



# 포팅 매뉴얼

## SWith 포팅 매뉴얼

| SSAFY 6기 공통 프로젝트 서울 5반 1팀 SWith

박주미 김도현 김수연 김윤하 이성재 한지희



### 목차

- 1 프로젝트 기술 스택
- 2 계정 정보
- 3 환경 설정 및 프로퍼티 파일
- 4 빌드 및 배포 방법
- 5 외부 서비스



### 프로젝트 기술 스택

1. 이슈 관리 : Jira
2. 형상 관리 : Gitlab
3. 커뮤니케이션 : Mattermost, Notion
4. 개발 환경
  - OS : Windows 10
  - IDE : IntelliJ 2021.3.1 / Visual Studio Code 1.63.2
  - Server : AWS EC2
    - Ubuntu 20.04 LTS
    - Jenkins 2.319.2
    - Docker 20.10.7
    - Nginx 1.18.0 (Ubuntu)
  - Database : MySQL 5.7
  - File Storage : Cloud Storage(Firebase)
  - OpenVidu 2.20.0
  - Express.js 4.17.2
  - Frontend
    - HTML5, CSS3, Javascript(ES6)
    - Vue 3.0.0
    - Vue-CLI 4.5.0
    - Vuex 4.0.0
    - Node.js 16.13.1
  - Backend
    - Java 1.8.0

- Spring Boot 2.6.2
- JPA(Hibernate)
- Gradle 7.3.2

## 계정 정보

### ➔ DB 접속 계정 정보

- Username : kimdocker
- Password : Ssafypjt1^^

### ➔ Jenkins 접속 계정 정보

- Username : swith
- Password : Ssafypjt1^^
- Jenkins 관리자 페이지
  - <http://i6a501.p.ssafy.io:9090>로 접속
    - 주의 : https가 아니라 http 입니다!

## 환경 설정 및 프로퍼티 파일

### Frontend

- .env

```
frontend
└── .env
```

```
1 VUE_APP_BASE_URL_DEV=http://localhost:8080
2 VUE_APP_KAKAO_CLIENT_ID=
3 VUE_APP_KAKAO_REDIRECT_URI=https://i6a501.p.ssafy.io/member
4 VUE_APP_GOOGLE_CLIENT_ID=
5 VUE_APP_GOOGLE_REDIRECT_URI=https://i6a501.p.ssafy.io/login
6 VUE_APP_OPENVIDU_SERVER_URL=https://i6a501.p.ssafy.io:4443
7 VUE_APP_OPENVIDU_SERVER_SECRET=MY_SECRET
8 VUE_APP_EXPRESS_SERVER_URL=https://i6a501.p.ssafy.io
```

- 1 : backend와 통신하기 위한 URL
- 2 : Kakao client ID
- 3 : Kakao redirect URI
- 4 : Google client ID
- 5 : Google redirect URI
- 6 : OpenVidu server URL
- 7 : OpenVidu server secret key
- 8 : Express.js server의 socket 연결을 위한 URL

### Backend

- application.properties

```

backend
└─ src
   └─ main
      └─ resources
         └─ application.properties

```

- DB 설정
- Email 설정
- Firebase 설정
  - 서비스 접속을 위한 비공개 키
  - 파일이 저장되는 경로
- **CORS 설정**

```

backend
└─ src
   └─ main
      └─ java
         └─ com
            └─ swith
               └─ api
                  └─ config
                     └─ SecurityConfig.java

```

- host에서 오는 요청에 대해 CORS 허용

```

79      @Bean
80      CorsConfigurationSource corsConfigurationSource() {
81          CorsConfiguration configuration = new CorsConfiguration();
82
83          configuration.addAllowedOrigin("http://localhost:8081");
84          configuration.addAllowedOrigin("https://i6a501.p.ssafy.io");

```

- **Kakao/Google 소셜 로그인 설정**

```

backend
└─ src
   └─ main
      └─ java
         └─ com
            └─ swith
               └─ api
                  └─ service
                     └─ AuthServiceImpl.java

```

- Kakao redirect URI

```

86      private String getAccessTokenByKakao(String authCode) {
87          HttpHeaders headers = new HttpHeaders();
88          headers.add("Content-type", "application/
89
90          MultiValueMap<String, String> params = new LinkedMultiValueMap
91          params.add("grant_type", "authorization_code");
92          params.add("client_id", "b87b2face727a7093e3816185ab2697c");
93          params.add("code", authCode);
94          params.add("redirect_uri", "https://i6a501.p.ssafy.io/members/

