2023 후기 졸업과제 착수보고서
지도교수 김태운

소프트웨어 개발자를 위한 통합 배포 관리 플랫폼

Integrated Deployment Management Platform for Software Developer

2024년 02월 19일

부산대학교 정보컴퓨터공학부/전기컴퓨터공학부
척척학사 팀(4번/분과C)
201824413 구성현
201812118 김찬호
201724551 임주은

차 례

간추린 글

현대 소프트웨어 개발에서는 빠르게 변화하는 요구에 대응하기 위해 CI/CD가 필수적이며, 이에 개발자들은 운영 지식도 필요로 합니다. 그러나 인프라 관련 지식은 개발자의 부담이 될 수 있습니다. 따라서 CI/CD 파이프라인을 추상화하여 개발자들이 플랫폼 형태로 이용할 수 있도록 하는 것이 목표입니다.

개발자들의 부담을 줄이고 개발 효율을 높여, 아이디어를 빠르게 구현하고 테스트 및 배포할 수 있는 환경을 제공합니다. 또한, 표준화된 개발 프로세스와 프로젝트 관리를 통해 생산성을 향상시키고 유지보수를 용이하게 합니다.

사용자 인증, 프로젝트 생성, 지속적인 통합 및 배포, 롤백 등의 요구 사항을 분석하고, 롤백 시 정보 유실, 클라우드 자원 비용 등의 제약을 대응하기 위한 대책을 마련합니다.

React를 사용한 UI, Nginx, Spring 등의 기술 스택과 시스템 구성을 설명하고, 각 요소의 역할과 기능을 소개합니다.

1. 과제 배경 및 목표

1.1. 과제 배경

현대의 소프트웨어 개발 환경에서는 빠르게 변화하는 요구사항에 대응하기 위해 지속적인 통합 및 배포(CI/CD)가 필수적이다. 이를 위해서는 CI/CD 파이프라인을 관리하고 운영해야 한다. 이 과정에서 개발자와 운영팀의 협업은 필연적이며 이를 위해 개발자들에게 개발 뿐만 아니라 운영에 대한 기본 지식을 보유하도록 요구된다.

최근 인프라 관련 도구들은 빠른 속도로 다양화 되고 변화해가고 있고 그에 따른 인지 부하가 증가하고 있다. 그런 이유로 개발자가 인프라를 관리 및 설정하는 것에 대해 학습하도록 하는 것은 효율이 좋지 않고, 장기적으로 이것이 개발자의 부담을 유발하고 생산성을 떨어뜨린다.

1.2. 과제 목표

개발자들이 배포 과정에서 겪는 부담을 최소화하면서 빠르고 안정적으로 소프트웨어를 배포할 수 있도록 CI/CD 파이프라인을 구축하고 운영하는 기능을 추상화 하여 플랫폼 형태로 개발자에게 제공한다. 개발자가 UI형태로 확인하고 활용할 수 있도록 제공함으로써 개발 효율을 높인다.

배포 전반 과정의 작업 방식을 표준화된 형식으로 제공함으로써 일관된 형태로 인프라를 관리한다. 또한 배포된 프로젝트에 대한 오너십을 관리하여 추후에 배포된 코드에 대한 특이사항이 생겼을 때, 빠르게 대응할 수 있도록 프로젝트 관련 정보를 유지보수한다.

1.3. 기대 효과

운영에 대한 지식을 습득하고 유지하는 과정에서 발생하는 인지적 부담을 줄여, 개발자들은 자신의 역할에 더 집중할 수 있도록 돕는다면 개발 프로세스의 효율성을 높일 수 있다. 그리고 개발자들은 아이디어를 신속하게 구현하고 테스트 및 배포할 수 있는 환경을 제공받을 수 있다. 이를 통해 개발자들의 인지부하를 낮추고, 개발자가 맡은 역할에 집중할 수 있도록 하여 작업 효율을 높일 수 있도록 한다.

