우리, 다 배포 매뉴얼

본 문서는 안드로이드로 개발된 우리,다을 사용하기 위한 가이드를 프론트엔드(안드로이드)와 백엔드 장고, 스프링 서버를 배포하는 과정에 대해 서술하고 있습니다.

# **1. 안드로이드 배포**

## 1.1. 안드로이드 개발 환경

- Android Studio Dolphin (2021.3.1 Patch1)

- targetSDK 32

- minSDK 21

- Kotlin

## 1.2. AOS 기술 스택

- Retrofit 2.9.0 (통신 라이브러리)

- Coroutines Flow (비동기 데이터 처리 라이브러리)

- Glide 4.13.2 (이미지 로드 라이브러리)

- ViewModel-ktx 2.6.0-alpha01

- livedata-ktx 2.6.0-alpha01

- Naver Login(네이버 로그인 라이브러리)

- Kakao Login(카카오 로그인 라이브러리)

- Room (내부 데이터 베이스 라이브러리)

- ML Kit:object-detection-custom:17.0.0 (객체 감지 라이브러리)

- ML Kit:translate:17.0.1(번역 라이브러리)

## 1.3. 앱 빌드를 위한 과정

**1.3.1. Android Studio 빌드 방법**

1.3.1.1. local.properties 값 설정

1. Gradle Scripts > local.properties 에

BASE\_URL="<http://j8d202.p.ssafy.io:>"

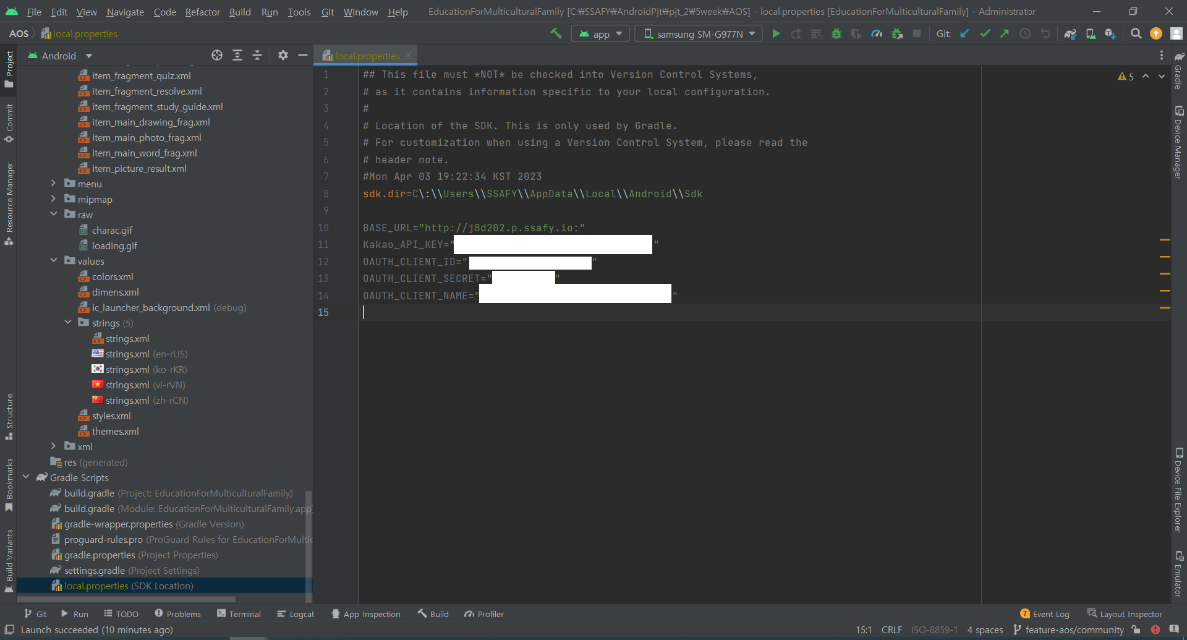
Kakao\_API\_KEY=""

OAUTH\_CLIENT\_ID=""

OAUTH\_CLIENT\_SECRET=""

OAUTH\_CLIENT\_NAME=" "

를 추가한다.



