



# 김용현

## BRIEF INTRODUCTION

DevOps Engineer의 로드맵을 걸어가는 김용현입니다.

현재 레브잇에서 데브옵스 업무를 맡고 있습니다.

## CONTACT ME AT

010-2763-9988

cdfgogo0615@naver.com

<https://github.com/Hulkong>

## CORE COMPETENCIES

- 플랫폼 개발
- 쿠버네티스 구성 및 운영
- AWS 기반 클라우드 플랫폼 관리 및 운영
- 네트워크 구축 (AWS VPC, DX, OpenVPN, IPsec VPN)
- CI&CD 구성 및 운영
- 관측시스템 구성 및 운영

## CAREER

### (주)레브잇 (2023.11 ~ 현재)

Engineering / DevOps Engineer

### (주)야놀자 (2022.06 ~ 2023.10)

Cloud Platform / Platform Engineer

### (주)카사코리아 (2021.09 ~ 2022.06)

DevOps Cell / DevOps Engineer

### (주)오픈메이트 (2016.11 ~ 2021.08)

운영사업팀 / 책임 연구원 / developer

## SKILLS SUMMARY

AWS Service

CI & CD

Kubernetes

Observability

Linux

Security

Network

Java & Kotlin

## EDUCATIONAL HISTORY

### 성균관대학교 대학원[Seoul]

정보보호학 석사 졸업 (학점: 4.17 / 4.5) | 2016.03 ~ 2018.08

### 한국산업기술대학교[경기]

컴퓨터공학과 학사 졸업 (학점: 3.97 / 4.5) | 2014.03 ~ 2016.02

# (주)레브잇 (2023.11 ~ 현재)

## 가용성 확보

진행기간: 약 2주 소요

프로젝트 인원 수: 총 1명(+α)

본인 포지션: 데브옵스

본인이 사용한 스킬: Kubernetes, Elasticache, Redis, Graviton, Auto Scaling, 트러블슈팅, 시스템 모니터링

### Impact

2025년 1분기 동안 서비스 가용성 확보와 관련된 핵심 작업들을 진행하여, Elasticache 최적화, 쿠버네티스 ReadinessProbe 설정 개선, 트래픽 급증에 대응한 전략 등을 통해 서비스의 가용성과 성능을 크게 향상시켰습니다. 이로 인해 서비스 안정성 확보와 트래픽 급증 대응 능력이 향상되었으며, 향후 발생 가능한 장애를 예방할 수 있는 예방 조치를 마련하는 데 기여했습니다.

### Action

- 크론잡 관련 지표 알람 설정: 임계치 초과 시 슬랙으로 알람을 송신하여 빠른 대응 시스템을 마련하고 문제를 즉시 인지할 수 있도록 했습니다.
- 카펜터(오토 스케일러) 인스턴스 스펙 제한: "r6g", "r6gd", "r7g", "c6g", "c6gd", "c7g", "m6g", "m6gd", "m7g" 스펙으로 제한하여 비용 절감과 성능 향상을 동시에 달성했습니다.
- Kubernetes ReadinessProbe 실패 해결: 502 Bad Gateway 문제를 해결하기 위해 startupProbe와 initialDelaySeconds 설정 방법을 적용하여 안정적인 서비스 운영을 지원했습니다.
- ELB 에러 트러블슈팅: 502, 503, 504 에러에 대한 원인 분석 및 해결책을 제시하여 트래픽 급증 시 안정성을 확보하고 대응 방안을 마련했습니다.
- 신규 추천 서비스 불안정 문제 해결: Redis와 추천 서비스 간 트래픽 급증 문제를 해결하기 위해 Redis 샤드 수 증축과 추천 서비스 worker 수 증대 등의 조치를 취해 서비스 안정성을 확보했습니다.
- 결제 서비스 가용성 확보: 결제 서비스의 Active-Active 환경 전환과 헬스체크 모니터링 강화로 결제 관련 장애 예방 및 서비스 가용성을 향상시켰습니다.

### Outcome

- Elasticache 최적화와 Graviton 적용을 통해 리소스 최적화와 비용 절감을 달성.
- 트래픽 급증 대응 전략을 통해 서비스의 가용성을 안정적으로 확보.
- 서비스 안정성 향상 및 고객 경험 개선을 위한 리소스 최적화 작업을 성공적으로 수행.
- 미래 장애 예방을 위한 예방 조치들을 마련하고, 시스템의 안정적 운영을 위한 기반을 구축했습니다.

## 2025.03월 인프라 비용 절감 리딩

진행기간: 약 2주 소요

프로젝트 인원 수: 총 2명(+α)

본인 포지션: 데브옵스

본인이 사용한 스킬: Elasticache 최적화, Graviton, 리소스 최적화, 비용 분석

### Impact

2025년 3월에 2월 대비 \$39,122.38(약 39백만원, 8.38%)의 인프라 비용 절감을 달성했습니다. 이번 비용 절감은 Elasticache의 효율적인 스펙 다운과 백엔드 서비스의 Graviton 2, 3세대 적용을 통해 이루어진 세밀한 리소스 관리의 결과로, 리소스 활용의 효율성을 극대화하고, 비용 절감과 함께 지속 가능한 관리 방안을 마련하는 데 중요한 역할을 했습니다.

### Action

- Elasticache의 스펙 다운 및 스케일 스케줄링을 통해 리소스를 효율적으로 사용하고 비용 절감을 달성했습니다.
- 올웨이즈 백엔드 서비스의 Graviton 2, 3세대 범위 제한을 통해 서버 효율성을 높이고, 비용을 절감했습니다.
- 미사용 서비스 제거 및 불필요한 리소스를 정리하여 추가적인 비용 절감을 실현했습니다.
- 카테고리 별 비용 분석 환경을 구축하여, 향후 비용 절감 가능한 영역을 명확히 파악하고 관리할 수 있도록 했습니다.

### Outcome

- 3월에 2월 대비 \$39,122 (약 39백만원, 8.38%)의 비용 절감.
- 리소스 관리 및 최적화를 통해 효율적인 자원 활용과 지속 가능한 비용 절감을 달성.
- 정확한 데이터 기반 분석을 통해 미래 비용 절감 가능 영역을 파악하고 관리.
- 목표를 계속 높여가며 향후 지속 가능한 비용 절감 전략을 채택하여 회사의 재정적 효율성을 강화.

## 2025.02월 인프라 비용 절감 리딩

진행기간: 약 2주 소요  
프로젝트 인원 수: 총 5명  
본인 포지션: 데브옵스  
본인이 사용한 스킬: 비용 최적화 전략, EC2, Graviton, 리소스 최적화, 데이터 분석

### Impact

2025년 2월에 1월 대비 \$171,294.41(약 171백만원, 26.83%)의 인프라 비용 절감을 달성했습니다. 이번 비용 절감으로 리소스 최적화 및 비용 관리 효율성이 크게 향상되었으며, 조직의 재정 건전성에 긍정적인 영향을 미쳤습니다. 절감을 통해 향후 투자와 성장에 대한 재정적 기반을 마련하는 데 기여했습니다.

### Action

- 비용 절감 액션 리스트 도출 및 방향성 제시를 통해 지속 가능한 비용 절감 전략을 수립.
- EC2 인스턴스에 Graviton을 적용하여 비용을 절감.
- 사용하지 않는 리소스 삭제 및 모니터링을 통해 유휴 자원 제거.
- 각 서비스와 리소스의 효율적인 사용을 모니터링하고, 불필요한 자원 제거를 통해 고정비 절감을 목표로 했습니다.

### Outcome

- 2월에 1월 대비 \$171,294 (약 171백만원, 26.83%)의 비용 절감.
- 비용 절감을 통해 리소스 관리와 ROI 증대, 전체 비용 구조 최적화.
- 목표 달성을 위한 효율적 리소스 관리와 협력적 노력을 통해 조직 차원에서의 성장과 발전.

## 2025.01월 인프라 비용 절감 리딩

진행기간: 약 2주 소요  
프로젝트 인원 수: 총 5명(+a)  
본인 포지션: 데브옵스  
본인이 사용한 스킬: 비용 최적화 전략, EC2, CloudFront, Kubernetes, VPC, Savings Plan

### Impact

1월에 2024년 11월과 비교하여 인프라 비용을 \$94,623 (약 95백만원, 12.91%) 절감한 성과를 이뤄냈습니다. 이로 인해 비용 절감을 통한 효율적인 리소스 관리가 이루어졌으며, 비용 절감을 통해 더 큰 ROI를 창출하고, 전체적인 비용 구조의 최적화가 이루어졌습니다.

### Action

- 비용 절감 액션 리스트 도출 및 방향성 제시를 통해 지속 가능한 비용 절감 전략을 수립.
- Atlas Org 삭제 및 최적화
- always-assets 비용 절감
- data-prod-vpc 내 VPC Endpoint 설치로 데이터드 관련 Data Transfer 비용 절감
- 정적 콘텐츠 CloudFront를 이용하여 비용 절감
- Kubernetes 클러스터 도메인 통신을 ALB 기반 도메인 통신으로 변경
- data-prod-vpc 내 VPC Endpoint 설치로 NatGW 비용 절감
- 기존 추천 & 검색 서비스 파드 개수 줄이기
- Graviton 서버로 이전하여 비용 절감
- 스케일링 스케줄링 변경, 온디맨드 → 스팟으로 변경
- 인스턴스 스펙 조정 및 Savings Plans 적용
- 파드 리소스 최적화
- 미사용 서비스 제거 & 과스펙 서비스 스펙 다운
- 보안 관련 서비스 및 리소스 제거 및 스펙 다운
- 태그 추가로 리소스 추적 및 최적화

### Outcome

- 1월에 2024년 11월과 비교하여 \$94,623 (약 95백만원, 12.91%)의 비용 절감을 달성.
- 비용 절감을 통해 리소스 관리와 ROI 증가, 비용 구조 최적화에 기여.
- 목표 달성을 위한 효율적 리소스 관리 전략이 효과적으로 실행됨.

