포팅 메뉴얼

목차

목차
사용 기술 스택
빌드 / 배포 메뉴얼
    VS Code AWS SSH 접속
    AWS 환경 설정
    빌드 & 배포
KAKAO Oauth
SMTP

사용 기술 스택

OS

- windows 10, ubuntu 20.04LTS

backend

- spring boot 2.7.2, gradle, java 11
- jpa/querydsl, mysql, jdbc
- spring security, lombok, swagger3.0.0
- openvidu, oauth2

server

- docker, nginx

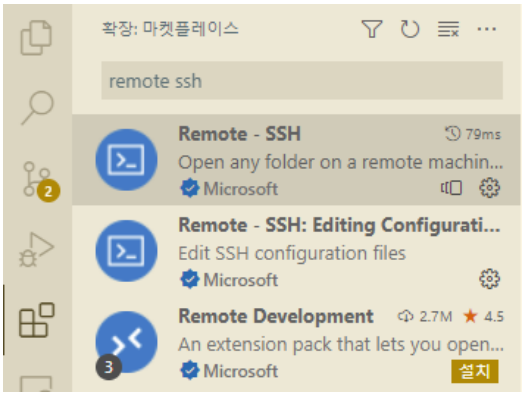
frontend

- react.js 18.0.0, react-router, redux, react-persist, react-mui
- sockjs, stompjs
- 채팅톡 openapi
- openvidu

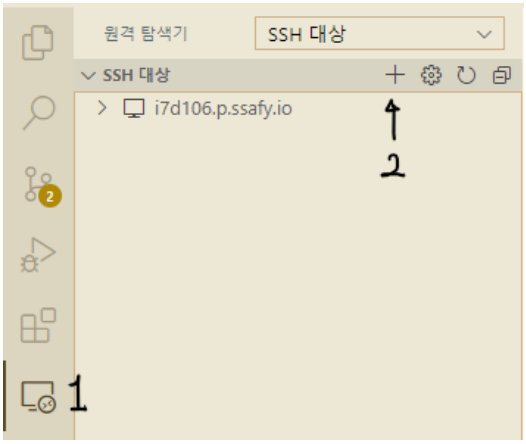
빌드 / 배포 메뉴얼

VS Code AWS SSH 접속

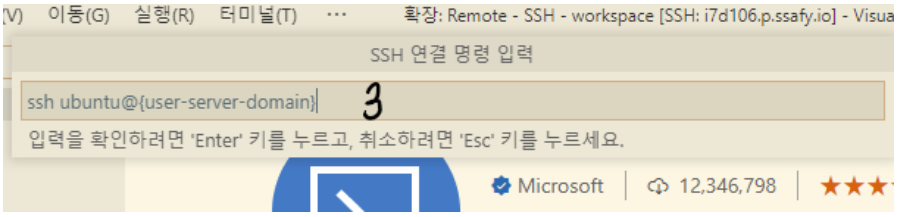
- Visual Studio Code 최신버전 설치 Download Link
- 확장 프로그램 → Remote SSH 검색 후 설치



