



포팅 매뉴얼

프로젝트 사용 도구

- 이슈 관리: JIRA
- 형상 관리: GitLab
- 커뮤니케이션: Notion, Mattermost, Discord, Gether
- 디자인: Figma
- UCC: VLO
- CI/CD: Jenkins

- FrontEnd

- React 버전 : 18.2.0
- node.js 18.13.0
- 상태 관리 라이브러리
 - react-redux v8.0.5
 - redux-tool-kit 1.9.1
- router v6.6
- JSX
- WebRTC
 - openvidu
- Facemesh 라이브러리
- SCSS

- BackEnd

- IntelliJ 2022.3.1
- Spring boot 2.7.7
- Java 1.8
- Gradle 7.6
- Spring Data JPA
- Swagger 2.0
- Spring Security
- Lombok

- DB

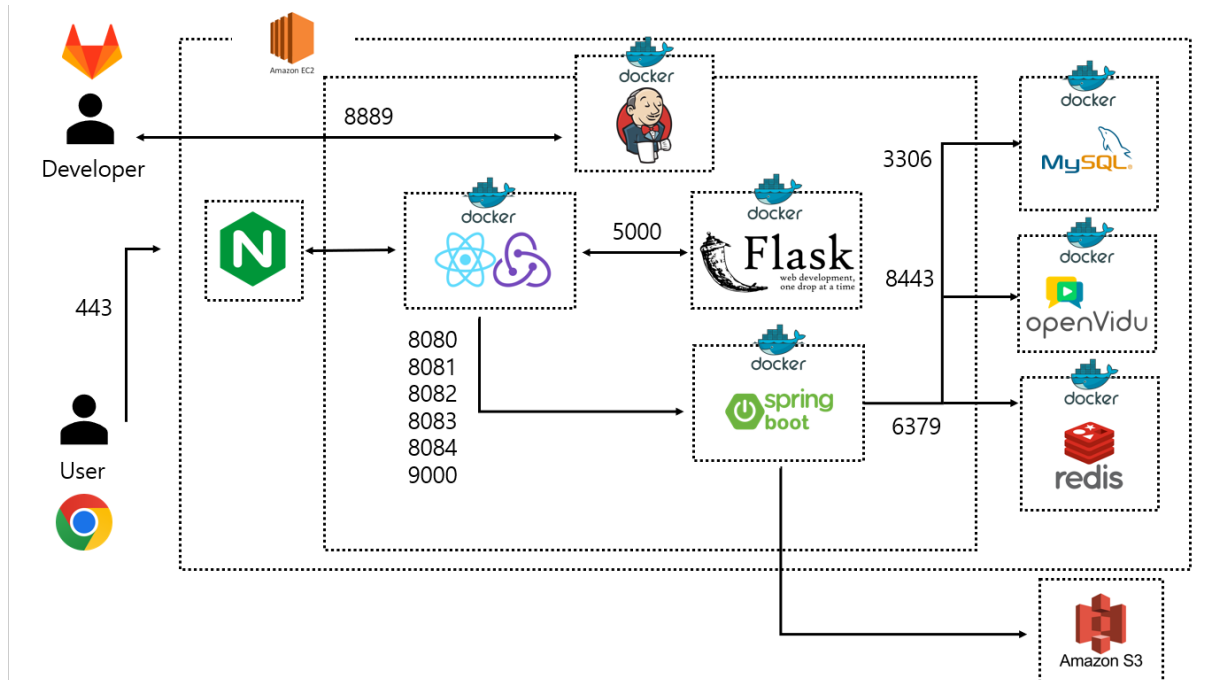
Mysql 8.0.31

Redis

- OS

- Ubuntu

- Docker 23.0.1
- Jenkins LTS
- 기타 편의 툴
 - Postman
 - Termius



설정 파일 및 환경 변수 정보

React

- .env

```
WDS_SOCKET_PORT=0

REACT_APP_BASE_URL=REST API(BACKEND) 요청 URL

REACT_APP_NAVER_CLIENT_ID=네이버 클라이언트 ID
REACT_APP_NAVER_CLIENT_SECRET=네이버 클라이언트 SECRET
REACT_APP_NAVER_REDIRECT_URI=네이버 리다이렉트 URI

REACT_APP_OPENVIDU_SERVER_SECRET=MY SECRET
```