## Cross존 간 통신 비용 절감

진행기간: 약 1주 소요  
프로젝트 인원 수: 총 1명  
본인 포지션: 데브옵스  
본인이 사용한 스킬: Private ALB 설정, AWS 네트워크 구성

### Impact

- Cross존 간 DataTransfer 비용 절감: Public ALB 대신 Private ALB를 사용하여, 존 간 데이터 전송 비용을 무료로 절감하였고, 네트워크 비용을 최적화했습니다.
- 비용 절감 외에도 보안이 향상되었으며, AWS의 기본적인 네트워크 보안 규칙을 적용하여, 존 간 트래픽을 Private하게 관리하게 되었습니다.

### Situation

- 기존에 서버 간 통신이 Public ALB를 통해 이루어졌으며, 이로 인해 Cross존 간 DataTransfer 비용이 발생하는 문제가 있었습니다.
- 비용 절감과 보안 측면에서 더 나은 방법을 찾기 위해, Private ALB를 활용하여 트래픽을 안전하게 처리하면서도 비용을 줄이는 방안이 필요했습니다.

### Action

1. Private ALB 설정을 통해 Cross존 간 통신을 Public ALB가 아닌 Private ALB를 사용하여, 비용 절감 및 보안 강화를 동시에 달성했습니다.
2. VPC 피어링 및 라우팅 테이블을 설정하여, Private ALB를 통한 서버 간 통신이 원활히 이루어지도록 구성했습니다.
3. 기존 Public ALB에서 Private ALB로의 전환 작업을 최소한의 다운타임으로 완료하고, 관련된 네트워크 설정을 최적화했습니다.

### Outcome

- Cross존 간 DataTransfer 비용을 100% 절감하였고, Public ALB 대신 Private ALB를 사용함으로써 비용 최적화를 성공적으로 수행했습니다.
- 보안 측면에서도 Private ALB를 통한 트래픽 통신으로 보안 강화와 네트워크 관리가 향상되었습니다.
- AWS 네트워크 비용 절감 및 서비스의 효율성을 높여, 전체적인 인프라 비용 절감 효과를 가져왔습니다.

## MongoDB 내재화 및 EC2 비용 최적화

진행기간: 약 2달 소요  
프로젝트 인원 수: 총 5명(+a)  
본인 포지션: 데브옵스  
본인이 사용한 스킬: AWS, EKS, Savings Plan, Reserved Instance, 비용 최적화 전략

### Impact

- MongoDB 내재화로 인한 비용 절감: MongoDB Atlas on AWS의 기존 Contract with Usage 비용이 약 80% 절감되어 \$240,000(약 3억 5천만 원)으로 예상됨.
- EC2 + EBS 비용 최적화: 온디맨드를 Savings Plan(SP)으로 변경하여 예상 비용 \$61,000(약 8천 8백만 원)으로 최적화됨.
- AWS 크레딧 지원 예상: 온디맨드 비용의 3개월 크레딧(약 \$250,000(약 3억 6천만 원))을 AWS와의 미팅을 통해 지원받을 수 있을 것으로 예상됨.

### Situation

- MongoDB Atlas on AWS의 Contract with Usage 서비스에서 발생한 비용이 \$285,000 ~ \$318,000(약 4억 1천만 원 ~ 4억 5천만 원) 사이였음.
- 트래픽 급증, 데이터 송신, 신규 추천 모델 구축 등으로 인해 AWS 비용이 증가.
- MongoDB 내재화 및 EC2 리소스 최적화를 통해 비용 절감 방안을 마련해야 했음.
- \*Reserved Instance(RI)와 Savings Plan(SP)를 선택할 때 스펙 유연성을 고려한 최적화가 필요.

### Action

1. MongoDB 내재화 진행: MongoDB Atlas on AWS에서 EC2와 EBS를 활용하여 비용 절감을 위한 내재화를 진행.
2. AWS 할인 옵션 분석: Reserved Instance(RI)와 Savings Plan(SP)의 할인폭을 비교한 후, 스펙의 유연성이 더 큰 SP(Savings Plan)을 선택하여, EC2 및 EBS 비용을 최적화.
3. 비용 예측: MongoDB Atlas on AWS 비용을 \$302,000(약 4억 4천만 원)에서 \$61,000(약 8천 8백만 원)로 절감, 약 80% 절감이 예상됨.
4. AWS 크레딧 요청: AWS와의 정례 미팅에서 MongoDB 내재화로 인한 EC2 및 EBS 온디맨드 비용에 대해 크레딧을 요청할 계획을 세움. 예상 크레딧은 \$250,000(약 3억 6천만 원).

### Outcome

- MongoDB 내재화를 통해Atlas에서 발생하던 비용을 EC2 및 EBS로 대체, 예상되는 절감액은 \$240,000(약 3억 5천만 원)으로 약 80% 절감.
- EC2 및 EBS 최적화: Savings Plan(SP)을 적용하여 예상 비용을 \$61,000(약 8천 8백만 원)으로 최적화.
- AWS 크레딧 지원: AWS와의 미팅을 통해 \$250,000(약 3억 6천만 원) 크레딧을 지원받을 것으로 예상, 이는 MongoDB EC2 + EBS 온디맨드 비용의 3개월치에 해당.

## Elasticache MSA 구조로 분리하여 가용성 확보

진행기간: 약 1주 소요  
프로젝트 인원 수: 총 1명  
본인 포지션: 데브옵스  
본인이 사용한 스킬: Terraform 활용

### Impact

- Redis 클러스터를 MSA(마이크로서비스 아키텍처) 형태로 분리하여 가용성을 확보하였고, 서비스 장애 발생 시 리스크 감소와 트래픽 분산 효과를 얻었습니다.
- 단일 Redis 인스턴스의 의존성을 없애고, 각 서비스별 독립적인 Redis 인스턴스로 분리함으로써 고가용성과 성능 최적화를 달성했습니다.
- 자동화된 Terraform 코드를 사용하여 인프라의 코드화와 구성 관리의 효율성을 극대화했습니다.

### Situation

- 기존의 Mono 형태의 Redis 클러스터에서 발생하는 단일 장애점(Single Point of Failure) 문제를 해결하기 위해, 고가용성을 보장하는 MSA 구조로의 분리가 필요했습니다.
- Redis의 성능 문제와 확장성 부족을 해결하고, 각 서비스의 독립적인 리소스 관리가 필요하였습니다.

### Action

- Terraform을 활용하여 Redis 클러스터를 MSA 구조로 분리하는 자동화된 인프라 구성을 설계하고 구현했습니다.
- 각 마이크로서비스별로 독립적인 Redis 인스턴스를 구성하여 서비스 간 리소스 격리를 구현하고, 성능과 가용성을 최적화했습니다.
- 고가용성을 보장하기 위해 Redis replication과 클러스터링을 적용하여 장애 발생 시 빠르게 대응할 수 있도록 했습니다.
- 기존 인프라와의 호환성을 고려하여 서비스 마이그레이션 계획을 세우고, 최소한의 다운타임으로 구조 변경을 완료했습니다.

### Outcome

- Redis 클러스터를 MSA 구조로 성공적으로 분리하여, 서비스 독립성과 가용성을 확보하였으며, \*\*단일 장애점(Single Point of Failure)\*\*을 제거했습니다.
- 서비스별 Redis 인스턴스를 통한 성능 향상과 효율적인 리소스 관리를 달성했습니다.
- 자동화된 Terraform 코드로 인프라 관리의 효율성을 높였으며, 향후 확장성과 유지보수가 용이한 구조를 구축했습니다.

## AWS 인프라 비용 최적화(CloudFront + DataTransfer 비용 절감)

진행기간: 약 2주 소요  
프로젝트 인원 수: 총 2명  
본인 포지션: 데브옵스  
본인이 사용한 스킬: Datadog 연결을 PrivateLink를 이용, 백엔드 서비스 CloudFront 적용

### Impact

- 비용 절감: CloudFront와 DataTransfer 관련 비용을 \$101,939.69에서 \$73,454.80로 절감, 약 28% 비용 절감.
- 비용 최적화를 통해 네트워크 트래픽 및 데이터 전송 비용을 크게 낮추었으며, 서비스 안정성을 유지하면서도 비용 효율성을 높였습니다.

### Situation

- CloudFront와 DataTransfer 관련 비용이 상당히 높아지고 있었고, 트래픽 증가로 인해 비효율적인 데이터 전송이 발생하고 있었습니다.
- Datadog와 백엔드 서비스에서 발생하는 데이터 전송 비용을 절감하는 것이 중요한 과제로 떠오르며, 온프레미스 및 IaaS 형태의 배치성 Job을 EKS로 마이그레이션하는 작업과 병행하여 네트워크 비용 최적화가 필요했습니다.
- Datadog와 CloudFront 적용에 따른 데이터 전송 비용을 절감하고, 네트워크 환경을 최적화해야 하는 상황이었습니다.

### Action

- Datadog 연결을 PrivateLink를 이용하여 데이터 전송 경로를 최적화하고, 데이터 전송 비용을 절감할 수 있도록 조정했습니다.
- 백엔드 서비스에 CloudFront 적용하여 정적 콘텐츠 서빙의 비용 효율성을 높였으며, 데이터 전송 비용을 크게 줄였습니다.
- EKS 마이그레이션을 통해 온프레미스 및 IaaS 형태의 배치성 Job을 EKS로 이동시켜, 네트워크 리소스와 비용 관리를 최적화했습니다.
- CloudFront와 DataTransfer의 비용 분석을 통해 불필요한 트래픽을 줄이고, 서비스 최적화를 위한 구체적인 액션 아이템을 도출하여 실행했습니다.

### Outcome

- CloudFront와 DataTransfer 비용을 \$101,939.69에서 \$73,454.80로 절감하였으며, 28% 비용 절감을 달성했습니다.
- PrivateLink를 통한 Datadog 연결과 CloudFront 적용으로 효율적인 네트워크 리소스 관리와 비용 최적화를 구현했습니다.
- EKS 마이그레이션과 병행한 네트워크 최적화로, 운영 효율성을 높이고 비용 절감을 달성하였습니다.

