

목차

빌드 및 배포

1. 개발 환경
2. 설정 파일 목록
3. 설정 파일 상세
4. MySQL 설치
- 5~7. 배포

외부 서비스

1. 구글 소셜 로그인

빌드 및 배포

1. 개발 환경

- Server: AWS EC2 Ubuntu 20.04 LTS
- Python: 3.9.13
- MySQL: 8.0.30
- Storage: AWS S3
- FastAPI: 0.82.0
- Django: 3.2 LTS
- Pytorch: 1.12.1
- Anaconda: 3.9
- node.js: 16.17.0 (LTS)
- React-Native: 0.69
- Android Studio: Chipmunk 2021.2.1 Patch 2
- React-query v4.2.3
- VSCode: Stable Build
- Nginx: 1.18.0

2. 설정 파일 목록

Django

- my_settings.py

FastAPI

- secrets.json

Nginx

- `/etc/nginx/sites-available/django`

3. 설정 파일 상세

Django - my_settings.py

```
DATABASES = {  
    'default': {  
        'ENGINE': 'django.db.backends.mysql',  
        'NAME': 'epari',  
        'USER': '계정 이름',  
        'PASSWORD': '계정 비밀번호',
```

```

    'HOST': 'j7a201.p.ssafy.io',
    'PORT': '3306'
}
}

SECRET_KEY = 'django-insecure-)=^21_2y&$hxvwg%v55rfp(qq@vm=-iq2jkg)+cn63x*o4sm'
TOUR_API_KEY =
'MowXeM423Bmo5hAcSq5Bp5gyrI0gZg5mDPqV5p5iK1RN6ZzHIFByEonurgCm3Fv5j6vSbgtw4Yncr
nffClugOg%3D%3D'

AWS_S3_ACCESS_KEY_ID = S3 KEY ID
AWS_S3_SECRET_ACCESS_KEY = S3 접속 KEY
AWS_STORAGE_BUCKET_NAME = 'dolarge'

```

FastAPI - secrets.json

```

{
  "DB": {
    "user": "계정 이름",
    "password": "계정 비밀번호",
    "host": "j7a201.p.ssafy.io",
    "port": 3306,
    "database": "plant"
  }
}

```

구글 로그인

```

{
  "type": "service_account",
  "project_id": "PROJECT ID",
  "private_key_id": "PRIVATE KEY ID",
  "private_key": "PRIVATE KEY",
  "client_email": "CLIENT EMAIL",
  "client_id": "CLIENT ID",
  "auth_uri": "https://accounts.google.com/o/oauth2/auth",
  "token_uri": "https://oauth2.googleapis.com/token",
  "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
  "client_x509_cert_url": "CLIENT CERT URL"
}

```

Nginx

```

server {
    listen 80;
    server_name j7a201.p.ssafy.io;

    location /epari {
        include proxy_params;
        proxy_pass http://localhost:8000/epari;
    }

    location /ai {
        include proxy_params;
        proxy_pass http://localhost:8001/ai;
    }
}

```

4. MySQL 설치

```

sudo docker pull mysql:8.0.30
sudo docker run -d -p 3306:3306 -v /home/ubuntu/mysql_data:/var/lib/mysql -e
MYSQL_ROOT_PASSWORD=비밀번호 --name mysql-container mysql:8.0.30

sudo docker exec -it mysql-container bash
mysql -u root -p

mysql> CREATE USER 계정명@'%' identified by '비밀번호';
mysql> GRANT ALL PRIVILEGES ON *.* to 계정명@'%;
mysql> FLUSH PRIVILEGES;
mysql> exit

#이후 mysql workbench에서 계정 생성 후 'epari' 스키마 생성, epari.sql 실행

```

5. 배포 - django

python 설치

```

git clone https://github.com/pyenv/pyenv.git ~/.pyenv
sed -Ei -e '/^([^\#]|$)/ {a \
export PYENV_ROOT="$HOME/.pyenv"
a \
export PATH="$PYENV_ROOT/bin:$PATH"
a \
' -e ':a' -e '$!{n;ba};}' ~/.profile
echo 'eval "$(pyenv init --path)"' >> ~/.profile

```

```
echo 'eval "$(pyenv init -)"' >> ~/.bashrc

source ~/.profile
source ~/.bashrc

pyenv install 3.9.13
pyenv global 3.9.13
python -V
```

project 클론

```
# /home/ubuntu/ 내에 설치
cd ubuntu
git clone ~
```

폴더구조

```
home/
  ubuntu/
    S07P22A201
      backend
        django
          epari_backend
            settings.py
            ...
            accounts
            plantbook
            ...
            manage.py
            my_settings.py
            requirements.txt
          fastapi
        frontend
```

