

# 포팅매뉴얼

## ▼ 1. 프로젝트 기술 스택

### 1-1. Tools

- Notion
- Jira
- Git
- IntelliJ Community Edition v2021.3.3
- Spring Tool Suite v3.9.14.RELEASE
- VS Code v1.69.2
- GitHub Desktop
- Postman
- MobaXterm
- MySQL WorkBench 8.0 CE

### 1-2. 기술 스택

#### 1) FrontEnd

- React v18.2.0
- Redux v4.2.0
- Cloudinary 1.5.0
- node.js v16.16.0
- Javascript
- HTML / CSS

#### 2) BackEnd

- Java v1.8(zulu8)
- SpringBoot v2.7.1

#### 3) DataBase

- MySQL v8.0.30-0ubuntu0.20.04.2

#### 4) Server

- Docker v20.10.17
- Jenkins v2.346.3
- Nginx v1.18.0(Ubuntu)

#### 5) Hadoop

- MapReduce v3.2.1
- Sqoop v1.4.7

## ▼ 2. EC2 설정

### 초기 설정

```
sudo apt update
sudo apt upgrade
sudo apt install build-essential
```

### java 설치

```
# 설치
sudo apt-get install openjdk-8-jdk
# 버전확인
java -version
```

### timezone 설정

```
sudo rm /etc/localtime
sudo ln -s /usr/share/zoneinfo/Asia/Seoul /etc/localtime
```

### hostname 설정

```
sudo hostnamectl set-hostname webterview.localdomain
sudo vi /etc/hosts
# 맨 윗줄을 변경한다.
# 127.0.0.1 webterview.localdomain webterview localhost4 localhost4.localdomain4
```

## ▼ 3. MySQL 설치 및 워크벤치 연결

### 3-1. EC2에서 MySQL 설치

#### 1) 설치

```
# 설치
sudo apt install mysql-server
# root계정 접속
sudo mysql -u root -p
```

#### 2) 계정 생성 및 권한 부여

```
CREATE USER 'wayg'@'%' IDENTIFIED BY 'lllkhC202*';
FLUSH PRIVILEGES;
```

```
show grants for 'wayg'@'%';
grant all privileges on *.* to 'wayg'@'%';
```