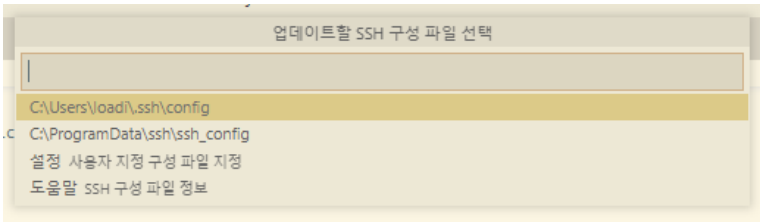
- 1.원격 탐색기 → 2.+버튼으로 새로운 연결 추가



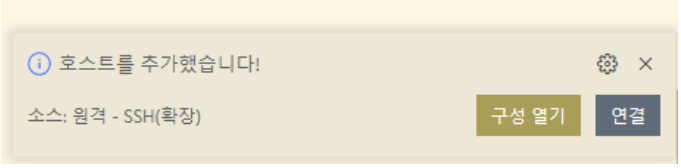
- ssh {계정}@{서버 도메인 또는 서버 주소} 입력 후 엔터



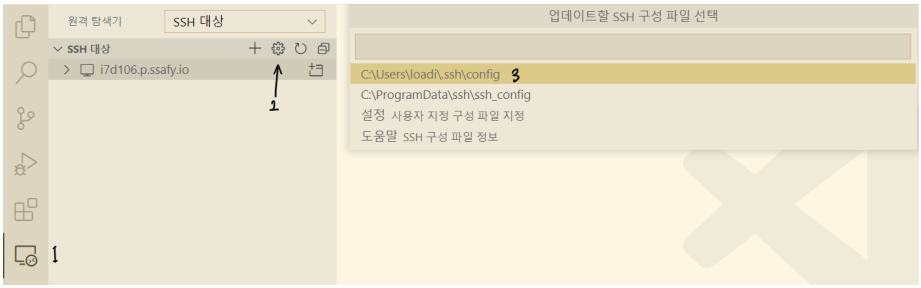
- 구성파일 첫번째꺼 선택 후 엔터



- 우측 하단에 뜨는 창에서 구성 열기 선택



또는 원격 탐색기 → 구성 → 아까 구성파일 선택한거 선택



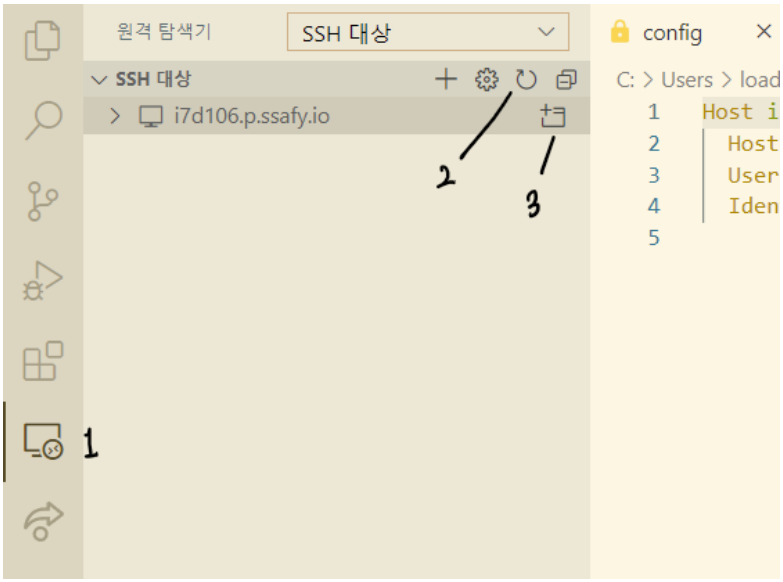
- config file이 열리고 다음과 같이 적혀있습니다.

```
Host {서버 주소 또는 서버 도메인}
  HostName {서버 주소 또는 서버 도메인}
  User ubuntu
```

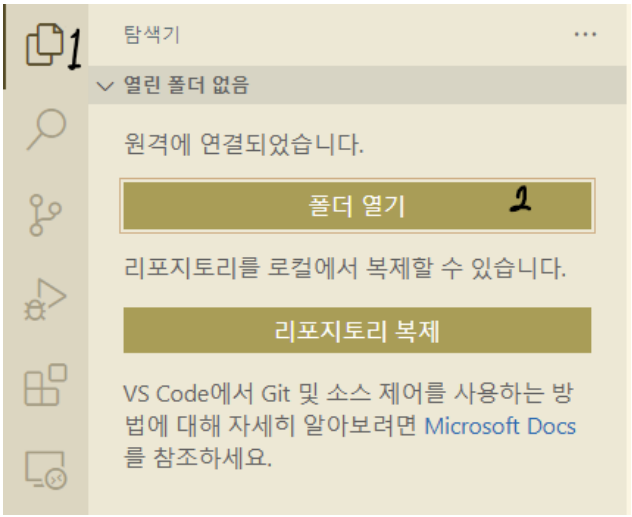
여기에 ssh 접속에 필요한 pem 키의 위치를 적어줍니다.

```
Host {서버 주소 또는 서버 도메인}
  HostName {서버 주소 또는 서버 도메인}
  User {계정}
  IdentityFile {pem키의 디렉토리 위치}

e.g)
Host i7d106.p.ssafy.io
  HostName i7d106.p.ssafy.io
  User ubuntu
  IdentityFile C:\Users\loadi\Desktop\Project\I7D106T.pem
```

