



시나브로

포팅 매뉴얼

D203

팀장: 윤선영

팀원: 김호균, 양동기, 이아현, 이진우

1.프로젝트 개발 환경..... 4

협업 도구.....4

Back-end.....4

Front-end.....4

Media Server.....5

Server.....5

2. 백엔드 빌드 방법..... 6

이클립스에 프로젝트 임포트.....6

프로젝트 빌드.....9

3. 프론트엔드 빌드 방법 12

프로젝트 열기..... 12

프로젝트 빌드..... 12

4. EC2 세팅 14

EC2 에 필요한 도구 설치..... 14

5. 자동 빌드 및 배포 방법(with Jenkins) 14

Jenkins 설치 및 설정.....	14
프론트엔드, 백엔드 Docker 설정	23
MySQL Docker 설정	24
OpenVidu CE 설치 및 실행.....	25
nginx 설정.....	26
결론	27

6. MySQL 워크벤치 사용법 28

1. 프로젝트 개발 환경

협업 도구

- 1) 형상 관리: GitLab
- 2) 이슈 관리: Jira
- 3) 소통 및 자료 공유: MatterMost, Webex, Figma
- 4) 와이어프레임 (UI / UX): Figma

Back-end

- 1) OS: Window 10
- 2) JAVA: OpenJDK 11.0.15 버전
- 3) IDE: 이클립스 2022-06 버전
- 4) Framework: Spring Boot 2.7.8 버전
 - Spring Boot Data JPA
 - QueryDSL 5.0.0 버전
 - Spring Boot Security
 - Swagger-ui 3.0.0 버전
 - ModelMapper 3.1.1 버전
 - Lombok
 - jjwt 0.11.5 버전

Front-end

- 1) OS: Window 10

- 2) Node.js: 14.17.x 버전
- 3) IDE: Visual Studio Code 1.74 버전
- 4) Framework: React 18.2.0 버전
 - Axios 1.3.1 버전
 - Openvidu-browser 2.25.0 버전
 - React-dom 18.2.0 버전
 - React-icons 4.7.1 버전
 - React-redux 8.0.5 버전
 - React-router-dom 6.8.0 버전
 - React-scripts 5.0.1 버전
 - Styled-component 5.3.6 버전

