포팅메뉴얼

버전

BackEnd

- springboot : 2.7.1
- jdk : 1.8.0_342
- maven : 4.0.0
- MySQL : 8.0.30-0ubuntu0.20.04.2
- Intellij : 2022.1.3

FrontEnd

- node 14.19.2
- npm 6.14.17
- react 18.1.0
- redux 4.2.0
- ckeditor 5.0.2
- ckeditor 4.0.0
- router-dom 6.3.0

Server

- AWS EC2
- Ubuntu : 20.04 LTS
- nginx : 1.18.0 (Ubuntu)
- Jenkins : 2.346.2

환경 설정

Server 환경 설정

- MySQL 설치

```
# 우분투 서버 업데이트, 업그레이드
$ sudo apt-get update
$ sudo apt-get upgrade

# 설치
$ sudo apt-get install mysql-server

# 구동
$ sudo systemctl start mysql.service

# 설치
$ sudo apt-get install mysql-server

# 구동
$ sudo systemctl start mysql.service
```

```
# MySQL 접속
$ sudo mysql

# 현재 mysql에서 기본으로 세팅되어있는 유저 확인
mysql> SELECT user,authentication_string,plugin,host FROM mysql.user;

# DB에 외부 접속하기 위해 새로운 계정 생성
# (% : 어떤 ip로도 접속 가능하도록 하기 위함)
mysql> CREATE USER '계정이름'@'%' IDENTIFIED BY '비밀번호';

# 권한 부여
mysql> GRANT ALL PRIVILEGES ON *.* TO '계정이름'@'%' WITH GRANT OPTION;
mysql> FLUSH PRIVILEGES;
```

- java 설치

```
# openjdk 설치
$ sudo apt-get install openjdk-8-jdk

# java 버전 확인
$ java -version,
$ javac -version

## java 환경 변수 설치
# javac 위치 확인
$ which javac
# ==>  /usr/lib/jvm/java-8-openjdk-amd64/bin/ 위치가 여기라면
# $JAVA_HOME은 /usr/lib/jvm/java-8-openjdk-amd64으로 설정하도록 한다.

# 환경변수 설정 위해 profile 편집기에 진입
$ sudo vi /etc/profile
```

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export PATH=$JAVA_HOME/bin:$PATH
export CLASS_PATH=$JAVA_HOME/lib:$CLASS_PATH
```

```
# 위의 내용을 하단 추가
# esc -> :wq! -> enter 해서 저장
# 수정 완료 후에 profile을 reload를 위해서 리부팅을 진행한다
$ sudo reboot now
```

- nginx 설치

```
# nginx 설치
$ sudo apt-get install nginx
```

빌드 및 배포

FrontEnd 빌드 & 배포

```
git clone https://lab.ssafy.com/s07-webmobile2-sub2/S07P12C109.git

cd S07P12C109/build
npm install
npm run build
sudo cp -r web /home/ubuntu/app/web
sudo systemctl restart nginx

# 프론트는 수동배포로 진행함
```

let’s encrypt SSL로 https 적용

```
# Nginx 실행
$ sudo service nginx start

# certbot 다운로드
$ wget https://dl.eff.org/certbot-auto

# snap을 이용하여 core 설치 -> snap을 최신 버전으로 유지하기 위해 설치
$ sudo snap install core

# core를 refresh 해준다.
$ sudo snap refresh core

# 기존에 잘못된 certbot이 설치되어있을 수도 있으니 삭제 해준다.
$ sudo apt remove certbot

# certbot 설치
$ sudo snap install --classic certbot

# certbot 명령을 로컬에서 실행할 수 있도록 snap의 certbot 파일을 로컬의 cerbot과 링크(연결)
$ ln -s /snap/bin/certbot /usr/bin/certbot

# certbot을 이용해 ssl 인증서를 받아온 뒤 cerbot이 스스로 nginx설정을 해주도록한다.
# 아래 명령을 수행하면 여러 질문이 등장한다. 질문에 맞춰 답을 해주면 된다.
# domain을 입력하라는 질문에서는 본인이 사용할 Domain을 입력해주면 된다.
$ sudo certbot --nginx
```

BackEnd 빌드 & 배포

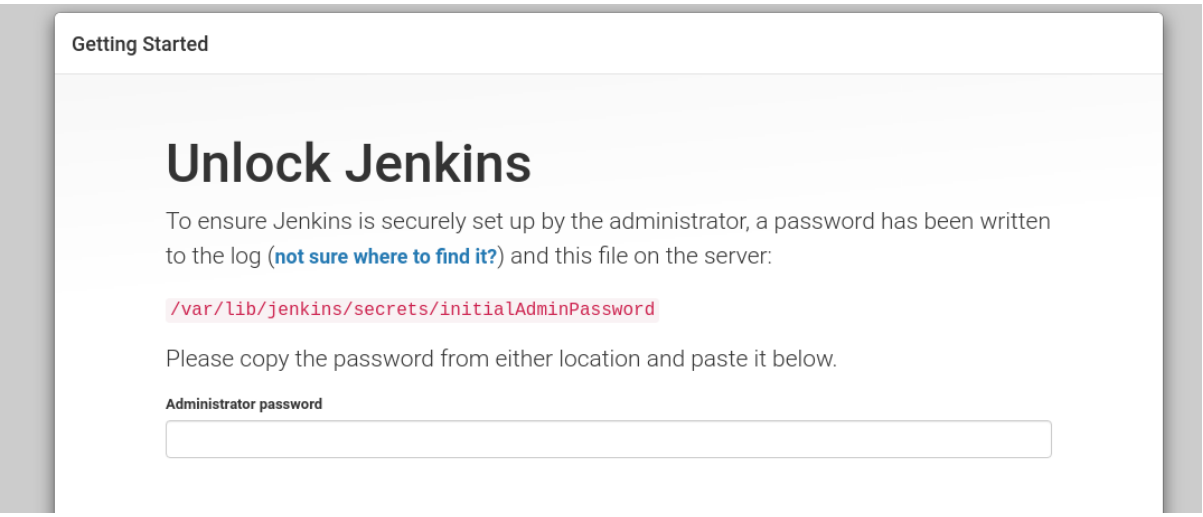
- jenkins 설치

