



5. 포팅 매뉴얼

1. 프로젝트 기술 스택
2. 서버 아키텍처
3. 프로젝트 빌드 방법 (로컬 서버)
 - 3.1. Gitlab에서 프로젝트 클론하기
 - 3.2. 스프링부트 WAS 빌드
 - 3.2.1. gradle로 직접 빌드하는 방법 (CMD 버전)
 - 3.2.2. gradle로 직접 빌드하는 방법 (IntelliJ)
 - 3.2.3. 도커 컨테이너를 이용하는 방법
 - 3.3. 프론트엔드 웹 서버 빌드
 - 3.3.1. 도커 컨테이너를 이용한 빌드
 - 3.4. docker-compose를 이용하여 프론트, 백엔드 동시 빌드
4. 프로젝트 빌드 방법 (운영 서버)
 - 4.1. VSCode를 이용한 ssh 접속
 - 4.2. 무료 DNS 등록
 - 4.3. Let's encrypt 인증서 발급
 - 4.4. MySQL 빌드
 - 4.5. openvidu 서버 빌드 방법
 - 4.6. 스프링부트 API, 프론트 서버 빌드 방법
5. 소셜 로그인 설정
 - 5.1. 네이버 소셜 로그인
 - 5.2. 카카오 소셜 로그인

1. 프로젝트 기술 스택

BE	Infra <ul style="list-style-type: none"> • AWS EC2 • Docker 20.10.17 • Docker-compose 1.25.0 • Jenkins 2.346.2 Development <ul style="list-style-type: none"> • Java 1.8.0_192(Zulu 8.33.0.1-win64) • Spring boot 2.7.1 • spring-data-jpa 2.7.1 • hibernate-core-5.6.9.Final • spring-security:5.7.2 • projectlombok:1.18.24 DB <ul style="list-style-type: none"> • mysql 8.0.28
	FE Development <ul style="list-style-type: none"> • html5 • css3 • js(es6) • Vue 3 - 3.2.13 • Pinia • node.js - 16.16.0 LTS • openvidu - 2.22.0 • kalidokit - 1.1.5 • unity

2. 서버 아키텍처

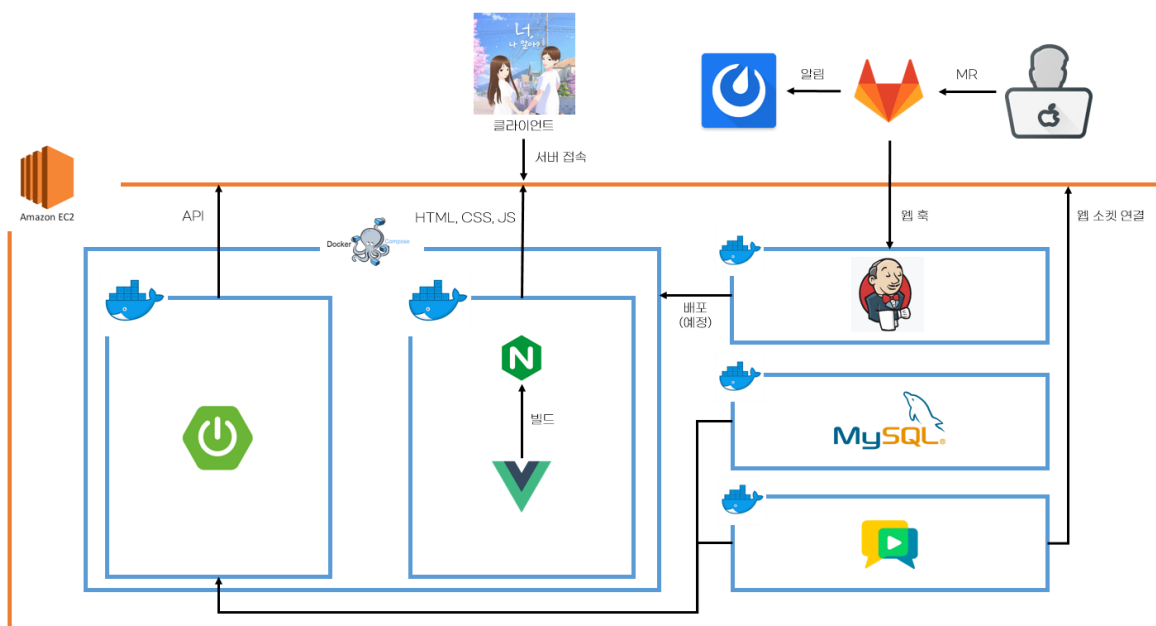


그림 1) 서버 아키텍처

본 프로젝트의 아키텍처는 위와 같습니다. 각 서버 리소스는 특정 포트로 식별 가능하며 접근할 수 있습니다.

각 서버의 포트 번호는 다음과 같습니다. (도커 배포 기준)

서버	HTTP 포트	HTTPS 포트
nginx(프론트엔드)	3000	443
tomcat(백엔드)	8080	8443
Jenkins	8085	-
DB	3306	-
openvidu	-	4443

3. 프로젝트 빌드 방법 (로컬 서버)

3.1. Gitlab에서 프로젝트 클론하기

1. 작업할 공간에 폴더를 하나 생성합니다.
2. 생성한 폴더를 열고 해당 위치에서 `Git Bash` 를 열어줍니다. (`CMD` 와 같은 다른 터미널도 상관없습니다!)
3. `git clone https://lab.ssafy.com/s07-webmobile1-sub2/S07P12D104.git` 를 터미널에 입력해줍니다.
4. 그러면 `S07P12D104` 폴더가 생깁니다. 앞으로 이 폴더를 `root directory` 라고 하겠습니다. 이후 프론트엔드, 백엔드에 따라 원하는 작업 공간으로 가서 빌드 과정을 수행해주시면 됩니다.
 - 프론트엔드 : `FE/u-know-me`
 - 백엔드 : `BE/u-know-me`

3.2. 스프링부트 WAS 빌드

스프링부트 WAS를 빌드하는 방법은 두 가지 있습니다.

- 프로젝트를 gradle로 직접 빌드하는 방법
- 도커 컨테이너를 이용하는 방법

3.2.1. gradle로 직접 빌드하는 방법 (CMD 버전)

1. **Win + R** 을 누르고 **cmd** 를 입력하고 확인 버튼을 누릅니다. 그러면 명령 프롬프트 창을 띄울 수 있습니다.
2. 백엔드 작업 공간으로 이동해줍니다. 저의 경우에는 백엔드 작업 공간이 **C:\Users\multicampus\Desktop\workspace\S07P12D104\BE\u-know-me** 입니다. 앞에 **cd** 명령어를 붙이시면 해당 디렉토리로 이동할 수 있습니다.

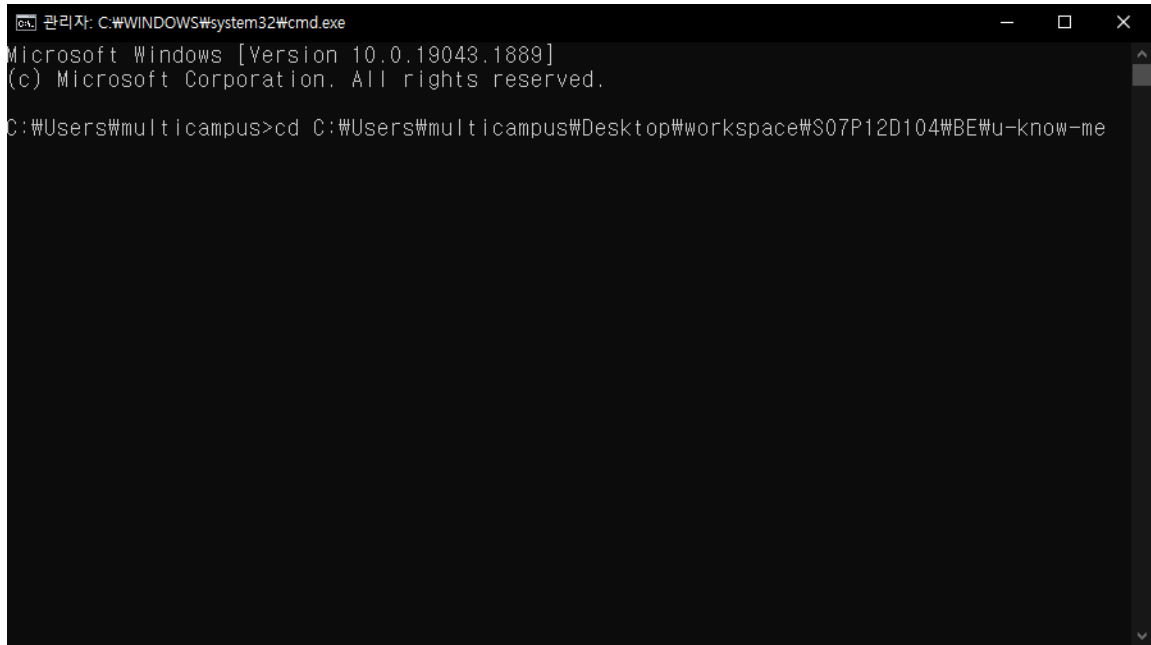


그림 2) 명령 프롬프트 cd 명령어

3. 그 후 **gradle** 을 이용하여 빌드해줍니다. **cmd** 에 **gradlew clean build** 명령어를 입력합니다. 그러면 서버 내부에서 진행하는 테스트 코드를 수행한 후 빌드 파일이 생깁니다.
4. **cd build/libs** 명령어를 입력해서 빌드 파일이 있는 위치로 이동한 후 **java -jar u-know-me-0.0.1-SNAPSHOT.jar** 명령어를 입력해줍니다.
→ 추가로 **java -jar u-know-me-0.0.1-SNAPSHOT.jar --spring.profiles.active=prod** 이라고 입력하면 운영 환경으로 설정하여 배포할 수 있지만, 현재 서버 호스트 출처가 **https://uknowme.mo00.com:8443** 으로 어플리케이션에 직접 설정해두어서 정상 작동하지 않을 수 있습니다.. 추후에 환경 변수로 설정할 수 있도록 리팩토링 해보겠습니다!
5. 위의 과정을 마치면 로컬 환경에서 서버 빌드 및 배포가 되었습니다.
http://localhost:8080/swagger-ui/ 로 접속하시면 API를 확인하실 수 있습니다.