Media Server

- 1) OS: Ubuntu 버전
- 2) OpenVidu: 2.25.0 버전

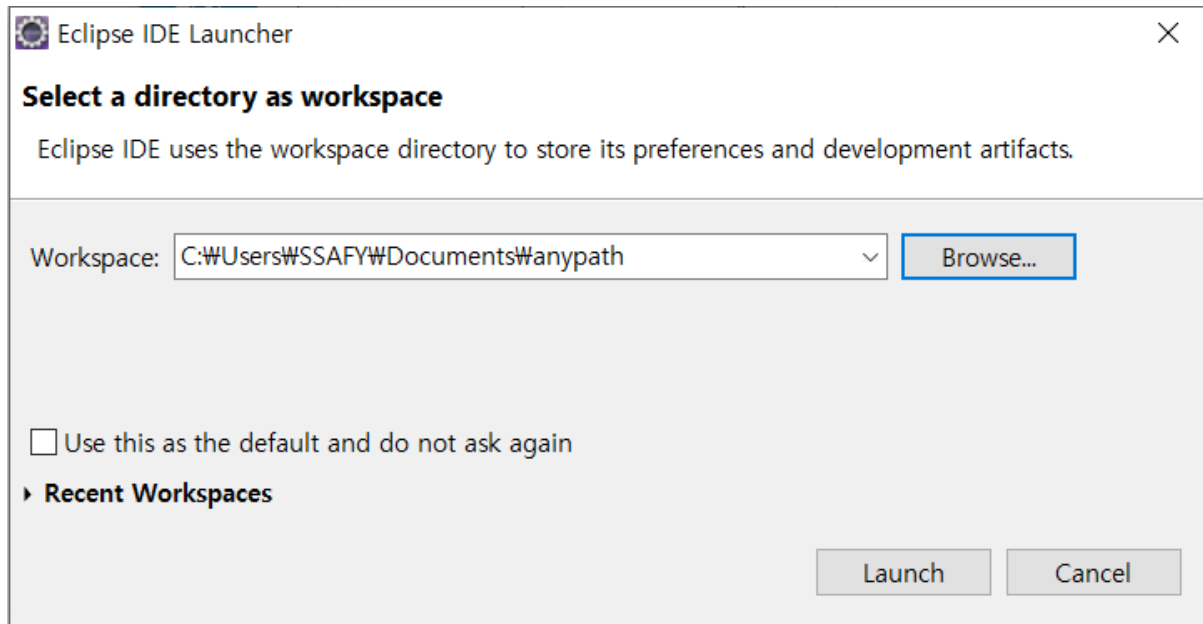
Server

- 1) OS: Ubuntu
- 2) 웹서버: nginx
- 3) 컨테이너: docker
- 4) 자동화 도구: Jenkins

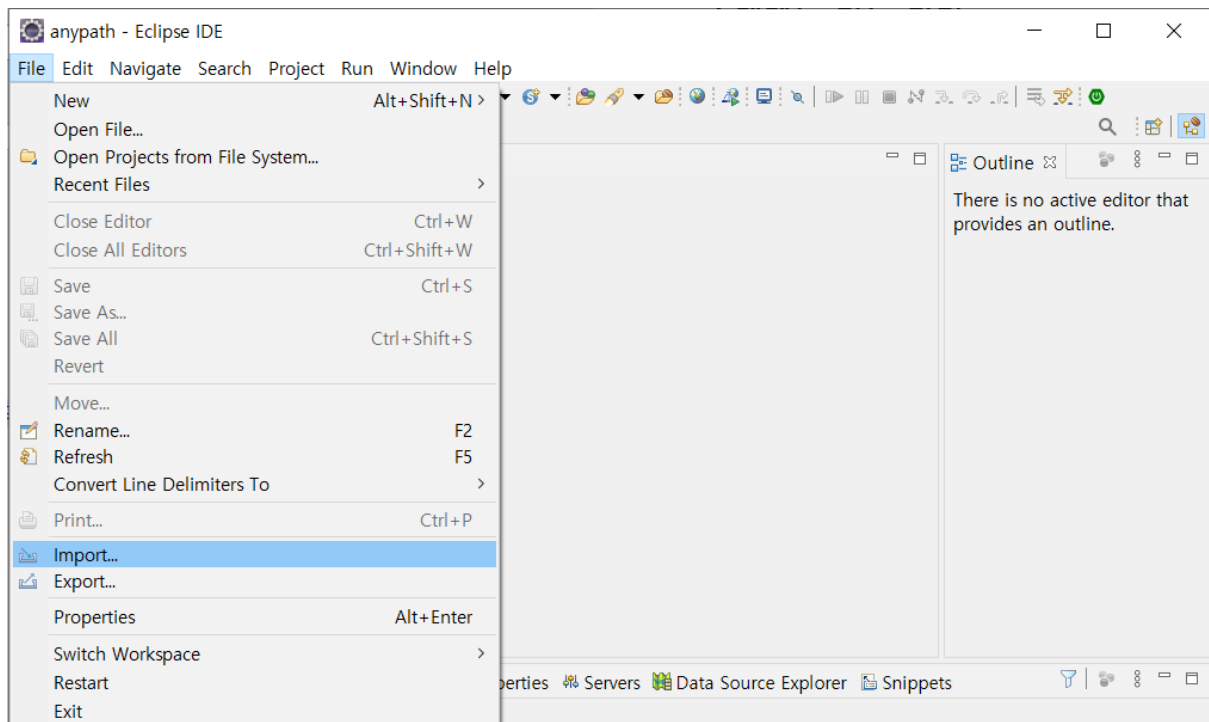
2. 백엔드 빌드 방법

이클립스에 프로젝트 импорт

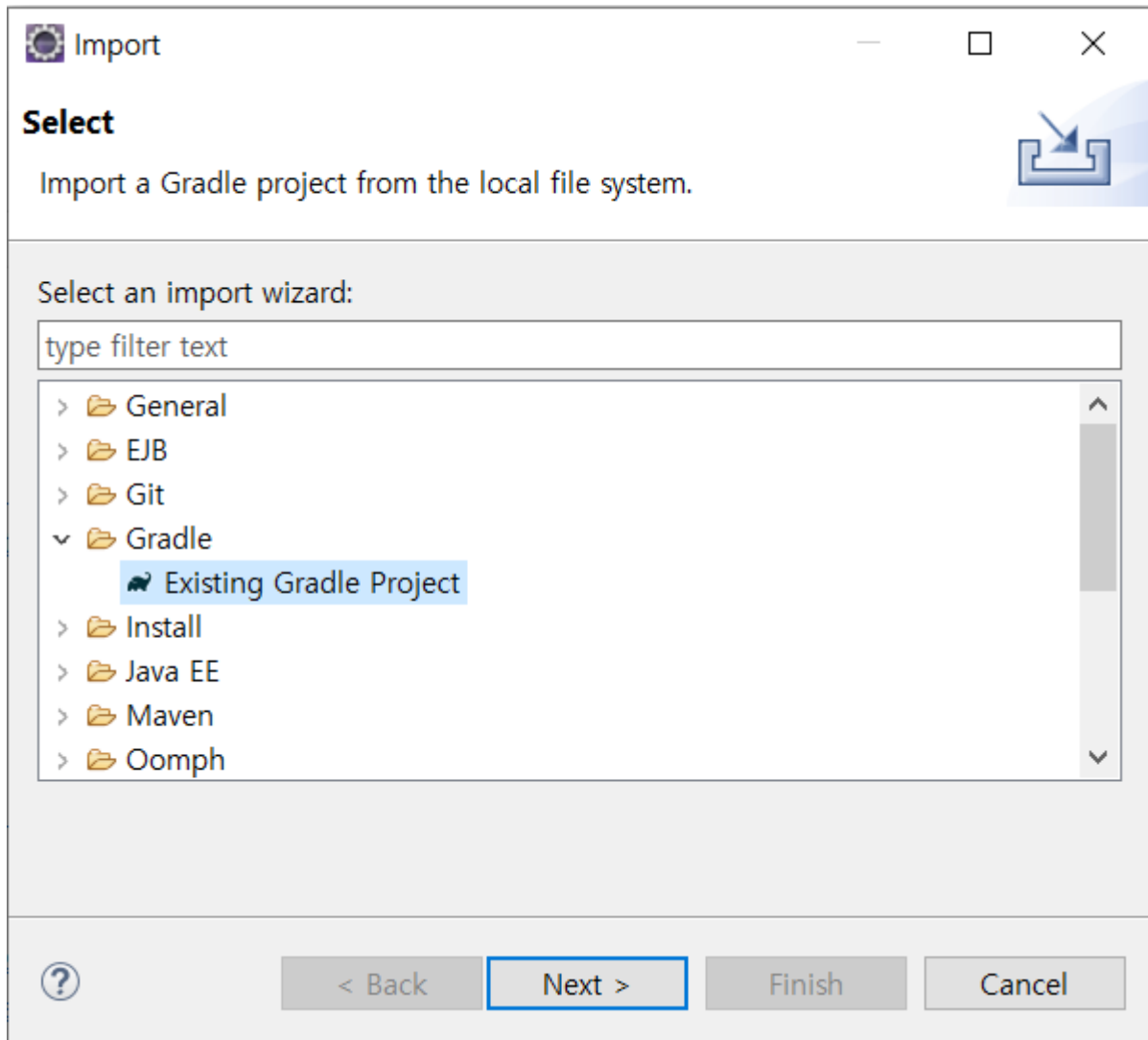
- 1) 이클립스 실행, Workspace 경로는 클론한 프로젝트 경로와 다르기만 하면 된다.



