

포팅 메뉴얼

원본 노션 페이지

목차

1. 사용도구
2. 개발도구
3. 개발환경
4. 환경변수 형태
5. Server 구축
6. 빌드 및 실행
7. 외부 서비스 등록

1. 사용 도구

- 이슈 관리 : Jira
- 형상 관리 : GitLab
- 커뮤니케이션 : Notion, MatterMost
- 디자인 : Figma
- UCC : 모바비
- CI/CD : Jenkins

2. 개발 도구

- Visual Studio Code : 1.75.0
- STS : 3.9.14

3. 개발 환경

- FrontEnd
 - Node.js : 18.13.0
 - React : 18.2.0
- BackEnd
 - Java
 - Version : zulu openjdk version "11.0.18" 2023-01-17 LTS
 - SpringBoot : 2.7.7
 - Lombok : 1.18.24
 - Properties File

- application.properties
- application-settings.properties (.gitignore로 제외됨)
- application-dev.properties (.gitignore로 제외됨)
- application-prod.properties (.gitignore로 제외됨)
- Server (EC2 Server)
 - OS : Ubuntu 20.04.5 LTS (GNU/Linux 5.15.0-1028-aws x86_64)
 - Nginx : 1.23.2
 - Jenkins 2.375.2
 - MySQL 8.0.32
- 외부 서비스
 - 네이버 CLOVA OCR
 - OpenVidu : 2.25.0
 - 소셜로그인
 - 구글
 - 카카오

4. 환경변수 형태

- FrontEnd
- BackEnd
 - ▼ application.properties

```
# Run
spring.profiles.active = [실행할 설정 prod or dev]

# Include
spring.profiles.include = [포함할 설정 파일]

#port
server.address = 0.0.0.0
server.port = [배포할 포트 번호]

# mapper location settings
mybatis.type-aliases-package =
mybatis.mapper-locations =

# swagger3
# http://localhost:8080/swagger-ui/index.html
spring.mvc.pathmatch.matching-strategy =

# jwt
# 1 day
jwt.token-validity-in-seconds = [jwt token 시간 1 = 1ms]
# 1 day
jwt.refresh-token-validity-in-seconds = [jwt token 시간 1 = 1ms]

# character set
spring.http.encoding.charset =
spring.http.encoding.enabled =
spring.http.encoding.force =

# file-path
file.images-dir = [프로필 이미지 저장 경로]
file.notice-dir = [공지사항 업로드 파일 저장 경로]
```

- ▼ application-dev.properties & application-prod.properties

```
# Front URL
url.front-url = [실행중인 프론트 URL]

# jwt
```

```

jwt.header = [Token Header]
jwt.secret = [Token 비밀 키]
jwt.access-token = AccessToken
jwt.refresh-token = RefreshToken

# OAuth
# Google
spring.security.oauth2.client.registration.google.client-id = [구글 API 클라이언트 ID]
spring.security.oauth2.client.registration.google.client-secret = [구글 API 클라이언트 secret]
spring.security.oauth2.client.registration.google.scope = [로그인 후 요청할 값]

# Kakao
spring.security.oauth2.client.registration.kakao.client-id = [카카오 API 클라이언트 ID]
spring.security.oauth2.client.registration.kakao.redirect-uri = [카카오 API 리다이렉트 URL]
spring.security.oauth2.client.registration.kakao.authorization-grant-type = authorization_code
spring.security.oauth2.client.registration.kakao.client-authentication-method = POST
spring.security.oauth2.client.registration.kakao.client-name = [클라이언트 이름]
spring.security.oauth2.client.registration.kakao.scope = [로그인 후 요청할 값]
spring.security.oauth2.client.provider.kakao.authorization-uri = [카카오 인증 URI]
spring.security.oauth2.client.provider.kakao.token-uri = [카카오 토큰 URI]
spring.security.oauth2.client.provider.kakao.user-info-uri = [카카오 유저 정보 URI]
spring.security.oauth2.client.provider.kakao.user-name-attribute = [카카오 유저 이름 속성]

```

▼ application-settings.properties

```

# MySQL
spring.datasource.driver-class-name = com.mysql.cj.jdbc.Driver
spring.datasource.url = [접속할 MySQL 주소]
spring.datasource.username = [접속할 유저 이름]
spring.datasource.password = [접속할 유저 암호]

# OpenVidu
OPENVIDU_URL = [도메인 이름 or IP 주소]
OPENVIDU_SECRET = [비밀키]

```

○ 제외된 설정파일을 도커 컨테이너(젠킨스)에 업로드하는 방법

```

설정파일을 모두 작성 후 properties.zip로 압축

// ssh로 ec2서버에 파일 전송
scp -i I8D109T.pem [전송할 파일 이름] [도메인]:~/

// 예
// scp -i I8D109T.pem properties.zip ubuntu@i8d109.p.ssafy.io:~/

// ec2 접속
ssh -i I8D109T.pem ubuntu@i8d109.p.ssafy.io

// ec2에서 docker jenkins로 복사
docker cp properties.zip jenkins:/var/jenkins_home/workspace/Spring/backend/src/main/resources

// 접속
docker exec -it jenkins /bin/bash
cd /var/jenkins_home/workspace/Spring/backend/src/main/resources

// zip 파일이면 unzip
unzip properties.zip

```

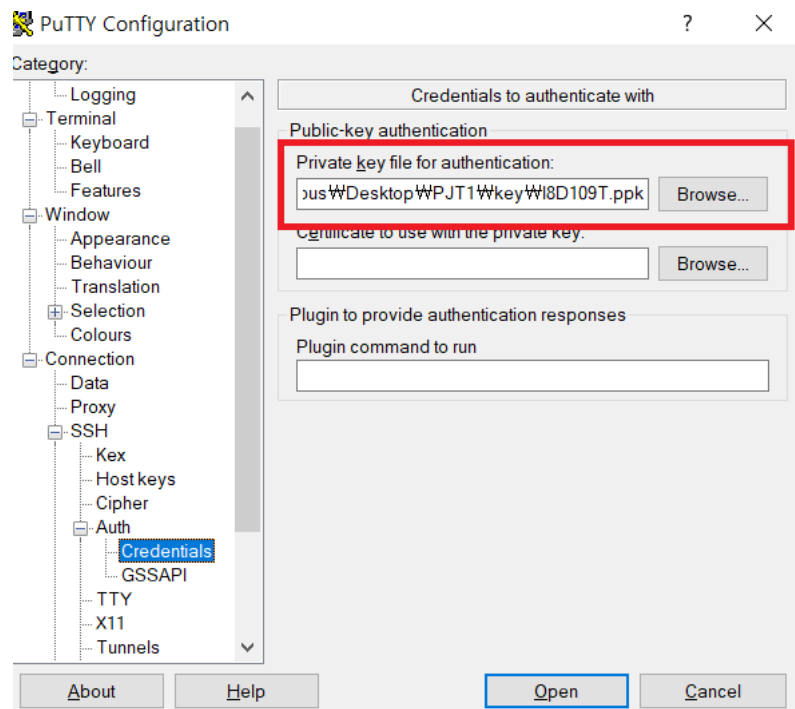
5. Server 구축

▼ 상세 내용

putty 설정

putty 설치 후 설정

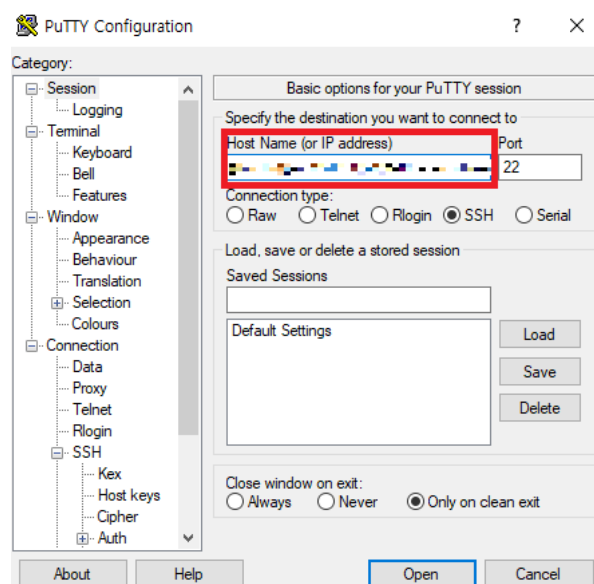
pem키 받은 후 puttygen을 이용해 pem키를 ppk 키로 변경



변환한 키를 Connection/SSH/Auth/Credentials 에 등록

Session 으로 돌아와서 Host Name 은 public ipv4 주소 입력

SSH 설정 하고 포트 확인



설정해둔 값을 save하고 open



기본 값 : ubuntu 입력

Ubuntu에 Docker 설치하기

먼저 기본인 패키징 툴을 업데이트 업그레이드 하기

```
apt update & apt upgrade
```

다음으로 Docker 설치에 필요한 필수 패키지 설치

```
sudo apt-get install apt-transport-https ca-certificates curl gnupg-agent software-properties
```

설치가 되었다면 Docker의 GPG Key 인증하기

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

‘OK’ 가 출력된다면 정상 작동한 것

다음으로 Docker Repository를 등록하기(Docker 환경 구축시 필수!!)

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

repository 등록이 완료 되었다면 apt-get 패키징 툴을 통해 도커 설치

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

설치가 완료되면

```
docker -v
```

하였을때 버전이 나온다면 설치가 잘 된것

Docker 실행하기

```
sudo systemctl enable docker
```

Docker를 활성화 시키고

```
sudo service docker start
```

그리고 Docker가 잘 돌아가고 있는지 확인한다.

```
service docker status
```

```
ubuntu@ip-10.10.10.10:~$ service docker status
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enable)
   Active: active (running) since Mon 2021-12-13 13:11:32 UTC; 2min 15s ago
   TriggeredBy: ● docker.socket
     Docs: https://docs.docker.com
    Main PID: 536 (dockerd)
      Tasks: 7
     Memory: 106.8M
    CGroup: /system.slice/docker.service
            └─536 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.>

Dec 13 13:11:31 ip-10.10.10.10 dockerd[536]: time="2021-12-13T13:11:31.665934197Z" l>
Dec 13 13:11:31 ip-10.10.10.10 dockerd[536]: time="2021-12-13T13:11:31.665993420Z" l>
Dec 13 13:11:31 ip-10.10.10.10 dockerd[536]: time="2021-12-13T13:11:31.666003435Z" l>
Dec 13 13:11:31 ip-10.10.10.10 dockerd[536]: time="2021-12-13T13:11:31.666593775Z" l>
Dec 13 13:11:32 ip-10.10.10.10 dockerd[536]: time="2021-12-13T13:11:32.349634656Z" l>
Dec 13 13:11:32 ip-10.10.10.10 dockerd[536]: time="2021-12-13T13:11:32.494560849Z" l>
Dec 13 13:11:32 ip-10.10.10.10 dockerd[536]: time="2021-12-13T13:11:32.794354809Z" l>
Dec 13 13:11:32 ip-10.10.10.10 dockerd[536]: time="2021-12-13T13:11:32.805295374Z" l>
Dec 13 13:11:32 ip-10.10.10.10 systemd[1]: Started Docker Application Container Engi>
Dec 13 13:11:32 ip-10.10.10.10 dockerd[536]: time="2021-12-13T13:11:32.945410735Z" l>
lines 1-21/21 (END)
```

다음과 같이 출력된다면 도커가 정상적으로 실행되고 있는 것이다.