3.2.2. gradle로 직접 빌드하는 방법 (IntelliJ)

1. 인텔리제이를 통해 해당 프로젝트를 열어줍니다.

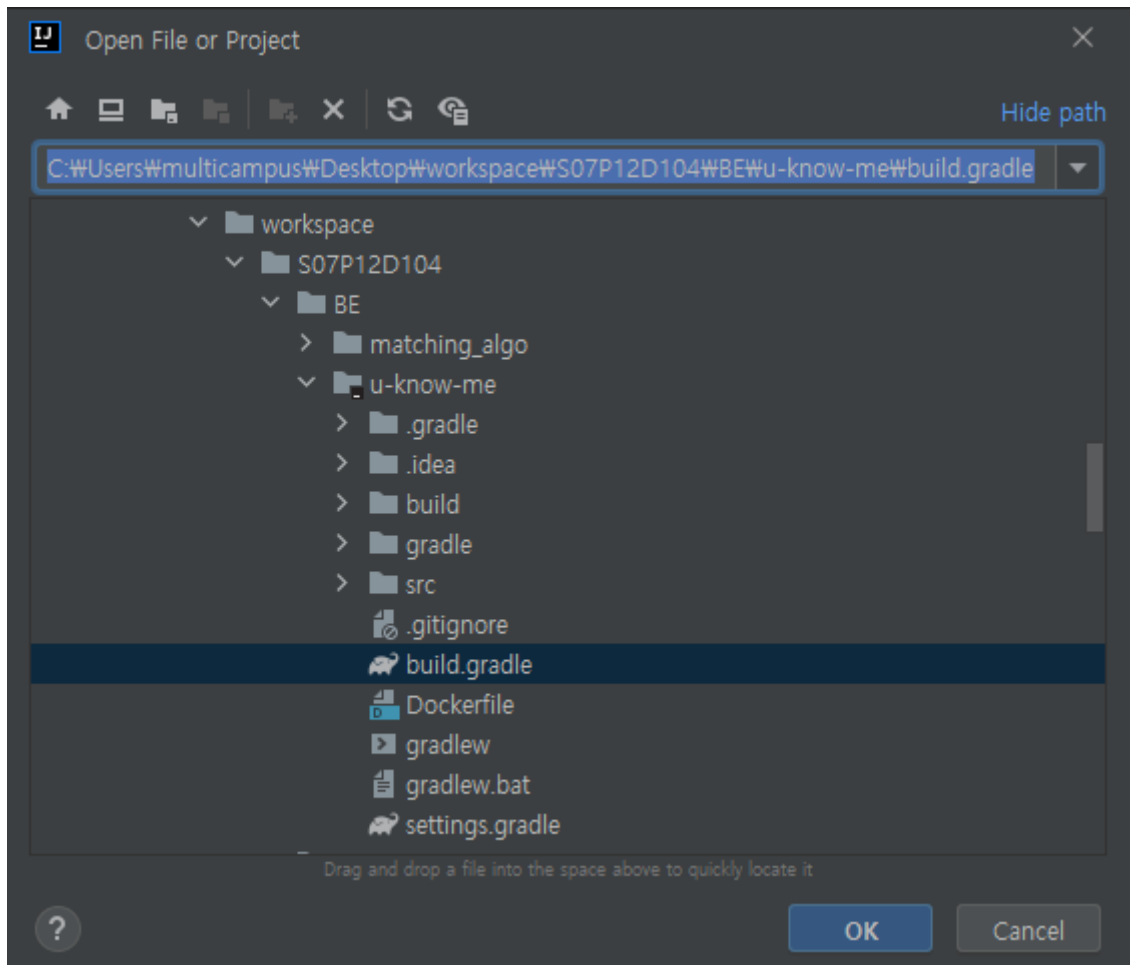


그림 3) 오픈 프로젝트를 통해 백엔드 프로젝트 열기

2. **Alt + F12** 를 눌러 터미널을 열어줍니다.

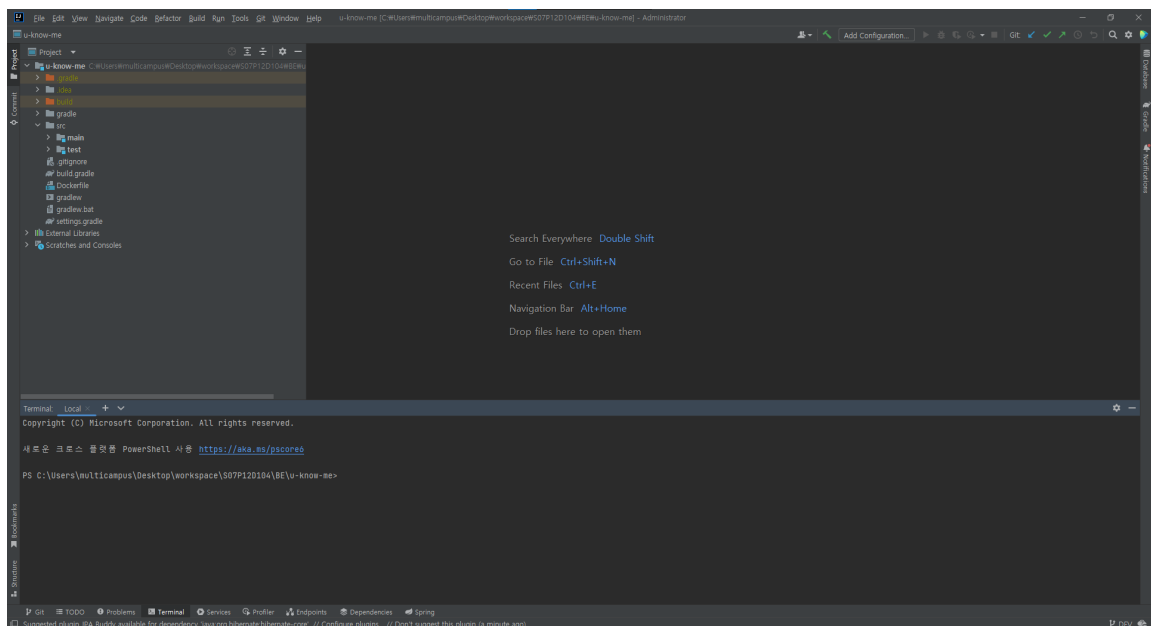


그림 4) 터미널을 연 상태의 IntelliJ

3. 터미널에 `./gradlew clean build` 명령어를 입력합니다.
4. `BUILD SUCCESSFUL` 이 뜨면 `cd build/libs` 명령어를 입력해서 빌드 파일이 있는 위치로 이동한 후 `java -jar u-know-me-0.0.1-SNAPSHOT.jar` 명령어를 입력해줍니다.
5. 스프링부트 서버가 정상적으로 올라가면 <http://localhost:8080/swagger-ui/> 로 접속해서 API를 확인하실 수 있습니다.

3.2.3. 도커 컨테이너를 이용하는 방법

도커를 이용하려면 도커를 미리 설치해주셔야 합니다.

1. `Win + R` 을 누르고 `cmd` 를 입력하고 확인 버튼을 눌러 CMD 창을 띄웁니다.
2. `work directory` 기준 `BE/u-know-me` 디렉토리로 이동해줍니다.
3. `dir` 명령어를 입력한 후 폴더에 `Dockerfile` 이 있는지 확인해줍니다.
4. `docker build -t (도커 허브 아이디)/(이미지 이름) .` 으로 도커 이미지를 생성해줍니다.

```
C:\Users\multicampus\Desktop\workspace\WS07P12D104\BE\u-know-me>docker build -t mungmnb777/be-server .
[+] Building 211.3s (17/17) FINISHED
=> [internal] load build definition from Dockerfile                                0.0s
=> => transferring dockerfile: 684B                                              0.0s
=> [internal] load .dockerignore                                                  0.0s
=> => transferring context: 2B                                                    0.0s
=> [internal] load metadata for docker.io/library/openjdk:8-jdk-alpine          0.0s
=> [internal] load metadata for docker.io/library/gradle:7.2.0-jdk-alpine       2.3s
=> [internal] load build context                                                  0.5s
=> => transferring context: 57.03MB                                              0.5s
=> [temp_build_image 1/8] FROM docker.io/library/gradle:7.2.0-jdk-alpine@sha256:04aeea196cc778a6ee43e62107b917c1 0.0s
=> [stage-1 1/3] FROM docker.io/library/openjdk:8-jdk-alpine                    0.0s
=> CACHED [temp_build_image 2/8] WORKDIR /usr/app                               0.0s
=> [temp_build_image 3/8] COPY build.gradle settings.gradle /usr/app/            0.2s
=> [temp_build_image 4/8] COPY gradle /usr/app/gradle                          0.1s
=> [temp_build_image 5/8] COPY --chown=gradle:gradle . /home/gradle/src         0.2s
=> [temp_build_image 6/8] RUN chown -R gradle /home/gradle/src                 2.6s
=> [temp_build_image 7/8] COPY . .                                              0.2s
=> [temp_build_image 8/8] RUN gradle clean build                                204.5s
=> CACHED [stage-1 2/3] WORKDIR /usr/app                                         0.0s
=> [stage-1 3/3] COPY --from=TEMP_BUILD_IMAGE /usr/app/build/libs/u-know-me-0.0.1-SNAPSHOT.jar . 0.1s
=> exporting to image                                                            0.3s
=> => exporting layers                                                            0.2s
=> writing image sha256:2c4a33f272ab2e7ac50faee5fa4d0d727e36151d69b9a69faf78927db92180bc 0.0s
=> naming to docker.io/mungmnb777/be-server                                     0.0s
```

그림 5) 도커 이미지 빌드

5. `docker run -p 8080:8080 (도커 허브 아이디)/(이미지 이름)` 으로 컨테이너를 생성해줍니다. 그러면 8080번 포트로 서버가 열리게 됩니다!