```

- Google redirect URI

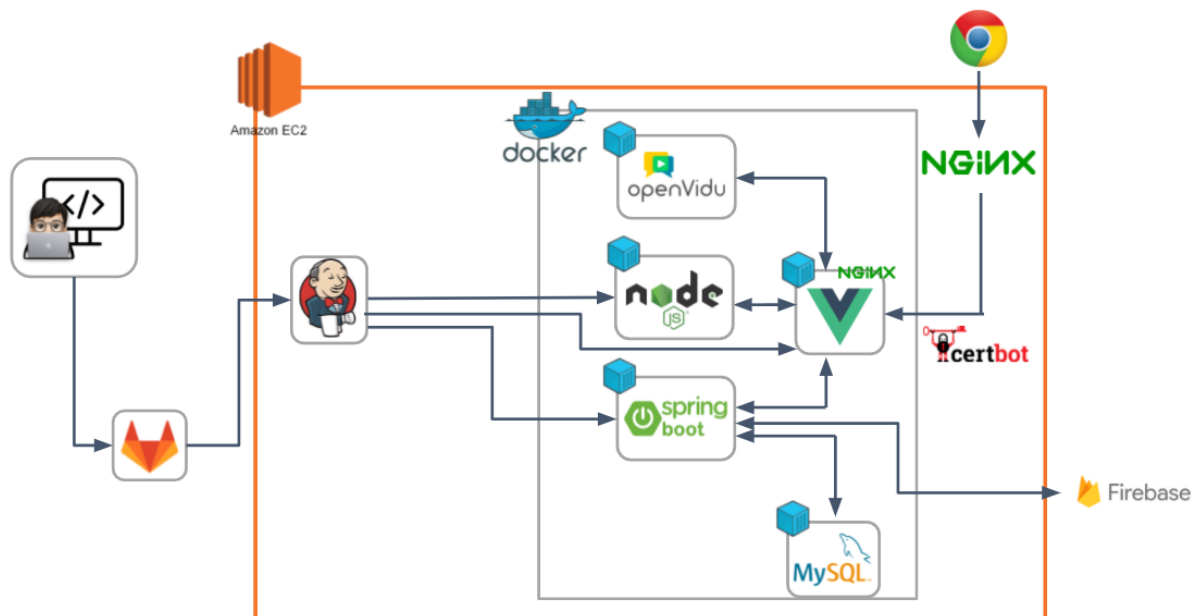
```

144     public String getIdTokenByGoogle(String authCode) {
145         String idToken = "";
146         String reqURL = "https://oauth2.googleapis.com/token";
147
148         try {
149             URL url = new URL(reqURL);
150             HttpURLConnection conn = (HttpURLConnection) url.openConnection();
151             conn.setRequestMethod("POST");
152             conn.setDoOutput(true);
153
154             BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(
155                 conn.getOutputStream(), "UTF-8"));
156             String sb = new StringBuilder();
157             sb.append("grant_type=authorization_code");
158             sb.append("&client_id=343513890539-mvk01v00kfnp5vdcvfu95h");
159             sb.append("&redirect_uri=https://i6a501.p.ssafy.io/login/");

```

## 📢 빌드 및 배포 방법

### 0. 서버 아키텍처



### 1. Docker 설치

- Windows : <https://www.docker.com/get-started>
- Ubuntu(Linux) and etc.
  - Ubuntu : <https://docs.docker.com/engine/install/ubuntu/>

```

# Set up the repository
$ sudo apt-get update
$ sudo apt-get install \
    ca-certificates \
    curl \

```

```

gnupg \
lsb-release
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
$ echo \
"deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

# Install Docker Engine
$ sudo apt-get install -y docker.io
$ sudo usermod -a -G docker $USER

```

- etc : <https://docs.docker.com/engine/install/>

## 2. OpenVidu 설치

- On premises : <https://docs.openvidu.io/en/stable/deployment/pro/on-premises/>

```

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

```

- 실행 / 재실행 / 중지

```

$ ./openvidu start # 실행
$ ./openvidu restart # 재실행
$ ./openvidu stop # 중지

```

## 3. Git Clone

```

$ git clone https://lab.ssafy.com/s06-webmobile1-sub2/S06P12A501.git

```

## 4. 빌드 및 배포 (Dockerize)

- Frontend

frontend 디렉토리로 이동 후 docker image 생성, docker container 실행

```

$ cd ./frontend
$ docker build -t frontend-image .
$ docker run -d -p 8081:8080 --rm --name frontend-container frontend-image

```

- Backend

backend 디렉토리로 이동 후 docker image 생성, docker container 실행

```

$ cd ./backend
$ ./gradlew clean build
$ docker build -t backend-image .
$ docker run -d -p 8080:8080 --rm --name backend-container backend-image

```

- OpenVidu

/opt/openvidu로 이동 후 openvidu 실행

```

$ cd /opt/openvidu
$ ./openvidu start

```

- Express.js (whiteboard-server)

whiteboard-server 디렉토리로 이동 후 docker image 생성, docker container 실행

```

$ cd ./whiteboard-server
$ docker build -t whiteboard-server-image .
$ docker run -d -p 3000:3000 --rm --name whiteboard-server-container whiteboard-server-image

```