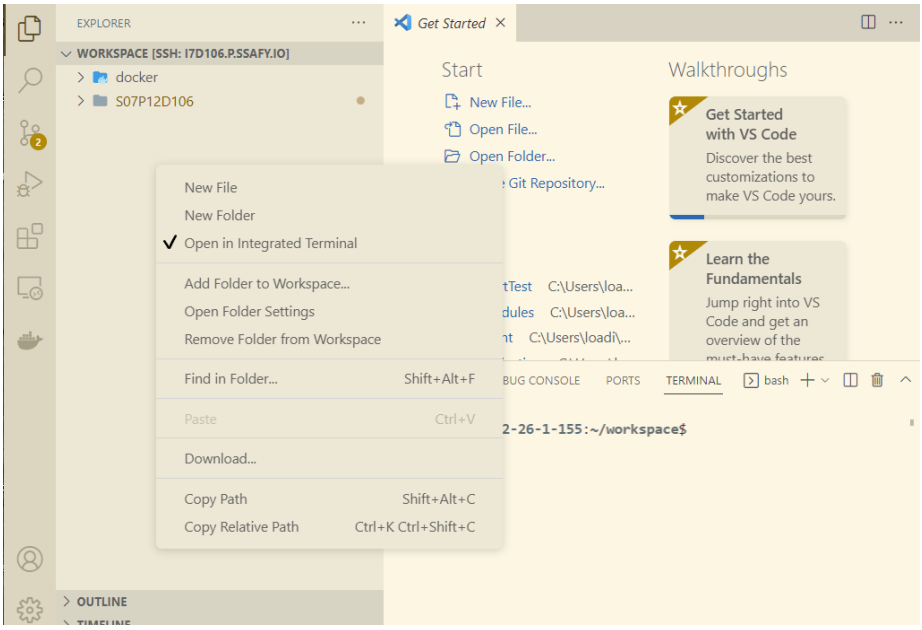
- 원격 탐색기 → 새로고침 → 새 창에서 호스트 연결



- 탐색기 → 폴더 열기 → 원하는 디렉토리 선택 후 확인



- 작업 공간에서 파일 확인 가능,
  우클릭 후 터미널 open 하여 사용



AWS 환경 설정

방화벽

- 방화벽 설정은 반드시 super user 명령어로 실행

```
# AWS 터미널에 명령어 입력

# ufw 설치 확인 및 버전 확인
# 보통 기본적으로 설치 되어있음
$ ufw --version

# 없을 시 설치
$ sudo apt install ufw

# ufw 활성화
$ sudo ufw enable

# ssh
$ sudo ufw allow 22

# http / https
$ sudo ufw allow 80
$ sudo ufw allow 443

# Mysql
$ sudo ufw allow 3306

# api
$ sudo ufw allow 8080
```

JDK 11

- aws에 open jdk 설치

```
# apt 최신으로 update
$ sudo apt-get update && sudo apt-get upgrade
```

```
# open jdk install
$ sudo apt-get install openjdk-11-jdk

# 설치 확인
$ java -version
$ javac -version
```

MySql

- mysql Install

```
# apt update
$ sudo apt-get update

# myqsl server install
$ sudo apt-get install mysql-server

# mysql 실행 확인
$ sudo service mysql status
or
$ sudo systemctl status mysql

# mysql secure 설정
# y 연타 root 계정 설정
$ sudo myslq_secure_installation
```

- mysql 계정 및 db 설정

```
# root 계정으로 mysql 입장
# 초기 비밀번호 없으므로 그냥 엔터
$ mysql -u root -p

# mysql root 비밀번호 설정
mysql> alter user 'root'@'localhost' identified with mysql_native_password by 'password';
mysql> flush privileges;

# mysql 계정 생성
mysql> create user 'user'@'%' identified by 'password';

# mafia database 생성
mysql> create database mafia;

# 접근 권한 설정
mysql> grant all privileges on mafia.* to 'user'@'%';
mysql> flush privileges;
```

- 외부 접속 설정

```
$ sudo vi /etc/mysql/mysql.conf.d/mysqld.cnf

# vim editor 화면에서 bind address를 0.0.0.0으로 변경
bind-address = 0.0.0.0

# :wq로 저장하고 나오기

# mysql 리스타트
$ sudo service mysql restart
or
$ sudo systemctl restart mysql
```

NginX

- Install

```
# apt update
$ sudo apt update

# nginx 설치
$ sudo apt install nginx

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

- nginx config

```
server {
  # front end 빌드 파일 위치
  root /home/ubuntu/workspace/S07P12D106/front/build;

  # front end 인덱스 파일
  index index.html;
  # front end index 파일 매핑
  location / {
    try_files $uri $uri/ /index.html;
  }

  # api 요청 redirect
  # 외부에서 /api uri로 요청 들어올 시
  # 내부 서버의 8000포트로 redirect
  # 내부 api 서버가 다른 포트 사용하면 변경
  location /api {
    proxy_pass http://localhost:8000;
  }

  # https well known port
  listen 443 ssl; # managed by Certbot

  # https 적용을 위한 ssl 인증서 설정
  ssl_certificate /etc/letsencrypt/live/{domain-name}/fullchain.pem; # managed by Certbot
  ssl_certificate_key /etc/letsencrypt/live/{domain-name}/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 {
  # 80포트로 들어오는 요청을 https 로 리다이렉트
  if ($host = i7d106.p.ssafy.io) {
    return 301 https://$host$request_uri;
  } # managed by Certbot

  listen 80 ;
  listen [::]:80 ;
  server_name i7d106.p.ssafy.io;
  return 404; # managed by Certbot
}

server {
  location / {
    # 소켓통신은 헤더 업그레이드를 해주어야 한다.
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection 'upgrade';
    proxy_http_version 1.1;

