Gitlab 소스 클론 이후 빌드 및 배포 매뉴얼

작성 - 9기 C205 손효민

목차

Gitlab 소스 클론 이후 빌드 및 배포 매뉴얼

1. 사용한 JVM, 웹 서버, WAS 제품 등의 종류와 설정값, 버전 (IDE버전 포함) 기재

1-1. 종류

- 웹서버 : Nginx

1-2. 기술 스택

구분	기술스택	상세내용	버전
공통	형상관리	Gitlab	-
	이슈관리	Jira	-
	커뮤니케이션	Mattermost	5.3.1
		Notion	2.0.49
		discord	1.0.55
		kakaotalk	-
	기타 편의 툴	Postman	10.16
		PuTTY	0.78
		Termius	8.0.2
		MobaXterm	23.2
	UI/UX	Figma	-
	OS	Windows	10
Back-End	DB	MySQL	8.0.33
		Redis	7.0.12
		MariaDB	11.2.0 Alpha
		JPA	-
	Java	OpenJDK	11.0.15
		Spring Boot	2.4.5
		Spring 내장 tomcat	2.4.5
	Build	Gradle	8.2.1
	IDE	IntelliJ IDEA	2023.1.4
Front-End	React	React	18.2.0
		Redux	8.1.1
		Redux - toolkit	1.9.5
		React-router-dom	6.14.1
		React-Query	3.39.3
	WebRTC	Openvidu	2.28.0-
	Node.js		18.16.1
	Axios		1.4.0

	face-api.js		0.22.2
	IDE	Visual Studio Code	1.80.1
Server	서버	AWS EC2	-
	플랫폼	Ubuntu	20.04.6 LTS
	배포	Docker	24.0.5
		Docker Compose	2.20.2
		Docker Desktop	4.21.1

2. 배포 매뉴얼

1) 서버에 OpenVidu를 반드시 먼저 배포합니다.

   A. 위의 안내에 따라 OpenVidu를 서버에 먼저 설치합니다.

   B. .env 파일에서 OpenVidu 환경 설정을 합니다

   C. 설정 후 OpenVidu 서버를 실행합니다. (Ctrl + C를 누르면 백그라운드로 실행됨)

2) Git clone 및 배포 과정

   A. 로컬 컴퓨터에서 Git clone을 받습니다.

```
https://lab.ssafy.com/s09-webmobile1-sub2/S09P12C205.git
```

   B. AWS EC2 서버에 Redis DB를 배포합니다.

      i. 도커를 이용하여 redis를 다음과 같이 실행합니다. 백엔드 배포 전에 반드시 선행되어야 합니다.

      ii. **3-5. Redis** 의 단계의 명령어를 실행합니다.

   C. 로컬 컴퓨터에서 프론트엔드 프로젝트를 빌드하고 AWS EC2 서버에 배포합니다.

      iii. Git clone을 통해 받은 프로젝트의 frontend 디렉토리의 루트 경로에서 **3-2. Frontend** 의 단계의 명령어를 실행합니다.

      iv. Nginx와 react가 함께 배포됩니다.

   D. 로컬 컴퓨터에서 백엔드 프로젝트를 빌드하고 AWS EC2 서버에 배포합니다.

      i. 프로젝트 폴더 내에 있는 backend 디렉토리의 루트 경로에서 **3-3. Backend** 의 단계의 명령어를 실행합니다.

      ii. Backend 경로에 Dockerfile이 있습니다. 이를 이용하여 Docker Container를 통해 프론트엔드를 배포할 준비를 합니다.

