포팅 메뉴얼

1. Gitlab 소스 클론 이후 빌드 및 배포할 수 있도록 정리한 문서
   1) 사용한 JVM , 웹서버, WAS 제품 등의 종류와 설정값, 버전(IDE버전 포함) 기재
   2) 빌드 시 사용되는 환경 변수 등의 주요 내용 상세 기재
   3) 배포 시 특이사항 기재
   4) DB 접속 정보 등 프로젝트(ERD)에 활용되는 주요 계정 및 프로퍼티가 정의된 파일 목록

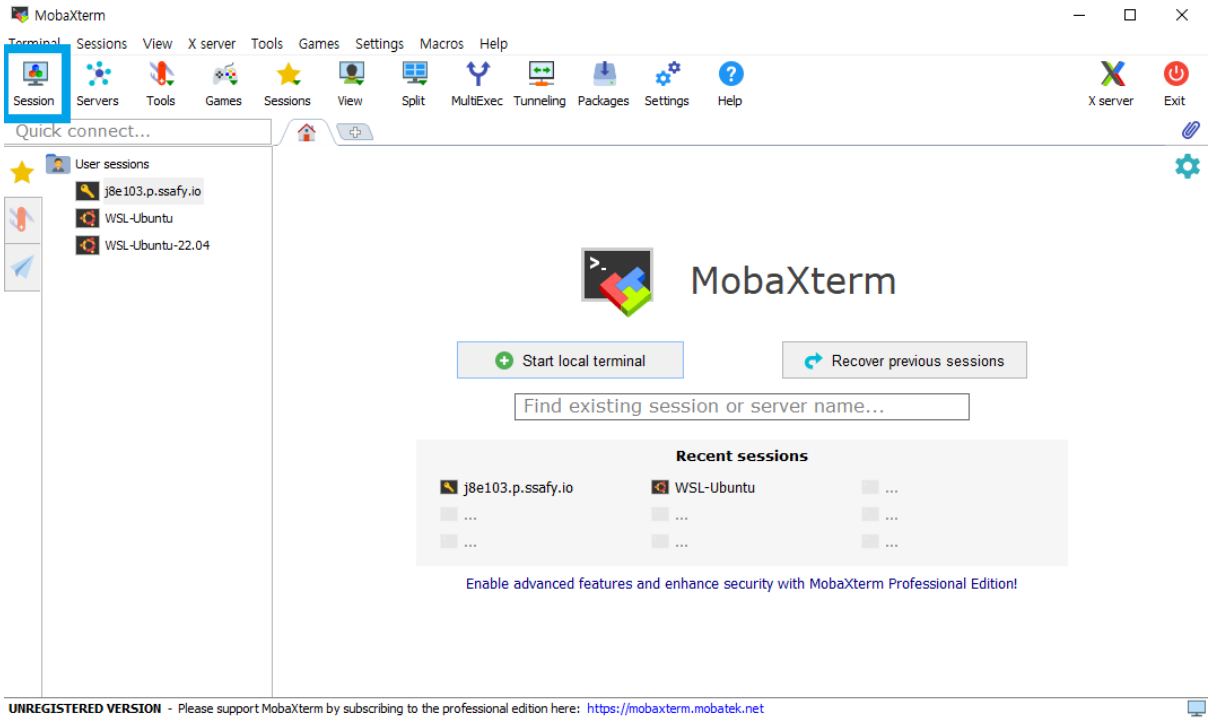
버전 정보

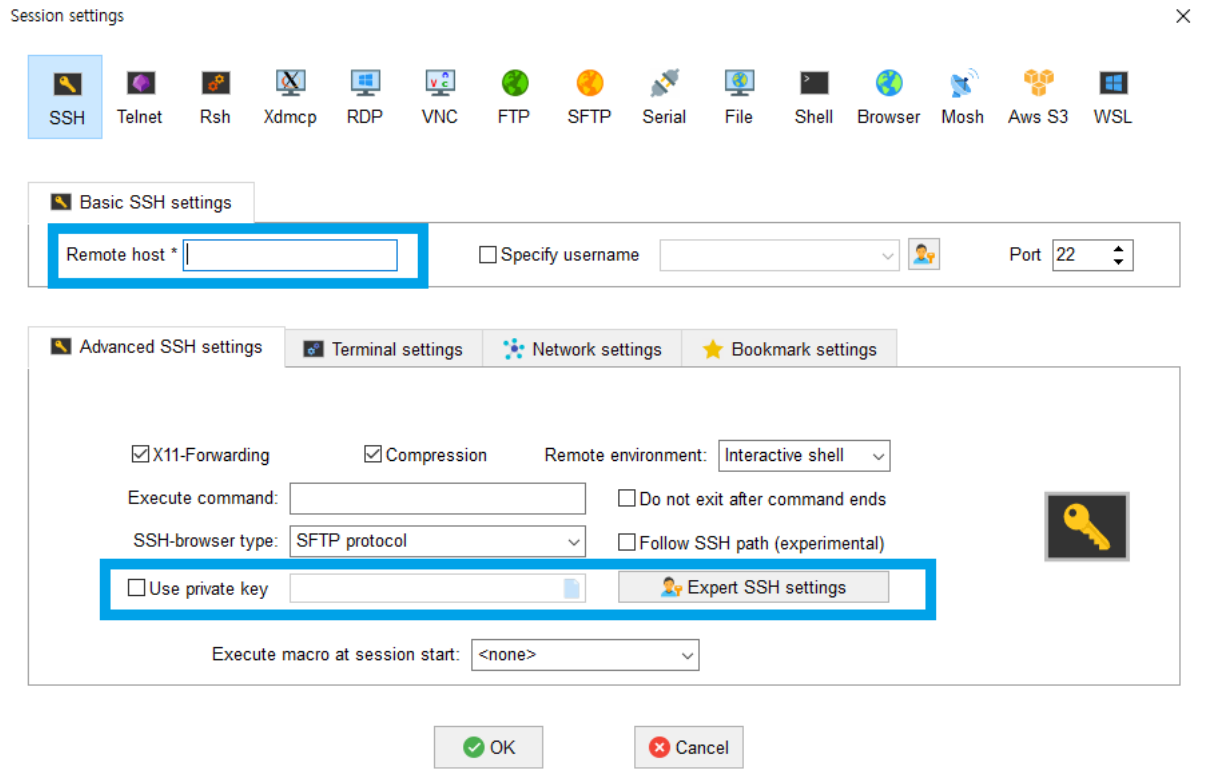
- 호스트 머신(EC2) 설정
  - Ubuntu : `20.04.6 LTS`
  - Docker : `23.0.1`
  - Docker-compose : `1.25.0`
- Frontend 설정
  - Node : `18.13.0`
  - npm : `9.3.1`
  - Nginx : `nginx/1.18.0 (Ubuntu)`
- Backend 설정
  - openjdk version : `jdk-11.0.17`
  - Java(TM) SE Runtime Environment 18.9 (build 11.0.17+10-LTS-269)
  - Java HotSpot(TM) 64-Bit Server VM 18.9 (build 11.0.17+10-LTS-269, mixed mode)
  - Spring Framework : `5.3.24`
  - Spring Boot : `2.7.7`
- IDE 설정
  - IntelliJ : `2022.3.1`

