E207 : 상어

삼성 SW 청년 아카데미 7기 부울경캠퍼스
공통프로젝트 (07.05 ~ 08.19)

포팅 매뉴얼

담당 컨설턴트 : 최호근
이상진(팀장), 김준구, 인예림, 정민지

<<목차>>

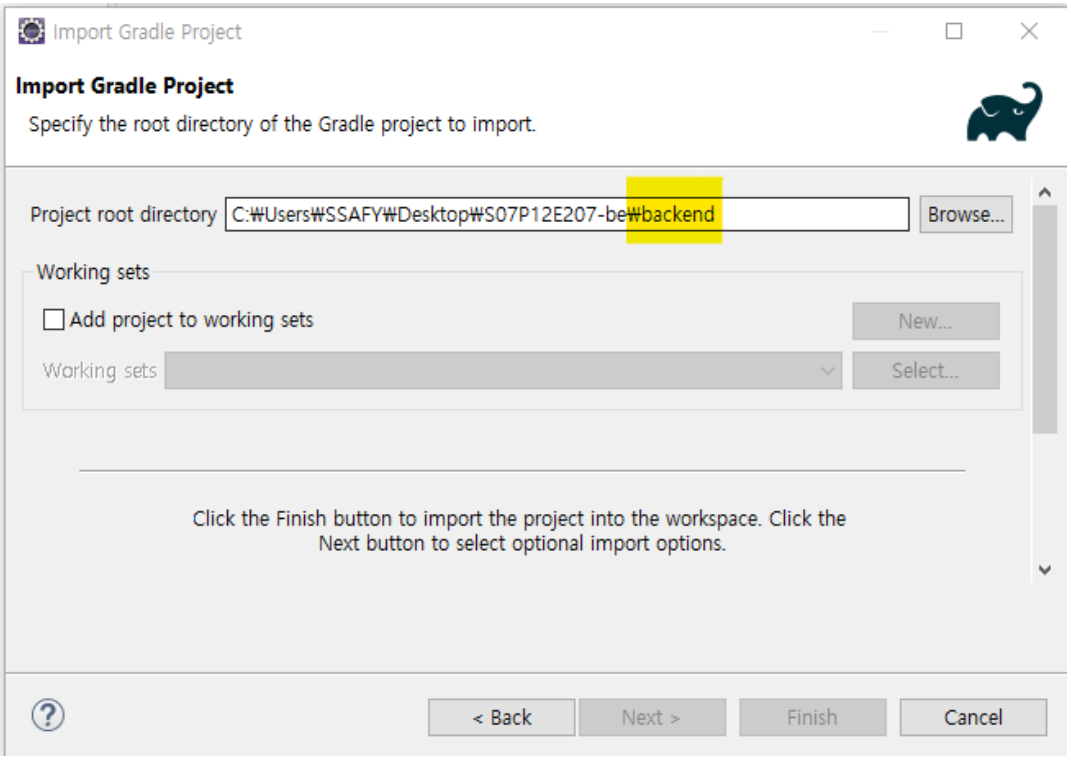
---

1. 프로젝트 기술 스택

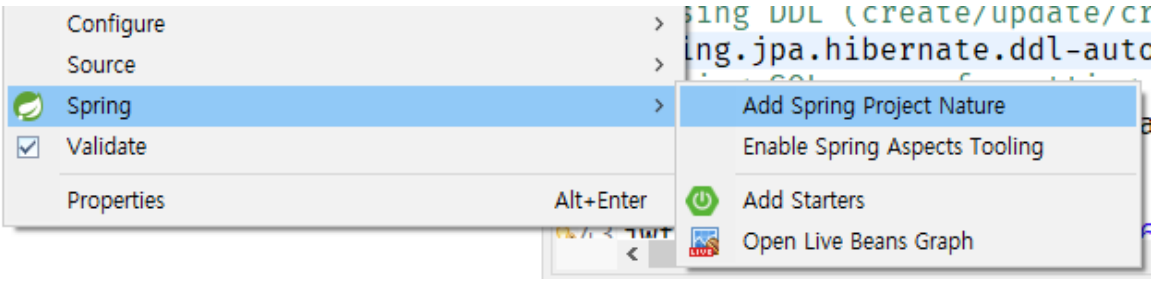
a. 이슈관리 : Jira
b. 형상관리 : GitLab
c. 커뮤니케이션 : mattermost, Notion, Figma, Kakaotalk
d. 개발환경
    i. OS : Windows 10
    ii. IDE
        1. Eclipse : 2020-06
        2. STS : 3.9.14
        3. Visual Studio Code : 1.70.0
    iii. Database : MySQL 5.7.39
    iv. Server : AWS EC2 (Ubuntu 20.04.4 LTS)
e. 상세 개발환경
    i. Backend
        1. Java 1.8.0_192(Zulu 8.33.0.1-win64)
        2. Spring Boot Gradle 6.7
        3. lombok : 1.18.24, Querydsl-jpa : 4.4.0, swagger2 : 3.0.0
    ii. FrontEnd
        1. HTML5, CSS3, JavaScript(ES6)
        2. React v18, React-Router-Dom v6
        3. Node.js 14.17.0

2. 빌드 상세내용

a. Backend
이클립스 내에서 프로젝트 실행



Eclipse에서 /backend 폴더를 Existing Gradle Project로 import 합니다. import 이후 refresh, gradle창에서 gradle clean 을 실행합니다.