또한, 내부 개발자 플랫폼을 통해 개발 프로세스의 표준화와 자동화가 가능해진다. 이는 개발자들이 일관된 방식으로 코드를 작성하고, 빌드 및 배포하도록 유도하며, 운영 측면에서 발생할 수 있는 중복 작업을 최소화할 수 있다. 이로써 프로젝트의 생산성을 향상시키고 개발 프로세스의 일관성을 보장하여 유지보수를 용이하게 한다.

마지막으로, 배포된 프로젝트에 대한 오너십을 관리하고 프로젝트 관련 정보를 관리함으로써 향후 문제 발생 시 신속한 대응이 가능해진다. 이는 시스템의 안정성을 유지하고 사용자들에게 신뢰감을 제공하는 데 도움이 될 것으로 기대된다.

2. 요구 조건 분석과 제약 사항

2.1. 요구 조건 분석

1. 사용자 인증
    - 로그인 시 GitHub 인증 및 OAuth 2.0 승인 프레임워크를 사용하여 인증된 사용자로써 권한을 얻는다.
    - 데이터베이스에 사용자의 데이터를 저장하고 관리한다.
2. 프로젝트 생성
    - 특정 GitHub 레포지토리를 선택하고 프로젝트로 생성할 수 있다.
3. 지속적인 통합
    - 대상 프로젝트의 소스코드가 변경되면 테스트 및 빌드를 수행한다.
    - 결과물을 바탕으로 컨테이너 이미지가 생성된다.
4. 지속적인 배포
    - 이미지가 성공적으로 생성되면 사용자는 UI를 통해 해당 컨테이너 이미지를 쿠버네티스에 배포한다.
    - 배포 성공시 접속가능한 IP 주소 혹은 URL을 제공받는다.
5. 롤백
    - 버전 ID를 제공하면 해당하는 버전으로 배포 메니페스트를 롤백한다.
6. 서비스 제공
    - 웹 어플리케이션 형태로 서비스를 제공하여 편의성과 접근성을 높인다.
    - 별도의 환경설정, 시크릿 관리, 비용관리 등 부차적인 과정을 추상화하여 사용자에게 편의를 제공한다.

2.2. 현실적 제약 사항 분석 결과 및 대책

2.2.1 제약 사항

1. 롤백 시 정보 유실
   특정 버전으로 롤백을 수행했을 때 예기치 않은 정보 유실이 발생할 수 있다.
2. 퍼블릭 클라우드 자원 활용
   AWS에서 컴퓨팅 자원을 할당받아 사용하는 과정에서 발생하는 서버 비용이 상당할 수 있다.
3. 대량의 동시 요청
   사용자들이 동시에 많은 요청을 보낼 경우 서버의 부하가 증가할 수 있다.

2.2.2 대책

1. S3 저장소 활용
   안정적인 서비스 제공을 위해 롤백 기능 수행 시 이미지 및 관련 데이터를 저장하는 데에는 비용이 적게 소요되는 AWS S3와 같은 서비스를 활용하여 백업한다.
2. 비용 절약
   클라우드 서비스에서 제공하는 크레딧 혹은 프리티어 자원을 적극 활용한다.
3. 모니터링 및 오토 스케일링
   자원 사용 현황을 모니터링하고, 사용자의 트래픽이나 부하에 따라 클라우드 자원을 자동으로 확장 또는 축소하는 스케일링 기능을 활용하여 필요한 자원 만큼만 할당함으로서 비용을 절감하고 안정적으로 서비스를 제공한다.

