

포트폴리오

프로젝트명	Amazon EKS와 CI & CD 파이프라인을 활용한 청약 신청 웹사이트 구축
프로젝트 기간	2024.04.19 ~ 2024.06.07
프로젝트개요	EKS와 CI & CD를 활용해 원스톱 서비스를 지원하는 청약 웹 사이트 제작
구축 환경	Amazon Web Service 및 Github
주요 사용 기술	Amazon EKS, Amazon ECR, Jenkins, argoCD, Github, Docker container, HELM, kustomize, Aurora mySQL, Prometheus, Grafana
고려사항	모니터링, 운영의 고가용성 및 빠른 확장성, 비용효과, 보안 및 백업
프로젝트 내용	<ul style="list-style-type: none"> Amazon EKS 클러스터 환경 구성 및 청약 신청 웹사이트 구축 CI & CD 파이프라인을 통해 웹사이트 빌드 및 배포 자동화 <p>핵심 구현 기술(고려사항)</p> <ol style="list-style-type: none"> Kubernetes Service(EKS) <ol style="list-style-type: none"> Deployment, Service, SC, PV/PVC 오토스케일링, 로드밸런서 (고가용성, 확장성) Jenkins <ol style="list-style-type: none"> LoadBalancer DNS로 웹에서 접근 (운영 효율성) Docker를 활용한 이미지 빌드 및 파이프라인 구성 빌드된 이미지 Amazon ECR 저장 이미지 빌드 내역 확인 가능 (운영 효율성) argoCD <ol style="list-style-type: none"> LoadBalancer DNS로 웹에서 접근 (운영 효율성) EKS 내에 설치 (확장성) Github 내용과 환경을 비교해 Sync (운영 효율성) Github <ol style="list-style-type: none"> Code repository로 이용 (그룹 협업성) Code history 확인 용이 Amazon Service <ol style="list-style-type: none"> RDS를 multi-AZ 데이터베이스로 사용 (고가용성, 백업) 클러스터와 RDS 접근 bastion 분리 (보안) Amazon ECR을 사용해 용량만큼 비용발생 (비용효과) Monitoring Service <ol style="list-style-type: none"> LoadBalancer DNS로 웹에서 접근 (운영 효율성) EKS 내에 설치 (확장성)

- c. Prometheus를 통한 자원 정보 데이터 수집
- d. Grafana를 통해 수집한 데이터 시각화 (모니터링)
- e. EBS 볼륨 데이터 저장
- f. HELM 차트를 이용해 배포 (운영 효율성)