```
## Jenkins 설치를 위해 Repository key 추가
$ wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -

# 서버의 sources.list에 Jenkins 패키지 저장소를 추가
$ sudo sh -c 'echo deb https://pkg.jenkins.io/debian-stable binary/ > \
    /etc/apt/sources.list.d/jenkins.list'

# 패키지 인덱스 정보 업데이트
$ sudo apt-get update

## Jenkins 패키지 설치
$ sudo apt-get install jenkins

# Jenkins 실행하기
$ sudo systemctl start jenkins

# Jenkins 상태 체크하기
$ sudo systemctl status jenkins

## Jenkins 포트 변경하기
sudo vi /lib/systemd/system/jenkins.service

# HTTP_PORT를 9090 포트로 변경
# esc -> :wq! -> enter 해서 저장
HTTP_PORT=9090

# 서비스 재시작
$ sudo service jenkins restart
# 정상여부 확인
$ sudo systemctl status jenkins

## 초기 패스워드 확인
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

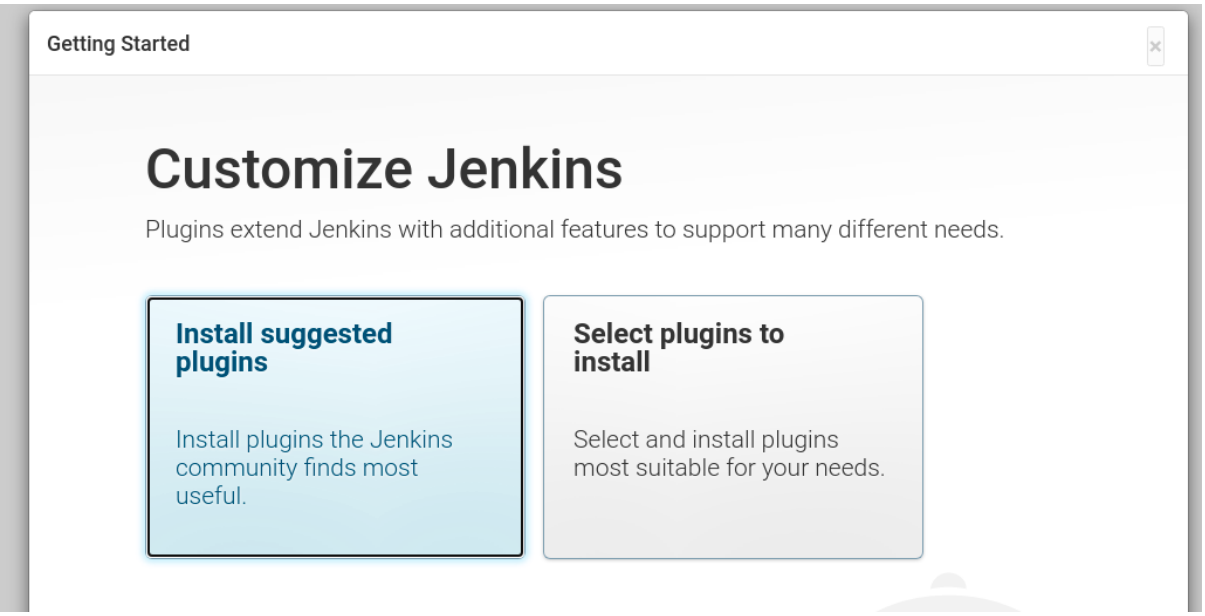
- jenkins 초기 세팅

  i7C109.p.ssafy.io:9090 접속 후

  password는 위의 나온 값으로 jenkins 로그인 진행함.