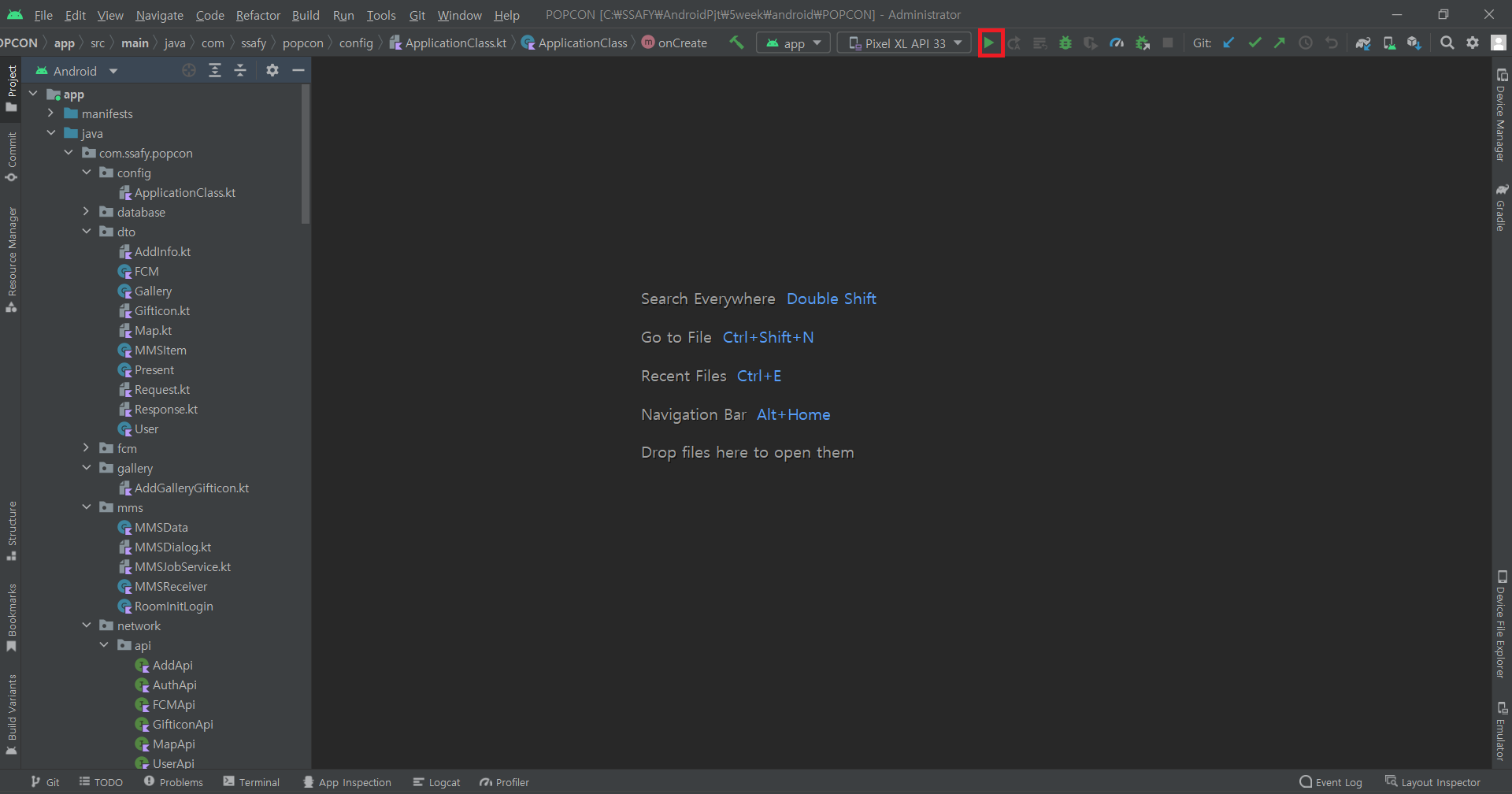
1. BASE\_URL을 제외한 나머지 값들은 [ 외부 서비스 설정 ]을 참고한다.

1.3.1.2. App

1. 스마트폰을 연결한다.
2. 왼쪽 상단에 Run/Debug Configurations이 app으로 선택되어 있는지 확인 후, 스마트폰이 연결 되어있는지 확인한다.



1. 녹색 버튼을 눌러 앱을 설치 및 실행한다.