젠킨스 컨테이너 실행

먼저 젠킨스 이미지를 다운로드 한다.

```
docker pull jenkins/jenkins:lts
```

그리고 젠킨스 컨테이너를 띄운다

```
docker run -d --name jenkins -p 8080:8080 -v /jenkins:/var/jenkins_home -v /usr/bin/docker:/usr/bin/docker -v /var/run/docker.sock:/var/run/docker.sock
```

- **d** : 컨테이너를 **데몬**으로 띄웁니다.
- **p 8080:8080** : 컨테이너 외부와 내부 포트를 **포워딩**합니다. 좌측이 **호스트** 포트, 우측이 **컨테이너** 포트입니다.
- **v /jenkins:/var/jenkins_home** : 도커 컨테이너의 데이터는 **컨테이너가 종료되면 휘발**됩니다. 도커 컨테이너의 데이터를 보존하기 위한 여러 방법이 존재하는데, 그 중 한 방법이 **볼륨 마운트**입니다. 이 옵션을 사용하여 젠킨스 컨테이너의 **/var/jenkins_home** 이라는 디렉토리를 호스트의 **/jenkins** 와 마운트하고 데이터를 보존할 수 있습니다.
- **-name jenkins** : 도커 컨테이너의 이름을 설정합니다.
- **u root** : 컨테이너가 실행될 리눅스의 사용자 계정을 **root** 로 명시합니다.
- **v /usr/bin/docker:/usr/bin/docker -v /var/run/docker.sock:/var/run/docker.sock** 젠킨스 컨테이너에서도 호스트 서버의 도커를 사용하기 위한 바인딩입니다. 이렇게 컨테이너 내부에서 설치없이, 외부의 도커를 사용하는 방식을 **DooD(Dock out of Docker)** 라고 합니다.

[1]

- **p 8080:8080** 젠킨스의 default port 인 8080 을 호스트 서버와 매핑해줍니다.

하지만 위와 같은 명령어를 모두 외우고 있다가, 도커 컨테이너를 실행할 때 마다 입력하게 된다면 굉장히 번거롭다. 따라서 도커는 **docker-compose** 라는 것을 지원한다. 도커 컴포즈는 여러 컨테이너의 실행을 한번에 관리할 수 있게 도와주는 도커의 도구이다. 하지만 저희와 같이 하나의 컨테이너만 필요한 상황에서도 유용하게 사용할 수 있다.

```
sudo apt install docker-compose
```

위 명령을 이용하여 **docker-compose** 를 설치합니다.

그리고 도커를 실행할 경로에 **docker-compose.yml** 이란 파일을 만들고, 아래의 내용을 작성해줍니다.

```
version: "3"
services:
  jenkins:
    image: jenkins/jenkins:lts
    user: root
    volumes:
      - ./jenkins:/var/jenkins_home
    ports:
      - 8080:8080
```

생성한 **docker-compose.yml**

이 존재하는 경로에서 아래의 명령을 실행하면 복잡한 명령 없이도 도커 컨테이너를 실행할 수 있습니다.

```
sudo docker-compose up -d
```

그리고 **jdk** 버전과 **node** 버전을 **docker**에 설치한다.

프로젝트의 버전과 같아야 한다.

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

java -version
```

java -version 을 했을 때 버전 정보가 나온다면 잘 설치된 것이다.

다음으로 JAVA_HOME 설정이다.

```
readlink -f $(which java)
/usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java
```

위 명령어를 통해 출력되는 결과는 다양할 수 있다. 이때, 해당 출력 결과의 끝에서 `/bin/java` 를 제외한 경로를 `[경로]` 라고 하자.

```
sudo vi /etc/profile
```

위 명령어를 통해 켜진 편집기에서 맨 마지막 줄 아래 코드를 입력한다.

```
export JAVA_HOME=[경로]
export PATH=$PATH:$JAVA_HOME/bin

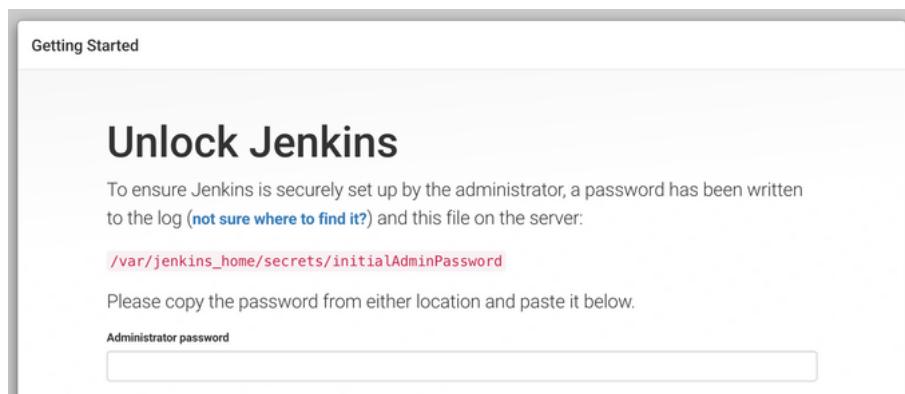
(예시)
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/jre
export PATH=$PATH:$JAVA_HOME/bin
```

마지막으로 JAVA_HOME 적용을 해준다

```
source /etc/profile
```

젠킨스 설정

도커를 사용하여 젠킨스 컨테이너가 EC2 인스턴스에 성공적으로 띄워졌다면, EC2의 퍼블릭 IP를 통해 외부에서 접속할 수 있다. localhost:8080으로 접속하면 아래와 같은 화면이 보인다.



```
sudo docker logs jenkins
```

위 명령을 사용하면, `jenkins` 컨테이너에 출력된 로그를 확인할 수 있다. 젠킨스를 최초로 설치하고 실행하면 사진에서 요구하는 initial admin password를 출력해준다.

```
*****
*****
*****

Jenkins initial setup is required. An admin user has been created and a password generated.

Please use the following password to proceed to installation:

XXXXXXXXXXXXXXXXXXXXXXXXXXXX
```



```
This may also be found at: /var/jenkins_home/secrets/initialAdminPassword
```

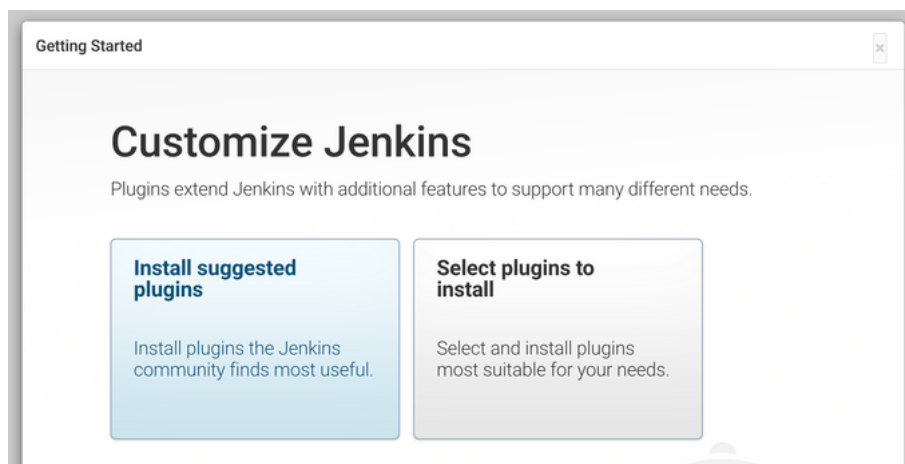
```
*****  
*****  
*****
```

위에서 표시된 `xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx` 를 웹사이트에 넣어주시면 된다.

혹은 아래의 명령으로 `jenkins` 컨테이너 내부에 직접 접속하여, `/var/jenkins_home/secrets/initialAdminPassword` 파일의 내용을 조회하는 방법도 있습니다.