가상환경 활성화 및 라이브러리 설치

```
#django 폴더로 이동, 가상환경 생성 및 활성화
python -m venv venv
source venv/bin/activate

#라이브러리 설치
pip install -r requirements.txt
```

데이터 불러오기

```
python manage.py migrate
python manage.py loaddata location.json title.json plant.json
```

gunicorn 설치

```
pip install gunicorn
```

service 파일 수정

```
sudo vi /etc/systemd/system/gunicorn.service

#아래 내용 작성
[Unit]
Description=gunicorn daemon
After=network.target

[Service]
User=ubuntu
Group=www-data
WorkingDirectory=/home/ubuntu/S07P22A201/backend/django
ExecStart=/home/ubuntu/S07P22A201/backend/django/venv/bin/gunicorn \
    --workers 3 \
    --bind 127.0.0.1:8000 \
    epari_backend.wsgi:application

[Install]
WantedBy=multi-user.target
#여기까지
```

서비스 실행 및 등록

```
sudo systemctl daemon-reload

sudo systemctl start gunicorn
sudo systemctl enable gunicorn
```

nginx 설치

```
sudo apt-get update
sudo apt-get install -y nginx
```

파일 작성

```
sudo vi /etc/nginx/sites-available/django

#아래 내용 작성 후 저장
server {
    listen 80;
    server_name j7a201.p.ssafy.io;

    location /epari {
        include proxy_params;
        proxy_pass http://localhost:8000/epari;
    }

    location /ai {
        include proxy_params;
        proxy_pass http://localhost:8001/ai;
    }
}

# 위까지 작성 후 아래 명령어 실행
sudo ln -s /etc/nginx/sites-available/django /etc/nginx/sites-enabled
```

80번 포트 종료 후 nginx 재실행

```
sudo lsof -t -i tcp:80 -s tcp:listen | sudo xargs kill
sudo systemctl restart nginx
systemctl status nginx.service
```

백그라운드에서 실행

```
nohup gunicorn --bind 0.0.0.0:8000 epari_backend.wsgi:application &
```

6. 배포 - fastapi

anaconda 3 설치

```
wget https://repo.anaconda.com/archive/Anaconda3-2019.10-Linux-x86_64.sh
bash Anaconda3-2019.10-Linux-x86_64.sh
vi ~/.bashrc
```

conda 가상환경 생성 및 활성화

```
#ubuntu/s07p22a201/backend/fastapi로 이동 후  
conda create -n venv python=3.9.13  
conda activate venv
```

fastapi 설치

```
pip install -r requirementst.txt  
conda install pytorch torchvision torchaudio cpuonly -c pytorch
```

데이터 불러오기

```
python load.py
```

백그라운드에서 실행

```
nohup python main.py &
```

7. 배포 - frontend

- 본 프로젝트는 playstore 및 appstore에는 배포하지 않았으므로, 실제 배포할 경우의 간단한 절차에 대해서만 적습니다.

구글 개발자 등록

<https://play.google.com/apps/publish/signup/>

비용은 \$25이며 1회 등록으로 평생 이용이 가능하다.

서명키 생성

1. Windows 환경의 경우 해당 명령어를 활용해 서명키를 생성한다.

```
keytool -genkeypair -v -storetype PKCS12 -keystore my-upload-key.keystore -alias my-key-alias -keyalg RSA -keysize 2048 -validity 10000
```

생성된 my-upload-key.keystore를 android/app에 위치 시킨다.

1. android/gradle.properties의 부분에 서명키 생성 시 등록한 keystore와 key의 password를 입력한다.

```
MYAPP_UPLOAD_STORE_FILE=my-upload-key.keystore
MYAPP_UPLOAD_KEY_ALIAS=my-key-alias
MYAPP_UPLOAD_STORE_PASSWORD=*****
MYAPP_UPLOAD_KEY_PASSWORD=*****
```

1. 다음과 같이 android/app/build.gradle을 수정한다.

```
...
android {
    ...
    defaultConfig { ... }
    signingConfigs {
        release {
            if (project.hasProperty('MYAPP_UPLOAD_STORE_FILE')) {
                storeFile file(MYAPP_UPLOAD_STORE_FILE)
                storePassword MYAPP_UPLOAD_STORE_PASSWORD
                keyAlias MYAPP_UPLOAD_KEY_ALIAS
                keyPassword MYAPP_UPLOAD_KEY_PASSWORD
            }
        }
    }
    buildTypes {
        release {
            ...
            signingConfig signingConfigs.release
        }
    }
}
...
```

빌드

빌드 형식에 따라 명령어와 파일 위치가 상이하다.