프론트엔드 웹 서버의 경우 도커 컨테이너를 이용하는 것이 편하기 때문에 도커 컨테이너를 이용한 빌드 방법에 대해서 설명하겠습니다. 방법은 3.2.3 에서 진행한 방법과 똑같습니다.

3.3.1. 도커 컨테이너를 이용한 빌드

1. `Win + R` 을 누르고 `cmd` 를 입력하고 확인 버튼을 눌러 CMD 창을 띄웁니다.
2. `work directory` 기준 `FE/u-know-me` 디렉토리로 이동해줍니다.
3. `dir` 명령어를 입력한 후 폴더에 `Dockerfile` 이 있는지 확인해줍니다.
4. `docker build -t (도커 허브 아이디)/(이미지 이름) .` 으로 도커 이미지를 생성해줍니다. (점까지 붙여야 돼요)
5. `docker run -p 3000:80 (도커 허브 아이디)/(이미지 이름)` 으로 컨테이너를 생성해줍니다. 그러면 3000번 포트로 프론트 웹 서버가 열리게 됩니다!

→ 프론트 빌드 시 알아두어야 할 부분이 있습니다! 프론트 대부분의 로직이 스프링 부트 WAS와 ajax 통신을 통해서 데이터를 가져오게 됩니다. 현재 ajax 통신에서 HOST 설정이 `https://uknowme.mo00.com:8443` 으로 되어있는데, 로컬에서 빌드하시는 경우 이 주소를 변경 해주어야 합니다. (위의 스프링 부트 빌드 방법을 따라하셨다면 `http://localhost:8080` 으로 변경해주시면 됩니다.

```
FROM node:lts-alpine as build-stage
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
RUN npm run build

FROM nginx:stable-alpine as production-stage
COPY --from=build-stage /app/dist /usr/share/nginx/html
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

도커 파일의 내용은 위와 같습니다!

3.4. docker-compose를 이용하여 프론트, 백엔드 동시 빌드

1. `Win + R` 을 누르고 `cmd` 를 입력하고 확인 버튼을 눌러 CMD 창을 띄웁니다.
2. `work directory` 로 이동해줍니다.
3. `docker-compose up --build` 를 통해 실행시켜주시면 프론트엔드, 백엔드 동시에 빌드 및 배포하실 수 있습니다!


