목차

Ⅰ. 빌드 및 배포

1. 개발 환경

- Server: AWS EC2 Ubuntu 20.04 LTS
- JAVA JDK: 11.0.15.1
- MySQL: 8.0.29
- Redis: 3.0.504
- SpringBoot: 2.7.1
- Node.js: 16.16.0 LTS
- IntelliJ Ultimate: 2022.1.3
- VSCode: Stable Build
- Hibernate: 5.6.9 Final
- Vue.js: 2.7.7
- Vuetify: 2.6.7
- OpenVidu: 2.22.0
- Nginx: 1.18.0
- Docker: 20.10.17
- Teachable Machine

2. 설정 파일 목록

<SpringBoot>

- application.properties

- application-oauth.properties

<Nginx>

/etc/nginx/sites-available/default

3. 설정 파일 상세

<SpringBoot>

- application.properties

Port

server.port=8082

Redis

spring.redis.host=localhost

spring.redis.port=6379

Swagger

spring.mvc.pathmatch.matching-strategy=ant_path_matcher

MySQL

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

DB url

spring.datasource.url=jdbc:mysql://i7a802.p.ssafy.io:3306/jpa?characterEncoding=UTF-8&serverTimezone=UTC&

#spring.datasource.url=jdbc:mysql://localhost:3306/jpa?characterEncoding=UTF-8&serverTimezone=UTC&

DB username

spring.datasource.username=계정명

```
# DB password
spring.datasource.password=비밀번호

# JPA options
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=none
spring.jpa.properties.hibernate.format_sql=true
spring.jpa.hibernate.database=mysql
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5InnoDBDialect
spring.jpa.properties.hibernate.default_batch_fetch_size = 1000

# multipart - file upload
#spring.servlet.multipart.location=C:₩₩Temp₩₩upload
spring.servlet.multipart.max-file-size=10MB
spring.servlet.multipart.max-request-size=50MB
file.upload.dir.window=C:₩₩Temp₩₩upload
file.upload.dir.linux=/home/ubuntu/upload

#mail
spring.mail.host=smtp.naver.com
spring.mail.port=465
spring.mail.username=kimsm9822
spring.mail.password=FSS9R491S7UF
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.ssl.enable=true
spring.mail.properties.mail.smtp.ssl.trust=smtp.naver.com
```

#oauth

spring.profiles.include=oauth

- application-oauth.properties

#Oauth

spring.security.oauth2.client.registration.google.client-id=531257147697-gedu73e4t7ugg9dt0vhd5k9ddla98akg.apps.googleusercontent.com

spring.security.oauth2.client.registration.google.client-secret=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

spring.security.oauth2.client.registration.google.scope=profile,email

#spring.security.oauth2.client.registration.google.redirect-uri=http://localhost:8082/api/auth/login/google/callback

#spring.security.oauth2.client.registration.google.redirect-uri=http://localhost:8080/auth/google/redirect

spring.security.oauth2.client.registration.google.redirect-uri=https://i7a802.p.ssafy.io/auth/google/redirect

spring.security.oauth2.client.registration.naver.client-id=YTUxblXLw0BuEFXiYUFV

spring.security.oauth2.client.registration.naver.client-secret=DlnbN38WFt

#spring.security.oauth2.client.registration.naver.redirect-uri=http://localhost:8080/auth/naver/redirect

spring.security.oauth2.client.registration.naver.redirect-uri=https://i7a802.p.ssafy.io/auth/naver/redirect

spring.security.oauth2.client.registration.naver.authorization_grant_type=authorization_code

spring.security.oauth2.client.registration.naver.scope=name,email,profile_image

spring.security.oauth2.client.registration.naver.client-name=Naver

spring.security.oauth2.client.provider.naver.authorization_uri=https://nid.naver.com/oauth2.0/authorize

spring.security.oauth2.client.provider.naver.token_uri=https://nid.naver.com/oauth2.0/token

spring.security.oauth2.client.provider.naver.user-info-uri=https://openapi.naver.com/v1/nid/me

spring.security.oauth2.client.provider.naver.user_name_attribute=response

<Nginx>

```
# Default server configuration
server {
        server_name i7a802.p.ssafy.io;
        location / {
                root /home/ubuntu/dist;
                try_files $uri $uri/ /index.html =404;
        }
        location /api {
                proxy_pass http://localhost:8082/api;
        }
        location /gan {
                proxy_pass http://localhost:8083/gan;
        }

    listen [::]:443 ssl ipv6only=on; # managed by Certbot
    listen 443 ssl; # managed by Certbot
```

```
        ssl_certificate /etc/letsencrypt/live/i7a802.p.ssafy.io/fullchain.pem; # managed by Certbot
        ssl_certificate_key /etc/letsencrypt/live/i7a802.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 {
        if ($host = i7a802.p.ssafy.io) {
            return 301 https://$host$request_uri;
        } # managed by Certbot


            listen 80 default_server;
            listen [::]:80 default_server;


            server_name i7a802.p.ssafy.io;
        return 404; # managed by Certbot
}
```

4. 우분투 패키지 설치

본 프로젝트는 수동배포로 진행하여 과정이 다소 복잡합니다.