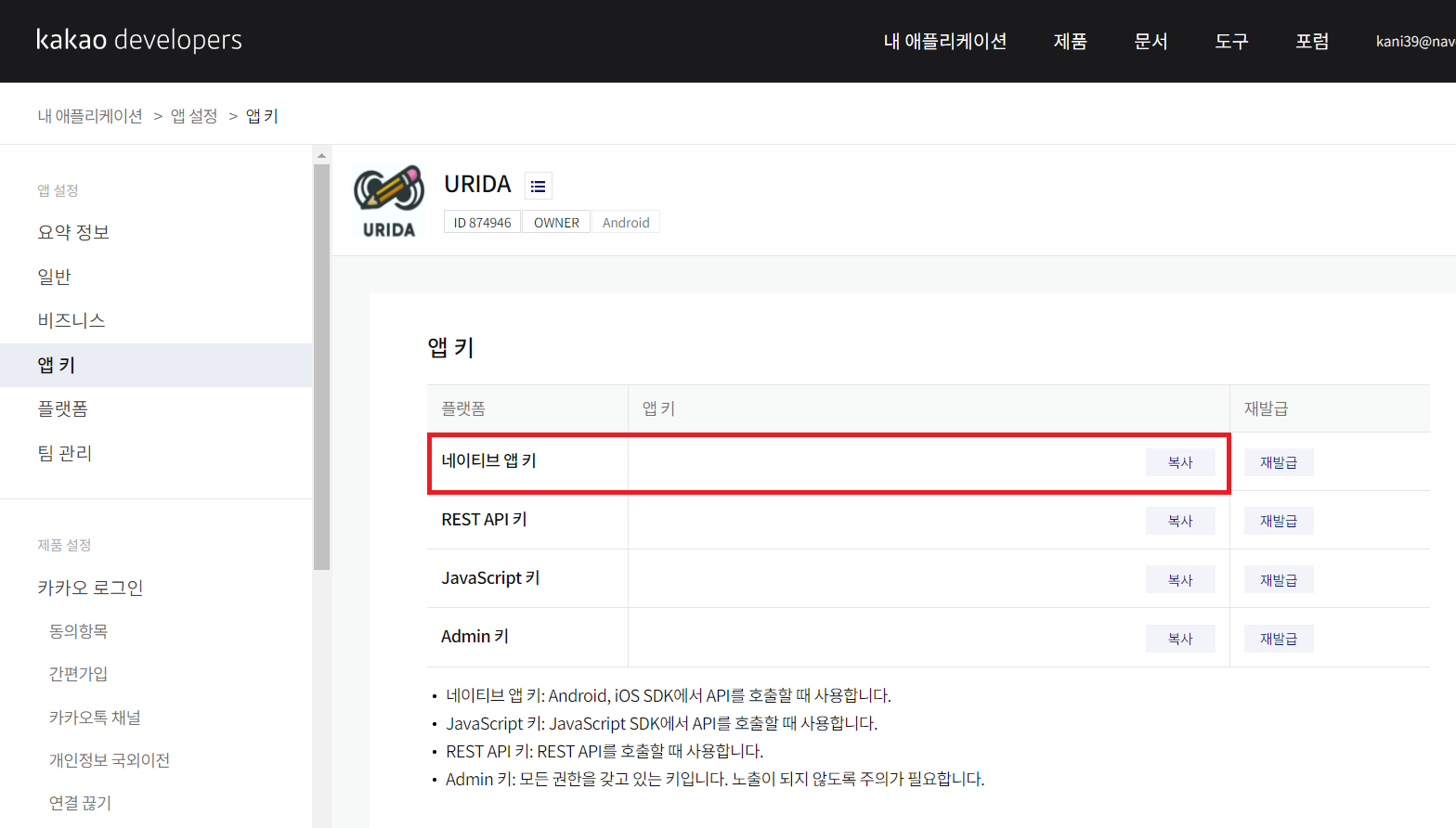


**1.3.2. 외부 서비스 설정**

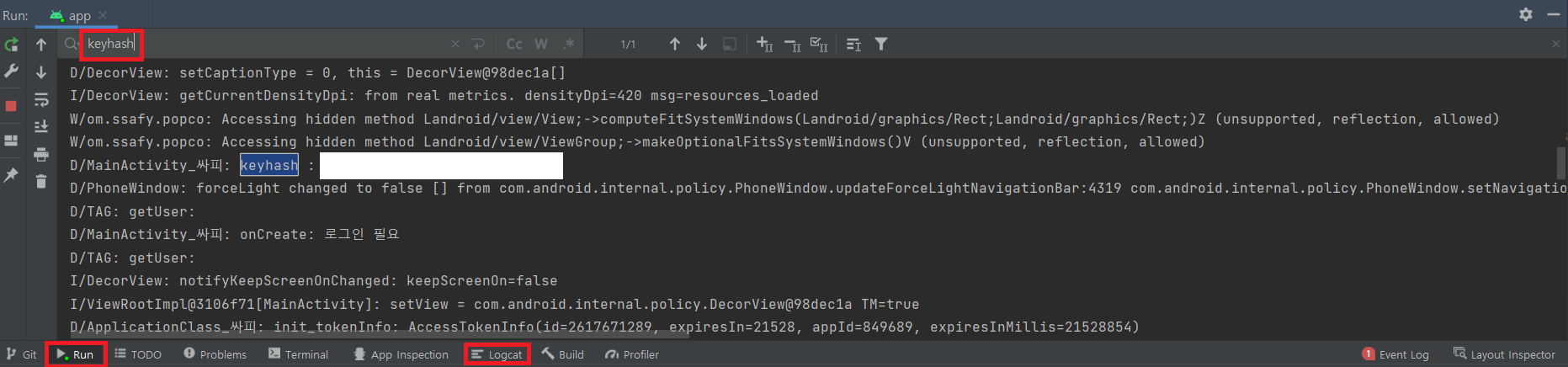
1.3.2.1. Kakao Dev 설정 (소셜 로그인)