### 3) 외부접속 허용

```
sudo vi /etc/mysql/mysql.conf.d/mysql.cnf
# bind-address=0.0.0.0로 변경한다.
```

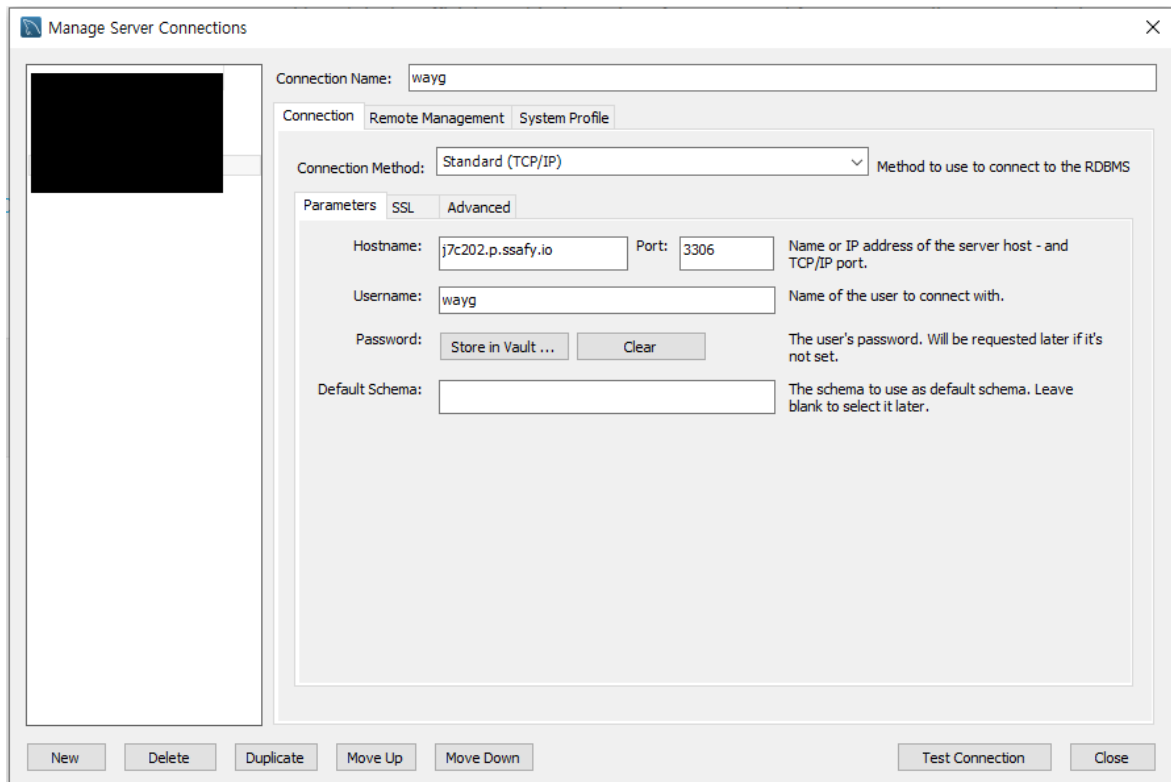
```
# 포트 허용
sudo ufw allow out 3306/tcp
sudo ufw allow in 3306/tcp
# mysql 재시작
sudo service mysql restart
```

## 3-2. Workbench 연결 및 DDL 실행

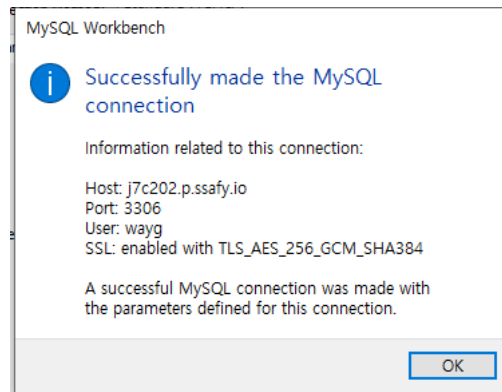
### 1) Workbench 연결

새 connection을 설정한다.

HostName에 도메인, Username과 Password에는 위에서 설정한 webtview와 llkjhC205\*를 입력한다.



Test Connection에서 이 창이 떴다면 잘 연결된 것이다.



## 2) DDL script 실행

git에 있는 wayg/src/main/resources/sql/webtview.sql 파일을 워크벤치에서 전부 실행시킨다.

# ▼ 4. Docker, Docker Compose 설치

## 4-1. Docker 설치

### 1) 기본 설정, 사전 설치

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

### 2) 자동 설치 스크립트 활용

```
sudo wget -qO- https://get.docker.com/ | sh
```

### 3) Docker 서비스 실행하기 및 부팅 시 자동 실행 설정

```
sudo systemctl start docker
sudo systemctl enable docker
```

### 4) Docker 그룹에 현재 계정 추가

```
sudo usermod -aG docker ${USER} # ${USER} 대신 ubuntu를 넣어 진행했다.
sudo systemctl restart docker
```

## 4-2. Docker Compose 설치

### 1) 설치

```
$ sudo curl -L "https://github.com/docker/compose/releases/download/1.24.1/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

### 2) 권한 설정

```
sudo chmod +x /usr/local/bin/docker-compose
```

### 3) 심볼릭 링크 설정

```
sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose
```

## 4-3. Docker 명령어