<JDK>

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

설치 후

```
$ java -version
```

<nginx>

```
$ sudo apt install nginx
```

설치 후 3 번의 설정 파일처럼 서버 이름 설정 후 인증서 발급 필요

<certbot>

repository 에 certbot 설치

$ sudo add-apt-repository ppa:certbot/certbot

python-certbot-nginx 설치

$ sudo apt-get install python-certbot-nginx

SSL 인증서 생성하고 적용하기

$ sudo certbot --nginx

<docker>

업데이트 및 HTTP 패키지 설치 Permalink

$ sudo apt update

$ sudo apt-get install -y ca-certificates ₩

curl ₩

software-properties-common ₩

apt-transport-https ₩

gnupg ₩

lsb-release

GPG 키 및 저장소 추가 Permalink

$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

<redis>

#설치

$ sudo docker pull redis:latest

#도커에 네트워크 생성

$ sudo docker network create redis-net

볼륨 잡아서 컨테이너 실행

$ sudo docker run --name redis -p 6379:6379 --network redis-net -v /home/ubuntu/redisvolume -d redis:latest redis-server --appendonly yes

<mysql>

$ sudo docker pull mysql:8.0.29

$ sudo docker run -d -p 3306:3306 -v /home/ubuntu/mysql_data:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=루트비밀번호 --name mysql-container mysql:8.0.29

mysql 컨테이너 접속

$ sudo docker exec -it mysql-container bash

$ mysql -u root –p

$ 패스워드 입력

접속 후 계정 생성

$ mysql> CREATE USER 계정명@'%' identified by '비밀번호';

$ mysql> GRANT ALL PRIVILEGES ON *.* to 계정명@'%';

$ mysql> FLUSH PRIVILEGES;

$ mysql> exit;

이후 workbench 로 접속하여 'jpa' 스키마 생성, dump.sql 실행

<GAN 가상 시창 기능용 서버를 위한 패키지 설치>

이 기능을 사용하려면 설치해야 하지만 실험적인 기능이고 설치가 다소 복잡하여 필수사항은 아닙니다.

<아나콘다 설치>

$ wget https://repo.anaconda.com/archive/Anaconda3-2019.10-Linux-x86_64.sh

배쉬 진입 후 설치 진행

$ bash Anaconda3-2019.10-Linux-x86_64.sh

# 배쉬로 진입
$ vi ~/.bashrc

진입 후 아래 내용 추가

$ export PATH="/home/username/anaconda3/bin:$PATH"

이후 .bashrc 실행

$ source ~/.bashrc

<fastapi>

$ pip install uvicorn

$ pip install fastapi

<pytorch>

$ conda install pytorch torchvision torchaudio cpuonly -c pytorch

$ pip install scikit-image

$ pip install dominate

$ pip install pandas

$ pip install visdom

$ pip install --upgrade pip setuptools wheel

$ pip install opencv-python

python 서버 가동 준비 완료

5. FE / BE 빌드 및 배포

<FE>

npm install --legacy-peer-deps

nom run build

생성된 dist 폴더는 /home/ubuntu/ 아래에 위치

<BE-java>

gradle bootJar 로 빌드하여 jar 파일 생성

- nohup java –jar -Duser.timezone=Asia/Seoul 파일이름.jar &

백그라운드로 서버 가동

<BE-python>

Gitlab /backend-python/ 아래에 있는 /pose-transfer 폴더를 /home/ubuntu/ 에 배포

pose-transfer 디렉토리에 있는 main.py 백그라운드 실행

$ nohup python main.py &

6. OpenVidu 배포

- 기존 OpenVidu 가 있다면 삭제

$ sudo docker ps –a

#openvidu, kurento media server 등의 컨테이너가 존재한다면 삭제한다.

$ sudo docker rm #컨테이너 모두 삭제를 원할 경우

$ sudo docker rm $(docker ps -a) # 이미지도 삭제

$ sudo docker rmi # 이미지 전체 삭제를 원할 경우

$ sudo docker rmi $(docker images)

- OpenVidu 설치

$ sudo su # 관리자 권한

$ cd /opt # openvidu 가 설치되는 경로

openvidu on promises 설치

$ curl https://s3-eu-west1.amazonaws.com/aws.openvidu.io/install_openvidu_latest.sh | bash

$ exit

- OpenVidu 설정 및 SSL 인증서 적용

$ cd /opt/openvidu

$ vi .env

DOMAIN_OR_PUBLIC_IP=
OPENVIDU_SECRET=MY_SECRET
CERTIFICATE_TYPE=letsencrypt
LETSENCRYPT_EMAIL=인증서 발급 시 입력한 이메일
HTTP_PORT=8442 HTTPS_PORT=8443 ...
# ESC 입력 후 :wq! 로 저장 후 나가기