```
관리자: 명령 프롬프트 - docker-compose up --build
unknown flag: --build

C:\Users\multicampus\Desktop\workspace\SO7P12D104>docker-compose up --build
[*] Building 3.7s (17/29)
=> [s07p12d104_web-front internal] load build definition from Dockerfile                                0.0s
=> => transferring dockerfile: 309B                                                                    0.0s
=> [s07p12d104_web-front internal] load .dockerignore                                                  0.0s
=> => transferring context: 2B                                                                            0.0s
=> [s07p12d104_web-back internal] load build definition from Dockerfile                                0.0s
=> => transferring dockerfile: 684B                                                                      0.0s
=> [s07p12d104_web-back internal] load .dockerignore                                                  0.0s
=> => transferring context: 2B                                                                            0.0s
=> [s07p12d104_web-front internal] load metadata for docker.io/library/nginx:stable-alpine             2.0s
=> [s07p12d104_web-front internal] load metadata for docker.io/library/node:lts-alpine                 2.1s
=> [s07p12d104_web-back internal] load metadata for docker.io/library/gradle:7.2.0-jdk-alpine           1.6s
=> [s07p12d104_web-back internal] load metadata for docker.io/library/openjdk:8-jdk-alpine              0.0s
=> [s07p12d104_web-back stage-1 1/3] FROM docker.io/library/openjdk:8-jdk-alpine                      0.0s
=> [s07p12d104_web-back temp_build_image 1/8] FROM docker.io/library/gradle:7.2.0-jdk-alpine@sha256:04aaea196cc7 0.0s
=> => transferring context: 57.03MB                                                                      1.2s
=> CACHED [s07p12d104_web-front production-stage 1/2] FROM docker.io/library/nginx:stable-alpine@sha256:98a1e378 0.0s
=> [s07p12d104_web-front build-stage 1/6] FROM docker.io/library/node:lts-alpine@sha256:f94079d73e46f76a250240f4 0.0s
=> [s07p12d104_web-front internal] load build context                                                  1.5s
=> => transferring context: 136.77MB                                                                      1.5s
=> CACHED [s07p12d104_web-back temp_build_image 2/8] WORKDIR /usr/app                                0.0s
=> CACHED [s07p12d104_web-back temp_build_image 3/8] COPY build.gradle settings.gradle /usr/app/         0.0s
=> CACHED [s07p12d104_web-back temp_build_image 4/8] COPY gradle /usr/app/gradle                      0.0s
=> [s07p12d104_web-back temp_build_image 5/8] COPY --chown=gradle:gradle . /home/gradle/src             0.4s
=> [s07p12d104_web-back temp_build_image 6/8] RUN chown -R gradle /home/gradle/src                     0.5s
```

그림 7) 도커 컴포즈를 이용한 서버 빌드

4. 프로젝트 빌드 방법 (운영 서버)

4.1. VSCode를 이용한 ssh 접속

1. Remote - SSH 설치

vscode에서 `ctrl + shift + x` 를 누르면 extensions 탭으로 넘어갈 수 있습니다. 해당 화면에서 검색창에 `ssh` 를 검색하면 `Remote - SSH` 라는 extension이 나오는데 `install` 버튼을 눌러 다운로드 받아줍니다.

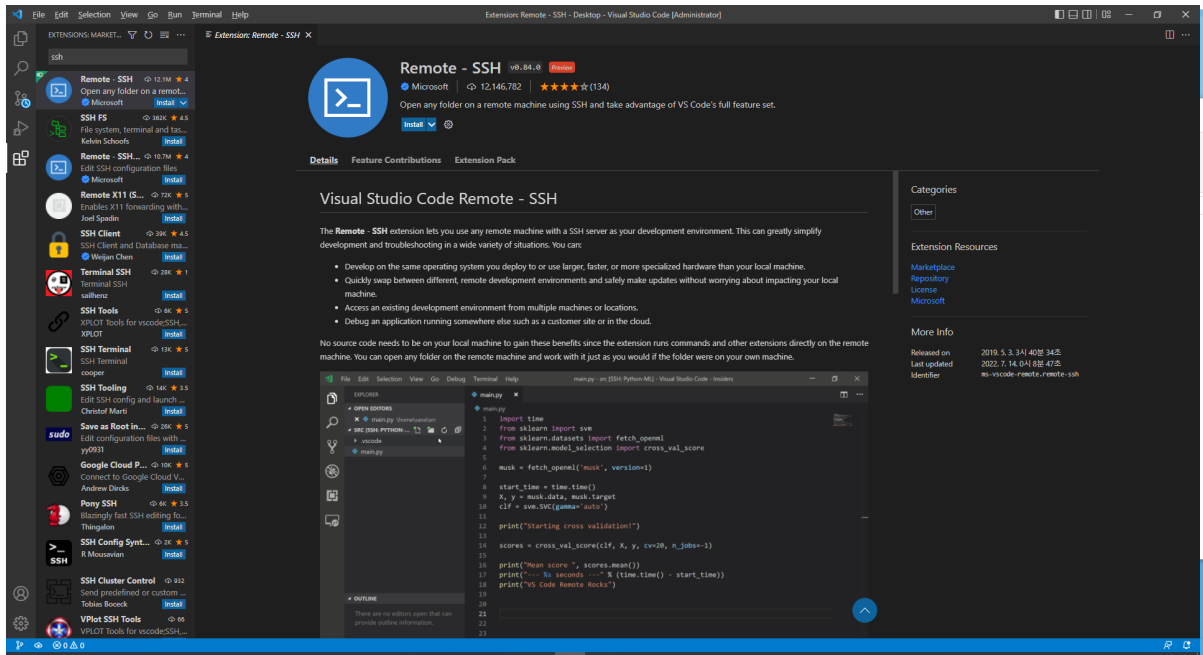


그림 8) extensions 탭

2. SSH 설정 파일 등록

원래 터미널에서 `ssh -i 계정명@IP주소` 로 연결할 수 있지만 계속 터미널에 명령어를 입력하기는 번거로우니 설정 파일을 등록해줍니다.

우선 **f1** 버튼을 눌러 **ssh** 를 검색합니다. 그리고 **Remote-SSH:Open SSH Configuration File...** 이라는 탭을 선택합니다.

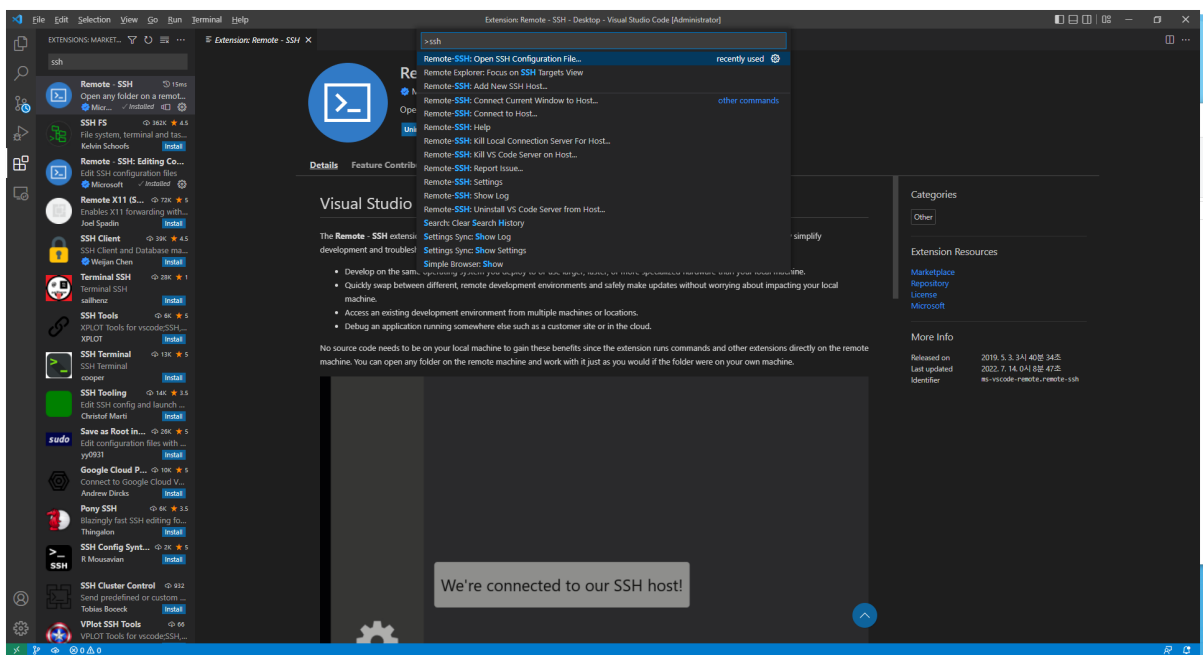


그림 9) ssh 검색

위의 버튼을 눌렀다면 같은 자리에 SSH 구성 파일 리스트가 나열됩니다. 여기서 `C:\Users\<계정명>\.ssh\config` 를 선택합니다.

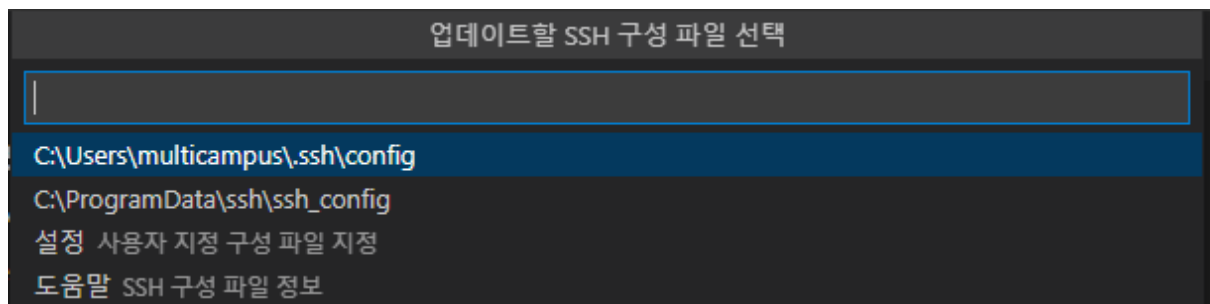


그림 10) SSH 구성 파일 리스트

그러면 SSH 구성 파일이 열립니다.

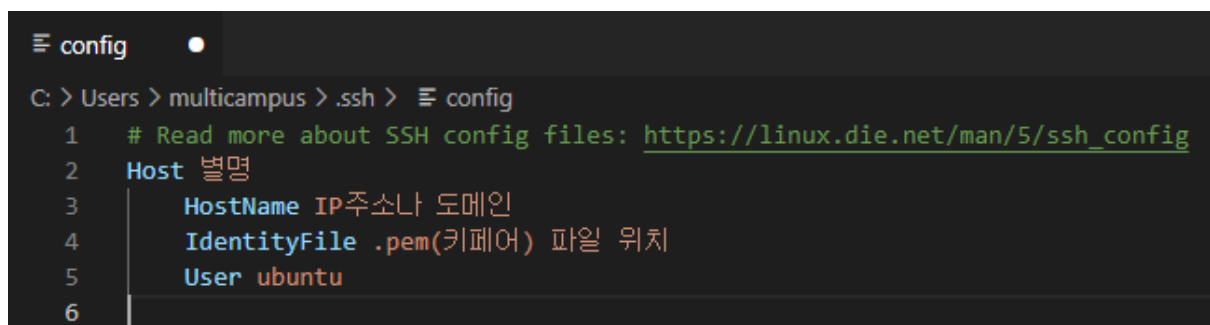


그림 11) SSH 구성 파일

각 요소에 대해 자세히 알아보겠습니다.

- **Host** : **Remote SSH** 의 이름을 설정해주면 됩니다. (해당 인스턴스가 무엇인지 알기 쉽게 이름을 정합니다.)
- **HostName** : AWS EC2 인스턴스의 **public IP** 나 도메인을 적으면 됩니다.
- **IdentityFile** : 현재 **.pem** 파일이 저장되어있는 위치를 작성하면 됩니다.
- **User** : 계정 이름을 설정한다. 우리는 **ubuntu** 를 사용합니다.
- **Port** : 기본값인 22번 포트가 아니라 다른 포트로 ssh 접근을 한다면 입력해줍니다.

3. SSH 세션 접속

입력을 다하고 저장한 후 좌측 탭에서 **Remote Explorer** 탭으로 이동합니다.

SSH TARGET 에 config에서 설정한 Host명으로 아이콘이 하나 생깁니다. Host명 우측의 폴더 아이콘을 클릭합니다.

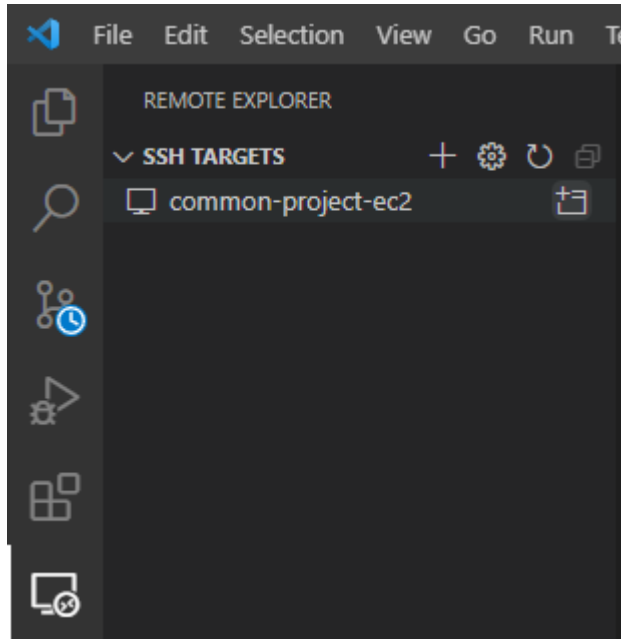


그림 12) Remote Explorer

그럼 새 vscode 창이 뜨면서 Linux, Windows, macOS를 선택하는 창이 나옵니다. 우리는 우분투를 사용하기 때문에 리눅스를 선택합니다.

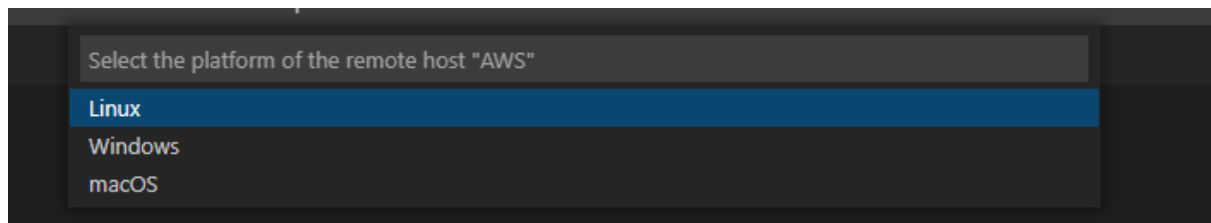


그림 13) AWS 플랫폼 선택창

그러면 SSH를 이용하여 AWS EC2 인스턴스를 vscode에서 편집할 수 있게 됩니다.

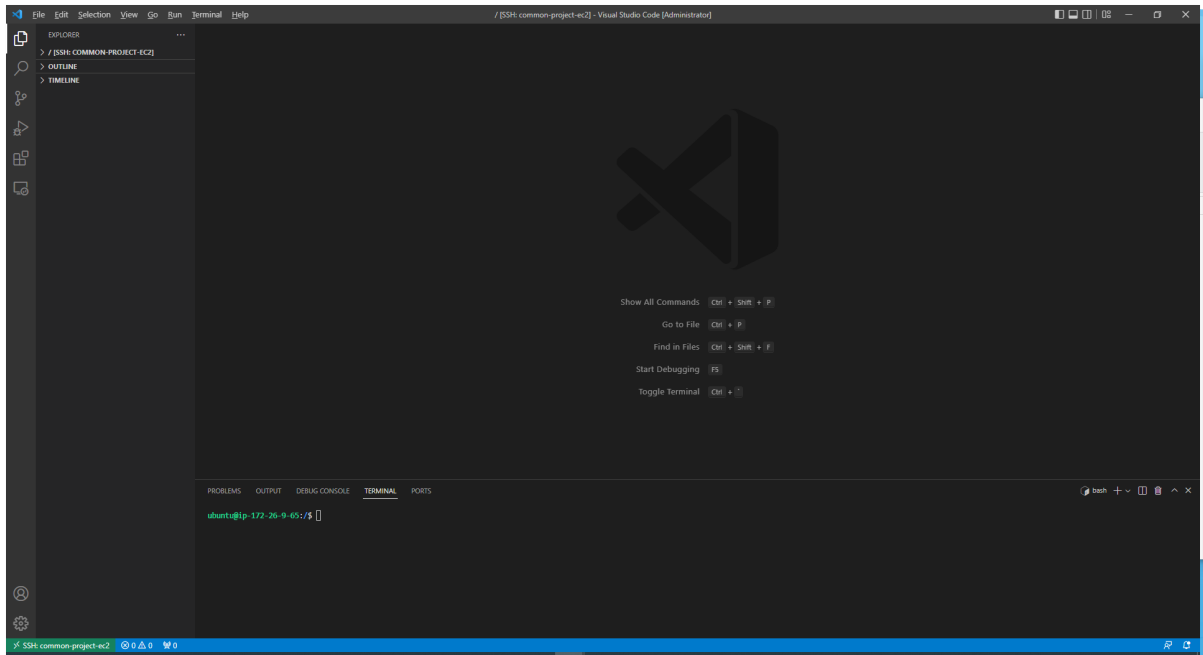


그림 14) Remote SSH에 연결된 모습

4.2. 무료 DNS 등록

1. EC2 터미널에서 `curl ifconfig.me` 명령어를 입력하고 현재 공인 IP가 무엇인지 확인합니다.
2. <https://freedns.afraid.org/>에 접속해줍니다.
3. 왼쪽 For Members 탭에서 Subdomains를 클릭합니다.

For Members:		
[Main Menu]
[Domains]
[Subdomains]
[Web Forward]
[Dynamic DNS]
[IPv6 Reverse]
[Backup DNS]
[Preferences]
[Registry]
[Logout]

그림 15) 서브 도메인

4. 로그인 후 입력란을 다 입력해줍니다.

Add a new subdomain

Type: A [explanation](#)


Subdomain:

Domain: mooo.com (public)

Destination: 14.46.141.226

TTL: For our premium supporter seconds (optional)

Wildcard: ☐ Enabled for all subscribers ([more info](#))



[\[Different Image \]](#)

Save!

그림 16) 서브 도메인 생성

- Type - 그대로 해줍니다.
 - Subdomain - 원하시는 도메인 이름으로 설정해주시면 됩니다!
 - Domain - 이 부분도 원하시는 것을 선택해주시면 됩니다!
 - Destination - 아까 처음에 확인했던 공인 IP를 등록해주시면 됩니다.
- 입력란을 모두 채운 후에 Save 버튼을 눌러 저장해줍니다. 그러면 이후에 **Subdomain** + **Domain** 주소를 입력하실 수 있게 됩니다!
- 저희의 경우에는 Subdomain을 **uknowme**, Domain을 **mooo.com** 으로 조합하여 **uknowme.mooo.com** 으로 설정하였습니다.

4.3. Let's encrypt 인증서 발급

우리 프로젝트에서는 opnvidu를 사용하기 위해 ssl 인증서를 발급받아야 했습니다. 저희는 let's encrypt 인증서 발급 방식 중 **standalone** 방식을 이용하여 인증서를 발급받았습니다.

이 방식은 80번 포트로 가상 standalone 웹 서버를 띄워 인증서를 발급받는 방식으로 동시에 여러 도메인에 대해 인증서를 발급받을 수 있다는 장점이 있지만 인증서 발급 전에 nginx 서버를 중단해야 한다는 단점이 있습니다.

