포팅 매뉴얼

목차

1. 개발 환경

- node.js : 18.16.1
- vscode : 1.81.1
- jdk : 11.0.16
- springboot : 2.7.13
- intellij : community edition 2023.1.4
- docker : 24.0.5
- mysql : 8.0.33
- nginx : 1.18.0 (Ubuntu)
- jenkins : 2.418

2. 설정 파일

- springboot
  - build.gradle

```
plugins {
  id 'java'
  id 'org.springframework.boot' version '2.7.13'
  id 'io.spring.dependency-management' version '1.0.15.RELEASE'
}

group = 'com.ssafy'
version = '0.0.1-SNAPSHOT'

java {
  sourceCompatibility = '11'
}
```

```
configurations {
  compileOnly {
    extendsFrom annotationProcessor
  }
}

repositories {
  mavenCentral()
}

dependencies {
  implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
  implementation 'org.springframework.boot:spring-boot-starter-web'
  implementation 'org.springframework.boot:spring-boot-starter-security'
  implementation 'org.springframework.boot:spring-boot-starter-validation'


  // querydsl
  implementation 'com.querydsl:querydsl-jpa:5.0.0'
  implementation 'com.querydsl:querydsl-core:5.0.0'
  implementation 'com.querydsl:querydsl-collections:5.0.0'
  annotationProcessor "com.querydsl:querydsl-apt:${dependencyManagement.importedProperties['querydsl.version']}:jpa" // querydsl JPAAn
  annotationProcessor "jakarta.annotation:jakarta.annotation-api" // java.lang.NoClassDefFoundError (javax.annotation.Generated) 대응 코
  annotationProcessor "jakarta.persistence:jakarta.persistence-api" // java.lang.NoClassDefFoundError (javax.annotation.Entity) 대응 코드

  // jwt
  implementation 'io.jsonwebtoken:jjwt-api:0.11.2'
  implementation 'io.jsonwebtoken:jjwt-impl:0.11.2'
  implementation 'io.jsonwebtoken:jjwt-jackson:0.11.2'

  // lombok
  compileOnly 'org.projectlombok:lombok'
  annotationProcessor 'org.projectlombok:lombok'
  testCompileOnly 'org.projectlombok:lombok'
  testAnnotationProcessor 'org.projectlombok:lombok'

  // db
  runtimeOnly 'com.h2database:h2:1.4.199'
  runtimeOnly 'com.mysql:mysql-connector-j'

  //spring-email
  implementation group: 'org.springframework.boot', name: 'spring-boot-starter-mail', version: '2.6.3'
  implementation group: 'org.springframework', name: 'spring-context-support', version: '5.3.15'

  //swagger
  implementation group: 'io.springfox', name: 'springfox-swagger-ui', version: '2.9.2'
  implementation group: 'io.springfox', name: 'springfox-swagger2', version: '2.9.2'

  developmentOnly 'org.springframework.boot:spring-boot-devtools'

  testImplementation 'org.springframework.boot:spring-boot-starter-test'
  testImplementation 'org.springframework.security:spring-security-test'
}

test {
  exclude '**/*'
}

tasks.named('test') {
  useJUnitPlatform()
}

// Querydsl 설정부
def generated = 'src/main/generated'

// querydsl QClass 파일 생성 위치를 지정
tasks.withType(JavaCompile) {
  options.getGeneratedSourceOutputDirectory().set(file(generated))
}

// java source set 에 querydsl QClass 위치 추가
sourceSets {
  main.java.srcDirs += [ generated ]
}

// gradle clean 시에 QClass 디렉토리 삭제
clean {
  delete file(generated)
}
```