```
$ sudo docker exec -it jenkins /bin/bash  
$ cat /var/jenkins_home/secrets/initialAdminPassword
```

그런 다음 아래 화면에서 install suggested plugins를 눌러준다.



이후 요구되는 정보들을 입력하면 젠킨스 기본 설정이 완료된다.

인스톨 되는 동안 jenkins 안에 Docker를 설치해준다.

jenkins container 접속

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

다음으로 docker를 설치해주면 되는데 위의 docker 설치하기 과정과 동일하게 설정해주면 된다.

젠킨스로 CI


Dashboard >

+ 새로운 Item

 사람

 빌드 기록

 프로젝트 연관 관계

 파일 핑거프린트 확인

 Jenkins 관리

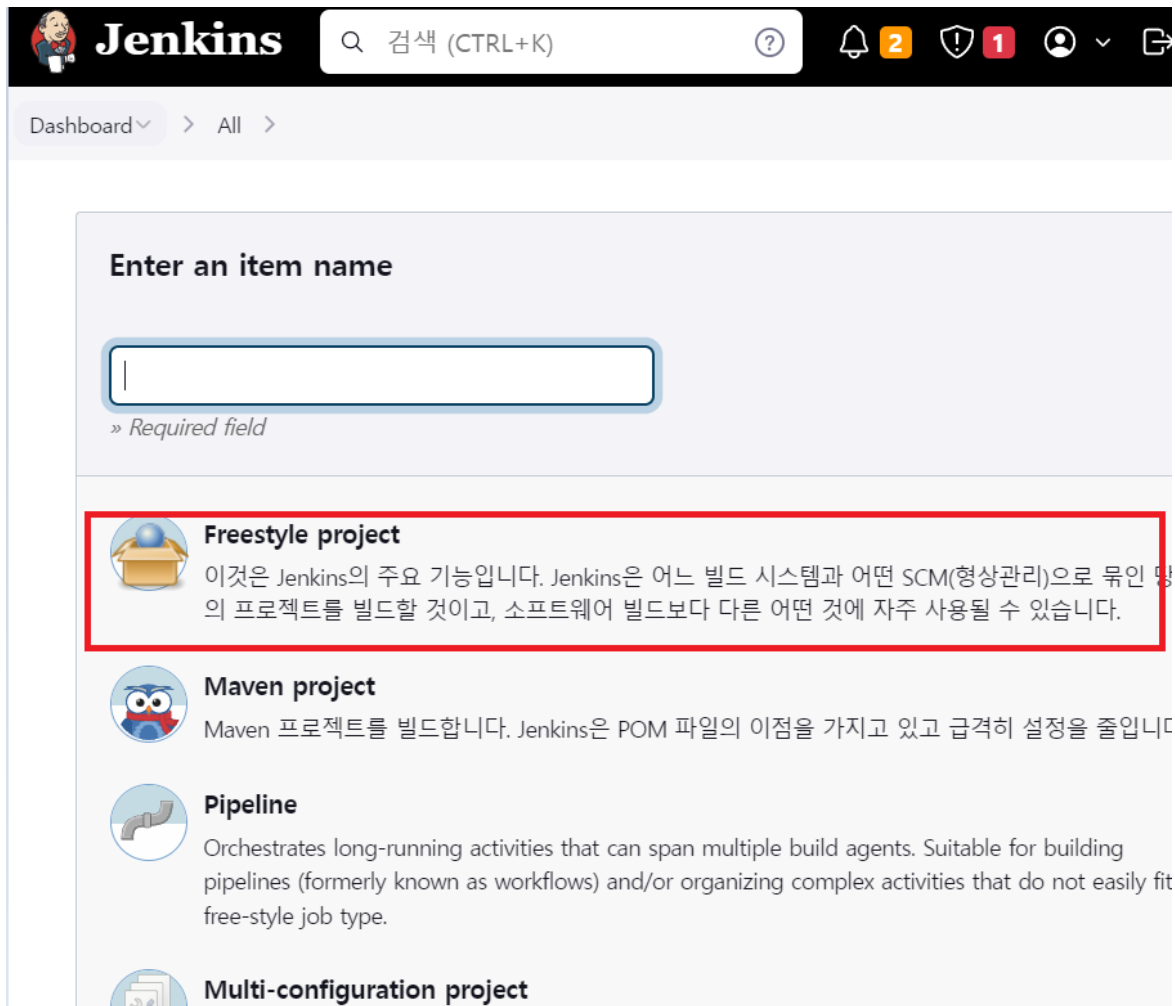
 My Views

빌드 대기 목록



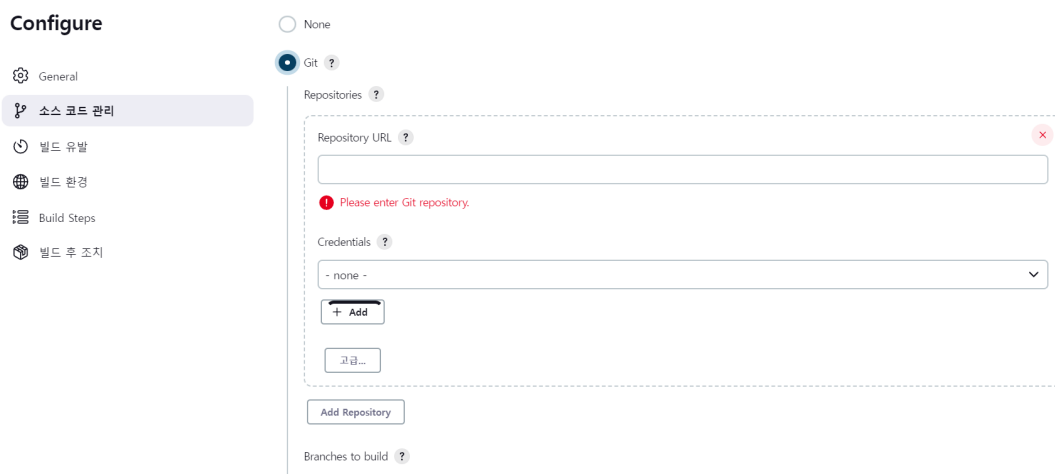
빌드 대기 항목이 없습니다

먼저 새로운 item을 누른다



그리고 설정할 이름을 입력하고 Freestyle project를 누른다

그러면 설정 창이 뜰텐데 아래와 같이 설정해준다.



repository url에 git 프로젝트 url을 써주고 credentials에 add를 눌러 git 계정을 추가해준다.

Domain

Global credentials (unrestricted) ▼

Kind

Username with password ▼

Scope ?

Global (Jenkins, nodes, items, all child items, etc) ▼

Username ?

☐ Treat username as secret ?

Password ?

username은 git 아이디를 입력하고 password 에는 git에서 access token을 발급받아 적어준다.

Repository URL ?

❗ Failed to connect to repository : Command "git ls-remote -h -- https://lab.ssafy.com/s08-webmobile2-sub2/S08P12D109 HEAD" returned status code 128:
 stdout:
 stderr: remote: HTTP Basic: Access denied. The provided password or token is incorrect or your account has 2FA enabled and you must use a personal access token instead of a password. See
https://lab.ssafy.com/help/topics/git/troubleshooting_git#error-on-git-fetch-http-basic-access-denied
 fatal: Authentication failed for 'https://lab.ssafy.com/s08-webmobile2-sub2/S08P12D109.git/'

아이디와 비밀번호를 잘못입력했으면 위와 같은 화면이 뜬다

그리고 아래에 어느 브랜치를 사용할지 적어준다

Branches to build ?

Branch Specifier (blank for 'any') ?

*/feat/be

Add Branch

Repository browser ?

(자동)

Additional Behaviours

Add ▾

그리고 build steps 을 설정한다. add build step을 눌러 아래와 같이 맞춰주면 된다. Maven 설정 법은 아래에서 설명하겠다

Build Steps


≡ Invoke top-level Maven targets ?

Maven Version

maven ▾

Goals

clean package ▾

 고급...

Execute shell ?

Command

See [the list of available environment variables](#)

```

docker build -t test-server ./backend
docker rm -f test-server || true
docker run -d -p 8082:18081 -v images:/usr/app/images --name

```

고급...

처음 화면으로 돌아가 jenkins 관리로 들어가준다. global tool configuration 으로 들어와 maven 을 설정한다. 만약 없다면 maven plugin을 설치한다.

add maven을 하고 프로젝트의 maven 버전과 맞춰준다.

Maven

Name

maven

☒

Install automatically ?

Install from Apache

Version

3.6.3

Add Installer ▾

그리고 서버에 올릴 깃 폴더에 dockfile을 작성해준다.

```

FROM adoptopenjdk/openjdk11
WORKDIR /usr/app
COPY target/code_meets-0.0.1-SNAPSHOT.jar code_meets.jar

```

```
EXPOSE 18081
CMD ["java", "-jar", "code_meets.jar"]
```

visual studio code 로 작업하면 편하다.

그리고 jenkins에서 빌드를 해보면 잘 될 것이다.

react 도 마찬가지로 item 생성하고 Dockerfile을 깃 폴더에 넣어준다

```
FROM node:18WORKDIR
/usr/app
COPY package.json .
RUN npm install
COPY . .
EXPOSE 3000
CMD ["npm", "start"]
```

6. 빌드 및 실행

- FrontEnd

```
npm install
npm start
```

- Backend

```
STS 실행
프로젝트 우클릭
Run As
Spring Boot App
```

7. 외부 서비스 등록

- ▼ 상세내용

Nginx (Letsencrypt) 및 OpenVidu

- ▼ 상세내용

Nginx

1. 설치 : `sudo apt-get install nginx`
2. 설치 확인 : `sudo nginx -v`
3. 중지 : `sudo systemctl stop nginx`
 - a. ssh 인증서 발급을 위해 중지해야 합니다

Let's Encrypt

1. 설치 : `sudo apt-get install letsencrypt`
2. 인증서 적용 및 .pem키 발급: `sudo letsencrypt certonly --standalone -d [도메인]`

명령어 입력 후 아래의 선택지가 발생

 - 이메일 입력 (선택)

- 서비스 이용 동의 (필수)
- 정보 수집 (선택)

위의 과정이 끝나고 Congratulations! 메시지가 보이면 성공

.pem 키의 위치도 조회 가능

3. 인증서 조회 : `sudo ls /etc/letsencrypt/live/[도메인]`

openvidu를 사용하지 않는다면 설치된 nginx의 설정파일을 수정하고 서비스를 다시 시작하면 되지만 openvidu를 사용한다면 openvidu의 실행 서비스 중 nginx가 포함되어 있어 더는 진행하지 않는다

OpenVidu

1. 루트 계정으로 변경 : `sudo su`

2. 폴더 이동 : `cd /opt`

3. openvidu 설치 : `curl https://s3-eu-west-1.amazonaws.com/aws.openvidu.io/install/openvidu/latest.sh | bash`

4. 폴더 이동 : `cd openvidu`

5. 설정 파일 작성 : `nano .env`

- .env

다음과 같은 내용을 작성한다

- DOMAIN_OR_PUBLIC_IP : 도메인 주소
- OPENVIDU_SECRET : 비밀키. 임의로 작성하면 된다. 스프링부트 설정파일에 추가해야하므로 별도로 기록하거나 기억해야한다.
- CERTIFICATE_TYPE : letsencrypt
- LETSENCRYPT_EMAIL : letsencrypt 인증서 생성 시 이메일을 작성했으면 여기에도 작성

```
# 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=[도메인 주소]

# OpenVidu SECRET used for apps to connect to OpenVidu server and users to access to OpenVidu Dashboard
OPENVIDU_SECRET=[비밀키]

# 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=[letsencrypt 이메일]
```

6. 실행 : `./openvidu start`

7. 중지 :

```
-----
OpenVidu Platform is ready!
-----
```



```
* OpenVidu Server: https://DOMAIN_OR_PUBLIC_IP/

* OpenVidu Dashboard: https://DOMAIN_OR_PUBLIC_IP/dashboard/

-----
```

터미널에 위와 같은 문구가 나온다면

`Ctrl + C` 입력 후 `./openvidu stop` 을 입력하여 openvidu를 중지한다

8. nginx 설정 파일 생성

```
docker-compose exec nginx cat /etc/nginx/conf.d/default.conf > custom-nginx.conf
```

```
docker-compose exec nginx cat /etc/nginx/nginx.conf > nginx.conf
```

9. 파일 수정

- docker-compose.yml

```
nano docker-compose.yml
```

```
nginx:
  ...
  volumes:
    ...
    - ./custom-nginx.conf:/custom-nginx/custom-nginx.conf
    - ./nginx.conf:/etc/nginx/nginx.conf
```

- custom-nginx.conf

아래의 설정 내용은

front(react)가 8083 포트로 배포중이며

back(springboot)가 8082 포트로 배포중인 상황이다

또한 소셜 로그인을 위해 /oauth , /login 경로에 대한 설정을 지정했으며

swagger접속을 위해 별도로 설정하였다

```
nano custom-nginx.conf
```

```
# Your App frontend
upstream yourapp {
    server localhost:8083;
}

# Your App backend
upstream backend {
    server localhost:8082;
}

upstream openviduserver {
    server localhost:5443;
}

.
.
.