Spring

- application.yml: back/beauduck-{service}/src/main/resources

```
server:
  port: {port num}
  servlet:
    context-path: /
  encoding:
    charset: UTF-8
    enabled: true
```

```

    force: true

ssl:
  enabled: true
  key-store: classpath:keystore.p12
  key-store-password: {password}
  key-store-type: PKCS12

spring:
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://i8b306.p.ssafy.io:3306/common_pjt?serverTimezone=Asia/Seoul
    username: root
    password: {root password}

mvc:
  pathmatch:
    matching-strategy: ant_path_matcher

jpa:
  hibernate:
    ddl-auto: none
    naming:
      physical-strategy: org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl
  properties:
    hibernate:
      format_sql: true
      show_sql: true

cloud:
  aws:
    credentials:
      accessKey: {AWS IAM AccessKey}      # AWS IAM AccessKey 적기
      secretKey: {AWS IAM SecretKey}      # AWS IAM SecretKey 적기
    s3:
      dir: {directory path}
      bucket: {bucket name}
    region:
      static: ap-northeast-2
    stack:
      auto: false

```

- dockerfile: back/beauduck-{service}, beauduck-front/
- dockerfile(Frontend)

```

# 가져올 이미지를 정의
FROM node:14
# 경로 설정하기
WORKDIR /app
# package.json 워킹 디렉토리에 복사 (.은 설정한 워킹 디렉토리를 뜻함)
COPY package.json .
# 명령어 실행 (의존성 설치)
RUN npm install
# 현재 디렉토리의 모든 파일을 도커 컨테이너의 워킹 디렉토리에 복사한다.
COPY . .

# 각각의 명령어들은 한줄 한줄씩 캐싱되어 실행된다.
# package.json의 내용은 자주 바뀌진 않을 거지만
# 소스 코드는 자주 바뀌는데
# npm install과 COPY . . 를 동시에 수행하면
# 소스 코드가 조금 달라질때도 항상 npm install을 수행해서 리소스가 낭비된다.

# 3000번 포트 노출
EXPOSE 3000

# npm start 스크립트 실행
CMD ["npm", "start"]

```

- dockerfile(Backend)

```

FROM adoptopenjdk/openjdk8
COPY build/libs/beauduck-auth-0.0.1-SNAPSHOT.jar app.jar
ENTRYPOINT ["java", "-jar", "app.jar"]

```

- Jenkinsfile: back/beauduck-{service}

```

pipeline {
  agent any
  environment {
    DOCKER = 'sudo docker'
  }

  stages {
    stage('Clone Repository') {
      steps {
        checkout scm
        echo 'Checkout Scm'
      }
    }

    stage('Build image') {
      steps {
        sh 'ls -al'
        dir('back/beauduck-auth'){
          sh 'ls -al'
          sh 'chmod +x ./gradlew'
          sh './gradlew build'
          sh 'docker build -t feat-back-auth .'
        }
        echo 'Build image...'
      }
    }

    stage('Remove Previous image') {
      steps {
        script {
          try {
            sh 'docker stop feat-back-auth'
            sh 'docker rm feat-back-auth'
          } catch (e) {
            echo 'fail to stop and remove container'
          }
        }
      }
    }

    stage('Run New image') {
      steps {
        sh 'docker run --name feat-back-auth -d -p 8080:8080 feat-back-auth'
        echo 'Run New image'
      }
    }
  }
}

```

Docker

- docker-compose.yml: /redis (아래 redis에서 확인)

Nginx

- custom.conf: /etc/nginx/sites-available (아래 nginx에서 확인)

빌드 및 배포

Java

- Java 설치

```
sudo apt-get update && sudo apt-get upgrade
sudo apt-get install openjdk-8-jdk
java -version
```

