



AGORA

역삼역 인력 사무소



목 차

✓ 빌드 및 배포

1. 개발 환경	-----	3
2. 설정 파일	-----	5
3. 환경 변수	-----	6
4. Jenkins	-----	10
5. Docker	-----	13
6. Nginx	-----	14
7. Openvidu	-----	15
8. Database	-----	16
9. CI/CD	-----	17

✓ 외부 서비스

1. Google Login	-----	25
2. Kakao Login	-----	27

✓ 빌드 및 배포

1. 개발 환경

Client

- | | |
|--------------------|-------------|
| - NodeJS | 18.14.0 LTS |
| - ReactJS | 18.2.0 |
| - Axios | 1.2.4 |
| - Recoil | 0.7.6 |
| - Styled-component | 5.3.6 |

Server

- | | |
|-------------------|---------------|
| - Openjdk | 11 |
| - Spring Boot | 2.7.7 |
| - Spring Security | 2.7.7 starter |
| - Spring Data JPA | 2.7.7 starter |
| - Swagger | 3.0.0 |
| - jjwt | 0.9.1 |
| - QueryDSL | 1.0.10 |

1. 개발 환경

Database

- MYSQL 8.0
- Redis 7.0

Infra

- Docker 20.10.23
- Jenkins 2.375.2
- Nginx 1.18.0
- AWS
- GCP

2. 설정 파일

Server

- application-prod.yml server/src/main/resources
- gcp-account-file.json server/src/main/resources

Infra(AWS)

- custom.conf /etc/nginx/conf.d/custom.conf

Infra(GCP)

- .env /opt/openvidu/.env

- gcp-account-file.json 파일은 gcp에서 제공됩니다.

3. 환경 변수

Server

spring.config.active
on-profile

실행 환경

spring.redis
host
port
password

redis host
redis port
redis password

spring.gcp
config.file
project.id
bucket.id
dir.name

gcp 설정 파일
gcp project id
저장되는 bucket id
이미지 저장 폴더

mvc.pathmatch
matching-strategy

swagger 방식

datasource
url
username
password
driver-class-name

mysql url
mysql username
mysql password
사용할 driver 종류

Server

spring.security.oauth2

client.registration

google

client-id

google client id

client-secret

google secret

redirect-uri

google login redirect uri

scope

google로부터 받을 정보

server.port

실행할 port 번호

openvidu.hostname

openvidu hostname

openvidu.secret

openvidu secret

Infra (custom.conf)

```
location /api {
```

```
    proxy_pass (backend port url)
```

```
}
```

client 에서 요청은 /api/oauth/google로 요청

```
location /api/oauth/google {
```

```
    proxy_pass
```

```
    http://localhost:8081/oauth2/authorization/google
```

```
}
```

```
location /api/oauth2/callback/google {
```

```
    proxy_pass
```

```
    http://localhost:8081/oauth2/callback/google
```

```
}
```

```
server {
```

```
    listen 80;
```

```
    server_name domain name;
```

```
    return 301 https://\$server\_name\$request\_uri;
```

```
}
```


Infra(openvidu .env)

DOMAIN_OR_PUBLIC_IP

프로젝트 domain name

OPENVIDU_SECRET

openvidu secret

CERTIFICATE_TYPE

인증서 타입 선택

LETSencrypt_EMAIL

letsencrypt 선택 시 유효한
이메일 작성

4. Jenkins

설치

```
sudo apt-get update
```

```
sudo apt-get install openjdk-11-jdk -y
```

```
curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo tee \
  /usr/share/keyrings/jenkins-keyring.asc > /dev/null
```

