**Barrier Free**

**삼성SW청년아카데미 서울캠퍼스 6기**

**Gitlab 소스 클론 이후 빌드 및 배포 과정 정리**

**A504팀 Weclusive**

**유민상, 남정현, 정현정, 허은아, 황수진**

**목 차**

1. **사용한 기술 스택**
2. **빌드 과정**
3. **배포 과정**
4. **MySQL DB 접속**

**1) 사용한 기술 스택**

**1) 이슈 관리** : Jira

**2) 형상 관리** : Gitlab

**3) 빌드/배포 관리** : Jenkins

**4) 커뮤니케이션** : Mattermost, Notion, Discord

**5) 개발 환경**

**1. 운영체제** : Windows10

**2. IDE**

가) STS 3.9.14 RELEASE

나) Visual Studio Code 1.64.2

다) UI/UX : Figma

**3. 데이터베이스** : MySQL Workbench 8.0.27

**4. 서버** : AWS EC2

가) Ubuntu 20.04 LTS

나) Docker 20.10.7

다) Nginx 1.21.6

**6) 세부 사항**

**1. Backend**

가) Java (Open-JDK zulu 1.8)

나) Spring Boot 2.5.6

다) Maven 3.8.4

라) JPA 2.5.6

마) lombok 1.18.22

바) Swagger2 2.9.2

사) JWT 0.9.1

아) OAuth2, Spring Security 2.5.6

자) JDBC 2.5.6

**2. Frontend**

가) React 17.0.2

나) Node.js 16.13.2

다) npm 8.1.2

라) react-redux 7.2.6

마) redux 4.1.2

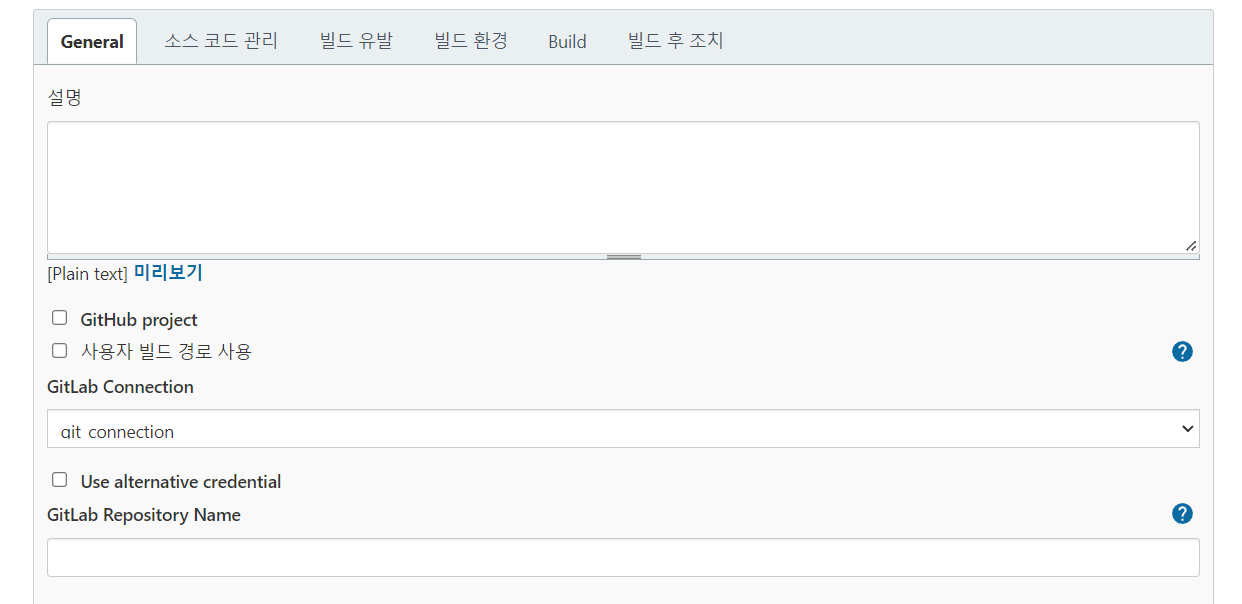
바) react-route 6.2.1,

사) react-router-dom 6.2.1

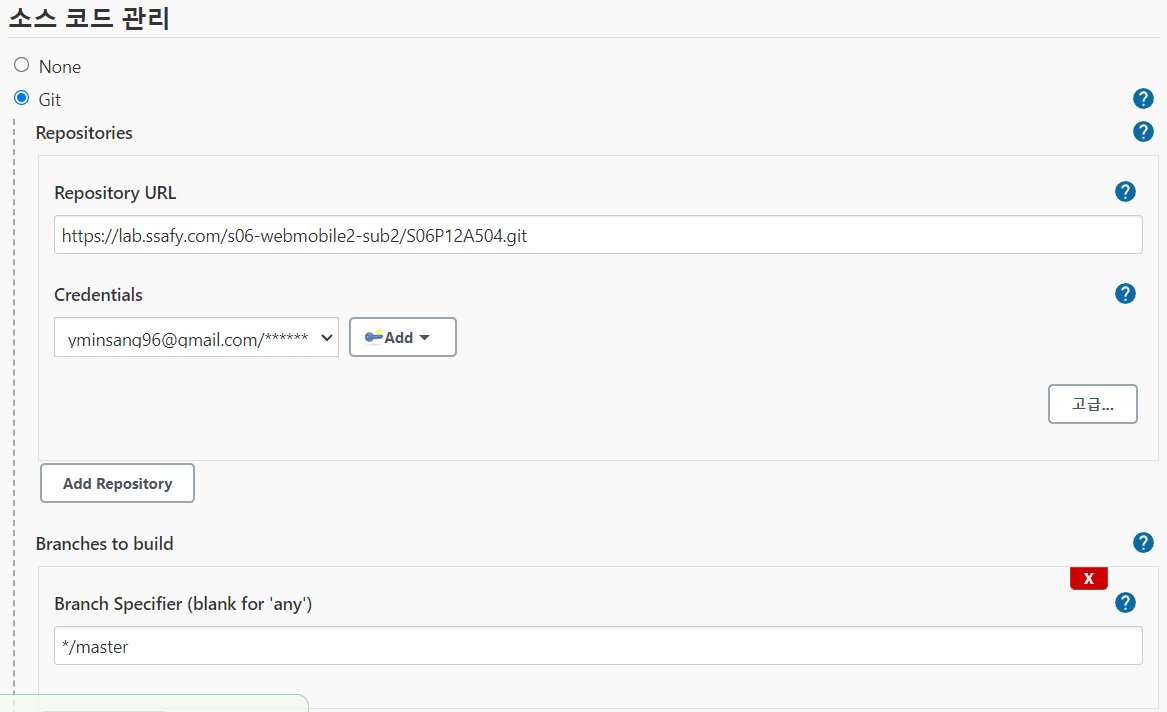
**2) 빌드 과정**

Gitlab과 연동된 Jenkins의 관리페이지로 접속합니다. (<https://i6a504.p.ssafy.io:9090>)

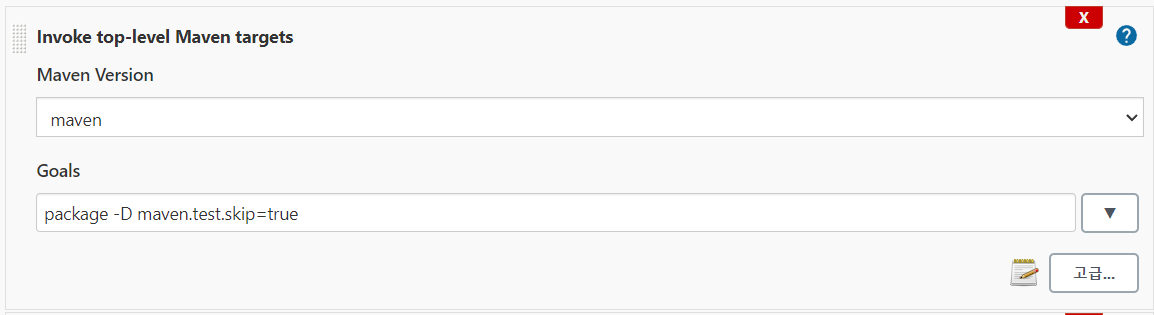
해당 관리페이지에서 barrierfree 아이템에서 구성 탭으로 진입한 뒤



먼저 Gitlab과 연동된 GitLab Connection을 선택하고

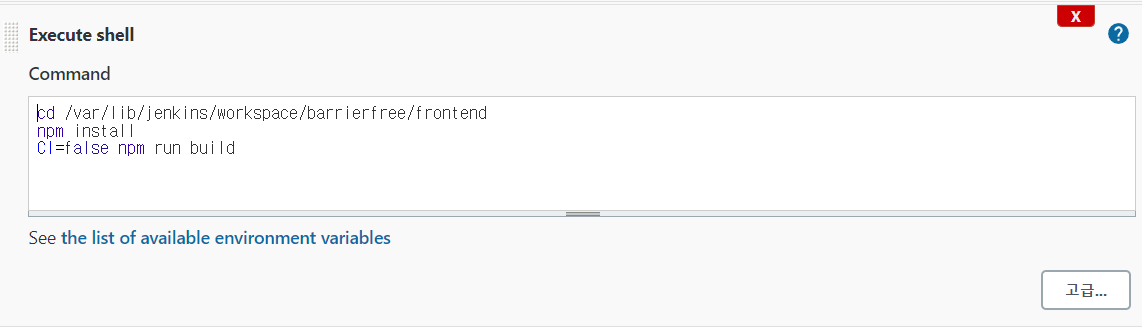


GitLab과 연동할 때 필요한 Repository URL과 Credentials를 입력하고 master 브랜치와 연결해줍니다.