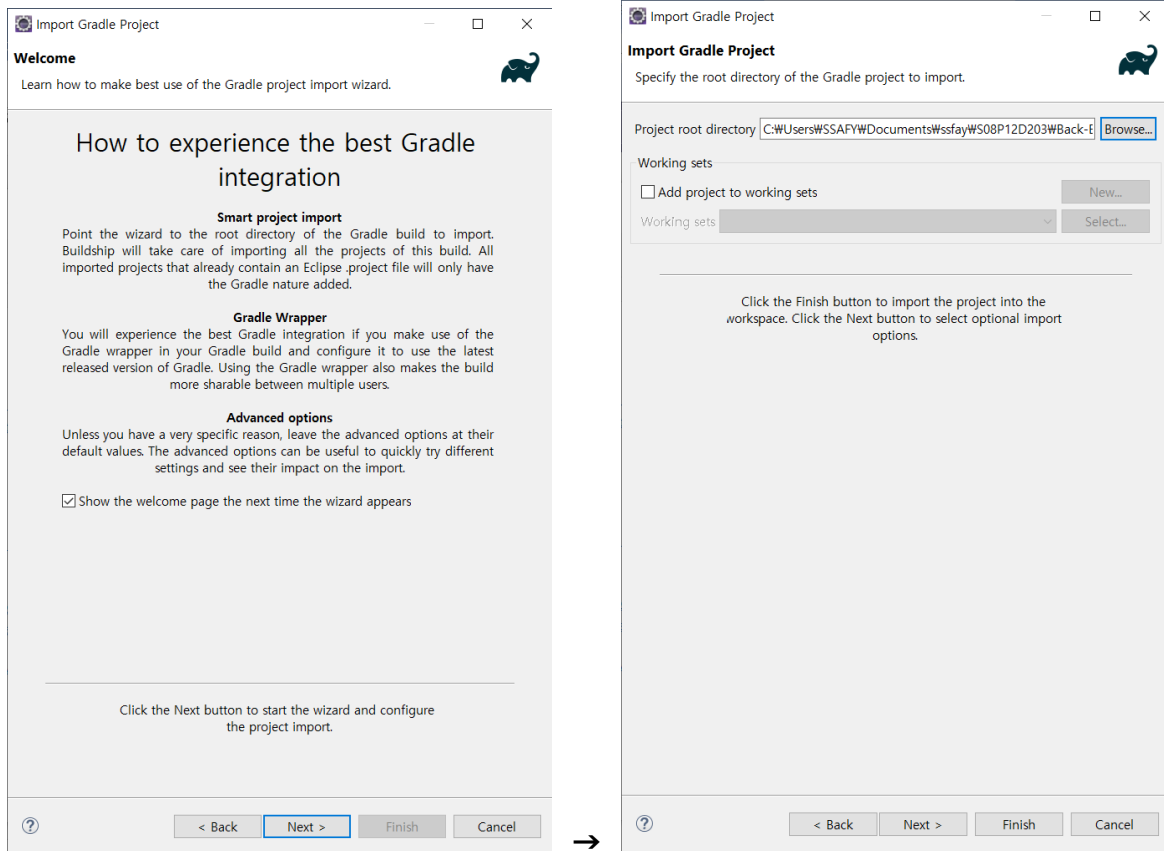
- 2) File > Import 로 진입



3) Existing Gradle Project 클릭

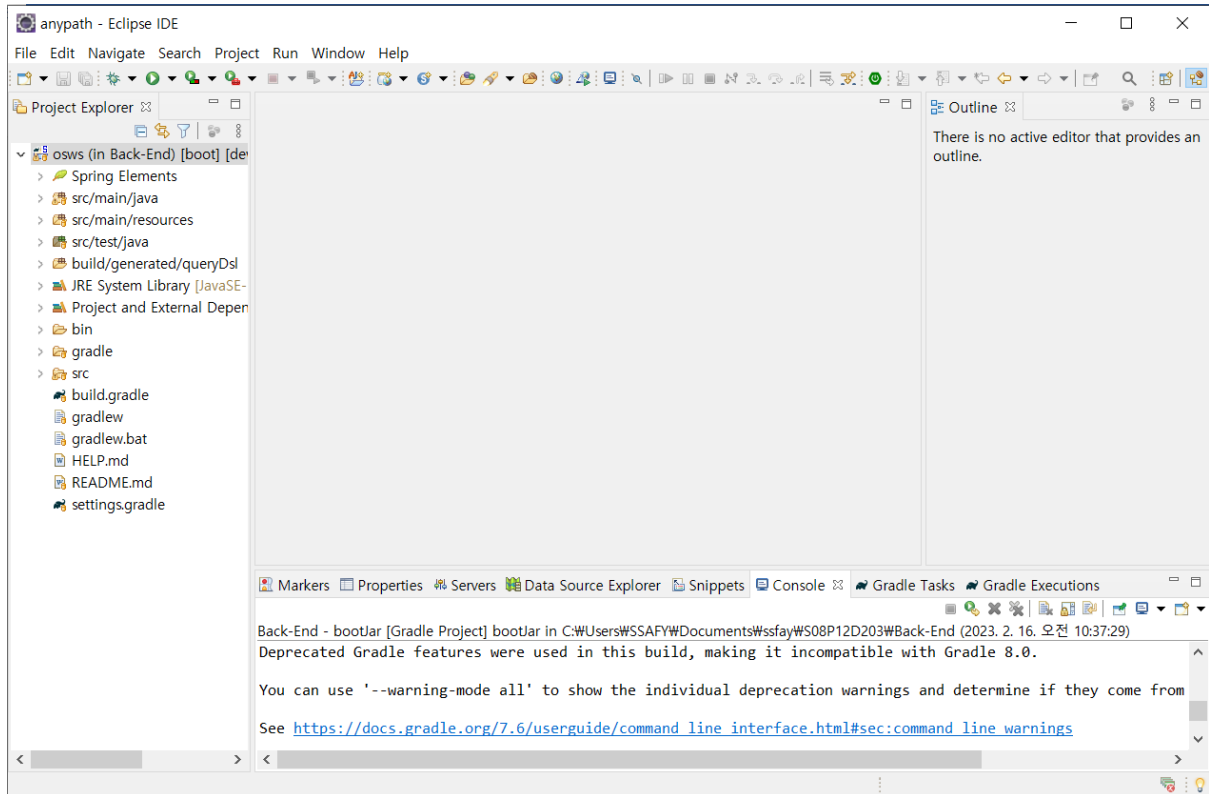


4) Next > Project Root Directory 를 프로젝트가 있는
폴더WS08P12D203WBack-End 로 지정후 Finish 클릭