- Jenkins 초기 플러그인 설치

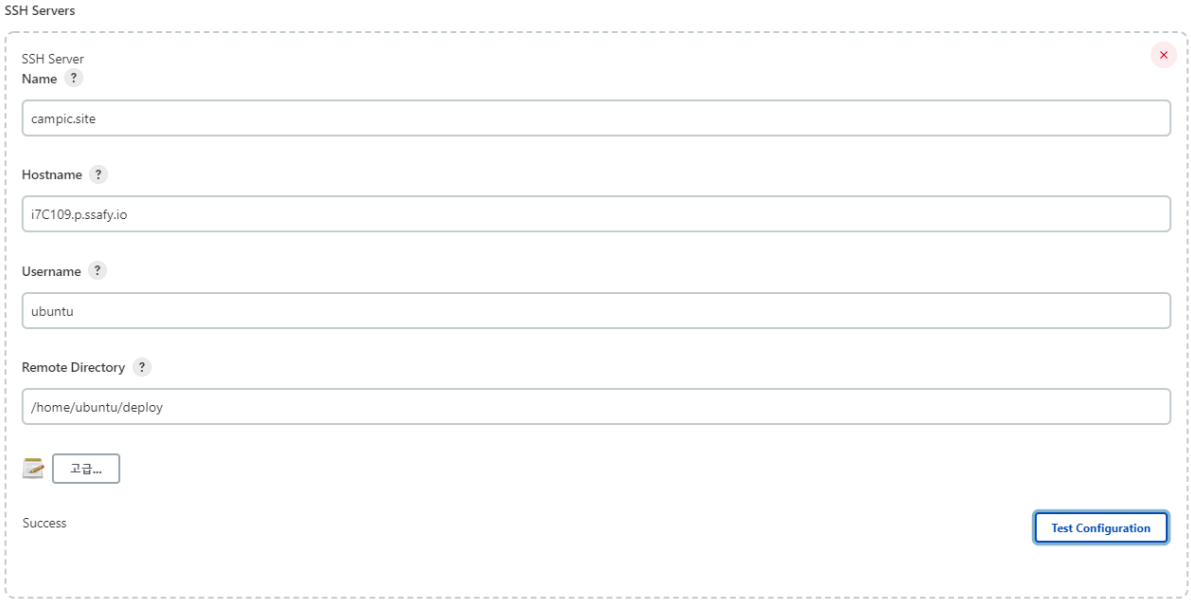


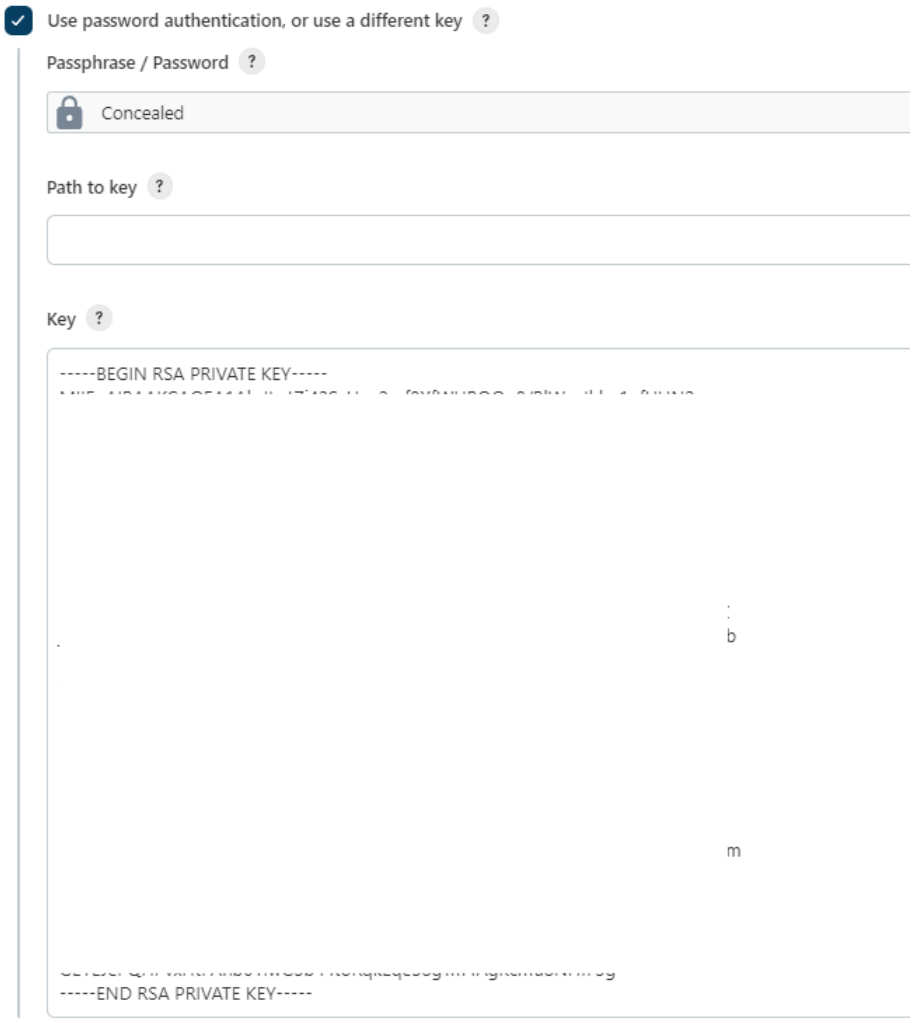
- Jenkins 관리자 로그인

  id : admin / pw : 위 나온 값.