E. Backend gradle 의존성

```
plugins {
	id 'java'
	id 'org.springframework.boot' version '2.7.14'
	id 'io.spring.dependency-management' version '1.0.15.RELEASE'
}

group = 'com.ssafy'
version = '0.0.1-SNAPSHOT'

java {
	sourceCompatibility = '11'
}

configurations {
	compileOnly {
		extendsFrom annotationProcessor
	}
}

repositories {
	mavenCentral()
}

dependencies {
	implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
	implementation 'org.springframework.boot:spring-boot-starter-data-
redis'
	implementation 'org.springframework.boot:spring-boot-starter-web'
    implementation 'io.openvidu:openvidu-java-client:2.28.0'
	implementation 'org.springframework.boot:spring-boot-starter-
validation'
	implementation 'io.jsonwebtoken:jjwt-api:0.11.2'
	implementation 'io.jsonwebtoken:jjwt-impl:0.11.2'
	implementation 'io.jsonwebtoken:jjwt-jackson:0.11.2'
	compileOnly 'org.projectlombok:lombok'
	developmentOnly 'org.springframework.boot:spring-boot-devtools'
	runtimeOnly 'com.mysql:mysql-connector-j'
	runtimeOnly 'org.mariadb.jdbc:mariadb-java-client:2.7.4'
	annotationProcessor 'org.projectlombok:lombok'
	testImplementation 'org.springframework.boot:spring-boot-starter-test'
	testImplementation 'io.projectreactor:reactor-test'
	implementation 'org.springframework.boot:spring-boot-starter-mail'
}

tasks.named('test') {
	useJUnitPlatform()
}
```

F. AWS EC2의 Nginx 설정과 ssl 인증서 발급 및 적용

- Openvidu 같은 경우, 카메라를 사용하기 위해서는 반드시 https로 이용해야 하기에 SSL 인증서를 발급받아야 합니다. 인증서 발급을 위해서는 도메인이 필요

합니다.

- "2. Nginx 설정과 SSL 인증서 발급 및 적용" 단계의 명령어를 실행합니다.
- Nginx 를 다운받고, letsencrypt를 설치하고, 인증서 발급 후 /etc/nginx/sites-available에서 default파일에 nginx 설정을 해준 후, nginx 재시작 명령어를 입력해줍니다.
- 이렇게 실행을 마친다면, http로 80포트 접근시, 443 포트(https)로 리다이렉트 됩니다. 그리고 백엔드 url을 /api/**로 분기처리할 수 있습니다. https://도메인주소 로 접근하면 배포한 웹 페이지에 접속할 수 있게 됩니다.

3.빌드 시 사용되는 환경 변수 등의 주요 내용 상세 기재

3-1. 개요

비움 서비스의 배포 환경 및 흐름으로는 팀원들이 개발 완료한 프로젝트를 Gitlab에 push하면, 서버 담당자가 Front-end, Back-end를 빌드합니다.

각 프로젝트를 빌드한 후에 Docker 이미지를 만들고, 이를 Docker Hub에 push한 후, 이로부터 서비스에 필요한 이미지를 받아와 컨테이너로 띄웁니다.

서버의 경우에 SSAFY에서 지원받은 AWS EC2 인스턴스로 인프라를 구축하였습니다. 또한 Nginx는 리버스 프록시 서버로 구성했습니다. 그래서 8080포트는 Back-end 서버로, 3000 포트는 프론트 서버로 설정하여 Load Balancing이 가능하도록 구축하였습니다.

서버에 Openvidu를 제일 먼저 설치하여 포트 충돌이 없도록 했고 8081, 8443 포트 번호로 변경하여 Back-end의 openvidu application server와 연동이 잘 되도록 구축했습니다.