- aab

```
cd android && ./gradlew bundleRelease
```

android/app/build/outputs/bundle/app-release.aab에 위치

- apk

```
cd android && ./gradlew assembleRelease
```

android/app/build/outputs/apk/app-release.apk에 위치

Play Store에 앱 등록

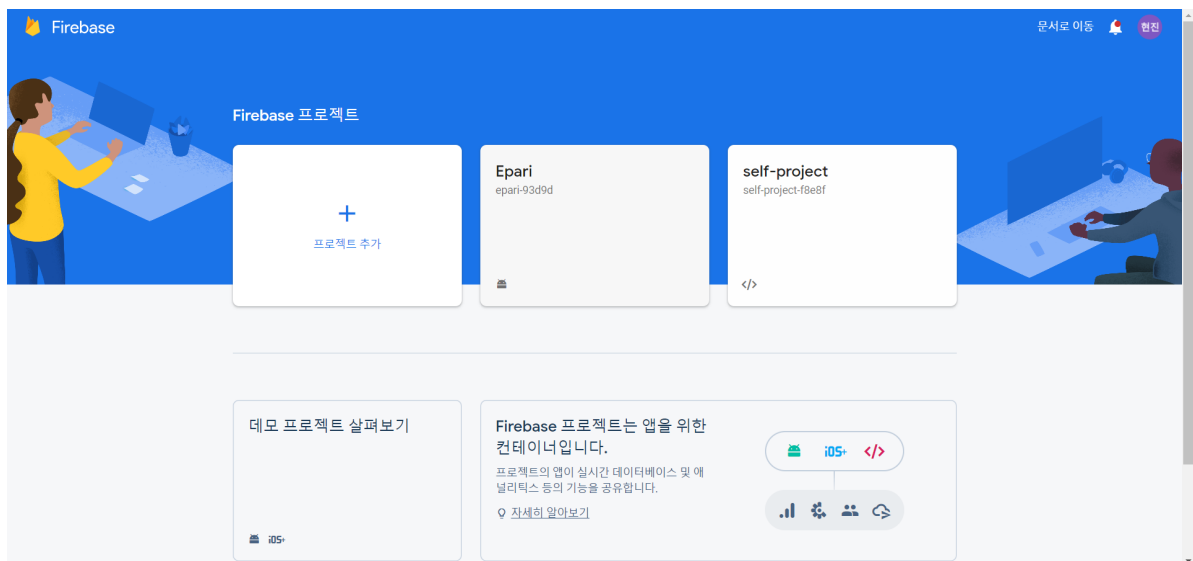
[Google Play Console](https://console.firebase.google.com/)에서 등록 작업을 진행한다.