## AWS 인프라 비용 최적화(EC2 비용 절감)

진행기간: 약 2주 소요

프로젝트 인원 수: 총 1명

본인 포지션: 데브옵스

본인이 사용한 스킬: FitCloud, AWS Trusted Advisor, kubecost, Karpenter NodePool, pod 리소스 최적화, 인스턴스 스펙 조정, arm64 아키텍처로 변경

### Impact

- EC2 비용 절감: 일일 비용이 대략 \$3,200 ~ \$5,000에서 \$1,900 ~ \$2,700으로 절감되었습니다.
- 비용 최적화를 통해 월간 EC2 리소스 비용을 \$1,500 ~ \$2,300 정도 절감하며, 효율적인 자원 사용을 실현했습니다.

### Situation

- AWS 인프라 비용 중에서 EC2 리소스가 상당한 비중을 차지하고 있었고, 비효율적인 리소스 사용과 과잉 프로비저닝으로 인해 높은 운영 비용이 발생하고 있었습니다.
- 기존 EC2 인스턴스 스펙 및 NodePool 설정이 최적화되지 않아, 과도한 리소스 소비 및 불필요한 비용이 발생했습니다.
- 비용 절감과 효율적인 리소스 사용을 위해 EC2 인스턴스 최적화 및 자원 관리를 개선하는 프로젝트가 필요했습니다.

### Action

- FitCloud와 AWS Trusted Advisor를 사용하여 EC2 비효율적인 리소스 사용 및 비용 절감 포인트를 식별했습니다.
- kubecost를 활용하여 클러스터 자원 사용 현황과 비용 분석을 통해 비효율적인 리소스를 제거하거나 최적화했습니다.
- Karpenter NodePool을 활용하여 EC2 인스턴스의 자동 스케일링을 최적화하고, 필요에 따라 인스턴스 크기를 자동으로 조정하여 불필요한 리소스를 절감했습니다.
- Pod 리소스 최적화를 통해 비효율적인 리소스 소비를 줄였고, 리소스 할당을 적절하게 최적화하여 EC2 비용을 낮추었습니다.
- 인스턴스 스펙 조정을 통해 과잉 프로비저닝된 인스턴스를 적정 크기로 축소하고, arm64 아키텍처로 변경하여 비용 효율성을 높였습니다.

### Outcome

- EC2 비용 절감을 실현하여, 일별 비용을 \$3,200 ~ \$5,000에서 \$1,900 ~ \$2,700로 절감했습니다.
- AWS 인프라 비용을 최적화하여 비효율적인 자원 소비를 줄였고, 운영 비용을 약 40% 이상 절감하는 데 성공했습니다.
- 자동 스케일링과 리소스 최적화를 통해 더 효율적이고 비용 효율적인 AWS 인프라를 구현했습니다.

## 올웨이즈 배치성 Job EKS 마이그레이션

진행기간: 약 3주 소요

프로젝트 인원 수: 총 1명

본인 포지션: 데브옵스

본인이 사용한 스킬: 컨테이너라이징, 코드 리팩토링, K8S CronJob(+Job), Karpenter, VPA

### Impact

- 올웨이즈 서비스의 모든 배치성 Job을 EKS로 마이그레이션하여 운영 효율성과 가용성을 크게 향상시켰습니다.
- 정산 관련 Job들을 EKS로 마이그레이션하면서 리소스 관리가 자동화되었고, 비용 효율성도 크게 개선되었습니다.
- Job 관리가 클라우드 네이티브 환경에서 더욱 효율적이고 일관되게 되었습니다.

### Situation

- 온프레미스 및 IaaS 형태의 배치성 Job들은 관리 및 확장성 측면에서 비효율적이었으며, 장애 발생 시 신속한 대응이 어려운 상황이었습니다.
- 정산 Job은 빈번히 발생하고, 자원 스케일링이 필요한 상황이었지만, 기존 시스템에서는 비효율적인 자원 관리와 수동적인 작업 처리가 문제였습니다.
- EKS로의 마이그레이션을 통해 클라우드 기반의 자원 관리와 자동화된 리소스 스케일링을 목표로 했습니다.

### Action

- 컨테이너라이징: 기존 온프레미스 및 IaaS 환경에서 실행되던 배치성 Job들을 컨테이너화하여 EKS 환경에 맞게 변환했습니다.
- 코드 리팩토링: 배치성 Job들을 K8S CronJob과 Job으로 변환하여, 정기적 작업 관리와 단기 작업 처리를 클라우드 네이티브 환경에서 효과적으로 처리할 수 있도록 리팩토링했습니다.
- Karpenter: Karpenter를 사용하여 EKS 클러스터의 자동화된 자원 스케일링을 구현했습니다. 이를 통해 비용 최적화와 효율적인 리소스 관리를 달성했습니다.
- VPA (Vertical Pod Autoscaler): VPA를 적용하여 Pod의 리소스를 동적으로 조정, 리소스 낭비를 줄이고 최적화된 성능을 유지했습니다.
- 정산 Job EKS로 마이그레이션: 정산 관련 배치성 Job들을 EKS로 마이그레이션하여, 자동화된 리소스 관리와 효율적인 배치 처리가 가능해졌습니다.

### Outcome

- 모든 배치성 Job을 EKS로 마이그레이션함으로써, 자원의 관리와 모니터링이 자동화되었고, 운영의 효율성이 크게 향상되었습니다.
- 정산 관련 Job들의 EKS 환경에서의 관리가 가능해져, 정산 처리 시간이 줄어들고, 비용 최적화가 이루어졌습니다.
- Karpenter와 VPA를 활용하여 자원 스케일링을 자동화하고, 비용 효율성과 리소스 관리가 크게 향상되었습니다.
- 자동화된 배치성 작업 처리로 서비스 가용성이 개선되고, 장애 대응 속도가 빨라졌습니다.

## 올웨이즈 추천 및 검색 서비스 EKS 마이그레이션

진행기간: 약 2주

프로젝트 인원 수: 총 1명

본인 포지션: 데브옵스

본인이 사용한 스킬: Route53 weighted routing, preStop hook + terminationGracePeriodSeconds, StartupProbe, pod QoS, EndpointSlice, Gunicorn worker & thread, pod identity

### Impact

- 추천 및 검색 서비스를 EKS로 성공적으로 마이그레이션하여, 서비스 관리와 운영 효율성을 크게 향상시켰습니다.
- EKS 환경에서의 안정성을 확보하며, 서비스 가용성을 극대화시켰습니다.

### Situation

- 올웨이즈 추천 및 검색 서비스는 기존 환경에서 성능 저하와 운영 효율성 문제가 있었으며, 확장성과 운영 관리가 부족한 상황이었습니다.
- 기존 환경에서의 서비스 배포 및 관리가 비효율적이었고, 트래픽 증가와 서비스 확장에 대한 준비가 미비했습니다.
- Kubernetes 기반의 EKS로 마이그레이션을 통해 자동 스케일링, 효율적인 배포 및 관리 환경을 구축하고자 했습니다.

### Action

- Route53 weighted routing을 이용해 트래픽 분산을 최적화하고, EKS로의 트래픽 전환을 원활하게 진행하였습니다.
- preStop hook과 terminationGracePeriodSeconds를 사용하여 서비스 종료 시 안정성을 확보하고, 서비스 종료 후의 리소스 청소를 적절히 처리했습니다.
- StartupProbe를 활용하여, 서비스 초기화 과정에서 발생할 수 있는 문제를 사전에 방지하고, 서비스 안정성을 높였습니다.
- Pod QoS와 EndpointSlice를 설정하여 리소스 할당과 서비스 트래픽 관리를 최적화했습니다.
- Gunicorn worker 및 thread 설정을 통해 서비스 성능을 최적화하고, 트래픽 급증에 대비하였습니다.
- Pod identity를 사용하여 보안을 강화하고, 서비스 간 통신을 안전하게 처리할 수 있도록 설정했습니다.

### Outcome

- EKS로의 마이그레이션을 통해, 추천 및 검색 서비스의 가용성과 확장성을 크게 향상시켰습니다.
- 트래픽 분산과 서비스 종료 시 안정성 확보를 통해, 서비스가 더욱 신뢰성 있게 운영되었습니다.
- 리소스 관리 최적화와 성능 개선을 통해, 운영 비용 절감과 서비스 품질 향상에 기여하였습니다.

## 올웨이즈 추천 및 검색 서비스 경량화 및 리소스 최적화

진행기간: 약 2주

프로젝트 인원 수: 총 1명

본인 포지션: 데브옵스

본인이 사용한 스킬: Python, Flask, VPA, Datadog APM + Profiling

### Impact

- 애플리케이션의 로드 시간을 최대 11분에서 48초 이내로 감소시켜 서비스 가용성을 대폭 향상시켰습니다.
- 비효율적인 모듈화를 제거하여 코드 가시성을 확보하고, 서비스 관리의 효율성을 개선했습니다.
- 메모리 사용량을 20% 절감하여 자원 효율성을 극대화하고, 운영 비용 절감을 이루었습니다.

### Situation

- 올웨이즈 추천 및 검색 서비스는 초기 EC2 인스턴스 로드 시간이 약 11분에 달했으며, 이로 인해 서비스 가용성 및 운영 효율성에 큰 문제가 있었습니다.
- 애플리케이션 로드 시간이 길어지면서 ALB(Target Group)에 등록되는 시간이 늦어져, 서비스 시작에 시간 지연이 발생하고 있었습니다.
- 메모리 사용량이 많아져, 리소스 최적화가 필요했으며, 이를 통해 운영 비용 절감 및 효율적인 리소스 관리가 중요한 과제가 되었습니다.
- 애플리케이션 리팩토링 및 경량화 작업을 통해 로드 시간 단축과 메모리 사용 최적화를 위한 솔루션이 요구되었습니다.

### Action

- 애플리케이션 로드 시간 단축:
  - EC2 및 애플리케이션 로드 시간이 최대 11분에서 48초 이내로 단축되도록 애플리케이션 최적화 및 ALB Target Group 등록 시간을 대폭 개선했습니다.
  - Flask 기반 애플리케이션을 리팩토링하여 불필요한 초기화 작업을 제거하고, 초기화 속도를 향상시켰습니다.
  - VPA(Vertical Pod Autoscaler)를 활용해 자동 스케일링 및 리소스 할당 최적화를 진행하여, 리소스 사용량을 조정하고, 빠른 서비스 시작이 가능하도록 했습니다.
- 비효율적 모듈화 제거 및 코드 가시성 확보:
  - 애플리케이션 내 비효율적인 모듈화를 제거하고, 코드 가시성을 높여 디버깅 및 운영 관리의 효율성을 높였습니다.
  - 코드 품질 개선을 통해 팀원 간 협업을 원활하게 하고, 문제 해결 속도를 빠르게 했습니다.
- 메모리 사용량 최적화:
  - Datadog APM 및 Profiling 도구를 활용하여 메모리 사용량을 모니터링하고, 최적화 지점을 도출하여 20% 절감을 달성했습니다.
  - 프로파일링을 통한 병목 현상 분석을 통해 불필요한 메모리 소비를 줄이고, 메모리 최적화를 실현했습니다.