EC2 원격 접속

MobaXTerm 설치

https://mobaxterm.mobatek.net/download.html





- Remote host에 Public IP 입력
- Specify username에 사용할 유저 이름 입력 (현 프로젝트는 ubuntu를 사용합니다)
- Use private key에 pem 키 입력
- OK 누르면 연결 완료

Docker와 Docker-compose 설치

1) 사전 패키지 설치

```
sudo apt update
sudo apt install apt-transport-https ca-certificates curl software-properties-common
```

2) 키 생성

: 저장소 인증

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

: 키 등록 확인

```
sudo apt-key fingerprint OEBFCD88
```

3) Repository 추가 후 Update

```
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
sudo apt update
```

4) Docker 설치

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-compose
```

: 설치 확인

```
sudo systemctl status docker
```

5) Docker 권한 설정 (현재 유저 이름)

```
sudo usermod -aG docker ${USER} // 현재 유저이름으로 자동 등록
```

방화벽 설정

1) ufw 설치

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install ufw
```

2) ufw 상태 확인

: 열려있는 포트 및 패킷 허용 여부 확인

```
sudo ufw status
sudo ufw status verbose
```

3) 활성/ 비활성화

```
sudo ufw enable     // 활성
sudo ufw disable    // 비활성
```

4) 특정 포트 열기

```
ufw allow 포트번호
```

- `22` : SSH
- `80` : HTTP
- `443` : HTTPS
- `5000` : Spring
- `9090` : Jenkins

Jenkins 설정

젠킨스 이미지 가져오기

docker-compose.yml 작성

```
version: '3'

services:
        jenkins:
                image: jenkins/jenkins:lts
                container_name: jenkins
                volumes:
                        - /usr/bin/docker:/usr/bin/docker
                        - /var/run/docker.sock:/var/run/docker.sock
                        - /jenkins:/var/jenkins_home
                ports:
                        - "9090:8080"
                privileged: true
                user: root
                environment:
                        TZ: "Asia/Seoul"

....
```