```
sudo apt update

// letsencrypt 패키지 설치
sudo apt-get install letsencrypt -y
```

```
// 실행중인 nginx 종료
service nginx stop

// SSL 인증
certbot certonly --standalone -d uknowme.mo00.com
```

certbot 명령을 실행시켰을 때 만약 인증서가 없다면 인증서를 발급받는 과정을 거칩니다.

```
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Plugins selected: Authenticator standalone, Installer None
Enter email address (used for urgent renewal and security notices) (Enter 'c' to cancel)
:
```

다음은 서비스 약관에 동의하는지 묻습니다. 동의 해줍니다.

```
Please read the Terms of Service at
https://letsencrypt.org/documents/LE-SA-v1.2-November-15-2017.pdf. You must agree in o
rder to register with the ACME server at https://acme-v02.api.letsencrypt.org/directo
ry

(A)gree/(C)ancel:
```

다음은 이메일 주소를 공유할 것인지를 묻습니다. 공유한다면 Y, 아니면 N을 입력하면 됩니다.

```
Would you be willing to share your email address with the Electronic Frontier Foundat
ion, 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 we
b, EFF news, campaigns, and ways to support digital freedom.

(Y)es/(N)o:
```

인증 완료 후 `certbot certificates` 명령어를 통해 제대로 발급이 되었는지 확인해줍니다.

```
● ubuntu@ip-172-26-9-65:/$ sudo certbot certificates
Saving debug log to /var/log/letsencrypt/letsencrypt.log

-----
Found the following certs:
Certificate Name: unknowme.moood.com
Domains: unknowme.moood.com
Expiry Date: 2022-10-23 14:27:25+00:00 (VALID: 67 days)
Certificate Path: /etc/letsencrypt/live/unknowme.moood.com/fullchain.pem
Private Key Path: /etc/letsencrypt/live/unknowme.moood.com/privkey.pem
-----
```

그림 17) certbot 인증서 확인

4.4. MySQL 빌드

MySQL 도커 파일은 프로젝트에서 따로 제공하지 않습니다. `Dockerfile` 및 설정 파일들은 직접 생성해주셔야 합니다.

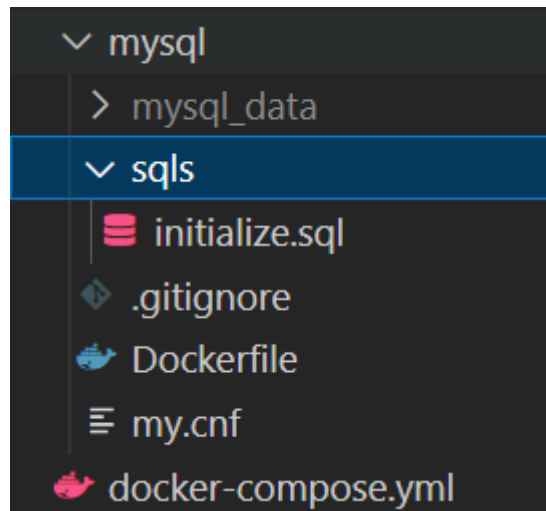


그림 18) MySQL 도커 설정

MySQL 도커 빌드에 필요한 파일들은 `my.cnf` 라는 설정 파일, 도커 이미지를 빌드하기 위한 `Dockerfile`, 초기 데이터베이스 생성에 필요한 `initialize.sql` 이 필요합니다. `.gitignore` 의 경우 추후에 깃에 올리기 위해서 사용하는 파일인데, mysql 데이터를 깃에서 무시하기 위해 사용합니다.

각각의 파일에 들어가는 코드는 다음과 같습니다.

- **Dockerfile**

```
FROM mysql:8.0.28

ADD ./my.cnf /etc/mysql/conf.d/my.cnf

EXPOSE 3306
```


- **initialize.sql**

```
DROP DATABASE IF EXISTS unknowme;

CREATE DATABASE unknowme;
```

- **my.cnf**

```
[mysqld]
character-set-server=utf8

[mysql]
default-character-set=utf8

[client]
default-character-set=utf8
```

- **docker-compose.yml**

```
version: "3"

services:
  db-mysql:
    build: ./mysql
    ports:
      - "3306:3306"
    restart: unless-stopped
    volumes:
      - ./mysql/mysql_data:/var/lib/mysql
      - ./mysql/qls/./docker-entrypoint-initdb.d/
    environment:
      MYSQL_ROOT_PASSWORD: susung!susung@12
      MYSQL_DATABASE: unknowme
```

모든 파일을 생성한 후에 docker-compose.yml이 있는 위치에서 `docker-compose up --build` 으로 컨테이너를 생성해주면 MySQL 컨테이너가 서버에 올라가고 DB에 접근할 수 있게 됩니다.

저희 프로젝트에서는 JPA를 이용하기 때문에 스프링 부트 서버를 실행하면 스키마가 생성하여 DB를 사용할 수 있게 됩니다!

4.5. openvidu 서버 빌드 방법

저희는 openvidu 배포를 on premises 방법으로 배포하였습니다. 자세한 내용은 공식 문서를 참고하였습니다.

Docker 와 Docker Compose 를 설치한 후 공식 문서를 따라 설치해주시면 됩니다.

- Ubuntu 도커 설치 방법 - <https://rootkey.tistory.com/142>
- Ubuntu 도커 컴포즈 설치 방법 - <https://makepluscode.tistory.com/73>

설치가 끝나셨다면 설정이 필요합니다. 공식 문서를 따라 openvidu를 설치하셨다면 `cd /opt/openvidu` 명령어를 통해 `.env` 파일이 있는 위치로 이동합니다.

