



UNDER THE SEA

삼성SW청년아카데미 서울캠퍼스 6기

Gitlab 소스 클론 이후 빌드 및 배포 과정 정리

A105팀 UTS

연진우, 남정현, 백승윤, 오재우, 정현정, 황수진

목 차

1. 사용한 기술 스택
2. 배포 세팅(Jenkins)
3. 빌드 및 배포

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. 운영 체제 : Windows10
 2. IDE
 - 가) Visual Studio Code 1.64.2
 - 나) UI/UX : Figma, StoryBook
 3. 데이터베이스 : MySQL Workbench 8.0.27
 4. 서버 : AWS EC2
 - 가) Ubuntu 20.04 LTS
 - 나) Docker 20.10.13

6) 세부 사항

1. Backend

- 가) Typescript 4.6.2
- 나) Node.js 16.14.0
- 다) swagger-ui-express 4.3.0
- 라) typeORM 0.3.0
- 마) npm 8.5.5
- 바) express 4.17.3

2. Frontend

- 가) React 17.0.2
- 나) npm 8.5.5
- 다) webpack 5.70.0
- 라) recoil 0.6.1
- 마) react-hook-form 7.28.1
- 바) react-router-dom 6.2.2
- 사) storybook 6.4.19
- 아) axios 0.26.1
- 자) antd 4.19.2
- 차) framer-motion 6.2.8
- 카) Typescript 4.6.2

3. BlockChain

가) Ethers 5.6.2

나) solidity 0.7.20

다) Typescript 4.6.3

라) Hardhat 2.9.2

2. 배포 세팅(Jenkins)

Gitlab과 연동된 Jenkins의 관리페이지로 접속합니다. (<http://j6a105.p.ssafy.io:9090>)

1) jenkins 플러그인 설치

jenkins 관리 → 플러그인 관리에서 필요한 플러그인 설치 (설치가능 탭에서 Search 가능)

- Blue Ocean을 검색 후 검색된 모든 플러그인
- gitlab을 검색하고 검색된 모든 플러그인
- Node Js 플러그인

2) jenkins에 gitlab 연결

jenkins 관리 → 시스템 설정

Gitlab

☒ Enable authentication for '/project' end-point

GitLab connections

Connection name
connection

A name for the connection

Gitlab host URL
https://lab.ssafy.com

The complete URL to the Gitlab server (e.g. http://gitlab.mydomain.com)

Credentials
GitLab API token Add

API Token for accessing Gitlab

고급...

Test Connection

삭제

추가

- 시스템 설정 후 내리면 GitLab 부분
 - Connection name → 원하는 아무 이름이나 작성
 - Gitlab host URL에 gitlab 주소 작성
→ https://xxx.xxxxxx.com 까지만 기입
 - Credentials → Add → jenkins를 눌러 새로 생성

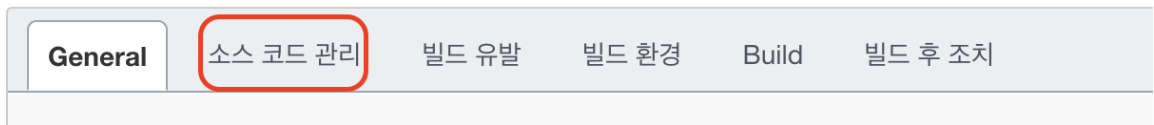
- 위 화면과 같은 화면이 나오면 Kind → GitLab API token으로 변경
- API tokens는 gitlab에서 발급받은 personal token 입력
 - personal token 발급 방법
 - gitlab으로 가서 User Settings → Access Tokens

- 원하는 token name과 만료 기간을 정해주고 api에 체크한 후
Create personal access token을 누름 → 생성 완료
→ 화면 상단에 token 발급된 거 확인 가능
- API tokens에 발급한 token 값 입력해주고
- ID → gitlab에서 사용하는 아이디 기입 (@뒤를 제외한 아이디만!)
- Add하면 추가 완료
- Credentials에 방금 만든 것으로 설정해주면 완료 → Apply 및 저장도 해주기