Project Explorer에서 프로젝트를 우클릭하고 [Spring] 메뉴의 [Add Spring Nature]를 선택합니다. Spring Nature 설정이 완료되면, 프로젝트에는 [Spring Elements]가 생성됨을 확인할 수 있습니다. (기존에 이미 Spring Project Nature가 적용된 있는 경우 안 보일 수 있습니다.)

Lombok 설치

Getter, Setter 등 자동완성 어노테이션을 위해 프로젝트에 Lombok을 설치합니다. 최신 버전 다운로드 후 해당 이클립스 버전을 찾아서 설치해줍니다. 설치 이후 프로젝트의 Project and External Dependencies에서 lombok.jar를 확인합니다.

문자 인코딩 설정

-Dfile.encoding=UTF-8

이클립스가 설치된 폴더로 이동해서, eclipse.ini 설정 파일을 열고 해당 옵션을 추가합니다.

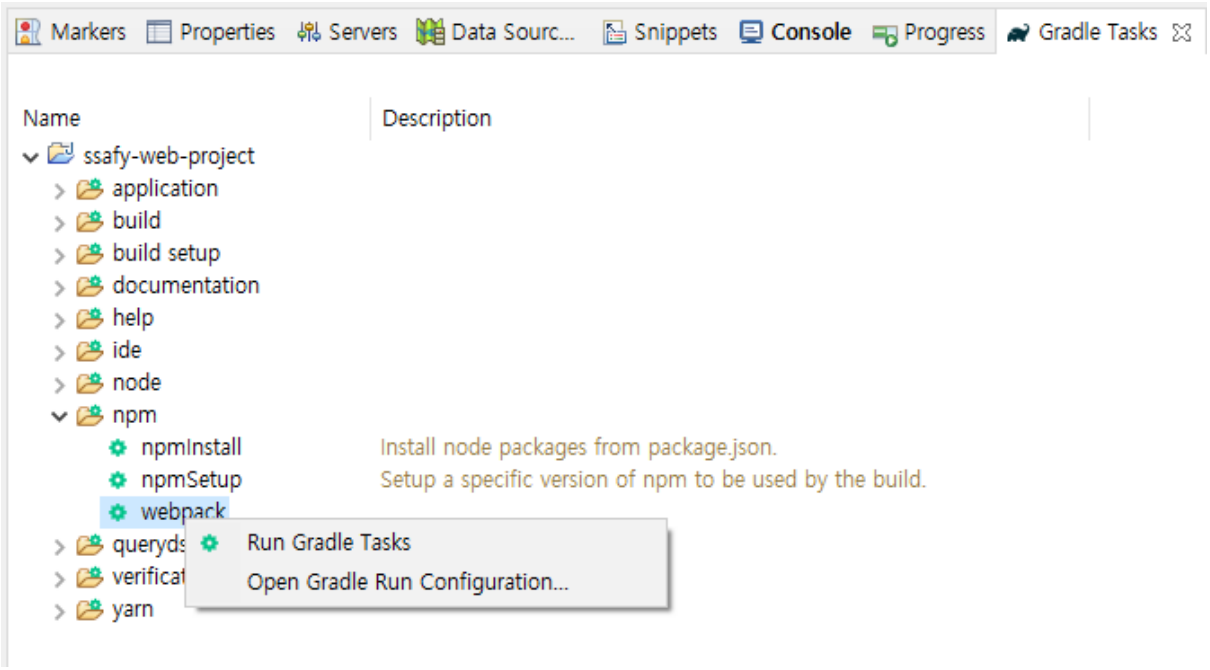
application.properties 파일 확인

```
server.address=localhost
…
spring.datasource.url=jdbc:mysql://localhost:3306/ssafy_web_db?useUnicode=true&characterEncoding=utf8&serverTimezone=Asia/Seoul&zeroDateTimeBehavior=convertToNull&rewriteBatchedStatements=true&useLegacyDatetimeCode=false
spring.datasource.hikari.username=root
spring.datasource.hikari.password=8y^T-Js.F3jb8N8
```

application.properties 파일에서 WAS 설정, DB 설정, jwt 등을 적용할 수 있습니다. 우선 로컬에서 테스트 시, 주소를 localhost로 두고 로컬에

설치된 MySQL DB 설정을 확인하고 맞게 변경해줍니다. mySQL의 버전은 5.7.39여야 하고 ssafy_web_db라는 스키마가 미리 설치되어 있어야 합니다.

```
create database IF NOT EXISTS `ssafy_web_db` collate utf8mb4_general_ci;
```



프로젝트 내의 build.gradle 을 우클릭하여 refresh build gradle을 실행시킵니다. Gradle Tasks 탭에서 ssafy-web-project -> npm -> webpack -> run gradle task를 통해 빌드합니다. 빌드가 완료된 걸 확인한 뒤, 프로젝트에서 Run as -> Spring Boot App을 실행합니다. 정상적으로 실행되었다면, Swagger 링크인 http://localhost:8080/swagger-ui/#/ 로 접속이 가능합니다.