- springboot
  - application-prod.properties

```
# Server Setting
server.port=8080

# Server encoding
server.servlet.encoding.charset=UTF-8
server.servlet.encoding.force=true

# MySQL Setting
spring.datasource.url=jdbc:mysql://<도메인 url>/mereview?serverTimezone=UTC&useUniCode=yes&characterEncoding=UTF-8
spring.datasource.username= <DB 유저>
spring.datasource.password= <DB 비밀번호>
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect

# jpa
spring.jpa.hibernate.ddl-auto=none
spring.jpa.properties.hibernate.default_batch_fetch_size=100
spring.jpa.open-in-view=true
spring.jpa.properties.hibernate.format_sql=true
spring.jpa.defer-datasource-initialization=true
# reserved words error config (add backticks)
spring.jpa.properties.hibernate.auto_quote_keyword=true
spring.jpa.properties.hibernate.globally_quoted_identifieres=true

# swagger error config
spring.mvc.pathmatch.matching-strategy=ant_path_matcher

# log
logging.level.org.hibernate.SQL=trace
logging.level.com.ssafy.mereview=info
logging.file=/var/log/mereviewlog.log

# Upload File
spring.servlet.multipart.max-file-size=100MB
spring.servlet.multipart.max-request-size=100MB
spring.servlet.multipart.enabled=true

# File Path setting
file.dir=/var/images/

# jwt token
app.jwt.secret=secretKeysecretKeysecretKeysecretKeysecretKeysecretKeysecretKeysecretKeysecretKey
app.jwt.accessExpirationMs : 300000000
app.jwt.refreshExpirationMs : 600000000
app.jwt.emailCodeExpirationMs : 450000000

# spring mail
spring.mail.host=smtp.gmail.com
spring.mail.port=587
spring.mail.username=<user email>
spring.mail.password=<user password>
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.starttls.enable=true
```

- mysql
  - MYSQL_USER=<secret>
  - MYSQL_DATABASE=”mereview”
  - MYSQL_ROOT_PASSWORD=<secret>
  - MYSQL_PASSWORD=<secret>

3. 외부 서비스 정보

- tmdb api key : tmdb회원가입 후 api key 발급받아 사용
- spring mail : 구글 계정 관리에서 보안 탭의 앱 비밀번호 클릭 → 앱 비밀번호를 메일 / Window컴퓨터로 설정 후 생성 버튼 클릭