Docker, Docker-Compose 설치

- apt-transport-https : 패키지 관리자가 https를 통해 데이터 및 패키지에 접근할 수 있도록 한다.
- ca-certificates : certificate authority에서 발행하는 디지털 서명. SSL 인증서의 PEM 파일이 포함되어 있어 SSL기반 앱이 SSL 연결이 되어 있는지를 확인할 수 있다.
- curl : 특정 웹사이트에서 데이터를 다운로드 받을 때 사용
software-properties-common : PPA(Personal Package Archive)를 추가하거나 제거할 때 사용한다

```
sudo apt update && sudo apt-get upgrade
sudo apt install apt-transport-https ca-certificates
sudo apt install curl gnupg-agent software-properties-common
```

- Docker 공식 GPG 키 추가

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add
```

- stable repository를 세팅하기 위한 명령어 실행
- add-apt-repository: PPA 저장소를 추가해준다. apt 리스트에 패키지를 다운로드 받을 수 있는 경로가 추가됨

```
sudo add-apt-repository \
"deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable"
```

- 가장 최신 버전의 Docker 엔진을 설치한 후, 버전 확인

```
sudo apt update
sudo apt install docker-ce docker-ce-cli containerd.io
docker -v
```

- docker-compose 설치

```
curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-c
chmod +x /usr/local/bin/docker-compose
docker-compose --version
```

Mysql

- MySQL Docker Image 다운로드, 태그 버전 생략하면 최신 버전 다운로드

```
docker pull mysql:8.0.31
```

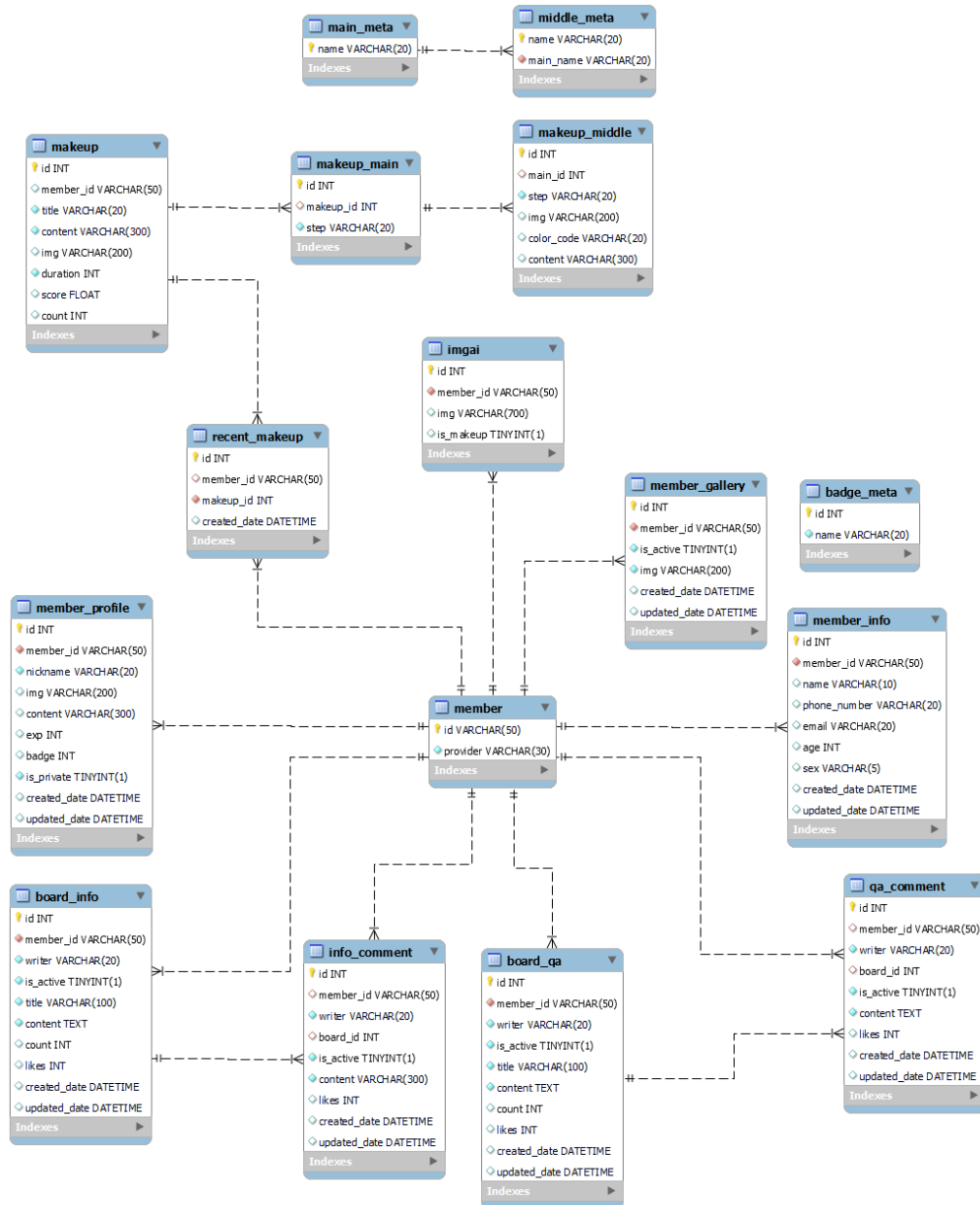
- MySQL Docker 컨테이너 생성 및 실행