위의 파일이 위치하는 경로에서 아래 명령어 입력

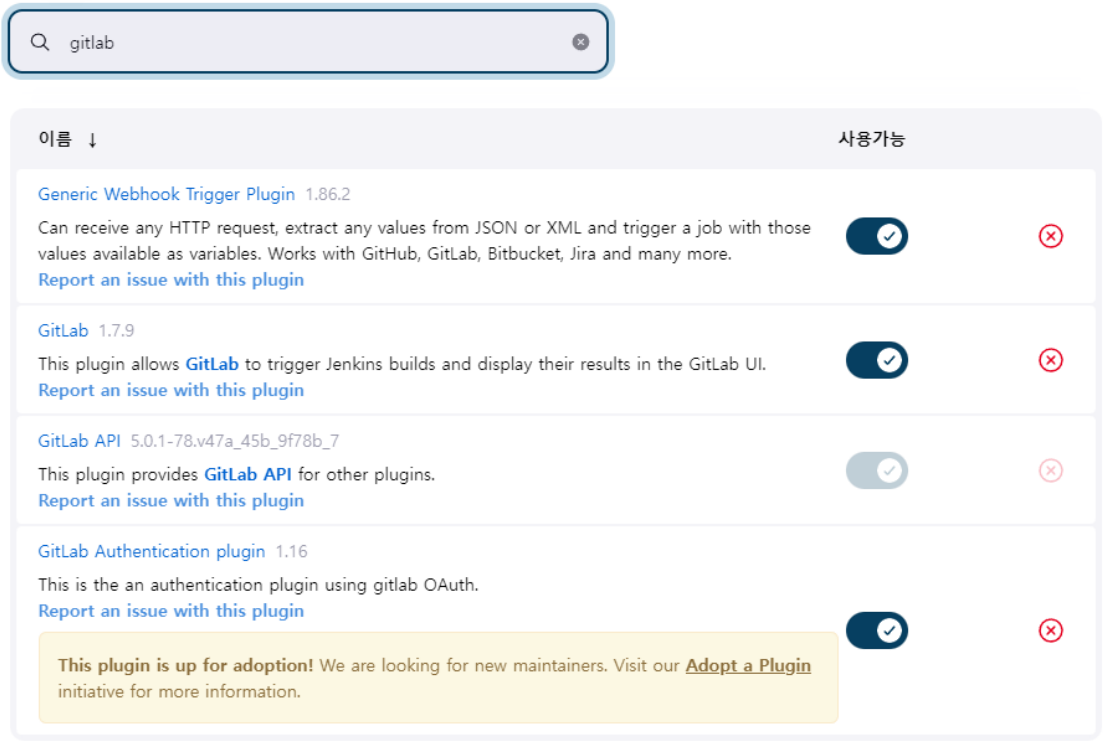
```
sudo docker-compose up -d
```

젠킨스 접속

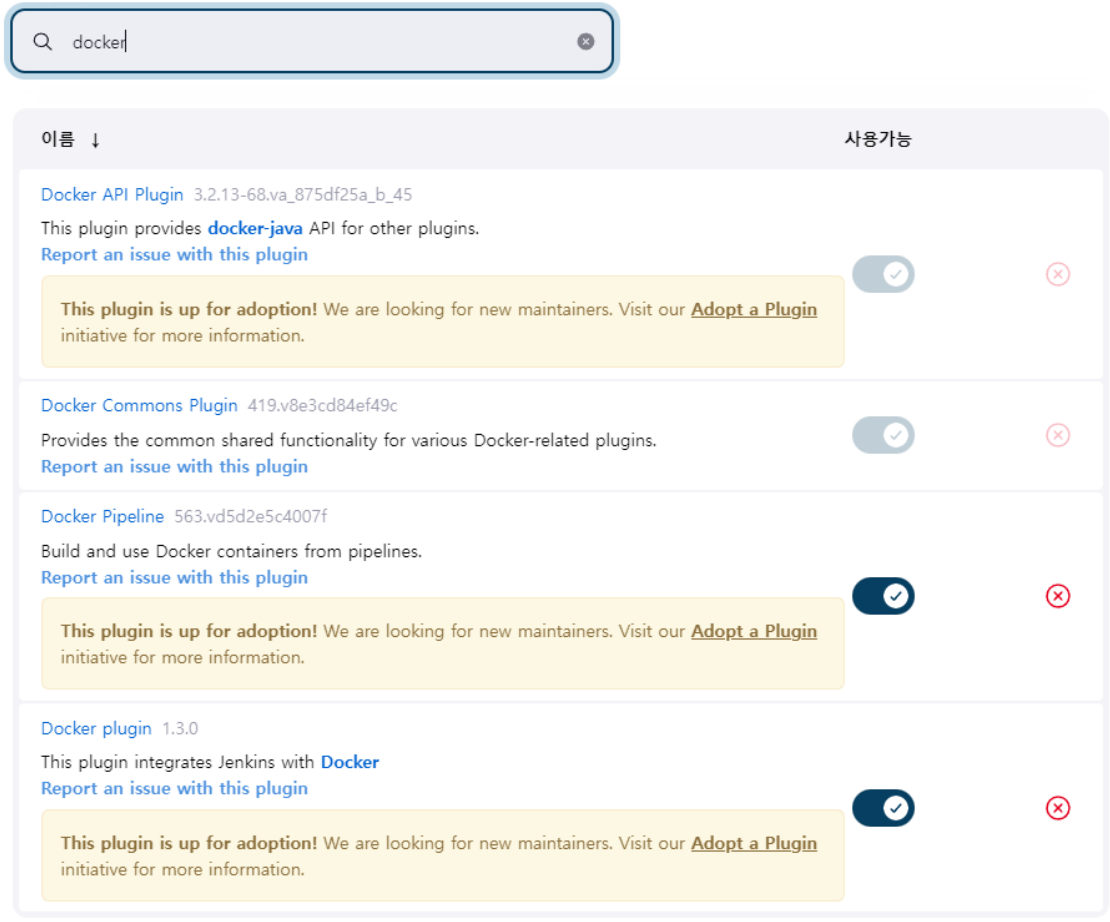
1) docker compose 파일에서 설정한 9090 포트로 접속

2) 플러그인 설치

- GitLab


gitlab
이름 ↓
사용가능
Generic Webhook Trigger Plugin 1.86.2
Can receive any HTTP request, extract any values from JSON or XML and trigger a job with those values available as variables. Works with GitHub, GitLab, Bitbucket, Jira and many more.
Report an issue with this plugin
GitLab 1.7.9
This plugin allows GitLab to trigger Jenkins builds and display their results in the GitLab UI.
Report an issue with this plugin
GitLab API 5.0.1-78.v47a_45b_9f78b_7
This plugin provides GitLab API for other plugins.
Report an issue with this plugin
GitLab Authentication plugin 1.16
This is the an authentication plugin using gitlab OAuth.
Report an issue with this plugin
This plugin is up for adoption! We are looking for new maintainers. Visit our Adopt a Plugin initiative for more information.

