1. 버전 정리
   1. 백엔드
      1. JDK

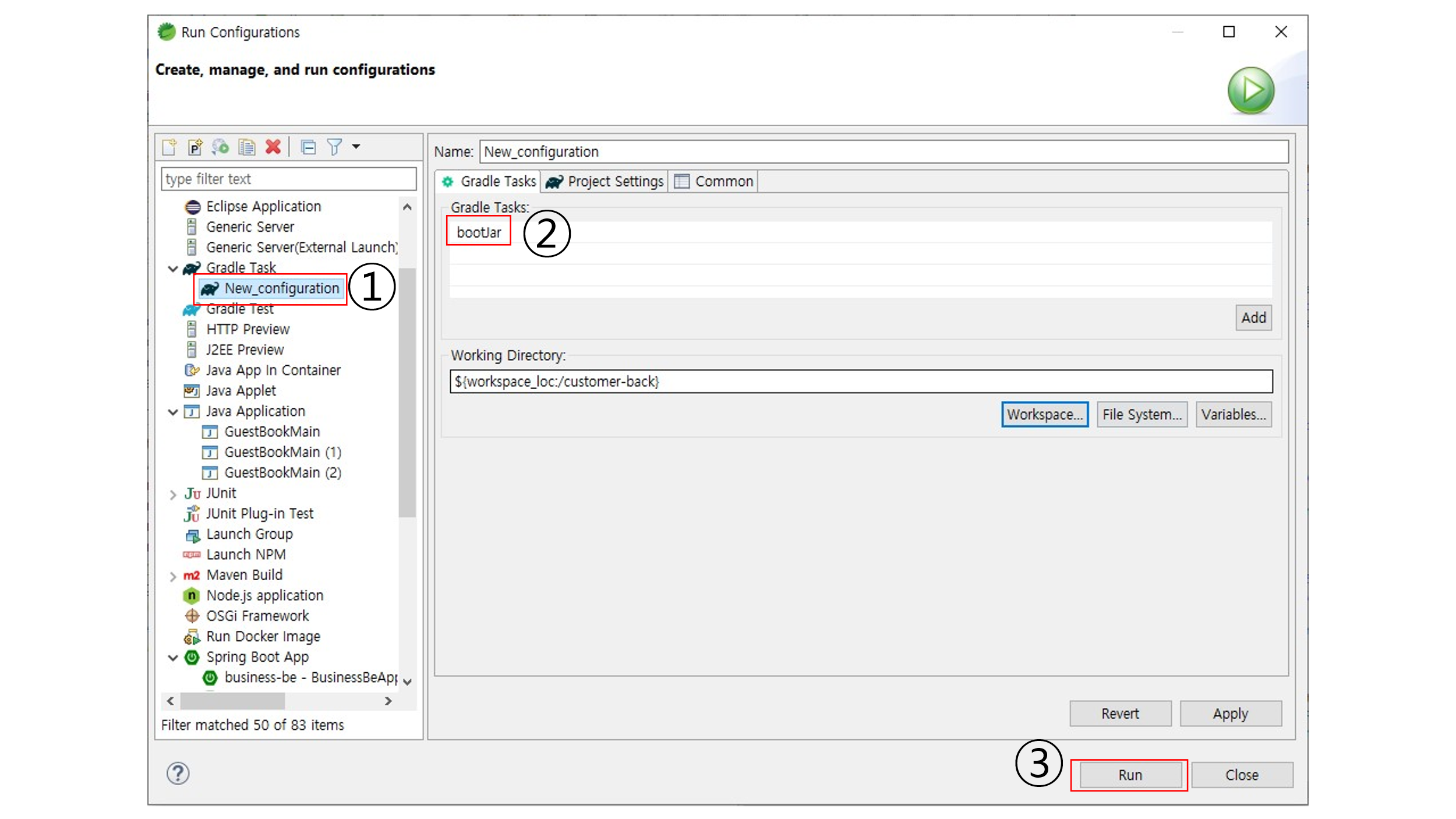
* Openjdk version 1.8.0 352
* OpenJDK Runtime Environment (build 1.8.0\_352-8u352-ga-1~20.04-b08)
* OpenJDK 64-Bit Server VM (build 25.352-b08, mixed mode)
  + 1. SpringBoot
* SpringBoot 2.7.3
* Gradle 7.5.1
  + 1. DB
* 10.9.3-MariaDB-1:10.9.3+maria~ubu2204
  1. 프론트엔드
     1. Node.js
* Node.js 16.17.0 LTS
  1. CI/CD
     1. Aws ec2
* Ubuntu 20.04.5 LTS (GNU/Linux 5.15.0-1019-aws x86\_64)
  + 1. Docker
* Docker version 20.10.21, build baeda1f
  + 1. Jenkins
* Jenkins 2.376
  + 1. Nginx
* Nginx/1.18.0 (Ubuntu)