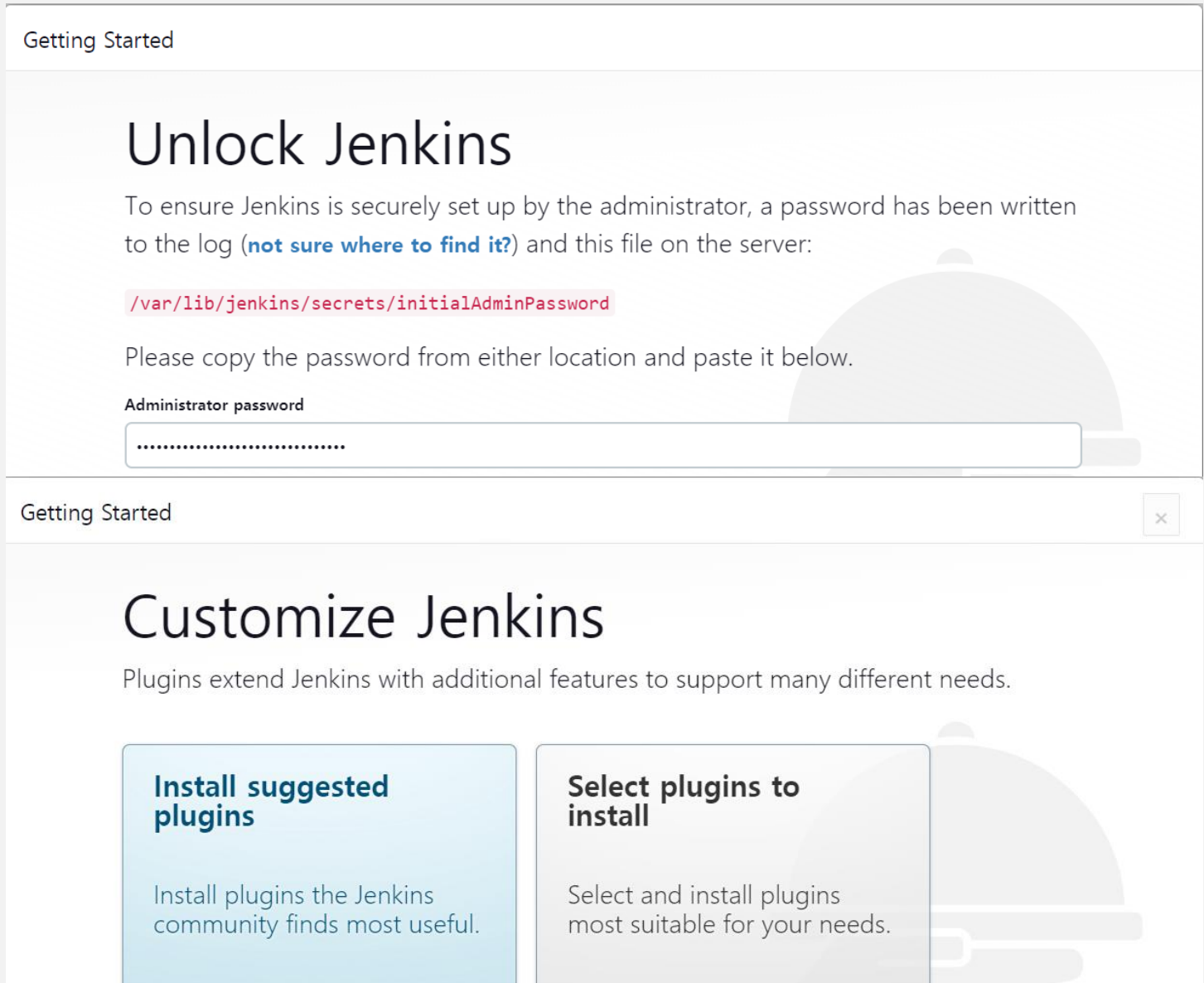
```
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
  https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
  /etc/apt/sources.list.d/jenkins.list > /dev/null
```

```
sudo apt-get update
```