- Docker

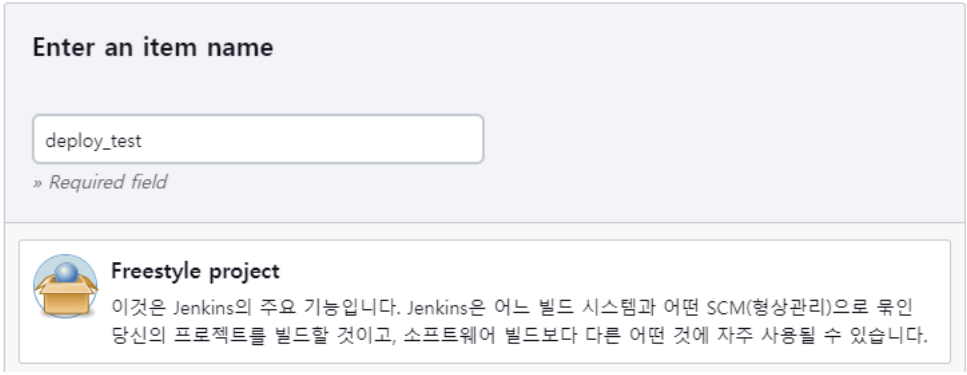

docker
이름 ↓
사용가능
Docker API Plugin 3.2.13-68.va_875df25a_b_45
This plugin provides docker-java API for other plugins.
Report an issue with this plugin
This plugin is up for adoption! We are looking for new maintainers. Visit our Adopt a Plugin initiative for more information.
Docker Commons Plugin 419.v8e3cd84ef49c
Provides the common shared functionality for various Docker-related plugins.
Report an issue with this plugin
Docker Pipeline 563.vd5d2e5c4007f
Build and use Docker containers from pipelines.
Report an issue with this plugin
This plugin is up for adoption! We are looking for new maintainers. Visit our Adopt a Plugin initiative for more information.
Docker plugin 1.3.0
This plugin integrates Jenkins with Docker
Report an issue with this plugin
This plugin is up for adoption! We are looking for new maintainers. Visit our Adopt a Plugin initiative for more information.