```
# 현재 실행중인 컨테이너
docker ps
# 모든 컨테이너
docker ps -a
# 이미지 목록
docker images
# 컨테이너 중지
docker kill [컨테이너이름|컨테이너ID]
# 컨테이너 시작
docker start [컨테이너이름|컨테이너ID]
# 컨테이너 삭제
docker rm [컨테이너이름|컨테이너ID]
# 이미지 삭제
docker rmi [이미지이름|이미지ID]
# 실행중인 컨테이너 shell 환경으로 접속
docker exec -it [컨테이너이름|컨테이너ID] bash
# 컨테이너 로그
docker logs -f [컨테이너이름|컨테이너ID]
```

## ▼ 5. Nginx 설치 및 설정

### 5-1. Nginx 설치

```
sudo apt update
sudo apt install nginx
```

### 5-2. SSL 인증서

#### 1) certbot 설치

```
sudo add-apt-repository ppa:certbot/certbot
sudo apt install python-certbot-nginx
```

#### 2) SSL 인증서 가져오기

```
# nginx 플러그인을 사용한다.
sudo certbot --nginx -d j7c202.p.ssafy.io
```

차례대로 이메일, 서비스 약관 동의절차를 수행한다.

```
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Plugins selected: Authenticator nginx, Installer nginx
Enter email address (used for urgent renewal and security notices) (Enter 'c' to
cancel): waygc@gmail.com

- - - - -
Please read the Terms of Service at
https://letsencrypt.org/documents/LE-SA-v1.2-November-15-2017.pdf. You must
agree in order to register with the ACME server at
https://acme-v02.api.letsencrypt.org/directory
- - - - -
(A)gree/(C)ancel: A

- - - - -
Would you be willing to share your email address with the Electronic Frontier
```

```
Foundation, a founding partner of the Let's Encrypt project and the non-profit
organization that develops Certbot? We'd like to send you email about our work
encrypting the web, EFF news, campaigns, and ways to support digital freedom.
- - - - -
(Y)es/(N)o: Y
```

위 절차가 끝나면 https를 어떻게 설정할지 묻는데, 2를 선택해서 모든 http 연결을 https로 리다이렉팅 시키도록 한다.

```
Obtaining a new certificate
Performing the following challenges:
http-01 challenge for example.com
Waiting for verification...
Cleaning up challenges
Deploying Certificate to VirtualHost /etc/nginx/sites-enabled/example.com

Please choose whether or not to redirect HTTP traffic to HTTPS, removing HTTP access.
- - - - -
1: No redirect - Make no further changes to the webserver configuration.
2: Redirect - Make all requests redirect to secure HTTPS access. Choose this for
new sites, or if you're confident your site works on HTTPS. You can undo this
change by editing your web server's configuration.
- - - - -
Select the appropriate number [1-2] then [enter] (press 'c' to cancel): 2
```

### 5-3. default 설정

```
server_name j7c202.p.ssafy.io; # 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/ =404;
    charset utf-8;
    proxy_redirect off;
    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;
    proxy_set_header X-NginX-Proxy true;

    client_max_body_size 10M;

    proxy_pass http://localhost:3000/;
}

location /api {
    error_page 405 =200 $uri;
    proxy_redirect off;
    charset utf-8;
    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;
    proxy_set_header X-NginX-Proxy true;
    client_max_body_size 10M;
    proxy_pass http://localhost:8080/api;
}

listen [::]:443 ssl ipv6only=on; # managed by Certbot
listen 443 ssl; # managed by Certbot
ssl_certificate /etc/letsencrypt/live/j7c202.p.ssafy.io/fullchain.pem; # managed by Certbot
ssl_certificate_key /etc/letsencrypt/live/j7c202.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 ($host = j7c202.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

    listen 80 ;
    listen [::]:80 ;
    server_name j7c202.p.ssafy.io;
    return 404; # managed by Certbot
}
```

## ▼ 6. Jenkins 설치 및 설정

### 6-1. Docker로 Jenkins 설치

#### 1) jenkins 이미지 파일 내려받기

```
docker pull jenkins/jenkins:lts
```

#### 2) jenkins 이미지 컨테이너로 실행

ubuntu와 젠킨스 컨테이너 볼륨을 연결한다.

- ubuntu의 /jenkins 와 컨테이너의 /var/jenkins
- ubuntu의 /home/ubuntu/.ssh 와 컨테이너의 /root/.ssh
- ubuntu의 /var/run/docker.sock 와 컨테이너의 /var/run/docker.sock (바깥에 설치된 docker를 jenkins 속 도커에서도 사용할 수 있도록 한다.)

이름은 jenkins, 계정은 root로 한다.