1. 내 애플리케이션 > 앱 설정 > 앱 키에 들어가서 네이티브 앱 키를

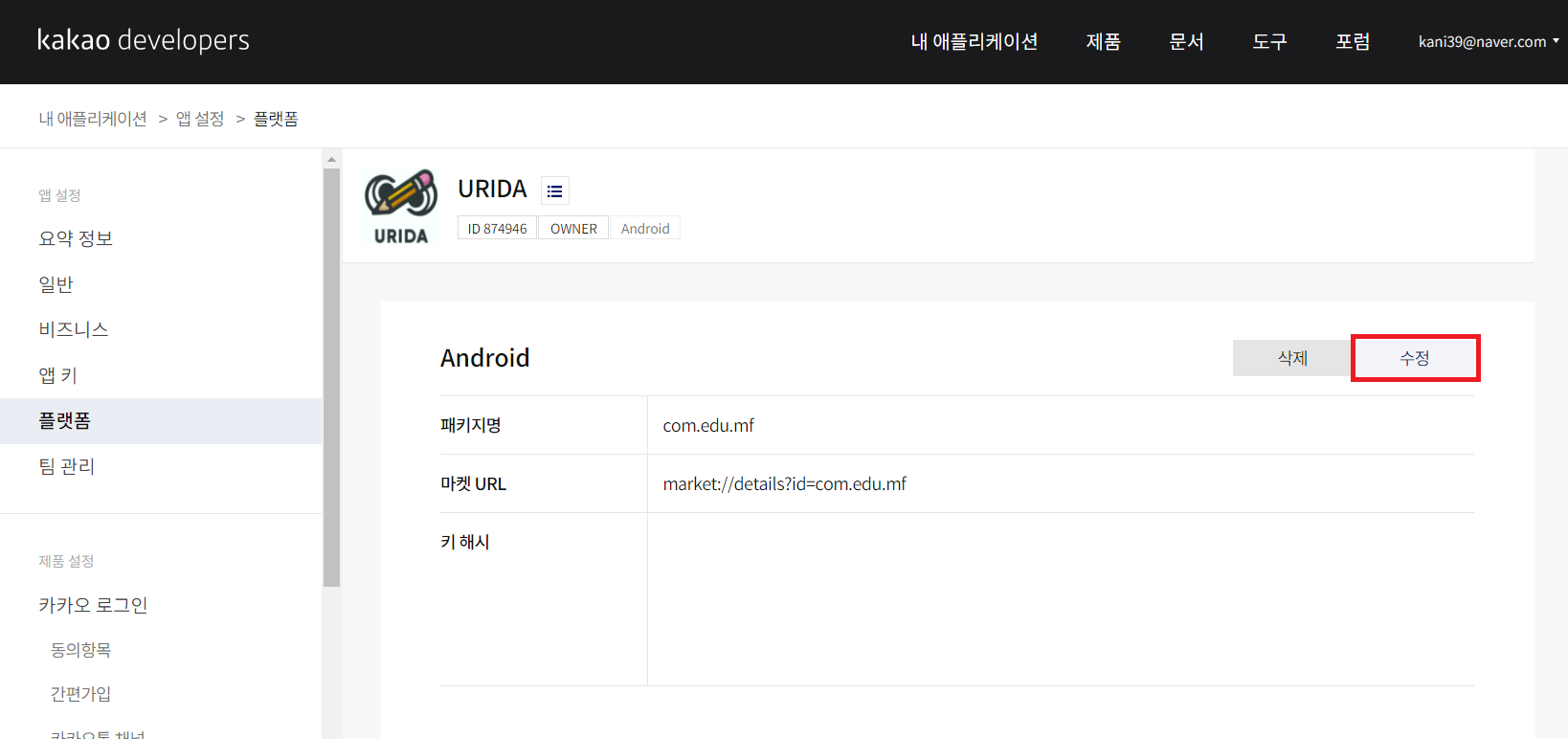
안드로이드 스튜디오의 Gradle Scripts > local.properties의 ‘Kakao\_API\_KEY‘에 추가한다.