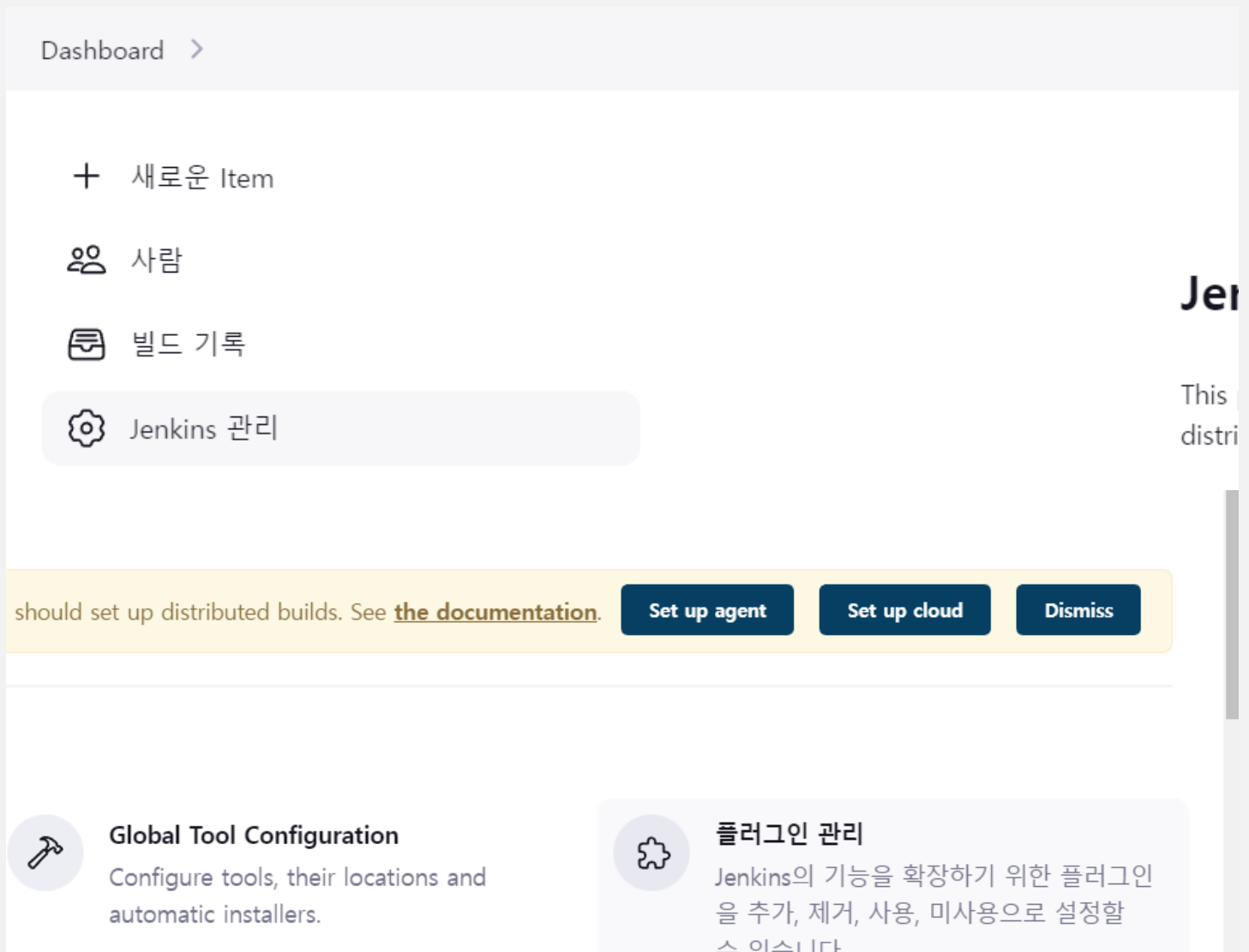
```
sudo apt-get install jenkins
```

- 공식문서에 따른 설치 명령어입니다.

설정



- 설치 후 domain:8080 으로 접속
- `sudo cat /var/lib~~` 명령어로 password 확인
- 필수 플러그인 설치
- 설치 완료 후 Jenkins 접속 password 설정



- Jenkins 관리 -> 플러그인 관리 -> available plugins
- 필요한 플러그인 설치
- docker와 gitlab을 검색하여 연관된 플러그인 모두 설치

5. Docker

설치

```
sudo apt-get update
```

```
sudo apt-get install \  
  ca-certificates \  
  curl \  
  gnupg \  
  lsb-release
```

```
sudo mkdir -m 0755 -p /etc/apt/keyrings
```

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg |  
sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
```

```
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
```

```
sudo apt-get install docker-ce docker-ce-cli containerd.io  
docker-buildx-plugin docker-compose-plugin
```

- 공식문서에 따른 설치 명령어입니다.

설정

```
sudo chmod 666 /var/run/docker.sock
```

- docker 권한 변경하여 sudo 키워드를 넣지 않아도 됩니다.

6. Nginx

설치

```
sudo apt update -y
```

```
sudo apt install nginx -y
```

SSL인증서 설치

```
sudo apt remove certbot
```

```
sudo snap install --classic certbot
```

```
sudo certbot --nginx
```

7. Openvidu

설치

```
sudo su
```

```
cd /opt
```

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

8. Database

MYSQL 설치

```
docker run --name mysql -d -p 3306:3306 --e  
MYSQL_ROOT_PASSWORD=PW mysql:8
```

- docker run 명령어로 mysql을 컨테이너에서 실행

Redis 설치

```
docker run --name redis --network name -p 6379:6379 -d redis  
--requirepass PASSWORD
```

- docker run 명령어로 redis 를 컨테이너에서 실행
- --requirepass로 password 필수 지정

9. CI/CD

Client Dockerfile

```
FROM node:14
ENV REACT_APP_SERVER_BASE_URL (backend base url)
WORKDIR /app
COPY client/package*.json ./
RUN npm install
COPY client/. .
RUN npm run build
```