```
docker run -d -p 9000:8080 -p 50000:50000 -v /jenkins:/var/jenkins -v /home/ubuntu/.ssh:/root/.ssh -v /var/run/docker.sock:/var/run/docker.sock jenkins/jenkins:lts
```

### 6-2. jenkins 접속

#### 1) <http://j7c202.p.ssafy.io:9000> 로 접속한다.

#### 2) 암호입력

```
# 방법 1 - 로그로 암호를 알아낸다.
docker logs jenkins
# 방법 2 - 젠킨스 컨테이너에 접속해 암호파일을 확인한다.
docker exec -it jenkins bash
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

#### 3) 플러그인 설치 - Install suggested plugins

#### 4) 계정 생성

### 6-3. GitLab과 연동

#### 1) ssh 키 생성

전부 enter를 입력한다.

```
ssh-keygen
```

#### 2) GitLab Deploy key 등록

id\_rsa.pub에 있는 public key 값을 복사한다. 'ssh-rsa'로 시작해서 이메일주소로 끝나는 모든 것을 복사해야 한다.

```
cat /home/ubuntu/.ssh/id_rsa.pub
```

Preferences → SSH Keys에서 새 SSH key를 등록한다. key에 복사한 값을 붙여넣고, Expiration date를 설정한다. title은 아무거나 해도 된다.

User Settings > SSH Keys

Q Search page

## SSH Keys

SSH keys allow you to establish a secure connection between your computer and GitLab.

### Add an SSH key

Add an SSH key for secure access to GitLab. [Learn more.](#)

#### Key

ssh-rsa  
[REDACTED]  
ubuntu@[REDACTED].localdomain

Begins with 'ssh-rsa', 'ssh-dss', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', 'ssh-ed25519', 'sk-ecdsa-sha2-nistp256@openssh.com', or 'sk-ssh-ed25519@openssh.com'.

#### Title

jenkins

Key titles are publicly visible.

#### Expiration date

2022-09-30

Key becomes invalid on this date.

Add key

### 3) jenkins credential 등록

id\_rsa에 있는 private key 값을 복사한다. -----BEGIN OPENSSH PRIVATE KEY----- 부터 -----END OPENSSH PRIVATE KEY-----까지 전부 복사해야 한다.

```
cat /home/ubuntu/.ssh/id_rsa
```

[http://j7c202.p.ssafy.io:9000/credentials/store/system/domain/\\_/](http://j7c202.p.ssafy.io:9000/credentials/store/system/domain/_/) 에 접속해서 Add Credentials를 클릭한다.

Kind는 SSH Username with private key로 설정한다.

ID는 파이프라인 스크립트 작성 시 credentialsId로 사용될 이름을 쓴다.

Username은 root, Private Key는 위에서 복사한 값을 전부 붙여넣는다.



## New credentials

Kind

SSH Username with private key



Scope ?

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



ID ?

gitlab\_ssh

Description ?

Username

root

☐ Treat username as secret ?

Private Key

☒ Enter directly

Key

Enter New Secret Below

-----BEGIN OPENSSH PRIVATE KEY-----

-----END OPENSSH PRIVATE KEY-----

## 6-4. 플러그인 추가 설치

젠킨스 관리 > Plugin Manager > 설치가능 목록에서 gitlab을 검색해 GitLab, Generic Webhook Trigger, Gitlab API, GitLab Authentication을 설치한다.

docker를 검색해서 Docker, Docker Commons, Docker Pipeline, Docker API 를 설치한다.