    # backend api 리다이레트
    # 외부에서 8080포트로 들어오는 요청을 localhost 8000으로 redirect 한다.
    # 만약 backend port 다른거 사용하게 되면 변경
    proxy_pass http://localhost:8000;

  }

  # 우리 프로젝트의 api 요청은 모두 8080 포트로 들어오도록 되어있다.
  # openvidu의 경우 https 가 필요해서 ssl 설정
  listen 8080 ssl;

  # https를 위한 ssl 설정
  # key는 certbot을 이용하여 발급받을 수 있다.
```

```
  ssl_certificate /etc/letsencrypt/live/{domain-name}/fullchain.pem;
  ssl_certificate_key /etc/letsencrypt/live/{domain-name}/privkey.pem;
  include /etc/letsencrypt/options-ssl-nginx.conf;
  ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;
}
```

SSL

- snap install

```
$ sudo apt update
$ sudo apt install snapd
$ sudo snap install core; sudo snap refresh core
```

- certbot install

```
$ sudo snap install --classic certbot
$ sudo ln -s /snap/bin/certbot /usr/bin/certbot
$ sudo certbot --nginx

# 이후 나오는 멘트를 잘 읽어보고
# email과 domain name을 잘 입력해주면

# 다음과 같은 문구가 나오며 인증서가 발급된다.
Successfully received certificate.
Certificate is saved at: /etc/letsencrypt/live/{domain-name}/fullchain.pem
Key is saved at:         /etc/letsencrypt/live/{domain-name}/privkey.pem
```

Docker

- install

```
# old version 삭제
$ sudo apt-get remove docker docker-engine docker.io containerd runc

# apt update
$ sudo apt-get update

# docker 설치를 위한 레파지토리 설정
## 1. 필요한 패키지 다운로드
$ sudo apt-get install \
    ca-certificates \
    curl \
    gnupg \
    lsb-release

## 2. 도커 공식 GPG 등록
$ sudo mkdir -p /etc/apt/keyrings
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

## 3. 다운로드 레파지토리 설정
$ echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

# apt update
# 레파지토리 설정 이후 다시 업데이트 해주세요.
$ sudo apt-get update

# 도커 설치
# 별다른 설정이 없다면 자동으로 최신 버전으로 다운로드
$ sudo apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin

# 특정 버전으로 다운로드 받기
$ sudo apt-get install docker-ce=<VERSION_STRING> docker-ce-cli=<VERSION_STRING> containerd.io docker-compose-plugin
```

```
# 설치 완료 후 hello world 실행 시켜보기
$ sudo docker run hello-world
```

OpenVidu

- 방화벽

```
# 방화벽 해제
## openvidu stun/turn 서버에서 사용함
$ sudo ufw allow 3478

## kurento media server에서 connection을 위해 사용함
$ sudo ufw allow 40000:57000/tcp
$ sudo ufw allow 40000:57000/udp

## Turn 서버가 중계 미디어 connection을 위해 사용함
$ sudo ufw allow 57001:65535/tcp
$ sudo ufw allow 57001:65535/udp

# README
## 서버 내부적으로 80, 443, 3478, 5442, 5443, 6379, 8888 포트는 사용중이면 안된다.
## 위 포트는 오픈비두 플랫폼에서 사용될 수 있다.
```

- Install
  ⚠️ 오픈비두 설치 전에 도커가 꼭 설치되어 있어야 합니다.

```
# super user 권한으로 작업 진행
$ sudo su

# 설치 위치로 이동
$ cd /opt

# 오픈비두 최신버전 다운로드 스크립트 실행
$ curl https://s3-eu-west-1.amazonaws.com/aws.openvidu.io/install_openvidu_latest.sh | bash
# 특정 버전을 다운받고 싶은 경우
$ curl https://s3-eu-west-1.amazonaws.com/aws.openvidu.io/install_openvidu_2.22.0.sh | bash
```

- 환경설정

```
# 다운로드 완료 후 openvidu folder로 이동
$ cd openvidu

# 오픈비두 환경설정 파일 수정
$ vi .env
```

```
# 파일을 보면 주석이 많이 달려있는데
# 꼼꼼하게 읽어보셔야 합니다.
# 필요한 내용들이 모두 적혀있습니다.