1. 빌드
   1. 백엔드
      1. Gui 빌드(STS 기준)
2. 프로젝트 우클릭->Run As->Run Configurations

Gradle Task->New\_configuration 클릭한 뒤

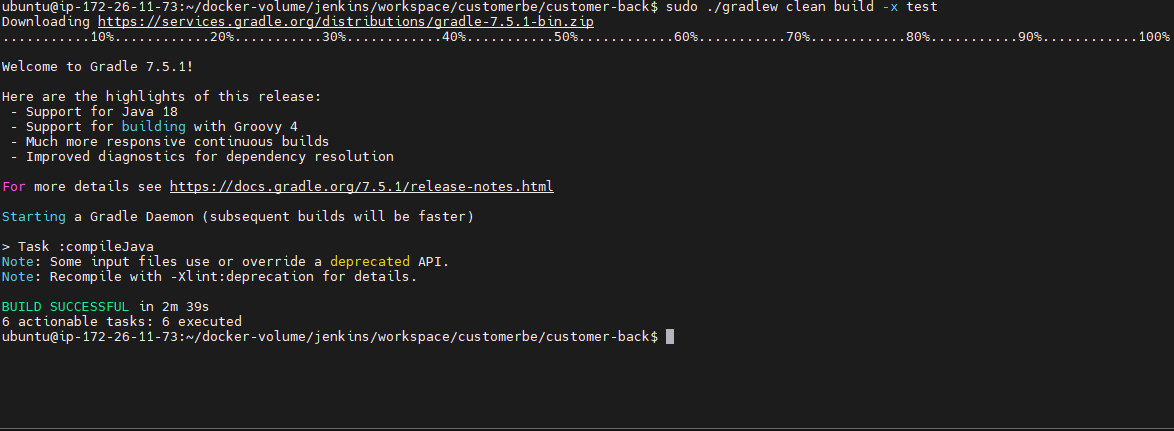
Gradle Tasks에 bootJar 추가해주고

Run 버튼을 클릭합니다. 빌드가 완료되면 해당 프로젝트의 build/libs 폴더에 jar파일이 생성됩니다.