Command 버전 배포 방법

배포할 때는, Windows command 창에서 다음과 같이 빌드를 진행합니다.

```
cd C:\Users\SSAFY\<....>\backend
gradlew clean build
```

빌드에 성공하면, /backend/buid/libs 경로에 ssafy-web-project-1.0-SNAPSHOT 파일이 생성되는 걸 확인할 수 있습니다. 해당 파일을 아래의 커맨드로 이클립스 내에서와 동일하게 실행할 수 있습니다.

```
java -jar ssafy-web-project-1.0-SNAPSHOT.jar
```

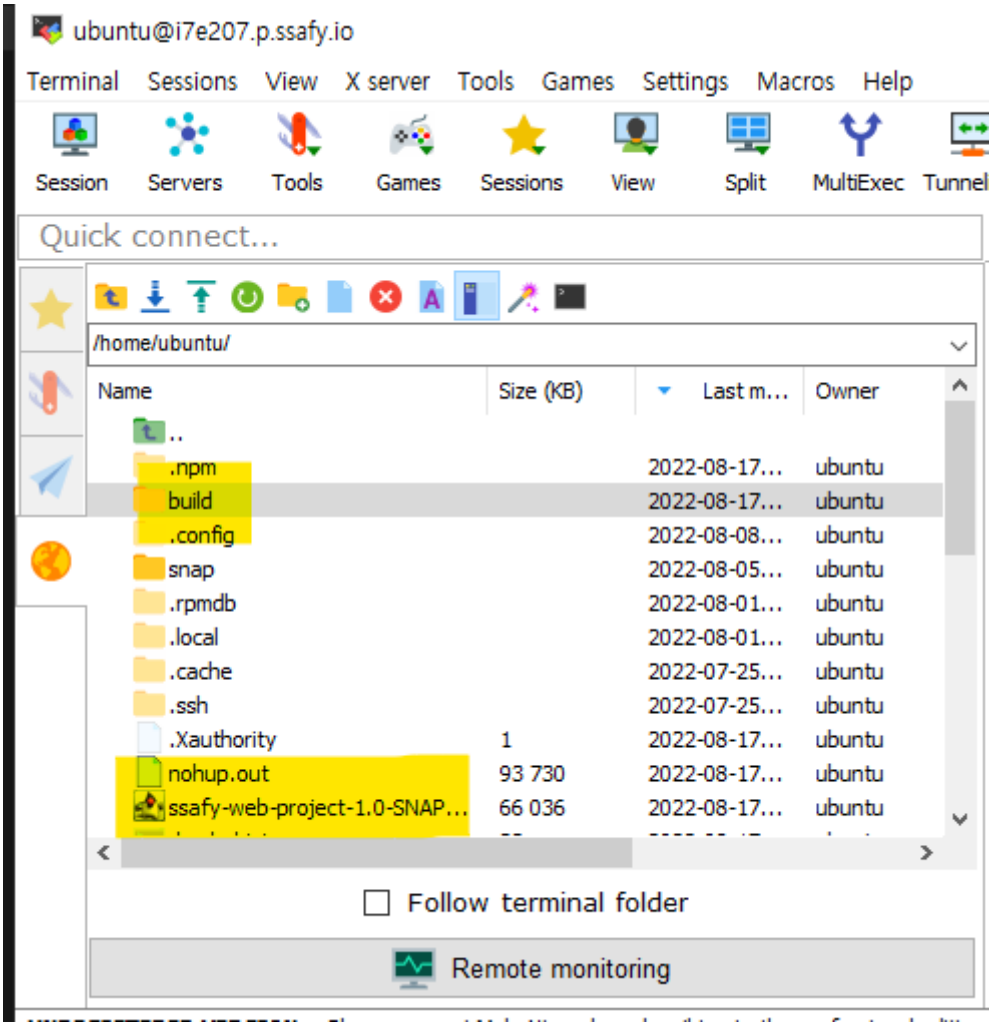
b. Frontend

```
npm install
npm run build
```

Node.js 14.17.0 버전을 설치한 뒤, /frontend-developing 폴더에서 npm install로 node_modules를 설치해줍니다. npm run build 명령어로 build파일을 생성합니다. npm start 명령어로 localhost환경에서 테스트해 볼 수 있습니다.