외부 서비스 - Firebase

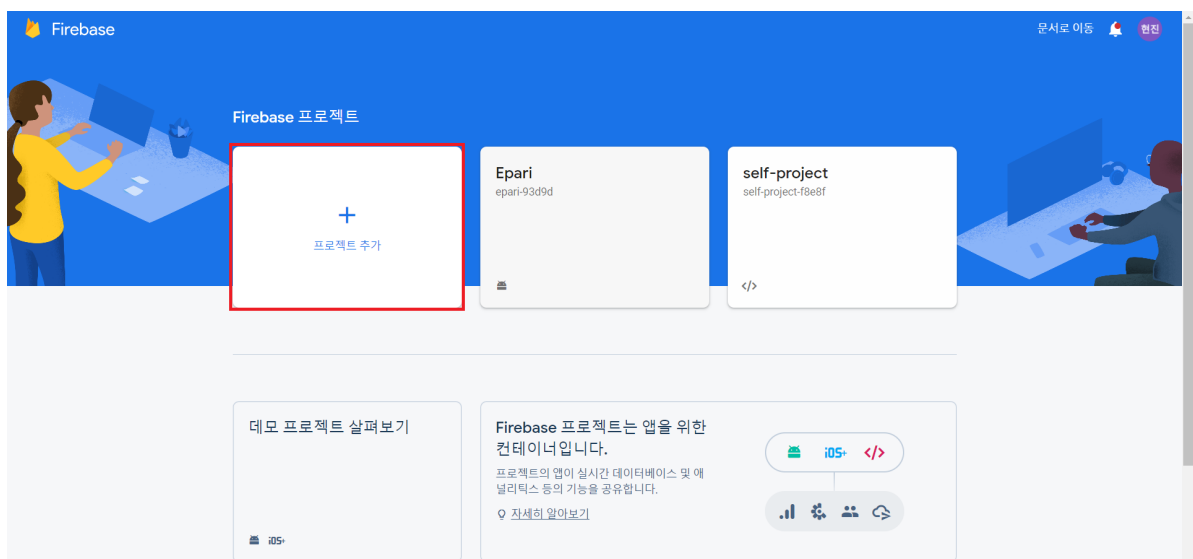
프로젝트 생성

1-1. Firebase console 접속

<https://console.firebase.google.com/>



1-2 프로젝트 추가



1-3 프로젝트 이름 입력

× 프로젝트 만들기(1/3단계)


프로젝트 이름을 지정하여 시작하기[Ⓞ]

프로젝트 이름

epari

epari-728f6

계속



1-4 Google 애널리틱스 계정 선택 후 프로젝트 생성

× 프로젝트 만들기(3/3단계)

Google 애널리틱스 구성

Google 애널리틱스 계정 선택 또는 만들기[Ⓞ]

Default Account for Firebase


이 계정에서 자동으로 새 속성 만들기 ✎

프로젝트를 만들면 선택한 Google 애널리틱스 계정에 새 Google 애널리틱스 속성이 생성되고 Firebase 프로젝트에 연결됩니다. 이 연결을 통해 제품 간에 데이터 흐름이 활성화됩니다. Google 애널리틱스 속성에서 Firebase로 나보낸 데이터에는 Firebase 서비스 약관이 적용되지만 Google 애널리틱스로 가져온 Firebase 데이터에는 Google 애널리틱스 서비스 약관이 적용됩니다.

[자세히 알아보기](#)