```
FROM nginx:alpine
COPY --from=0 /app/build /usr/share/nginx/html
COPY client/nginx/nginx.conf /etc/nginx/conf.d/default.conf
EXPOSE 3000
CMD ["nginx", "-g", "daemon off;"]
```

```
# client/nginx/nginx.conf
server {
    listen 3000;
    location / {
        root /usr/share/nginx/html;
        index index.html;
        try_files $uri $uri/ /index.html;
    }
}
```

Client Dockerfile 설명

- ENV로 컨테이너 실행 시 추가할 환경 변수 추가
- jenkins에서 프로젝트 클론 후 빌드되기 때문에 client/폴더로 파일 접근
- 빌드된 파일 컨테이너 nginx 위치로 이동
- client nginx.conf 파일에 빌드된 페이지 경로 설정

Server Dockerfile

```
FROM adoptopenjdk:11-hotspot AS builder
ENV USE_PROFILE dev
...
COPY server/gradlew .
COPY server/gradle gradle
COPY server/build.gradle .
COPY server/settings.gradle .
COPY server/src src
RUN chmod +x ./gradlew
RUN ./gradlew clean compileQuerydsl
RUN ./gradlew clean bootJar

FROM adoptopenjdk:11-hotspot
COPY --from=builder build/libs/*.jar app.jar

EXPOSE 8081
ENTRYPOINT ["java", "-jar", "ENV 설정" "/app.jar"]
```

- 프로젝트 빌드 후 jar 파일 복사
- 환경변수 설정 후 ENTRYPOINT 에 추가
(예시 : "-Dspring.profiles.active=\${USE_PROFILE}")

Jenkins Build (CI)

Enter an item name

» This field cannot be empty, please enter a valid name



Freestyle project

이것은 Jenkins의 주요 기능입니다. Jenkins은 어느 빌드 시스템과 어떤 SCM(형상관리)으로 묶인 당신의 트웨어 빌드보다 다른 어떤 것에 자주 사용될 수 있습니다.



Maven project

Maven 프로젝트를 빌드합니다. Jenkins은 POM 파일의 이점을 가지고 있고 급격히 설정을 줄입니다.



Pipeline

Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines and/or organizing complex activities that do not easily fit in free-style job type.

- Jenkins pipeline 선택하여 프로젝트 생성

Build Triggers

- ☐ Build after other projects are built ?
- ☐ Build periodically ?
- ☒ Build when a change is pushed to GitLab. GitLab webhook URL: <http://i8a705.p.ssafy.io:8080/project/agora-release> ?

Enabled GitLab triggers

- ☐ Push Events
- ☐ Push Events in case of branch delete
- ☐ Opened Merge Request Events
- ☐ Build only if new commits were pushed to Merge Request ?
- ☒ Accepted Merge Request Events

☒ Filter branches by regex ?

Source Branch Regex

(release)

Target Branch Regex

(main)

- ☐ Filter merge request by label

Secret token ?

deae7f47f376f9576231bf1df99cc7b

Generate

- Merge가 되었을 때 빌드가 동작하도록 설정
- 정규식으로 source 브랜치와 target 브랜치 지정

```

1 pipeline {
2   agent any
3   stages {
4     stage('repository clone') {
5       steps {
6         git branch: 'release', credentialsId: 'gitlab', url: 'https://lab.ssafy.com/s08-webmobile1-sub2/S08P12A705.git'
7       }
8     }
9     stage('stop web service') {
10      steps {
11        script {
12          try {
13            sh 'docker rm -f backend && docker rm -f client'
14          } catch (e) {
15            echo 'no container running'
16          }
17        }
18      }
19    }
20    stage('parallel build') {
21      parallel {
22        stage('client build') {
23          steps {
24            sh 'docker build -f client/Dockerfile -t nowgnas/agora:client .'
25          }
26        }
27        stage('server build') {
28          steps {
29            sh 'docker build -f server/Dockerfile -t nowgnas/agora:backend .'
30          }
31        }
32      }
33    }
34    stage('push build images') {
35      steps {
36        sh 'docker login -u nowgnas -p dltdnjs!!'
37        sh 'docker push nowgnas/agora:backend'
38        sh 'docker push nowgnas/agora:client '
39      }
40    }
41  }
42 }

```

- jenkins에서 release 브랜치를 clone 해서 client 와 server를 빌드
- clien와 server 이미지 병렬 빌드
- 빌드된 이미지 docker hub에 push

Build Triggers

☒ Build after other projects are built ?

Projects to watch

agora-release,

☒ Trigger only if build is stable