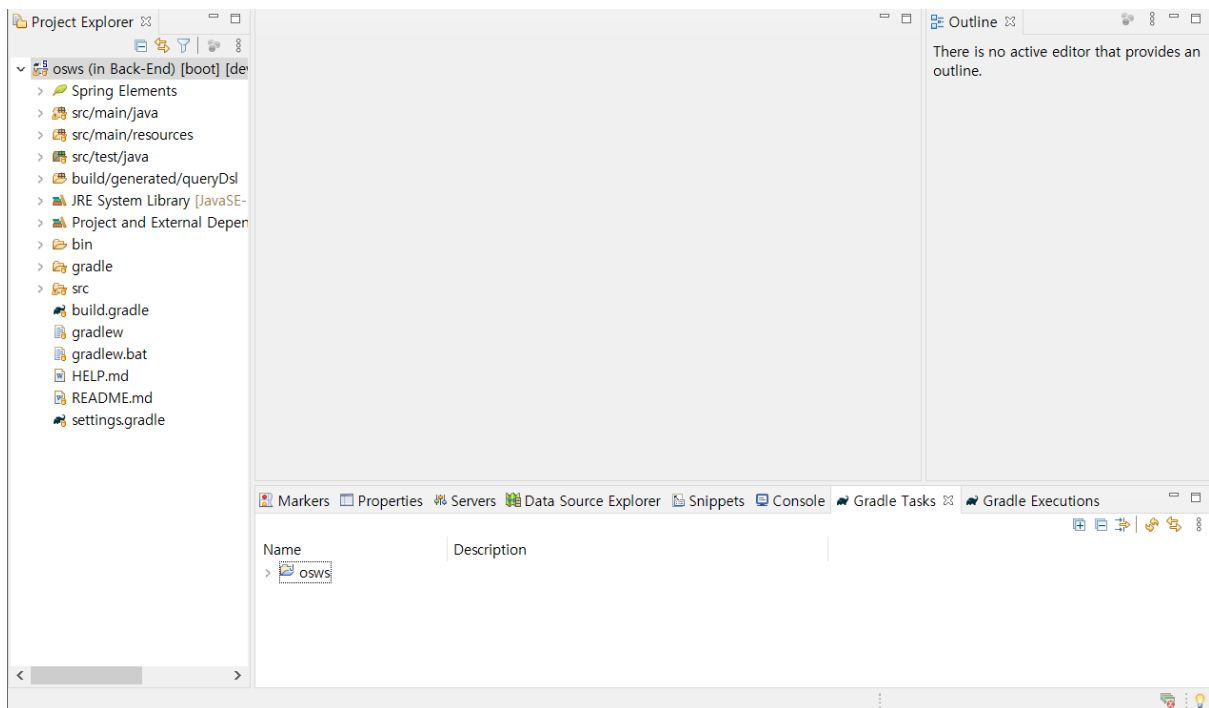


프로젝트 빌드

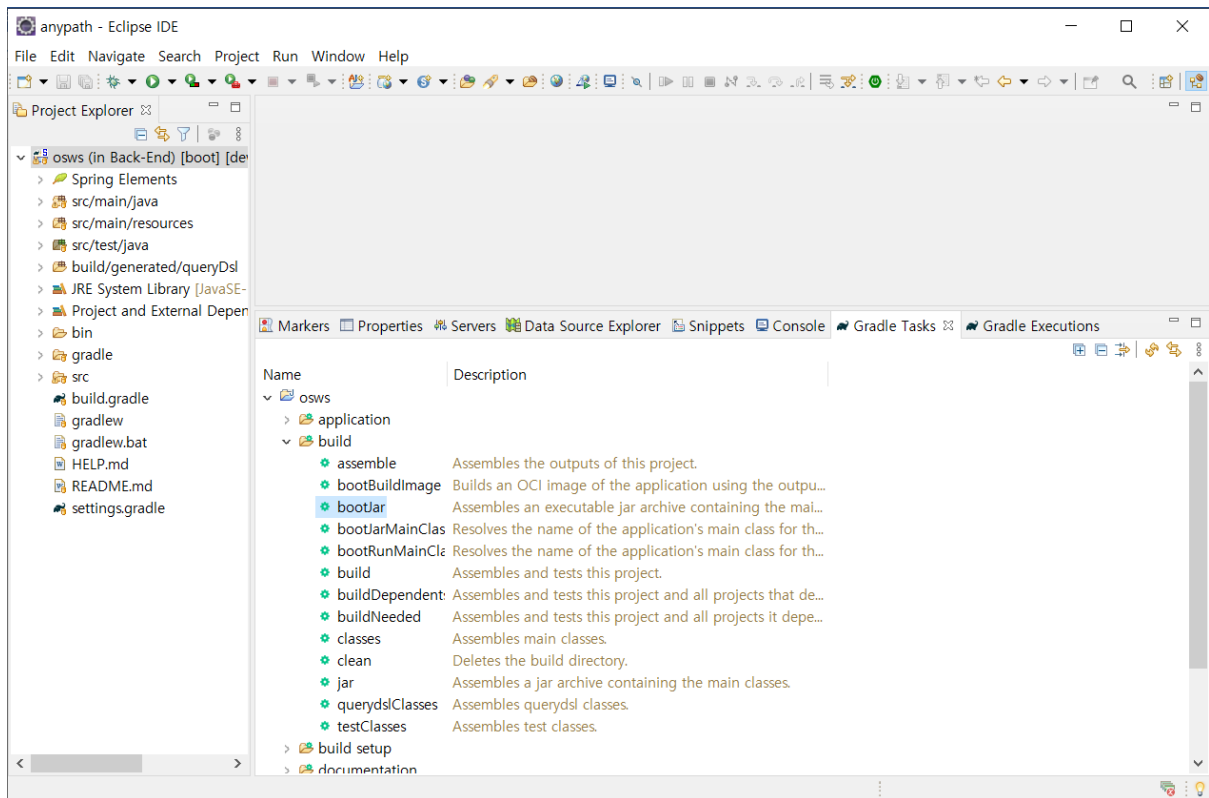
1) 임포트된 osws 프로젝트 클릭



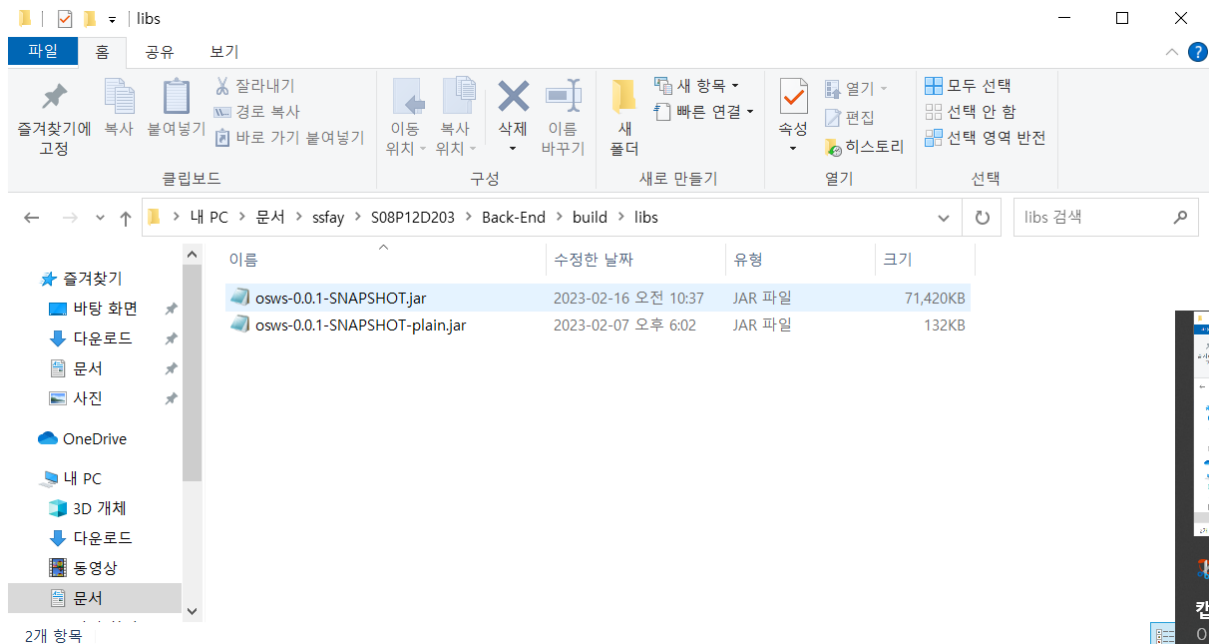
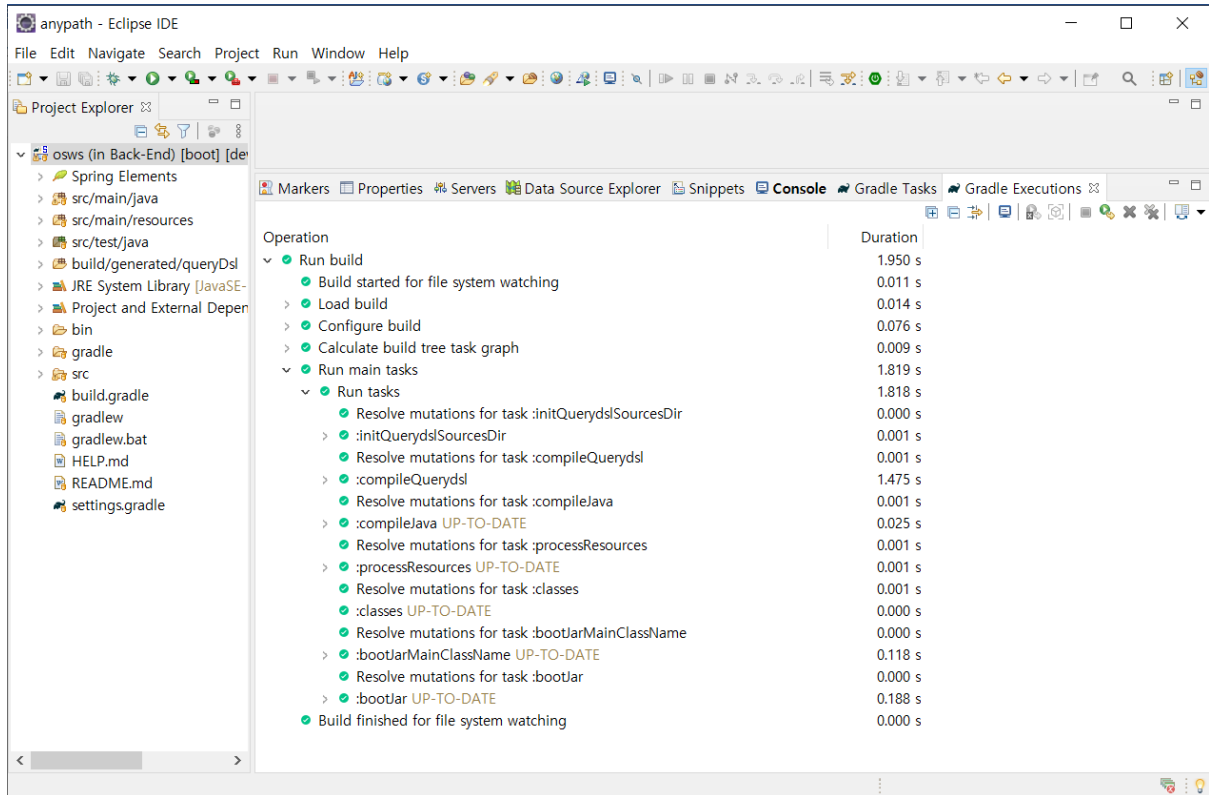
2) 하단에 GradleTasks 클릭



3) osws > build > bootJar 클릭



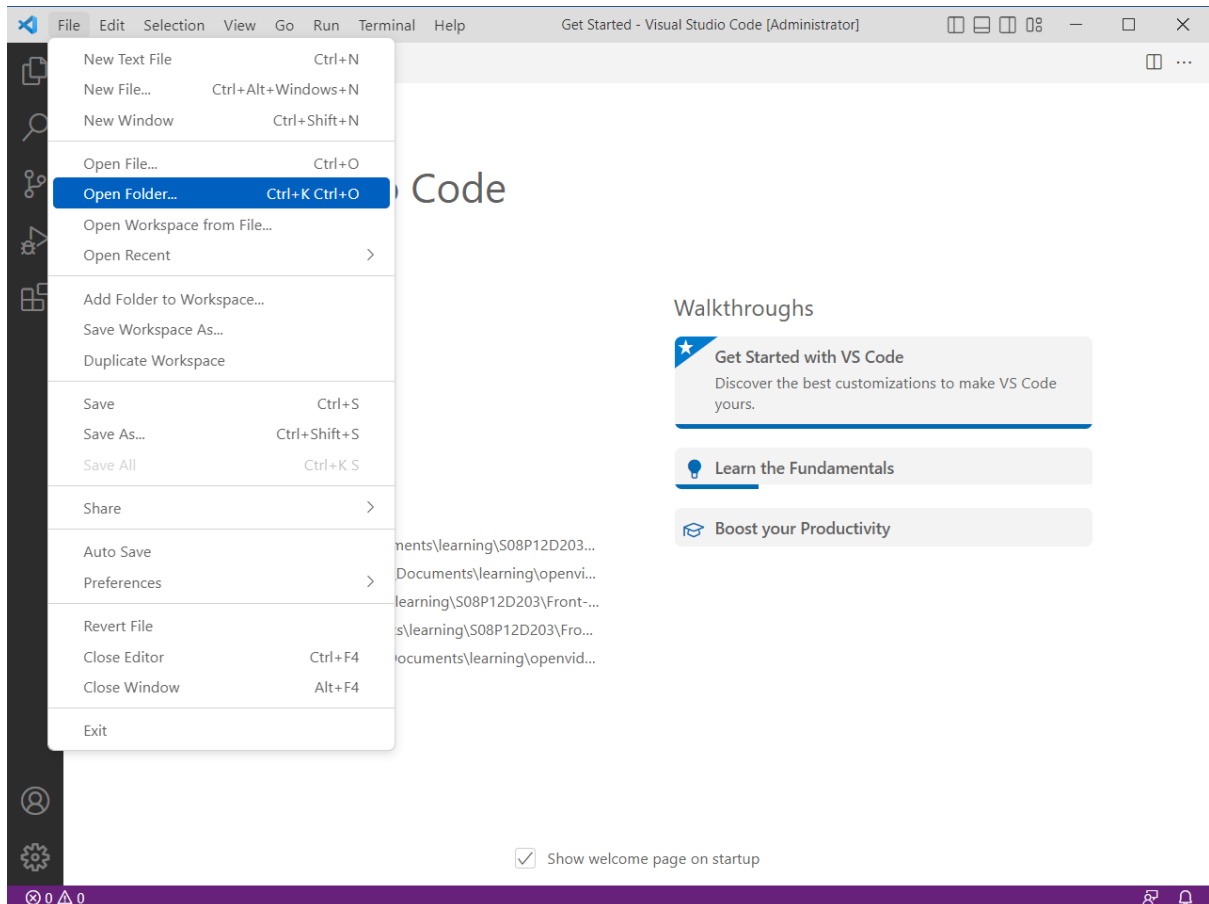
4) build 가 완료되면 {클론한 경로}\WS08P12D203\Back-End\build\libs 에서 .jar 파일을 확인할 수 있다.