# Your App
location / {
    proxy_pass http://yourapp; # Openvidu call by default
}

#####
# OpenVidu Locations #
#####

.
.
.

location /openvidu/cdr {
    allow all;
    deny all;
    proxy_pass http://openviduserver;
}

# backend
location /api {
    proxy_pass http://backend;
}
```

```
# social login
location /oauth2 {
    proxy_pass http://backend;
}

location /login {
    proxy_pass http://backend;
}

#swagger
location ~ ^/(swagger|webjars|configuration|swagger-resources|v2|csrf|swagger-ui|v3) {
    proxy_pass http://localhost:8082;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
}

#####
# LetsEncrypt #
#####

.
.
.
```

10. openvidu 재시작 : `./openvidu restart`

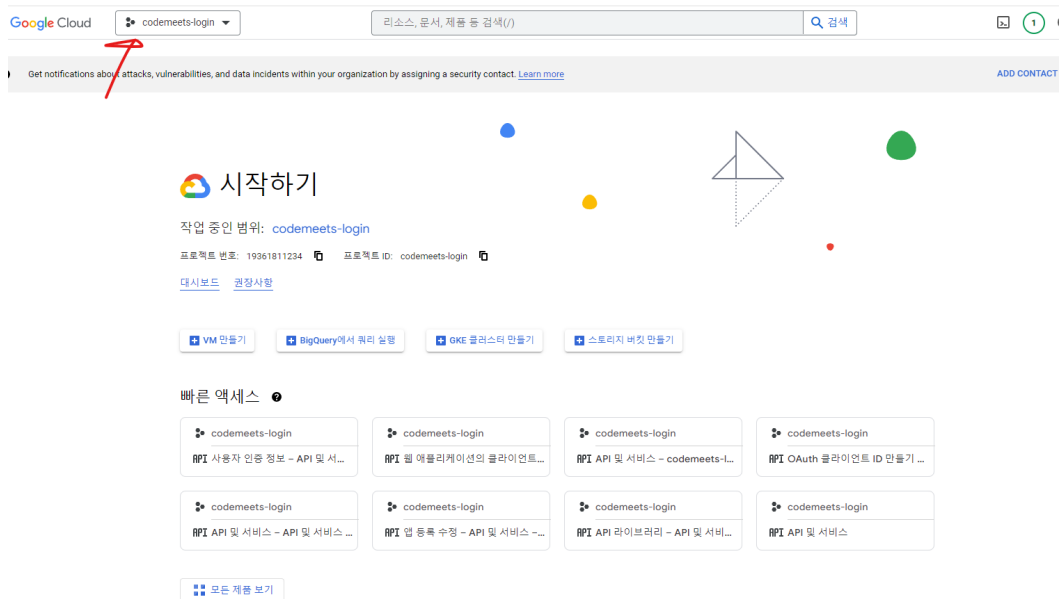
소셜로그인

▼ 상세 내용

Google

▼ 상세 내용

- 구글에 로그인 후 아래 주소로 접속
<https://console.cloud.google.com>
- 프로젝트 생성



프로젝트 선택

새 프로젝트

프로젝트 및 폴더 검색



최근

별표표시된 프로젝트

전체

	이름	ID
✓ ☆ ●	codemeets-login ?	codemeets-login
☆ ●	codemeets-gmail ?	codemeets

취소 열기

새 프로젝트



projects 할당량이 10개 남았습니다. 할당량 증가를 요청하거나 프로젝트를 삭제하세요. [자세히 알아보기](#)

[MANAGE QUOTAS](#)

프로젝트 이름 *

My Project 30997



프로젝트 ID: enduring-broker-378014입니다. 나중에 변경할 수 없습니다. [수정](#)

위치 *

조직 없음

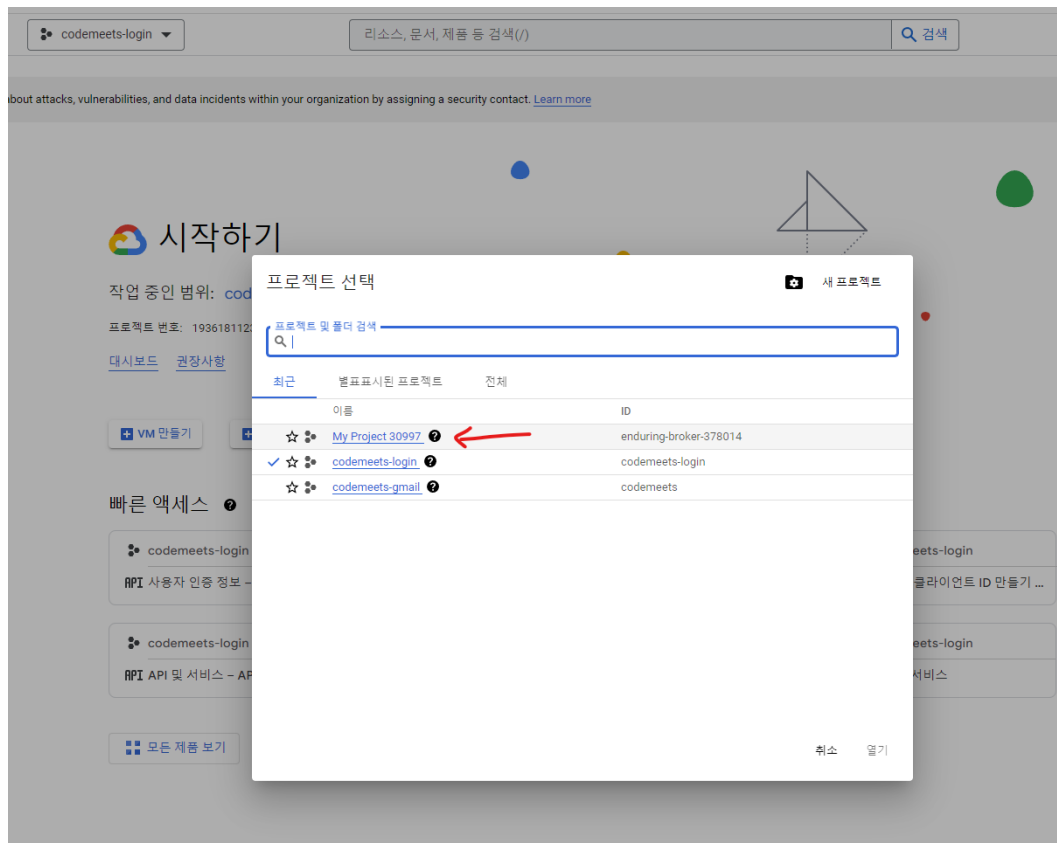
[찾아보기](#)

상위 조직 또는 폴더

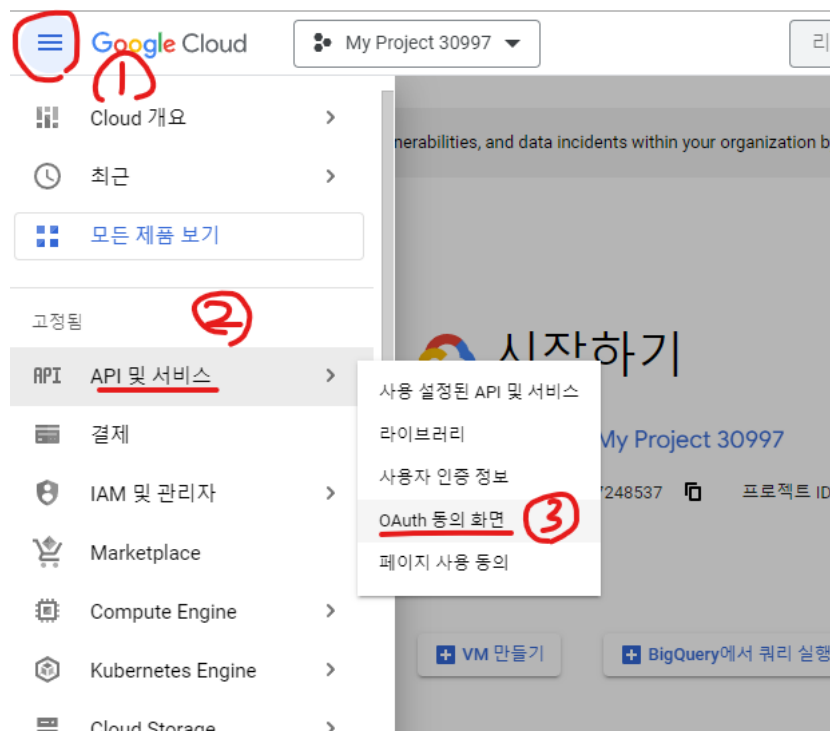
만들기

취소

- 프로젝트로 이동



- OAuth 동의화면으로 이동



- 동의화면 설정

OAuth 동의 화면

대상 사용자를 비롯해 앱을 구성하고 등록하려는 방식을 선택하세요. 프로젝트에는 하나의 앱만 연결할 수 있습니다.

User Type

☐ 내부 ⓘ

조직 내 사용자만 사용할 수 있습니다. 인증을 위해 앱을 제출할 필요는 없습니다. [사용자 유형 자세히 알아보기](#) ↗

☒ 외부 ⓘ

Google 계정이 있는 모든 테스트 사용자가 사용할 수 있습니다. 앱이 테스트 모드로 시작되고 테스트 사용자 목록에 추가된 사용자에게만 제공됩니다. 앱을 프로덕션에 푸시할 준비가 되면 앱을 인증해야 할 수도 있습니다. [사용자 유형 자세히 알아보기](#) ↗

만들기

OAuth 경험에 대한 [의견 보내기](#)

앱 이름 *
test login

동의를 요청하는 앱의 이름

사용자 지원 이메일 *
cmcodemeets@gmail.com

사용자가 동의 관련 질문을 위해 문의할 때 이용합니다.

앱 로고

로고이며 사용자가 앱을 알아보는 데 도움이 되고 OAuth 동의 화면에 표시될 수 있습니다.

업로드할 로고 파일 [찾아보기](#)

사용자가 앱을 알아보는 데 도움이 되도록 동의 화면에 대한 이미지(1MB 이하 크기)를 업로드합니다. 허용되는 이미지 형식은 JPG, PNG, BMP입니다. 최적의 결과를 위해서는 로고가 120x120px 크기의 정사각형이어야 합니다.

앱 도메인

나와 내 사용자를 보호하기 위해 Google에서는 OAuth를 사용하는 앱만 승인된 도메인을 이용할 수 있도록 허용합니다. 다음 정보가 동의 화면에서 사용자에게 표시됩니다.

애플리케이션 홈페이지
https://i8d109.p.ssafy.io/codemeets/login

사용자에게 홈페이지 링크를 제공합니다.

애플리케이션 개인정보처리방침 링크
https://i8d109.p.ssafy.io

사용자에게 공개 개인정보처리방침 링크를 제공합니다.

애플리케이션 서비스 약관 링크

사용자에게 공개 서비스 약관 링크를 제공합니다.

승인된 도메인

동의 화면 또는 OAuth 클라이언트 구성에서 도메인이 사용되면 여기에서 사전 등록해야 합니다. 앱이 인증을 거쳐야 하는 경우 [Google Search Console](#)로 이동하여 도메인이 승인되었는지 확인하세요. 승인된 도메인 한도에 대해 [자세히 알아보세요](#).

승인된 도메인 1 *
ssafy.io

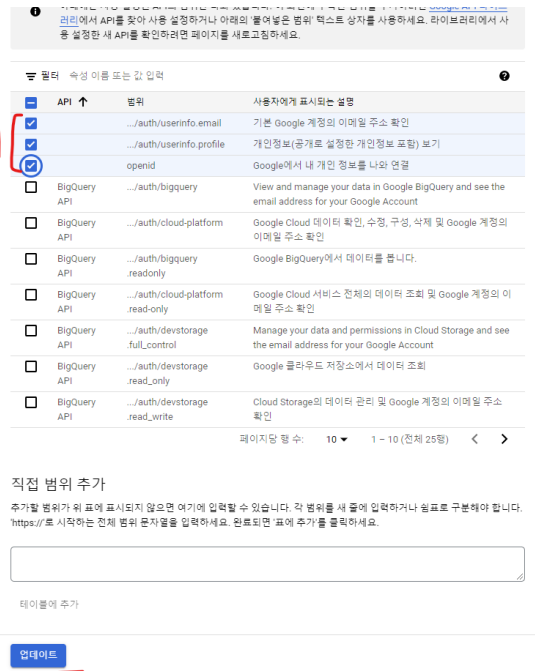
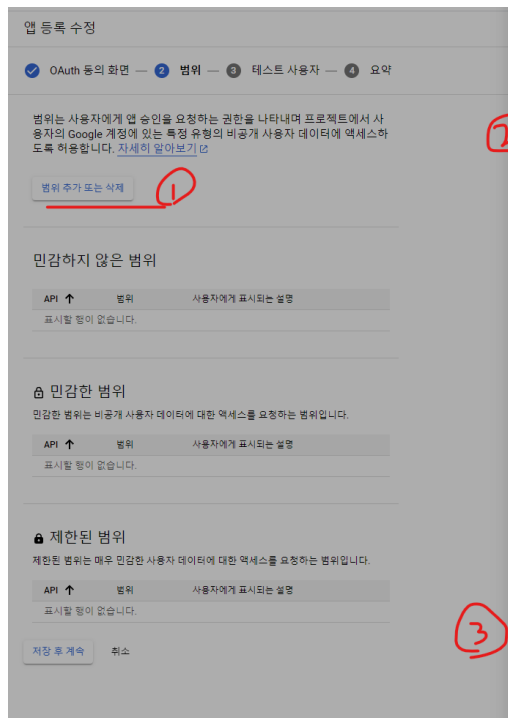
[+ 도메인 추가](#)

개발자 연락처 정보

이메일 주소 *
cmcodemeets@gmail.com

이 이메일 주소는 Google에서 프로젝트 변경사항에 대해 알림을 보내기 위한 용도입니다.

[저장 후 계속](#) [취소](#)



이하 생략하고 계속 진행

OAuth 동의 화면

수정

사용자 유형

외부

앱 이름

test login

지원 이메일

cmcode-meets@gmail.com

앱 로고

입력한 값 없음

애플리케이션 홈페이지 링크

https://i8d109.p.ssafy.io/codemeets/login

애플리케이션 개인정보처리방침 링크

https://i8d109.p.ssafy.io

애플리케이션 서비스 약관 링크

입력한 값 없음

승인된 도메인

ssafy.io

담당자 이메일 주소

cmcode-meets@gmail.com

범위

수정

API ↑	범위	사용자에게 표시되는 설명
...	.../auth/userinfo.email	기본 Google 계정의 이메일 주소 확인
...	.../auth/userinfo.profile	개인정보(공개로 설정한 개인정보 포함) 보기