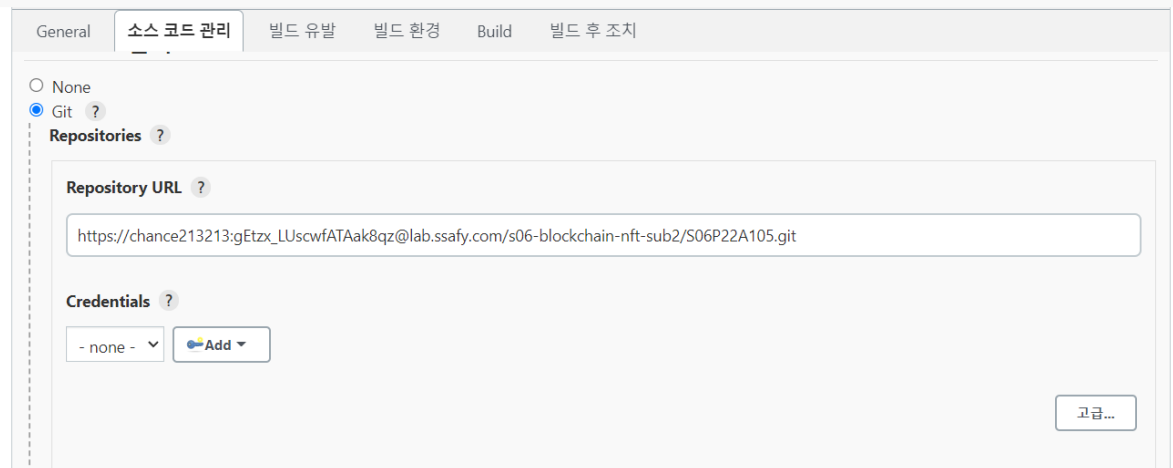
3) jenkins에 프로젝트 연결 후 기본 설정하기

- dashboard → 새로운 Item
- Enter an item name에 프로젝트 이름 작성 (아무거나 상관없음)
- Freestyle project 선택 후 완료


4) 프로젝트 git 연결하기



- 프로젝트로 들어가서 구성 → 소스코드 관리 탭에서 Git을 선택



- Repository URL에 git project clone 주소 작성
 - Failed to connect to repository ... ERROR 나는 경우
 - https://gitlabID:PersonalAccessToken@Repository주소로 작성
 - PersonalAccessToken 아까 위에서 발급 받은 것
 - Credentials → Add → Jenkins



Jenkins Credentials Provider: Jenkins

Add Credentials

Domain
Global credentials (unrestricted)

Kind
SSH Username with private key

Scope
Global (Jenkins, nodes, items, all child items, etc)

ID

Description

Username

☐ Treat username as secret

Private Key
☐ Enter directly

Passphrase

Add Cancel

- kind → SSH Username with private key
 - ID → gitlab ID (@뒤 제외)
 - Username → gitlab에 로그인하는 email 형식
 - Private key → Enter directly → Add
 - 입력하는 곳에 ssh key를 입력해야함
 - gitlab으로 가서 User Settings → SSH keys에서 key를 입력해야 함
 - 이때 입력하는 키는 배포할 서버에서 생성 가능
 - 서버 접속 후
 - `ssh-keygen -t rsa -C "gitlabID(이메일형식)"`
 - `cat home/ubuntu/.ssh/id_rsa.pub` 열고 출력된 key 값 전체 복사
 - gitlab으로 돌아와서 ssh key 입력해 주고 타이틀, 유효기간 설정해주면 키 생성 완료 → 이 key를 이제 jenkins에 입력
- Branches to build에 배포될 브랜치 설정

Branches to build ?

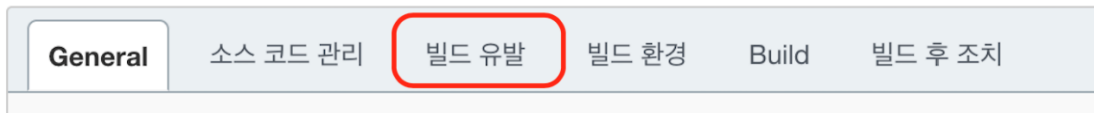
Branch Specifier (blank for 'any') ?