3. 설계 문서

3.1. 기술 스택 및 시스템 구성

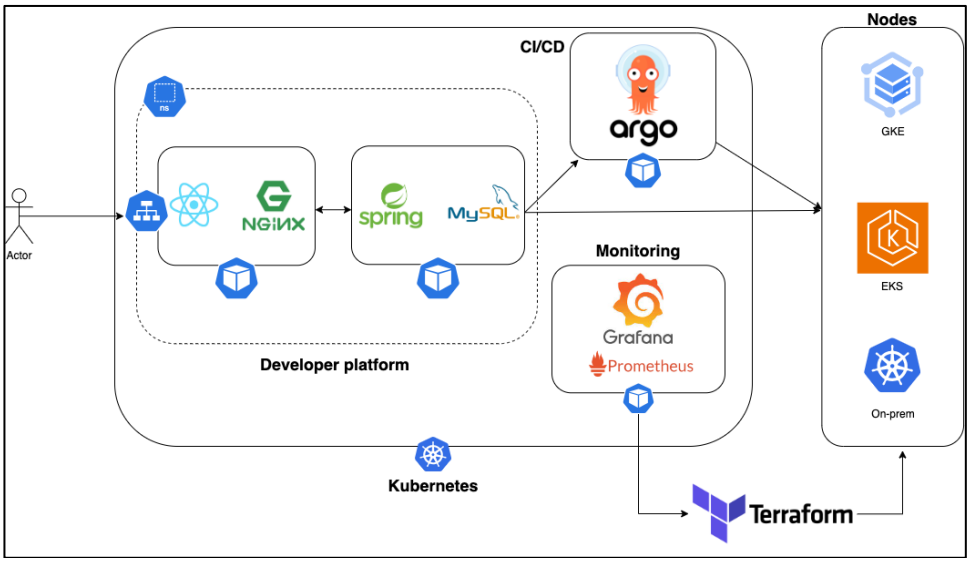

그림 1<시스템 구성>

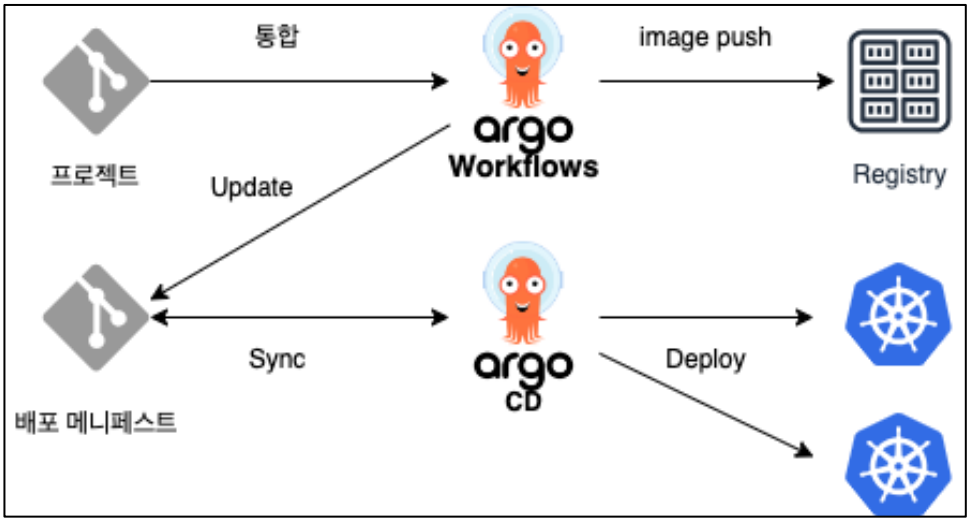

그림 2 <CI/CD파이프라인>

요소	내용
UI	React 프레임워크를 사용하여 사용자가 보는 웹의 로그인 화면, 프로젝트 관리 페이지을 만든다. 로그인 기능과 프로젝트 생성, 프로젝트 상세화면, 배포 로그와 롤백 기능을 이용할 수 있다.
Nginx	웹 서버로서 요청을 받아 백엔드 서버로 전달한다.
Backend	Java 프레임워크인 Spring을 사용해 로그인, 회원가입, 가상 머신 생성 등의 사용자 요청을 처리한다.
Argo Workflows	컨테이너 네이티브 워크플로우 엔진으로 지속적인 통합(CI)을 수행한다.
Argo CD	지속적인 배포(CD)를 수행하고, 사용자 요청으로 필요 시 버전을 롤백한다.
클라우드 플랫폼	AWS 혹은 GCP를 사용해서 클라우드 환경을 구축한다.
데이터베이스	MySQL을 사용해서 데이터베이스를 구축하고 사용자 정보 및 배포와 관련된 메타데이터와 로그를 저장한다.
GitHub Container Registry	컨테이너 이미지 저장소로 배포할 이미지를 저장하고 관리한다.
Object Storage	롤백 기능 수행 시 현재 배포된 이미지 및 관련 데이터를 Object Storage에 백업한다.
Prometheus, Grafrana	모니터링 도구로 클러스터의 상태 및 성능 지표 수집 및 시각화를 한다.
테스트	JUnit을 활용하여 단위 테스트 수행한다.
형상 관리	Git과 GitHub을 사용해 효율적으로 프로젝트를 관리한다.

