임동현

Im Donghyeon
DevOps Engineer

고객에게 안정적인 서비스를 제공할 수 있도록
맡은 바 최선을 다 하고 항상 노력합니다.

Personal Information

Education

한성대학교 정보통신공학과
2011.03 - 2017.02
다우기술 기업연계형 장기현장실습 인턴
2016.07 - 2016.12
SBA-RAPA 클라우드 컴퓨팅 기술인재 양성과정
2018.07 - 2018.08

Skills

AWS, Aliyun, KT, NAVER, Kubernetes, Docker
GitLab, Space, AWS CodeSeries, GitHub Action
Terraform, Python, Shell Script, PowerShell
ChatGPT Plus, Cursor Pro

Certificates

Certified Kubernetes Administrator 2022.11
AWS SysOps Associate 2020.02
리눅스마스터 2 급 2018.06
정보기기운용기능사 2018.03
네트워크관리사 2 급 2018.03

Links

https://velog.io/@dongdorrong
https://github.com/dongdorrong
https://stackshare.io/dongdorrong/toolchains

Experience

인스웨이브 (2025.03 ~) Cloud Engineer

1. 기업용 애플리케이션 및 웹 표준 UI/UX 플랫폼 인프라 구축 및 운영
- W-Sharing 솔루션 EC2 → EKS 마이그레이션
- CSAP 인증을 위한 SignSquare 솔루션 NCP 공공 기반 NKS 클러스터 구축
- 비용 최적화 작업 (클라우드 계정 통·폐합, 불필요 자원 정리)

디렉셔널 (2023.08 ~ 2024.08) DevOps Engineer

1. 주식 대차 거래 플랫폼 및 가상 화폐 투자 클럽 인프라 총괄

2. 온프레미스 기반 쿠버네티스 구축 (v1.30.1)
- Prometheus-Thanos, Grafana 기반 모니터링 시스템 구축
- Sentry On Kubernetes 기반 에러 트래킹 시스템 구축
- GitOps 적용을 위해 ArgoProject 의 ArgoCD 와 ArgoRollouts 도입
- KeyCloak 을 활용한 Kuberenetes 사용자 인증 및 권한 관리
- KeyCloak 기반 ArgoCD 및 Grafana 통합 인증 (OIDC) 구축