*/develop

X

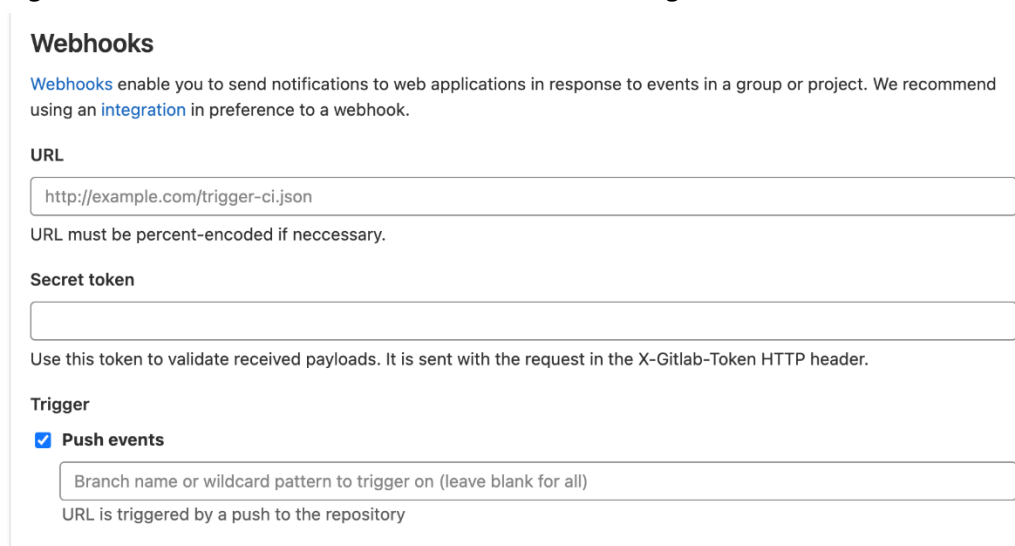
3. 빌드 및 배포

1) 프로젝트 빌드 기본설정



General 소스 코드 관리 **빌드 유발** 빌드 환경 Build 빌드 후 조치

- 구성 → 빌드 유발
- Build when a change is pushed to GitLab~~~ 체크 → 고급 → 맨 아래 generate 클릭
- generate까지 생성되면 key가 하나 생성되는데 복사해서 잘 가지고 있기!
- jenkins 설정 apply → 저장
- gitlab으로 가서 배포하고자 하는 프로젝트에서 Settings → Webhooks



Webhooks

Webhooks enable you to send notifications to web applications in response to events in a group or project. We recommend using an [integration](#) in preference to a webhook.

URL

URL must be percent-encoded if neccessary.

Secret token

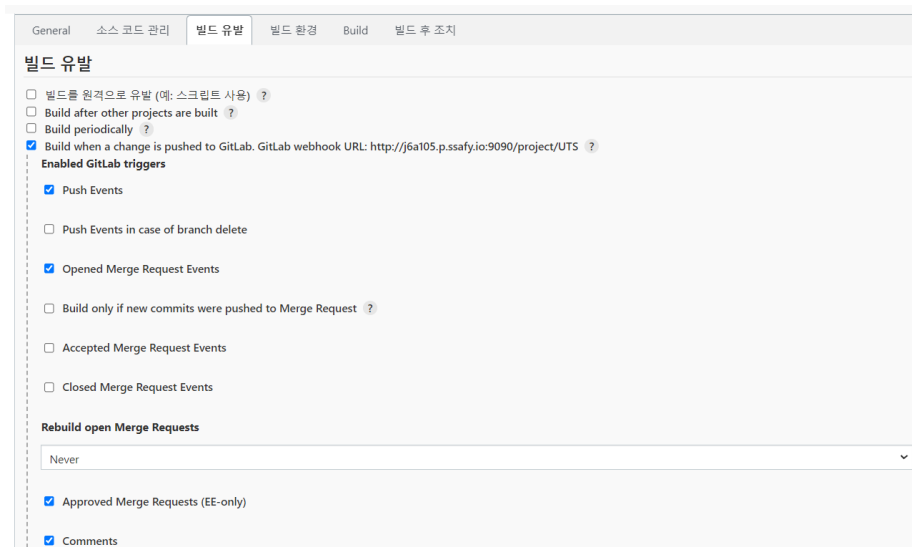
Use this token to validate received payloads. It is sent with the request in the X-Gitlab-Token HTTP header.

Trigger

☒ **Push events**

URL is triggered by a push to the repository

- URL에 jenkins 프로젝트 url을 입력 (빌드유발 탭에서 확인 가능)



General 소스 코드 관리 **빌드 유발** 빌드 환경 Build 빌드 후 조치

빌드 유발

☐ 빌드를 원격으로 유발 (예: 스크립트 사용) ?

☐ Build after other projects are built ?

☐ Build periodically ?

☒ Build when a change is pushed to GitLab. GitLab webhook URL: <http://j6a105.p.ssafy.io:9090/project/UTS> ?

Enabled GitLab triggers

☒ Push Events

☐ Push Events in case of branch delete

☒ Opened Merge Request Events

☐ Build only if new commits were pushed to Merge Request ?