젠킨스 프로젝트 생성

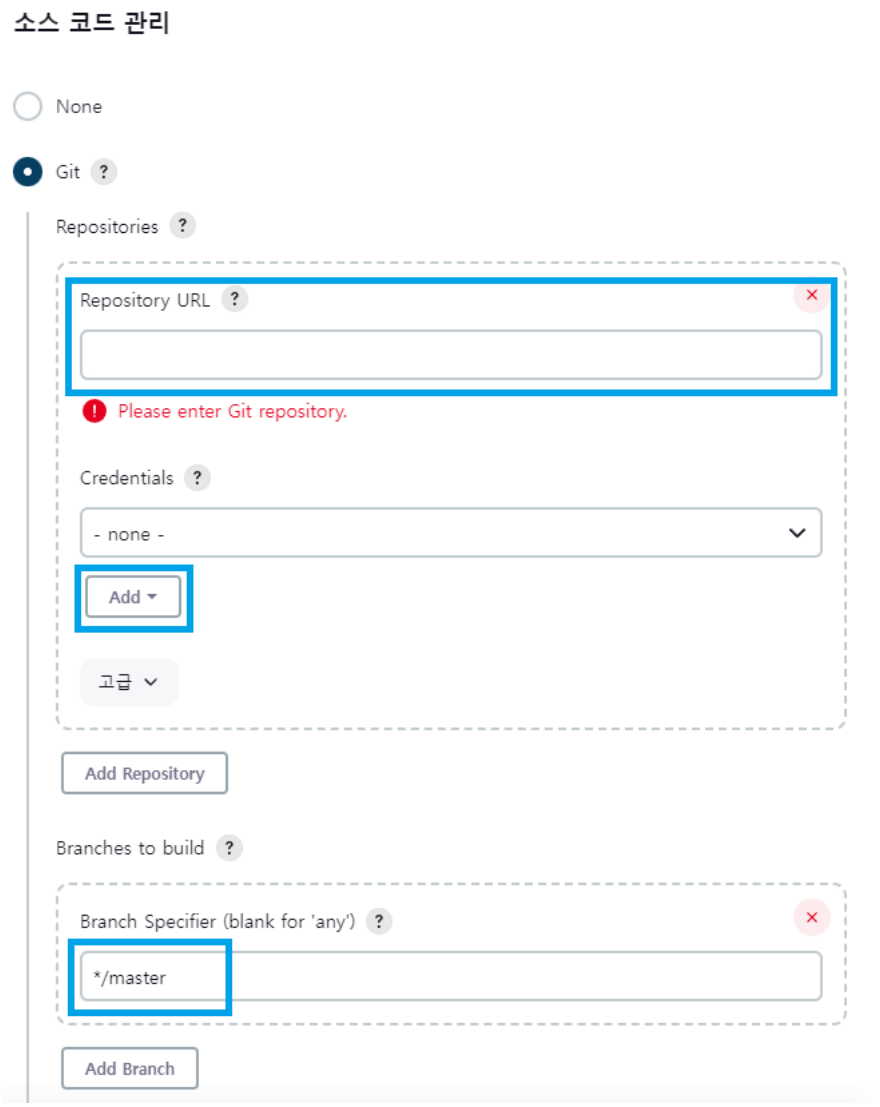
1) Dashboard에서 `새로운 Item` 선택

2) 프로젝트 명 기입 및 타입 선택

: 현 프로젝트에서는 Freestyle project로 선택했습니다

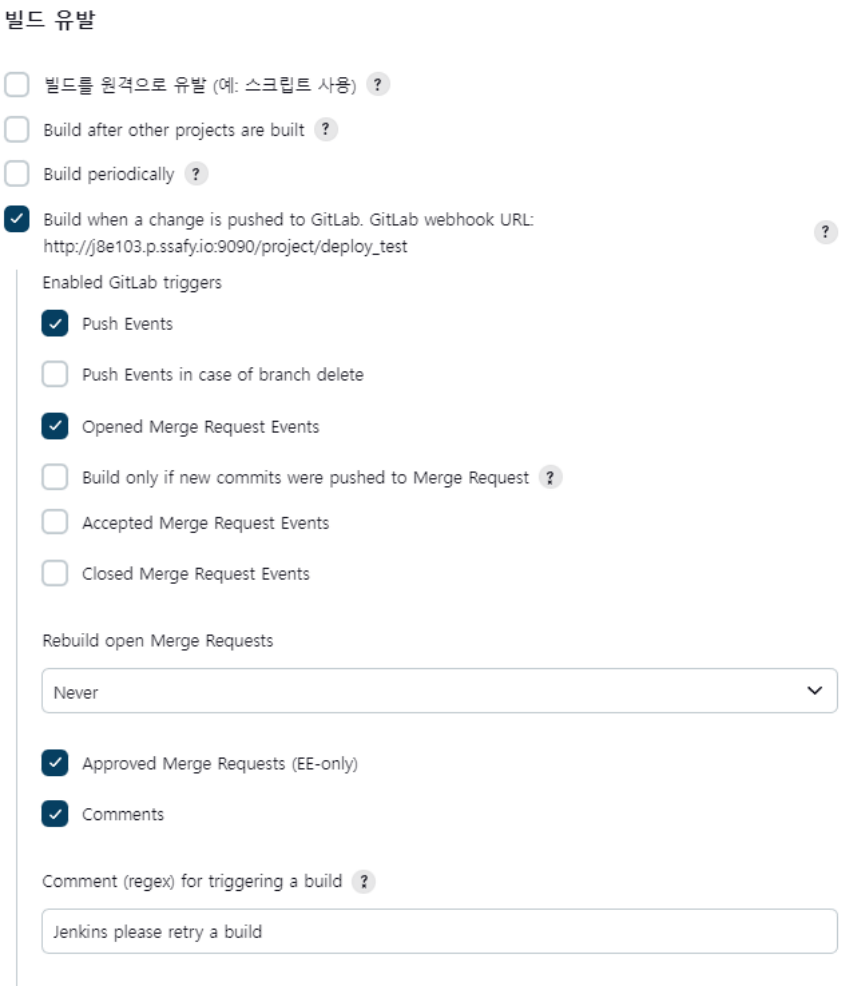


3) `소스 코드 관리` 에서 gitlab 연결

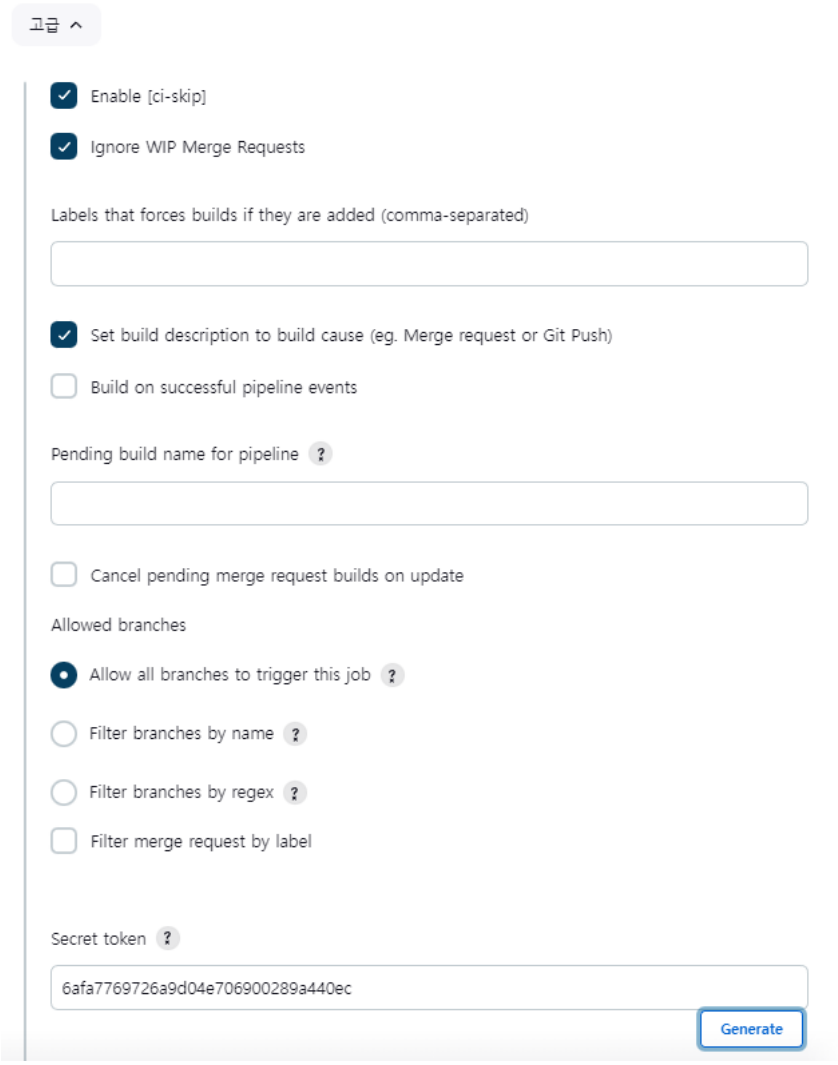


- Repository URL에 젠킨스에 연결할 Repository URL 입력
- Add를 눌러 레포지토리와 연결할 레포지토리 유저 선택 (인증용)
- 빌드를 진행할 타겟 브랜치 설정