	openid	Google에서 내 개인 정보를 나와 연결

테스트 사용자

수정

사용자 0명(테스트 0명, 기타 0명)/사용자 한도100명

필터 속성 이름 또는 값 입력

사용자 정보

표시할 행이 없습니다.

대시보드로 돌아가기

• 사용자 인증 정보 생성

Google Cloud My Project 30997 리소스, 문서, 제품 등 검색(/)

API API 및 서비스

사용 설정된 API 및 서비스 라이브러리 사용자 인증 정보 OAuth 동의 화면 페이지 사용 동의

사용자 인증 정보 + 사용자 인증 정보 만들기 삭제 삭제된 사용자 인증 정보 복원

사용 설정한 API에 액세스하려면

API 키

이름 표시할 API 키가 없습니다.

OAuth 2.0 클라이언트 ID

이름 생성일 ↓ 표시할 OAuth 클라이언트가 없습니다.

API 키

발당량과 액세스 권한을 확인하기 위해 간단한 API 키로 프로젝트를 확인합니다.

OAuth 클라이언트 ID

앱에서 사용자 데이터에 액세스할 수 있도록 사용자 동의를 요청합니다.

서비스 계정

로봇 계정을 사용하여 서버 간의 앱 수준 인증을 사용 설정합니다.

사용자 인증 정보 선택 도움말

사용할 사용자 인증 정보의 유형을 결정할 수 있도록 몇 가지 질문을 합니다.

아래 설정 중 승인된 리디렉션 URI는

1. https 서비스인 경우
2. 로컬에서 18081 포트로 백엔드 서비스 (Spring Boot)가 실행되는 경우
3. http 서비스인 경우

적용된다

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

이름 *
웹 클라이언트 1

OAuth 2.0 클라이언트의 이름입니다. 이 이름은 콘솔에서 클라이언트를 식별하는 용도로만 사용되며 최종 사용자에게 표시되지 않습니다.

아래에 추가한 URI의 도메인이 **승인된 도메인**으로 OAuth 등의 화면에 자동으로 추가됩니다.

승인된 자바스크립트 원본 ?

브라우저 요청에 사용

+ URI 추가

승인된 리디렉션 URI ?

웹 서버의 요청에 사용

URI 1 *
https://i8d109.p.ssafy.io/login/oauth2/code/google

URI 2 *
http://localhost:18081/login/oauth2/code/google

URI 3 *
http://i8d109.p.ssafy.io/login/oauth2/code/google

+ URI 추가

참고: 설정이 적용되는 데 5분에서 몇 시간이 걸릴 수 있습니다.

만들기 취소

- 설정 완료

클라이언트 ID와 비밀번호는 Spring Boot 설정 파일에 필요하다

OAuth 클라이언트 생성됨

API 및 서비스의 사용자 인증 정보에서 언제든지 클라이언트 ID와 보안 비밀에 액세스할 수 있습니다.

i OAuth 액세스는 [OAuth 동의 화면](#)에 나열된 [테스트 사용자](#)로 제한됩니다.

클라이언트 ID	[REDACTED]	🔗
클라이언트 보안 비밀번호	[REDACTED]	🔗
생성일	2023년 2월 16일 PM 11시 34분 23초 GMT+9	
상태	✅ 사용 설정됨	

📄 JSON 다운로드

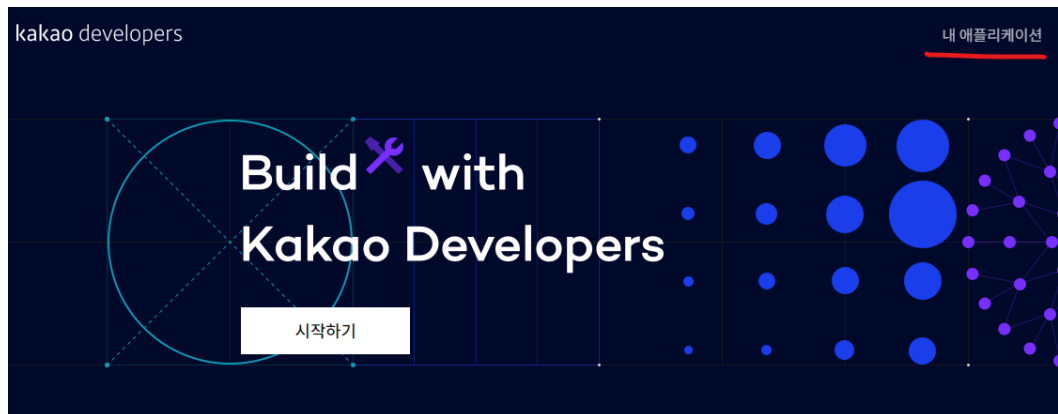
[확인](#)

Kakao

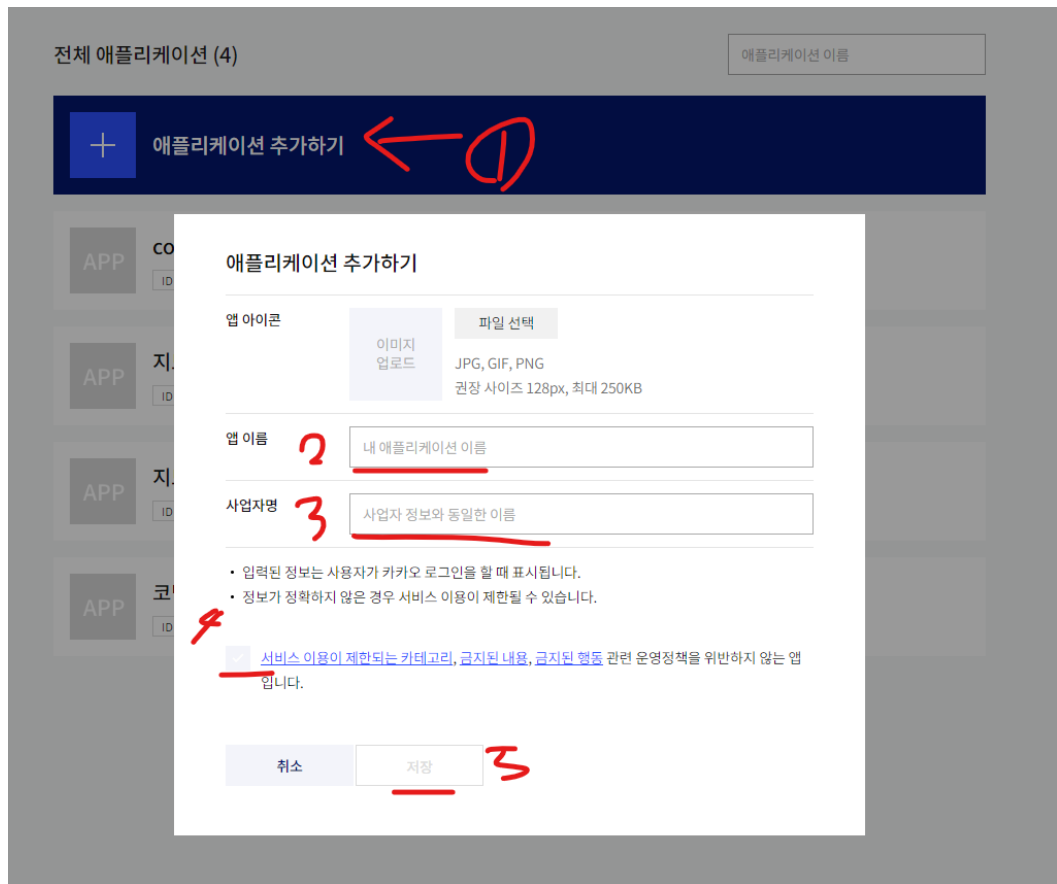
▼ 상세 내용

1. 홈페이지 접속 및 내 애플리케이션 접속 후 로그인

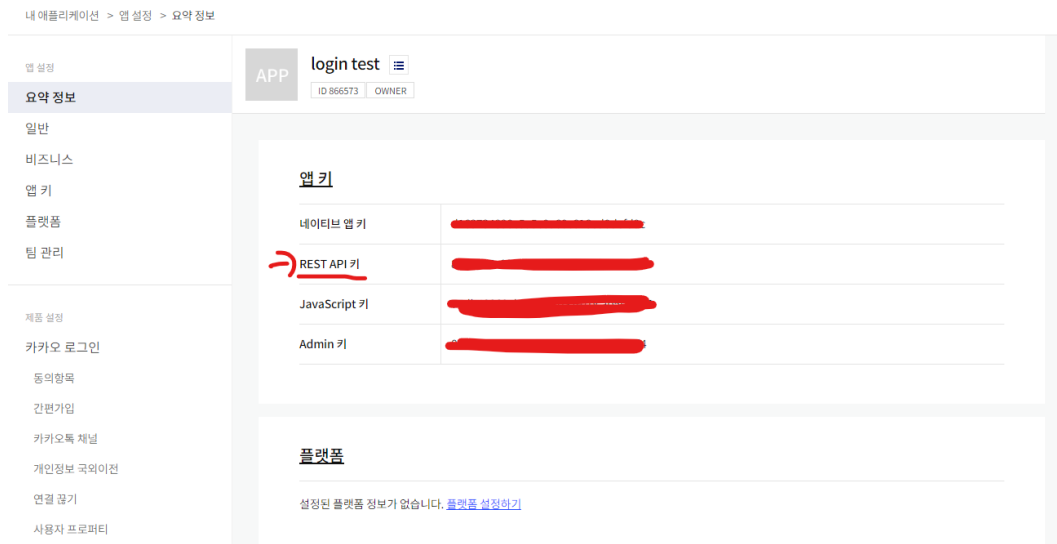
<https://developers.kakao.com/>



2. 애플리케이션 생성



3. 애플리케이션을 생성하면 발급받는 앱 키 중 REST API 키를 사용한다



4. 카카오 로그인 설정

앱 설정

약관 정보

일반

비즈니스

앱 키

플랫폼

팀 관리

제품 설정

카카오 로그인

동의항목

간편가입

카카오톡 채널

개인정보 국외이전

연결 끊기

사용자 프로퍼티

보안

보안 이벤트

고급

메시지

카카오톡 채널

APP login test

ID 866573 OWNER

카카오 로그인 OFF

동의 화면 미리보기

활성화 설정

상태 OFF

카카오 로그인 API를 활용하면 사용자들이 번거로운 회원 가입 절차 대신, 카카오톡으로 서비스를 시작할 수 있습니다. 상태가 OFF일 때도 카카오 로그인 설정 항목을 변경하고 서버에 저장할 수 있습니다. 상태가 ON일 때만 실제 서비스에서 카카오 로그인 화면이 연결됩니다.

Redirect URI

Redirect URI 등록

- 카카오 로그인에서 사용할 OAuth Redirect URI를 설정합니다. (최대 10개)
- REST API로 개발하는 경우 필수로 설정해야 합니다.

카카오 로그인 활성화

카카오 로그인을 활성화 하시겠습니까?

동의 항목이 설정되지 않으면 회원식별값만 전달됩니다. 동의 항목 설정이 필요한지 확인하세요.

취소

활성화

활성화 설정

상태

ON

카카오 로그인 API를 활용하면 사용자들이 번거로운 회원 가입 절차 대신, 카카오톡으로 서비스를 시작할 수 있습니다.
상태가 OFF일 때도 카카오 로그인 설정 항목을 변경하고 서버에 저장할 수 있습니다.
상태가 ON일 때만 실제 서비스에서 카카오 로그인 화면이 연결됩니다.

OpenID Connect 활성화 설정

상태

OFF

카카오 로그인의 확장 기능인 OpenID Connect를 활성화합니다.
이 설정을 활성화하면 카카오 로그인 시 사용자 인증 정보가 담긴 ID 토큰을 액세스 토큰과 함께 발급받을 수 있습니다.

Redirect URI

[Redirect URI 등록](#)

- 카카오 로그인에서 사용할 OAuth Redirect URI를 설정합니다. (최대 10개)
- REST API로 개발하는 경우 필수로 설정해야 합니다.

위의 주소는 https 서비스를 위한 URI
아래의 주소는 로컬환경을 위한 URI 이다.

Redirect URI

Redirect URI

카카오 로그인에서 사용할 OAuth Redirect URI를 설정합니다.

여러개의 URI를 줄바꿈으로 추가해주세요. (최대 10개)

REST API로 개발하는 경우 필수로 설정해야 합니다.

예시: (O) <https://example.com/oauth> (X) <https://www.example.com/oauth>