### Outcome

- EC2 인스턴스 로드 시간을 최대 11분에서 48초 이내로 단축시켜, 서비스 시작 시간을 크게 개선하였으며, ALB에 등록되는 시간도 현저히 단축되었습니다.
- 비효율적인 모듈화 제거 및 코드 리팩토링을 통해 코드 가시성을 확보하고, 디버깅 및 문제 해결 시간을 단축시켰습니다.
- 메모리 사용량을 20% 절감하여, 리소스 관리 최적화와 비용 절감에 기여하였고, 운영 효율성을 크게 향상시켰습니다.

## 데이터독 대시보드 및 알람 고도화

진행기간: 약 1주

프로젝트 인원 수: 총 1명

본인 포지션: 데브옵스

본인이 사용한 스킬: Datadog Dashboard, Synthetic Monitoring, APM, Monitoring

### Impact

- 올웨이즈 서비스의 Observability 환경을 고도화하여, 모니터링 및 알림 시스템을 강화했습니다.
- 기존의 데이터독 대시보드를 최신화 및 대체하여, 서비스의 성능 모니터링 및 문제 발생 시 신속한 대응이 가능해졌습니다.
- Synthetic Monitoring 및 APM(Application Performance Monitoring) 도입을 통해, 서비스의 성능 및 건강 상태에 대한 깊이 있는 분석이 가능해졌고, 문제 예측과 조기 경고가 이루어졌습니다.

### Situation

- 기존의 모니터링 및 대시보드 시스템은 올웨이즈 서비스의 요구 사항에 맞지 않았으며, 알람 설정이 부족해 서비스 문제를 사전에 감지하고 대처하기 어려웠습니다.
- Observability 환경이 부실하여, 문제 발생 시 빠른 대응이 어려웠고, 이에 따라 서비스의 안정성을 확보하는데 한계가 있었습니다.
- 올웨이즈 서비스의 성능 및 운영 상태를 정확하고 빠르게 모니터링하기 위해 대시보드와 알람 고도화가 필요했습니다.

### Action

#### 1. 기존 대시보드 대체:

- Datadog 대시보드를 최신화하고, 올웨이즈 서비스에 최적화된 대시보드를 설계하여 실시간 모니터링이 가능하도록 구축했습니다.
- 데이터 시각화 및 상태 모니터링에 필요한 핵심 지표들을 선별하여 직관적이고 효율적인 대시보드를 제작했습니다.

#### 2. Synthetic Monitoring 도입:

- Synthetic Monitoring을 활용하여, 서비스의 가용성 및 성능을 실시간으로 모니터링하고, 자동화된 테스트를 통해 외부 요청 시나리오를 시뮬레이션하여 서비스 장애를 빠르게 감지할 수 있도록 설정했습니다.

#### 3. APM(Application Performance Monitoring) 설정:

- APM을 이용해 서비스 성능 분석 및 지연 시간을 측정하고, 트랜잭션 성능을 모니터링하여 서비스 병목 현상을 식별하고 개선점을 도출했습니다.

#### 4. 알람 시스템 고도화:

- Datadog 알람 시스템을 설정하여, 서비스의 중요 지표(예: 오류율, 지연 시간, 자원 사용량 등)에 대해 자동화된 알람을 생성하도록 했습니다.
- 알람 트리거를 정확하게 설정하여, 문제가 발생하기 전에 미리 감지하고 팀에 즉시 알림을 제공할 수 있도록 했습니다.

### Outcome

- 올웨이즈 서비스의 Observability 환경을 성공적으로 구축하고, 서비스 모니터링 및 경고 시스템을 강화했습니다.
- 기존 대시보드를 대체하여, 효율적이고 직관적인 대시보드를 통해 서비스 성능 및 상태를 실시간으로 확인할 수 있게 되었고, 문제 발생 시 신속한 대응이 가능해졌습니다.
- Synthetic Monitoring을 통해 서비스의 가용성과 성능을 테스트 및 예측할 수 있는 시스템을 구축했습니다.
- APM을 활용하여 성능 모니터링을 강화하고, 서비스 최적화를 위한 데이터 기반 인사이트를 도출했습니다.
- 알람 시스템을 고도화하여 문제가 발생하는 즉시 경고를 받아 서비스 가용성을 더욱 향상시켰습니다.

## 올웨이즈 Python 기반 백엔드 서비스 EKS 마이그레이션

진행기간: 약 2주

프로젝트 인원 수: 총 1명

본인 포지션: 데브옵스

본인이 사용한 스키ル: Route53 weighted routing, preStop hook + terminationGracePeriodSeconds, readinessProbe, pod QoS, Gunicorn worker & thread, pod identity

### Impact

- Python 기반 쇼츠 백엔드 서비스를 성공적으로 EKS 환경으로 마이그레이션 하여, 서비스 안정성 및 확장성을 크게 향상시켰습니다.
- EKS 환경으로의 이전을 통해, 운영 효율성을 높이고 서비스의 유연성과 장애 대응 속도를 개선했습니다.
- Route53 weighted routing, preStop hook, readinessProbe, terminationGracePeriodSeconds, Gunicorn을 사용하여 서비스의 배포 및 롤백 전략을 강화하고, 서비스 가용성을 보장했습니다.

### Situation

- 기존의 Python 기반 백엔드 서비스는 비효율적인 환경에서 운영되고 있었고, 이를 EKS 클러스터로 마이그레이션하여 보다 효율적이고 안정적인 운영 환경을 구축해야 했습니다.
- 서비스 가용성과 탄력성을 높이기 위해 자동화된 배포 및 관리 시스템이 필요했으며, EKS의 확장성을 활용할 수 있도록 기존 시스템을 클라우드 환경에 적합하게 재설계해야 했습니다.
- 쇼츠 백엔드 서비스의 배포 및 유지보수 효율성을 높이기 위한 해결책으로, Kubernetes 기반의 EKS 클러스터로의 마이그레이션이 결정되었습니다.

### Action

#### 1. EKS 클러스터 설계 및 설정:

- Route53 weighted routing을 설정하여, 서비스의 트래픽 분산 및 배포를 효율적으로 관리하고, 서비스의 가용성을 높였습니다.
- EKS 클러스터 내에 Pod QoS를 설정하여, 리소스 관리와 서비스 우선순위를 조정하고, 자원의 최적화를 이뤄냈습니다.

#### 2. 서비스 배포 전략 및 고가용성 확보:

- preStop hook과 terminationGracePeriodSeconds를 사용하여, 서비스의 배포와 롤백 시 안정성을 확보했습니다. 이를 통해 서비스가 종료되기 전에 필요한 작업을 진행하고, 서비스 중단 없이 배포할 수 있었습니다.
- readinessProbe를 사용하여 서비스의 상태를 체크하고, 트래픽을 적시에 수용할 수 있도록 했습니다.
- Gunicorn worker 및 thread 설정을 통해, 서비스의 성능 최적화와 동시 처리 성능을 개선했습니다.

#### 3. Kubernetes 리소스 및 배포 자동화:

- Pod Identity를 설정하여 권한 관리 및 보안성을 강화하였고, 자동화된 배포 시스템을 구축하여, 향후 시스템의 확장성 및 유연성을 보장했습니다.
- EKS 클러스터의 Pod 관리 및 자동 확장 기능을 활용하여, 서비스가 예상보다 많은 트래픽을 처리할 수 있도록 시스템을 최적화했습니다.

### Outcome

- 쇼츠 백엔드 서비스를 EKS 환경으로 마이그레이션 완료하여, 서비스 안정성과 확장성을 크게 향상시켰습니다.
- 트래픽 분산 및 서비스 가용성을 보장하는 Route53 weighted routing을 성공적으로 설정하여, 서비스 다운타임을 최소화하고 실시간 트래픽 관리가 가능해졌습니다.
- EKS 클러스터 내에서의 자원 관리 및 서비스 배포가 원활히 이루어져, 빠르고 안정적인 서비스 운영을 가능하게 했습니다.
- 배포 후 안정성 확보를 위한 preStop hook, terminationGracePeriodSeconds, readinessProbe 등을 설정하여, 서비스 중단 없이 원활한 배포 및 롤백을 지원했습니다.
- Gunicorn worker 및 thread 최적화로 서비스의 동시 처리 성능을 개선하고, 리소스 활용 효율성을 향상시켰습니다.

## 데이터독 기반 Observability 구축

진행기간: 약 2주

프로젝트 인원 수: 총 1명

본인 포지션: 데브옵스

본인이 사용한 스킬: Datadog

### Impact

- 데이터독 기반 Observability를 구축하여, 서비스 및 인프라의 가시성과 문제 해결 능력을 크게 향상시켰습니다.
- Beta, Prod 환경에 Datadog Operator 에이전트를 성공적으로 설치하여, 실시간 모니터링과 로그 수집을 자동화하고, 어플리케이션과 인프라에 대한 상세한 통찰을 제공하게 되었습니다.
- 기본 대시보드와 알람 시스템을 설정하여, 문제가 발생할 경우 즉각적인 경고를 받을 수 있도록 했습니다.
- 로그, APM 및 메트릭 데이터를 Beta 및 Prod 환경에서 수집하여, 성능 문제나 예기치 않은 장애를 조기에 감지할 수 있었습니다.
- NPM, CI 파이프라인 및 외부 서비스와 Datadog 통합을 통해, 서비스의 전체적인 모니터링 환경을 통합하고, 데이터를 효율적으로 처리할 수 있는 시스템을 구축하였습니다.