4. Nginx & Let’s Encrypt & Certbot 설정

- Nginx 설치

```
sudo apt install nginx
```

- Let's Encrypt 설치

```
sudo apt-get install letsencrypt
```

- Certbot 설치

```
sudo apt-get install certbot python3-certbot-nginx
```

- Certbot 동작

```
sudo certbot --nginx
```

이메일 입력

약관 동의 - Y

이메일 발송 동의 - Y or N

도메인 입력 i9c211.p.ssafy.io

- nginx.conf 파일 수정

```
##
# You should look at the following URL's in order to grasp a solid understanding
# of Nginx configuration files in order to fully unleash the power of Nginx.
# https://www.nginx.com/resources/wiki/start/
# https://www.nginx.com/resources/wiki/start/topics/tutorials/config_pitfalls/
# https://wiki.debian.org/Nginx/DirectoryStructure
#
# In most cases, administrators will remove this file from sites-enabled/ and
# leave it as reference inside of sites-available where it will continue to be
# updated by the nginx packaging team.
#
# This file will automatically load configuration files provided by other
# applications, such as Drupal or Wordpress. These applications will be made
# available underneath a path with that package name, such as /drupal8.
#
# Please see /usr/share/doc/nginx-doc/examples/ for more detailed examples.
##

# Default server configuration
#
server {
        listen 80 default_server;
        listen [::]:80 default_server;

        # SSL configuration
        #
        # listen 443 ssl default_server;
        # listen [::]:443 ssl default_server;
        #
        # Note: You should disable gzip for SSL traffic.
        # See: https://bugs.debian.org/773332
        #
        # Read up on ssl_ciphers to ensure a secure configuration.
        # See: https://bugs.debian.org/765782
        #
        # Self signed certs generated by the ssl-cert package
        # Don't use them in a production server!
        #
        # include snippets/snakeoil.conf;

         root /var/www/html;

        # Add index.php to the list if you are using PHP
        index index.html index.htm index.nginx-debian.html;

        server_name _;

        location / {
```

```
                # First attempt to serve request as file, then
                # as directory, then fall back to displaying a 404.
                try_files $uri $uri/ =404;
        }

        # pass PHP scripts to FastCGI server
        #
        #location ~ \.php$ {
        #       include snippets/fastcgi-php.conf;
        #
        #       # With php-fpm (or other unix sockets):
        #       fastcgi_pass unix:/var/run/php/php7.4-fpm.sock;
        #       # With php-cgi (or other tcp sockets):
        #       fastcgi_pass 127.0.0.1:9000;
        #}

        # deny access to .htaccess files, if Apache's document root
        # concurs with nginx's one
        #
        #location ~ /\.ht {
        #       deny all;
        #}
}

# Virtual Host configuration for example.com
#
# You can move that to a different file under sites-available/ and symlink that
# to sites-enabled/ to enable it.
#
#server {
#       listen 80;
#       listen [::]:80;
#
#       server_name example.com;
#
#       root /var/www/example.com;
#       index index.html;
#
#       location / {
#               try_files $uri $uri/ =404;
#       }
#}
#
#

server {

        # SSL configuration
        #
        # listen 443 ssl default_server;
        # listen [::]:443 ssl default_server;
        #
        # Note: You should disable gzip for SSL traffic.
        # See: https://bugs.debian.org/773332
        #
        # Read up on ssl_ciphers to ensure a secure configuration.
        # See: https://bugs.debian.org/765782
        #
        # Self signed certs generated by the ssl-cert package
        # Don't use them in a production server!
        #
        # include snippets/snakeoil.conf;

        root /var/www/html;

        client_max_body_size 100M;

        # Add index.php to the list if you are using PHP
        index index.html index.htm index.nginx-debian.html;
        server_name i9c211.p.ssafy.io; # managed by Certbot


        location / {
                # First attempt to serve request as file, then
                # as directory, then fall back to displaying a 404.
                proxy_pass http://172.25.1.3:3000;
        }

        location /api {
                proxy_pass http://172.27.1.3:8080;
        }

        location ~ ^/(swagger|webjars|configuration|swagger-resources|v2|csrf) {
               proxy_pass http://172.27.1.3:8080;
               proxy_set_header Host $host;
               proxy_set_header X-Real-IP $remote_addr;
               proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
```

```
                proxy_set_header X-Forwarded-Proto $scheme;
        }

        # pass PHP scripts to FastCGI server
        #
        #location ~ \.php$ {
        #       include snippets/fastcgi-php.conf;
        #
        #       # With php-fpm (or other unix sockets):
        #       fastcgi_pass unix:/var/run/php/php7.4-fpm.sock;
        #       # With php-cgi (or other tcp sockets):
        #       fastcgi_pass 127.0.0.1:9000;
        #}

        # deny access to .htaccess files, if Apache's document root
        # concurs with nginx's one
        #
        #location ~ /\.ht {
        #       deny all;
        #}


    listen [::]:443 ssl ipv6only=on; # managed by Certbot
    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/i9c211.p.ssafy.io/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/i9c211.p.ssafy.io/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 ($host = i9c211.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


        listen 80 ;
        listen [::]:80 ;
    server_name i9c211.p.ssafy.io;
    return 404; # managed by Certbot


}
```

5. docker 설치

```
sudo apt-get update
sudo apt-get install ca-certificates curl gnupg

sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
sudo chmod a+r /etc/apt/keyrings/docker.gpg

echo \
  "deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \
  "$(. /etc/os-release && echo "$VERSION_CODENAME")" stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update

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

6. react docker, docker-compose 설정

- react.dockerfile

```
FROM node:18

RUN npm install -g serve

RUN mkdir ./build

COPY ./build ./build

ENTRYPOINT ["serve", "-s", "build"]
```

- docker-compose.yml

```
version: "3.3"

services:
  react:
    build:
      context: .
      dockerfile: react.dockerfile
    container_name: "react"

    expose:
      - "3000"

    networks:
      default_bridge:
        ipv4_address: 172.25.1.3

networks:
  default_bridge:
    ipam:
      driver: default
      config:
        - subnet: 172.25.1.0/24
```

7. springboot docker, docker-compose 설정

- springboot.dockerfile

```
FROM openjdk:11-jdk-slim

ARG JAR_FILE=build/*.jar

COPY ${JAR_FILE} app.jar

EXPOSE 8080

ENTRYPOINT [ "java", "-jar", "/app.jar" ]
```

- docker-compose.yml

```
version: "3.3"

services:

  springboot:
    container_name: "springboot"

    build:
      context: "."
      dockerfile: "springboot.dockerfile"

    volumes:
      - "./images:/var/images"
      - "./logs:/var/log"

    expose:
      - "8080"

    networks:
      default_bridge:
        ipv4_address: 172.27.1.3


networks:
  default_bridge:
    ipam:
      driver: default
      config:
        - subnet: 172.27.1.0/24
```

8. jenkins & mysql docker, docker-compose 설정

- jenkins.dockerfile