표1 <기술 스택>

3.2. 사용 기술

1. GitHub OAuth 인증

GitHub OAuth는 GitHub에서 제공하는 인증 방식 중 하나로, 외부 애플리케이션이 GitHub 사용자의 인증을 처리할 수 있도록 해주는 프로토콜이다. 사용자가 웹 서비스에 로그인할 때 GitHub 인증을 통해 자신의 계정을 인증하고 권한을 부여할 수 있다.

GitHub OAuth 인증은 OAuth 2.0 프로토콜을 기반으로 합니다. 사용자가 자신의 계정 정보를 외부 애플리케이션과 공유할 때 발생할 수 있는 보안 문제를 해결합니다.

이는 사용자가 웹 서비스에 로그인한 후, 권한 부여 과정을 통해 GitHub API를 통해 자신의 레포지토리에 접근할 수 있음을 의미합니다. 이를 통해 사용자는 손쉽게 자신의 프로젝트 소스코드를 웹 서비스로 가져와서 관리할 수 있습니다.

2. 컨테이너 이미지 빌드

이미지 빌드 기술은 사용자의 소스 코드를 기반으로 컨테이너 이미지를 생성하는 과정을 의미합니다. 이는 사용자가 웹 서비스에 로그인한 후, 자신의 GitHub 레포지토리에 있는 소스 코드를 가져와서 컨테이너 이미지로 빌드하는 과정을 수행합니다. 이미지 빌드는 Dockerfile이라는 설정 파일을 사용하여 이미지를 계층을 정의합니다.

이러한 도커 빌드 기술을 통해 사용자는 손쉽게 자신의 프로젝트 소스 코드를 기반으로 도커 이미지를 생성할 수 있습니다. 이를 통해 프로젝트 환경을 쉽게 구축하고 배포할 수 있으며, 다양한 환경에서의 실행을 보장할 수 있습니다.

3. 컨테이너 오케스트레이션

컨테이너 오케스트레이션 기술은 여러 개의 도커 컨테이너를 자동으로 배포, 관리, 확장하는 기술을 의미합니다. 이는 사용자가 생성한 도커 이미지를 자동으로 배포하고, 필요에 따라 컨테이너를 확장하거나 축소하여 서비스의 가용성과 확장성을 유지하는 역할을 합니다. 컨테이너 오케스트레이션 툴로는 Kubernetes(쿠버네티스)를 사용합니다.

4. 개발 일정 및 역할 분담

4.1. 개발 일정

구분	작업 일정													
	3월				4월					5월				6월
요구사항 분석														
인프라 구축														
REST 정의 및 DB 설계														
Github 인증 구현														
프론트엔드 구현														
CI/CD 파이프라인 설정														
중간 보고서														
API연동														
모니터링 설정 및 연동														
테스트 및 디버깅														
배포 테스트														
최종 배포														
최종 보고서														
발표 준비														

표2 <개발 일정>

4.2. 역할 분담

이름	분류	세부 역할 분담
구성현	인프라	● Kubernetes 클러스터 설정 및 관리 ● 모니터링 구축 ● CI/CD 파이프라인 설정 ● Infra as Code 유지보수
김찬호	프론트엔드	● 웹페이지 구현 ■ React 활용 ● 사용자 경험 설계 및 화면 디자인
임주은	백엔드	● 사용자 인증 모듈 구현 ● DB 모델링 ■ 테이블 설계

표3 <역할 분담>

그림 및 표 차례

그림 차례

표 차례