3. 배포 특이사항

a. 빌드 파일 업로드



Command 버전으로 빌드한 /backend/build/lib/ 파일과 /fronted-developing/build 폴더를 AWS EC2에 업로드합니다. 업로드 시에는 MobaXterm을 사용했습니다.

b. 배포 명령어 정리
다음과 같은 명령어로 배포할 수 있습니다.

sudo service nginx restart

Nginx 웹서버가 사용자에게 보내는 /build 파일을 갱신합니다.

ps -ef | grep ssafy-web-project-1.0-SNAPSHOT.jar

현재 API 웹서버를 사용하고 있는 프로세스를 찾습니다.

kill - 9 <PID>

해당 프로세스를 UID를 통해삭제합니다.

nohup java -jar ssafy-web-project-1.0-SNAPSHOT.jar &

API서버에 해당하는 .jar 파일을 백그라운드로 실행시킵니다.

sudo su **opt/openvidu/openvidu start**

Openvidu를 실행시킵니다. 성공 시, https://i7e207.p.ssafy.io:8443/#/ 으로 데모 프로그램 실행이 가능합니다. Ctrl+c로 백그라운드에서 실행하도록 전환할 수 있습니다.

c. EC2 세팅 과정
EC2에 java, MySQL, letsencrypt, Docker, Docker-promise, Openvidu On premises, NginX를 설치합니다.

sudo apt-get update **sudo apt-get -y install dpkg**

먼저, apt 데이터베이스를 업데이트합니다. 그다음, .deb 패키지를 설치 할 수있도록 dpkg를 설치합니다.

1. java 설치 AWS EC2에 java Zulu 8.33.0.1-linux64 (build 1.8.0_192-b01)

.deb파일을 다운로드 후 EC2에 업로드하고, 아래 명령어로 java를 설치합니다.

```
sudo dpkg -i zulu8.33.0.1-jdk8.0.192-linux_amd64.deb
```

설치 후, java –version으로 버전을 확인합니다.

2. MySQL 설치(전체 설치 가이드 링크)

```
sudo apt-get update
sudo apt install wget -y
```

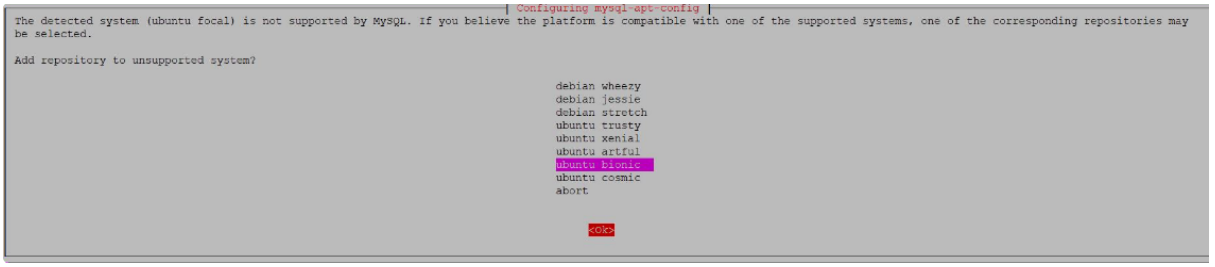
먼저 apt 데이터베이스를 업데이트한 뒤, wget을 설치합니다.

```
wget
https://dev.mysql.com/get/mysql-apt-config_0.8.12-1_all.deb
```

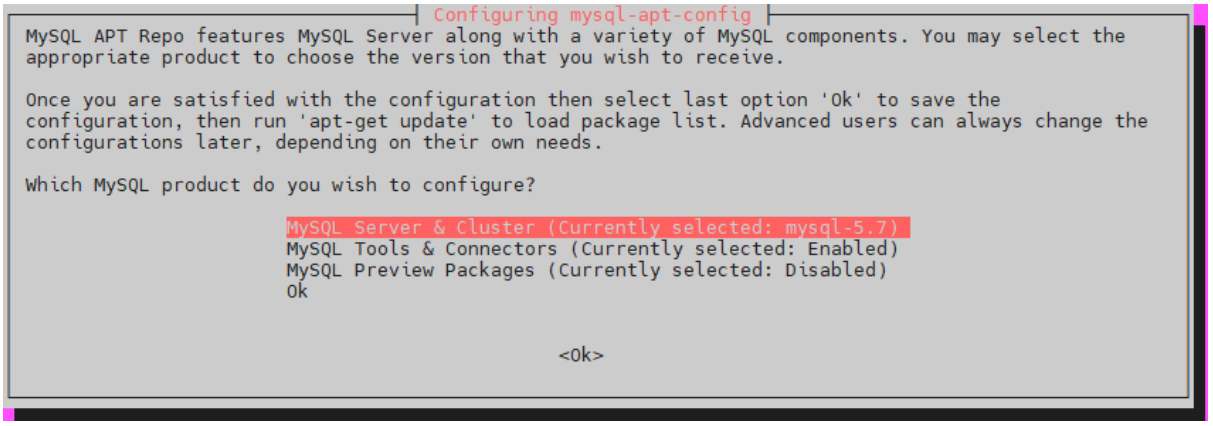
/home/ubuntu 경로에서, MySQL APT 리포지터리를 wget 명령어로 다운로드합니다.

```
sudo dpkg -i mysql-apt-config_0.8.12-1_all.deb
```