```
https://i8d109.p.ssafy.io/login/oauth2/code/kakao
http://localhost:18081/login/oauth2/code/kakao
```

취소

저장

5. 동의항목 설정

내 애플리케이션 > 제품 설정 > 카카오 로그인 > 동의항목

앱 설정

약관 정보

일반

비즈니스

앱 키

플랫폼

팀 관리

제품 설정

카카오 로그인

동의항목

간편가입

카카오톡 채널

개인정보 국외이전

연결 끊기

사용자 프로퍼티

보안

보안 이벤트

고급

메시지

카카오톡 채널

푸시 알림

고급 설정

허용 IP 주소

APP login test

ID 866573 OWNER

카카오 로그인 ON

동의 화면 미리보기

동의항목

카카오 로그인으로 서비스를 시작할 때 동의 받는 항목을 설정합니다. 미리 보기를 통해 사용자에게 보여질 화면을 확인할 수 있습니다. 사업자 정보를 등록하여 비즈 앱으로 전환하고 비즈니스 채널을 연결하면 권한이 없는 동의 항목에 대한 검수 신청을 할 수 있습니다.

비즈니스 설정 바로가기

개인정보

항목 이름	ID	상태
닉네임	profile_nickname	● 사용 안함 <div>설정</div>
프로필 사진	profile_image	● 사용 안함 <div>설정</div>
카카오계정(이메일)	account_email	● 사용 안함 <div>설정</div>
이름	name	○ 권한 없음
성별	gender	● 사용 안함 <div>설정</div>
연령대	age_range	● 사용 안함 <div>설정</div>
생일	birthday	● 사용 안함 <div>설정</div>

동의 항목 설정

항목

닉네임 / profile_nickname

동의 단계

☐ 필수 동의

카카오 로그인 시 사용자가 필수로 동의해야 합니다.

☒ 선택 동의

사용자가 동의하지 않아도 카카오 로그인을 완료할 수 있습니다.

☐ 이용 중 동의

카카오 로그인 시 동의를 받지 않고, 항목이 필요한 시점에 동의를 받습니다.

☐ 사용 안함

사용자에게 동의를 요청하지 않습니다.

동의 목적 [필수]

사용자 닉네임 대체

개발자 앱 동의 항목 관리 화면내에 입력하는 사실이 실제 서비스 내용과 다를 경우 API 서비스의 거부 사유가 될 수 있습니다.

취소

저장

6. 사용자 프로퍼티

일반

비즈니스

앱 키

플랫폼

팀 관리

제품 설정

카카오 로그인

동의할목

간편가입

카카오톡 채널

개인정보 국외이전

연결 끊기

사용자 프로퍼티

보안

보안 이벤트

고급

메시지

카카오톡 채널

푸시 알림

카카오 로그인 ON

사용자 프로퍼티

사용자 정보는 각 프로퍼티에 저장되며, 프로퍼티를 추가할 수 있습니다.
팀관리에 등록된 계정으로 카카오 로그인을 실행하면, 설정한 프로퍼티 정보가 정상 수집되는지 목록에서 확인할 수 있습니다.

프로퍼티 추가

프로퍼티 삭제

계정 정보	카카오 로그인 연결된 팀원 계정
id	카카오 로그인을 실행한 팀원 계정이 없습니다.
status	
registered_at	
예약된 프로퍼티	
msg_blocked	
사용자 프로퍼티	

프로퍼티 추가

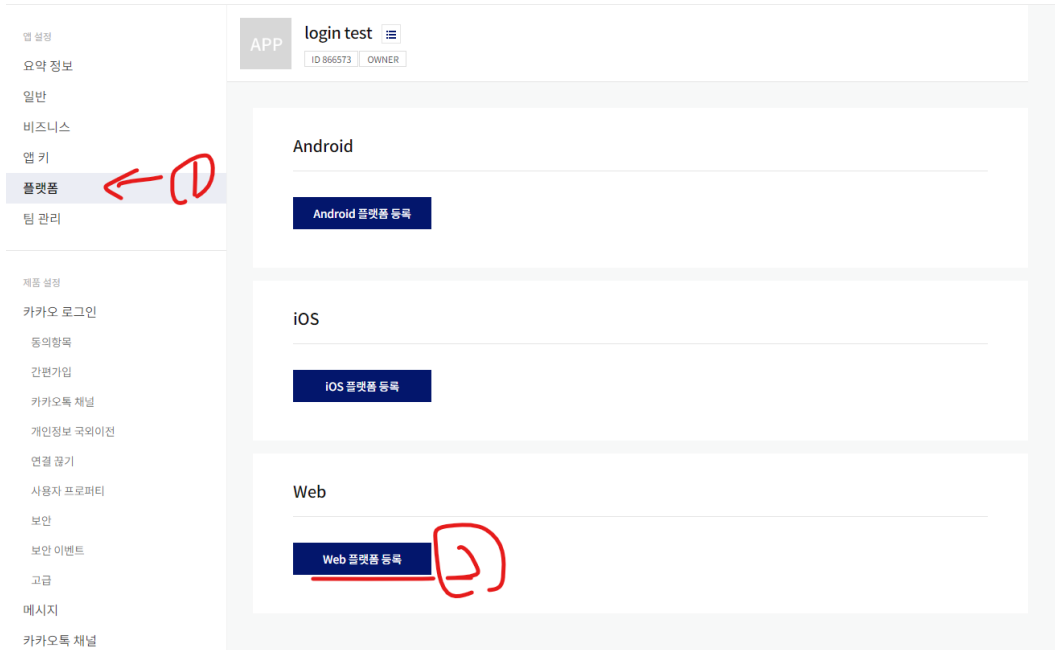
이름 [필수]

카카오 로그인 프로퍼티

취소

추가

7. 플랫폼 추가



Web 플랫폼 등록

사이트 도메인

JavaScript SDK, 카카오톡 공유, 카카오맵, 메시지 API 사용시 등록이 필요합니다.

여러개의 도메인은 줄바꿈으로 추가해주세요. 최대 10까지 등록 가능합니다. 추가 등록은 포럼(데브톡)으로 문의주세요.

예시: (O) <https://example.com> (X) <https://www.example.com>

<https://i8d109.p.ssafy.io>

기본 도메인

기본 도메인은 첫 번째 사이트 도메인으로, 카카오톡 공유와 카카오맵 메시지 API를 통해 발송되는 메시지의 Web 링크 기본값으로 사용됩니다.

<https://i8d109.p.ssafy.io>

취소

저장

Spring Boot 설정

- 4 환경변수 형태 참고바람

CLOVA OCR(NAVER)

▼ 상세 내용

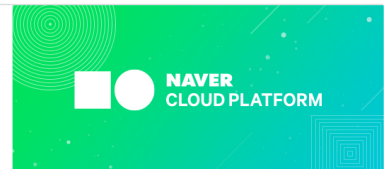