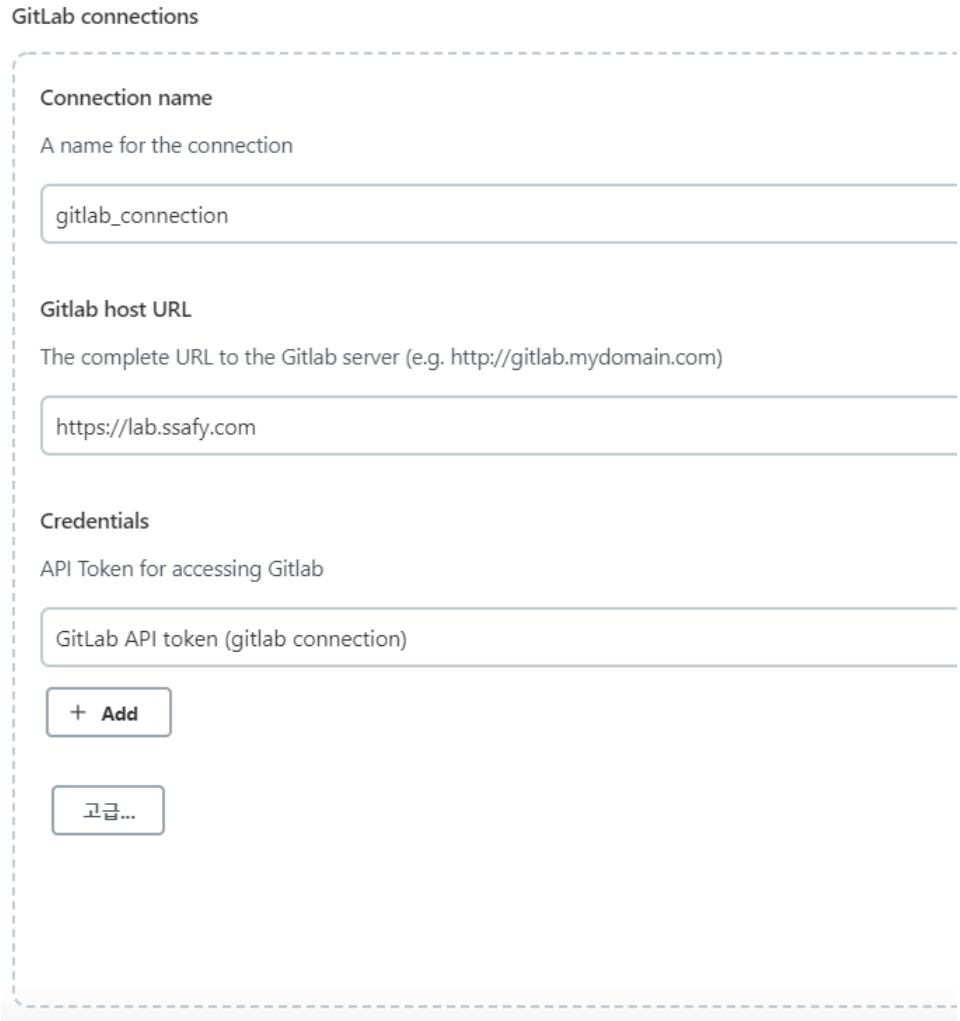
- GitLab과 연결하기 위한 Plugin 설치
  Jenkins 관리 > 플러그인 관리 > 설치가능에서 Publish Over SSH, GitLab plugin 설치.

- AWS와 젠킨스 연결
  Jenkins 관리 > 시스템 설정 > Publish over SSH 설정에서 아래 내용을 기입해주고 고급 버튼을 눌러서 RSA 키에 ssh 접속 시 필요한 pem 값 기입한 후 Test Configuration Success 확인