씨알에스큐브 (2021.03 ~ 2023.03) DevOps Engineer

1. CRScube 솔루션 인프라 전반 관여 및 3 개 솔루션 전담 엔지니어

2. AWS Lambda 기반 자동화 구현
- DR 리소스 배포
- AWS System Manager Run Command
- ECS Fargate Task 덤프 생성

3. ISO27001 심사 관련 인프라 보안 개선 보조 업무 수행

디딤 365 (2018.11 ~ 2021.02) System Engineer

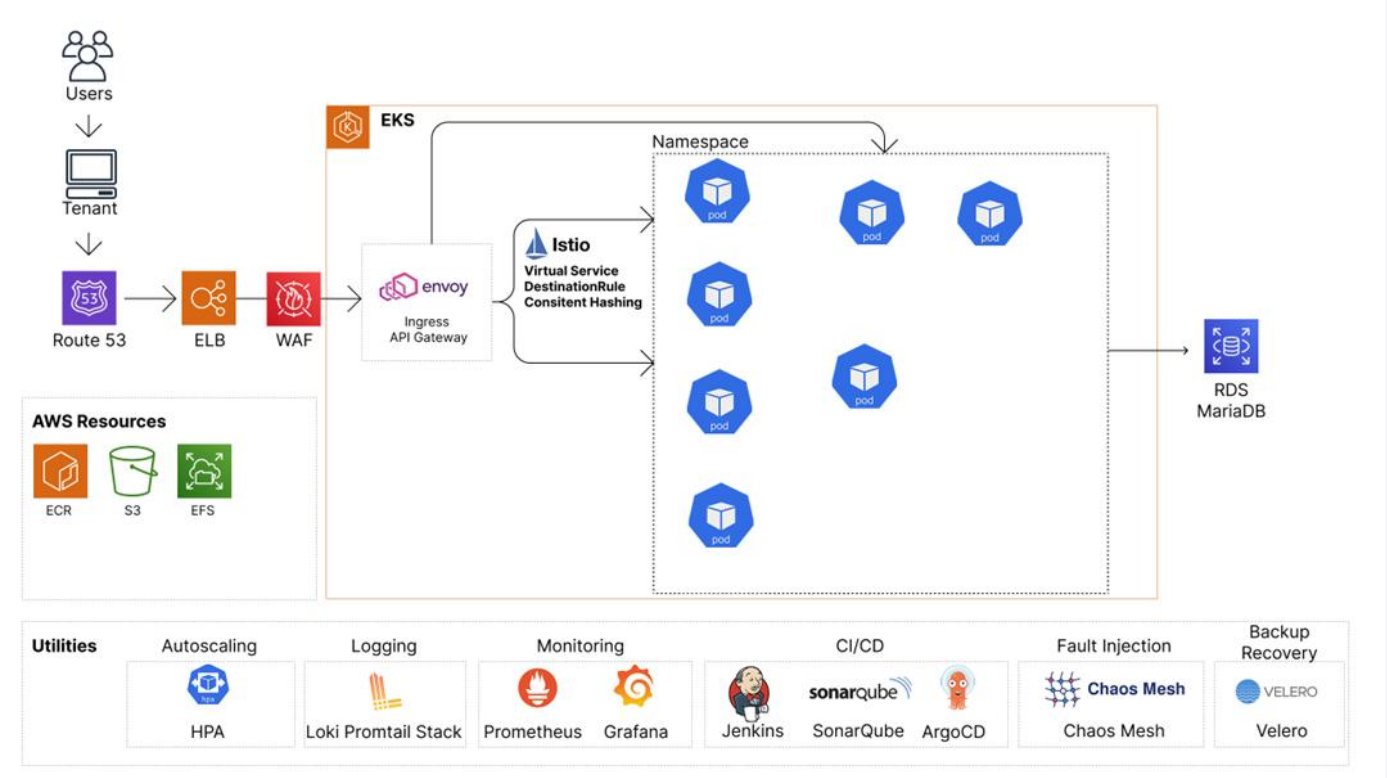
1. 전담 고객사 서비스 운영 관리 및 장애 대응 (총 약 150 대 서버)
- 서울 공공자전거, 한국 전력거래소 교육 시스템, KT 스카이라이프

2. 신규 서비스 도입에 따른 클라우드 서버 및 서비스 설치 지원
- Apache, PHP, Tomcat, MySQL, Java, Nginx, Node.js, IIS
- MySQL, MariaDB, PostgreSQL, MSSQL