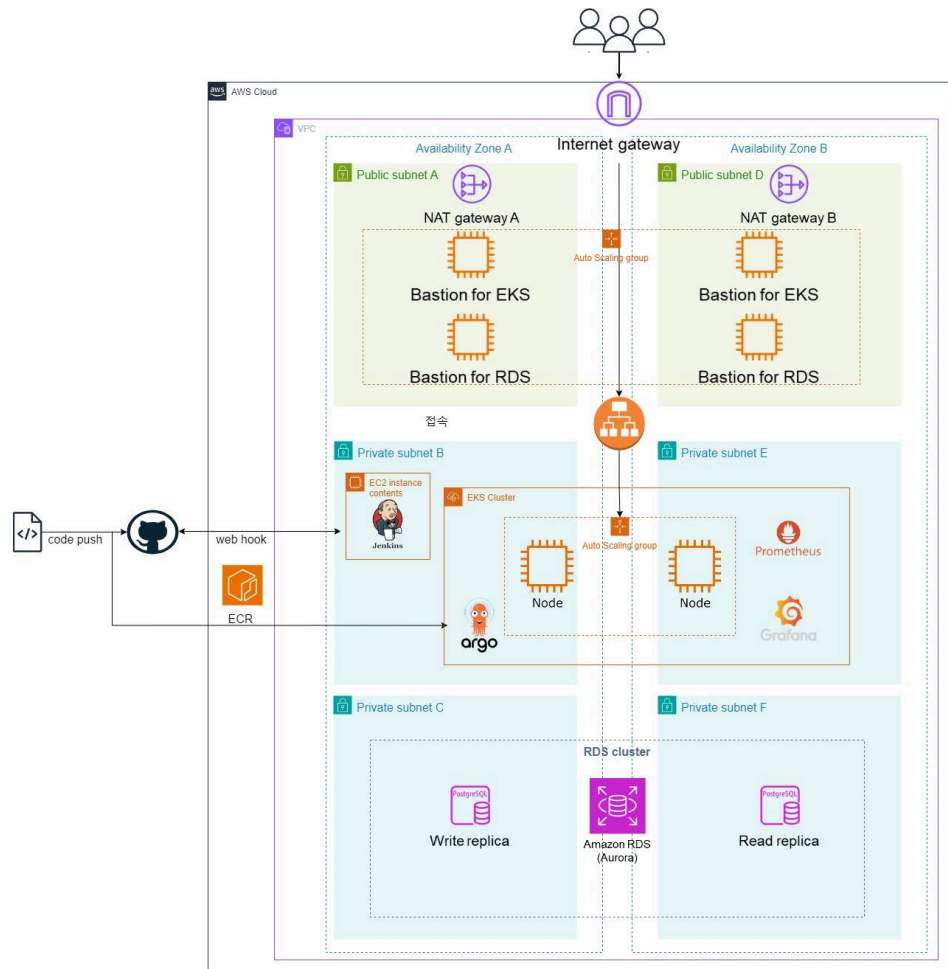
7. Wordpress

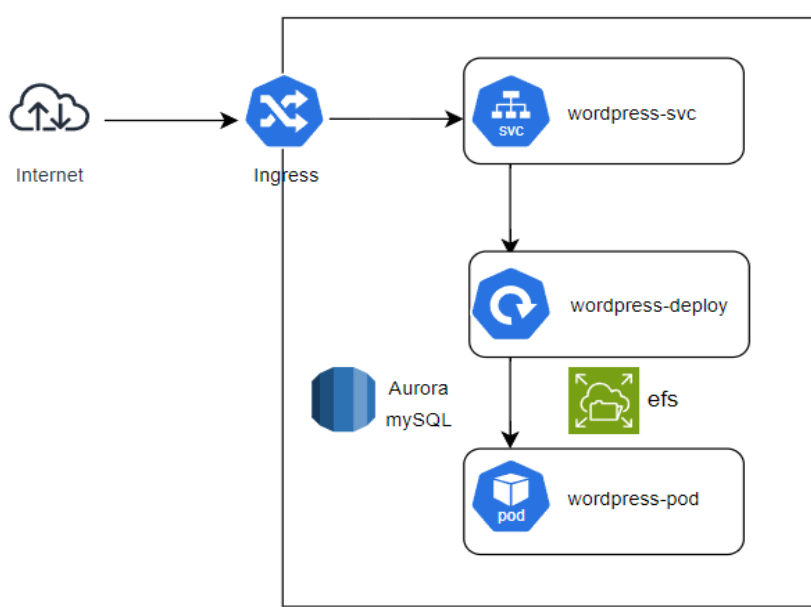
- a. LoadBalancer DNS로 웹에서 접근 (운영 효율성)
- b. Deployment 단계에서 DB 연결
- c. EFS 볼륨으로 content 파일 공유 (백업)

Amazon Linux 2를 기반으로 Kubernetes 환경을 구성하기 위해 EKS를 설치하고 클러스터 내부의 효율적인 관리를 위해 명령 도구로 Kubectl을 사용하였습니다. 클러스터를 구성함으로써 확장성을 높여 특정 청약 공고로 인해 유저 트래픽이 몰릴 때에도 성능 저하없이 원활한 환경에서 이용가능하도록 구현하였습니다.