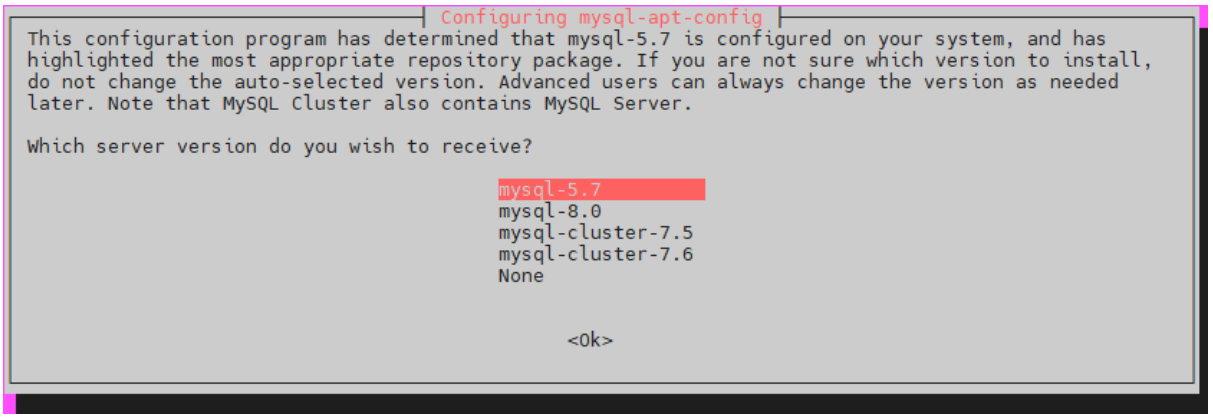
deb 패키지를 dpkg를 이용해서 설치합니다.



mysql-apt-config 창에 진입하게 됩니다. Ubuntu Bionic 옵션을 선택하고 Ok를 누릅니다.



설치하고 싶은 제품으로 MySQL Server & Cluster를 선택합니다.



MySQL 5.7을 선택하고 OK를 누릅니다. 다시 돌아온 창에서 OK를 입력하면 APT 설정이 완료됩니다.

```
sudo apt-get update
sudo apt-cache policy mysql-server
sudo apt install -f mysql-community-server=5.7.35-1ubuntu18.04
sudo apt install -f mysql-server=5.7.35-1ubuntu18.04
```

먼저, apt 데이터베이스를 업데이트한 뒤, 설정한 mysql-server 버전을 설치합니다.

```
mysql_secure_installation
```

해당 명령어를 통해 보안 설정창에 진입합니다. 이곳에서 root 계정의 비밀번호를 설정합니다. (프로젝트 기간 사용한 비밀번호는 8y^T-Js.F3jb8N8입니다.) mysql -u root -p 명령어 이후, 설정한 비밀번호를 입력 시 mysql 쉘로 접속하는 것을 확인합니다.

3. letsencrypt 설치 및 SSL 인증서 발급

```
sudo apt-get install letsencrypt
sudo letsencrypt certonly --standalone -d <i7e207.p.ssafy.io>
```

먼저, letsencrypt를 설치합니다. snap의 경우 Ubuntu 20.04 버전에 이미 설치되어 있습니다. 이후, letsencrypt로 인증서 발급을 시도합니다. 설치 후 congratulations 을 확인하면 발급이 완료된 것입니다. –standalone 방식으로 발급받고 이후 nginx를 설치 후, 설정파일로 nginx에 SSL 인증서를 설정할 것입니다. -d 옵션 뒤에는 도메인을 입력해야 합니다.

4. docker, docker-compose 설치
   먼저 포트를 열어줍니다.

```
sudo apt install -y docker-ce
sudo curl -L
https://github.com/docker/compose/releases/latest/download/do
cker-compose-$(uname -s)-$(uname -m) -o
/usr/local/bin/docker-compose
```

openvidu on premise를 서버에서 구동하기 위해서 docker와 docker-compose를 설치합니다. docker compose의 경우 공식문서를 참고하여 curl 명령어를 이용해서 다운로드합니다. 권한 부여 이후 버전 체크를 통해 정상 설치를 확인합니다.