3-2. Frontend

- Frontend 프로젝트 내에 Dockerfile 생성 (src나 build 있는 곳)

```
# nginx 이미지를 사용합니다. 뒤에 tag 가 없으면 latest 를 사용합니다.
#FROM nginx
FROM nginx

# work dir 고정
WORKDIR /app

# work dir 에 build 폴더 생성 /app/build
RUN mkdir ./build

# host pc 의 현재경로의 build 폴더를 workdir 의 build 폴더로 복사
ADD ./build ./build

# nginx 의 default.conf 를 삭제
RUN rm /etc/nginx/conf.d/default.conf

# host pc 의 nginx.conf 를 아래 경로에 복사
COPY ./nginx.conf /etc/nginx/conf.d

# 3000 포트 오픈
EXPOSE 3000

# container 실행 시 자동으로 실행할 command. nginx 시작함
CMD ["nginx", "-g", "daemon off;"]
```

- Frontend 프로젝트 내에 nginx.conf 파일 생성 (src 폴더나 build 있는 곳)

```
server {
    listen 3000;
    location / {
        root    /app/build;
        index   index.html;
        try_files $uri $uri/ /index.html;
    }
}
```

- 배포 과정

```
# Visual Studio Code 의 Terminal 에서
npm run build

docker build -t [도커허브계정아이디]/[레포지토리명]:[태그명] .

docker push [도커허브계정아이디]/[레포지토리명]:[태그명]


# Ubuntu 서버에서

sudo docker pull [도커허브계정아이디]/[레포지토리명]:[태그명]

sudo docker images

sudo docker run --rm -d -p 3000:3000 --name [원하는 이름] [프론트이미지 ID]
```

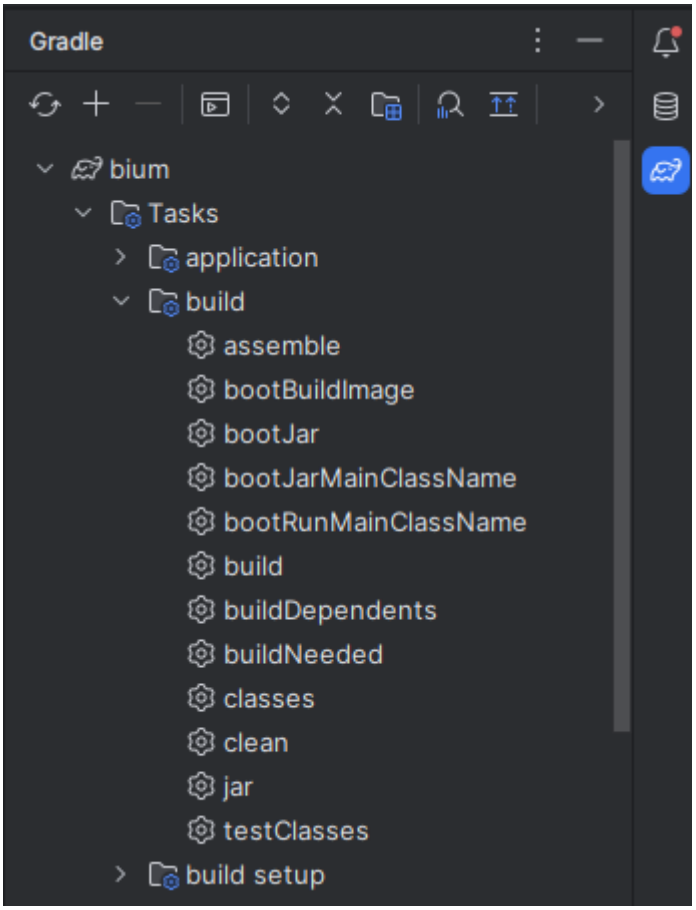
3-3. BackEnd

- BackEnd 프로젝트 내에 Dockerfile 생성 (src나 build, greadlew 있는 곳)
  - ■ 프로젝트 배포 시 Docker Hub 계정 아이디는 sondoh100, 레포지토리명은 bium 을 사용했습니다. 태그명은 백엔드 프로젝트는 back, 프론트 프로젝트는 front를 사용했습니다.

```
# open jdk java11 버전의 환경
FROM openjdk:11-jdk

# JAR_FILE 변수 정의 -> 기본적으로 jar file 이 2 개이기 때문에 이름을 특정해야 함
ARG JAR_FILE=./build/libs/bium-0.0.1-SNAPSHOT.jar

# JAR 파일 메인 디렉토리에 복사
COPY ${JAR_FILE} app.jar

EXPOSE 8080

# 시스템 진입점 정의
ENTRYPOINT ["java","-jar","/app.jar"]
```