1. 앱을 실행한다.
2. Run 또는 Logcat에 keyhash를 검색하여 복사한다.

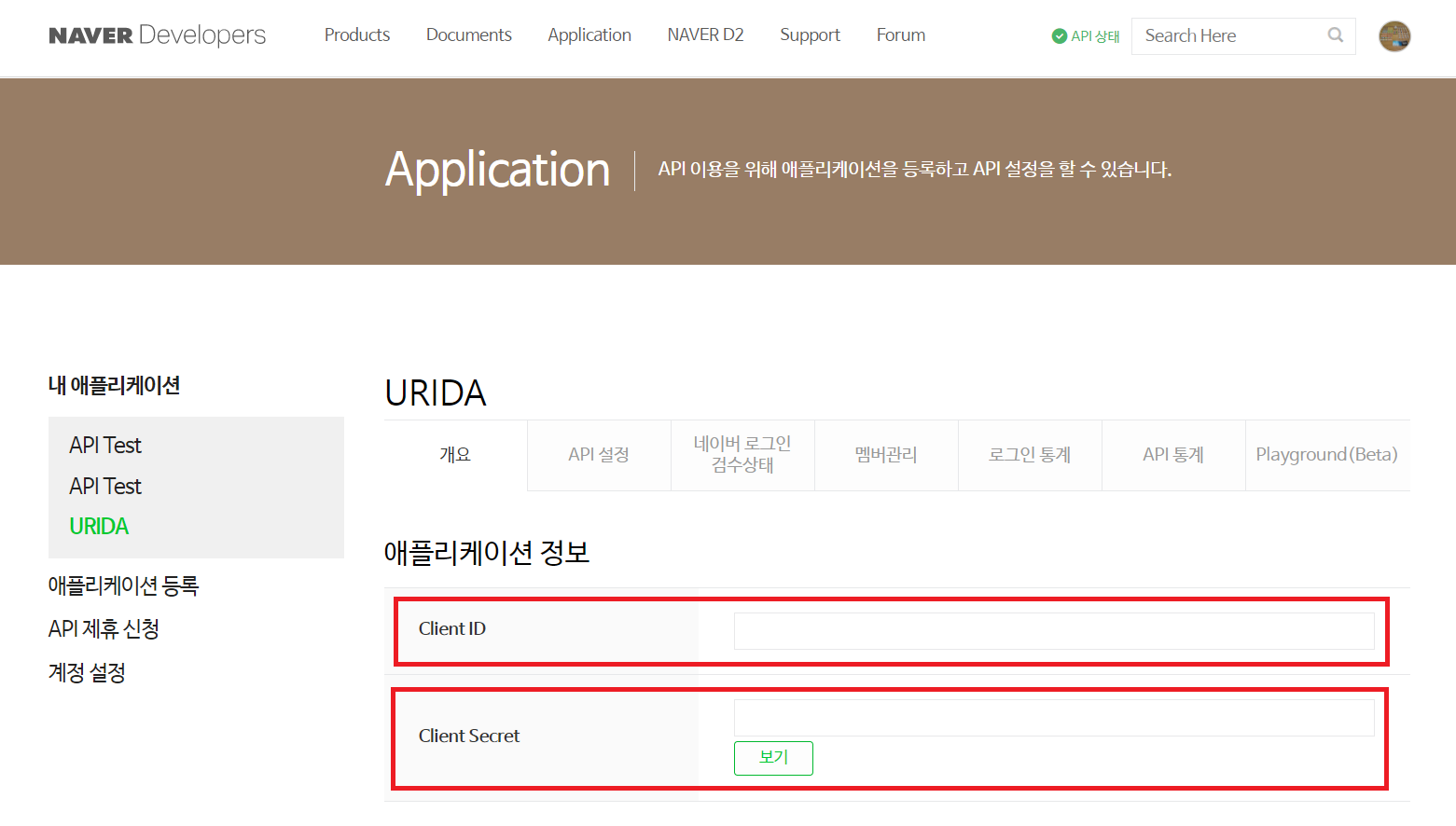


1. 내 애플리케이션 > 앱 설정 > 플랫폼에 들어가서 Android의 수정 버튼을 클릭 후 키 해시에 (3)의 값을 붙여넣는다.



1.3.2.2. Naver Dev 설정 (소셜 로그인)

1. Application > 내 애플리케이션 > 개요 > 애플리케이션 정보에 들어간다.



1. Client ID를 복사하여 안드로이드 스튜디오의 Gradle Scripts > local.properties의 ‘OAUTH\_CLIENT\_ID’에 붙여 넣는다.
2. Client Secret를 복사하여 안드로이드 스튜디오의 Gradle Scripts > local.properties의 ‘OAUTH\_CLIENT\_SECRET’에 붙여 넣는다.
3. 안드로이드 스튜디오의 Gradle Scripts > local.properties의 ‘OAUTH\_CLIENT\_NAME’에 서비스 이름인 URIDA를 넣는다.

# **2. api 서버 배포 과정**

## 2.1. 개발환경

- JAVA : Oracle Java 11.0.16

- Spring Boot 2.7.9

- Python Version : 3.9.13 (local) , 3.8.10(EC2 ubuntu)

- Django version : 3.2.13

- IDE : Intelli J 2022.3

- IDE 2: vs code latest(최신) 버전

-빌드 도구 : (스프링) Jenkins 와 docker

(장고) pip 22.3.1 과 가상환경

-API 서버를 위한 서버 관련 주요 라이브러리 목록과 버전

**Lombock 과 스프링 부트 tool 들 jdbc , swaager , 구글 클라우드와 연동, mysql(DB) 와 연동 라이브러리 등**

'org.projectlombok:lombok'   