Details

1. W-Sharing 솔루션 EC2 → EKS 마이그레이션, Jenkins & ArgoCD GitOps 구축

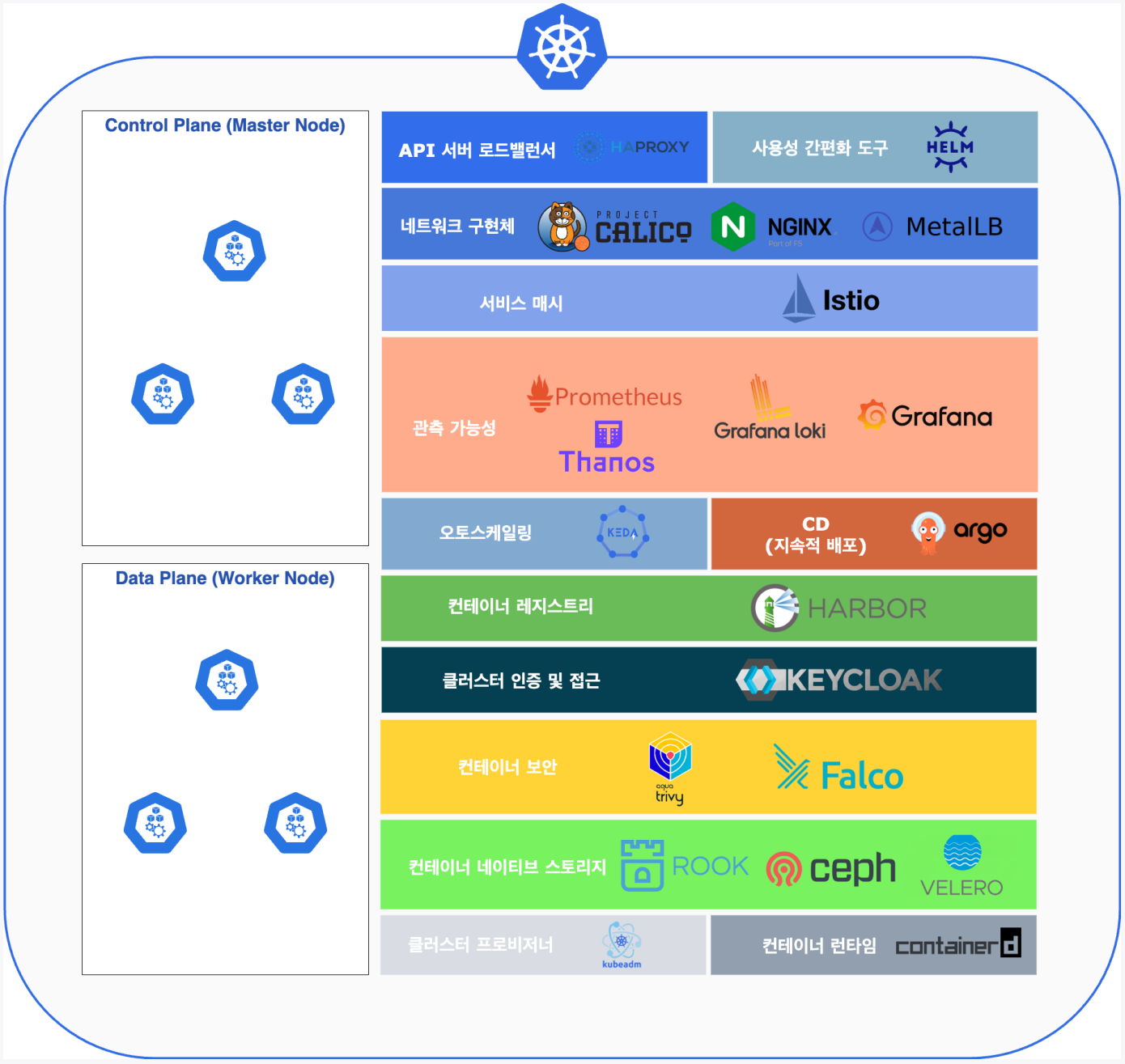
수동 관리되던 EC2 기반의 Tomcat 인프라를 EKS 로 이관하면서 컨테이너 기반의 자동화된 운영 환경을 구축하였습니다.
Ant 기반의 수동 빌드를 Jenkins Groovy 파이프라인으로 전환하고, 컨테이너 이미지 기반 CI/CD 자동화를 구현하였습니다. 또한 Istio, Prometheus, ArgoCD, Chaos-Mesh 등 다양한 도구를 활용해 관측, 배포, 복구까지 아우르는 클라우드 네이티브 인프라를 구축하였습니다.



Details

2. 온프레미스 기반 쿠버네티스 구축 (v1.30.1)

CNCF 에서 관리하는 프로젝트 중에서 커뮤니티가 활성화되어 있고 레퍼런스가 많은 것을 반영하여 쿠버네티스에 반영하였습니다.
이외에도 Istio, falco, bitnami kafka, redis, cnpg, airflow 등의 다양한 프로젝트를 PoC 해보며 시스템 구축을 진행했습니다.



3. Terraform 모듈 구현

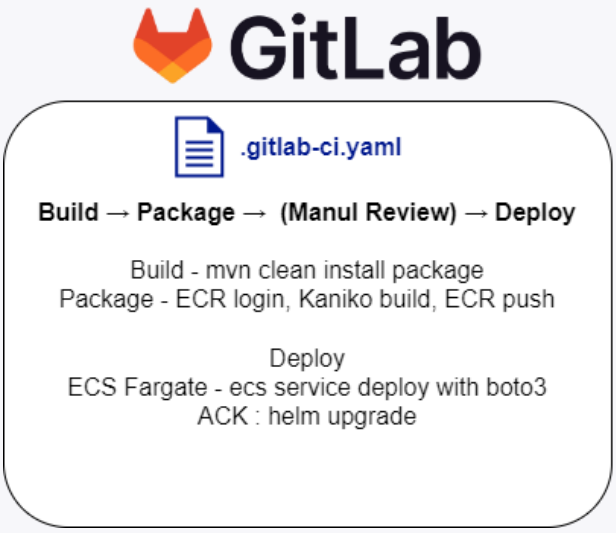
기존에 일일이 구현되어 있는 리소스를 Terraform 모듈로 변경하여 재사용성과 관리 효율성을 높이고, 코드를 단순화하고 인프라 구성 시간을 단축할 수 있었습니다.