- 배포 과정

  IntelliJ에서 Gradle -> bium -> Tasks -> build -> clean과 build 진행




```
docker build -t [도커허브계정아이디]/[레포지토리명]:[태그명] .

docker push [도커허브계정아이디]/[레포지토리명]:[태그명]

sudo docker pull [도커허브계정아이디]/[레포지토리명]:[태그명]

sudo docker images

sudo docker run --rm -d -p 8080:8080 --name [원하는 이름] [백이미지 ID]
```

3-4. OpenVidu

WebRTC는 OpenVidu의 on premises 공식문서 https://docs.openvidu.io/en/stable/deployment/ce/on-premises/ 를 참고하여 설치했습니다.

- Docker 및 Docker Compose 설치
  - Docker를 설치합니다.

```
# 도커 설치를 위해 필요한 패키지들을 설치
$ sudo apt update

# 도커 공식 GPG 키를 추가
$ sudo apt install apt-transport-https ca-certificates curl software-
properties-common

# 도커 저장소의 URL 을 /etc/apt/sources.list.d/docker.list 파일에 추가
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor
-o /usr/share/keyrings/docker-archive-keyring.gpg

# 다시 패키지 목록을 업데이트하고 도커를 설치
$ echo "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-
keyring.gpg] https://download.docker.com/linux/ubuntu focal stable" | sudo tee
/etc/apt/sources.list.d/docker.list

# 도커를 시작하고, 부팅 시 자동으로 실행되도록 설정
$ sudo apt update
$ sudo apt install docker-ce
```

- Docker Compose를 설치합니다.

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

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

$ sudo docker-compose --version
```

- 포트 열기

- OpenVidu를 처음 설치할 때 22, 80, 443, 3478, 40000-57000, 57001-65535 포트를 열고 방화벽을 해제합니다. 또한 5442, 5443, 6379, 8888 포트를 열어둡니다.

```
ufw allow ssh
ufw allow 80/tcp
ufw allow 443/tcp
ufw allow 3478/tcp
ufw allow 3478/udp
ufw allow 40000:57000/tcp
ufw allow 40000:57000/u 에
ufw allow 57001:65535/tcp
ufw allow 57001:65535/udp
ufw enable
```

- OpenVidu 설치

```
$ sudo su

cd /opt

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

- OpenVidu의 .env 파일 내용

```
# 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=i9c205.p.ssafy.io

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

# 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=sondoh100@naver.com
```

1. Letsencrypt 인증서를 받았기 때문에 letsencrypt 의 certificate type으로 명시하고, 인증서 받을 때 작성했던 이메일 주소를 넣었습니다.

2. Nginx 설정과 SSL 인증서 발급 및 적용

   ■ Nginx 설치

```
# nginx 설치
$ sudo apt install nginx

# nginx 상태 확인
$ sudo systemctl status nginx

# nginx 실행 시작
$ sudo systemctl start nginx
```

   ■ SSL 설치

```
# let's Encrypt 설치
sudo apt-get install letsecrypt

# Certbot 설치
sudo apt-get install python3-certbot-nginx

# Certbot 동작
sudo certbot --nginx
```

- AWS EC2 서버의 nginx 설정 파일 – default

```
# nginx 환경 설정
sudo vi /etc/nginx/sites-available/default
```