3. 프론트엔드 빌드 방법

프로젝트 열기

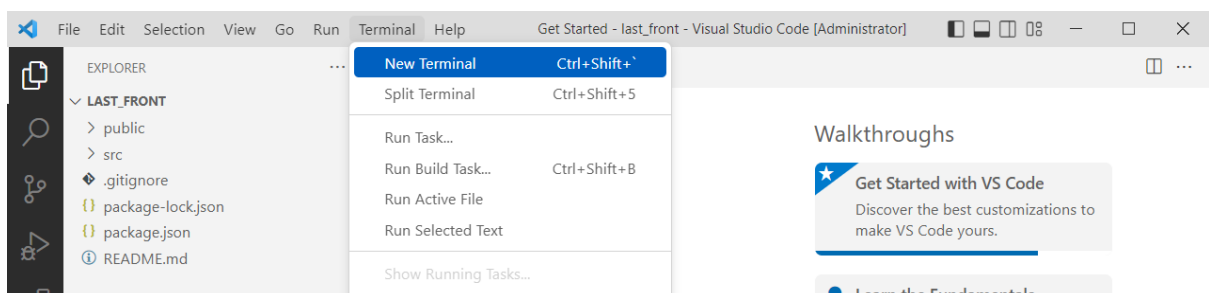
1) Visual Studio Code 실행 > File > Open Folder 클릭



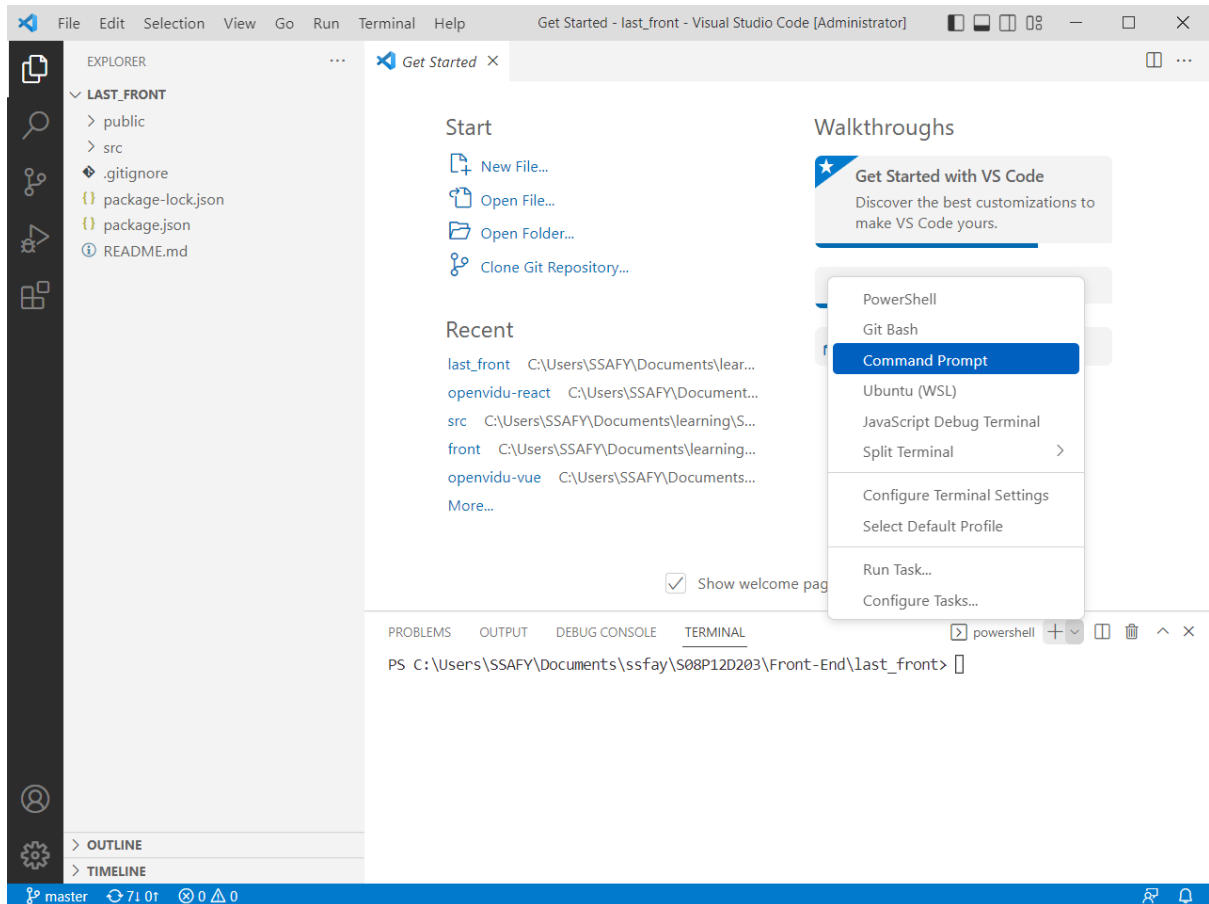
2) {클론한 경로}WS08P12D203WFront-EndWlast_front 열기

프로젝트 빌드

1) Terminal > New Terminal 클릭



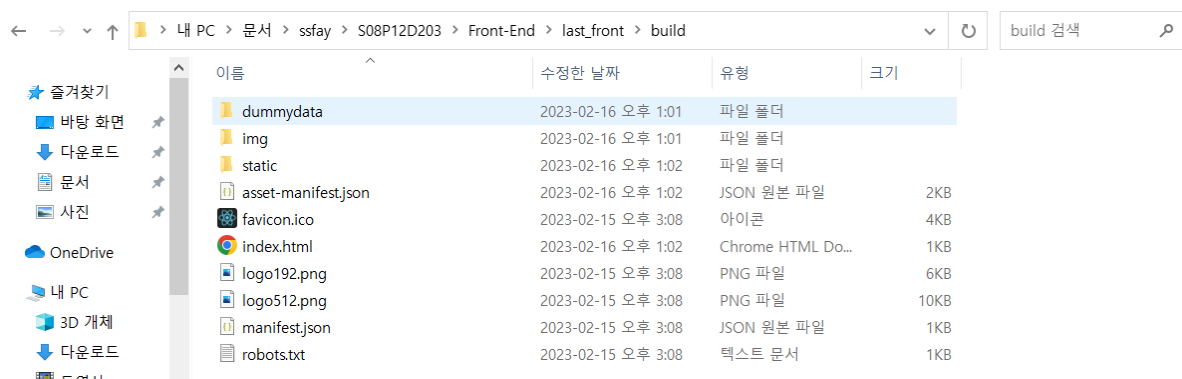
2) 하단 터미널창 우측 V 모양 클릭 후 Command Prompt 선택



3) Terminal 이 cmd 창으로 변경되면 아래 명령어를 순서대로 입력하면 된다.

```
$npm install  
$npm run build
```

4) {클론한 경로} W S08P12D203 W Front-End W last_front W build 내부에서 build 한 파일을 확인 할 수 있습니다.