# 오픈비두 서버의 도메인 / 서버 주소 를 설정합니다.
DOMAIN_OR_PUBLIC_IP={your-domain-or-ip}
e.g) DOMAIN_OR_PUBLIC_IP=i7d106.p.ssafy.io

# 오픈비두 서버에 접근하기 위한 암호 설정
OPENVIDU_SECRET={your-password}
e.g) OPENVIDU_SECRET=SSAFYD106_SECRET

# 오픈비두는 HTTPS가 필수로 적용되어야 합니다.
# 어떤 방식으로 certificate 할지 결정합니다.
# certificate type은
# - selfsigned
# - owncert
# - letsencrypt
# 세가지 방식이 있습니다.
# 각각의 방식에 대해서는 openvidu 의 설명을 참고하세요
```

```
# 본 프로젝트에서는 letsencrypt 방식을 채택합니다.
CERTIFICATE_TYPE=letsencrypt

# letsencrypt 방식의 certificate는 email이 필요합니다.
# 사용가능한 email을 적어주세요
LETSENCRYPT_EMAIL=your_email@email.com

# 오픈비두 서버 접속시 사용되는 http 포트 번호입니다.
# letsencrypt방식의 certification을 사용할 경우
# 최초 실행시 무조건 80포트를 사용해야 합니다.
# openvidu에서 80번 포트를 이용해 ssl 인증서를 발급받습니다.
# 나중에는 다른걸로 바꿔도 상관 없습니다.
HTTP_PORT=80

# 오픈비두 서버 접속시 사용되는 https 포트번호 입니다.
# http://DOMAIN_OR_PUBLIC_IP:HTTP_PORT/ 로 들어오는 모든 요청은
# https://DOMAIN_OR_PUBLIC_IP:HTTPS_PORT/ 로 자동으로 redirect 됩니다.
# 본 프로젝트에서는 4443번 포트를 사용했습니다.
HTTPS_PORT=4443


# 저장 후 종료
```

- openvidu 실행

```
# nginx 종료
# nginx 에서 80포트를 listen 하고 있기 때문에.
# openvidu letsencrypt 발급을 위해 잠시 nginx를 종료한다.
$ sudo service nginx stop
or
$ sudo systemctl stop nginx

# openvidu 서버 실행
$ ./openvidu start

# openvidu 서버가 잘 실행이 되었는지 브라우저를 dashboard에 접속해보세요
# https://DOMAIN_OR_PUBLIC_IP/dashboard/
# 처음 실행시 약간의 시간이 소요되니
# 안들어가지면 조금 기다렸다가 다시 시도

# dashboard에 잘 접속이 되면 openvidu 종료
$ ./openvidu stop

# 오픈비두는 기본으로 그룹콜 example이 포함되어 있습니다.
# 이를 삭제 해주어야 합니다.
$ rm docker-compose.override.yml

# nginx에서 80포트를 사용하기 때문에
# .env 설정에서 http_port 번호를 다른 것으로 변경해줍니다.
# 본 프로젝트에서는 4442번을 사용했습니다.
$ vi .env

HTTP_PORT=4442
# 저장 후 종료

# 오픈비두 재실행
$ ./openvidu start

# nginx 재실행
$ sudo service nginx start
of
$ sudo systemctl start nginx
```

Nodejs

- install

```
$ curl -fsSL https://deb.nodesource.com/setup_lts.x | sudo -E bash -
$ sudo apt-get install -y nodejs
```

빌드 & 배포

빌드와 배포는 AWS 서버에서 진행됩니다.

먼저 AWS 서버에 프로젝트를 git clone 받습니다.

Backend Build

- application.properties

```
# database 계정 설정
spring.datasource.username={username}
spring.datasource.password={password}

# Openvidu 설정
openvidu.url={your-openvidu-server-url}
openvidu.secret={your-openvidu-server-password}

# SMTP 설정
spring.mail.host=smtp.naver.com
spring.mail.port=465
spring.mail.username={자신의 이메일 ID}
spring.mail.password={자신의 이메일 password}
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.ssl.enable = true
```

- Build