- Nginx 파일에 SSL 인증서 설정 및 리버스 프록시 구축

```
server {
        location / {
                # First attempt to serve request as file, then
                # as directory, then fall back to displaying a 404.
                ###try_files $uri $uri/ =404;
                proxy_pass http://localhost:3000;
                proxy_buffer_size       128k;
                proxy_buffers   4 256k;
                proxy_busy_buffers_size 256k;
        }

        location /api/ {
                # First attempt to serve request as file, then
                # as directory, then fall back to displaying a 404.
                # try_files $uri $uri/ =404;
                proxy_pass http://localhost:8080;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_set_header Host $http_host;
                proxy_set_header X-Forwarded-Proto $scheme;
                proxy_set_header X-Forwarded-Proto https;
                add_header Content-Security-Policy "upgrade-insecure-requests";
        }

    listen [::]:443 ssl ipv6only=on; # managed by Certbot
    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/i9c205.p.ssafy.io-0001/fullchain.pem;
# managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/i9c205.p.ssafy.io-
0001/privkey.pem; # managed by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;
}

server {
    if ($host = i9c205.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

    listen 80 ;
    server_name i9c205.p.ssafy.io;
    return 404; # managed by Certbot
}
```

- Nginx 재시작

  Nginx 설정파일을 수정할 경우, nginx의 재시작이 필수입니다.

```
# 저희 프로젝트에서는 default로 했습니다.
$sudo ln -s /etc/nginx/sites-available/[파일명] /etc/nginx/sites-
enabled/[파일명]

# 다음 명령어에서 successful이 뜨면 nginx를 실행할 수 있다.
$sudo nginx -t

# nginx 실행 재시작
$ sudo systemctl restart nginx

# nginx 상태 확인
$ sudo systemctl status nginx
```

2. OpenVidu를 처음 설치 후 start 할때, 기본 포트인 HTTP는 80, HTTPS 는 443 포트를 유지한 상태에서 https://DOMAIN_OR_PUBLIC_IP:443/ 로 정상 접속이 되는지 확인해야 합니다.
3. 기본 포트인 상태에서 정상 접속을 확인한 후 포트 번호를 변경해줘야 합니다.

■ 포트 번호를 변경하는 이유는, 프론트, 백엔드 프로젝트에서 사용하는 nginx 포트와 동일하여 제대로 실행되지 않기 때문에 오픈 비두의 포트번호를 변경합니다.

- OpenVidu 실행 및 종료

```
https://DOMAIN_OR_PUBLIC_IP:HTTPS_PORT/

# cd /opt/openvidu에서 아래 명령어로 오픈비두 실행
./openvidu start

# 오픈비두 중지
./openvidu stop

# 오픈비두 재시작
./openvidu restart
```

3-5. Redis

- Docker 로 redis의 이미지를 가져온다

```
$ docker pull redis
```

- /home/ubuntu 에서 docker_config 폴더를 생성한다. 이 폴더 안에서 redis_config.conf 파일을 만들어 redis config를 설정한다.

```
# 어떤 네트워크 인터페이스로부터 연결할 수 있도록 할 것인지 관리 (여기에서는
Anywhere)
bind 0.0.0.0

# 사용 포트 관리
port 6380

# Master 노드의 기본 사용자(default user)의 비밀번호 설정
requirepass [사용하고자 하는 비밀번호]

# Redis 에서 사용할 수 있는 최대 메모리 용량. 지정하지 않으면 시스템 전체 용량
maxmemory 1g

# maxmemory 에 설정된 용량을 초과했을때 삭제할 데이터 선정 방식
# - volatile-ttl : expire time(TTL)이 가장 적게 남은 key 제거 (minor TTL)
maxmemory-policy volatile-ttl

# DB 데이터를 주기적으로 파일로 백업하기 위한 설정입니다.
# Redis 가 재시작되면 이 백업을 통해 DB 를 복구합니다.

#save 900 1      # 15분 안에 최소 1개 이상의 key 가 변경 되었을 때
#save 300 10     # 5분 안에 최소 10개 이상의 key 가 변경 되었을 때
#save 60 10000   # 60초 안에 최소 10000 개 이상의 key 가 변경 되었을 때
```