- OpenVidu 포트 개방

$ sudo apt update

$ sudo apt install netfilter-persistent

$ sudo apt install iptables-persistent

$ sudo service iptables start

$ sudo iptables -A INPUT -p udp --match multiport --dports 40000:65535 -j ACCEPT

$ sudo iptables -A INPUT -p tcp --match multiport --dports 40000:65535 -j ACCEPT

$ sudo iptables -I INPUT 1 -p tcp --dport 22 -j ACCEPT

$ sudo iptables -I INPUT 1 -p tcp --dport 80 -j ACCEPT

$ sudo iptables -I INPUT 1 -p tcp --dport 443 -j ACCEPT

$ sudo iptables -I INPUT 1 -p tcp --dport 3478 -j ACCEPT

$ sudo iptables -I INPUT 1 -p udp --dport 3478 -j ACCEPT

$ sudo iptables -I INPUT 1 -p tcp --dport 8082 -j ACCEPT

$ sudo iptables -I INPUT 1 -p tcp --dport 8443 -j ACCEPT

$ sudo iptables -A INPUT -p tcp --dport 8080 -j ACCEPT

$ sudo service iptables save

$ sudo service iptables restart sudo

$ sudo netfilter-persistent save

$ sudo netfilter-persistent start

- OpenVidu 서비스(관련 컨테이너) 실행

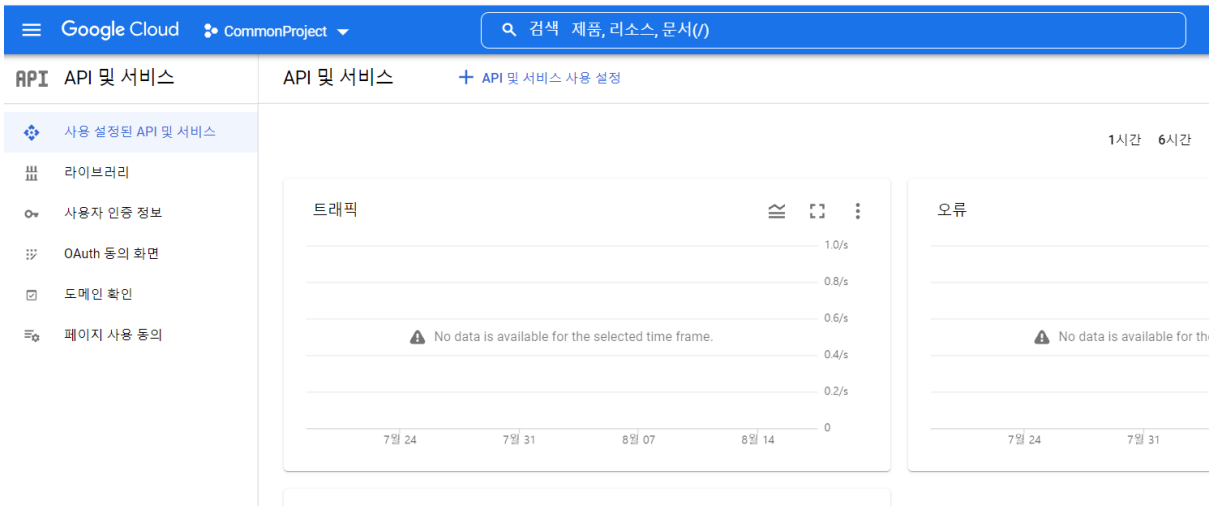
$ sudo ./openvidu start # /opt/openvidu 위치에서