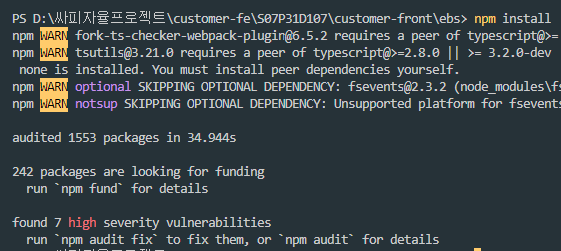


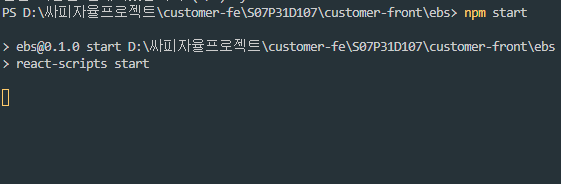
* + 1. 명령어 빌드

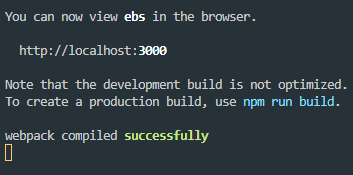
Gradlew파일이 있는 경로로 이동해서 sudo ./gradlew clean build -x test를 입력합니다. 자동적으로 빌드가 진행되고 마찬가지로 해당 프로젝트의 build/libs 폴더에 jar파일이 생성됩니다. 또한 위에서 입력한 것 처럼 clean이 붙게 되면 기존에 만들었던 jar을 지우고 jar을 생성해줍니다.



* 1. 프론트엔드
     1. 빌드



프론트엔드 프로젝트가 있는 경로에 가서, npm install을 해서 필요한 라이브러리를 다운받고, npm start를 통해 실행합니다.



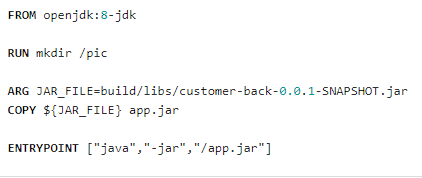
1. 배포 시 특이사항
   1. CI/CD

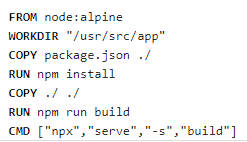
aws ec2에 Mysql을 설치했고, Docker와 Jenkins를 통한 CI/CD 구축 및 cloudflare 을 통한 https설정이 되어있습니다. 또한 추가적인 ec2 서버를 통해서 사용자/점주 각각 다른 서버에서 돌아가는 환경을 구축했습니다. 다만 다른 서버라고 해서 설정이 다르지 않고 거의 같기 때문에 여기서는 사용자 서버에 대한 설치/설정법을 설명하겠습니다.

* + 1. Openjdk 설치
* sudo apt-get update 입력후
* sudo apt-get install openjdk-8-jdk 를 입력해서 openjdk를 설치합니다.
  + 1. Docker 설치
* sudo apt-get install apt-transport-https ca-certificates curl gnupg-agent software-properties-common
* curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add –
* sudo add-apt-repository \ "deb [arch=amd64] https://download.docker.com/linux/ubuntu \ $(lsb\_release -cs) \ stable"
* sudo apt-get update && sudo apt-get install docker-ce docker-ce-cli containerd.io

위의 명령어를 순서대로 실행해서 Docker를 설치합니다.

* Backend의 경우 gitlab의 springboot 프로젝트의 pom.xml이 있는 위치에 Dockerfile을 생성합니다.
* Frontend의 경우 프로젝트의 package.json파일이 있는 위치에 Dockerfile을 생성합니다.
* Dockerfile은 다음과 같이 작성합니다. 위의 것이 백엔드, 아래의 것이 프론트입니다.





* + 1. Nginx 설치
* sudo apt-get install nginx를 입력해서 nginx를 설치합니다.
  + 1. Docker에 mariadb 설치 및 실행
* docker pull mariadb

docker run --name dbimg -d -p 13107:3306 --restart=always -e MYSQL\_ROOT\_PASSWORD=root

docker exec -it mariadb /bin/bash mariadb

mysql -u root -p 를 순서대로 실행합니다. 각 명령어는

mariadb 이미지를 생성하고 실행시킨 뒤에, mariadb 컨테이너의 bash로 진입하고, mariadb를 실행시키는 명령어입니다.

이후

create schema `ebs`;

create user 'd107'@'%' identified by 'gumi107';

grant all privileges on `ebs`.\* to 'd107'@'localhost';

flush privileges;

쿼리를 순서대로 실행해서 권한을 부여합니다.