- Redis 를 Docker container로 실행합니다. Volume name은 redis_data로 지정해줬습니다. 또한 Redis의 비밀번호로 bium6890을 설정했습니다.

```
$ sudo docker run -d --restart=always --name=bium_redis -p 6380:6380 -e TZ=Asia/Seoul -v
/home/ubuntu/docker_config/redis_config.conf:/etc/redis/redis.conf -v redis_data:/data redis:latest
redis-server /etc/redis/redis.conf
```

3-6. Maria DB

- SSAFY 제공한 Maria DB를 사용했습니다. (프로젝트 설정 정보 -> MariaDB -> 생성)

4.배포 시 특이사항

- OpenVidu가 사용하는 포트가 많기 때문에 반드시 서버에 먼저 설치해야 합니다.
- 오픈비두 실행 시 80, 443 포트로 먼저 실행한 후에 포트번호를 변경해야 합니다. 프로젝트에서는 80을 8081로, 443을 8443으로 변경했습니다.
- 백엔드의 application.properties 에서 openvidu server 포트를 변경된 포트번호인 8443으로 변경해주어야 합니다.

5. DB 접속 정보 등 프로젝트에 활용되는 주요 계정 및 프로퍼티가 정의된 파일 목록

- 백엔드 프로젝트의 application.properties의 DB 관련 설정

```
#database
spring.jpa.hibernate.naming.implicit-
strategy=org.springframework.boot.orm.jpa.hibernate.SpringImplicitNamingStrateg
y
spring.jpa.hibernate.naming.physical-
strategy=org.springframework.boot.orm.jpa.hibernate.SpringPhysicalNamingStrateg
y
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL57Dialect
spring.data.web.pageable.one-indexed-parameters=true
spring.datasource.url=jdbc:mysql://stg-yswa-kr-practice-db-
master.mariadb.database.azure.com:3306/s09p13c205?serverTimezone=UTC&useUnicode
=true&characterEncoding=utf8
#spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.driver-class-name=org.mariadb.jdbc.Driver
spring.datasource.hikari.username=S09P13C205
spring.datasource.hikari.password=DwNXBVEgMc

spring.datasource.hikari.maximumPoolSize=2
```

- 백엔드 프로젝트의 application.properties의 Redis 관련 설정

```
#redis
#spring.data.redis.host=localhost
#spring.data.redis.port=6379
spring.data.redis.host=i9c205.p.ssafy.io
spring.data.redis.port=6380
spring.data.redis.pw=bium6890
spring.main.allow-bean-definition-overriding=true
```

- 백엔드 프로젝트의 application.properties의 SMTP 관련 설정

```
spring.mail.host=smtp.gmail.com
spring.mail.port=587
spring.mail.username=c205bium@gmail.com
spring.mail.password=uykuvyvgabeljtdj
spring.mail.properties.mail.smtp.starttls.enable=true
spring.mail.properties.mail.smtp.auth=true
```

- 백엔드 프로젝트의 application.properties의 server address 관련 설정

```
#it will be set build date by gradle. if this value is @build.date@, front-end
is development mode
build.date=@build.date@
server.port=8080
# ??
#server.address=localhost
# ??
server.address=0.0.0.0
server.servlet.contextPath=/api
# Charset of HTTP requests and responses. Added to the "Content-Type" header if
not set explicitly.
server.servlet.encoding.charset=UTF-8
# Enable http encoding support.
server.servlet.encoding.enabled=true
# Force the encoding to the configured charset on HTTP requests and responses.
server.servlet.encoding.force=true
#openvidu
server.ssl.enabled=false
```

- 백엔드 프로젝트의 application.properties의 OpenVidu 관련 설정

```
# ??
# OPENVIDU_URL: http://localhost:4443/
# OPENVIDU_SECRET: MY_SECRET
# ??
OPENVIDU_URL: https://i9c205.p.ssafy.io:8443/
OPENVIDU_SECRET: biumsecret
```