Generic Webhook Trigger Plugin을 설치한다.

## 6-5. jenkins 컨테이너 내부에 docker 설치

### 1) sudo, vi, wget 설치

```
# 컨테이너 접속
docker exec -it jenkins bash
```

```
apt update
apt install vim
apt install sudo
apt install wget
```

## 2) docker 설치

```
sudo apt update
sudo apt install apt-transport-https ca-certificates curl software-properties-common
sudo wget -qO- https://get.docker.com/ | sh
sudo systemctl start docker
sudo systemctl enable docker
```

## 3) docker.sock 권한변경

```
sudo chmod 666 /var/run/docker.sock
```

## 6-6. jenkins 아이템 생성

1) new Item > pipeline을 선택한다.

### 2) Build Triggers 탭

“Build when a change is pushed to GitLab. GitLab webhook”을 선택한다. 여기서 맨 끝 url을 복사해놓는다.

General **Build Triggers** Advanced Project Options Pipeline

☐ Build after other projects are built ?

☐ Build periodically ?

☒ Build when a change is pushed to GitLab. GitLab webhook URL: <http://17c205.p.ssafy.io:8080/project/webview> ?

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

Never

☒ Approved Merge Requests (EE-only)

☒ Comments

Comment (regex) for triggering a build ?

Jenkins please retry a build

고급...

고급버튼 눌러서 아래의 Generate를 클릭한다. 생성된 Secret token을 복사한다.

☒ Enable [ci-skip]  
☒ Ignore WIP Merge Requests

Labels that forces builds if they are added (comma-separated)

☒ Set build description to build cause (eg. Merge request or Git Push)  
☐ Build on successful pipeline events

Pending build name for pipeline ?

☐ Cancel pending merge request builds on update

Allowed branches

☒ Allow all branches to trigger this job ?  
☐ Filter branches by name ?  
☐ Filter branches by regex ?  
☐ Filter merge request by label

Secret token ?

Generate

### 3) Pipeline 탭

Pipeline script from SCM을 선택한다.

Repository URL을 작성하고, gitlab의 Credential을 선택한다.

Branches to build에 빌드하고싶은 타겟브랜치를 설정한다.

Script Path에 JenkinsFile을 적는다. git repository 최상단에 JenkinsFile이 있어야 한다.

Definition

Pipeline script from SCM

SCM ?

Git

Repositories ?

Repository URL ?

https://lab.ssfy.com/ .git

Credentials ?

+ Add

고급...

Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ?

\*/develop

Add Branch

Repository browser ?

(자동)

Additional Behaviours

Add

Script Path ?

Jenkinsfile

## 6-7. git repository와 webhook 연결

7-6의 2)에서 저장한 url과 secret token을 붙여넣는다.

push event를 선택한다. 아래에 브랜치이름을 입력하면 그 브랜치에서 발생한 push event만 빌드가 시작된다. 7-6에서 정한 브랜치와 맞추는 것이 좋다. 아무것도 쓰지 않는다면 모든 브랜치의 push event마다 빌드하게 된다.

s07-webmobile1-sub2 > S07P12C205 > Webhook Settings > Webhook

Q Search page

### Webhook

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 it contains one or more special characters.

Secret token

Used to validate received payloads. Sent with the request in the `X-Gitlab-Token` HTTP header.

Trigger

☒ Push events

develop

Push to the repository.

☐ Tag push events

A new tag is pushed to the repository.