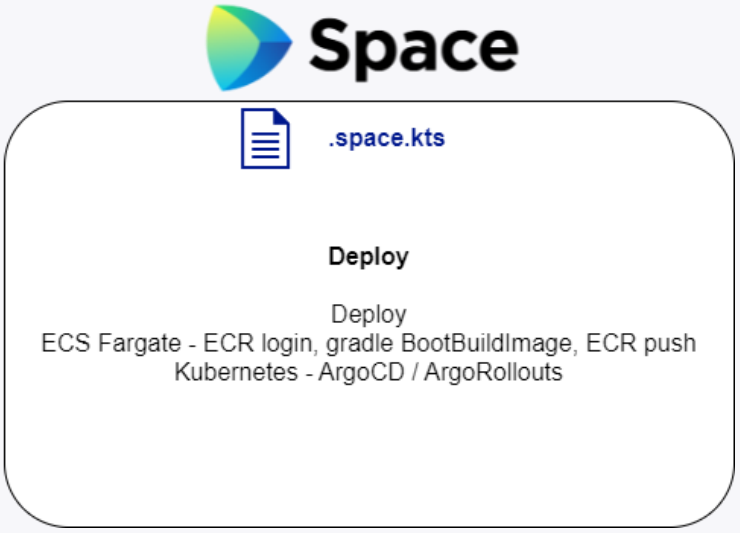
- ECS : LoadBalancer Listener, Target group, ECS Service, ECS Task Definition
- CodeSeries : CodeBuild, CodePipeline, EventBridge Rule, Event Target
- EC2 : EC2 Instance
- Batch : Batch Compute Environment, Batch Job Queue, Batch Job Definition, EventBridge Rule, Event Target
- CloudWatch Alert : SNS Topics, CloudWatch Alert

Details

4. 신규 API 서비스 배포를 위한 CI/CD 구성

서비스 개발 환경(Spring boot, Kotlin)을 고려하여 CI/CD 를 구성하고 Terraform 을 활용하여 AWS 리소스 생성
non-prod 환경은 manual review 단계 없이 즉시 배포되도록 하고, prod 환경은 배포 관리자가 승인 후 배포하도록 CI/CD 를 구성하였습니다. 애플리케이션은 AWS ECS Fargate 혹은 Kubernetes 에 배포되었습니다.





씨알에스큐브는 GitLab, 디렉셔널은 JetBrains Space 를 Git Repository 로 사용하였고 각 플랫폼에서 CI/CD 을 구현할 수 있도록 GitLabCI 와 Space Automation 기능을 제공하여 이를 활용하였습니다.

씨알에스큐브에서는 ECS 서비스에 애플리케이션을 배포할 수 있는 python boto3 를 활용한 커스텀 이미지를 활용하였고, Deploy 단계에서 해당 이미지에 ECS 서비스 이름, 이미지 태그 등을 실행 인자로 전달하여 배포를 수행하였습니다.

디렉셔널은 Gradle 플러그인의 bootBuildImage 태스크를 활용해서 CI 과정을 간소화하였습니다. 배포의 경우 AWS 에서는 CodePipeline, 쿠버네티스는 ArgoCD 와 ArgoRollouts 을 활용하였습니다.

5. AWS DNS 제한으로 인한 장애 발생 시 개선 방안 검토

여러 애플리케이션에서 'no such host', 'Could not resolve host' 메시지가 발생하였고 곧이어 장애가 발생하였습니다. 특별한 조치 없이 일정 시간이 지나고 장애가 해소되었는데, 일반적으로 DNS 에 질의를 할 수 없는 상황에서 위와 같은 상황이 발생하는 것을 인지하고 있었지만 관련된 이력을 찾을 수 없어 추후 재발 방지 차원에서 개선 방법을 검토하였습니다.

네트워크 인터페이스 별 초당 1024 개 패킷을 Route 53 Resolver 로 패킷을 보낼 수 있는데, 할당량에 도달하면 Route 53 Resolver 는 트래픽을 거부한다는 것을 확인하였습니다. AWS 공식 안내를 참고하여 DNS 캐싱을 통해 문제를 해결할 수 있다는 것을 확인하고 테스트를 진행하였고, 결과적으로 DNS 캐싱을 적용하고 나서 질의 부하가 감소하는 것을 확인하였습니다.