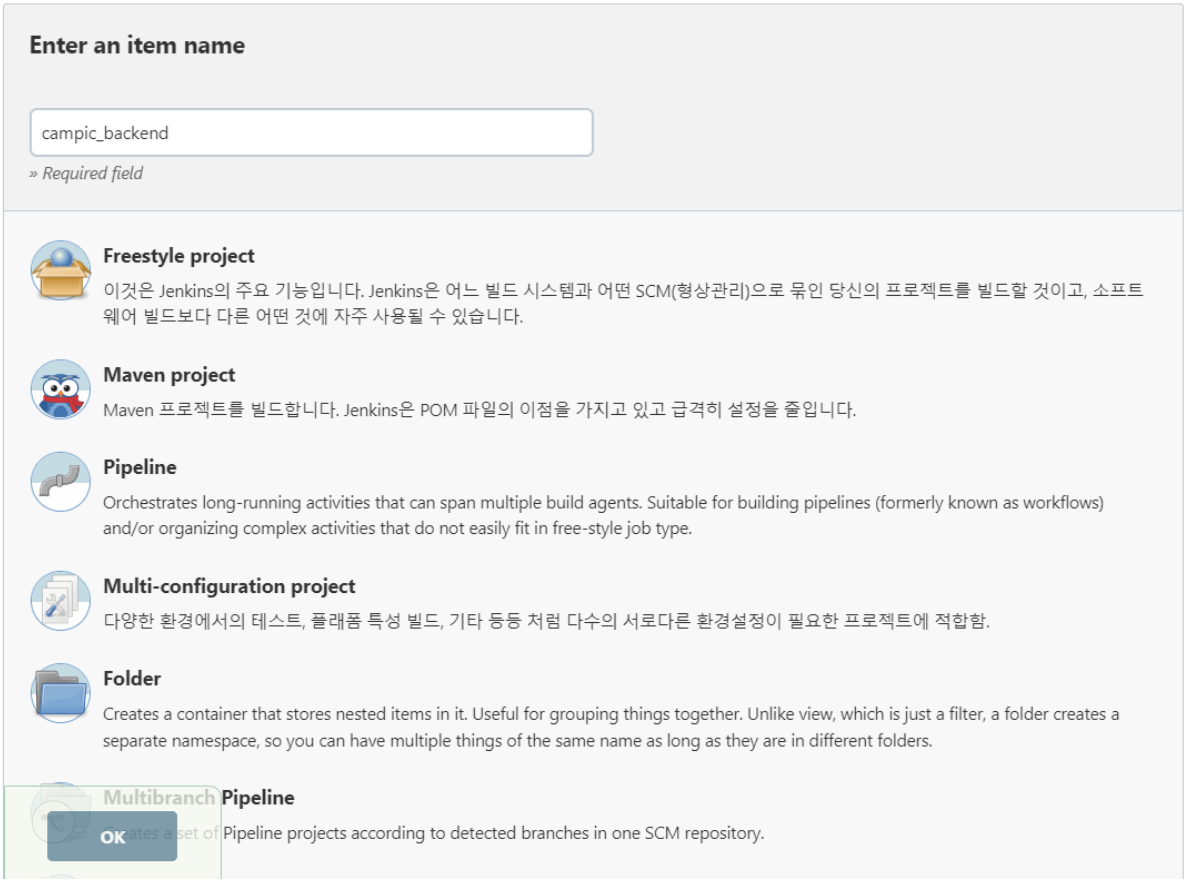


- GitLab과 관련된 설정
  Jenkins 관리 > 시스템 설정 > GitLab



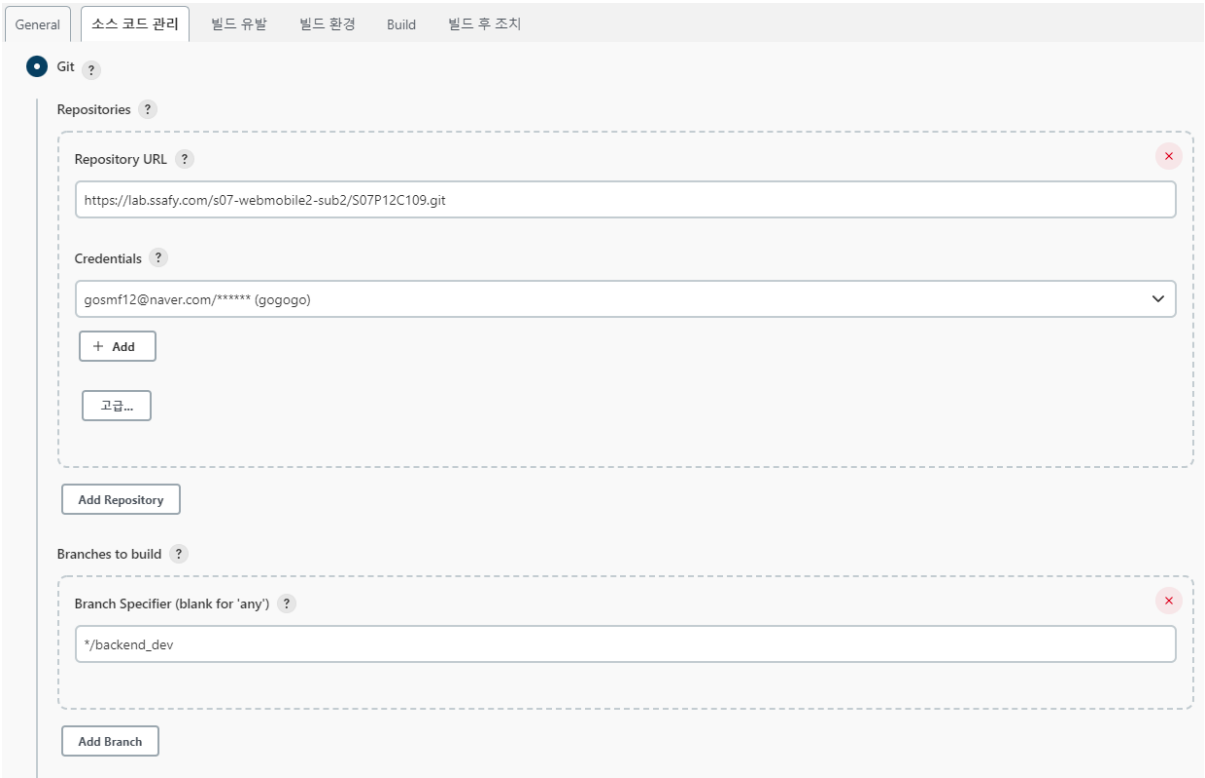
Credentials가 add를 눌러서 GitLab API token을 이용하여 등록한 후 Test Connection을 눌로 Success를 확인 (GitLab API token은 GitLab의 Settings > Access Token에서 발급)