☐ Comments

## ▼ 7. 로컬 빌드 방법

### 7-1. 프론트엔드

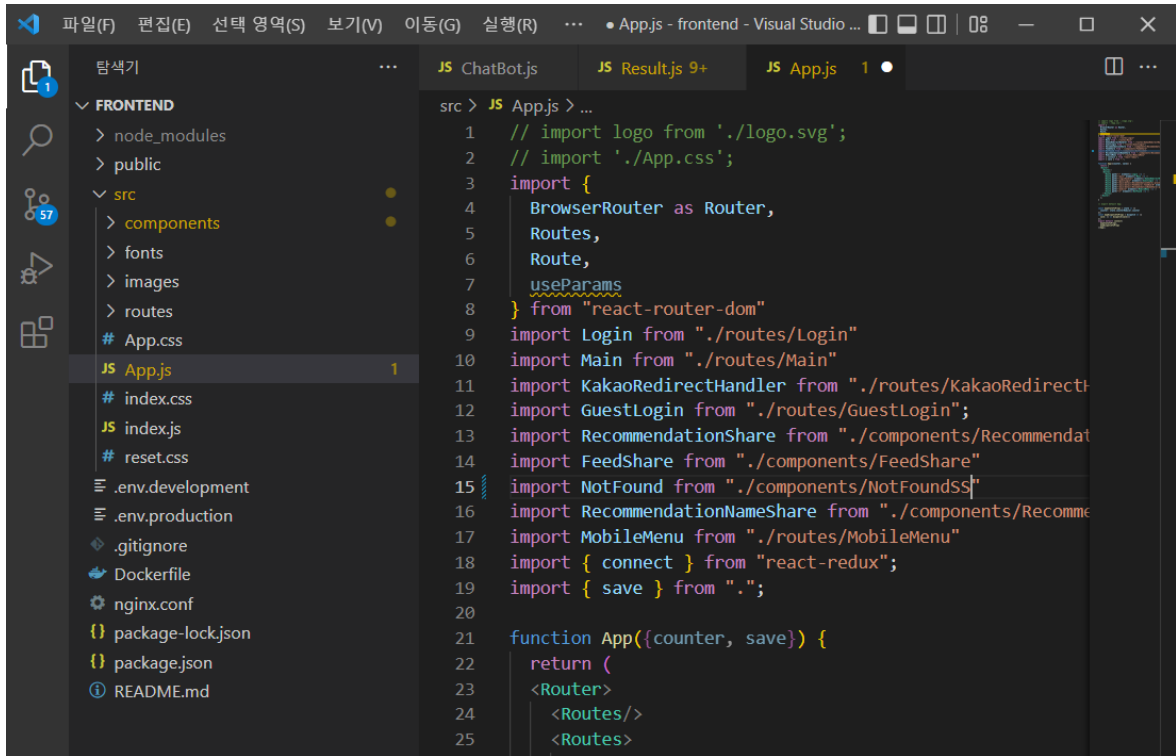
frontend 폴더에서 cmd창을 켜다. 다음 명령어를 치면 vs code가 실행된다.

```
code .
```

vs code 터미널에서 명령어를 작성한다.

```
npm install
npm run build
```

다음과 같이 진행된다.



빌드가 완료되면, 현재 나의 React.js 프로젝트 폴더에 dist 폴더가 생성된다. 이 폴더 안에 html, css, js 파일들이 추출된다.

dist >			
이름	수정된 날짜	유형	크기
css	2022-08-18 오후 1:12	파일 폴더	
fonts	2022-08-18 오후 1:11	파일 폴더	
img	2022-08-18 오후 1:12	파일 폴더	
original	2022-08-18 오후 1:11	파일 폴더	
resources	2022-08-18 오후 1:11	파일 폴더	
favicon	2022-08-18 오후 1:11	아이콘	
index	2022-08-18 오후 1:11	Chrome HTML Do...	