```
● ubuntu@ip-172-26-9-65:/opt/openvidu$ ll
total 96
drwxr-xr-x 12 root root 4096 Aug 15 15:59 ./
drwxr-xr-x 10 root root 4096 Aug 17 07:58 ../
-rw-r--r--  1 root root 8911 Aug 15 10:54 .env
-rw-r--r--  1 root root 8911 Aug 15 15:59 .env.save
drwxr-xr-x  2 root root 4096 Jul 30 13:50 cdr/
drwxrwxrwx  9 root root 4096 Aug 17 00:00 certificates/
drwxr-xr-x  2 root root 4096 Jul 30 13:50 coturn/
drwxr-xr-x  2 root root 4096 Jul 30 13:50 custom-layout/
drwxr-xr-x  2 root root 4096 Jul 30 13:49 custom-nginx-locations/
drwxr-xr-x  2 root root 4096 Jul 30 13:49 custom-nginx-vhosts/
-rw-r--r--  1 root root  825 Jul 30 13:49 docker-compose.override.yml
-rw-r--r--  1 root root 4409 Jul 30 13:49 docker-compose.yml
-rw-r--r--  1 root root    0 Aug 15 03:02 env
drwxr-xr-x  2 root root 4096 Jul 30 13:50 kms-crashes/
drwxr-xr-x  2 root root 4096 Aug 17 00:00 kurento-logs/
-rwxr-xr-x  1 root root 9331 Jul 30 13:49 openvidu*
drwxr-xr-x  2 root root 4096 Jul 30 13:49 owncert/
drwxr-xr-x  2 root root 4096 Jul 30 13:49 recordings/
○ ubuntu@ip-172-26-9-65:/opt/openvidu$
```

그림 19) openvidu 작업 디렉토리

그 후 `sudo nano .env` 를 입력합니다. 그러면 다음과 같은 화면이 등장합니다.

```
GNU nano 4.8 .env
# OpenVidu configuration
# -----
# Documentation: https://docs.openvidu.io/en/stable/reference-docs/openvidu-config/

# NOTE: This file doesn't need to quote assignment values, like most shells do.
# All values are stored as-is, even if they contain spaces, so don't quote them.

# Domain name. If you do not have one, the public IP of the machine.
# For example: 198.51.100.1, or openvidu.example.com
DOMAIN_OR_PUBLIC_IP=i7d104.p.ssafy.io

# OpenVidu SECRET used for apps to connect to OpenVidu server and users to access to OpenVidu Dashboard
OPENVIDU_SECRET=MY_SECRET

# Certificate type:
# - selfsigned: Self signed certificate. Not recommended for production use.
#               Users will see an ERROR when connected to web page.
# - owncert:    Valid certificate purchased in a Internet services company.
#               Please put the certificates files inside folder ./owncert
#               with names certificate.key and certificate.cert
# - letsencrypt: Generate a new certificate using letsencrypt. Please set the
#               required contact email for Let's Encrypt in LETSENCRYPT_EMAIL
#               variable.
CERTIFICATE_TYPE=letsencrypt

# If CERTIFICATE_TYPE=letsencrypt, you need to configure a valid email for notifications
LETSENCRYPT_EMAIL=mungmnb777@gmail.com

# Proxy configuration
# If you want to change the ports on which openvidu listens, uncomment the following lines

# Allows any request to http://DOMAIN_OR_PUBLIC_IP:HTTP_PORT/ to be automatically
[ File '.env' is unwritable ]
^G Get Help  ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos    M-U Undo      M-A Mark Text
^X Exit      ^R Read File  ^\ Replace    ^U Paste Text ^T To Spell   ^_ Go To Line   M-E Redo      M-6 Copy Text
```

그림 20) openvidu .env 설정 파일

- **DOMAIN_OR_PUBLIC_IP** : 서버의 도메인 네임을 작성하시면 됩니다.
- **OPENVIDU_SECRET** : openvidu 서버에 접속할 때 사용할 키 값을 입력하시면 됩니다.
- **CERTIFICATE_TYPE** : 저희는 letsencrypt를 발급 받아 사용할 것이기 때문에 **letsencrypt**를 사용해줍니다.
- **LETSENCRYPT_EMAIL** : letsencrypt를 설정할 이메일을 작성해줍니다.

여기까지만 작성하면 openvidu 서버를 작동할 수 있습니다. 하지만 이대로 작동시키는 경우 HTTP 포트가 80 번, HTTPS 포트가 443 번으로 열리기 때문에 포트 수정이 필요합니다. 아까 .env 파일에서 조금 내려가보면 **HTTP_PORT** 설정과 **HTTPS_PORT** 설정 부분이 있습니다.

```
GNU nano 4.8 .env
# - owncert: Valid certificate purchased in a Internet services company.
# Please put the certificates files inside folder ./owncert
# with names certificate.key and certificate.cert
# - letsencrypt: Generate a new certificate using letsencrypt. Please set the
# required contact email for Let's Encrypt in LETSENCRYPT_EMAIL
# variable.
CERTIFICATE_TYPE=letsencrypt

# If CERTIFICATE_TYPE=letsencrypt, you need to configure a valid email for notifications
LETSENCRYPT_EMAIL=mungmnb777@gmail.com

# Proxy configuration
# If you want to change the ports on which openvidu listens, uncomment the following lines

# Allows any request to http://DOMAIN_OR_PUBLIC_IP:HTTP_PORT/ to be automatically
# redirected to https://DOMAIN_OR_PUBLIC_IP:HTTPS_PORT/.
# WARNING: the default port 80 cannot be changed during the first boot
# if you have chosen to deploy with the option CERTIFICATE_TYPE=letsencrypt
# HTTP_PORT=80

# Changes the port of all services exposed by OpenVidu.
# SDKs, REST clients and browsers will have to connect to this port
HTTPS_PORT=4443

# Old paths are considered now deprecated, but still supported by default.
# OpenVidu Server will log a WARN message every time a deprecated path is called, indicating
# the new path that should be used instead. You can set property SUPPORT_DEPRECATED_API=false
# to stop allowing the use of old paths.
# Default value is true
# SUPPORT_DEPRECATED_API=true

# If true request to with www will be redirected to non-www requests

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos M-U Undo M-A Mark Text
^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell ^_ Go To Line M-E Redo M-6 Copy Text
```

그림 21) openvidu .env 설정 파일

`HTTP_PORT` 는 80번으로 그대로 두고, `HTTPS_PORT` 를 4443번으로 변경해줍니다. 작성을 완료 하셨다면 `Ctrl + x` 를 누르고 Y/N이 나오면 `y`, 그 후 `Enter` 를 누르시면 변경 사항이 저장 이 됩니다.

그리고 다시 우분투 화면으로 돌아오셔서 `./openvidu start` 명령어를 입력하시면 openvidu 서버가 빌드되면서 배포할 수 있습니다!


```

        location /.well-known/acme-challenge/ {
            root /var/www/certbot;
        }
    }

    server {
        listen 443 ssl;
        server_name unknowme.mo00.com;
        root /usr/share/nginx/html;

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

        ssl_certificate /etc/letsencrypt/live/unknowme.mo00.com/fullchain.pem;
        ssl_certificate_key /etc/letsencrypt/live/unknowme.mo00.com/privkey.pem;
    }
}

```

→ 입력 후 **esc** 를 누르고 **!wq** 를 누르시면 저장됩니다. **cat default.conf** 를 입력하시면 저장이 되었는지 확인하실 수 있습니다.

```

● ubuntu@ip-172-26-9-65:/opt/common-project/volumes-nginx/conf.d$ cat default.conf
server {
    listen      80;
    server_name unknowme.mo00.com;

    location / {
        return 301 https://unknowme.mo00.com:4443/$request_uri;
    }

    location /.well-known/acme-challenge/ {
        root /var/www/certbot;
    }
}

server {
    listen      443 ssl;
    server_name unknowme.mo00.com;
    root        /usr/share/nginx/html;

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

    ssl_certificate /etc/letsencrypt/live/unknowme.mo00.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/unknowme.mo00.com/privkey.pem;
}

○ ubuntu@ip-172-26-9-65:/opt/common-project/volumes-nginx/conf.d$ █

```

그림 23) nginx 설정 파일이 저장된 모습

3. 위의 과정을 완료하셨으면 `docker-compose.yml` 파일이 있는 위치로 이동해줍니다. 저의 경우에는 `/opt/common-project` 에서 깃 클론을 하였으므로 `/opt/common-project/common-project` 위치에 파일이 있습니다.

```
ubuntu@ip-172-26-9-65:/opt/common-project$ cd common-project/  
ubuntu@ip-172-26-9-65:/opt/common-project/common-project$ ls  
BE FE deploy_archive docker-compose.yml  
ubuntu@ip-172-26-9-65:/opt/common-project/common-project$
```

그림 24) 작업 디렉토리 위치

4. 해당 위치에서 `docker-compose up --build` 명령어를 입력해줍니다.