```
docker run --name mysql-container -e MYSQL_ROOT_PASSWORD=1234 -v mysql-volume:/var/lib/mysql -d -p 3306:3306 mysql:8.0.31
```

- MySQL Docker 컨테이너 접속

```
docker exec -it mysql-container bash
```

ERD



Jenkins

- 플더 생성

```
sudo mkdir -p /home/ubuntu/jenkins
```

- Docker에 Jenkins 설치

```
sudo docker run --name jenkins -d -p 9999:9999 -p 50000:50000 -v /home/ubuntu/jenkins:/var/jenkins_home -v /var/run/docker.sock:/var/run/docker.sock
```

- jenkins docker container에 접속

```
sudo docker exec -it jenkins /bin/bash
```

- Jenkins 접속

http://<your-aws-domain>:<jenkins port> 접속 후 admin password 입력
ex) http://i8b306.p.ssafy.io:9090

- Jenkins Admin Password 확인

```
cat /var/jenkins_home/secrets/initialAdminPassword
```

Pipeline List

S	W	Name ↓	최근 성공	최근 실패	최근 소요 시간	
✓	☀	dev-front-pipeline	1 hr 9 min #72	4 days 19 hr #1	11 sec	▶
✓	☀	feat-back-auth-pipeline	1 day 17 hr #13	4 days 19 hr #1	23 sec	▶
✓	☀	feat-back-board-pipeline	1 day 17 hr #4	—	33 sec	▶
✓	☀	feat-back-consult-pipeline	10 hr #11	3 days 10 hr #6	21 sec	▶
✓	☁	feat-back-flask-pipeline	25 min #30	10 hr #29	5 min 21 sec	▶
✓	☀	feat-back-makeup-pipeline	1 day 17 hr #10	—	18 sec	▶
✓	☁	feat-back-openvidu✓	33 sec #10	4 min 26 sec #9	11 sec	▶
✓	☀	feat-back-together-pipeline	1 day 17 hr #12	4 days 18 hr #5	17 sec	▶

Build History

<div>#12</div> <div>2023. 2. 11. 오후 6:33</div> <div>Started by GitLab push by 최규림</div>	<div>#10</div> <div>2월 12 일 03:02</div> <div>194 commits</div>	1s	455ms	15s	3s	575ms
<div>#11</div> <div>2023. 2. 11. 오후 6:30</div> <div>Started by GitLab push by 최규림</div>	<div>#9</div> <div>2월 12 일 02:28</div> <div>1 commit</div>	570ms	445ms	15s	3s	573ms
<div>#10</div> <div>2023. 2. 11. 오후 6:02</div> <div>Started by GitLab push by 최규림</div>	<div>#8</div> <div>2월 12 일 02:25</div> <div>1 commit</div>	1s	456ms	18s	3s	570ms
<div>#9</div> <div>2023. 2. 11. 오후 5:28</div> <div>Started by GitLab push by 최규림</div>	<div>#7</div> <div>2월 11 일 03:56</div> <div>1 commit</div>	994ms	454ms	16s	641ms	574ms
<div>#8</div> <div>2023. 2. 11. 오후 5:25</div> <div>Started by GitLab push by 최규림</div>	<div>#6</div> <div>2월 11 일 03:22</div> <div>1 commit</div>	600ms	458ms	16s	896ms	573ms
<div>#7</div> <div>2023. 2. 10. 오후 5:56</div> <div>Started by GitLab push by 최규림</div>	<div>#5</div> <div>2월 11 일 02:46</div> <div>1 commit</div>	786ms	645ms	15s	650ms	573ms
<div>#6</div> <div>2023. 2. 10. 오후 6:22</div> <div>Started by GitLab push by 최규림</div>	<div>#4</div> <div>2월 11 일 01:51</div> <div>1 commit</div>	732ms	514ms	23s	3s	590ms

Gitlab Webhook