$ sudo ./openvidu stop # 종료할 때는 같은 경로에서

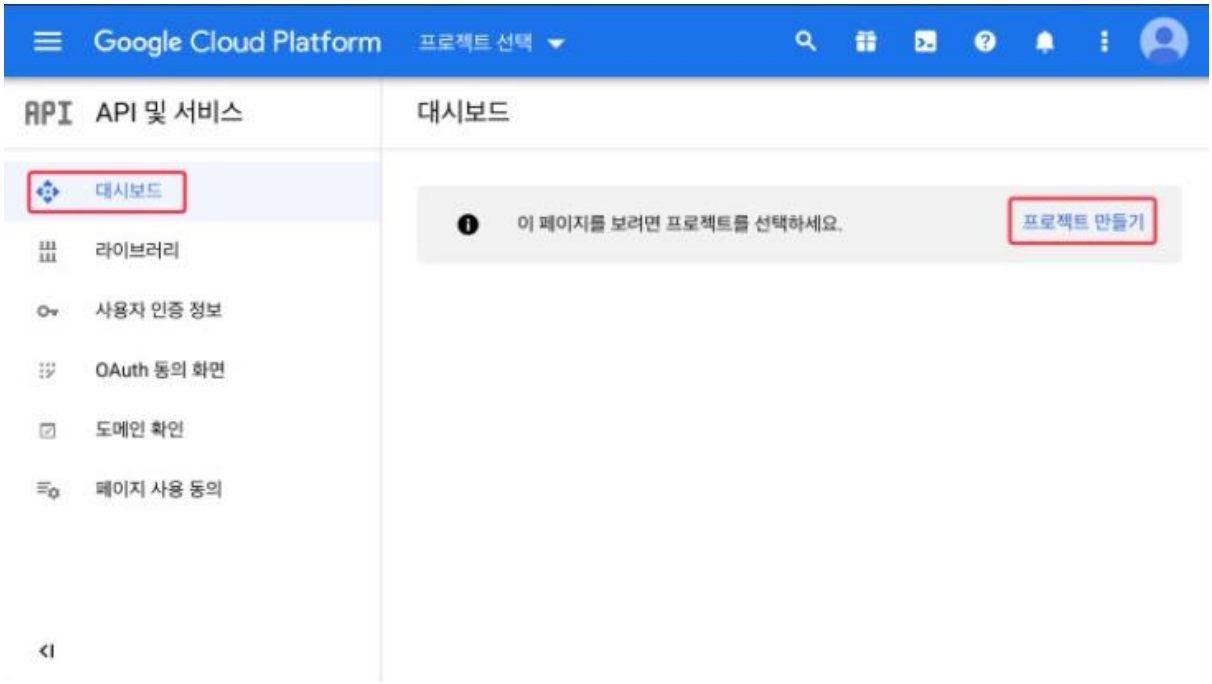
Ⅱ. 외부 서비스

1. 구글 로그인 설정

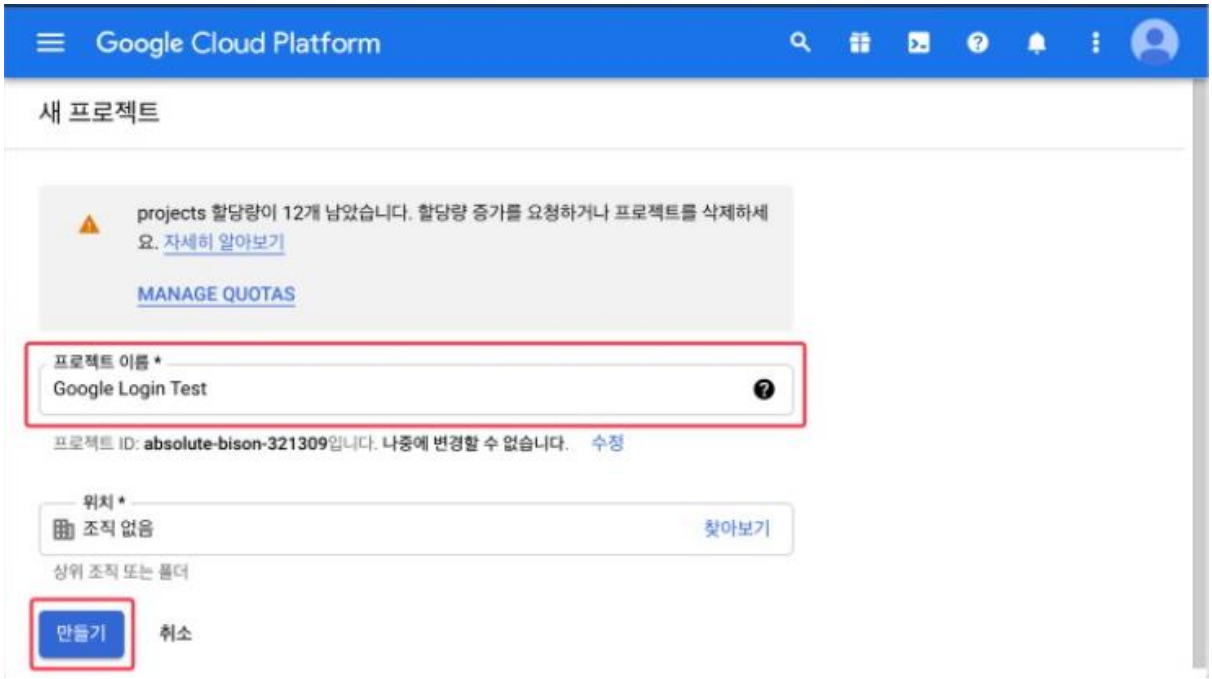
1) https://console.cloud.google.com/apis/dashboard 에 접속한다.