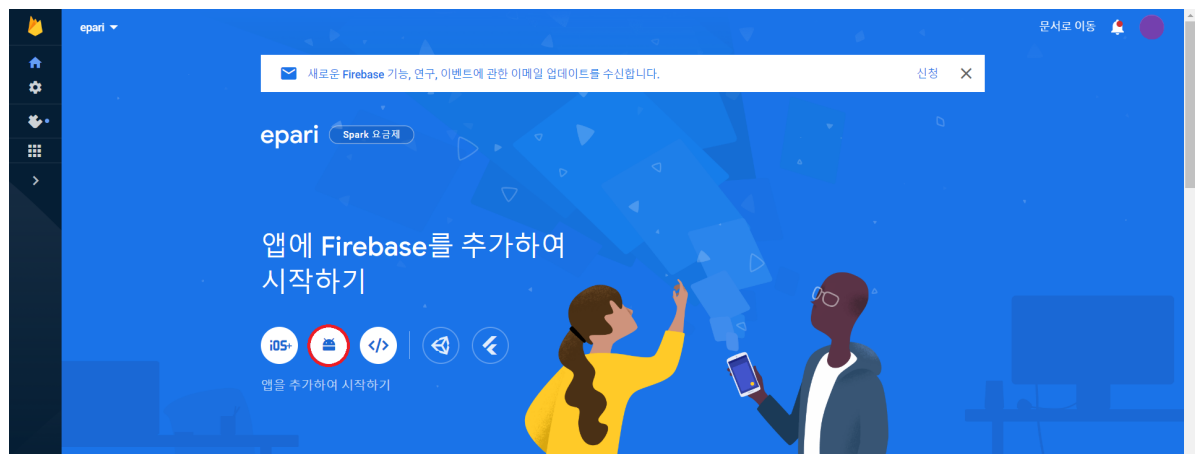
이전

프로젝트 만들기



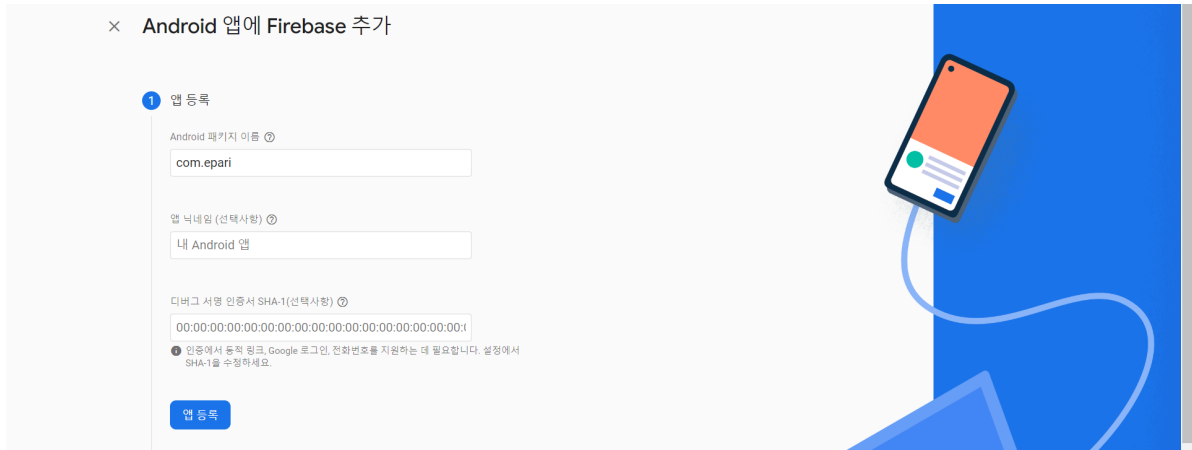
2. Firebase 프로젝트와 앱 연결

2-1 앱 추가하기(Android 기준)

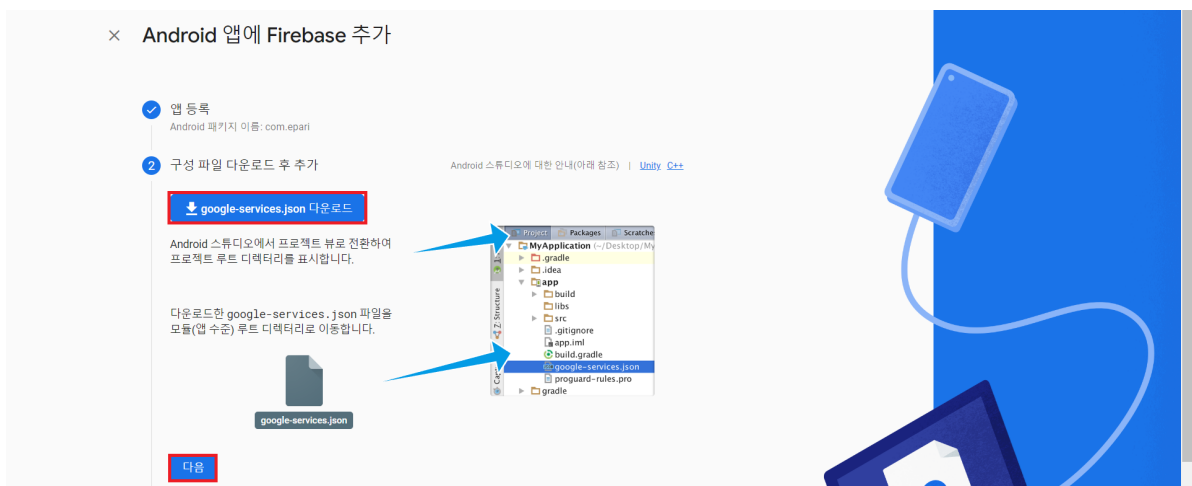


2-2 Android 패키지 이름 입력

- 패키지 이름은 com.company.appname 형식



2-3 google-services.json 파일을 받아 앱 폴더에 추가



2-4 google-services 플러그인 추가

- `/android/build.gradle` 파일

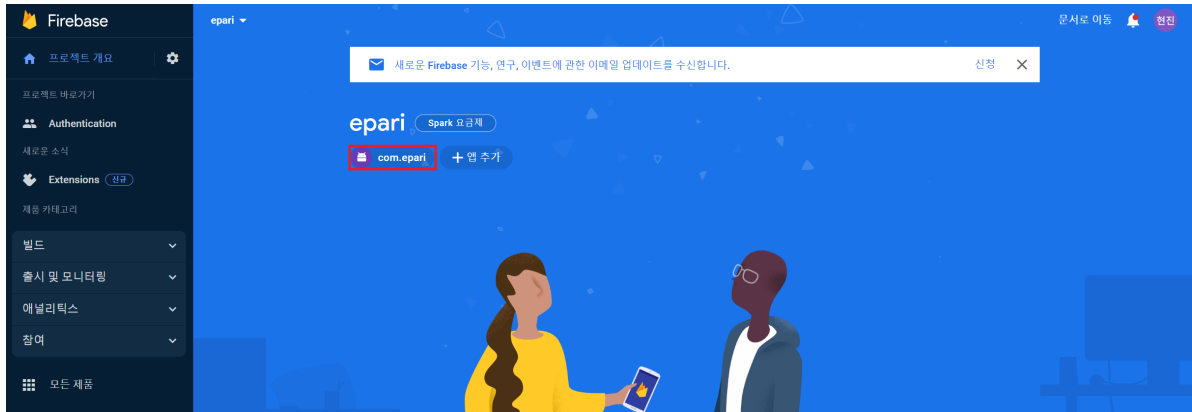
```
buildscript {
    dependencies {
        // ...
        classpath 'com.google.gms:google-services:4.3.14' // <- add this line
    }
}
```