이후 Build탭에서 Invoke top-level Maven targets를 추가해줍니다. 이 부분을 통해

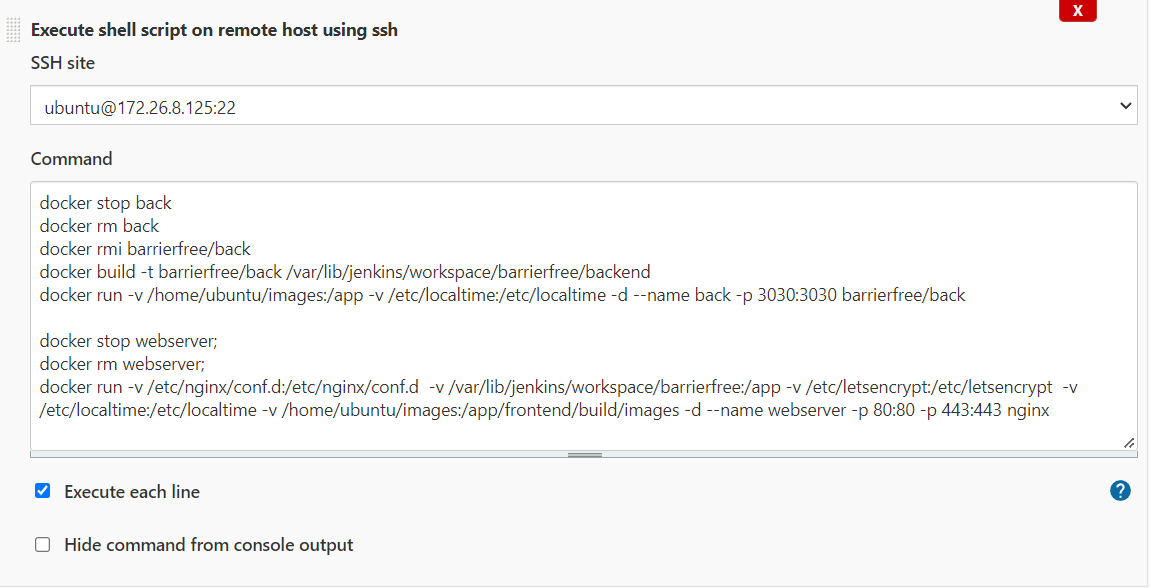
**package –D maven.test.skip=true** 로 Spring Boot 프로젝트를 빌드합니다.



아래에 Execute Shell 을 추가하고, EC2 쉘에서 실행될 명령어를 차례대로 적어줍니다.

첫번째 줄은 Jenkins가 받아온 Gitlab의 코드에서 프론트엔드 프로젝트가 있는 경로로 이동하는 것을 의미하고, 두번째 줄은 해당 위치에서 npm install 명령어를 실행해 필요한 node\_modules를 설치하는 과정입니다. 마지막 세번째 줄은 현재 경로의 React 프로젝트를 build하는 부분입니다. 이때 WARNING이 발생할 경우 CI환경에서 빌드가 안되도록 막는 라이브러리들이 존재하면 빌드가 실패하는데, 이 경우에는 앞에 CI=false를 적어주면 해결할 수 있습니다.

**3) 배포 과정**



배포는 Execute shell script on remote host using ssh로 SSH키 인증을 통해 쉘 스크립트를 실행해 진행해 보았습니다. 먼저 기존에 실행중인 백엔드 도커 컨테이너를 멈춘 뒤 제거하고, 해당 도커 이미지도 제거한 후에 백엔드 프로젝트 위치에서 도커 이미지를 빌드합니다.