- Jenkins에서 New Item 생성



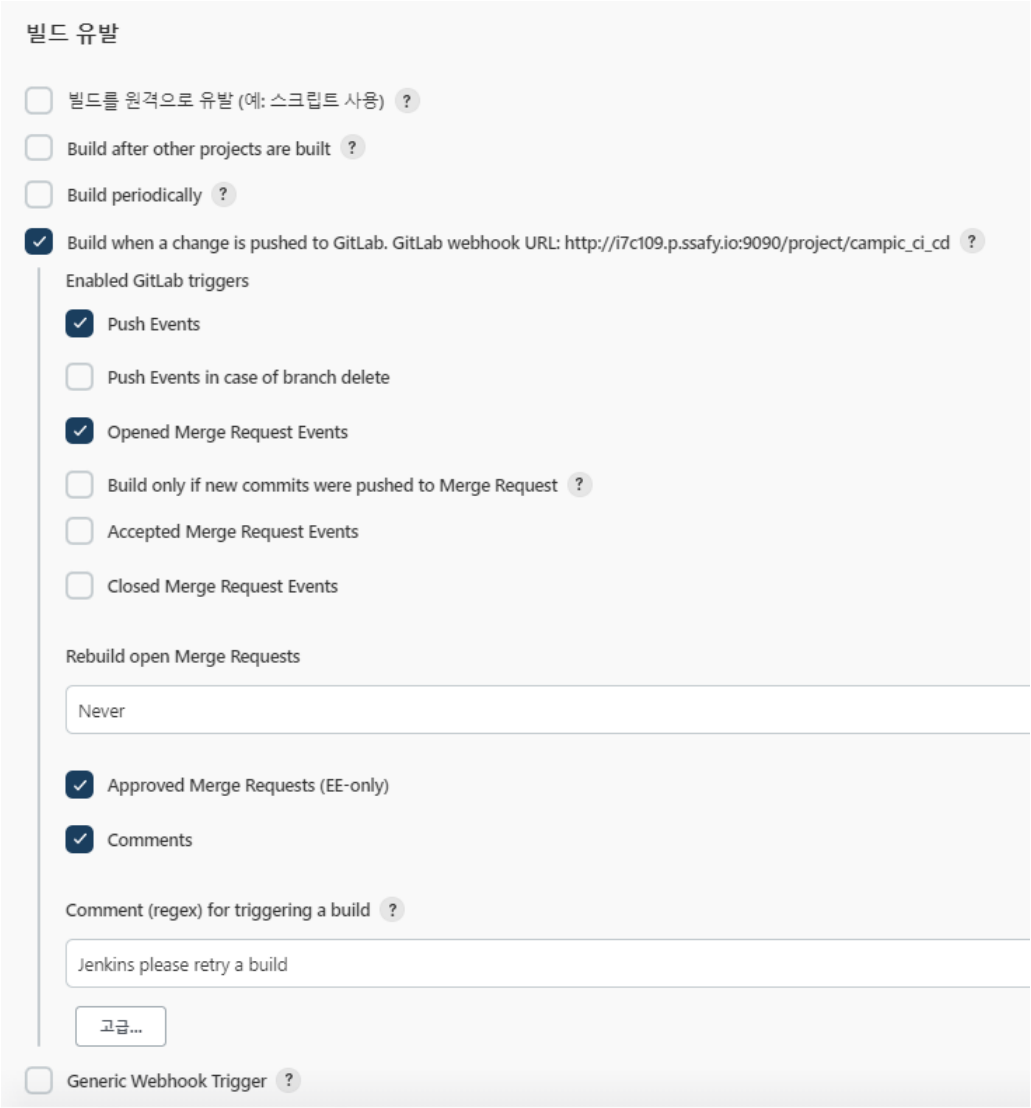
Item name을 기입한 후 Freestyle project 클릭

- 소스 코드 관리 설정하기



git을 pull을 실행할 repository를 등록합니다. 또한 원하는 branch를 설정할 수 있으니 여기서 ‘backend_dev’로 설정.