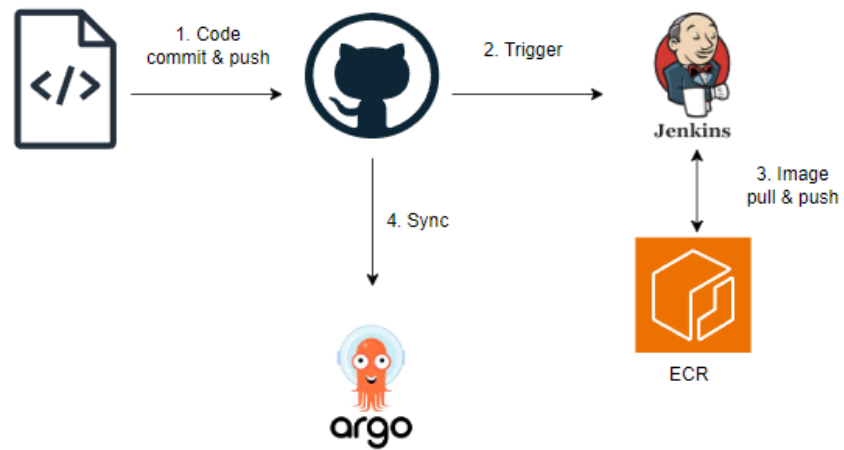
또한 CI & CD 파이프라인을 구성해 웹의 업데이트가 있을 때 딜레이 타임을 줄이고자 자동으로 빌드하고 업데이트가 가능하도록 구성하였습니다.

Architecture



<p>Architecture</p>	<p>프로젝트의 Architecture이며 Multi AZ를 통해 하나의 AZ가 무너지더라도 다른 AZ에서 정상적으로 작동하게 AutoScaling 기능을 추가해 구현하였습니다.</p> <p>Bastion for EKS는 aws 자격 인증 설정을 한 EKS로 접근 및 작업이 가능하도록하였습니다. Bastion for RDS는 DataBase에 접속 및 작업하기 위한 Bastion으로 작업 내용과 권한에 따라 구분을 주어 보안성을 높였습니다.</p> <p>EKS 클러스터를 Private Subnet에 구성해 외부의 접근을 막았습니다. EKS 클러스터를 사용해 자동으로 Scale in/out이 가능한 노드를 생성함으로써 확장성을 보장하였습니다.Jenkins Server는 빠른 빌드를 위한 고가용성 보다 파이프라인 구성을 위한 Docker를 원활히 사용하기 위해 EC2에 구성하였습니다. Node 내에 CD를 위한 arcoCD와 모니터링을 위한 Prometheus와 Grafana를 설치하였습니다.</p> <p>CI & CD 파이프라인에서 Github의 code를 수정하여 Jenkins에 Webhook을 설정해 이미지를 Amazon ECR에 빌드하고 ArgoCD는 ECR의 이미지와 Github의 code를 바탕으로 현재 상태와 비교해 Sync하게 됩니다.</p> <p>Aurora mySQL 또한 Multi AZ에 구성하여 고가용성을 보장하였고 wordpress에 연결하여 사용자 데이터와 게시글 데이터 등을 안전하게 보관하였습니다.</p>
<p>Website</p>	 <pre> graph LR Internet[Internet] --> Ingress[Ingress] Ingress --> wordpress-svc[wordpress-svc] wordpress-svc --> wordpress-deploy[wordpress-deploy] wordpress-deploy --> wordpress-pod[wordpress-pod] wordpress-pod --> Aurora[Aurora mySQL] wordpress-pod --> efs[efs] </pre> <p>Ingress는 AWS LoadBalancer Controller를 사용하였으며 wordpress-deployment를 위한 svc를 구성하였습니다. Deploy에 efs를 연결해 wordpress의 content 내용을 저장하고 rds에 사용자와 게시글 데이터 등을 저장할 수 있게 연동하였습니다.</p>

CI CD Pipeline



저장소 **Code repository**로 Github을 사용하여 팀원들의 협업성과 버전 관리의 용이성을 높였으며, CI tool로 Jenkins, CD tool로 argoCD, 그리고 이미지 저장소로 Amazon ECR을 사용하였습니다.

CD & CD Pipeline workflow

- Github code push
- Trigger 작동으로 Jenkins에서 이미지 빌드
- Amazon ECR 이미지 최신화
- 현재 상태와 Github 상태를 비교해 argoCD에서 Sync하여 변경내용으로 배포