4. EC2 세팅

EC2 에 필요한 도구 설치

EC2 에 Docker 를 설치한다.

```
$sudo apt-get update
$sudo apt-get install docker.io
```

개발 환경과 호환성을 위해 Node.js 14.17.4 버전을 설치한다.

```
$sudo apt-get install curl
$curl -o- https://raw.githubusercontent.com/nvm-
  sh/nvm/v0.39.1/install.sh | bash source ~/.bashrc
$nvm -v
$nvm install 14.17.4
```

Nginx 를 설치한다.

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

5. 자동 빌드 및 배포 방법(with Jenkins)

Jenkins 설치 및 설정

1) Jenkins 를 Docker 로 설치한다.

```
$sudo docker pull jenkins/jenkins:lts
$sudo docker run -d \ -p 9000:8080 \ -p 50000:50000 \ -v Jenkins
  volume:/var/jenkins_home \ -v
  /var/run/docker.sock:/var/run/docker.sock:ro \ -v
  /var/lib/docker/containers:/var/lib/docker/containers:ro \ --
  net=bridge \ --name jenkins \ jenkins/jenkins:lts
```

2) Jenkins 접속 주소인 <http://i8d203.p.ssafy.io:9000> 에 접속한다.

3) 아래 명령어를 통해 Jenkins container 로그를 확인하고 로그에
적혀있는 admin 암호를 복사한다.

```
$sudo docker logs jenkins
```

```
*****
*****
*****

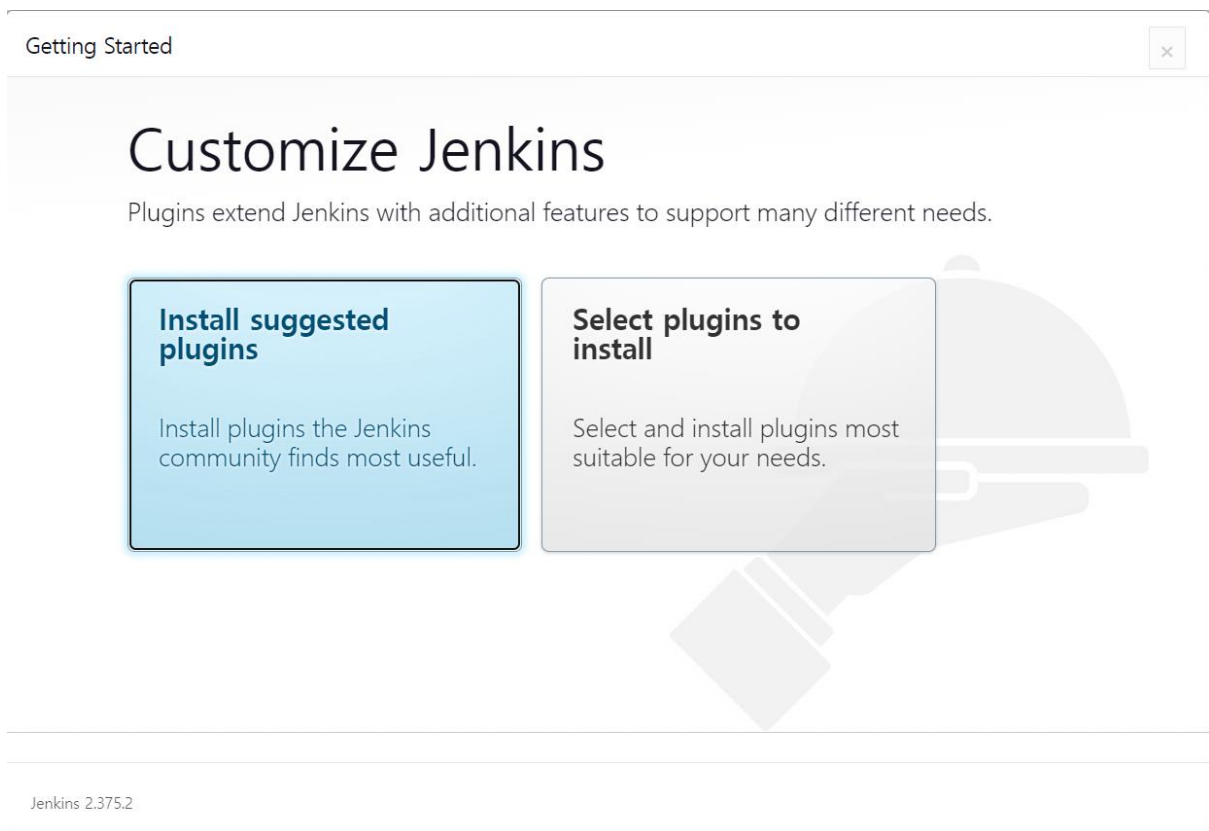
Jenkins initial setup is required. An admin user has been created and a password generated.
Please use the following password to proceed to installation:

4b935fab7b7348eb93b2cbe7490e687a

This may also be found at: /var/jenkins_home/secrets/initialAdminPassword

*****
*****
*****
```

4) 암호 입력 후 다음 단계에서 Install suggested plugins 설치한다.



5) 이후 나오는 Dashboard 화면에서 Jenkins 관리 > Plugin Manager > Available plugins 에서 아래 목록의 plugins 를 설치한다.

- Docker API Plugin
- Docker Commons Plugin
- Docker Pipeline
- Docker plugin


- Git plugin
- GitLab
- GitLab API Plugin
- GitLab Authentication plugin
- GitLab Branch Source Plugin
- Gradle Plugin
- NodeJS Plugin
- nvm-wrapper
- SCM API Plugin
- SSH Agent Plugin
- SSH Credentials Plugin
- SSH server

6) Jenkins dashboard 에서 새로운 Item 을 생성한다. (frontend 기준)


Enter an item name

frontend


» Required field


Freestyle project


이것은 Jenkins의 주요 기능입니다. Jenkins은 어느 빌드 시스템과 어떤 SCM(형상관리)으로 묶인 당신의 프로젝트를 빌드할 것이고, 소프트웨어 빌드보다 다른 어떤 것에 자주 사용될 수 있습니다.


Maven project


Maven 프로젝트를 빌드합니다. Jenkins은 POM 파일의 이점을 가지고 있고 급격히 설정을 줄입니다.


Pipeline

Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.


Multi-configuration project

다양한 환경에서의 테스트, 플레폼 특정 빌드, 기타 등등 처럼 다수의 서로다른 환경설정이 필요한 프로젝트에 적합함.


Folder

Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a

OK

7) Build Triggers 에서 'Build when a change is pushed to Gitlab' 을
체크하고, 우측에 나오는 webhook 주소와 하단의 Secret token 을
다시 확인할 수 있는 곳에 저장한다.

☒ Build when a change is pushed to GitLab. GitLab webhook URL: <http://i8d203.p.ssafy.io:9000/project/frontend> ?