### Situation

- 서비스 및 인프라에서 발생하는 문제를 빠르고 정확하게 파악할 수 있는 방법이 부족했으며, 실시간으로 인프라 상태와 애플리케이션 성능을 모니터링 할 필요성이 있었습니다.
- 기존에 운영 중인 시스템은 분산된 모니터링 툴을 사용하여, 전체적인 시스템 상태를 일관되게 파악하기 어려웠고, 장애 발생 시 대응 속도가 낮았습니다.
- 외부 서비스 및 CI 파이프라인에 대한 모니터링을 통합하여, 전체적인 시스템에 대한 가시성을 확보하는 것이 중요한 요구사항으로 떠올랐습니다.

### Action

- Datadog 설치 및 구성:
  - Datadog Operator를 사용하여 Beta 및 Prod 환경에 Datadog 에이전트를 설치하고, 서비스 상태를 실시간으로 모니터링할 수 있도록 했습니다.
  - 기본 대시보드와 알람 시스템을 설정하여, 문제가 발생할 경우 실시간으로 알림을 받도록 했습니다.
- 데이터 수집 및 모니터링 환경 구축:
  - 로그, APM, 메트릭 데이터를 Beta, Prod 환경에서 수집하여, 서비스의 성능 및 상태를 실시간으로 분석할 수 있는 환경을 구축했습니다.
  - NPM, CI 파이프라인, 외부 서비스와 Datadog 통합을 통해 전체적인 모니터링 환경을 중앙화하고, 서비스 및 인프라의 가시성을 확보했습니다.
- 알람 및 모니터링 개선:
  - 알람 시스템을 통해 장애 발생 전 경고를 받도록 설정하여, 서비스 장애 발생 전에 즉각적인 대응을 할 수 있도록 했습니다.
  - 메트릭 및 로그를 기반으로 성능 문제를 신속히 식별하고, 이를 해결하기 위한 조치를 빠르게 취할 수 있도록 했습니다.

### Outcome

- 모든 환경에서 Datadog를 통한 실시간 모니터링을 구축하여, 시스템의 가시성을 크게 향상시켰습니다.
- 문제 발생 시, 빠르게 경고를 받고 즉시 대응할 수 있는 대시보드 및 알람 시스템을 설정하여, 서비스 안정성을 강화했습니다.
- 로그, APM, 메트릭 데이터 수집을 통해, 서비스 성능과 인프라 상태를 실시간으로 파악하고, 장애를 예방할 수 있었습니다.
- 외부 서비스 및 CI 파이프라인과의 통합을 통해, 모든 서비스를 하나의 통합된 플랫폼에서 모니터링할 수 있도록 했습니다.

## 올웨이즈 NodeJS 기반 백엔드 서비스 EKS 마이그레이션

진행기간: 약 1주 소요

프로젝트 인원 수: 총 3명

본인 포지션: 데브옵스

본인이 사용한 스키ル: Route53 weighted routing, preStop hook + terminationGracePeriodSeconds, readinessProbe, pod QoS

### Impact

- 80%의 NodeJS 기반 백엔드 서비스를 AWS EKS 클러스터로 성공적으로 마이그레이션하여 서비스의 확장성과 관리 효율성을 크게 향상시켰습니다.
- Route53 weighted routing을 통해 트래픽을 EKS로 분배하며, 마이그레이션을 안전하고 단계적으로 수행할 수 있었습니다.
- 서비스 안정성을 높이기 위해 preStop hook, terminationGracePeriodSeconds, readinessProbe를 활용하여 서비스 중단 없이 마이그레이션을 진행하였고, pod QoS 설정을 통해 리소스 품질을 보장하며 안정적인 서비스 운영을 지원하였습니다.

### Situation

- NodeJS 기반 백엔드 서비스는 기존 서버 환경에서 운영되었으나, 확장성 및 유지보수의 어려움으로 인해 EKS로의 마이그레이션이 필요했습니다.
- 기존 인프라의 스케일링 및 배포 자동화가 부족하여, AWS EKS의 장점인 관리형 클러스터와 자동화된 배포 환경으로 전환해야 했습니다.
- 마이그레이션 과정에서 서비스의 중단 없이 안정적으로 트래픽을 전환하고, 새로운 클러스터 환경에서 고가용성 및 확장성을 보장하는 것이 주요 요구 사항이었습니다.

### Action

#### 1. EKS 환경 준비:

- EKS 클러스터 설정과 NodeJS 기반 백엔드 서비스의 배포 환경을 EKS에 최적화하였습니다.
- Route53 weighted routing을 사용하여 기존 서버와 EKS 클러스터 간의 트래픽을 안정적으로 분배하고, 점진적으로 트래픽을 EKS로 이동할 수 있도록 하였습니다.

#### 2. 서비스 마이그레이션:

- preStop hook을 사용하여 서비스 종료 시점을 정확하게 제어하고, terminationGracePeriodSeconds를 설정하여 서비스 종료와 재시작 시 충분한 시간을 확보했습니다.
- readinessProbe를 설정하여 서비스 준비 상태를 정확하게 판단하고, 클러스터 내에서 트래픽이 정상적으로 처리되도록 보장했습니다.
- pod QoS 설정을 통해 각 pod의 리소스 품질을 보장하고, 트래픽과 리소스의 효율적인 배분을 진행했습니다.

#### 3. 고가용성 및 확장성 보장:

- EKS 클러스터 내 자동화된 스케일링을 활용하여 트래픽 급증 시에도 안정적으로 서비스를 제공할 수 있도록 했습니다.
- 서비스의 안정성을 고려하여 중단 없는 마이그레이션을 목표로 했으며, 새로운 클러스터에서 안정적으로 서비스가 운영될 수 있도록 철저하게 테스트와 검증을 진행했습니다.

#### 4. 트래픽 전환 및 모니터링:

- Route53 weighted routing을 활용하여 트래픽을 EKS로 점진적으로 이동시켰고, 초기 트래픽을 분배한 후 실시간 모니터링을 통해 문제를 사전 식별하고 해결했습니다.

### Outcome

- 약 80%의 NodeJS 기반 백엔드 서비스를 EKS로 마이그레이션하여, 클라우드 환경에서 확장성과 효율성을 크게 향상시켰습니다.
- Route53 weighted routing을 통한 트래픽 분배로 서비스 중단 없이 EKS로의 전환을 완료했습니다.
- 서비스 안정성을 위한 preStop hook, terminationGracePeriodSeconds, readinessProbe 설정을 통해 서비스가 중단 없이 운영되도록 보장했습니다.
- 리소스 품질을 보장하고, 자동화된 클러스터 관리를 통해 백엔드 서비스 운영의 효율성을 크게 향상시켰습니다.

## 쿠버네티스(EKS) 구축

진행기간: 2023.11.15 ~ 2023.11.30

프로젝트 인원 수: 총 1명

본인 포지션: 데브옵스

본인이 사용한 스킬: Terraform, Add-on, ArgoCD

### Impact

- 쿠버네티스 아키텍처 설계 및 EKS 구축을 통해 서비스 마이그레이션을 위한 안정적이고 확장 가능한 인프라를 구축하였습니다.
- 다양한 Add-on 및 툴을 EKS 환경에 설치하여, 클러스터의 효율적인 운영과 관리가 가능하도록 하였습니다.
- EKS 환경 구축 후, Kubernetes 클러스터에 필요한 다양한 Add-on을 자동화된 방식으로 배포하고 관리하여, 운영 효율성을 크게 향상시켰습니다.

### Situation

- 서비스 마이그레이션을 위해 기존 환경에서 AWS EKS 클러스터로의 전환이 필요했습니다.
- 기존 온프레미스 환경에서 Kubernetes로의 마이그레이션을 위한 아키텍처 설계와 최적화가 필요했습니다.
- 쿠버네티스 클러스터의 효율적인 운영을 위해 다양한 관리 툴과 Add-on을 설치하고 구성해야 했습니다.
- EKS 클러스터의 설정 및 관리가 자동화되도록 Terraform과 ArgoCD를 활용하여 클러스터와 애플리케이션을 효율적으로 관리할 필요가 있었습니다.

### Action

#### 1. 쿠버네티스 아키텍처 설계:

- 네이밍 컨벤션 정의 및 격리 공간을 설계하여 환경 구성 시 효율성을 높였음.
- EKS 클러스터 버전 업그레이드 정책을 수립하여 클러스터의 안정성을 유지하며 버전 관리가 용이하도록 설계.
- 메니페스트 관리 도구로 ArgoCD를 선택하여 배포 파이프라인을 자동화하고, 지속적인 배포(CD) 프로세스를 구현.

#### 2. EKS 클러스터 구축:

- Terraform을 사용하여 Managed Node Group 및 Bottlerocket 기반의 EC2 인스턴스를 자동화하여 클러스터를 구축.
- VPC와 Subnet 설정을 통해 네트워크 구성을 최적화하고, 보안 그룹 및 라우팅 테이블 설정을 정의하여 네트워크 격리 및 보안을 강화.

#### 3. Addon 설치 및 관리:

- ArgoCD, Karpenter, CoreDNS, Kube-proxy, VPC-CNI와 같은 핵심 Add-on을 EKS 클러스터에 설치하여, 자동화된 배포, 클러스터 자원 관리, DNS 관리 등 다양한 기능을 구현.
- Argo Rollouts, AWS Ingress Controller, cert manager, cluster autoscaler, external dns, external secret, Kyverno, metrics server 와 같은 부가적인 Add-on을 설정하여 클러스터의 유연성, 보안성 및 모니터링 기능을 강화.
- Cluster Proportional Autoscaler, Keel, ArgoCD Notification을 설정하여 자동화된 환경을 구현하고, 리소스 관리 및 알림을 효율적으로 처리.

#### 4. EKS 클러스터 운영 효율화:

- Karpenter를 사용하여 EC2 인스턴스의 수요에 따라 자동으로 확장/축소하도록 설정하여 비용 최적화 및 리소스 효율성을 극대화.
- External Secret 및 Cert Manager를 통해 외부 비밀 관리 및 SSL 인증서를 자동으로 갱신하는 시스템을 구축.