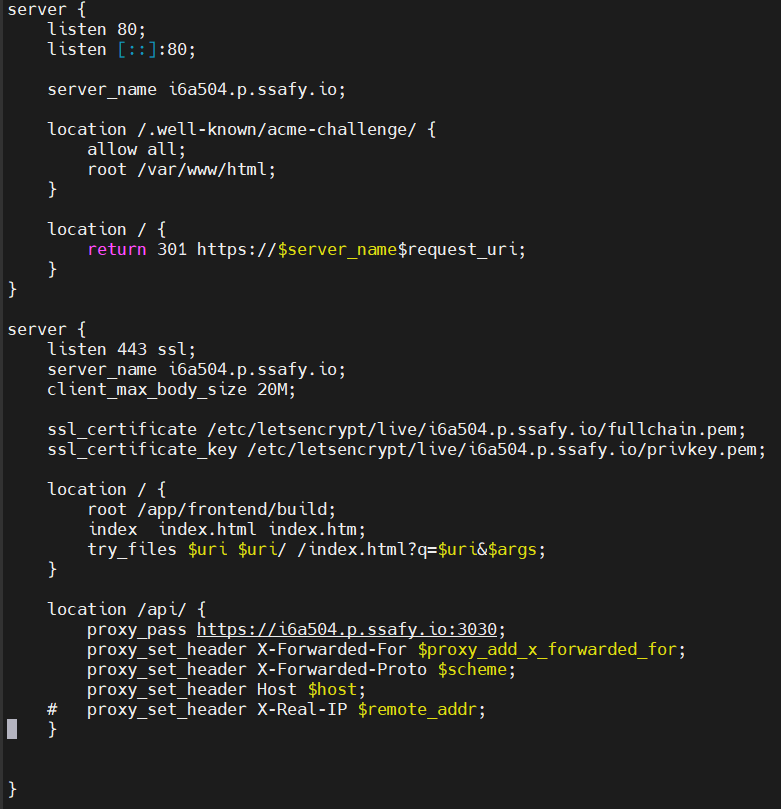
백엔드 프로젝트의 Dockerfile은 다음과 같습니다.



이후 barrierfree/back 이라는 이름의 도커 이미지로 도커 컨테이너를 실행합니다. 이때 이미지 파일 정보를 담은 /home/ubuntu/images디렉토리와 도커 컨테이너 타임존 설정을 위한 /etc/localtime 디렉토리를 도커 볼륨으로 설정해 마운트하고, 3030포트까지 설정해줍니다.

다음은 nginx 컨테이너입니다. 실행중인 도커 컨테이너를 멈춘 뒤 제거하고, EC2 서버에 미리 작성해놓은 nginx 설정파일이 있는 /etc/nginx/conf.d 디렉토리와 빌드한 프로젝트가 있는 디렉토리, HTTPS 설정을 위한 SSL키 값들이 있는 /etc/letsencrypt 디렉토리, 도커 컨테이너 타임존 설정을 위한 /etc/localtime 디렉토리, 이미지 파일이 있는 /home/ubuntu/images 디렉토리를 모두 도커 볼륨으로 지정해 마운트 해주고, http기본 포트인 80포트와 https 기본 포트의 443 포트로 설정한 뒤 실행 시켜 줍니다.

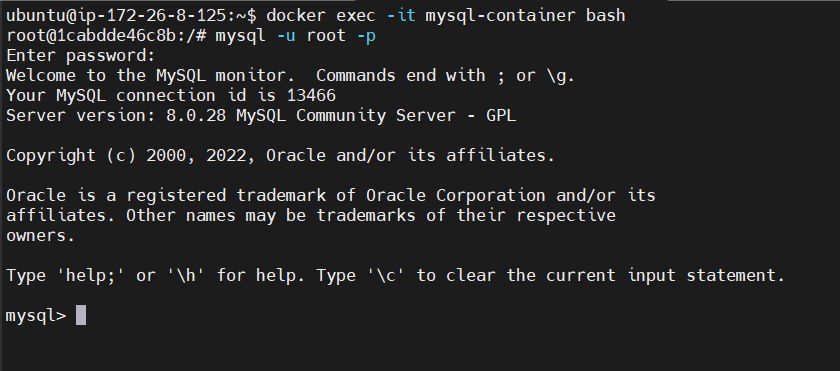
이 때 nginx 설정파일인 nginx.conf의 내용은 다음과 같습니다.



먼저 80포트로 들어올 경우(http로 접속할 경우) https로 반환해줍니다. 그 뒤 443포트로 들어올 경우 (https로 접속할 경우) /etc/letsencrypt에 있는 SSL 키값들을 통해 인증을 진행합니다. 해당 인증절차를 통과하면 React프로젝트가 webpack으로 빌드된 폴더에서 index.html을 찾아 클라이언트에게 표시해줍니다. 만약 도메인주소/api 경로로 접근할 경우 SpringBoot 프로젝트가 실행중인 3030포트로 연결해줍니다.

**4) MySQL DB 접속**

먼저 EC2 서버에 접속한 뒤 mysql이 실행중인 도커 컨테이너로 접속해야 합니다.



**docker exec –it [ mysql이 실행중인 컨테이너명] bash** 명령어로 bash로 접근한 뒤,

**mysql –u root –p** 명령어를 통해 root로 mysql에 접속합니다.

이후 초기 접속 시 설정했던 비밀번호를 입력하면 DB를 사용할 수 있습니다.