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 ▼

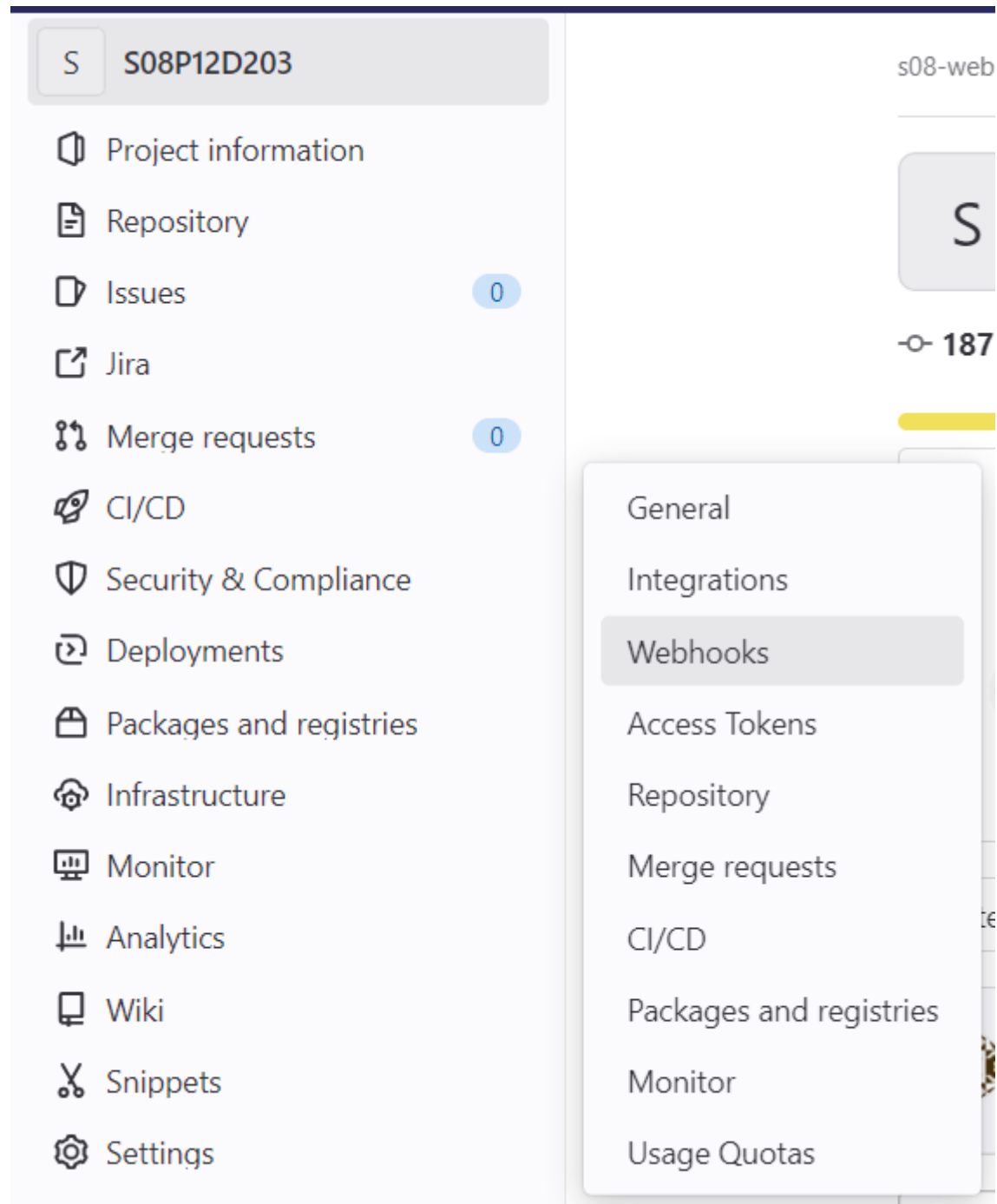
Secret token ?

96f58d5d6422cfc247874fe1ed248839

[Generate](#)

[Clear](#)

8) Gitlab 에서 Settings > Webhook 에 들어간다.



- 9) Webhook 설정 화면에서 10 번에서 따왔던 webhook 주소와 secret token 을 기입하고 Webhook 을 추가한다.

Q Search page

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 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

Push to the repository.

- 10) Jenkins 화면으로 돌아가서, 하단의 Pipeline 메뉴에서 Definition 에 'Pipeline script from SCM'을 선택 후, SCM 에 'Git'을 선택하고, Repository URL 에 프로젝트의 Gitlab 주소를 입력한다.

Pipeline

Definition

SCM ?

Repositories ?

Repository URL ?

! Failed to connect to repository : Command `"/usr/lib/git-core/git ls-remote -h -- https://lab.ssafy.com/s08-webmobile1-sub2/S08P12D203.git HEAD"` returned status code 128:
stdout:
stderr: remote: HTTP Basic: Access denied. The provided password or token is incorrect or your account has 2FA enabled and you must use a personal access token instead of a password. See https://lab.ssafy.com/help/topics/git/troubleshooting_git#error-on-git-fetch-http-basic-access-denied
fatal: Authentication failed for 'https://lab.ssafy.com/s08-webmobile1-sub2/S08P12D203.git/'

Credentials ?

- 11) Credentials 아래의 Add 버튼을 통해 credentials 을 추가한다.
Kind 에 'Username with password'를 선택하고, Username 에
GitLab 의 유저이름을, Password 에 본인 계정의 비밀번호를
기입하고, ID 에는 이 credential 이 어떤 건지를 표시할 수 있도록
적어준다.

Kind

Username with password

Scope ?

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

Username ?

Gitlab 유저이름

☐ Treat username as secret ?

Password ?

.....

ID ?

GitLab

- 12) 만들어준 Credential 을 선택한다.

SCM ?

Git

Repositories ?

Repository URL ?

https://lab.ssafy.com/s08-webmobile1-sub2/S08P12D203.git

Credentials ?

GitLab

+ Add

고급...

Add Repository

13) Script Path 에 작성한 Jenkinsfile 이름을 적어준다.

Script Path ?

Jenkinsfile_frontend

14) item 을 저장한다.

15) Jenkinsfile 에 필요한 credentials 을 만든다. (Docker hub, EC2 ssh)

16) Jenkinsfile 에 pipeline 을 작성한다.

```
pipeline {
  agent any
  tools {nodejs "nodejs"}
  stages {
    stage('Gitlab') {
      steps {
        script {
          git branch: 'master',
            credentialsId: 'GitLab',
            url: 'https://lab.ssafy.com/s08-webmobile1-
sub2/S08P12D203.git'
        }
      }
    }
    stage('Build') {
      steps {
        dir('Front-End/last_front') {
          script {
            sh "npm install"
            sh "npm run build"
          }
        }
        dir('Front-End') {
          script {
```


프론트엔드, 백엔드 Docker 설정

pipeline 상으로 작동할 수 있도록, Git 의 해당 경로에 프론트엔드 React 를 빌드할 Dockerfile 과 백엔드 Spring Boot 를 빌드할 Dockerfile 을 저장한다. 프론트엔드는 nginx container 기반으로 작성하기 때문에, 해당 nginx 설정파일인 nginx.conf 파일 역시 해당 경로에 맞추어 작성한다.

- 프론트엔드용 Dockerfile

```
FROM nginx

RUN mkdir /app

WORKDIR /app

RUN mkdir ./build

ADD ./last_front/build ./build

RUN rm /etc/nginx/conf.d/default.conf

COPY ./nginx.conf /etc/nginx/conf.d/

EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]
```

- 백엔드용 Dockerfile

```
FROM openjdk:11

ARG JAR_FILE=*.jar
COPY ${JAR_FILE} osws.jar
ENTRYPOINT ["java","-jar","/osws.jar"]

EXPOSE 8080
```

- 프론트엔드용 nginx.conf

```
server {
    listen 80;
    location / {
        root    /app/build;
        index   index.html;
        try_files $uri $uri/ /index.html;
    }
}
```

MySQL Docker 설정

- 1) EC2 내에서 MySQL docker image 를 빌드할 Dockerfile 을 생성한다.

```
FROM mysql:5.7

ADD ./mysql-init-files /docker-entrypoint-initdb.d

ENV MYSQL_USER [유저 이름]

ENV MYSQL_PASSWORD [유저 비밀번호]

ENV MYSQL_ROOT_PASSWORD [루트 비밀번호]

ENV MYSQL_DATABASE [DB 이름]

EXPOSE 3306
```


2) Dockerfile 에 써둔 것 처럼, mysql-init-files 라는 폴더를 해당 경로에 맞게 생성한 후 내부에 image 를 빌드할 때 DB 에 필요한 sql 파일을 작성해서 저장해 둔다.