테이블,뷰가 사용자/점주 db에서 생성된 뒤에 아래의 쿼리를 실행시켜 점주 db에서 사용자 db접근에 필요한 권한을 생성해줍니다. 테이블, 뷰에 관련된 자료는 dbdump 파일을 참조해주세요.

create user 'd107\_business'@'%' identified by 'business107';

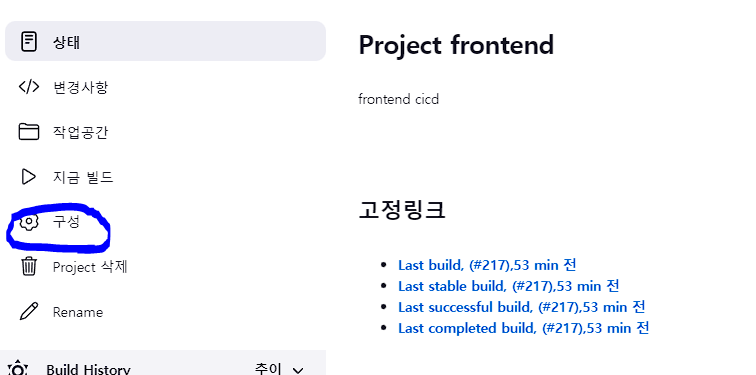
grant select on `ebs`.`v\_reservation` to 'd107\_business'@'%';

flush privileges;

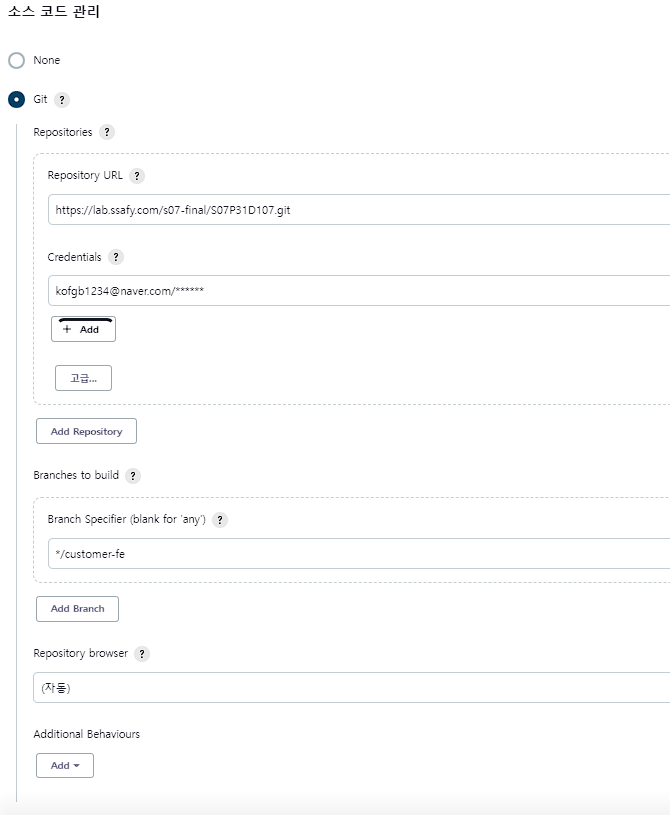
* + 1. Docker에 Jenkins 설치 및 실행
* sudo docker run -d --name jenkins -u root --privileged -p '9090:8080' -v '/home/ubuntu/docker-volume/jenkins:/var/jenkins\_home' -v '/var/run/docker.sock:/var/run/docker.sock' -v '/usr/bin/docker:/usr/bin/docker' jenkins/Jenkins

위의 명령어를 실행해서 Docker에 Jenkins를 설치하고 container를 올립니다.