```
# 프로젝트 폴더로 이동
$ cd project/Backend/mafia

# gradlew 실행 가능하도록 권한 변경
$ chmod +x gradlew

# 실행 가능한 Jar 파일로 빌드
$ ./gradlew clean bootJar

# gradle 프로젝트의 경우 ./build/lib에 빌드된다.
# 빌드 완료된 파일을 nohup 을 이용하여 background로 실행시키면 배포완료
$ nohup java -jar ./build/lib/{jarfilename}.jar &
```

Frontend Build

- react build

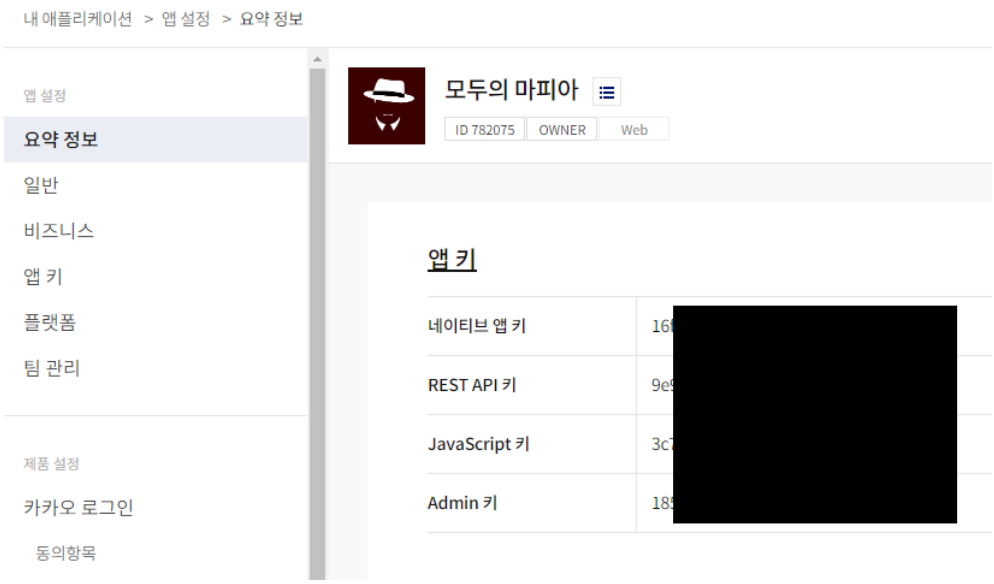
```
# 프로젝트 폴더로 이동
$ cd project/front/

