로젝트 문서화로 커뮤니케이션 비용 절감

- Notion을 통해 설계 사항 및 의견사항 정의서 작성



<https://github.com/team4-coding-gamja/UNBOX-INFRA>

Project 1 - UNBOX

주요 트러블슈팅

⚠ Case 1: 인프라 유지비용 과다 발생

! 문제 발생

실제 인프라 사용량이 130\$로 과다 발생

🔍 원인 분석

CostExplorer를 통해 비용 분석

- MSK에서 과도한 비용 발견
- ECS, CloudWatch에서 낭비되는 리소스 발견

🔧 해결 방안

오버 엔지니어링으로 판단되는 MSK를 제거
EKS를 통해 낭비되는 리소스를 줄이고, 직접 kafka와 모니터링 환경 운영

✓ 결과 및 개선

인프라 마이그레이션을 통해 예상 비용 약 52% 절감
MSK와 CloudWatch 제거를 통한 AWS 의존성 감소 및 이식성 향상

⚠ Case 2: IaC - Controller 간 리소스 관리 충돌

! 문제 발생

앱은 정상이지만 서비스 전체 503 에러 발생

🔍 원인 분석

AWS 콘솔을 통해 ALB 상태 확인

Aws elbv2 describe-target-health 명령어 확인 -> 등록된 Pod 0개
Terraform과 LBC의 타겟 그룹 불일치 확인

🔧 해결 방안

TargetGroupBinding 수정으로 임시 조치
IaC의 ALB 모듈 삭제 및 리팩토링

✓ 결과 및 개선

라우팅 불일치 문제 해결
해결 과정 문서화를 통한 대책 마련

Project 2 - LanderAI

AWS 서버리스 기반 랜딩 페이지 제작 플랫폼

프로젝트 기간

2026.06 ~ 2026.09 (개인 프로젝트)

담당 역할

개발, 인프라 설계 및 구현

Contributions & Impact

서버리스 구조를 통한 가용성 확보 및 비용 효율화

- Step Functions 기반으로 생성 Workflow를 분리해 확장성과 장애 복구성 향상
- 요청 기반 실행 구조를 통해 유휴 인프라 비용 최소화

RAG 기반 사용자 선호 반영 구조 설계

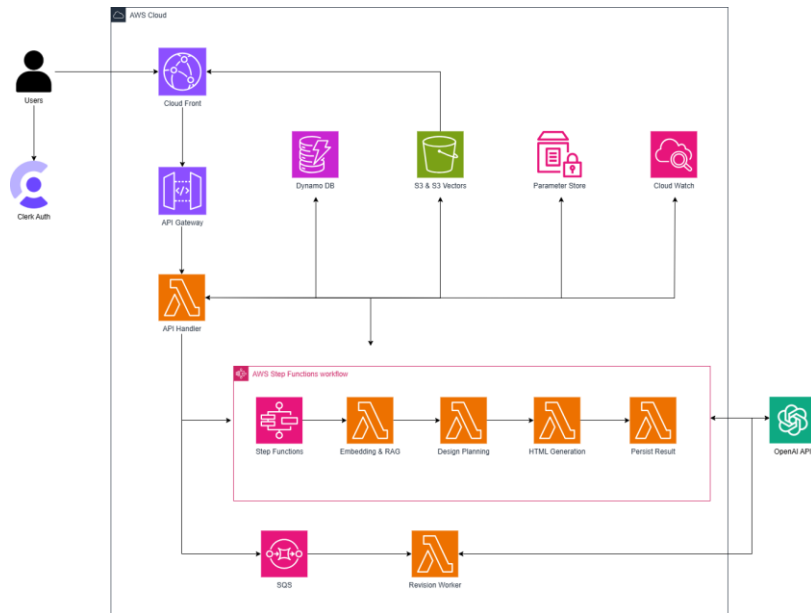
- 사용자가 선택한 결과를 저장해 후속 생성에 활용
- 임베딩을 통해 유사 요청에 대해 RAG 데이터 사용

DynamoDB 조회 구조 개선

- Scan 방식의 확장성 및 성능 문제를 확인하고 GSI 기반 방식으로 개선
- 데이터 10,000개 기준 약 23.6배 속도 개선

아키텍처 비교 실험을 통한 기술 도입 검증

- 기존 단일 Lambda 사용과 Step functions 구조의 처리 시간 및 요청당 비용 비교
- 성능과 비용 데이터를 기반으로 서버리스 Workflow 도입