5. OpenVidu On premises 2.20 버전 설치

```
ufw allow ssh
ufw allow 80/tcp
ufw allow 443/tcp
ufw allow 3306/tcp
ufw allow 3478/tcp
ufw allow 3478/udp
ufw allow 8080
ufw allow 8442/tcp
ufw allow 8442/udp
ufw allow 8443/tcp
ufw allow 8443/udp
ufw allow 40000:57000/tcp
ufw allow 40000:57000/udp
ufw allow 57001:65535/tcp
ufw allow 57001:65535/udp
```

```
ufw enable
```

Openvidu는 WebRTC를 위해 여러 가지 포트를 사용합니다. 공식문서를 참고하여, 해당 포트들을 열어줍니다. API 서버와 DB에 접근하기 위해 3306, 8080 포트도 추가로 열어줘야 합니다.

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

Openvidu를 설치하고 실행하기 앞서서, 루트 권한을 획득합니다. 공식문서에서 권고하는 대로 /opt 경로에서 openvidu 배포 버전을 다운받습니다.

```
nano /opt/openvidu/.env
```

설치 이후, /opt/openvidu/.env 파일을 열고 letsencrpyt 인증서 관련 환경변수를 설정해줍니다.

```
DOMAIN_OR_PUBLIC_IP=<i7e207.p.ssafy.io>
OPENVIDU_SECRET=MY_SECRET
CERTIFICATE_TYPE=letsencrypt
LETSENCRYPT_EMAIL=<your@email.com>
HTTP_PORT=8442
HTTPS_PORT=8443
```

공식 문서를 참고하여, 프로젝트에 맞는 환경변수들을 DOMAIN_OR_PUBLIC_IP, OPENVIDU_SECRET, CERTIFICATE_TYPE, LETSENCRYPT_EMAIL을 작성합니다. 사용하는 HTTP 포트, HTTPS 포트 번호도 8442와 8443으로 바꿔줍니다.

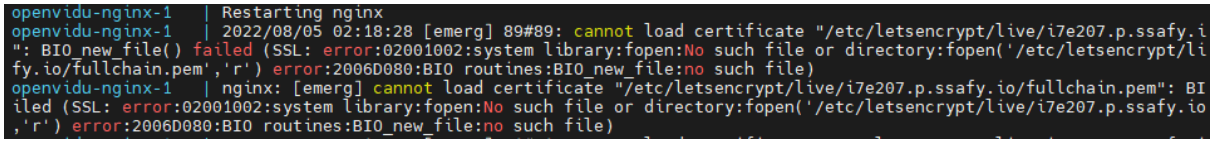
```
./openvidu start
```

.env파일을 저장한 다음, /opt/openvidu에서 openvidu를 시작합니다. 정상적으로 완료된다면, 아래와 같은 화면이 나오고 해당 링크에서 화상회의 데모를 실행해볼 수 있습니다.

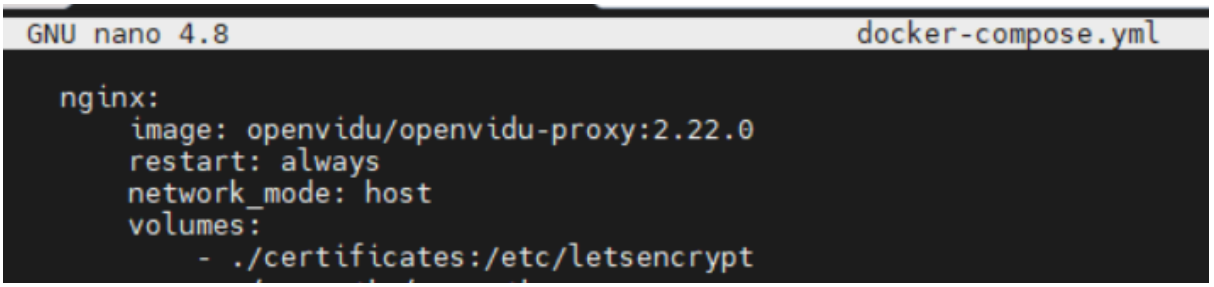
TrobuleShooting하기

정상적으로 실행되지 않는 경우, Docker-compose에서 사용하는 nginx가 이미 설치되어있는 인증서를 찾지 못하는 경우를 생각해볼 수 있습니다. openvidu가 구동되고 있을 때, ctrl+c로 콘솔창을 열고 Nginx 로그를 출력해봅니다.

sudo docker-compose logs nginx



위와 같이 로그에서, .pem 키를 찾지 못할 시, 해당 문서를 참고해서 openvidu가 사용하는 docker-compose.yml 파일을 수정해서 EC2의 pem키 경로를 사용할 수 있도록 volume 부분을 수정해줍니다.



nginx의 volumes: 옵션에서 현재 컨테이너가 참조하는 ./certificates파일이 EC2의 /etc/letsencrypt를 공유할 수 있도록 설정해준 뒤, 다시 오픈비두를 실행 해줍니다.