```
FROM jenkins/jenkins:latest
USER root

RUN apt-get update && apt-get install sudo

RUN apt-get update && apt-get install -y vim

# 공식문서 참조
RUN apt-get update && apt-get install -y lsb-release

RUN curl -fsSLo /usr/share/keyrings/docker-archive-keyring.asc \
  https://download.docker.com/linux/debian/gpg

RUN echo "deb [arch=$(dpkg --print-architecture) \
  signed-by=/usr/share/keyrings/docker-archive-keyring.asc] \
  https://download.docker.com/linux/debian \
  $(lsb_release -cs) stable" > /etc/apt/sources.list.d/docker.list

RUN apt-get update && apt-get install -y docker-ce-cli

RUN jenkins-plugin-cli --plugins "blueocean docker-workflow"

#nodejs 설치
ENV NVM_DIR /root/.nvm

RUN curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.35.3/install.sh | bash && \
    . $NVM_DIR/nvm.sh && \
    export NVM_DIR="$HOME/.nvm" && \
    [ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh" && \
    nvm install 18 && \
    n=$(which node) && n=${n%/bin/node} && chmod -R 755 $n/bin/* && cp -r $n/* /usr/local

ENV DEBIAN_FRONTEND=noninteractive

#java11 설치
RUN apt-get update && apt-get install -y openjdk-11-jdk
```

- docker-compose.yml

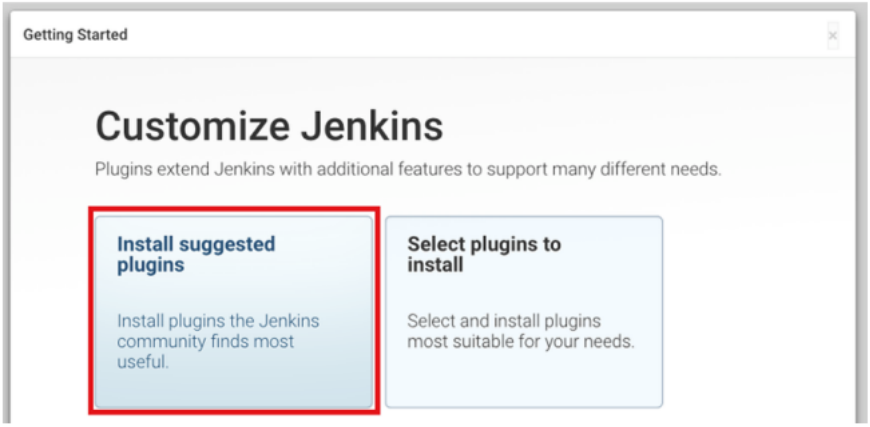
```
version: "3.3"

services:

  jenkins:
    container_name: "jenkins"

    build:
      context: "./dockerfiles"
      dockerfile: "jenkins.dockerfile"

    user: "root"
    privileged: true

    restart: "always"

    ports:
      - "8888:8080"
      - "50000:50000"



    volumes:
      - "/home/ubuntu/mereview/default/jenkins_home:/var/jenkins_home"
      #- "/var/run/docker.sock:/var/run/docker.sock"

    environment:
      - "TZ=Asia/Seoul"

  mysql:
    container_name: "mysql"

    image: "mysql:8.0.33"

    restart: "always"

    ports:
      - "3306:3306"
```

```
    volumes:
      - "/home/ubuntu/mereview/default/mysql_home:/var/lib/mysql"

    command:
      - "--character-set-server=utf8mb4"
      - "--collation-server=utf8mb4_unicode_ci"
      - "--skip-character-set-client-handshake"