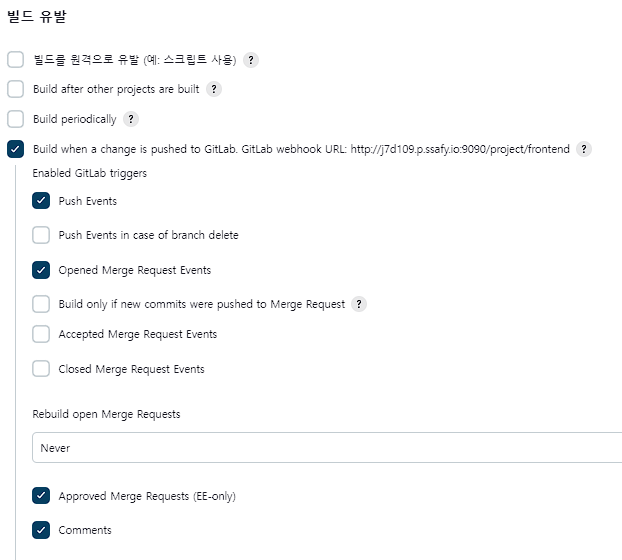
* sudo docker logs Jenkins 를 입력해서 Jenkins 설치에 필요한 패스워드를 가져옵니다.
* 원격서버주소:9090을 입력해서 Jenkins 설치 페이지로 넘어갑니다.
* 계정 설정을 한 뒤에 기본 패키지를 설치합니다
* 이후 설정->플러그인 매니저에서 gitlab에 관련된 플러그인을 설치합니다.
  + 1. Jenkins로 ci/cd 구축
* 젠킨스에 접속한 뒤 새로운 Item을 클릭합니다.
* Freestyle project을 클릭하고 해당 아이템의 이름을 정해주고 OK를 누릅니다.



* 아이템이 만들어지면 해당 아이템을 클릭한 뒤에, 좌측에 있는 구성을 누릅니다.
* 소스코드 관리의 git을 눌러서 repository url과 아이디를 작성해주고, 아래의 branches to build에 branch이름을 넣어서 어떤 브랜치를 빌드 할지 정해줍니다.



* 빌드 유발에서 push나 merge가 되었을 때 젠킨스에서 빌드를 하게끔 아래의 항목을 체크합니다.

- 빌드 유발 아래부분에 있는 고급 탭을 클릭해서 Secret Token을 생성합니다. 이는 gitlab에서 webhook을 설정하는데 필요합니다.

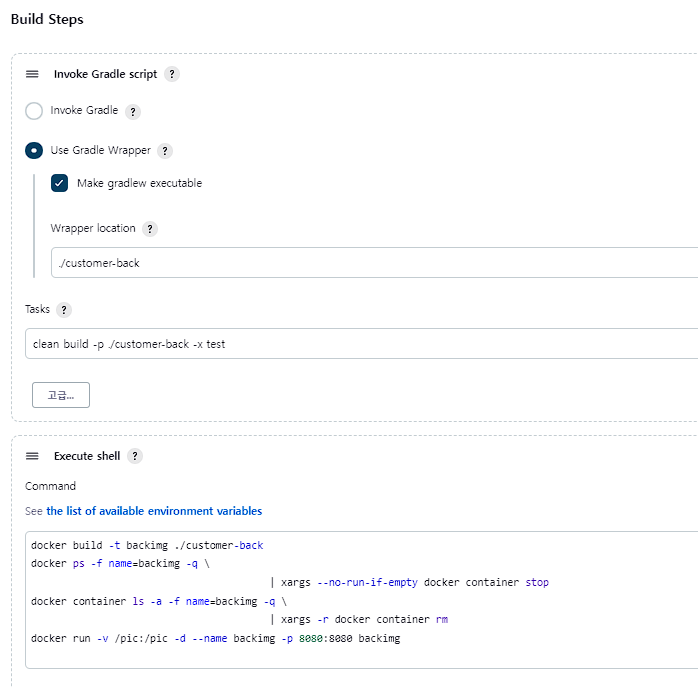


* Build steps에는 docker file을 만들어서 해당 도커 이미지를 돌리는 코드를 작성해주면 됩니다.