d. nginx 설치 및 환경파일 수정

openvidu on premise에서 내부적으로 Nginx를 사용하는 것과 별개로, 정적파일인 build 파일을 클라이언트에게 보내주는 웹 서버 역할을 위해 Nginx를 추가로 설치합니다. 그 다음, Nginx에 인증서를 등록해서 https 연결이 가능하도록 /etc/nginx sites-available/default 파일을 수정해줍니다.

```
server {
        listen 80 default_server;
        listen [::]:80 default_server;

        root /home/ubuntu/build;
        index index.html index.htm index.nginx-debian.html;
        server_name sangeo.site;
```

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


server {

        root /home/ubuntu/build;
        index index.html index.htm index.nginx-debian.html;
        server_name i7e207.p.ssafy.io; # managed by Certbot

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

    listen [::]:443 ssl ipv6only=on; # managed by Certbot
    listen 443 ssl; # managed by Certbot
    ssl_certificate
     /etc/letsencrypt/live/i7e207.p.ssafy.io/fullchain.pem; #
     managed by Certbot
    ssl_certificate_key
     /etc/letsencrypt/live/i7e207.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 ($host = i7e207.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

    listen 80 ;
    listen [::]:80 ;
    server_name i7e207.p.ssafy.io;
    return 404; # managed by Certbot
}
```

기존에 작성된 파일 위에, # managed by Certbot 인 코드 부분이 SSL 인증서를 위해 추가되어야 할 부분입니다. 별개로 서비스 도메인으로 접근 시 build 폴더의 index.html을 보내주도록 하여야 합니다. (API 서버는 스프링 TomCat WAS로 설정합니다.)

e. pkcs키 등록 및 application.properties에 두기

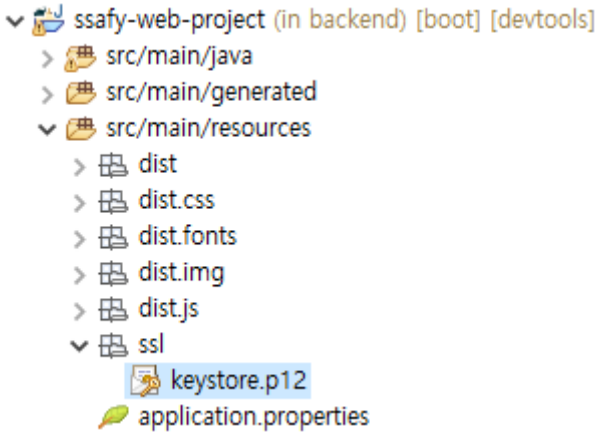
API 서버에도 https 연결을 해주기 위해서, certbot을 이용해 발급받은 .pem 키를 openssl을 이용해 .p12파일로 변경합니다.

```
sudo snap install –classic certbot
```

먼저, certbot을 설치합니다.

```
sudo su
cd /etc/letsencrypt/live/i7e207.p.ssafy.io/
openssl pkcs12 -export -inkey privkey.pem -in cert.pem -out keystore.p12
```

루트 권한을 얻고, .pem키가 있는 곳에서 keystore.p12로 변환합니다.



해당 파일을 로컬로 다운로드받고, /backend/main/java/resource/ssl 경로에 저장합니다.

application.properties에서, WAS가 API 서버로의 요청을 허용할 수 있도록 설정해줍니다.