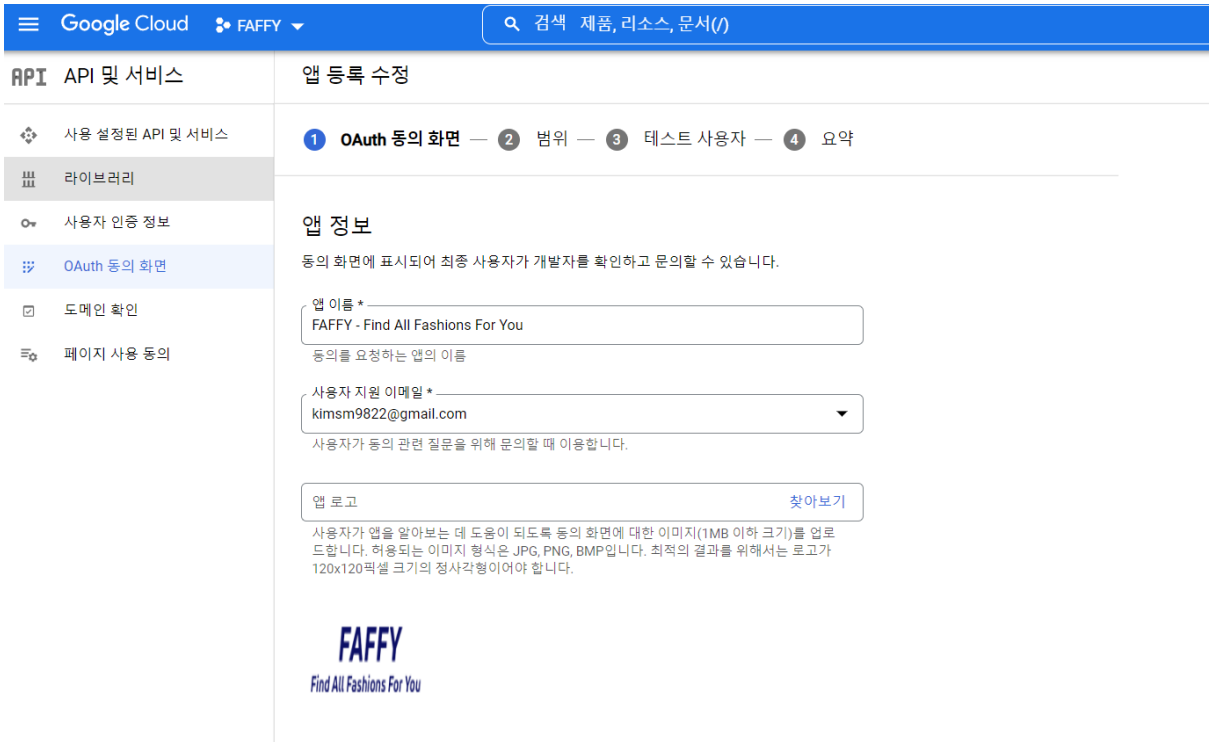
2) 프로젝트 생성하기



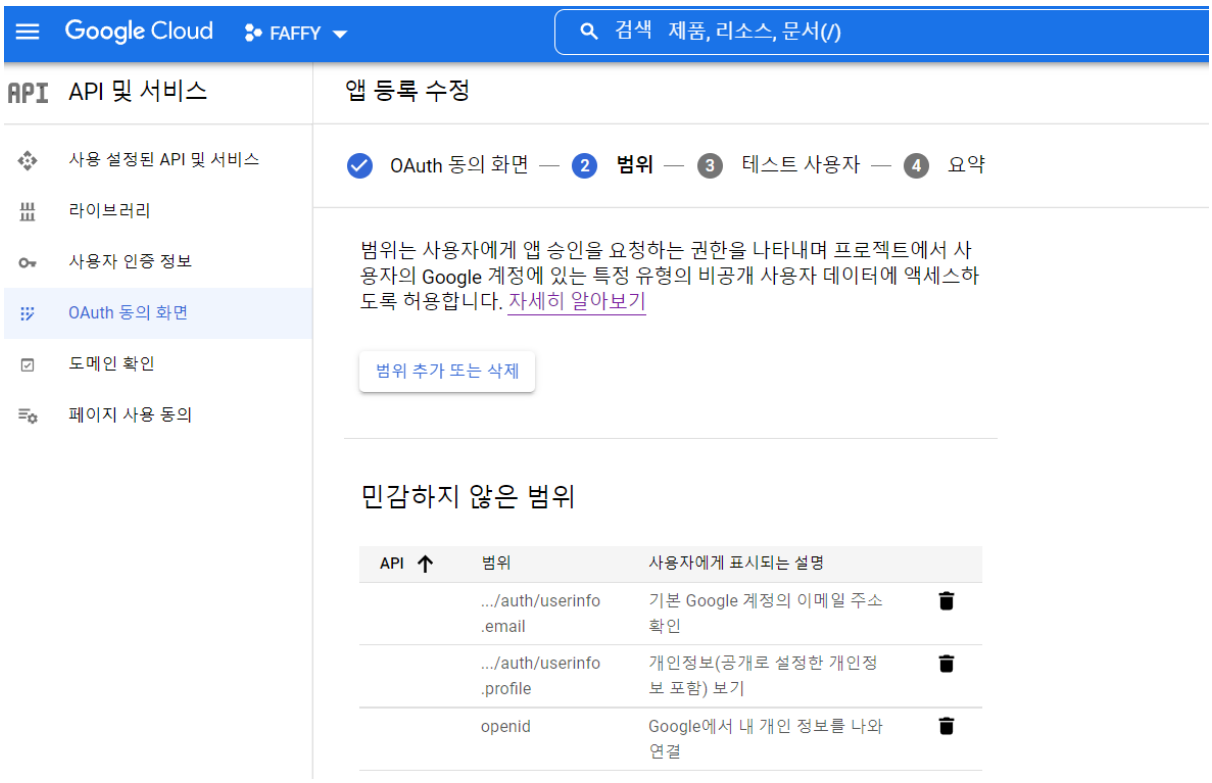
3) 프로젝트 이름 설정

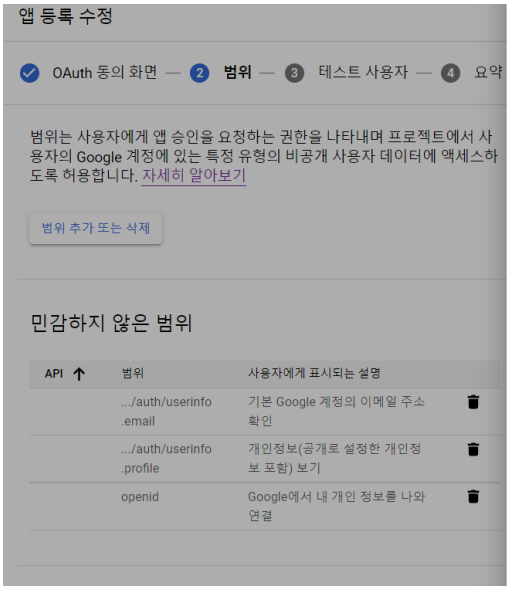