4) 빌드 유발 설정

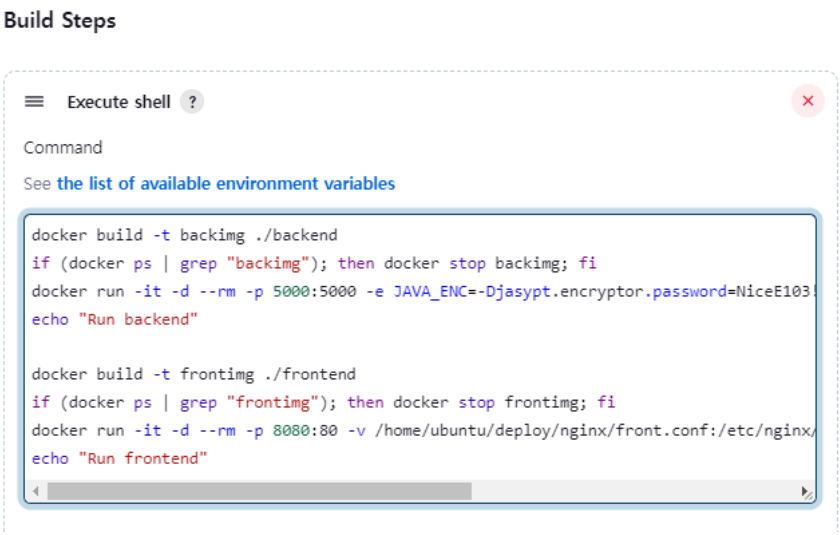


빌드를 발생할 이벤트 설정. push와 merge request를 선택.



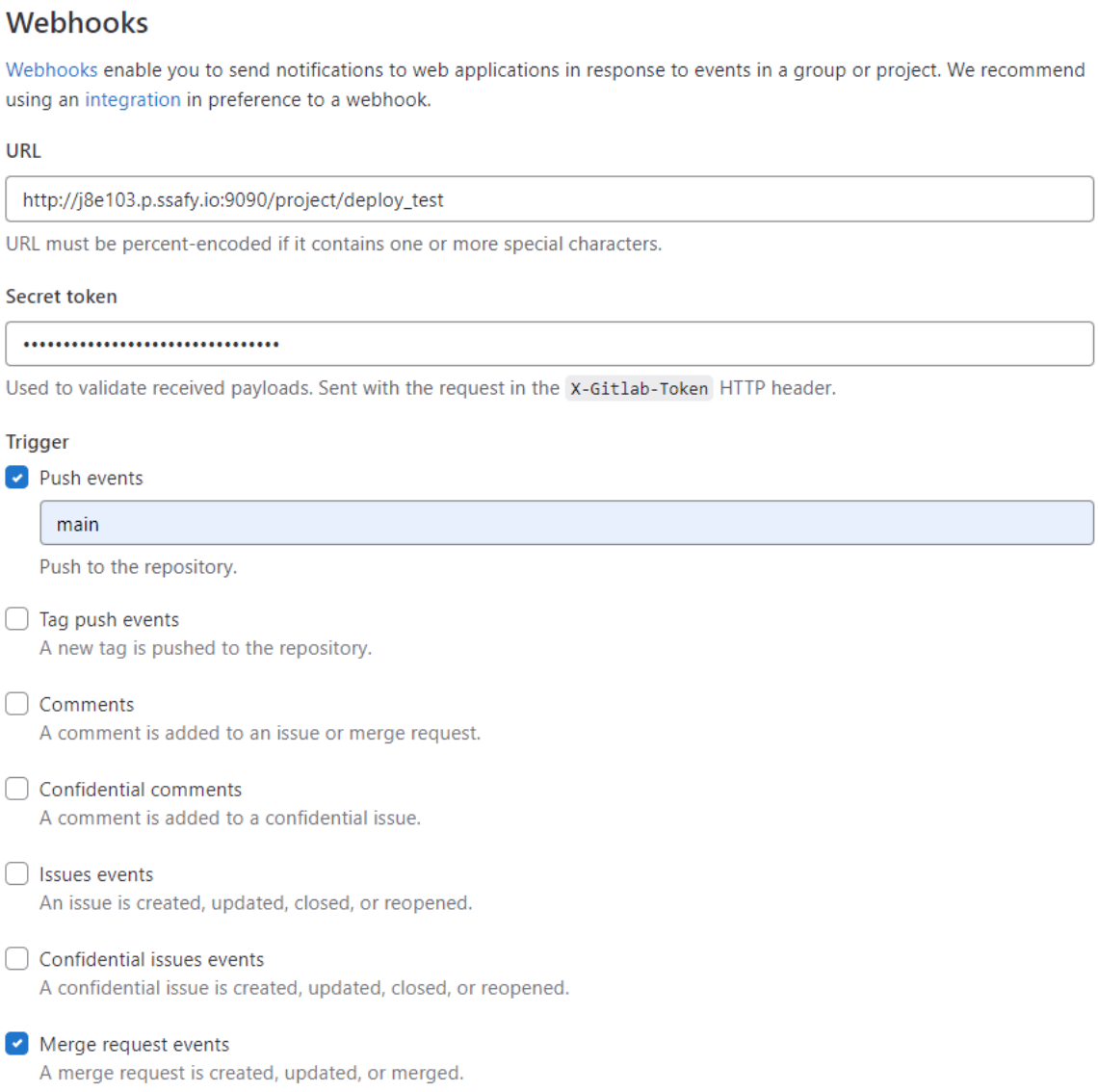
고급 탭에서 secret token을 생성. Gitlab에 web hook 연결할 때 사용됩니다.