- 빌드 유발 설정하기

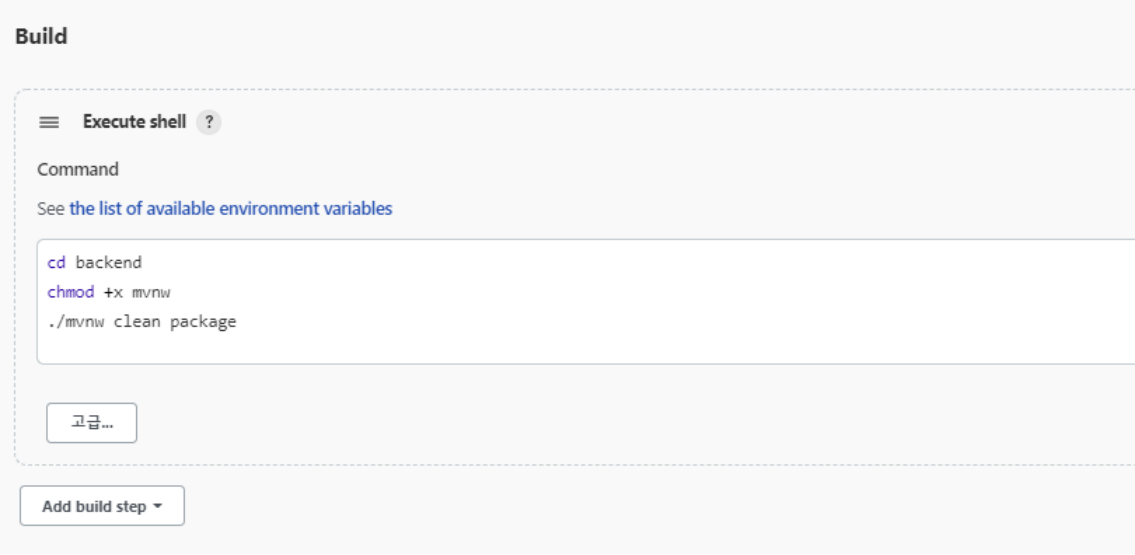


레포지토리 설정이 끝났으니 그 레포지토리에서 push가 됐을 때 빌드 유발을 눌러서 어떤 행동을 할지 설정

Build when a change is pushed to GitLab 선택

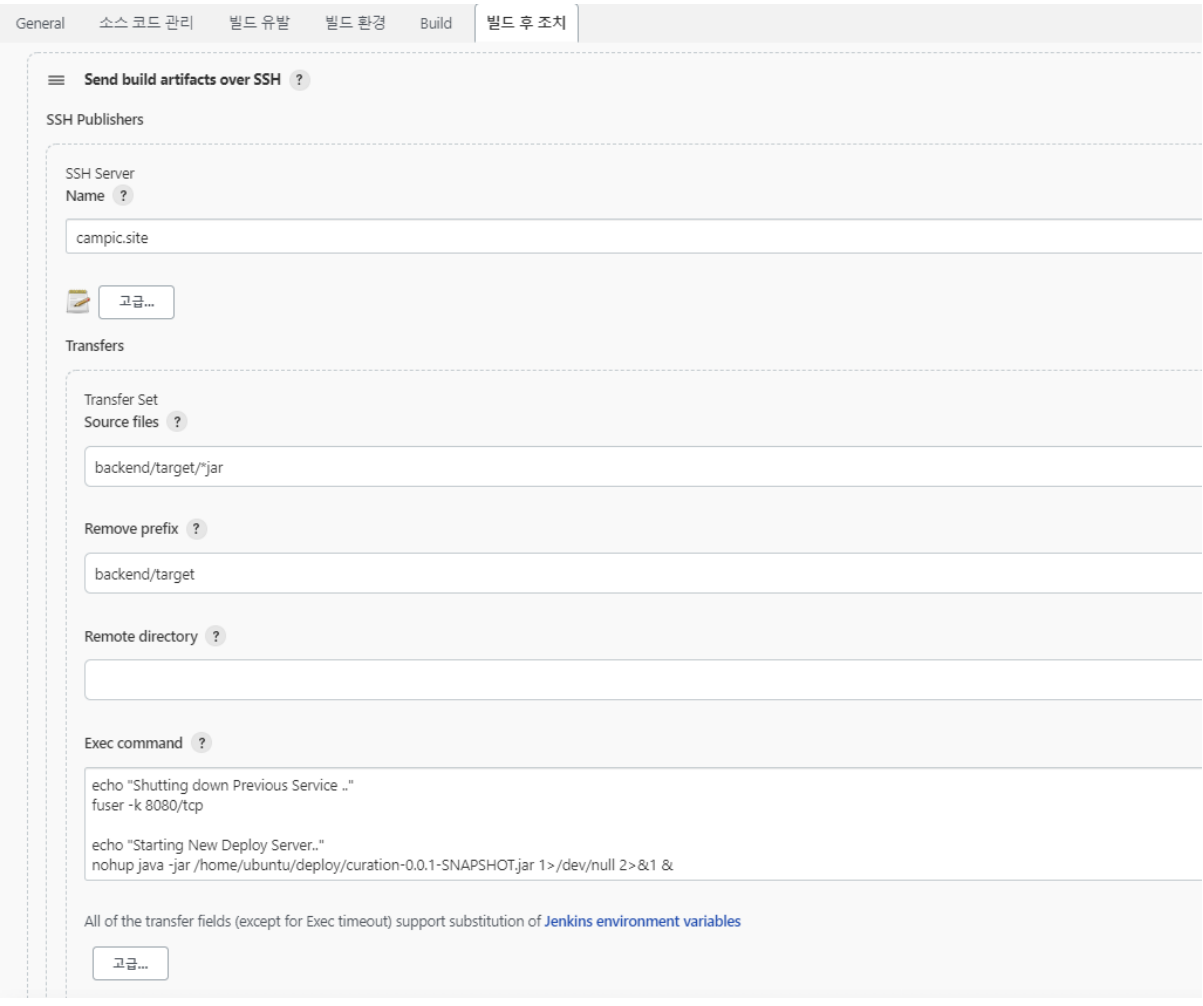
고급 버튼을 눌로 Gitlab의 Secret token을 생성하고 자동으로 push 이벤트를 감지할 수 있도록 webhook을 지정할 예정, gitlab의 setting > webhook으로 이동하여 아까 발급받은 secret token을 입력하고 trigger 부분엔 push 이벤트가 감지될 branch를 입력

- 빌드하기



Excute shell을 선택하고 command 입력 칸에 빌드하기 위해 실행할 코드를 입력

- 빌드 후 조치



빌드 후 자동 배포를 위해서 빌드후 조치에서 Send build artifiacts over SSH를 선택하고 위와 같이 명령어 입력 후 저장.