#### 5. 자동화 및 CI/CD 파이프라인 구현:

- Terraform을 통한 자동화된 인프라 프로비저닝과 ArgoCD를 통한 자동 배포 파이프라인을 구축하여, 빠르고 안전한 클러스터 환경을 구성.
- Continuous Integration 및 Continuous Deployment 시스템을 통해 매번 클러스터의 상태를 자동으로 추적하고, 애플리케이션 배포를 자동화하여 안정적인 서비스 운영을 지원.

### Outcome

- 서비스 마이그레이션을 위한 EKS 클러스터 구축이 성공적으로 완료되었으며, 이는 클라우드 환경에서의 안정성과 확장성을 크게 향상시켰습니다.
- 자동화된 클러스터 운영 및 다양한 Add-on 관리를 통해 리소스 효율성을 높이고, 개발 및 운영팀의 업무 부담을 경감시켰습니다.
- Terraform과 ArgoCD를 활용한 관리 및 배포 시스템은 인프라 변경 및 애플리케이션 업데이트 시 신속하고 안전한 적용을 가능하게 하여, 지속적인 서비스 제공을 보장했습니다.
- Karpenter와 Autoscaler를 사용하여 EC2 인스턴스의 확장/축소를 최적화하고, 비용 절감을 실현하였습니다.

## AWS 네트워크 아키텍처 재수립

진행기간: 2023.11.09 ~ 2023.11.14

주요내용: AWS환경 코드화 하기 전, 올웨이즈 네트워크 아키텍처 재수립

프로젝트 인원 수: 총 1명

본인 포지션(기여도): 테브옵스

본인이 사용한 스킬: VPC, Subnet, I/G, Nat G/W, Nacl, Security Group, Route table

### Impact

- 인프라 환경 정의 및 대역대 정리로 전체 네트워크 아키텍처가 명확하게 정리됨.
- AWS 서비스 및 리소스 네이밍 컨벤션을 수립하여, 향후 리소스 추가 및 관리에서 일관성 유지.
- 네트워크 구성을 제작하여, 다양한 팀과의 협업 시 명확한 시각적 자료 제공. 이를 통해 네트워크 트러블슈팅 시 빠른 문제 파악이 가능해짐.
- 이후 코드화 작업을 위한 강력한 기반 마련, 이후 인프라 자동화와 코드화 작업을 위한 청사진을 제공.

### Situation

- 이전에 사용하던 네트워크 아키텍처가 비효율적이고, 코드화되지 않아 네트워크 리소스 관리 및 트러블슈팅 시 어려움이 많았음.
- 새롭게 클라우드 인프라를 코드화하고, 지속 가능한 네트워크 관리를 위해 아키텍처 재구성이 필요했음.
- 기존 아키텍처의 복잡성과 비효율적인 대역대 사용으로 인해 새로운 네트워크 요구사항에 대응하기 어려움.

### Action

- 단순히 기존의 네트워크 구조를 수정하는 것이 아니라, 향후 확장성을 고려한 코드화된 클라우드 네트워크 구조를 수립하는 것이었음. 이를 위해 대역 대 정리 및 효율적인 리소스 관리를 위한 리소스 네이밍 컨벤션을 수립.
- 전체 인프라 아키텍처의 가장 중요한 부분인 네트워크 보안과 접근 통제 방안을 우선적으로 다루었음. 이를 통해 리소스를 효율적으로 분배하고, 불필요한 과잉 설계를 방지.
- 기존 시스템을 분석하고, 새로운 아키텍처 설계를 위해 빠르게 실행한 후, 문제를 점검하고 개선하는 사이클을 반복. 각 단계에서 발생한 문제점을 빠르게 학습하고 해결책을 반영.
- 비록 프로젝트 팀원은 1명이었지만, 다양한 부서와의 협업을 통해 피드백을 받으며 진행. 팀 간 긴밀한 소통과 빠른 피드백을 통해 프로젝트를 원활하게 진행.
- 이 프로젝트는 전체 네트워크 인프라의 재설계로, LEVIT의 전체 인프라 환경에 영향을 미쳤음. 이는 다른 부서와의 협업 및 장기적인 비용 절감, 효율성 향상에 기여할 것으로 예상됨.

# (주)야놀자 (2022.06 ~ 2023.10)

## 전사 EKS 마이그레이션 서비스 개발

진행기간: 2023.08.25 ~

주요내용: 전사 EKS 마이그레이션을 위한 서비스 개발

프로젝트 인원 수: 총 1명

본인 포지션(기여도): 개발

본인이 사용한 스킬: Spring-Boot(kotlin), Redis, Spring Cloud Config, Jenkins API, AWS SDK, Bitbucket API  
결과

## 전사 EKS 환경구축

진행기간: 2023.08.17 ~ 2023.09.18

주요내용: 전사 EKS 운영정책 고도화 및 환경구축

프로젝트 인원 수: 총 6명

본인 포지션(기여도): 운영, 개발

본인이 사용한 스킬: Terraform, Jenkins, ArgoCD, GitOps Mechanism, Add On services(Argo Rollouts, aws-auth, ebs-csi-driver, aws-loadbalancer, vpc-cni, calico, cluster-overprovisioning, coredns, dashboard, external-dns, external-secret, istio ingress G/W, istiod, karpenter, kube-proxy, kyverno, metrics-server, prometheus)

결과

- Stage 환경 싱글 클러스터 프로비저닝 / 관련 Add On 서비스 프로비저닝
- Stage 환경 멀티 클러스터(red/black) 프로비저닝 / 관련 Add On 서비스 프로비저닝
- 전사 공통 base helm chart 정의서 생성

## 전사 배포서비스(YOLO) 개발 및 운영 / CI&CD 운영

진행기간: 2023.04.01 ~

주요내용: CI&CD 안정화 및 고도화

프로젝트 인원 수: 총 1명

본인 포지션(기여도): 운영, 개발

본인이 사용한 스킬: java, kotlin, spring-boot

결과

- 배포서비스 YOLO SLA99.9% 지원
- Jenkins CI & ArgoCD 운영

## 메타 DB(CMDB) 수집기 스케줄링 동작 및 프로젝트 안정화

진행기간: 2023.03.05 ~ 2023.03.31

주요내용: 수집기 스케줄링 동작 및 프로젝트 안정화

프로젝트 인원 수: 총 3명

본인 포지션(기여도): 개발(30%)

본인이 사용한 스킬: java, spring-batch, Jenkins, pushgateway, prometheus, grafana, loki  
결과

- 배치 애플리케이션을 위한 Observability 설계 및 구현
- CI/CD 구축 및 고도화

## 메타 DB(CMDB) 수집기 모델링 및 저장 구현

진행기간: 2023.02.20 ~ 2023.03.03

주요내용: 설계된 모델을 바탕으로 수집기 구현

프로젝트 인원 수: 총 3명

본인 포지션(기여도): 개발(20%)

본인이 사용한 스킬: java, spring-batch

결과

- 특정 AWS 자원(route53, organization, RDB) 모델링 및 수집기 구현

## 인터파크 전자금융망 내에 민감정보 관리 내부서비스 구축

진행기간: 2023.01.09 ~ 2023.03.20

주요내용: 인터파크에서 애플리케이션 구동시 필요한 민감정보를 컨트롤 할 cloud properties를 전자금융망 내에 구축

프로젝트 인원 수: 총 3명(클라우드 인프라 1명, DBA 1명, 본인)

본인 포지션(기여도): 구축(25%), 개발(100%)

본인이 사용한 스킬: KMS, RDS, Elastic Beanstalk, Spring Boot, Jenkins

결과

- 기한 내 구성완료 및 공유
- 관련 가이드문서 작성

## Bitbucket 운영정책 수립 및 CI 고도화

진행기간: 2023.01.03 ~ 2023.01.31

주요내용: 팀 내부에 적용할 Bitbucket 운영정책 수립 및 CI 고도화

프로젝트 인원 수: 총 1명

본인 포지션(기여도): 개발(100%)

본인이 사용한 스킬: Jenkins, Bitbucket Server, Groovy, Shell Script, Sonarqube, detekt, ktlint, git hook

결과

- Bitbucket 운영정책 수립
- CI 적용(Lint 검사 -> 정적분석 -> 단위&통합테스트 -> Sonarqube)
- GitOps 적용
- 자동배포 적용
- 정적분석 후, 위반되는 코드에 분석결과 자동 코멘트
- 관련 가이드문서 작성

## 인터파크를 위한 민감정보 관리 내부서비스 구축

진행기간: 2022.12.08 ~ 2022.12.29

주요내용: 인터파크에서 애플리케이션 구동시 필요한 민감정보를 컨트롤 할 cloud properties를 구축

프로젝트 인원 수: 총 3명(클라우드 인프라 1명, DBA 1명, 본인)

본인 포지션(기여도): 구축(25%), 개발(100%)

본인이 사용한 스킬: KMS, RDS, Elastic Beanstalk, Spring Boot, Jenkins

결과

- 기한 내 구성완료 및 공유
- 관련 가이드문서 작성

## EKS 내부서비스 마이그레이션

진행기간: 2022.09.26 ~ 2022.12.05

주요내용: 팀 내부서비스를 EKS 환경에 마이그레이션을 진행

프로젝트 인원 수: 3명

본인 포지션(기여도): 인프라(80%), 지원(30%), 개발(60%)

본인이 사용한 스킬: Helm, Kustomize, ArgoCD, Argo Rollouts, ExternalDNS, Vault, Harbor, Jenkins

결과

- 팀 내부서비스 EKS DEV 환경에 마이그레이션 완료
- 마이그레이션 매뉴얼 작성완료
- EKS 관련문서 작성 및 공유

## EKS 이전에 필요한 모니터링 인프라 시범운영

진행기간: 2022.08.16 ~ 2022.09.27

주요내용: 인프라 변경으로 인한 모니터링 누락 없이 기존과 동일한 서비스를 제공

프로젝트 인원 수: 4명

본인 포지션(기여도): 개발(30%)

본인이 기여한 점: 기존 모니터링 대시보드와 EKS 환경에 특화된 서비스 모니터링 샘플 대시보드 제작완료

본인이 사용한 스킬: Prometheus, Grafana, cAdvisor, Node exporter, kube-static, metric, PromQL