5) Build Steps 설정

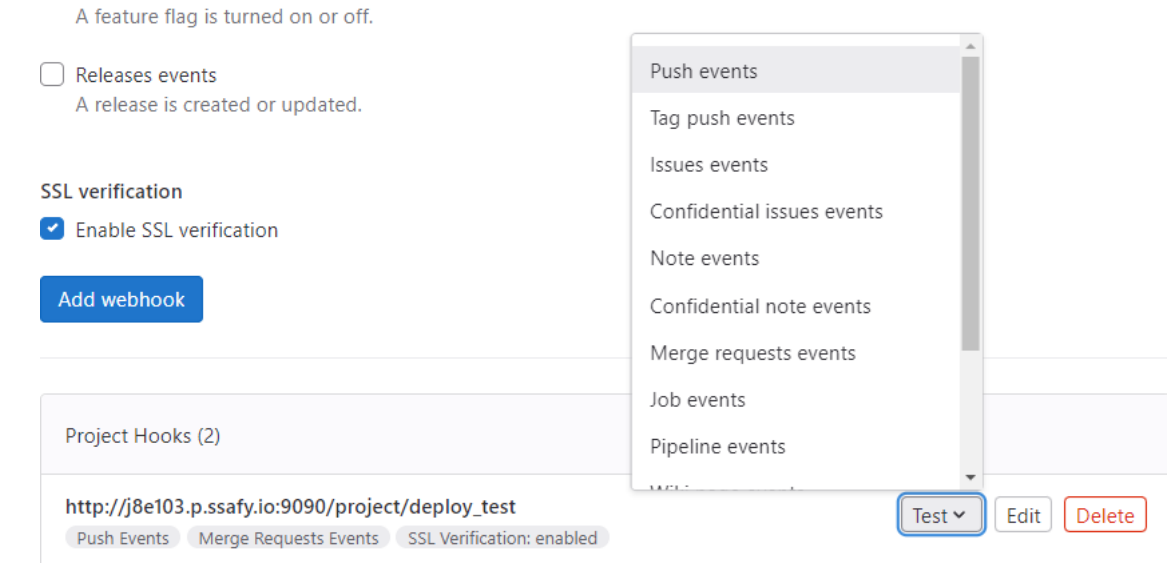


```
docker build -t backimg ./backend
if (docker ps | grep "backimg"); then docker stop backimg; fi
docker run -it -d --rm -p 5000:5000 -e JAVA_ENC=-Djasypt.encryptor.password=NiceE103! -e PROFILE=-Dspring.profiles.active=dev --nar
echo "Run backend"

docker build -t frontimg ./frontend
if (docker ps | grep "frontimg"); then docker stop frontimg; fi
docker run -it -d --rm -p 8080:80 -v /home/ubuntu/deploy/nginx/front.conf:/etc/nginx/conf.d/default.conf --name frontimg frontimg
echo "Run frontend"
```

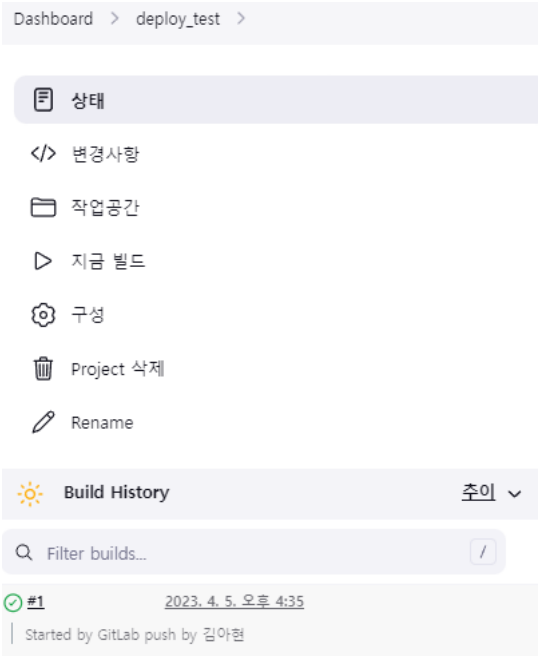
web hook 설정



Jenkins에서 생성한 URL과 Secret token을 기입하고, Trigger에 Gitlab에서 어떤 이벤트를 감지하고 Jenkins로 빌드 유발 시킬지에 대해 설정