3) MySQL 용 Docker image 를 만든다.

```
$sudo docker build -t xronace/d203-mysql:latest .
```

4) MySQL docker container 를 생성 및 실행한다.

```
$sudo docker run -dp 3306:3306 -v mysql-volume --net=bridge --name d203-mysql xronace/d203-mysql:latest
```

OpenVidu CE 설치 및 실행

1) EC2 내부에서 OpenVidu CE 를 다운로드 받는다. 이때 저장 경로는 /opt 를 추천한다. root 권한이 필요하다.

```
$sudo su
$cd /opt
$curl https://s3-eu-west-1.amazonaws.com/aws.openvidu.io/install_openvidu_latest.sh |
  bash
```

2) 다운로드 받은 경로에 들어가서 .env 파일을 편집한다.

```
$cd {다운로드 받은 경로}
$vi .env
```

3) .env 파일에 아래 설정을 수정한다. HTTP_PORT 와 HTTPS_PORT 는 주석처리 되어 있으니 잘 확인해야 한다.

```
10. DOMAIN_OR_PUBLIC_IP=i8d203.p.ssafy.io
13. OPENVIDU_SECRET={스프링 application.yml 내에 OPENVIDU_SECRET 값}
24. CERTIFICATE_TYPE=letsencrypt
27. LETSENCRYPT_EMAIL={유효한 이메일}
36. HTTP_PORT=8442
40. HTTPS_PORT=8443
```

4) , openvidu 폴더에서 openvidu 관련 docker 들을 다음 명령어로 실행한다.

```
$/openvidu start
```

nginx 설정

1) 설치한 nginx 를 임시로 중지한다.

```
$sudo systemctl stop nginx
```

2) Let's Encrypt 를 설치한다.

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

3) 무료 인증서를 발급받는다.

```
$sudo letsencrypt certonly --standalone -d i8d203.p.ssafy.io
```

4) 사용할 d203_nginx.conf 환경설정 파일을 /etc/nginx/sites-available 에 작성한다. 아래 두 명령어를 실행하면 수정할 수 있다.

```
$cd /etc/nginx/sites-available
```

```
$sudo vi d203_nginx.conf
```

- d203_nginx.conf 파일

```
server {  
    location /{  
        proxy_pass http://localhost:3000;  
    }  
  
    location /api{  
        proxy_pass http://localhost:5000/api;  
    }  
  
    listen 443 ssl;  
    ssl_certificate  
        /etc/letsencrypt/live/i8d203.p.ssafy.io/fullchain.pem;  
    ssl_certificate_key  
        /etc/letsencrypt/live/i8d203.p.ssafy.io/privkey.pem;
```

```
# include /etc/letsencrypt/options-ssl-nginx.conf; # managed by
  Certbot

# ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by
  Certbot
}

server {
    if ($host = i8d203.p.ssafy.io) {
        return 301 https://$host$request_uri;
    }

    listen 80;
    server_name i8d203.p.ssafy.io;
    return 404;
}
```

5) 적용을 위해 해당 파일을 링크로 연동한다.

```
$sudo ln -s /etc/nginx/sites-available/d203_nginx.conf
  /etc/nginx/sites-enabled/d203_nginx.conf
```

6) nginx 를 테스트 후, 성공적이면 nginx 서버를 재시작한다.

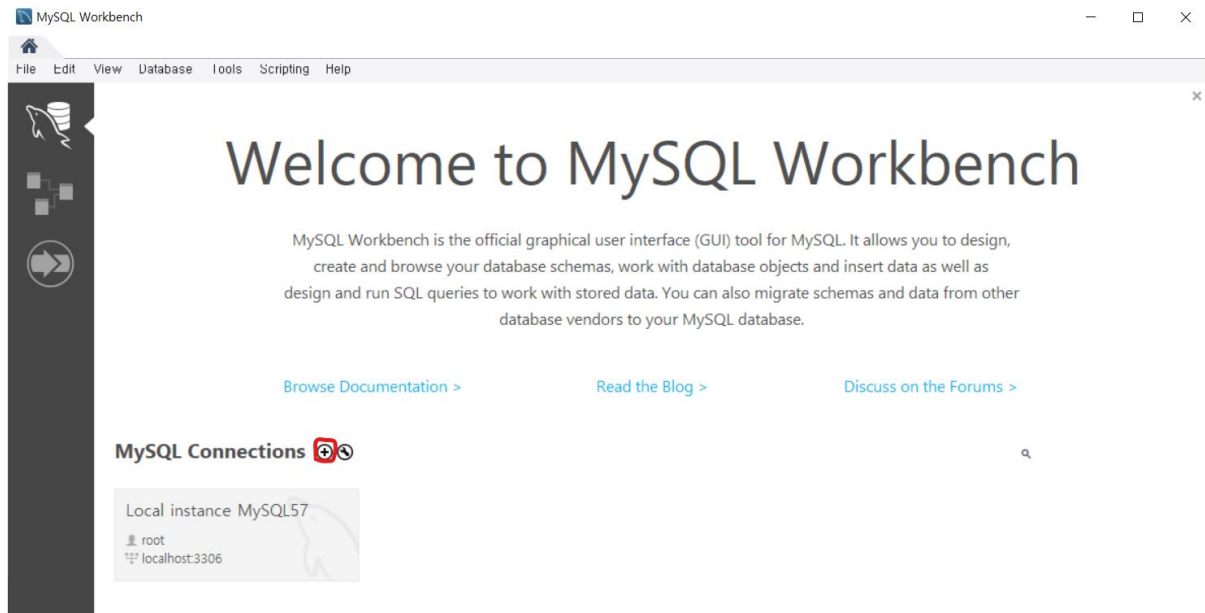
```
$sudo nginx -t
$sudo systemctl restart nginx
```

결론

위 과정을 모두 거치면 Gitlab 의 master 브랜치에 push 가 발생할 때
마다 자동으로 빌드 및 배포가 진행된다.

6. MySQL 워크벤치 사용법

- 1) 워크벤치 실행 후 나온 메인 화면에서 MySQL Connections 우측 ⊕ 버튼을 클릭한다.



- 2) 아래 사진과 같이 정보를 입력하고 Test Connection 을 실시합니다.
비어있는 Connection Name 창에는 원하는 이름을 입력하시면
됩니다. (password 는 [여기로](#) 들어가 확인하면 됩니다.)

Setup New Connection

Connection Name: Type a name for the connection

Connection Method: Method to use to connect to the RDBMS

Parameters SSL Advanced

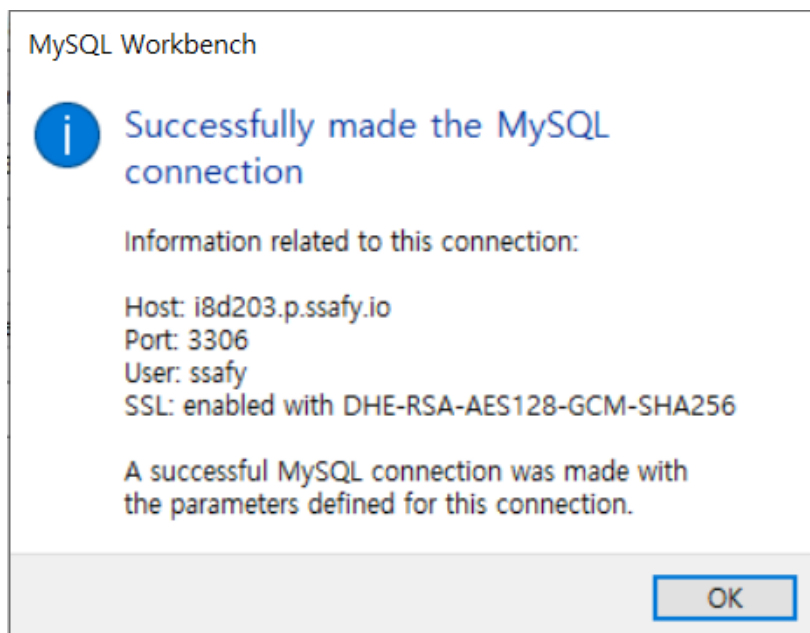
Hostname: Port: Name or IP address of the server host - and TCP/IP port.

Username: Name of the user to connect with.

Password: The user's password. Will be requested later if it's not set.

Default Schema: The schema to use as default schema. Leave blank to select it later.

- 3) Test Connection 결과가 아래 사진처럼 나오면 성공적으로 DB 에
접속할 수 있습니다.



4) 메인 페이지에서 추가된 데이터베이스 계정을 확인할 수 있습니다.