- 자동 빌드 & 배포 확인
  GitLab에서 push 되었을 때 자동으로 빌드, 배포되는 것을 확인.

프로퍼티 정의

NGINX 설정

```
sudo vim /etc/nginx/sites-available/default

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

        root /home/ubuntu/app/web;

        index index.html index.htm index.nginx-debian.html;

        server_name _;

        location / {
                try_files $uri $uri/ /index.html;
        }
}

server {
        root /home/ubuntu/app/web;

        index index.html index.htm index.nginx-debian.html;
        server_name www.campic.site campic.site; # managed by Certbot


        location / {
                # First attempt to serve request as file, then
                # as directory, then fall back to displaying a 404.
                try_files $uri $uri/ /index.html;
        }


    listen [::]:443 ssl ipv6only=on; # managed by Certbot
    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/campic.site/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/campic.site/privkey.pem; #managed by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot
```

```
}

server {
    if ($host = www.campic.site) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


    if ($host = campic.site) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

        listen 80 ;
        listen [::]:80 ;
    server_name www.campic.site campic.site;
    return 404; # managed by Certbot
}

server {
        listen 80;
        server_name 52.79.231.146;

        return 301 https://campic.site$request_uri;

}

server {
        listen 80;
        listen [::]:80 ;
        server_name i7C109.p.ssafy.io;

        return 301 https://campic.site$request_uri;
}

sudo systemctl restart nginx # nginx 설정 후 재시작
sudo systemctl status nginx # nginx 상태 확인
```

BackEnd Properties

```
server.ssl.key-store=classpath:keystore.12
server.ssl.key-store-type=PKCS12
server.ssl.key-store-password=yaho
server.port=8080
server.http2.enabled=true

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.username=campic
spring.datasource.password=hundredyaho
spring.datasource.url=jdbc:mysql://i7C109.p.ssafy.io:3306/campic

spring.mvc.hiddenmethod.filter.enabled=true

spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
logging.level.org.hibernate.type.descriptor.sql.BasicBinder=trace

spring.jwt.secret=campicC109

spring.mail.host=smtp.gmail.com
spring.mail.port=587
spring.mail.username=hundredyaho@gmail.com
spring.mail.password=pivlioomlhlbgyou
spring.mail.properties.mail.smtp.starttls.enable=true
spring.mail.properties.mail.smtp.auth=true

spring.servlet.multipart.enabled : true
spring.servlet.multipart.location : /home/ubuntu/app/photo/
spring.servlet.multipart.max-request-size : 30MB
spring.servlet.multipart.max-file-size :10MB
```

FrontEnd git ignore

```
# See https://help.github.com/articles/ignoring-files/ for more about ignoring files.

# dependencies
/node_modules
/.pnp
.pnp.js

# testing
/coverage

# production
/build

# misc
.env
.DS_Store
.env.local
.env.development.local
.env.test.local
.env.production.local

npm-debug.log*
yarn-debug.log*
yarn-error.log*
```

외부서비스 키

```
#kakao api 키 모음
REACT_APP_KAKAO_API = 251d71217a822a2f5e875e557daa6bbf
# grant type
REACT_APP_GRANT_TYPE=authorization_code
#카카오 키 모음
REACT_APP_KAKAO_AUTH_URL= https://kauth.kakao.com/oauth/authorize?client_id=de301c8b33f3903d4f52fc9e1bce5338&redirect_uri=https
REACT_APP_KAKAO_CLIENT_ID = de301c8b33f3903d4f52fc9e1bce5338
REACT_APP_KAKAO_REDIRECT_URL = https://campic.site/kakao
REACT_APP_KAKAO_CLIENT_SECRET = YKOYnUdPpKQczjwy4uyyMNuB448UsZJ5
#네이버 키 모음
REACT_APP_NAVER_AUTH_URL= https://nid.naver.com/oauth2.0/authorize?response_type=code&client_id=KrLqvyESB70w3I_OSpeU&state=6oHI
REACT_APP_NAVER_CLIENT_ID = KrLqvyESB70w3I_OSpeU
REACT_APP_NAVER_REDIRECT_URL = https://campic.site/naver
REACT_APP_NAVER_CLIENT_SECRET = 6oHIfgPc4o
#구글 키 모음
REACT_APP_GOOGLE_AUTH_URL=https://accounts.google.com/o/oauth2/v2/auth?client_id=499425273134-646fkjm96vt1urj44qniu2cr43pjhupt.a
REACT_APP_GOOGLE_CLIENT_ID = 499425273134-646fkjm96vt1urj44qniu2cr43pjhupt.apps.googleusercontent.com
REACT_APP_GOOGLE_REDIRECT_URL = https://campic.site/google
REACT_APP_GOOGLE_CLIENT_SECRET = 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
#날씨 api
REACT_APP_OPEN_WEATHER_MAP_API = eb72a7df3e7e80360742e692cfade6ad
```