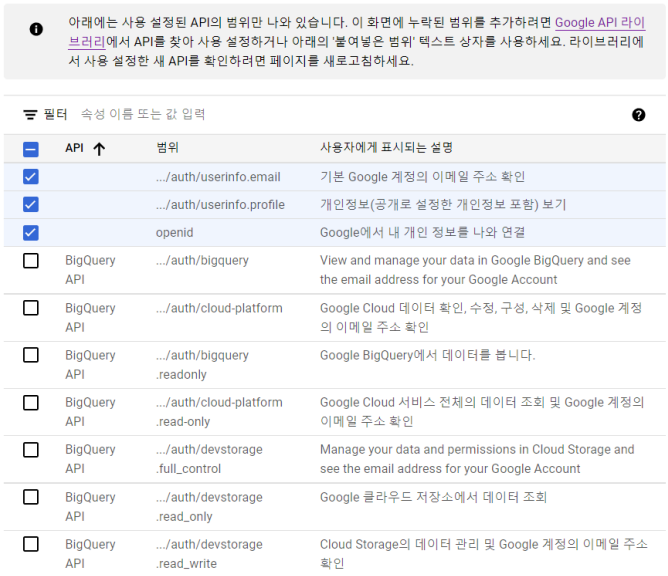
4) OAuth 동의 화면



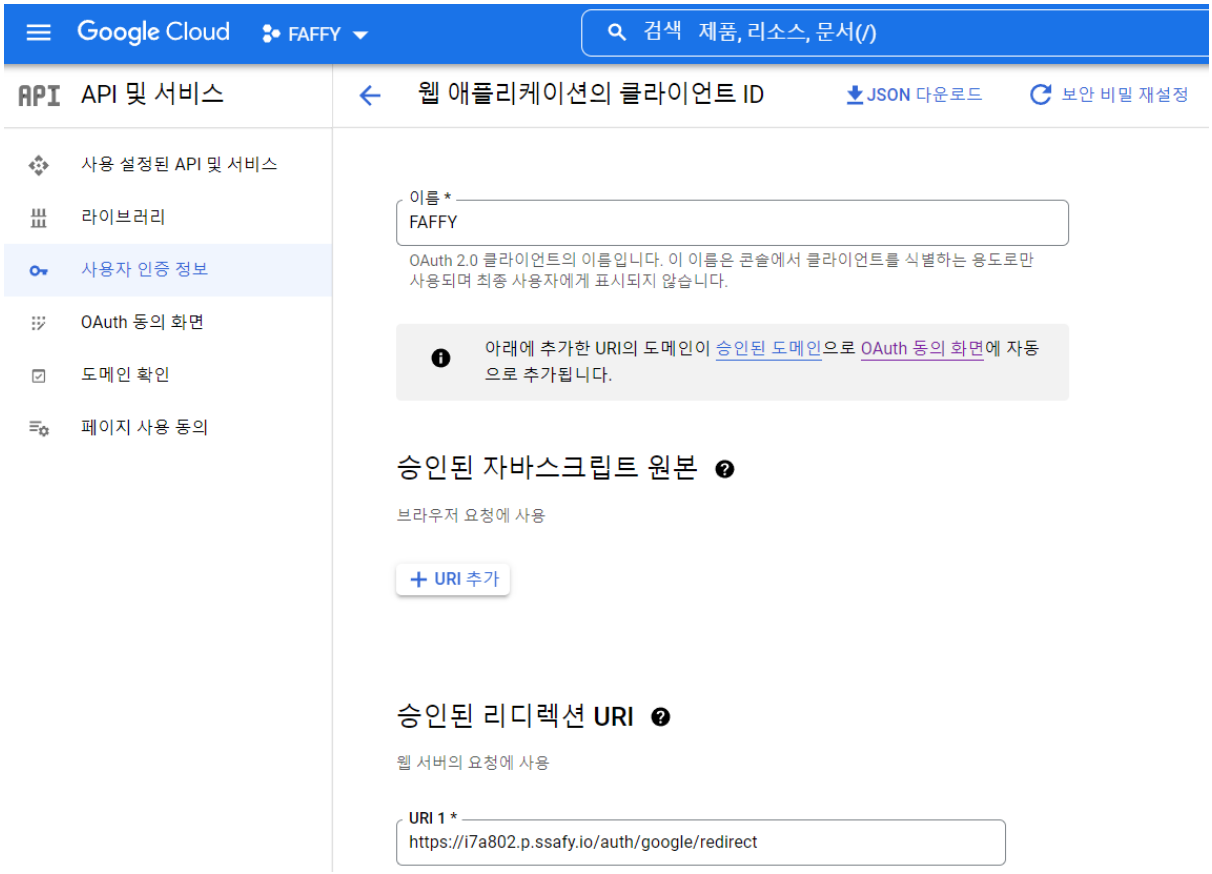
5) 범위 설정



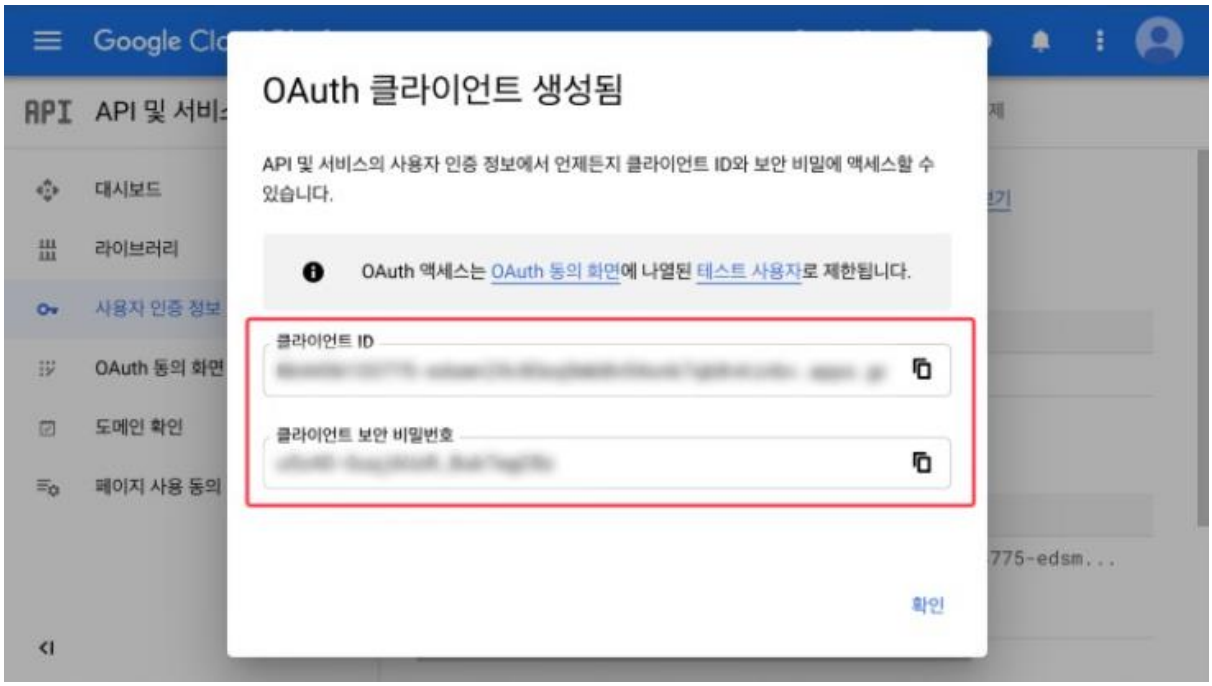




6) 사용자 인증 정보 및 리디렉션 URI