- `android/app/build.gradle` 파일

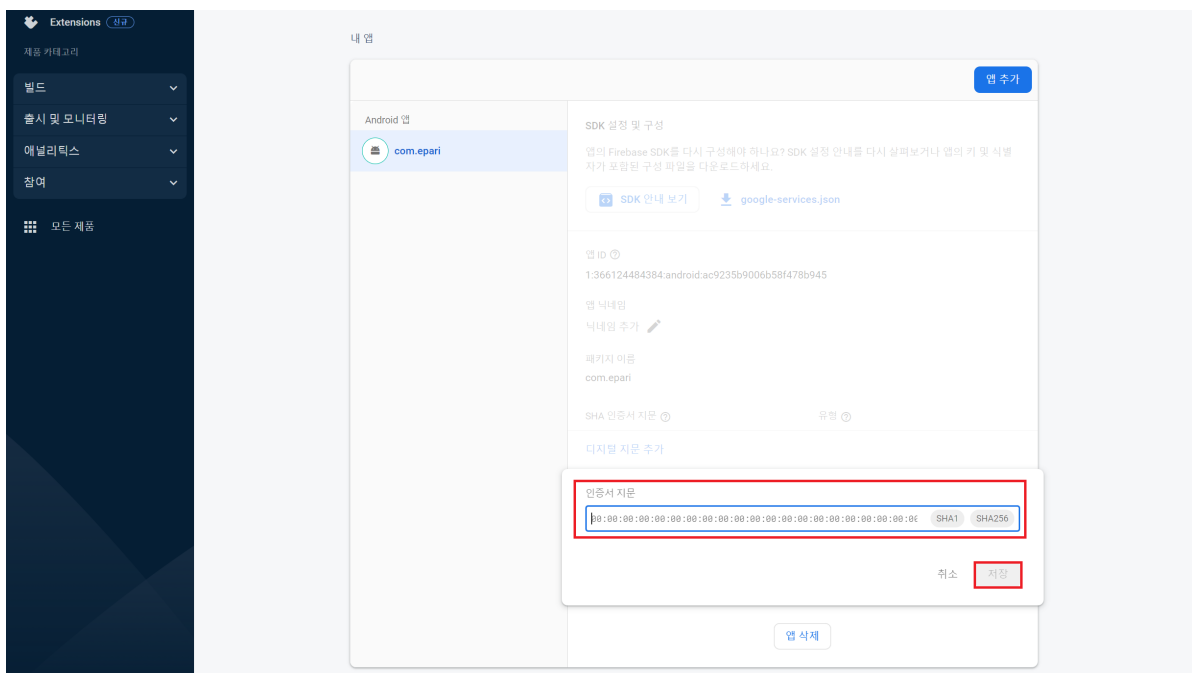
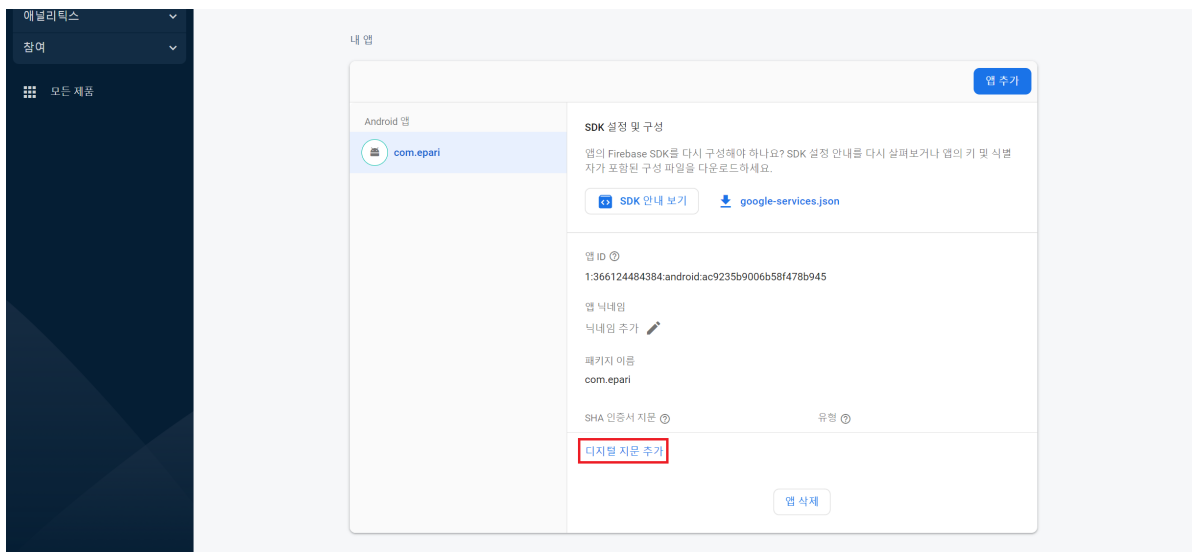
```
apply plugin: 'com.android.application'
apply plugin: 'com.google.gms.google-services' // <- Add this line
```