# npm build
$ npm run build

# 이후 build 디렉토리의 경로를 nginx의 root 에다 적어주면 된다!
```

KAKAO Oauth

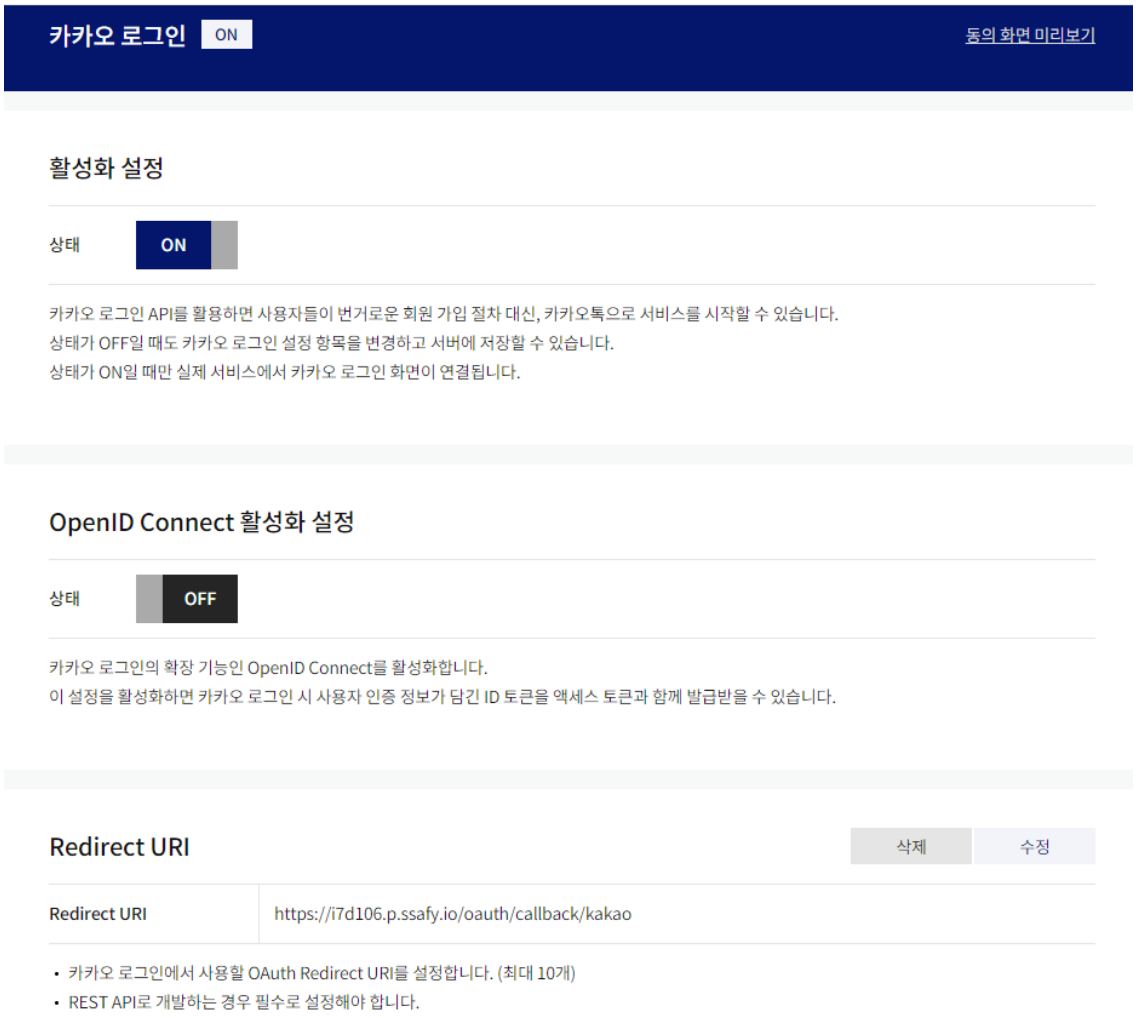
- 카카오 개발자 페이지 들어가서 내 어플리케이션 추가

  https://developers.kakao.com/docs/latest/ko/kakaologin/common

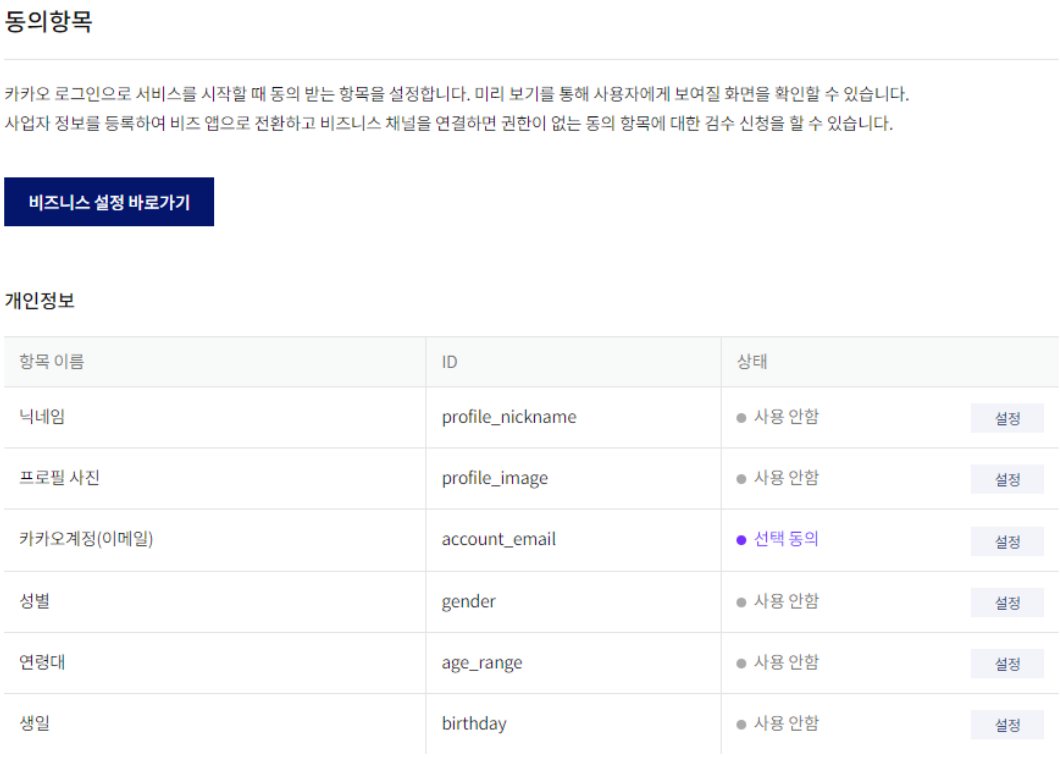
- 요약 정보에 REST API키를 저장한다.



- 내 애플리케이션 → 제품 설정 → 카카오 로그인에서 활성화 설정 상태 ON하고 Redirect URI 설정하기



- 내 애플리케이션 → 제품 설정 → 동의항목 설정에서 개인정보 부분에서 필요정보 설정을 한다.



- 내 애플리케이션 → 제품 설정 → 동의항목 → 동의항목 설정에서 동의단계 및 동의 목적 적기

- 내 애플리케이션 → 앱 설정 → 플랫폼에서 사이트 도메인을 등록한다.

코드 수정

- front → pages → SignInPage 30번째 줄에

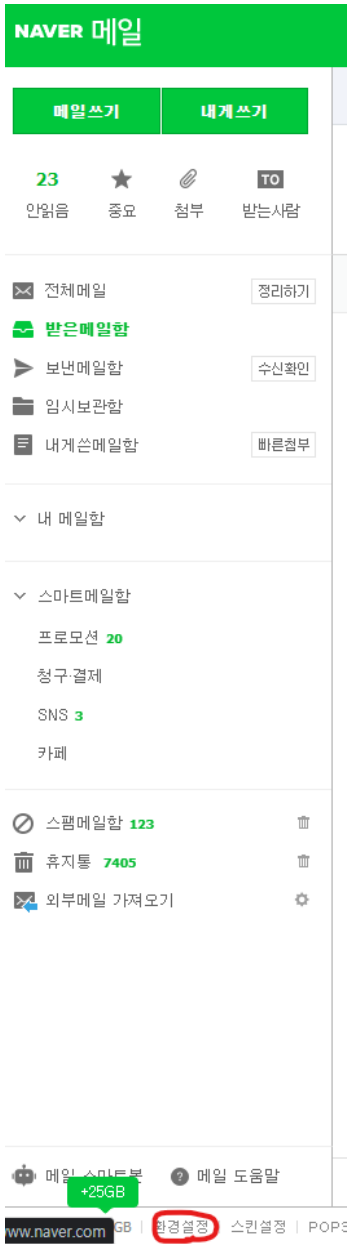
```
const clickKakaoLogin = () => {
    console.log("kakao 로그인");