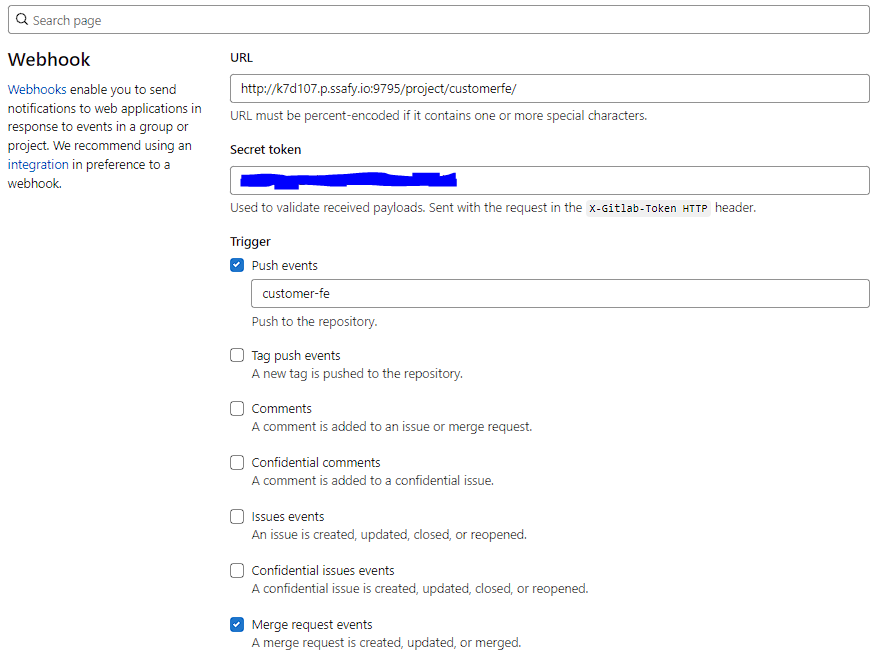


프론트의 경우에는 이렇게 작성하면 끝이지만, 백엔드의 경우에는 Invoke Gradle script을 통해서 dockerfile에 컴파일 관련 설정을 해주지 않아도 됩니다.

그래서 백엔드는 다음과 같이 코드를 작성하면 됩니다.



- 마지막으로 gitlab의 프로젝트 repository의, Settings->Webhooks에 설정을 해줍니다.



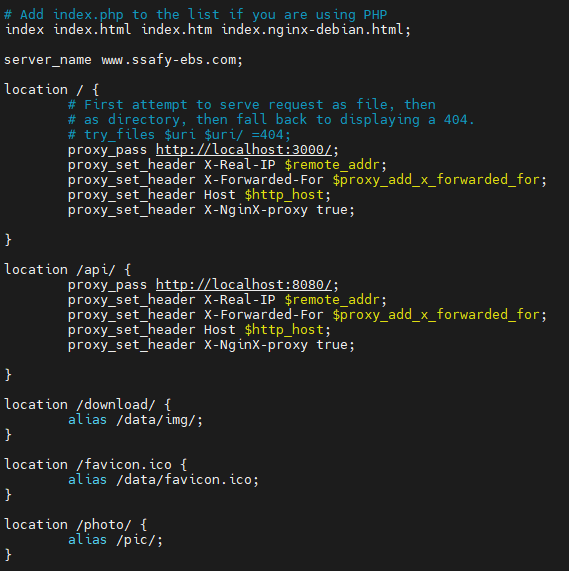
URL에는 jenkins에서 생성한 프로젝트 주소를 입력하되, job을 project으로 고쳐서 기입합니다.

Secret token에는 아까 위에서 생성한 Secret token을 입력합니다.

Trigger에는 Push events와 Merge request events를 체크하고 아래에 해당 트리거가 작동할 브랜치의 이름을 적어주면 ci/cd설정은 끝납니다.

