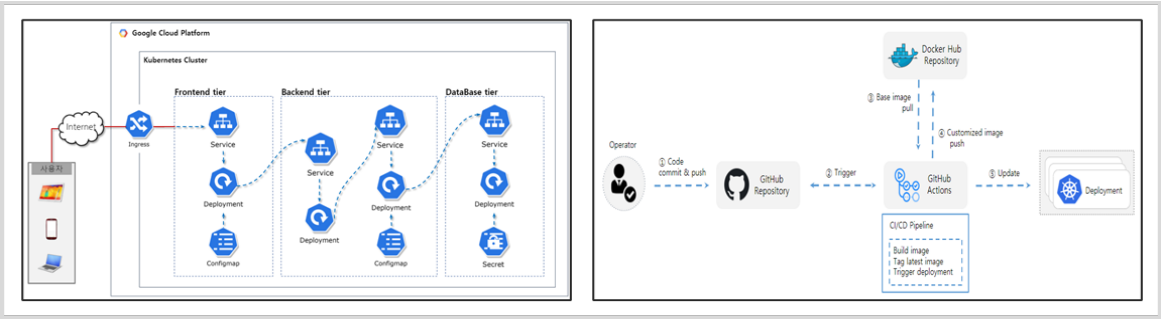
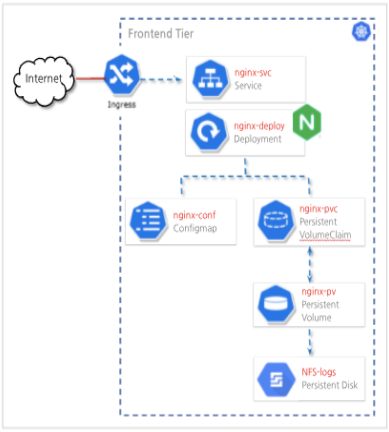
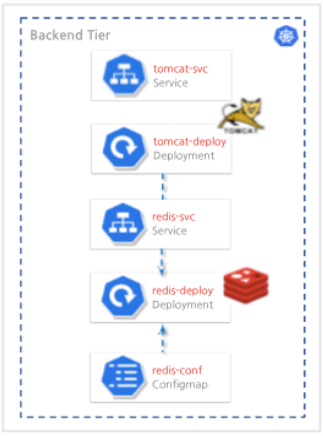
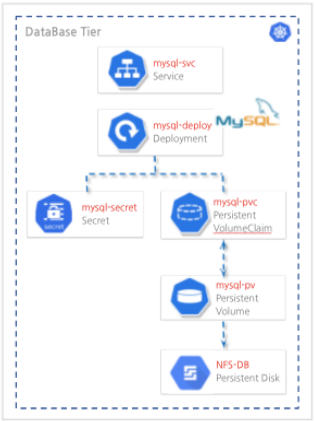
포트폴리오

프로젝트명	GKE를 이용한 3-Tier 연동 및 CI/CD 구성	프로젝트 기간	2021.08.09 – 2021.10.29
프로젝트 개요	Full Container 3-Tier architect의 MSA 환경 제공		
구축 환경	GCP(Google Cloud Platform) 및 Github action		
주요 사용 기술	GKE(Google Kubernetes Engine), GCP HTTP(S) LB, Managed Certificate, CloudSQL, Cloud Armor, Memory Store for Redis, Google Repository, Github action		
고려사항	보안 및 백업, 모니터링, 운영의 효율성 및 빠른 확장성, Cost effective		
프로젝트 내용	 ✓ GCP에서 제공하는 managed Kubernetes 서비스인 GKE를 이용하여 클러스터 환경 구성 ✓ Web-Was-DB가 연동되도록 3-Tier 구축 **[핵심 구현 기술 (고려사항)]** **1. Kubernetes Service** a. SVC, Deployment, Secret, configMap, PV/PVC b. HPA, 노드 오토스케일링(GKE 제공 기술) **2. Google Cloud Native Service** a. HTTP(S) 통신: SSL 인증(GCP SSL certificate) (보안) b. DDos 방어 및 WAF 기능: Cloud Armor (보안) c. Log 백업: 오프젝트 스토리지(cloud storage), Lifecycle management를 이용해서 Archive의 6개월 저장 후 삭제 (보안, Cost effective) d. Ingress Controller로 HTTP(S) 부하분산 (운영의 효율성) e. CloudSQL (운영의 효율성) f. Memory Store for Redis 활용 (운영의 효율성) **3. Github** a. Code repository로 Github 이용 b. Github action 쿠버네티스 환경으로 Web-Was-DB로 3tier 구성, HPA와 노드 오토스케일링을 이용하여 자원의 효율적인 사용과 신속한 서비스 확장이 가능하도록 구성하였습니다. 관리의 편의성을 가져가기 위해 GKE, CloudSQL, Redis와 같은 GCP managed 서비스를 활용해서 HTTP(S) 서비스가 가능하도록 SSL Certificate를 사용하였고, 외부 공격으로부터 방어하기 위해 Cloud Armor 서비스를 추가하였습니다. 그리고 Github/Github action으로 CI/CD Pipeline을 구축하여 어플리케이션 버전 업데이트 시 자동으로 배포될 수 있도록 구현하였습니다.		

	3-Tier 상세 구성도 및 설정
Frontend Tier	 **[Ingress]** Ingress controller는 GCP의 HTTP(S) LB를 사용했으며 Static IP, Google Managed Certificate, HTTP(S) redirection 등의 기술을 사용했습니다. **[nginx-svc]** Nginx deployment를 위한 SVC(NodePort) **[nginx-deployment]** 웹 엔진으로 Nginx 사용 **[nginx-configmap]** nginx.conf를 configmap을 이용하여 구현 **[PV/PVC (NFS 사용)]** Nginx access log를 보존하기 위해서 외부볼륨으로 NFS 사용
Backend Tier	 **[tomcat-svc]** Tomcat deployment를 위한 SVC(Cluster IP) **[tomcat-deployment]** Was로 tomcat 사용(war로 배포) **[redis-svc]** Redis deployment를 위한 SVC(Cluster IP) **[redis-deployment]** Tomcat 세션 클러스터용 (GCP는 멀티캐스트 라우팅을 지원하지 않으므로 tomcat 자체 클러스터를 구현할 수 없어 redis를 사용하여 세션 서버 구축) **[redis-configmap]** redis.conf를 configmap을 이용하여 구현
DataBase Tier	 **[mysql svc]** Mysql deployment를 위한 SVC(Cluster IP) **[mysql-deployment]** DB로 Mysql 컨테이너 활용(개발 환경) **[mysql-secret]** Mysql 접속을 위한 사용자, 비밀번호 설정 **[PV/PVC (NFS 사용)]** DB를 보존하기 위해서 외부볼륨으로 NFS 사용

CI/CD Pipeline	**[Github/Github action]** 저장소로서 Code repository로 Github를 사용하였으며, Github에서 제공하는 action을 이용하여 CI/CD Pipeline을 구현하였습니다. **[Github workflow]** a. Github code push b. Action에서 gradle build c. build 결과인 war파일 artifact 저장 d. artifact 다운로드 e. 컨테이너 이미지 build f. Docker repository로 push g. GKE에 배포