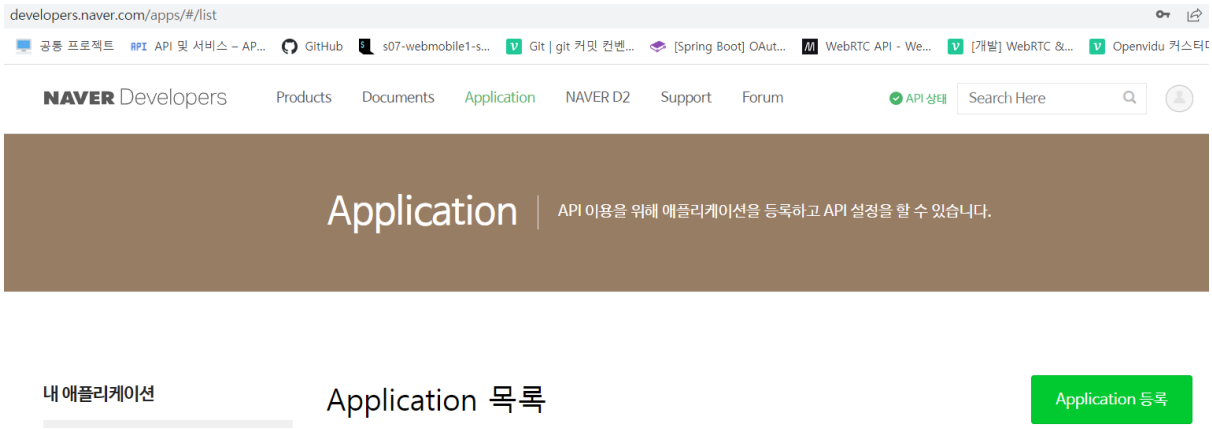


7) 클라이언트 id/secret 확인 및 메모

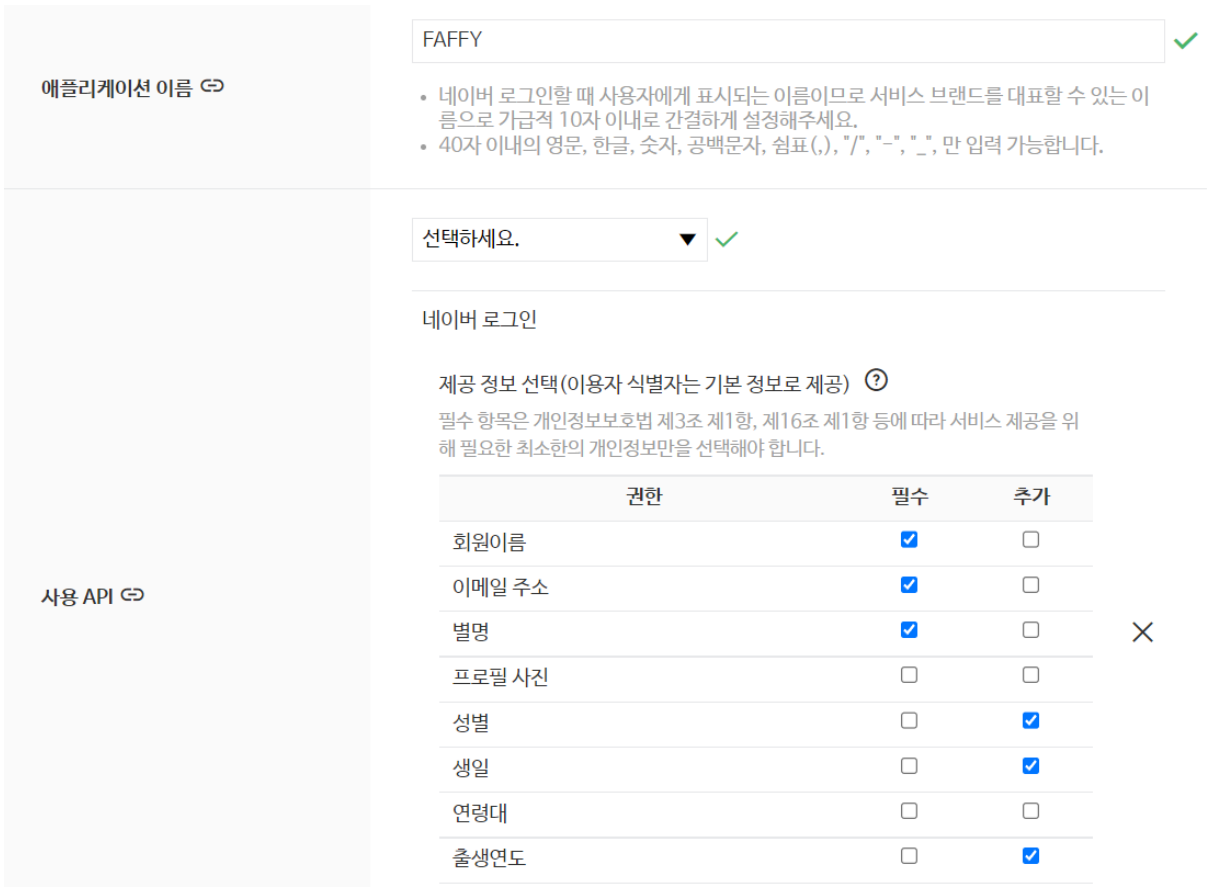


2. 네이버 로그인 설정

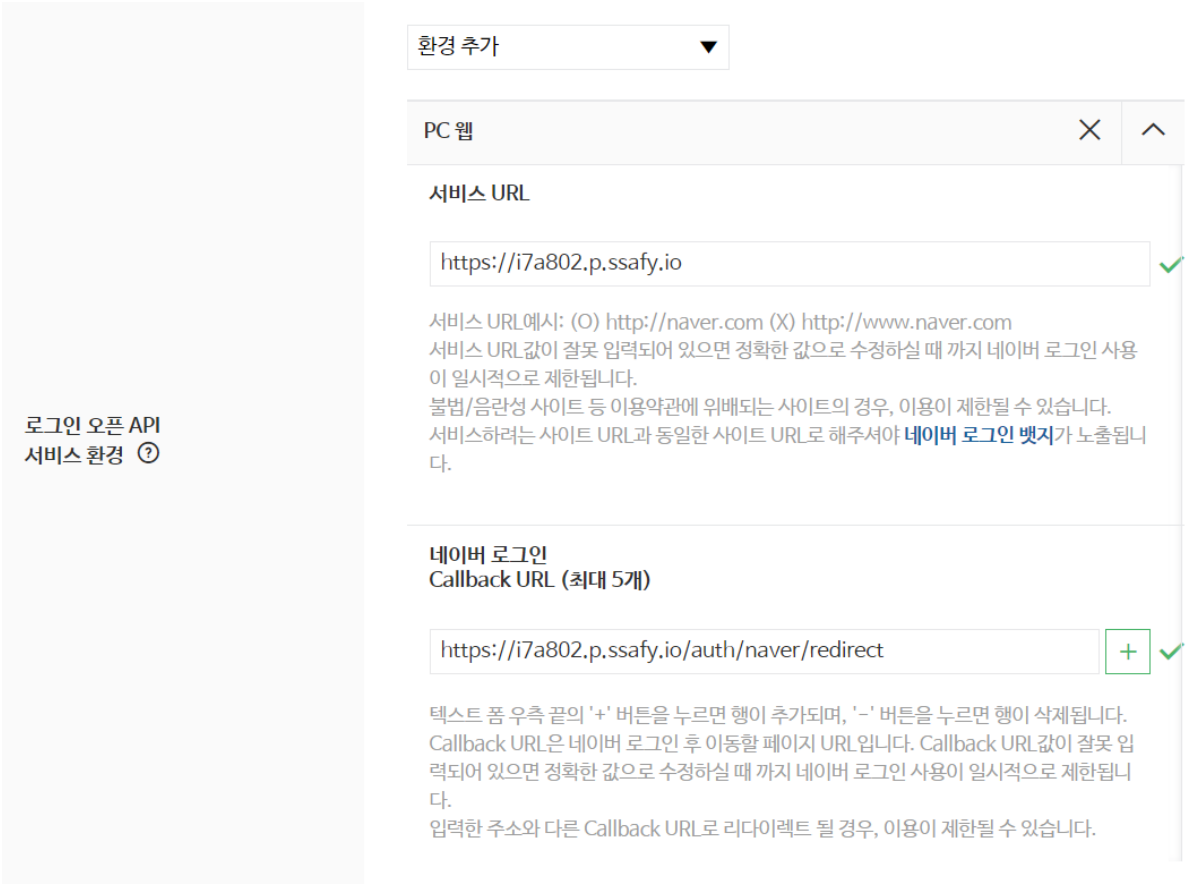
1) naver developers의 Application 탭에서 새로운 앱 등록



2) 어플리케이션의 이름과 네이버 로그인 API를 선택 후, 제공 정보 선택



3) 로그인 오픈 API 환경 설정 – PC 웹으로 추가하고, 서비스 URL, callback URL 입력



4) 클라이언트 id/secret 확인