```
server.address=172.26.15.237
server.ssl.key-store=classpath:ssl/keystore.p12
server.ssl.key-store-type=PKCS12
server.ssl.key-store-password=8y^T-Js.F3jb8N8
```

해당 서버를 로컬호스트로 변경하려면, ssl 키 설정을 주석처리하고, server.address=localhost로 변경합니다.

4. DB 접속 정보

a. EC2 MySQL DB Connection 연결

로컬에서 EC2에 설치된 MySQL에 연결하기 위해서, Workbench에서 새로운 connection을 생성합니다.

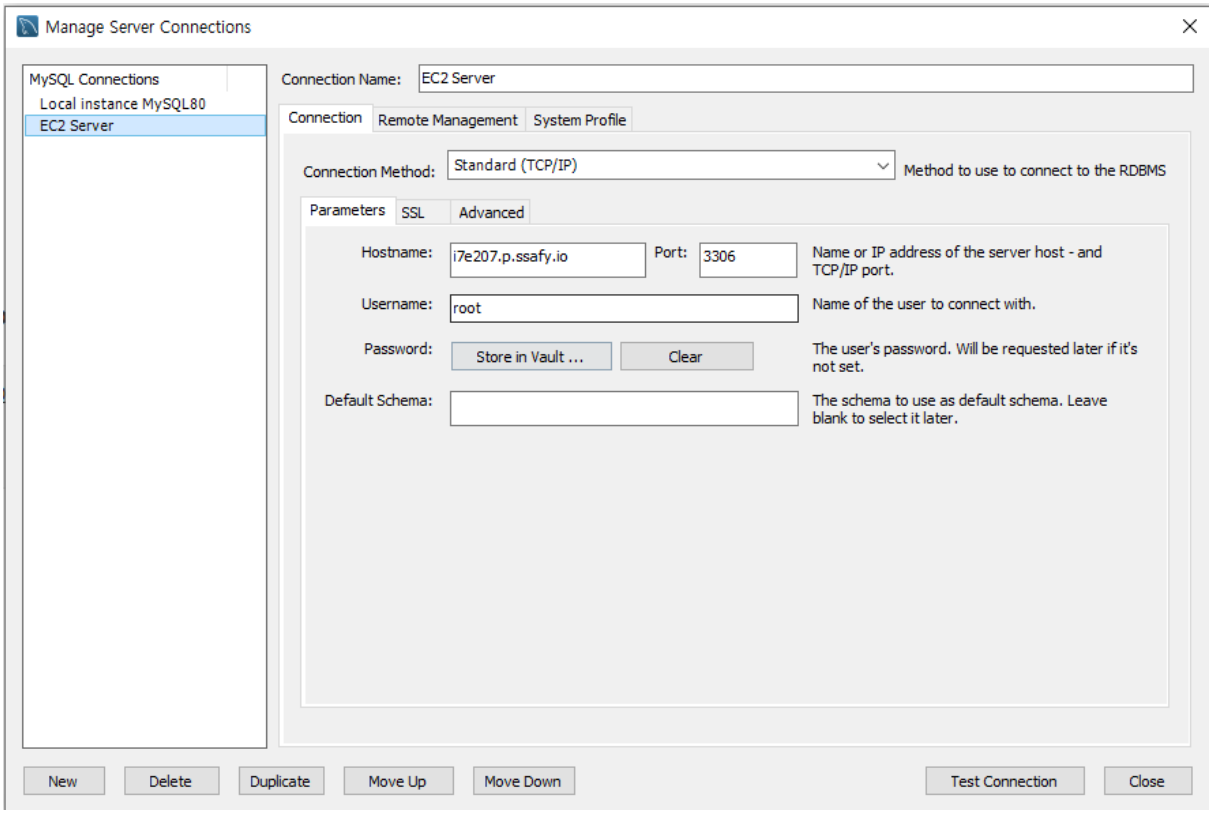
b. 스키마 생성

```
create database IF NOT EXISTS `ssafy_web_db` collate
utf8mb4_general_ci;
```

프로젝트에서 사용하는 스키마를 생성합니다. 인코딩을 UTF-8으로 설정해줍니다.

c. Dump 파일 실행

프로젝트의 덤프 파일을 실행해줍니다.



(/backend/src/main/resources/application.properties에는 hibernate-ddl-auto=none으로 설정되어 있습니다.)

5. 외부 서비스 정보

a. Cloudinary

i. 사이트 내에서 프로필, 자격증, 그림 분석과 같은 이미지 업로드 시 Cloudinary 라이브러리에 이미지가 저장됩니다.

ii. 저장 후, 반환된 경로는 DB에 저장합니다.

iii. 사이트에서 이미지 로드 시, DB에 있는 경로를 참조해 이미지를 가져옵니다.

iv. https://cloudinary.com/ 에서 가입 가능하며,

v.

```
class ImageUploader {
    async upload(blob){
        const data = new FormData();
        data.append('file', blob);
        data.append('upload_preset', 'j0olf3vr');

        const res = await fetch (
            'https://api.cloudinary.com/v1_1/[회원아이디]/upload',
            {
                method : 'POST',
                body: data,
            }
        );
        return await res.json();

    }
}

export default ImageUploader;
```

vi. 위와 같이 서비스를 등록하여 업로드 기능을 사용할 수 있습니다.

b. Naver 로그인
   i. 상어에서 Naver 로그인시에는 자동으로 User 권한으로 로그인됩니다. Naver 계정이 있는 모두가 사용가능합니다.
   ii. 처음 Naver 로그인시 자동으로 DB의 User 테이블에 데이터가 저장되어 회원가입 처리됩니다. 이때 따로 전화번호를 입력 받습니다. 사용자의 이름은 Naver에서 제공받습니다.
   iii. Naver 로그인을 통해 따로 아이디와 비밀번호를 등록하지 않고 로그인 가능합니다.
   iv. 개발을 위한 Naver 아이디 로그인 신청은 https://developers.naver.com/main/ 에서 가능하며 애플리케이션 등록을 통해 API 이용신청이 가능합니다.
c. 아임포트
   i. 상어에서는 회원가입시에 아임포트 API를 사용하여 KG이니시스 본인인증을 받고있습니다. 사용자는 별도의 서비스 가입없이 본인인증을 할 수 있습니다.
   ii. 본인인증은 카카오톡 인증, 네이버 인증서, PASS 인증서, 금융인증서, 토스인증서, PAYCO 인증서로 가능합니다.
   iii. 개발을 위한 아임포트 서비스 가입은 https://www.iamport.kr/ 에서 가능하며 회원가입, 로그인 후 결제 연동 -> 테스트 연동 추가로 가능합니다.