* + 1. ssl 및 nginx 적용

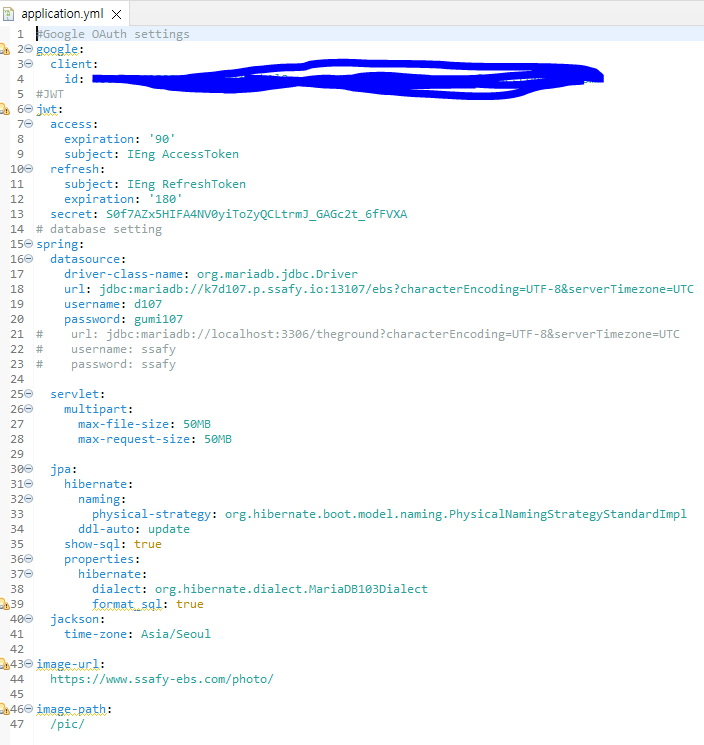
이번 프로젝트에서는 cloudflare를 사용했기 떄문에, cloudflare에서 자동으로 ssl을 적용시켜주었습니다. 그렇기 때문에 nginx의 conf.d만 설정해주면 됩니다.



위와 같이 설정을 해줍니다. location /back/에 있는 8000포트는 백엔드 서버가 돌아가는 포트번호이고, /에 있는 3000포트는 프론트가 돌아가는 포트입니다.

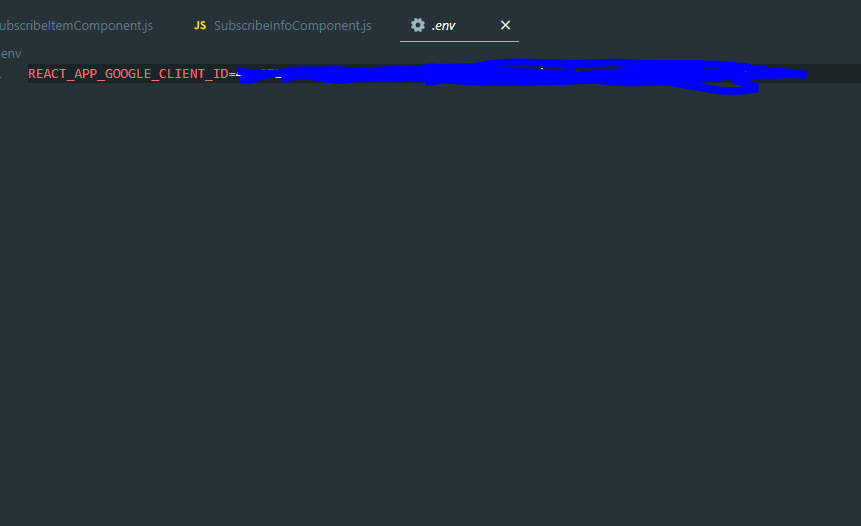
* 설정을 끝내고 나면 sudo service nginx restart를 입력해 nginx를 재시작합니다.

1. 프로퍼티 정의 파일 목록
   1. 백엔드
      1. Application.yml



Application.yml에는 database에 접속하기 위한 정보, 구글 소셜로그인에 필요한 정보, jwt 설정과 같은 정보가 담겨있습니다.

* 1. 프론트엔드
     1. SocialApi.js



.env 파일에는 소셜로그인을 하기 위한 정보가 있습니다.