- Gitlab repository에 push events(Merge Request events) 발생 시, 젠킨스에서 설정 프로젝트 자동 빌드 수행
- Webhook 리스트

Project Hooks (9)		
http://i8b306.p.ssafy.io:8889/project/feat-back-together-pipeline Push Events SSL Verification: enabled	Test ▾	Edit Delete
http://i8b306.p.ssafy.io:8889/project/feat-back-openvidu Push Events SSL Verification: enabled	Test ▾	Edit Delete
http://i8b306.p.ssafy.io:8889/project/feat-back-board-pipeline Push Events SSL Verification: enabled	Test ▾	Edit Delete
http://i8b306.p.ssafy.io:8889/project/feat-back-makeup-pipeline Push Events SSL Verification: enabled	Test ▾	Edit Delete
http://i8b306.p.ssafy.io:8889/project/feat-back-consult-pipeline Push Events SSL Verification: enabled	Test ▾	Edit Delete
http://i8b306.p.ssafy.io:8889/project/feat-back-auth-pipeline Push Events SSL Verification: enabled	Test ▾	Edit Delete
http://i8b306.p.ssafy.io:8889/project/feat-back-flask-pipeline Push Events SSL Verification: enabled	Test ▾	Edit Delete
http://i8b306.p.ssafy.io:8889/project/feat-back-flask-pipeline Push Events SSL Verification: enabled	Test ▾	Edit Delete
http://i8b306.p.ssafy.io:8889/project/dev-front-pipeline Push Events SSL Verification: enabled	Test ▾	Edit Delete

Nginx

- Nginx 설치 및 버전 확인

```
apt-get install nginx
nginx -v
```

- Nginx 중지

```
systemctl stop nginx
```

- Let's Encrypt 설치

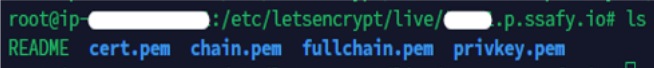
```
apt-get install letsencrypt
```

- 인증서 적용 및 .pem키 발급

```
sudo letsencrypt certonly --standalone -d [도메인]
```

- 발급 경로 확인

```
cd /etc/letsencrypt/live/[도메인]
```



```
root@ip-: /etc/letsencrypt/live/.p.ssafy.io# ls
README cert.pem chain.pem fullchain.pem privkey.pem
```

- Nginx 설정파일 생성
- /etc/nginx/site-available로 이동한 후, 적절한 이름의 파일 생성

```
sudo vim [파일명].conf
```

```

server {
    # 프론트 연결(포트 번호는 본인의 프론트 포트번호를 입력)
    location /{
        proxy_pass http://localhost:3000;
        [redacted]
    }

    [redacted]

    location /ws {
        proxy_pass http://localhost:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header Host $host;
        proxy_set_header Origin "";
    }

    listen 443 ssl; # managed by Certbot
    # 도메인 이름을 써줘야함
    ssl_certificate /etc/letsencrypt/live/i8b306.p.ssafy.io/fullchain.pem; # managed by Certbot
    # 도메인 이름을 써줘야함
    ssl_certificate_key /etc/letsencrypt/live/i8b306.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 {
    # 도메인 이름을 입력
    if ($host = i8b306.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

    listen 80;
    server_name i8b306.p.ssafy.io;
    return 404; # managed by Certbot
}

```

- 파일 연동 테스트