6. AWS System Manager Run Command 를 활용한 보안 점검 일괄 처리 자동화 구현

ISO27001 보안 점검 대상 인스턴스에 대해서 일괄적으로 점검 스크립트 파일을 실행하고, 점검 결과 파일을 S3 버킷에 업로드

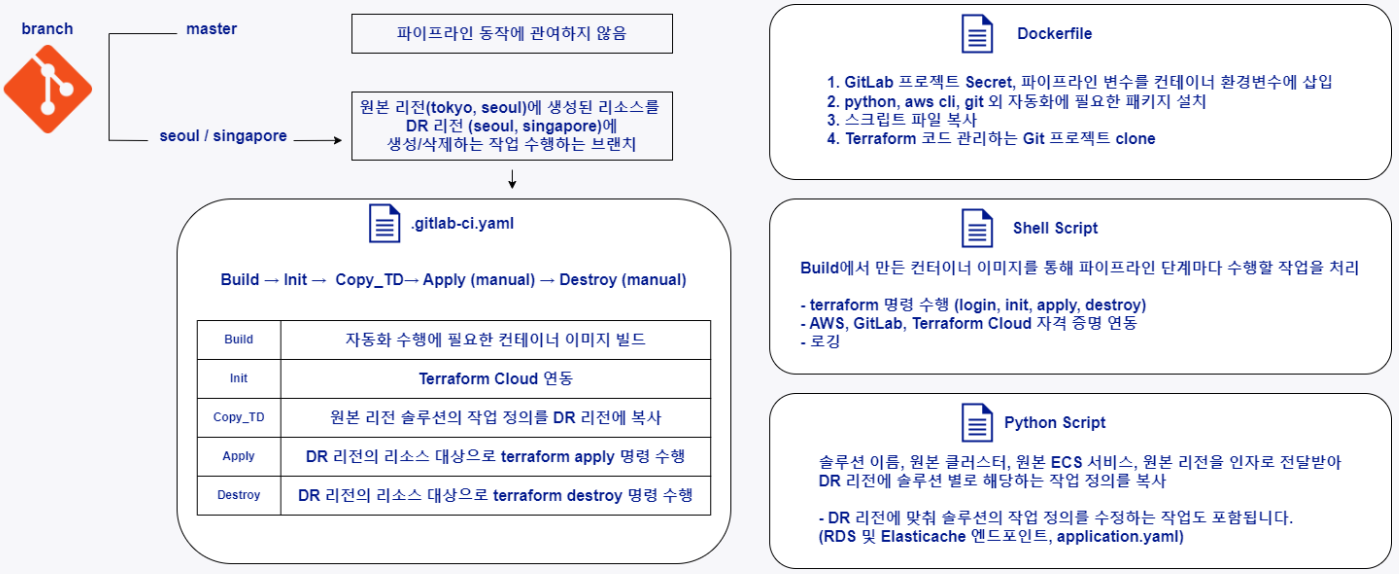
Lambda 함수를 통해 도쿄, 서울 리전의 인스턴스를 대상으로 RunCommand 동작에 필요한 AWS 리소스 생성과 권한 설정을 수행하고, 점검 완료 후 기존 인스턴스의 IAM 역할을 복원하고 리소스를 제거하도록 구현하였습니다. S3 버킷 1 개에 다중 리전 액세스 지점을 설정하여 도쿄, 서울 리전 인스턴스의 점검 결과를 취합하였고, 인스턴스의 기존 IAM 역할은 DynamoDB 에 저장합니다.

Details

7. AWS 솔루션 DR 환경 구축 및 DR 리소스 배포 자동화 구현

씨알에스큐브에서는 매년 말 BCP(Business Continuity Plan) 훈련을 진행하였고, 이를 토대로 고객사에 DR 환경으로 전환하는 것에 문제가 없음을 보고하였습니다. 당시 훈련 때마다 여러 솔루션에 대한 AWS 리소스를 생성하는 과정이 다소 복잡했기 때문에 GitLabCI 파이프라인을 활용해서 리소스를 일괄적으로 생성하고 제거하도록 자동화를 구현했습니다.