Push Events 유발 테스트



Jenkins, Gitlab 연결 성공 확인

Nginx + HTTPS 설정

참고 사이트 : https://node-js.tistory.com/32

docker-compose.yml 파일 작성

: 위에서 작성한 docker-compose.yml 파일에 추가로 작성

```
version: '3'

services:
....
        nginx:
```

```
                image: nginx:stable-alpine
                container_name: nginx
                restart: unless-stopped
                ports:
                        - "80:80"
                        - "443:443"
                volumes:
                        - ./data/nginx/nginx.conf:/etc/nginx/conf.d/nginx.conf
                        - ./data/certbot/conf:/etc/letsencrypt
                        - ./data/certbot/www:/var/www/certbot
                command: "/bin/sh -c 'while :; do sleep 6h & wait $${!}; nginx -s reload; done & nginx -g \"daemon off;\"'"
                environment:
                        TZ: "Asia/Seoul"

        certbot:
                image: certbot/certbot
                container_name: certbot
                volumes:
                         - ./data/certbot/conf:/etc/letsencrypt
                         - ./data/certbot/www:/var/www/certbot
                entrypoint: "/bin/sh -c 'trap exit TERM; while :; do certbot renew; sleep 12h & wait $${!}; done;'"
                environment:
                        TZ: "Asia/Seoul"
```

nginx.conf 파일 작성

상대 경로 `./data/nginx/` 에 nginx.conf 파일 작성

```
server {
     listen 80;
     listen [::]:80;

     location /.well-known/acme-challenge/ {
             allow all;
             root /var/www/certbot;
     }
}
```

```
sudo docker-compose up -d
```

인증서 발급

```
curl -L <https://raw.githubusercontent.com/wmnnd/nginx-certbot/master/init-letsencrypt.sh> > init-letsencrypt.sh
chmod +x init-letsencrypt.sh
vi init-letsencrypt.sh // 도메인, 이메일, 디렉토리 수정
sudo ./init-letsencrypt.sh // 인증서 발급
```

Https 적용

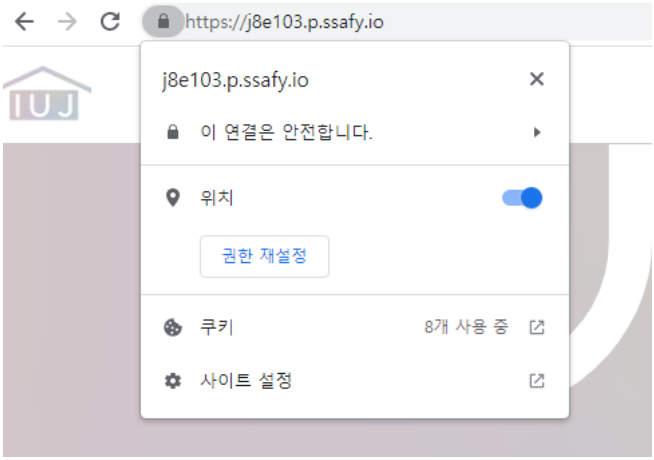
위에서 작성했던 nginx.conf 파일 수정

```
server {
        listen 80;
        listen [::]:80;

        location /.well-known/acme-challenge/ {
                root /var/www/certbot;
        }

        location / {
                proxy_pass 301 https://$host$request_uri;
        }
}

server {
        listen 443 ssl;
        server_tokens off;
```

```
        ssl_certificate /etc/letsencrypt/live/j8e103.p.ssafy.io/fullchain.pem;
        ssl_certificate_key /etc/letsencrypt/live/j8e103.p.ssafy.io/privkey.pem;
        include /etc/letsencrypt/options-ssl-nginx.conf;
        ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

        location / {
                proxy_pass  http://172.26.6.169:8080;
                proxy_set_header    Host                $http_host;
                proxy_set_header    X-Real-IP           $remote_addr;
                proxy_set_header    X-Forwarded-For     $proxy_add_x_forwarded_for;
        }
}
```

nginx 재시작

```
sudo docker-compose restart
```



Front-end 배포

도커 파일

```
FROM node:16.15.0 as build-stage
COPY package.json .
RUN npm install
COPY . .
RUN npm run build

FROM nginx:stable-alpine as production-stage
COPY --from=build-stage dist /usr/share/nginx/html
EXPOSE 8080
CMD ["nginx", "-g","daemon off;"]
```

nginx 설정

절대 경로 `/home/ubuntu/deploy/nginx/` 위치에 **front.conf** 작성

```
server {
        listen       80;
        listen  [::]:80;

        access_log  /var/log/nginx/host.access.log  main;

        location / {
                root   /usr/share/nginx/html;
                index  index.html index.htm;
```

```
                try_files $uri $uri/ /index.html;
        }

        location /assets {
                root /usr/share/nginx/html;
        }

        location /api {
                proxy_pass http://172.26.6.169:5000;
                proxy_set_header    Host                $http_host;
                proxy_set_header    X-Real-IP           $remote_addr;
                proxy_set_header    X-Forwarded-For     $proxy_add_x_forwarded_for;
        }
}
```

Back-end 배포

도커 파일

```
FROM adoptopenjdk/openjdk11 AS builder
COPY gradlew .
COPY gradle gradle
COPY build.gradle .
COPY settings.gradle .
COPY src src
RUN chmod +x ./gradlew
RUN ./gradlew bootJAR

FROM adoptopenjdk/openjdk11
COPY --from=builder build/libs/*.jar app.jar
EXPOSE 8080
ENTRYPOINT ["sh", "-c", "java ${JAVA_ENC} ${PROFILE} -jar /app.jar"]
```

환경 변수는 젠킨스에서 주입 후 빌드를 진행합니다.

- JAVA_ENC : Jasypt 적용한 secret key
- PROFILE : Spring boot의 profile 환경