```

sudo ln -s /etc/nginx/sites-available/[파일명].conf /etc/nginx/sites-enabled/[파일명].conf
sudo nginx -t

```

- Nginx 재시작

```

sudo systemctl restart nginx

```

- Nginx 상태 확인

```

systemctl status nginx

```

- keystore.p12 파일 생성

```
cd /etc/letsencrypt/live/[도메인주소]/
openssl pkcs12 -export -in fullchain.pem -inkey privkey.pem -out keystore.p12 -name [이름] -CAfile chain.pem -caname root
```

Redis

- docker-compose.yml

```
version: '3'

services:
  redis:
    image: 'bitnami/redis:latest'
    container_name: redis-master
    environment:
      - REDIS_REPLICATION_MODE=master
      - ALLOW_EMPTY_PASSWORD=yes
    networks:
      - test-network
    ports:
      - 6379:6379

  redis_slave-1:
    image: 'bitnami/redis:latest'
    container_name: redis-slaves-1
    environment:
      - REDIS_REPLICATION_MODE=slave
      - REDIS_MASTER_HOST=redis
      - ALLOW_EMPTY_PASSWORD=yes
    ports:
      - 6479:6379
    depends_on:
      - redis
    networks:
      - test-network

  redis_slave-2:
    image: 'bitnami/redis:latest'
    container_name: redis-slaves-2
    environment:
      - REDIS_REPLICATION_MODE=slave
      - REDIS_MASTER_HOST=redis
      - ALLOW_EMPTY_PASSWORD=yes
    ports:
      - 6579:6379
    depends_on:
      - redis
    networks:
      - test-network

  redis-sentinel-1:
    image: 'bitnami/redis-sentinel:latest'
    container_name: redis-sentinel-1
    environment:
      - REDIS_SENTINEL_DOWN_AFTER_MILLISECONDS=3000
      - REDIS_MASTER_HOST=redis
      - REDIS_MASTER_PORT_NUMBER=6379
      - REDIS_MASTER_SET=mymaster
      - REDIS_SENTINEL_QUORUM=2
    depends_on:
      - redis
      - redis_slave-1
      - redis_slave-2
    ports:
      - '26378:26379'
    networks:
      - test-network

  redis-sentinel-2:
    image: 'bitnami/redis-sentinel:latest'
    container_name: redis-sentinel-2
    environment:
      - REDIS_SENTINEL_DOWN_AFTER_MILLISECONDS=3000
      - REDIS_MASTER_HOST=redis
      - REDIS_MASTER_PORT_NUMBER=6379
      - REDIS_MASTER_SET=mymaster
```

```

- REDIS_SENTINEL_QUORUM=2
depends_on:
- redis
- redis_slave-1
- redis_slave-2
ports:
- '26380:26379'
networks:
- test-network

redis-sentinel-3:
image: 'bitnami/redis-sentinel:latest'
container_name: redis-sentinel-3
environment:
- REDIS_SENTINEL_DOWN_AFTER_MILLISECONDS=3000
- REDIS_MASTER_HOST=redis
- REDIS_MASTER_PORT_NUMBER=6379
- REDIS_MASTER_SET=mymaster
- REDIS_SENTINEL_QUORUM=2
depends_on:
- redis
- redis_slave-1
- redis_slave-2
ports:
- '26381:26379'
networks:
- test-network