결과

- EKS 환경의 모니터링 인프라 복잡도 낮춤
- 기존 모니터링 대시보드와 EKS 환경에 특화된 서비스 모니터링 샘플 대시보드 제작완료

## 계열사를 위한 민감정보 관리 내부서비스 구축

진행기간: 2022.07.25 ~ 2022.08.23

주요내용: 멤버사에서 애플리케이션 구동시 필요한 민감정보를 컨트롤 할 cloud properties를 구축

프로젝트 인원 수: 총 3명(클라우드 인프라 1명, DBA 1명, 본인)

본인 포지션(기여도): 구축(25%), 개발(100%)

본인이 사용한 스킬: KMS, RDS, Elastic Beanstalk, Spring Boot, Jenkins

결과

- 기한 내 구성완료 및 공유
- 관련문서 작성 후 팀내공유

# (주)카사코리아 (2021.09 ~ 2022.06)

## Kasa Korea 웹거래소 인프라 구축

진행기간: 2022.04 ~ 2022.04

주요내용: 코리아 웹거래소 오픈을 위한 인프라 구축

프로젝트 인원 수: 1명(본인)

본인 포지션(기여도): 인프라(100%)

본인이 기여한 점: 배포 및 전반적인 인프라환경 세팅

본인이 사용한 스킬: Route53, CloudFront, ArgoCD, DroneCI, Kubernetes

결과: 코리아 웹거래소 오픈

## Kasa Singapore 거래소의 Metric기반 Observability 환경구축

진행기간: 2022.03 ~ 2022.04

주요내용: Singapore Kasa거래소의 Metric기반 Observability 환경구축

프로젝트 인원 수: 1명(본인)

본인 포지션(기여도): 인프라(100%)

본인이 기여한 점: Metric 데이터 시각화

본인이 사용한 스킬: Github, Terraform, Ansible, docker, docker-compose, Telegraf, MSK, InfluxDB, Grafana, Slack

결과: Metric 데이터 시각화

## Kasa Singapore 거래소 AWS에 Site-To-Site VPN 연결

진행기간: 2022.01 ~ 2022.01

주요내용: Singapore Kasa거래소 AWS의 TransitG/W에 내부망에 존재하는 팔로알토 VPN 장비 연결

프로젝트 인원 수: 2명(본인포함)

본인 포지션(기여도): 인프라(50%)

본인이 기여한 점: 프라이빗 네트워크 구성

본인이 사용한 스킬: TransitG/W

결과: TransitG/W에 내부 VPN장비 연결

## Kasa Singapore 거래소 CI&CD 구축

진행기간: 2021.11 ~ 2021.12

주요내용: Singapore Kasa거래소 배포시스템 구축

프로젝트 인원 수: 1명(본인)

본인 포지션(기여도): 인프라(100%)

본인이 기여한 점: CI&CD 프로세스 구축으로 인한 개발자들 배포속도 향상

본인이 사용한 스킬: Github, Kustomize, Kubernetes, DroneCI, ArgoCD, argocd-vault-plugin, AWS Load Balancer Controller, Cluster Autoscaler, HPA, kube2iam, certmanager, AWS SecretsManager

결과: DroneCI, ArgoCD를 쿠버네티스 환경에 구축

## Kasa Singapore 거래소 AWS 인프라 구축

진행기간: 2021.11 ~ 2021.12

주요내용: Singapore Kasa거래소 인프라 dev, staging, prod구축

프로젝트 인원 수: 1명(본인)

본인 포지션(기여도): 인프라(100%)

본인이 기여한 점: AWS Control Tower 및 SSO를 도입하여 Account관리용이하게 함, GitOps전략을 적용하여 Terraform Cloud와 연동, Github IaC 관리, AWS IaC로 프로비저닝 및 관리

본인이 사용한 스킬: Github, Terraform, AWS Control Tower, AWS SSO, AWS IAM, AWS KMS, AWS Secrets Manager, AWS VPC(Subnet, Route Table, EIP, Security group, NACL, InternetG/W, NatG/W), Route53, ACM, CloudFront, S3, ALB, NLB, EC2, EKS, ECR, MSK, VPC Peering, RDS, Aurora

결과

- Github과 Terraform으로 코드 및 상태관리
- Terraform을 모듈화하여 staging/prod 구축때는 프로비저닝시간 포함하여 최대 1시간 이내로 모두 구축 완료

## Kasa Singapore Github 인프라 구축

진행기간: 2021.11 ~ 2021.11

주요내용: Terraform으로 Singapore Github 인프라구축 및 관리

프로젝트 인원 수: 1명(본인)

본인 포지션(기여도): 인프라(100%)

본인이 기여한 점: Github Terraform으로 코드화

본인이 사용한 스킬: Terraform Cloud, Github

결과: Github 코드로 관리

## Kasa Singapore 거래소 네트워크 아키텍쳐 설계 및 사전조사

진행기간: 2021.10 ~ 2021.10

주요내용: Singapore Kasa거래소 인프라 아키텍쳐 설계

프로젝트 인원 수: 1명(본인)

본인 포지션(기여도): 인프라(100%)

본인이 기여한 점: 네트워크 아키텍쳐 설계, AWS 서비스 아키텍쳐 설계, SaaS 서비스 및 3rd-party 구매 및 구축(Redis Cloud, Github, Slack, Sentry, Codecov, Mailgun, Twilio)

본인이 사용한 스킬: Confluence, DrawIO

결과: 초기 아키텍쳐 수립으로 인한 빠른 인프라 구축가능

# (주)오픈메이트 (2016.11 ~ 2021.08)

## 트렌드온 서비스 정기 월간 업데이트 배치 프로그램 제작 및 스케줄링 적용

진행기간: 2021.05 ~ 진행중

주요내용: 월간 Nice, KT 데이터를 테스트 및 운영 데이터베이스에 월간 자동 업데이트 함

프로젝트 인원 수: 1명(본인)

본인 포지션(기여도): 개발(100%), 인프라(100%)

본인이 기여한 점: 데이터 업데이트 자동화

본인이 사용한 스킬: docker, python, celery, SQS, ECS, CodeBuild, CodePipeline, SNS

결과: 월간 Nice, KT 데이터를 데이터베이스에 월간 자동 업데이트 함

## AWS dev/staging서버 인프라 변경

진행기간: 2021.05 ~ 2021.05

주요내용: 해당 서버의 인프라 환경을 ECS(EC2기반)에서 ECS(Fargate기반)으로 변경

프로젝트 인원 수: 1명(본인)

본인 포지션(기여도): 인프라(100%)

본인이 기여한 점: 사용한만큼 과금하는 Fargate서비스로의 마이그레이션을 통한 비용 절감

본인이 사용한 스킬: ECS, ALB, CodePipeline

결과: 독립적인 환경 구축 및 비용절감

## AWS Activate Portfolio 신청

진행기간: 2021.05 ~ 2021.05 주요내용: AWS Activate Portfolio 신청으로 인한 크레딧 지원

프로젝트 인원 수: 1명(본인)

본인 포지션(기여도): 인프라(100%)

본인이 기여한 점: AWS Activate Portfolio 신청의 모든 프로세스 담당

본인이 사용한 스킬: 영어

결과: 3개월 운영비용 확보

## 컨테이너 기반 Geoserver 구축

진행기간: 2021.05 ~ 2021.05

주요내용: 기존 설치형 Geoserver를 컨테이너 기반으로 변경

프로젝트 인원 수: 1명(본인)

본인 포지션(기여도): 인프라(100%)

본인이 기여한 점: 컨테이너 유지관리를 위한 구조 변경

본인이 사용한 스킬: docker, ECS, ECR

결과: Geoserver 구축

## 데이터 크롤링 프로그램 스케줄링 적용

진행기간: 2021.05 ~ 2021.05

주요내용: 스마트온 서비스의 코로나 데이터 수집을 위한 프로그램에 스케줄링 적용 및 오류발생시 슬랙채널에 메시지 푸쉬  
프로젝트 인원 수: 1명(본인)

본인 포지션(기여도): 개발(100%), 인프라(100%)

본인이 기여한 점: 스마트온 서비스의 코로나 데이터 수집을 위한 프로그램에 스케줄링 적용

본인이 사용한 스킬: docker, docker-compose, python, celery, SQS, ECS, CodeBuild, CodePipeline, SNS

결과: 9:00, 12:00, 18:00, 22:00에 주기적으로 데이터 크롤링

## 세이프온 서비스 AWS로 이관

진행기간: 2021.04 ~ 2021.04

주요내용: 세이프온 서비스 AWS로 이관

프로젝트 인원 수: 1명(본인)

본인 포지션(기여도): 인프라(100%)

본인이 기여한 점: 세이프온 AWS로 인프라 설계 및 이관

본인이 사용한 스킬: IAM, Route53, CloudFront, ACM, S3, ALB, ECS(EC2기반), ECR, CodeCommit, CodeBuild, CodeDeploy, CodePipeline, RDS

결과: AWS상에서 세이프온 서비스 운영

## 트렌드온 서비스 AWS로 이관

진행기간: 2021.04 ~ 2021.04

주요내용: 트렌드온 서비스 AWS로 이관

프로젝트 인원 수: 1명(본인)

본인 포지션(기여도): 인프라(100%)

본인이 기여한 점: 트렌드온 AWS로 인프라 설계 및 이관

본인이 사용한 스킬: IAM, Route53, CloudFront, ACM, S3, ALB, ECS(EC2기반), ECR, CodeCommit, CodeBuild, CodeDeploy, CodePipeline, RDS

결과: AWS상에서 트렌드온 서비스 운영

## 데이터온 서비스 AWS로 이관

진행기간: 2021.03 ~ 2021.03

주요내용: 데이터온 서비스 AWS로 이관

프로젝트 인원 수: 1명(본인)

본인 포지션(기여도): 인프라(100%)

본인이 기여한 점: 데이터온 AWS로 인프라 설계 및 이관

본인이 사용한 스킬: IAM, Route53, CloudFront, ACM, S3, ALB, ECS(EC2기반), ECR, CodeCommit, CodeBuild, CodeDeploy, CodePipeline, RDS, SQS

결과: AWS상에서 데이터온 서비스 운영

## 홈페이지/스마트온 서비스 AWS로 이관

진행기간: 2021.02 ~ 2021.02 주요내용: 홈페이지/스마트온 서비스 AWS로 이관

프로젝트 인원 수: 1명(본인)