    window.open('https://kauth.kakao.com/oauth/authorize?client_id={RESTAPI ID}&redirect_uri={도메인 URL}/oauth/callback/ka
  }
```

- backend에서 S07P12D106\Backend\mafia\src\main\java\com\ssafy\mafia\auth\Oauth2\KakaoService 위치에서 21, 22번째줄 수정

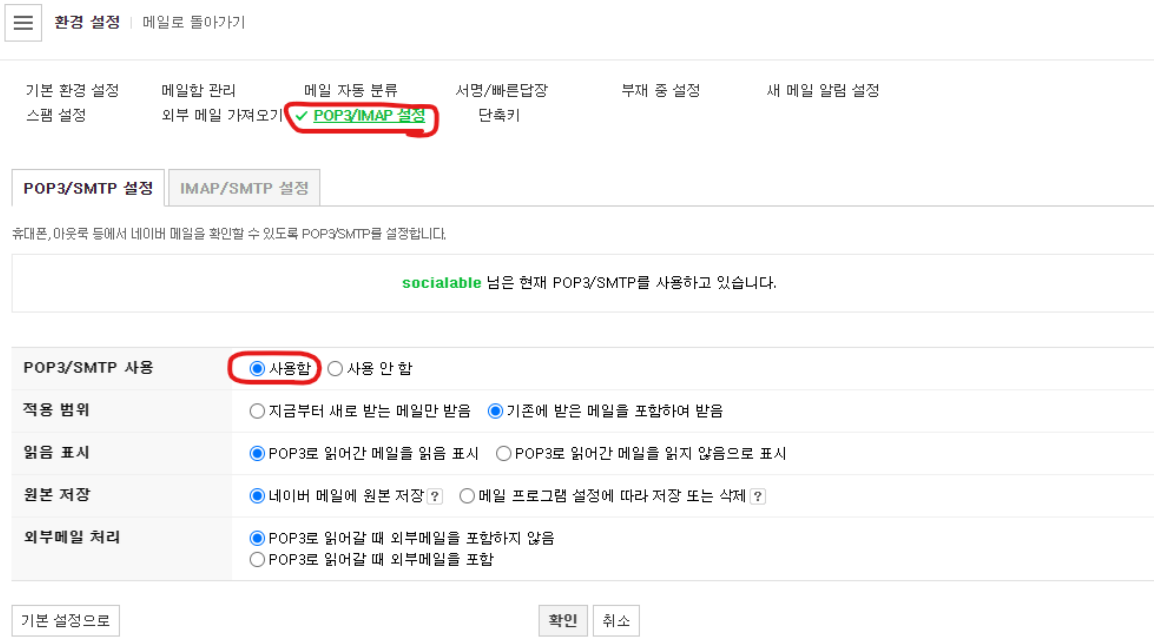
```
private final String kakaoOauth2ClinetId = "{자신의 REST API 아이디}";
private final String frontendRedirectUrl = "{자신의 도메인 URL}";
```

SMTP

- 네이버 메일 ->환경설정에 들어갑니다



- 아래 사진에 표시된 POP3/IMAP 설정에 들어가 POP3/SMTP 사용함으로 하기



- application.properties에 44번째 줄 부터 수정

```
#naver email
spring.mail.host=smtp.naver.com
spring.mail.port=465
spring.mail.username={자신의 이메일 ID}
spring.mail.password={자신의 이메일 password}
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.ssl.enable = true
```