    env_file:
      - "./env/mysql.env"
```

9. jenkins 빌드 & 배포 자동화

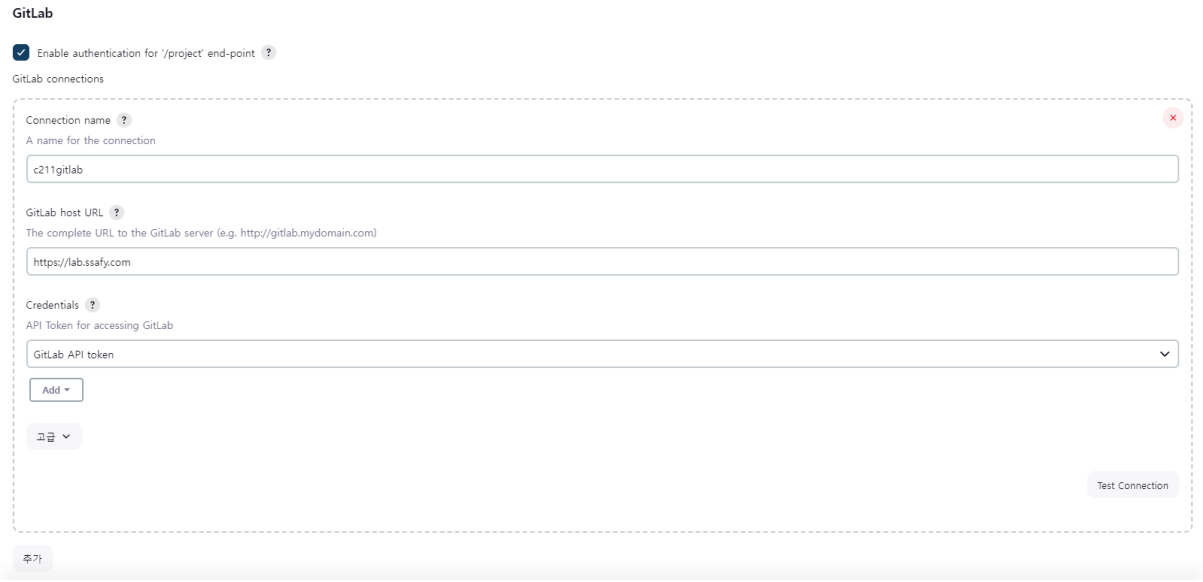
- Administrator password 입력 (docker logs {container-names})

- plugin 설치

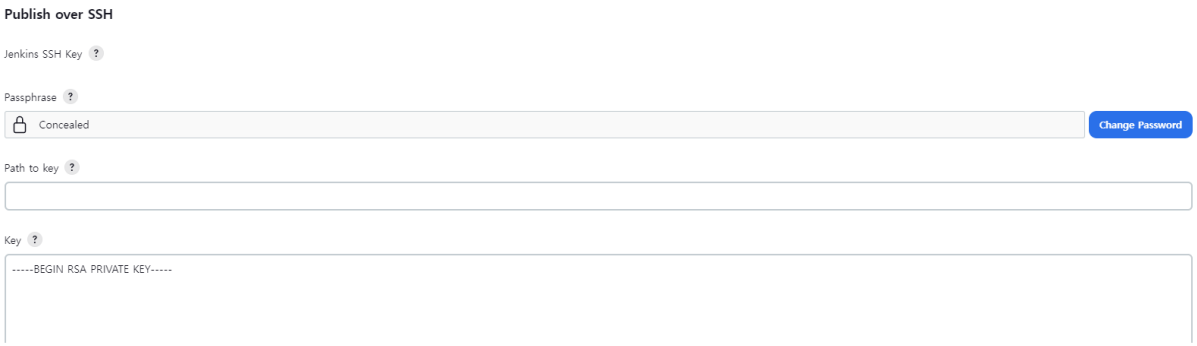


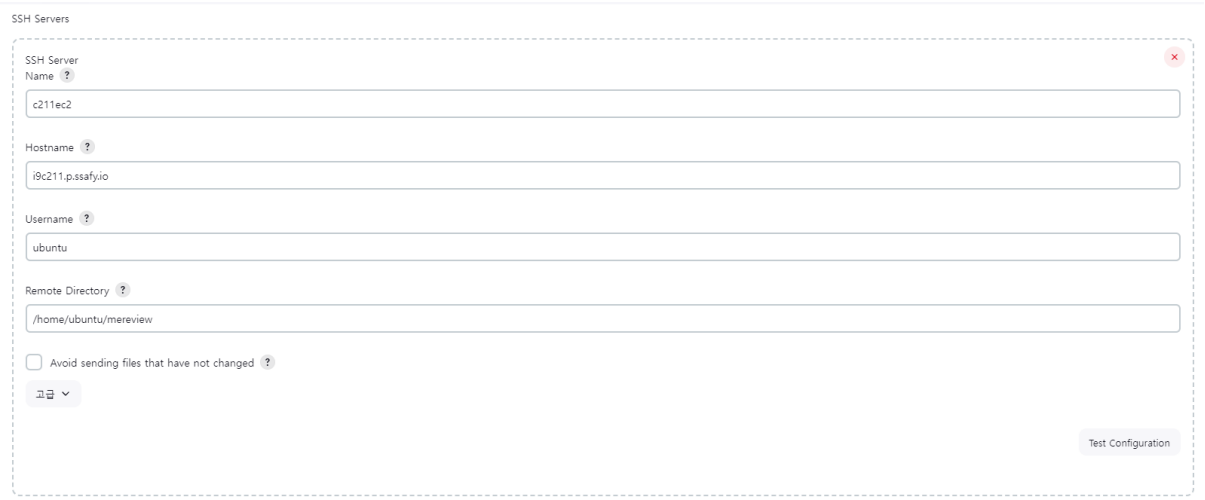
- 추가로 gitlab plugin & publish over ssh 설치

- gitlab repository와 연결
- gitlab repository 로 이동, 그 후 settings - Access Tokens 에서 키 발급 필요



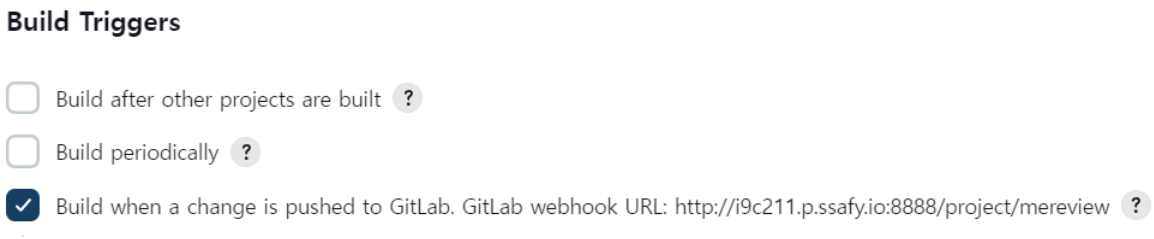
- ec2 서버와 연결
- Key에는 발급받은 .pem 키값을 넣기



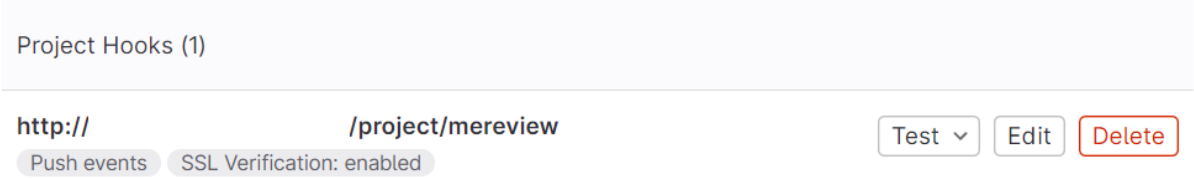


- new item 에서 pipeline 선택

- build trigger 설정



- build trigger - 고급 - secret token - generate 클릭후 값 복사
- 그 후, gitlab - settings - webhooks로 이동 - webhooks 생성



- 다시 jenkins item으로 이동 pipeline - pipeline script 작성