2-5 SHA 인증서 지문 추가

- 추가된 앱의 설정

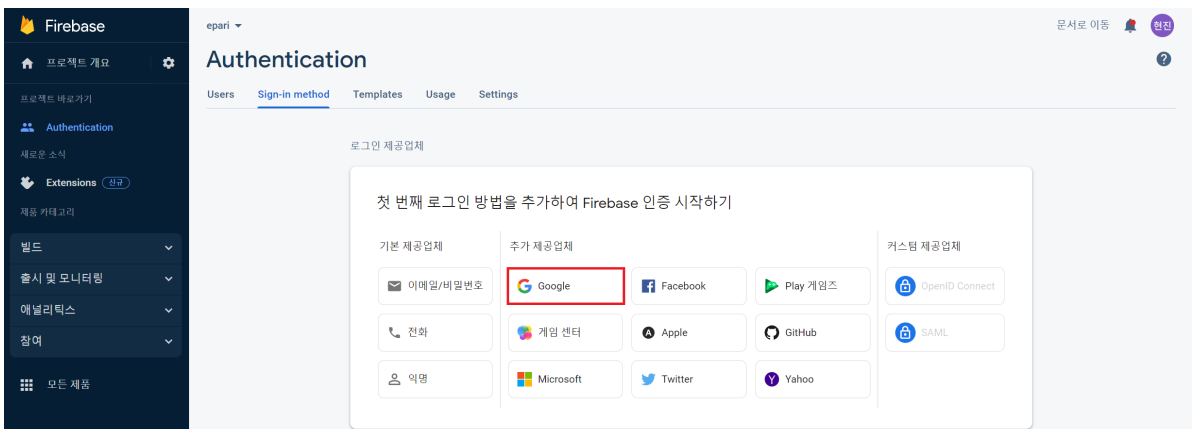
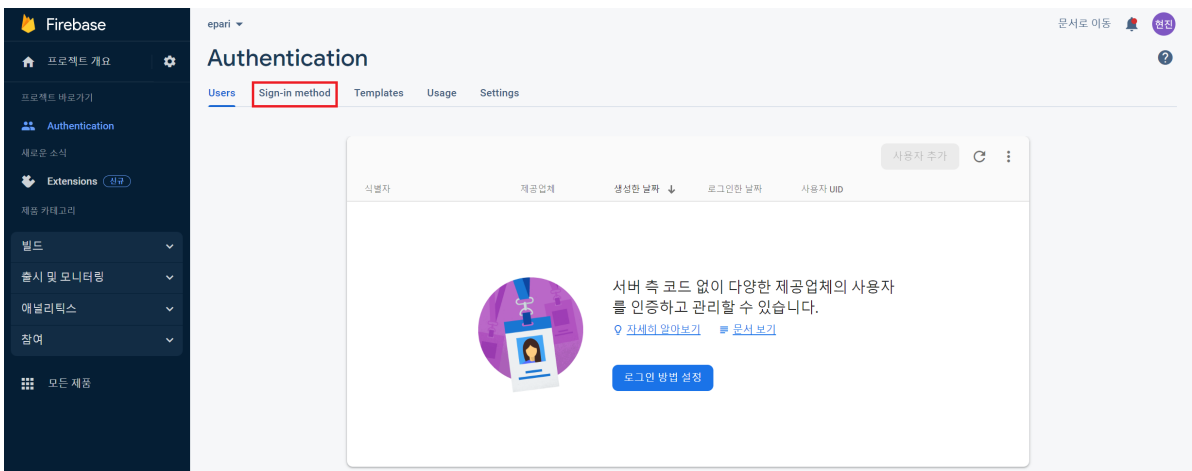
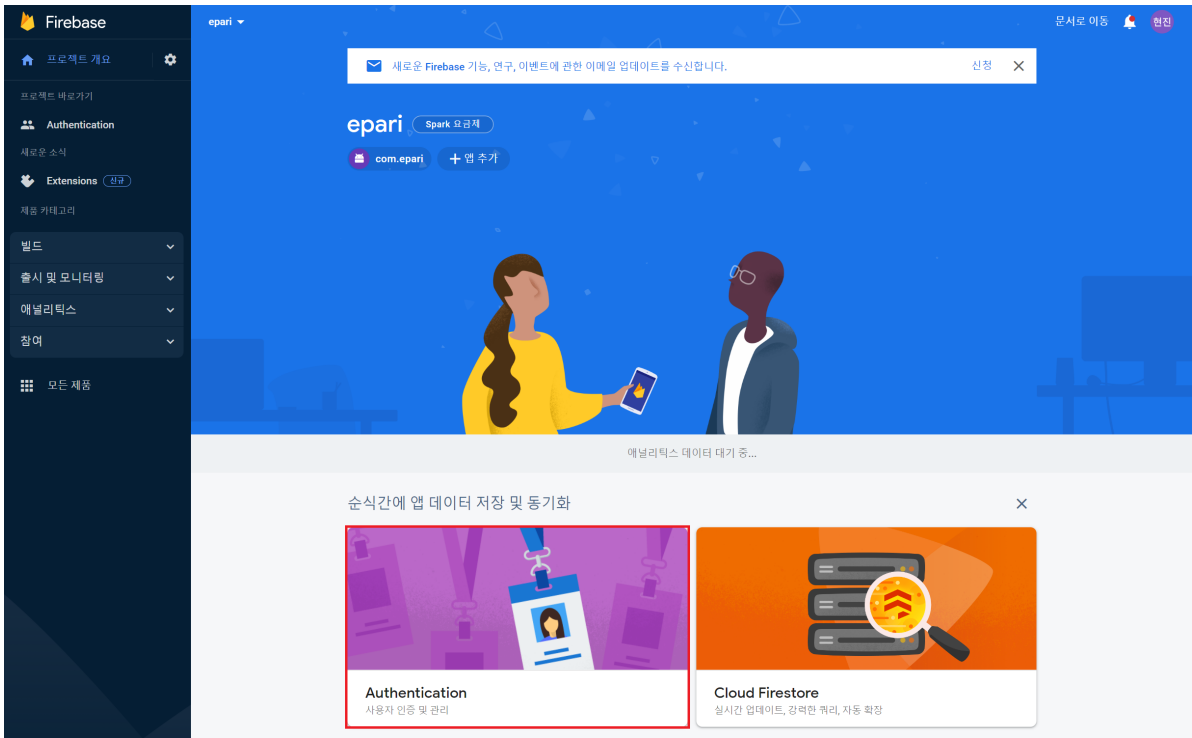


- SHA-1, SHA-256 인증서 지문을 입력 후 저장



3. Google 로그인 활성화

3-1 제공업체 활성화



epari

문서로 이동

회원

?

Authentication

UsersSign-in methodTemplatesUsageSettings

로그인 제공업체

Google

사용 설정

연결된 Apple 및 웹 앱에서는 Google 로그인이 자동으로 구성됩니다. Android 앱에 대한 Google 로그인을 설정하려면 [프로젝트 설정](#)에서 각 앱에 대해 [SHA-1 디지털 지문](#)을 추가해야 합니다.

계속하려면 아래에서 [프로젝트 수준 설정](#)을 업데이트하세요.

프로젝트의 공개용 이름

project-366124484384

프로젝트 지원 이메일

외부 프로젝트의 클라이언트 ID 허용 목록에 추가(선택사항)

추가

클라이언트 ID를 허용 목록에 추가하려면 Google 로그인을 사용 설정하세요.

웹 SDK 구성

웹 클라이언트 ID

웹 클라이언트 ID를 설정하려면 Google 로그인을 사용 설정하세요.

웹 클라이언트 보안 비밀번호

웹 클라이언트 보안 비밀번호를 설정하려면 Google 로그인을 사용 설정하세요.

취소

저장