☐ Accepted Merge Request Events

☐ Closed Merge Request Events

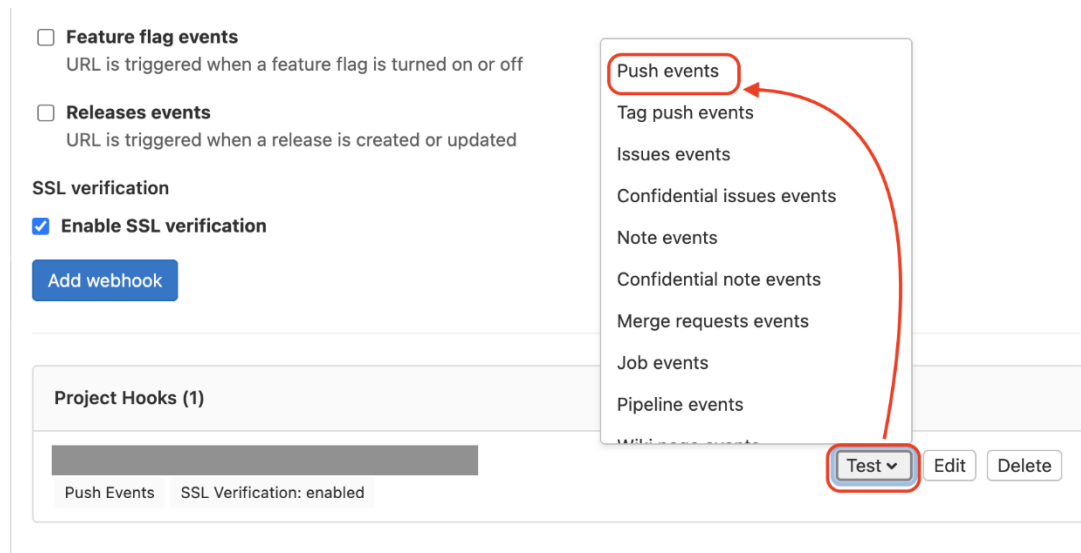
Rebuild open Merge Requests

☒ Approved Merge Requests (EE-only)

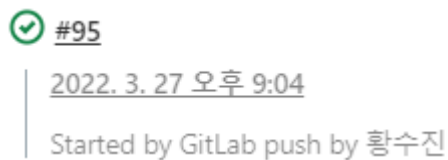
☒ Comments

- secret token에다가 아까 복사해둔 key를 입력 후 push events에는 푸시가 일어날 브랜치 입력
- 이제 아래의 Add webhook을 클릭
- hook이 생성되면 테스트에서 push events설정하고 test

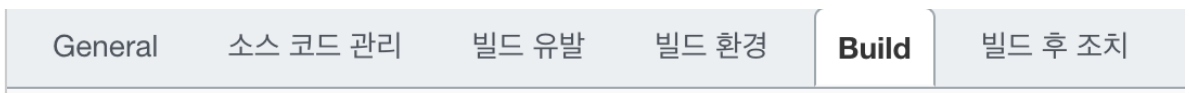
- 상단에 Hook executed successfully : HTTP 200 뜨면 성공



- jenkins로 가면 test build가 진행중이고 성공 여부를 확인 할 수 있음
(이때 실패하면 위에 설정 다 제대로 따라했는지 확인해보기)



2) 생성한 프로젝트 빌드와 배포 설정



- 구성 → Build
- Execute shell에 명령어 입력
- 제일 처음에는 도커 빌드부터 시작

```
docker stop ub # 현재 실행 중인 도커 중지
```

```
docker rm ub # 중지시킨 도커 삭제
```

```
docker rmi uts-backend # 중지시킨 도커 이미지 삭제
```

```
docker build -t uts-backend /var/lib/jenkins/workspace/UTS/backend # 도커 빌드
```

```
docker run -d --name ub -p 8080:8080 uts-backend # 도커 실행
```

```
docker stop uf
```

```
docker rm uf
```

```
docker rmi uts-frontend
```

```
docker build -t uts-frontend /var/lib/jenkins/workspace/UTS/frontend
```

```
docker run -d --name uf -p 3000:3000 uts-frontend
```

- /var/lib/jenkins/workspace/UTS : 프로젝트 폴더 위치

- frontend, backend 각각 도커로 띄워서 실행함

* frontend Dockerfile

```
FROM node:16

# root 에 app 폴더를 생성
RUN mkdir /app

# work dir 고정
WORKDIR /app

# work dir 에 build 폴더 생성 /app/build
RUN mkdir ./build

# host pc의 현재경로의 build 폴더를 workdir 의 build 폴더로 복사
ADD ./build ./build

COPY . /app
COPY package.json /app/package.json
RUN npm i

CMD ["npm", "start"]
```

*backend Dockerfile

```
FROM node:16

RUN mkdir /app
WORKDIR /app

#npm install 을 위해, package.json과 package-lock.json을 먼저 copy 해둠
COPY package*.json /app/
COPY tsconfig.json ./
COPY src /app/src

RUN ls -a

RUN npm install

# COPY . /app

EXPOSE 8080
#컨테이너가 켜지자마자 실행할 명령어
#npm start : package.json의 scripts에 있는 start 명령어를 실행
# CMD ["npm", "start"]

CMD [ "npm", "run", "start:dev" ]
```

- 이제 저장하고 Build Now를 클릭하면 build가 생성되고 자동으로 배포 진행됨