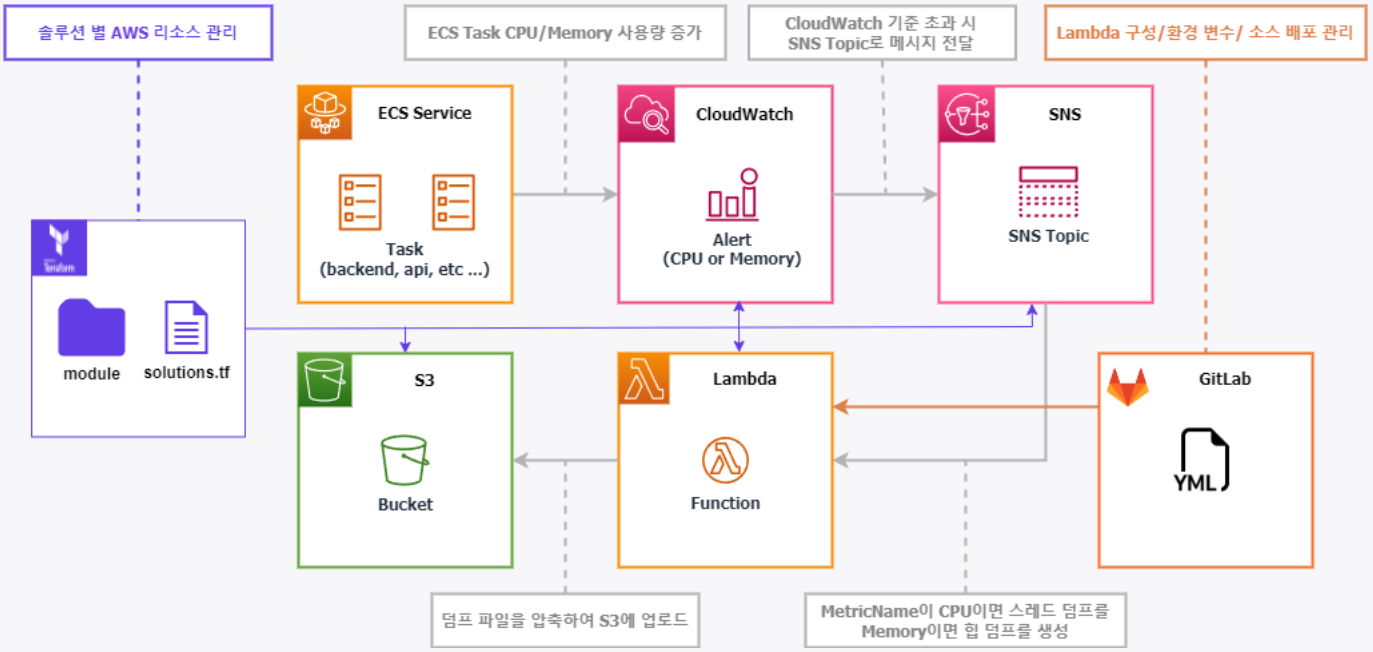
파이프라인과 자동화를 처리하는 컨테이너의 이미지를 만들고, 각 파이프라인 단계에서 해당 이미지를 불러와 단계 별 작업을 수행하도록 구현하였습니다.



8. ECS Task 덤프 생성 자동화 구현

ECS Service 의 CPU, Memory 사용량이 높아지면 Lambda 함수가 동작하게 됩니다. 이후 Lambda 함수는 Shell Script 를 ECS Task 에 주입하고, RunCommand 기능으로 ecs_excute_command 실행하여 주입된 스크립트 파일을 실행합니다. 스크립트 파일은 스레드/힙 덤프를 수행하고, 덤프 파일을 S3 버킷으로 전송합니다.

애플리케이션의 컨테이너 이미지를 빌드할 때 스크립트 파일을 복사하도록 Dockerfile 을 수정할 수도 있지만, 수많은 Dockerfile 을 일일이 작업하는 것이 번거로웠기 때문에 Lambda 를 통해 Shell Script 를 주입하도록 구현하였습니다.



Details

9. 공공자전거 인프라 운영

디딤 365 에서 공공자전거 인프라 운영을 담당하며, KT G-Cloud 환경에서 구축된 VM 기반의 서비스를 운영했습니다. Apache, Tomcat, MySQL 서버로 구성된 인프라를 관리하며, 각 서버는 이중화로 운영되어 서비스 안정성과 가용성을 보장하였습니다.

Tomcat 서버는 세션 클러스터링을 통해 이중화되어, 장애 발생 시에도 사용자의 세션이 유지되도록 구성되었습니다. Apache 서버 또한 AJP 설정을 통해 Tomcat 과 연동하여 부하 분산과 안정성을 확보하였습니다.

MySQL 데이터베이스는 이중화된 환경에서 운영되었으며, MHA(Master High Availability) 구성이 되어 있어, 데이터 무결성을 유지하면서도 고가용성을 보장할 수 있도록 설계되었습니다.

장애가 발생할 경우 고객사, 개발사, DBA 업체와 협력하여 장애 전파 및 대응 프로세스에 맞춰 신속한 복구를 진행하였습니다.