<https://github.com/nacgyun/LANDERAI-BE>



<https://landerai.un-box.click>

Project 2 - LanderAI

주요 트러블슈팅

✕ Case 1: DynamoDB 조회 성능 저하

! 문제 발생

로컬 환경에서 기능 테스트 중 특정 조회 기능에서 응답 지연 발생

🔍 원인 분석

해당 기능의 로직 확인

- Scan + FilterExpression 방식 사용
- 데이터가 증가할수록 불필요한 Read와 조회 시간이 함께 증가하는 구조

🔧 해결 방안

기존 방식과 GSI 방식을 테스트를 기반으로 도입

user_id를 GSI로 구성하여 자주 쓰이는 특정 기능에 대해 조회 효율 향상

✓ 결과 및 개선

로컬 환경 10,000건 데이터 기준

평균 조회시간 1,138.65ms > 48.28ms, 약 23.6배 단축

소비 읽기 용량 2,231 RRU > 1 RRU, 약 99.96% 감소

✕ Case 2: Clerk Production 로그인 장애

! 문제 발생

Clerk를 사용하는 Prod 환경에서 로그인이 활성화 되지 않는 장애 발생

🔍 원인 분석

콘솔의 오류 중 Clerk로 보내는 요청이 실패하는 것을 확인

로그인을 위한 요청을 받는 Clerk의 도메인이 달라 보안상의 이유로 cross-site cookie가 적용되는 것을 확인

🔧 해결 방안

Route 53에 CNAME을 통해 clerk로 요청을 보내도록 변경

해당 수정 부분 Terraform에 적용

✓ 결과 및 개선

Production 환경 로그인 장애 해결

CNAME 설정을 Terraform을 통해 관리해 인증 도메인 구성 재현성 확보

Project 3 – K8s 기반 모델 배포 자동화

Kubernetes 기반 모델 배포 및 운영 자동화

프로젝트 기간

2025.07 ~ 2025.10 (팀 프로젝트, 3인)

담당 역할

인프라 설계 및 구현 담당

Contributions & Impact

Kubernetes 기반 모델 배포 자동화 환경 구축

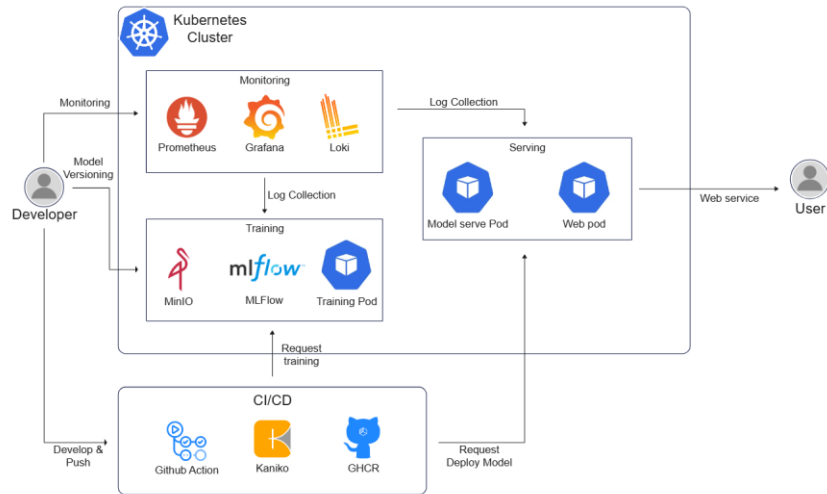
- Kubernetes 기반 학습 및 모델 서빙 환경 구축
- MLflow와 MinIO를 활용한 모델 및 아티팩트 관리
- 모델 생명주기를 관리할 수 있는 기반 마련

CI/CD 기반 모델 배포 자동화

- GitHub Actions, Kaniko, GHCR를 활용한 자동 배포 파이프라인 구축
- 코드 변경 시 Kubernetes Pod 자동 배포
- 반복적인 수동 배포 과정을 자동화하여 운영 효율성 향상 (15분 → 6분)

모니터링 환경 구축

- Prometheus, Grafana, Loki 기반 메트릭 및 로그 수집
- Kubernetes 클러스터와 서비스 상태 실시간 모니터링
- 장애 원인 분석 및 운영 가시성 확보



<https://github.com/nacgyun/mlopsgit>

Project 3 – K8s 기반 모델 배포 자동화

주요 트러블슈팅

Case 1: Calico BGP 피어링 장애

문제 발생

K8s 클러스터 내 calico-node Pod가 0/1 running
Master – node1 간 pod 네트워크 통신 불안정

원인 분석

Kubectl describe 명령어로 로그 확인
– Readiness probe failed: BIRD is not ready 확인
BGP 피어링 문제 가능성을 의심해 Master – node1 TCP 179 연결 테스트
– iptables 정책 확인
– Node1에서 179포트가 허용되지 않았음을 확인

해결 방안

호스트 iptables 수정 및 netfilter-persistent를 통해 영구화

결과 및 개선

Calico BGP 세션 정상 수립
노드 재부팅 후에도 정책 유지

Case 2: Grafana UI 외부 접근 장애

문제 발생

외부에서 Grafana UI 접근 시 timed out에러 발생

원인 분석

Grafana가 할당된 노드의 iptable 확인
– INPUT은 허용되었으나 Forward 차단상태
iptables 수정 후에도 오류 발생으로 NetworkPolicy 확인
– Monitoring 네임스페이스의 Networkpolicy가 Grafana 인바운드 차단

해결 방안

iptables는 netfilter-persistent를 통해 영구 적용
임시 허용 네트워크 정책 적용

결과 및 개선

운영 환경 최소 허용 정책 적용
불필요한 외부 접근 차단 유지

검토해주셔서 감사합니다.

인프라에 대한 깊은 열정과 경험을 바탕으로,
인프라를 더욱 안정적이고 효율적으로 만드는 데 기여하겠습니다.