- step 1. 서비스 이용 신청 및 약관 동의
- step 2. 도메인 생성
- step 3. Secret Key 생성 및 API Gateway 연결
- step 4. 프로젝트 연동
- step 5. proxy 설정

STEP 1. 서비스 이용 신청 및 약관 동의

NAVER CLOUD PLATFORM

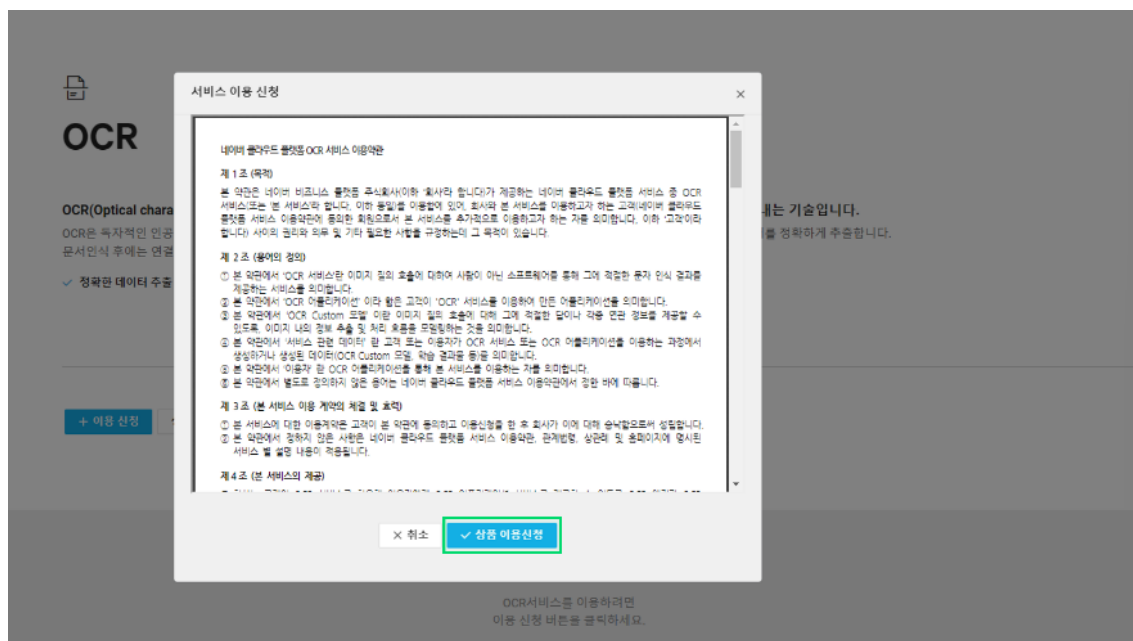
cloud computing services for corporations, IaaS, PaaS, SaaS, with Global region and Security Technology Certification

https://www.ncloud.com/product/aiService/ocr

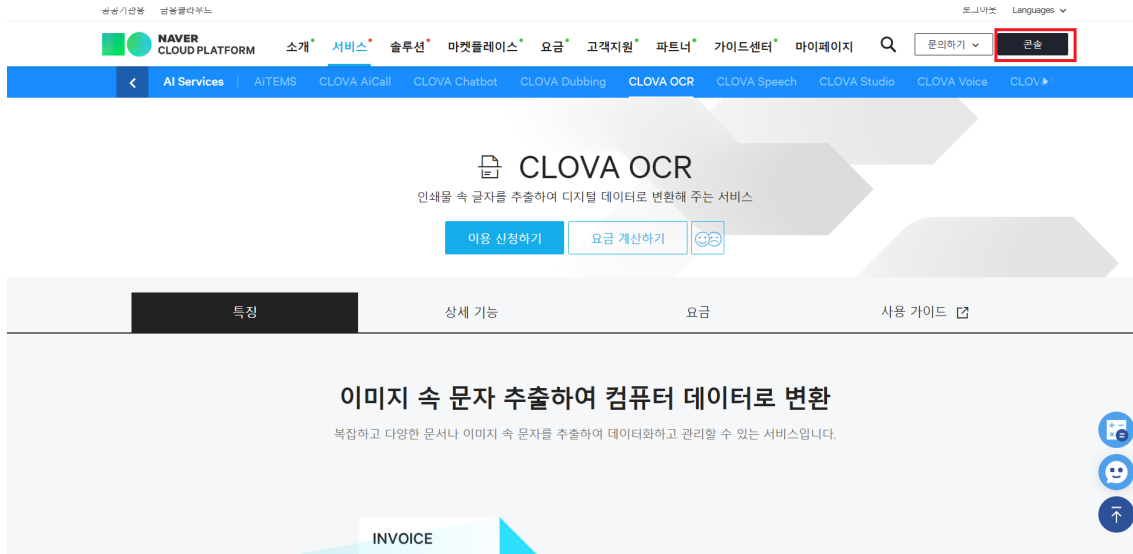


서비스 이용 약관은 CLOVA OCR에서 발생하는 데이터의 보관 및 사용에 대한 내용과 개인 정보 위수탁, 회사의 의무 및 고객의 의무와 관련된 내용을 담고 있습니다. 서비스를 기획하기 전에 CLOVA OCR 이용약관을 반드시 확인해 주십시오.

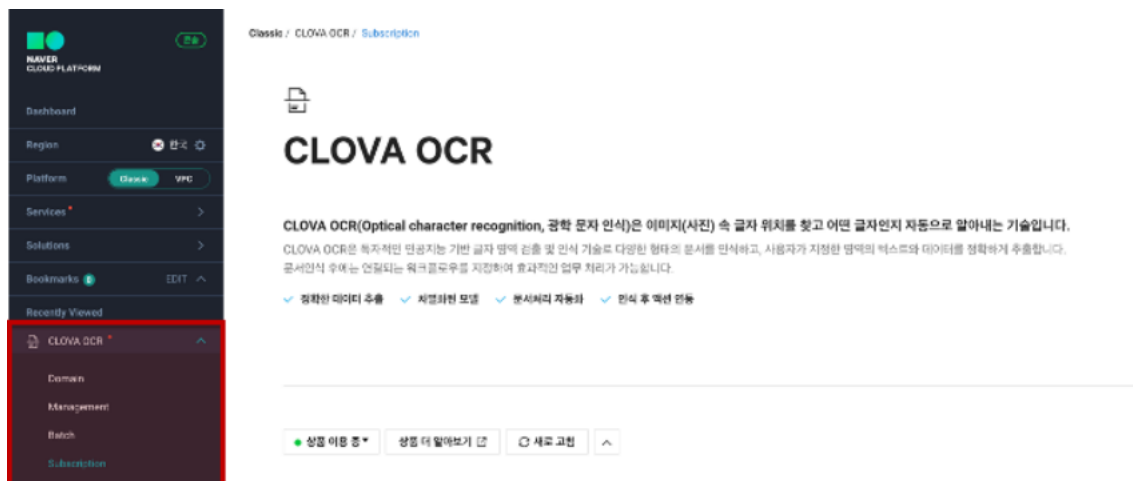
[상품 이용신청] 버튼을 클릭해 약관에 동의한 후 다음 단계로 이동해 주십시오.



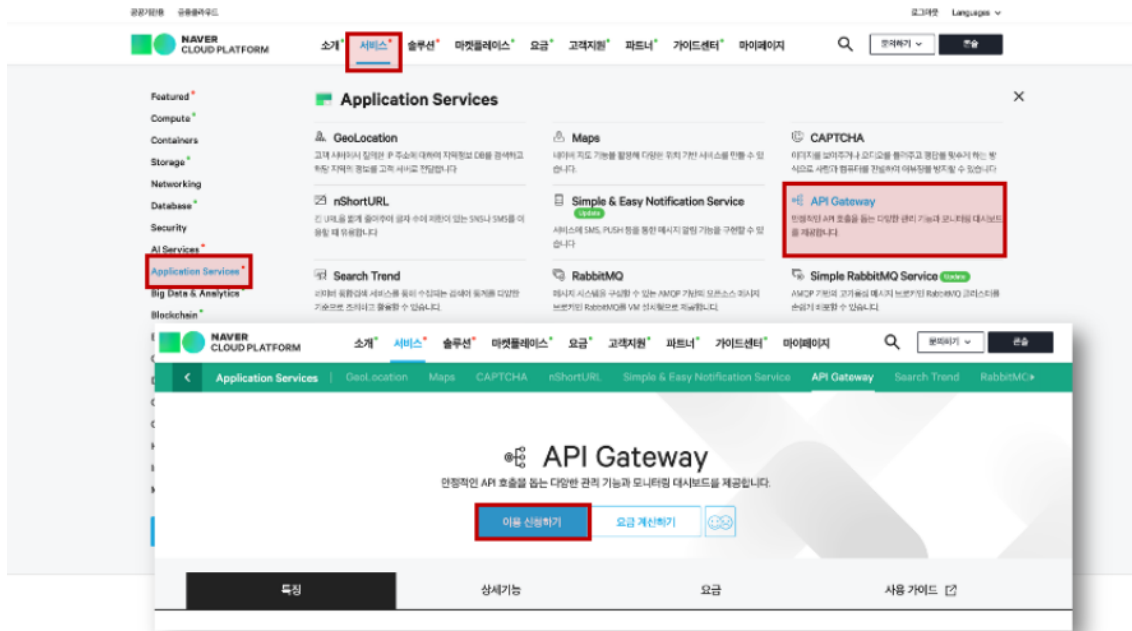
Step 2. 도메인 생성



Naver Cloud Platform 콘솔 왼쪽에 CLOVA OCR 이 보인다면 정상적으로 이용 신청이 된 것입니다.



추가로 Invoke URL 생성을 위해 API Gateway 이용 신청을 합니다.

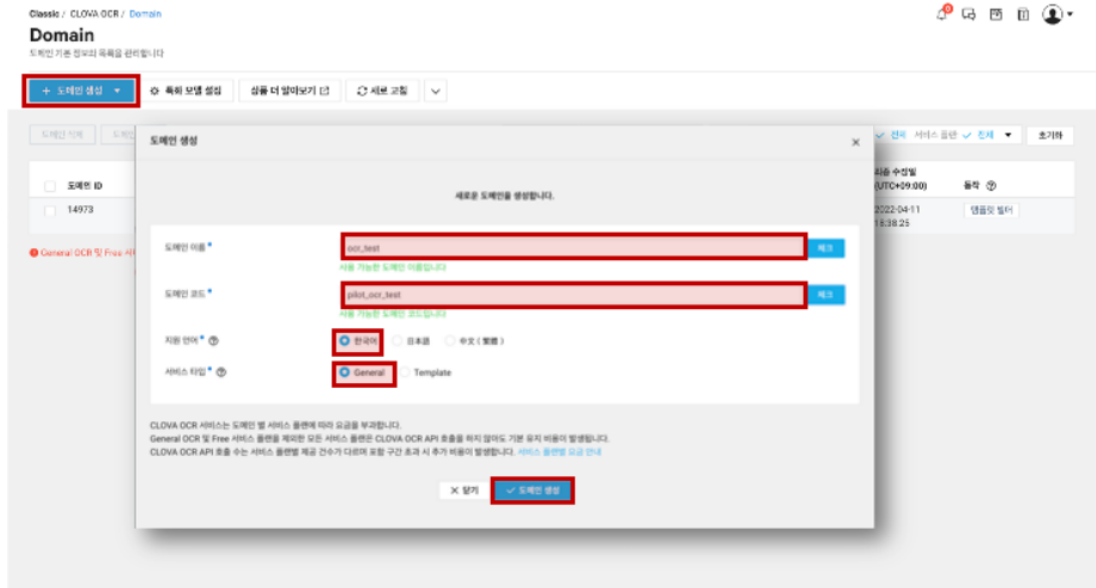


1. [도메인 생성]을 클릭해 주십시오.

- General / Template / Document 생성 메뉴가 나타납니다.
 - General / Template 도메인: 텍스트만 추출하는 Text OCR / 판독 영역 직접 지정을 통해 인식 값 추출 후 테스트 및 결과 전송이 가능한 템플릿 빌더 지원
 - Document 도메인: 머신러닝 기반으로 문서의 의미적 구조를 이해하는 특화 모델 엔진을 탑재하여 입력 정보(key-value) 자동 추출

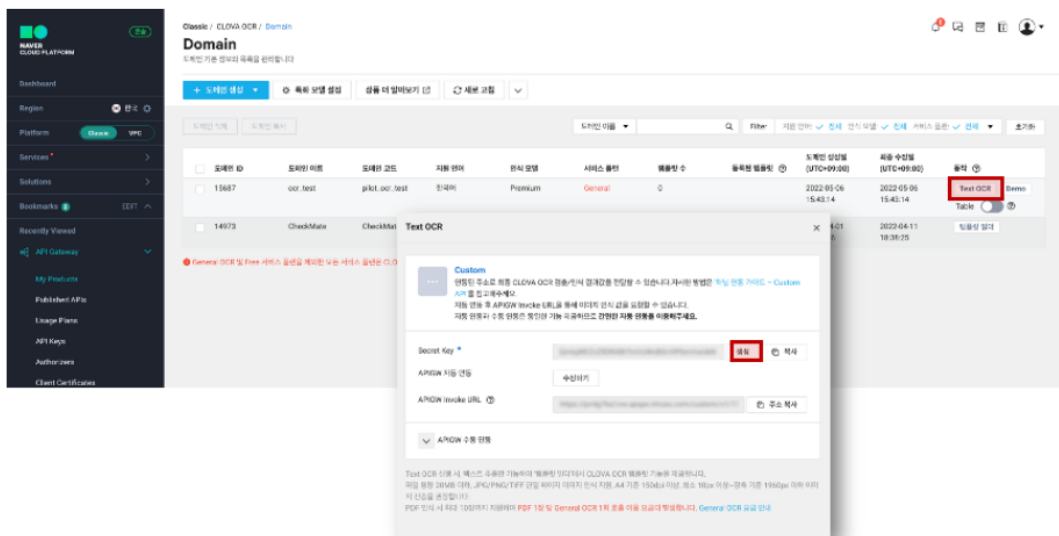
2. General / Template 생성: 도메인 이름과 도메인 코드를 입력해 주십시오.

- Document OCR은 도메인 생성 가이드 하단의 '특화 모델 추가'에서 자세하게 안내합니다.
- 도메인 이름/도메인 코드: 1글자 이상~최대 50글자까지 입력 가능하고 고유한 값으로 중복되지 않아야 합니다.
- OCR 빌더 내 '입력 필드' 입력 가능 문자: 한글, 영문(반각대문자 / 반각 소문자), 일본어(전각 가타카나 / 히라가나), 한문, 일부 특수문자(? & . _ - × ㄱ).
- 단, 도메인 코드 경우 영문 대/소문자(반각대문자 / 반각 소문자)와 일부 특수문자(. ? & . _ - × ㄱ)만 입력할 수 있습니다.
- 지원 언어: 한국어, 일본어, 대만어(중문 번체) 중 하나의 언어를 선택해 주십시오. 영어는 기본 지원되므로 영어가 혼용된 문서의 경우 선택한 지원 언어와 영어 문자를 모두 인식할 수 있습니다.
- 서비스 타입: 텍스트 추출만 가능한 General 타입과 템플릿 빌더를 제공하는 Template 타입을 지원합니다. General 타입은 Region당 하나의 General 도메인(Text OCR 실행) 만 생성할 수 있습니다. Template 타입 선택하면 인식 모델과 서비스 플랜 항목이 노출됩니다.



step 3. Secret Key 생성 및 API Gateway 연결

생성 된 도메인 우측 동작 메뉴에서 [Text OCR] 버튼을 클릭하여 Custom API Gateway 설정을 합니다. [생성] 버튼을 클릭하여 Secret Key를 발급합니다. OCR Invoke URL은 외부 연동 Endpoint에 입력할 OCR API 주소를 의미합니다. 위에서 언급했듯이 API Gateway 이용 신청이 되어 있는 경우, 자동 연동 (Interlock) 버튼을 클릭해 손쉽게 자동 연동할 수 있습니다.



step 4. 프로젝트 연동