```
node {
    stage('Build') {
        cleanWs()
       checkout scmGit(branches: [[name: '*/dev']], extensions: [], userRemoteConfigs: [[credentialsId: 'ssafygit', url: 'https://lab.

        dir("mereview-client") {
            sh "pwd"
            sh "sudo npm install"
            sh "sudo npm install sass -g"
            sh "sudo sass ./src/styles/scss:./src/styles/css"
            sh "sudo CI= npm run build"
        }

        dir('mereview-server') {
            // some block
            sh "pwd"
            sh "sudo chmod +x gradlew"
            withGradle {
                // some block
                sh './gradlew clean'
                sh './gradlew bootJar'
            }
        }
    }
    stage('Test') {
        //
    }
    stage('Deploy') {
        dir("mereview-client"){
      sshPublisher(publishers: [sshPublisherDesc(configName: 'c211ec2', transfers: [sshTransfer(cleanRemote: false, excludes: '', exec
sudo docker-compose down
sudo docker image rm fe-react
sudo docker-compose up -d''', execTimeout: 120000, flatten: false, makeEmptyDirs: false, noDefaultExcludes: false, patternSeparator: '
        }

      dir('mereview-server') {
       sshPublisher(publishers: [sshPublisherDesc(configName: 'c211ec2', transfers: [sshTransfer(cleanRemote: false, excludes: '', exe
sudo docker-compose down
sudo docker image rm be-springboot
sudo docker-compose up -d''', execTimeout: 120000, flatten: false, makeEmptyDirs: false, noDefaultExcludes: false, patternSeparator: '
        }

    }
}
```