```
3 stages {
4   stage('stop web service') {
5     steps {
6       script {
7         try {
8           sh 'docker rm -f server && docker rm -f client'
9         } catch (e) {}
10        echo 'no container running'
11      }
12    }
13  }
14  stage('run docker compose') {
15    steps {
16      sh 'docker login -u nowgnas -p dltkddnjs!!'
17      sh 'cd /home/ubuntu && docker-compose up -d'
18    }
19  }
20  stage('remove unused images') {
21    steps {
22      script {
23        try {
24          sh 'docker images -qf dangling=true | xargs -I{} docker rmi {}'
25        } catch (e) {
26          echo 'no unused images'
27        }
28      }
29    }
30  }
31 }
```

- 앞선 빌드가 끝나면 AWS의 docker compose 파일로 서비스 실행
- docker compos로 docker hub의 이미지 pull 한 후 실행
- docker-compose.yml 위치는 AWS의 /home/ubuntu

AWS docker-compose.yml

```
version: '3'
services:
  client:
    image: n[REDACTED]agora:client
    container_name: client
    ports:
      - 3000:3000
    environment:
      REACT_APP_SERVER_BASE_URL: https://i8a705.p.ssafy.io/api
    networks:
      - AGORA
  backend:
    image: r[REDACTED]/agora:backend
    container_name: backend
    ports:
      - 8081:8081
    environment:
      USE_PROFILE: prod
      HOST_NAME: a[REDACTED]
      SCHEMA: b[REDACTED]
      USERNAME: r[REDACTED]
      PASSWORD: d[REDACTED]
      JWT_SECRET: l[REDACTED]
      REDISHOST: r[REDACTED]
      REDISPASS: r[REDACTED]
      MEETING_HOST: https://[REDACTED]
      MEETING_SECRET: ss[REDACTED]
    networks:
      - AGORA
networks:
  AGORA:
    external:
      name: AGORA
```


✓ 외부 서비스

1. Google Login

cp-cloud-instance ▼

리소스, 문서, 제품 등 검색(/)

사용자 인증 정보

+ 사용자 인증 정보 만들기

삭제

삭제된 사용자 인증 정보 복원

사용 설정한 API에 액세스하려면

API 키

API 키

OAuth 클라이언트 ID

할당량과 액세스 권한을 확인하기 위해 간단한 API 키로 프로젝트를 확인합니다.

앱에서 사용자 데이터에 액세스할 수 있도록 사용자 동의를 요청합니다.

← OAuth 클라이언트 ID 만들기

클라이언트 ID는 Google OAuth 서버에서 단일 앱을 식별하는 데 사용됩니다. 앱이 여러 플랫폼에서 실행되는 경우 각각 자체 클라이언트 ID가 있어야 합니다. 자세한 내용은 [OAuth 2.0 설정](#)을 참조하세요. OAuth 클라이언트 유형을 [자세히 알아보세요](#).

애플리케이션 유형 *

웹 애플리케이션 ▼

승인된 리디렉션 URI ?

웹 서버의 요청에 사용

URI 1 *

https://i8a705.p.ssafy.io/api/oauth2/callback/google

+ URI 추가

AGORA

25

1. Google Login

- 사용자 인증 정보 선택, Oauth 클라이언트 ID 만들기 선택
- 프로젝트 이름 설정, redirect uri를 지정
- 현재 프로젝트의 redirect uri는 nginx에서 설정한 location에 따른 경로

2. Kakao Login

Web

[삭제](#)[수정](#)

사이트 도메인	https://i8a705.p.ssafy.io
---------	---------------------------

- 카카오 로그인 사용 시 Redirect URI를 등록해야 합니다. [등록하러가기](#)

Redirect URI

[삭제](#)[수정](#)

Redirect URI	https://i8a705.p.ssafy.io/api/oauth2/callback/kakao
--------------	---

- 카카오 로그인에서 사용할 OAuth Redirect URI를 설정합니다. (최대 10개)

개인정보

항목 이름	ID	상태	
닉네임	profile_nickname	● 필수 동의	설정
프로필 사진	profile_image	● 필수 동의	설정
카카오계정(이메일)	account_email	● 사용 안함	설정
이름	name	○ 권한 없음	

- redirect uri를 지정
- 동의 받을 필수 항목 지정
- client id 와 client secret 확인

*Naver 로그인도 페이지 인증 이슈로 설명 추가하지 않았습니다.