```

axios
    .post(
        "{CLOVA에서 받은 URL}",
        {
            images: [
                {
                    format: "png",
                    name: "medium",
                    data: {OCR 적용할 파일},
                },
            ],
        },
    )

```

```

    },
    requestId: "string",
    timestamp: 0,
    version: "V2",
  },
  {
    headers: {
      "X-OCR-SECRET": "{CLOVA 에서 받은 KEY}",
    },
    withCredentials: true,
    proxy: { host: "{CLOVA에서 받은 URL}" },
  }
)

```

CLOVA에서 외부 URL에서 오는 요청을 거부

proxy 통해 URL을 변경 후 전송을 하면 요청 승인 됩니다

아래의 사진 파일 요청 시



respon 결과

```

{
  "version": "V2",
  "requestId": "dcda1db5-7a19-41bb-8a77-dbb2e95a5050",
  "timestamp": 1617947318461,
  "images": [
    {
      "uid": "33894645ee8e4b8094fe309e6efd30a2",
      "name": "demo",
      "inferResult": "SUCCESS",
      "message": "SUCCESS",
      "validationResult": {
        "result": "NO_REQUESTED"
      },
      "fields": [
        {
          "valueType": "ALL",
          "boundingPoly": {
            "vertices": [
              {
                "x": 14.0,
                "y": 20.0
              },
              {
                "x": 273.0,
                "y": 20.0
              },
              {
                "x": 273.0,
                "y": 95.0
              },
              {
                "x": 14.0,
                "y": 95.0
              }
            ]
          },
          "inferText": "CLOVA",
          "inferConfidence": 1.0,
          "type": "NORMAL",
          "lineBreak": false
        },
        {
          "valueType": "ALL",
          "boundingPoly": {
            "vertices": [

```

```

        "x": 286.0,
        "y": 20.0
    },
    {
        "x": 454.0,
        "y": 20.0
    },
    {
        "x": 454.0,
        "y": 93.0
    },
    {
        "x": 286.0,
        "y": 93.0
    }
]
},
"inferText": "OCR",
"inferConfidence": 1.0,
"type": "NORMAL",
"lineBreak": true
},
{
    "valueType": "ALL",
    "boundingPoly": {
        "vertices": [
            {
                "x": 55.0,
                "y": 213.0
            },
            {
                "x": 107.0,
                "y": 213.0
            },
            {
                "x": 107.0,
                "y": 245.0
            },
            {
                "x": 55.0,
                "y": 245.0
            }
        ]
    },
    "inferText": "참고",
    "inferConfidence": 1.0,
    "type": "NORMAL",
    "lineBreak": true
},
{
    "valueType": "ALL",
    "boundingPoly": {
        "vertices": [
            {
                "x": 59.0,
                "y": 357.0
            },
            {
                "x": 146.0,
                "y": 357.0
            },
            {
                "x": 146.0,
                "y": 395.0
            },
            {
                "x": 59.0,
                "y": 395.0
            }
        ]
    },
    "inferText": "네이버",
    "inferConfidence": 1.0,
    "type": "NORMAL",
    "lineBreak": false
},
{
    "valueType": "ALL",
    "boundingPoly": {
        "vertices": [
            {
                "x": 150.0,
                "y": 359.0
            },
            {
                "x": 267.0,
                "y": 359.0
            }
        ]
    }
}

```

```

        },
        {
            "x": 267.0,
            "y": 395.0
        },
        {
            "x": 150.0,
            "y": 395.0
        }
    ]
},
{
    "inferText": "클라우드",
    "inferConfidence": 1.0,
    "type": "NORMAL",
    "lineBreak": false
},
{
    "valueType": "ALL",
    "boundingPoly": {
        "vertices": [
            {
                "x": 269.0,
                "y": 357.0
            },
            {
                "x": 385.0,
                "y": 357.0
            },
            {
                "x": 385.0,
                "y": 395.0
            },
            {
                "x": 269.0,
                "y": 395.0
            }
        ]
    },
    "inferText": "플랫폼의",
    "inferConfidence": 0.9999,
    "type": "NORMAL",
    "lineBreak": false
},
...
]
}
}

```

step 5. 프록시 설정

- react
 - package.json

```

...
},
"proxy": "https://282c769uda.apigw.ntruss.com/custom/v1/"
}
...

```

파일 하단에 proxy 관련 설정을 추가

- node_modules/react-scripts/config/webPackDevServer.config.js

해당 설정은 로컬 환경에서 진행할 시 필수 과정은 아닙니다.

```

...
module.exports = function (proxy, allowedHost) {
    const disableFirewall = true;

    return {
        ...
    }
}

```


disableFirewall 값을 true로 변경

도커 혹은 기타 외부 저장소 업로드 시 `git add -f` 명령어를 이용하여 .gitignore 무시하고 git 저장소에 업로드