networks:
test-network:
external: true

```

```
sudo docker-compose up
```

Openvidu

- openvidu 설치

```

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

```

- openvidu env 파일 수정

```

cd /etc/openvidu
nano .env

```

```

# For example: 198.51.100.1. or openvidu.example.com
DOMAIN_OR_PUBLIC_IP=

# OpenVidu SECRET used for apps to connect to OpenVidu server and
OPENVIDU_SECRET=

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

# If CERTIFICATE_TYPE=letsencrypt, you need to configure a valid
LETSencrypt_EMAIL=

# Proxy configuration
# If you want to change the ports on which openvidu listens, unco

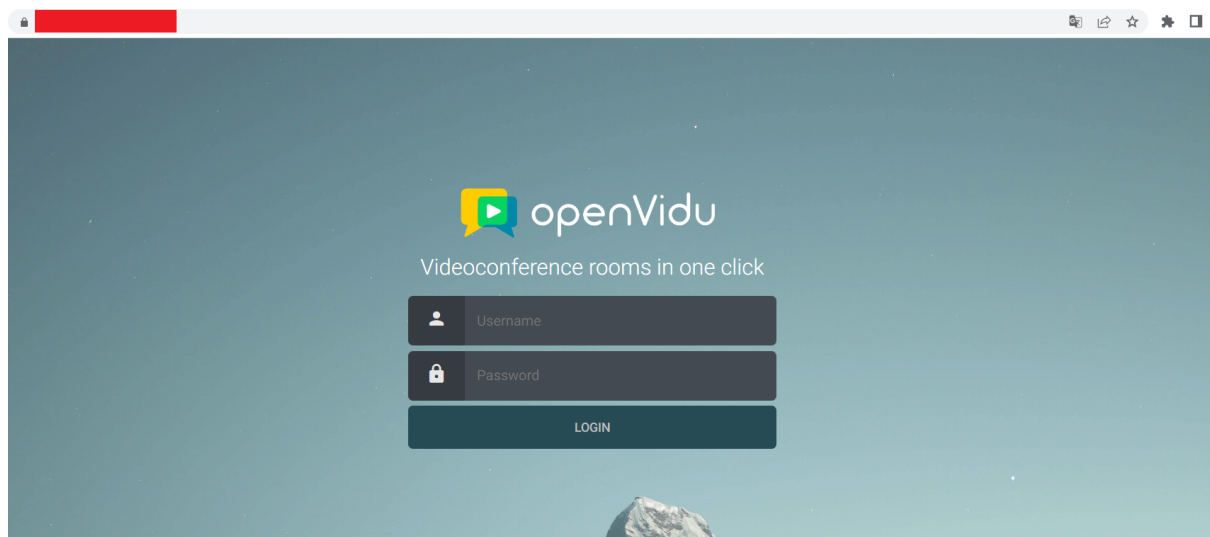
# Allows any request to http://DOMAIN_OR_PUBLIC_IP:HTTP_PORT/ to
# redirected to https://DOMAIN_OR_PUBLIC_IP:HTTPS_PORT/.
# WARNING: the default port 80 cannot be changed during the first
# if you have chosen to deploy with the option CERTIFICATE_TYPE=
HTTP_PORT=

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

```

- Openvidu 실행

```
./openvidu start
```



Flask

- 가상 환경 실행을 위한 pipenv 설치

```
$ pip install pipenv
```

- flask 폴더에서 실행 → 가상환경 만들기

```
$ pipenv shell
```

- flask 폴더 내에 있는 Pipfile 이용해서 설치됨

```
$ pipenv install
```

- 그 외 프로젝트에 사용되는 모듈과 각 모듈의 버전들은 requirements.txt 파일 내에 명시되어 있음

- Run Server ⇒ flask 실행 파일

```
pipenv run python app.py
```

외부 서비스 문서

1. 네이버 로그인

- <https://developers.naver.com/main> 접속
- 애플리케이션 등록

애플리케이션 이름

beauduck

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

카테고리

라이프스타일

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 개인정보를 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 type="checkbox"/>	<input type="checkbox"/>

PC 웹

서비스 URL

https://i8b306.p.ssafy.io/

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** 배치가 노출됩니다.

네이버 로그인

Callback URL (최대 5개)

	—
https://i8b306.p.ssafy.io/Api/Naver	—
	—
	—
	—

로고 이미지 ⇄



파일선택

네이버 로그인 연동 과정에서 사용자에게 보여지는 이미지이므로 서비스를 대표할 수 있는 이미지로 설정해주세요.

권장 크기는 140X140 사이즈이며 500KB 이하의 jpg, png, gif만 등록 가능합니다.