## 7-2. 백엔드

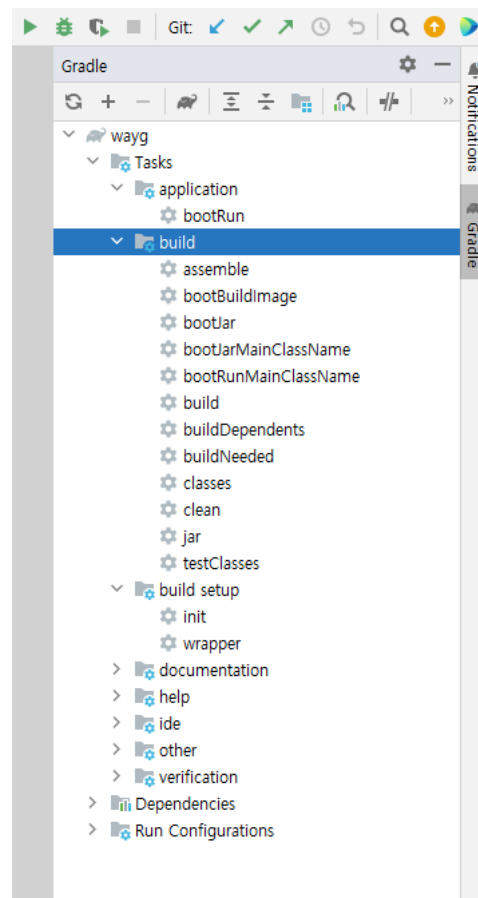
### 1) application.properties

src/main/resources/applicant.properties에서 DB 정보를 환경에 맞게 수정한다. 현재 프로젝트의 설정은 이렇다.

```
spring.datasource.url=jdbc:mysql://j7c202.p.ssafy.io:3306/wayg?useUnicode=yes&characterEncoding=UTF-8&serverTimezone=Asia/Seoul
spring.datasource.username=wayg
spring.datasource.password=1llkhhC205*
```

### 2) 빌드

gradle 탭의 clean, build를 클릭한다.



build/libs밑에 jar파일이 만들어진다.

이름	수정한 날짜	유형	크기
wayg-1.0.4-SNAPSHOT-202210061616...	2022-10-06 오후 4:16	Zulu jar file	63,174KB

### 3) 실행

jar파일이 있는 위치에서 cmd 창을 열고 다음 명령문을 작성한다.

```
java -jar 이름.jar
```

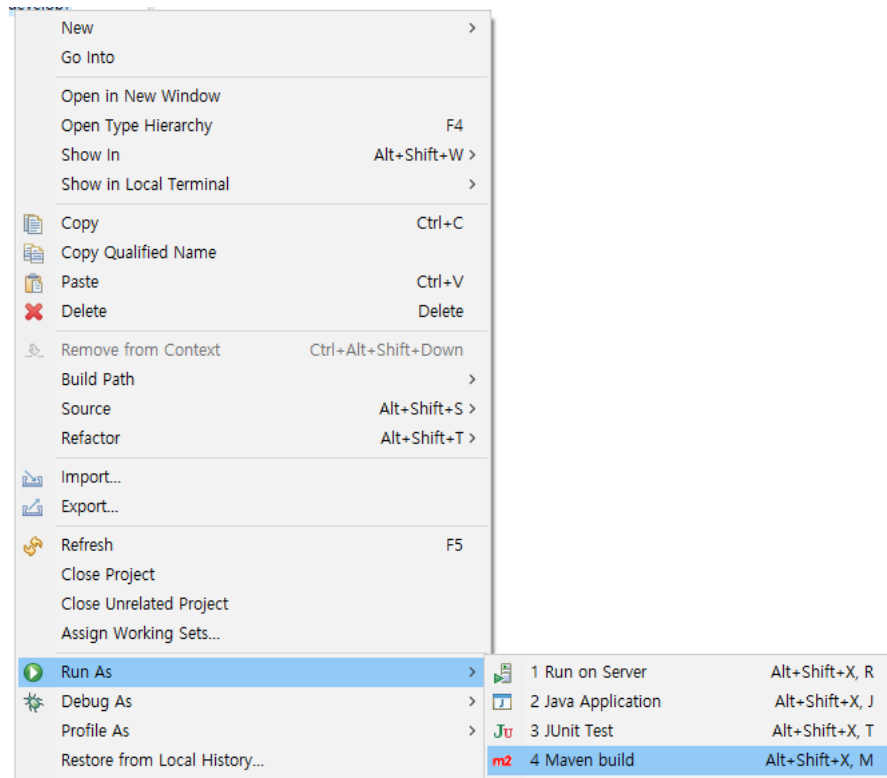
<http://localhost:8080/api> 주소로 spring 서버가 실행된다.