본인 포지션(기여도): 인프라(100%)

본인이 기여한 점: 홈페이지/스마트온 AWS로 인프라 설계 및 이관

본인이 사용한 스킬: IAM, Route53, CloudFront, ACM, S3, ALB, ECS(EC2기반), ECR, CodeCommit, CodeBuild, CodeDeploy, CodePipeline, RDS, SQS

결과

- AWS상에서 홈페이지 운영
- AWS상에서 스마트온 서비스 운영

## AWS Activate Founders 신청

진행기간: 2021.02 ~ 2021.02 주요내용: AWS Activate Founders 신청으로 인한 크레딧 지원

프로젝트 인원 수: 1명(본인)

본인 포지션(기여도): 인프라(100%)

본인이 기여한 점: AWS Activate Founders 신청의 모든 프로세스 담당

본인이 사용한 스킬: 영어

결과: AWS 마이그레이션을 위한 테스트환경 구축시의 비용확보 약 4개월 운영비용 확보

## 신규서비스 개발 / 인프라구축

진행기간: 2020.10 ~ 2020.12

주요내용: 공공데이터기반 누구나 쉽게 찾고 활용하는 데이터 플랫폼

프로젝트 인원 수: 8명

본인 포지션(기여도): 프론트엔드(40%), 인프라(100%), 유지보수(25%)

본인이 기여한 점: 프론트엔드 개발, AWS PaaS기반 인프라 구축, 대용량 데이터(CSV) API 기능 개발

본인이 사용한 스킬: ECS(EC2기반), RDS, Route53, CloudFront, S3, ELB, IAM, VPC, NuxtJS, Javascript, HTML, CSS, GitLab CI & CD, dJango, MongoDB

결과: 데이터온 서비스 운영

## 서비스 리뉴얼

진행기간: 2020.06 ~ 2020.09

주요내용: 서비스 리뉴얼

프로젝트 인원 수: 2명

본인 포지션(기여도): 프론트엔드(100%), 데이터베이스(100%), 유지보수(50%)

본인이 기여한 점: 프론트엔드 개발, 데이터업데이트

본인이 사용한 스킬: HTML, Javascript(es3), Jquery, CSS, D3, postgreSQL, Docker

결과: 트렌드온 version2 UI로 서비스 제공

## 홈페이지 리뉴얼

진행기간: 2020.05 ~ 2020.06

주요내용: (주)오픈메이트온 홈페이지 리뉴얼

프로젝트 인원 수: 2명

본인 포지션(기여도): 프론트엔드(100%), 인프라(100%), 유지보수(100%)

본인이 기여한 점: 프론트엔드 개발, 클라우드 IaaS 기반 인프라 구축, SPA 구조적용, 모바일 지원, SSL 적용

본인이 사용한 스킬: HTML5, Javascript(es5 이상), Jquery, CSS3

결과: 홈페이지 리뉴얼 UI로 운영

## 4.15 총선 페이지 개발

진행기간: 2020.04 ~ 2020.04

주요내용: 4.15 총선을 위한 해당 지역구의 후보자 정보 제공 서비스

프로젝트 인원 수: 3명

본인 포지션(기여도): 프론트엔드(50%), 유지보수(50%)

본인이 기여한 점: 프론트엔드 개발, SSL 적용, Google Analytics 적용, 모바일 지원

본인이 사용한 스킬: VueJS, Javascript(es5 이상), LeafletJS, D3

결과: 스마트온 서비스 내에 4.15 총선 결과 페이지 제공

## 유지보수

진행기간: 2020.02 ~ 2020.03

주요내용: (주)오픈메이트온 내부서비스 월 데이터 정기 업데이트, 웹 어플리케이션 유지보수

프로젝트 인원 수: 2명

본인 포지션(기여도): 백엔드(50%), 프론트엔드(50%), 데이터베이스(50%), 인프라(100%), 유지보수(50%)

본인이 기여한 점: 데이터 업데이트, 데이터 딥프로세스 제작

본인이 사용한 스킬: PostgreSQL, Docker, Spring, Mybatis, JSP, JSTL, HTML, CSS, Javascript(es3)

## 신규 서비스 인프라 환경구축

진행기간: 2019.10 ~ 2020.01

주요내용: 개발 및 운영 서비스 인프라 환경 구축

프로젝트 인원 수: 1명

본인 포지션(기여도): 인프라(100%), 유지보수(100%)

본인이 기여한 점: 클라우드 IaaS 기반 인프라 구축, GitLab 구축, Reverse Proxy Server 구축, On-premise 인프라 구축(개발용), Private Docker Repository 구축

본인이 사용한 스킬: Docker, NginX, Apache

결과

- 설치형 GitLab 운영

- 개발 & 운영 인프라 동기화

## 신규 서비스 개발

진행기간: 2019.10 ~ 2020.01

주요내용: 공공데이터 기반으로 빅데이터&인공지능으로 진단하고 처방하는 도시분석 서비스  
프로젝트 인원 수: 6명

본인 포지션: 프론트엔드(100%), 인프라(100%), 유지보수(50%)

본인이 기여한 점: 프론트엔드 개발, 클라우드 IaaS 기반 인프라 구축, SSL 적용

본인이 사용한 스킬: VueJS, Javascript(es5 이상), HTML, CSS, Vuetify

결과: 스마트온 서비스 제공

## Trend-On 서비스 SSL 구입 및 설치

진행기간: 2019.09 ~ 2019.09

주요내용: 해외 SSL 구매 중계사인 GoGetSSL을 통한 SectigoSSL 구입 및 설치  
프로젝트 인원 수: 1명

본인 포지션(기여도): 인프라(100%)

본인이 기여한 점: 해외 중계사를 통하여 구매비용 절감

본인이 사용한 스킬: 영어, NginX+SSL

결과: 서비스 신뢰도 확보 및 인증서 구매비용 절감

## 임시 홈페이지 개발

진행기간: 2019.08 ~ 2019.09

주요내용: 임시 홈페이지 개발  
프로젝트 인원 수: 1명

본인 포지션(기여도): 프론트엔드(100%), 백엔드(100%), 유지보수(100%)

본인이 기여한 점: 프론트엔드 개발, 백엔드 개발, 메일기능 적용, SSL 적용

본인이 사용한 스킬: HTML, Javascript, Jquery, CSS, Bootstrap, PHP

## (주)오픈메이트온 내부 서비스 고도화

진행기간: 2019.06 ~ 2019.07

주요내용: 관리자 로직 변경

프로젝트 인원 수: 1명

본인 포지션(기여도): 프론트엔드(100%), 백엔드(100%), 데이터베이스(100%), 유지보수(50%)

본인이 기여한 점: 프론트엔드 개발, 백엔드 개발, SPA구조 적용, RestfulAPI 적용, DML 생성 및 수정

본인이 사용한 스킬: HTML, Javascript, CSS, Jquery, Spring, Mybatis, PostgreSQL

결과: UI상 서비스 관리자들의 이용편의성 증대

## Safe-On 서비스 SSL 구입 및 설치

진행기간: 2019.04 ~ 2019.04  
주요내용: 해외 SSL 구매 중계사인 GoGetSSL을 통한 SectigoSSL 구입 및 설치  
프로젝트 인원 수: 1명  
본인 포지션(기여도): 인프라(100%)  
본인이 기여한 점: 해외 중계사를 통하여 구매비용 절감  
본인이 사용한 스킬: 영어, Apache2+SSL  
결과: 서비스 신뢰도 확보 및 인증서 구매비용 절감

## (주)오픈메이트 유지보수

진행기간: 2017.04 ~ 2019.05  
주요내용: 자사와 계약된 공기업/사기업/내부서비스 유지보수, 데이터 정기 업데이트  
프로젝트 인원 수: 8명  
본인 포지션(기여도): 프론트엔드(50%), 백엔드(50%), 인프라(10%)  
본인이 기여한 점: 프론트엔드 유지보수, 백엔드 유지보수, 소규모 앱 인프라 구축  
본인이 사용한 스킬: HTML, CSS, Javascript, Jquery, Spring  
결과: 유지보수(이랜드, JTI, BBQ, 국가정보자원관리원, 아모레퍼시픽, 하이트진로, 중앙일보, 서울신용보증재단)

## 서비스 모니터링 서버 개발

진행기간: 2018.11 ~ 2018.11  
주요내용: 모든 서비스 모니터링  
프로젝트 인원 수: 1명  
본인 포지션(기여도): 프론트엔드(100%), 백엔드(100%), 유지보수(100%)  
본인이 기여한 점: 프론트엔드 개발, 백엔드 개발  
본인이 사용한 스킬: ExpressJS, Javascript, HTML, Bootstrap  
결과: 수동적인 응대에서 실시간 응대로 변경함으로써 응대시간 단축

## 로그데이터 적재 API 개발

진행기간: 2018.05 ~ 2018.06  
주요내용: 주소정제 로그데이터 실시간 적재  
프로젝트 인원 수: 1명  
본인 포지션(기여도): 프론트엔드(100%), 백엔드(100%), 유지보수(100%)  
본인이 기여한 점: 프론트엔드 개발, 백엔드 개발  
본인이 사용한 스킬: Spring, Javascript, HTML

---

## 내부서비스 Restful API 개발

진행기간: 2018.03 ~ 2018.05  
주요내용: 서비스 데이터 Restful 기반 API제공  
프로젝트 인원 수: 1명  
본인 포지션(기여도): 백엔드(100%), 유지보수(100%)  
본인이 기여한 점: 백엔드 개발  
본인이 사용한 스킬: Spring, Mybatis, postgreSQL  
결과: 내부서비스 사이터 데이터 납품(대전마케팅공사)

---

## Web Map Service(WMS) 솔루션 제품 개발

진행기간: 2016.11 ~ 2017.03  
주요내용: Web Map Service(WMS) 솔루션 제작  
프로젝트 인원 수: 5명  
본인 포지션(기여도): 프론트엔드(10%)  
본인이 기여한 점: 프론트엔드 개발, 서버리소스 모니터링페이지 제작, API문서페이지 제작  
본인이 사용한 스킬: HTML, CSS, Javascript, Jquery