```
ubuntu@ip-172-26-9-65:/opt/common-project/common-project$ docker-compose up --build  
Building web-front  
Step 1/10 : FROM node:lts-alpine as build-stage  
--> b0cbdedc1b9d  
Step 2/10 : WORKDIR /app  
--> Using cache  
--> 89b53d3f07ee  
Step 3/10 : COPY package*.json ./  
--> Using cache  
--> c4817947586d  
Step 4/10 : RUN npm install  
--> Using cache  
--> f202b2762b1d  
Step 5/10 : COPY . .  
--> Using cache  
--> f4233a681ea9  
Step 6/10 : RUN npm run build  
--> Using cache  
--> b41827343960  
Step 7/10 : FROM nginx:stable-alpine as production-stage  
--> ccb911fdd2ca  
Step 8/10 : COPY --from=build-stage /app/dist /usr/share/nginx/html  
--> Using cache  
--> 4f37bafaa9f0  
Step 9/10 : EXPOSE 80  
--> Using cache  
--> 2472da76b330  
Step 10/10 : CMD ["nginx", "-g", "daemon off;"]  
--> Using cache  
--> d65e390008bc  
Successfully built d65e390008bc  
Successfully tagged common-project_web-front:latest  
Building web-back  
Step 1/17 : FROM gradle:7.2.0-jdk-alpine AS TEMP_BUILD_IMAGE  
--> b67899086bd6  
Step 2/17 : ENV APP_HOME=/usr/app
```

그림 25) Docker Compose로 빌드되는 모습

알아두셔야 할 부분이 있습니다!!!!

현재 저희 프로젝트의 도메인은 `uknowme.mo00.com` 을 기준으로 개발되었습니다! 혹여나 다른 도메인으로 진행하시게 되면 스프링 부트 서버의 설정 파일과 프론트엔드의 코드를 수정해야 하므로 조금 번거롭습니다. 따라서 가능하다면 도메인 이름을 `uknowme.mo00.com` 으로 하여야 편하게 개발하실 수 있습니다!

5. 소셜 로그인 설정

저희 서비스는 소셜 로그인을 제공하고 있습니다. 네이버와 카카오 소셜 로그인 설정하는 법에 대해 알아보겠습니다.

5.1. 네이버 소셜 로그인

1. 우선 네이버 개발자 센터로 접속해줍니다.
2. 로그인 후 위의 탭에서 **Application** - **애플리케이션 등록** 으로 들어가줍니다.
3. 애플리케이션 이름과 사용 API를 등록해줍니다. 사용 API의 경우 저희는 소셜 로그인을 사용할 것이기 때문에 **네이버 로그인** 을 선택해주시면 됩니다. 그리고 체크 박스가 뜰 텐데 **회원 이름** 과 **휴대전화번호** 만 체크해줍니다.

NAVER Developers Products Documents Application NAVER D2 Support Forum API 상태 Search Here

내 애플리케이션
애플리케이션 등록
API 제휴 신청
계정 설정

애플리케이션 등록 (API 이용신청)

애플리케이션의 기본 정보를 등록하면, 좌측 내 애플리케이션 메뉴의 서브 메뉴에 등록하신 애플리케이션 이름으로 서브 메뉴가 만들어집니다.

애플리케이션 이름

- 네이버 로그인할 때 사용자에게 표시되는 이름이므로 서비스 브랜드를 대표할 수 있는 이름으로 가급적 10자 이내로 간결하게 설정해주세요.
- 40자 이내의 영문, 한글, 숫자, 공백문자, 점표(.), "/", "-". 만 입력 가능합니다.

선택하세요. ▼ ✓

네이버 로그인

제공 정보 선택(이용자 식별자는 기본 정보로 제공) ⓘ

필수 항목은 개인정보보호법 제3조 제1항, 제16조 제1항 등에 따라 서비스 제공을 위해 필요한 최소한의 개인정보만을 선택해야 합니다.

권한	필수	추가
회원이름	<input checked="" type="checkbox"/>	<input type="checkbox"/>
이메일 주소	<input type="checkbox"/>	<input type="checkbox"/>
별명	<input type="checkbox"/>	<input type="checkbox"/>
프로필 사진	<input type="checkbox"/>	<input type="checkbox"/>
성별	<input type="checkbox"/>	<input type="checkbox"/>
생일	<input type="checkbox"/>	<input type="checkbox"/>
연령대	<input type="checkbox"/>	<input type="checkbox"/>
출생연도	<input type="checkbox"/>	<input type="checkbox"/>
휴대전화번호	<input checked="" type="checkbox"/>	<input type="checkbox"/>

[알림] 추가권한에 대한 네이버 로그인 공지사항을 확인하세요.

그림 26) 네이버 개발자 센터 - 네이버 로그인

4. 그 후 아래의 로그인 오픈 API 서비스 환경에서 **PC 웹**을 선택해줍니다.
5. 그러면 서비스 URL과 Callback URL을 작성하는 입력란이 등장하는데 서비스 URL의 경우 로컬에서 테스트할 때는 **http://localhost**를 입력해주시면 되고, 운영 환경에서는 **https://uknowme.mo00.com**으로 작성하시면 됩니다.
6. Callback URL은 로컬 환경에서는 **http://localhost:8080/member/oauth2/code/naver**, 운영 환경에서는 **https://uknowme.mo00.com:8443/member/oauth2/code/naver**으로 작성해줍니다.

로그인 오픈 API 서비스 환경 ②

환경 추가 ▼

PC 웹 × ^

서비스 URL

https://uknowme.mo00.com

서비스 URL예시: (O) http://naver.com (X) http://www.naver.com
서비스 URL값이 잘못 입력되어 있으면 정확한 값으로 수정하실 때 까지 네이버 로그인 사용이 일시적으로 제한됩니다.
불법/음란성 사이트 등 이용약관에 위배되는 사이트의 경우, 이용이 제한될 수 있습니다.
서비스하려는 사이트 URL과 동일한 사이트 URL로 해주셔야 **네이버 로그인 뱃지**가 노출됩니다.

네이버 로그인 Callback URL (최대 5개)

https://uknowme.mo00.com:8443/member/oauth2/code/naver -

http://localhost:8080/member/oauth2/code/naver +

텍스트 폼 우측 끝의 '+' 버튼을 누르면 행이 추가되며, '-' 버튼을 누르면 행이 삭제됩니다.
Callback URL은 네이버 로그인 후 이동할 페이지 URL입니다. Callback URL값이 잘못 입력되어 있으면 정확한 값으로 수정하실 때 까지 네이버 로그인 사용이 일시적으로 제한됩니다.
입력한 주소와 다른 Callback URL로 리다이렉트 될 경우, 이용이 제한될 수 있습니다.

로고 이미지 ⇄

파일선택

My APP

네이버 로그인 연동 과정에서 사용자에게 보여지는 이미지이므로 서비스를 대표할 수 있는 이미지로 설정해주세요.
권장 크기는 140X140 사이즈이며 500KB 이하의 jpg, png, gif만 등록 가능합니다.

그림 27) 네이버 개발자 센터 - 네이버 로그인

여기까지 네이버 개발자 센터에서의 설정은 끝났습니다! 다음은 저희 어플리케이션 코드 부분에서 살짝 수정이 필요합니다.

7. 우선 스프링 부트 `application.properties` 에서 설정을 살짝 수정해주셔야 합니다.