## 7-3. 하둡-맵 리듀스

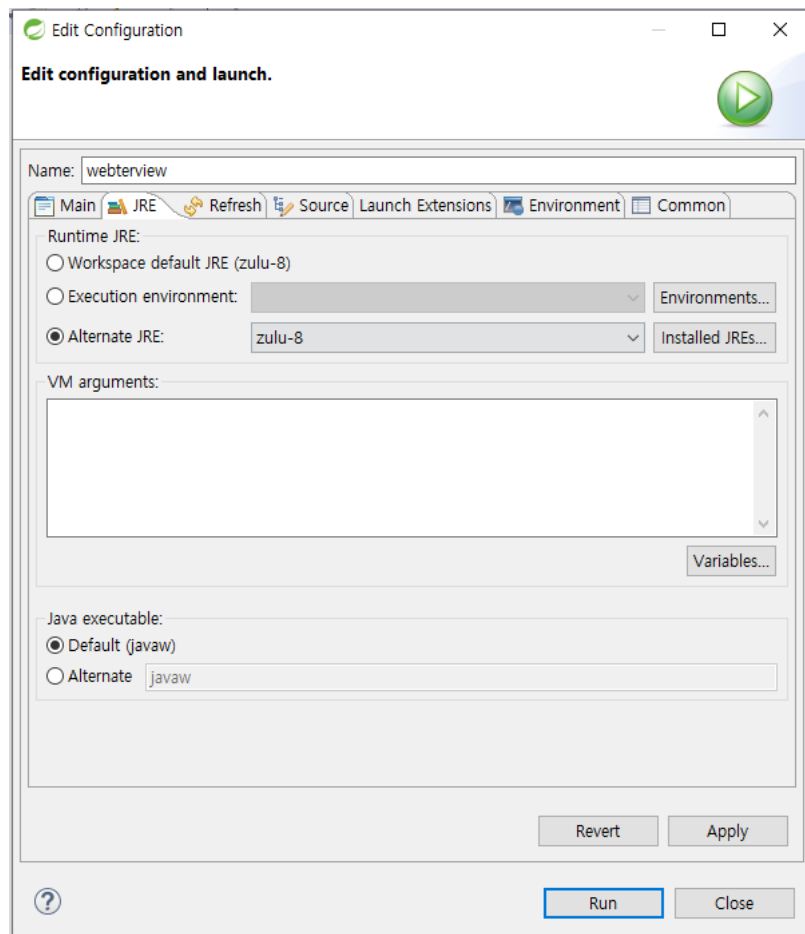
hadoop > wayg\_hadoop 위치에 있다.

### 1) 빌드

프로젝트 이름에서 오른쪽마우스키를 눌러 Run As > Maven Build를 클릭한다.









이름을 적은 후 JRE 탭에서 zulu-8을 선택한 후 Run한다.





target 위치에 wayg-1.0.jar이 생기면 빌드가 완료된 것이다.

이름	수정된 날짜	유형	크기
 classes	2022-09-26 오전 11:19	파일 폴더	
 generated-sources	2022-09-26 오전 11:19	파일 폴더	
 maven-archiver	2022-09-26 오전 11:19	파일 폴더	
 maven-status	2022-09-26 오전 11:19	파일 폴더	
 original-wayg-1.0.jar	2022-09-26 오전 11:19	Zulu jar file	12KB
 wayg-1.0.jar	2022-09-26 오전 11:19	Zulu jar file	6,874KB