```
# registration
## naver
spring.security.oauth2.client.registration.naver.client-id=YQdwIoQRJWLg8GBYAaZq
spring.security.oauth2.client.registration.naver.client-secret=
spring.security.oauth2.client.registration.naver.client-authentication-method=post
spring.security.oauth2.client.registration.naver.redirect-uri=https://uknowme.mooo.com:8443/member/oauth2/code/naver
spring.security.oauth2.client.registration.naver.authorization-grant-type=authorization_code
spring.security.oauth2.client.registration.naver.scope=name, mobile
spring.security.oauth2.client.registration.naver.client-name=Naver
```

그림 28) 스프링부트 `application.properties` 파일 - 네이버

개발자 센터 설정이 끝나고 내 애플리케이션으로 들어가면 저희 서비스의 개요를 볼 수 있습니다.

너나알아

개요	API 설정	네이버 로그인 검수상태	멤버관리	로그인 통계	API 통계	Playground(Beta)
----	--------	-----------------	------	--------	--------	------------------

애플리케이션 정보

Client ID	<input type="text" value="YQdwIoQRJWLg8GBYAaZq"/>
Client Secret	<div><input type="password" value="....."/> 보기</div>

그림 29) 네이버 개발자 센터 - 애플리케이션 정보

이 곳에 나와있는 `Client ID` 와 `Client Secret` 를 `application.properties`에 입력해줍니다. 각각 `client-id`, `client-secret`에 입력해주시면 돼요.

5.2. 카카오 소셜 로그인

1. [Kakao Developers](#) 홈페이지로 들어가줍니다.
2. 로그인 후 **내 애플리케이션** 탭에 들어가줍니다. 애플리케이션 추가하기 버튼을 누르고 **앱 이름** 과 **사업자명** 을 작성하고 저장 버튼을 눌러줍니다.

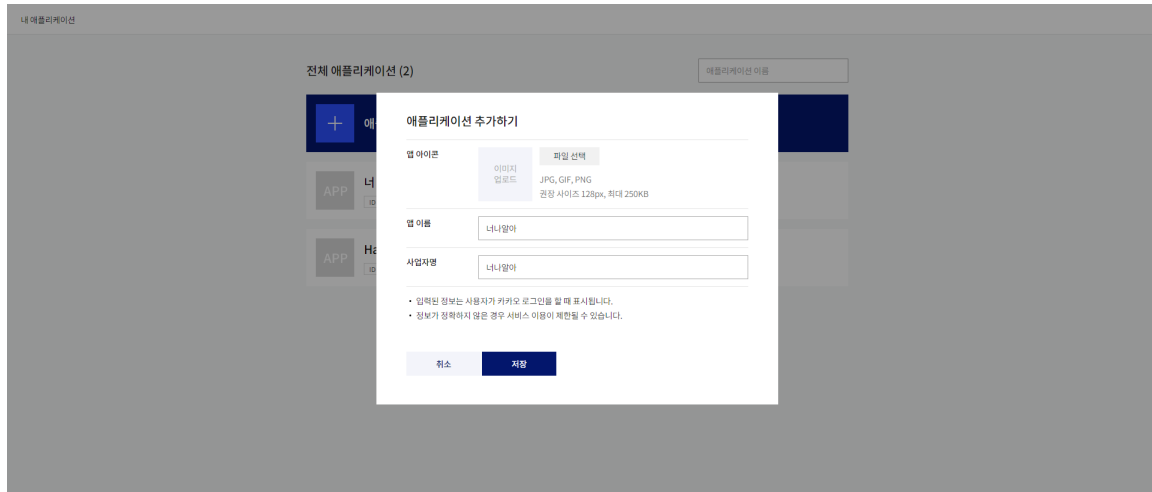


그림 30) 카카오 개발자 센터 - 애플리케이션 추가하기

- 그리고 생성한 애플리케이션 설정으로 들어가셔서 왼쪽의 **카카오 로그인** 을 눌러줍니다. 그리고 활성화 설정을 **ON** 으로 변경해줍니다.

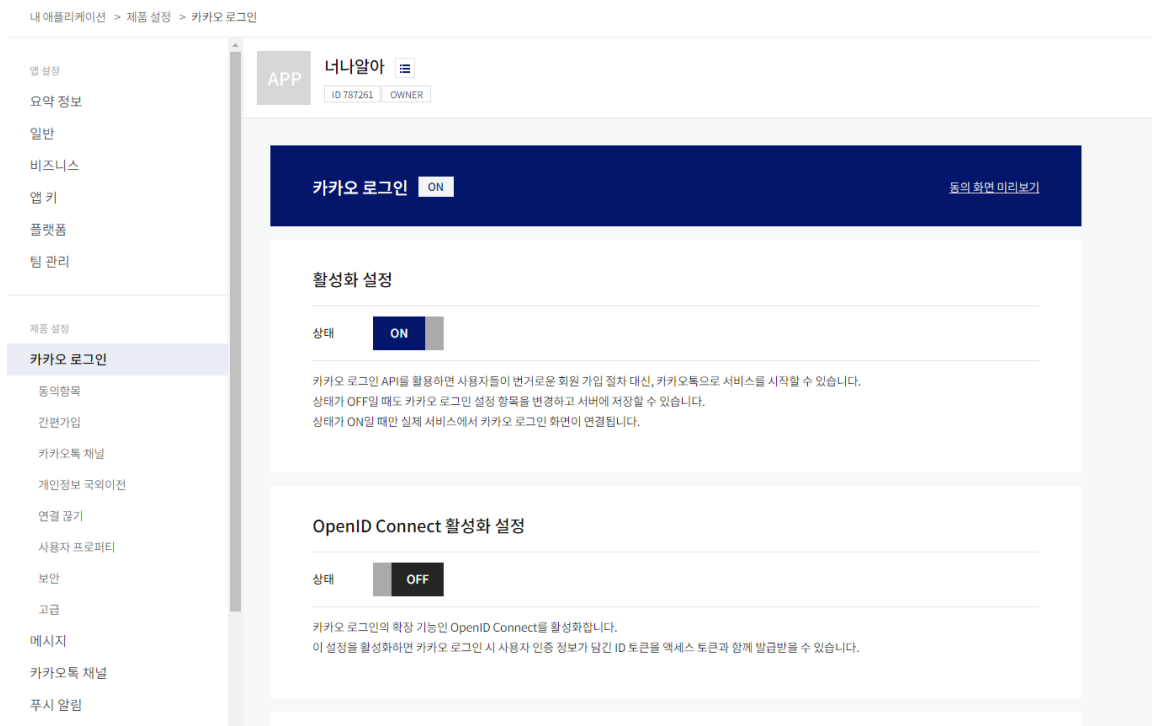


그림 31) 카카오 로그인 활성화

- 아래로 내려가서 Redirect URL도 설정해줍니다. 이 부분은 네이버 소셜 로그인 할 때와 비슷합니다. 로컬일 때는 `http://localhost:8080/member/oauth2/code/kakao` 로 설정해주시고 운영 환경에서는 `https://uknowme.moou.com:8443/member/oauth2/code/kakao` 로 설정해 주시면 됩니다.

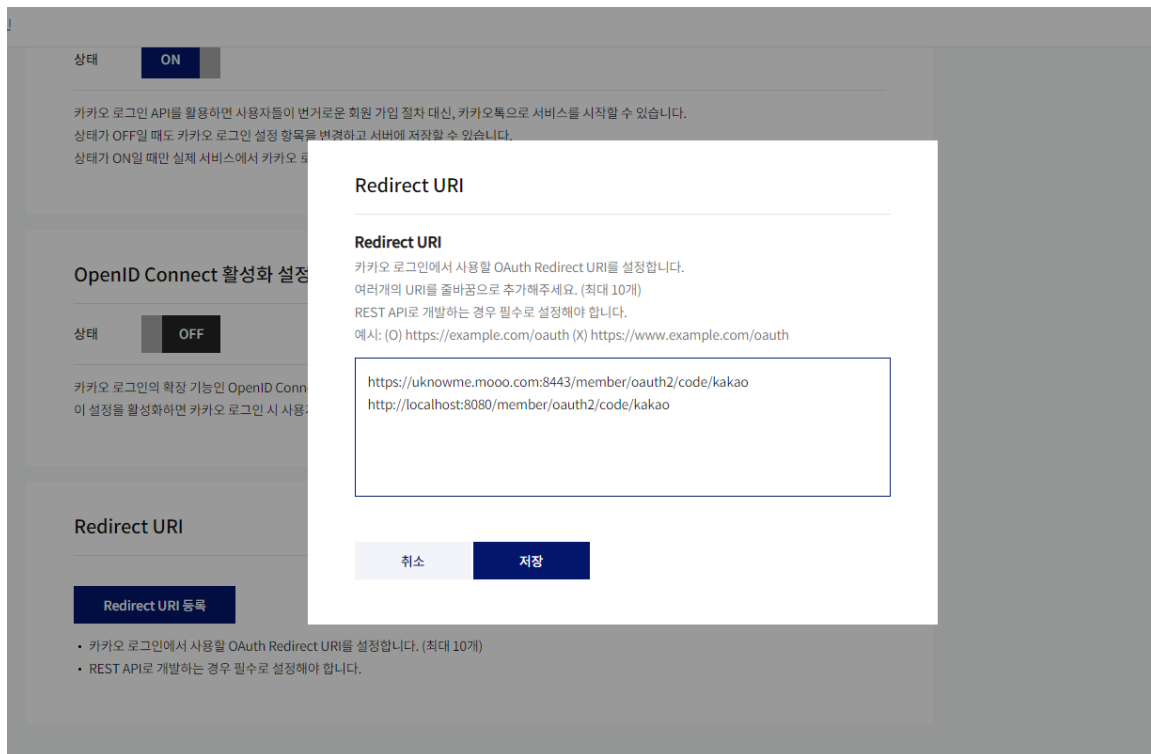


그림 32) Redirect URI 설정

5. 그리고 동의항목을 체크해줍니다. 저희 서비스는 **생일** 에다가 체크해주시면 됩니다.

카카오 로그인
ON
동의 화면 미리보기

동의항목

카카오 로그인으로 서비스를 시작할 때 동의 받는 항목을 설정합니다. 미리 보기를 통해 사용자에게 보여질 화면을 확인할 수 있습니다.

사업자 정보를 등록하여 비즈 앱으로 전환하고 비즈니스 채널을 연결하면 권한이 없는 동의 항목에 대한 검수 신청을 할 수 있습니다.

비즈니스 설정 바로가기

개인정보

항목 이름	ID	상태
닉네임	profile_nickname	● 사용 안함 설정
프로필 사진	profile_image	● 사용 안함 설정
카카오계정(이메일)	account_email	● 사용 안함 설정
성별	gender	● 사용 안함 설정
연령대	age_range	● 사용 안함 설정
생일	birthday	● 사용 안함 설정
출생 연도	birthyear	○ 권한 없음

그림 33) 동의항목 설정

여기까지 하셨으면 kakao developers에서 설정은 모두 끝났습니다.

- 네이버 소셜 로그인에서 한 것과 마찬가지로 `application.properties` 설정을 변경해주어야 합니다.

```
## kakao
spring.security.oauth2.client.registration.kakao.client-id=eeb1404c08508f16f1ff0f59d33806fe
spring.security.oauth2.client.registration.kakao.client-secret=
spring.security.oauth2.client.registration.kakao.client-authentication-method=post
spring.security.oauth2.client.registration.kakao.redirect-uri=http://localhost:8080/member/oauth2/code/kakao
spring.security.oauth2.client.registration.kakao.authorization-grant-type=authorization_code
spring.security.oauth2.client.registration.kakao.scope=birthday
spring.security.oauth2.client.registration.kakao.client-name=Kakao
```

그림 34) 스프링부트 `application.properties` 파일 - 카카오

`client-id` 는 좌측 **요약 정보** 탭을 누르시고 **REST API 키** 를 입력해주시면 됩니다.

`client-secret` 은 좌측 **카카오 로그인** - 보안 탭을 들어가셔서 Client Secret의 코드를 입력해주시면 됩니다.