'org.springframework.boot:spring-boot-devtools'  
'org.springframework.boot:spring-boot-starter-test'  
name: 'springfox-swagger2', version: '2.9.2'   
name: 'springfox-swagger-ui', version: '2.9.2'  
'org.springframework.boot:spring-boot-starter-web'  
'org.springframework.boot:spring-boot-starter-jdbc'  
'com.h2database:h2'  
'org.springframework.boot:spring-boot-starter-validation'  
group: 'org.springframework.cloud', name: 'spring-cloud-gcp-starter', version: '1.2.5.RELEASE'  
'com.mysql:mysql-connector-j'

-AI 모듈을 위한 서버 관련 주요 라이브러리 목록과 버전:

**오디오와 이미지 처리 라이브러리, 문자열 처리 라이브러리, 텐서플로우, 파이 토치, 장고 등**

astropy==5.2.2

charset-normalizer==2.1.1

Django==3.2.13

django-imagekit==4.1.0

djangorestframework==3.14.0

librosa==0.10.0.post2

matplotlib==3.7.1

msgpack==1.0.4

numba==0.56.4

numpy==1.23.5

oauthlib==3.2.2

omegaconf==2.3.0

pandas==1.5.3

pilkit==2.0

Pillow==9.4.0

scikit-learn==1.2.1

sklearn2==0.0.13

sklearn3==0.0.1

sniffio==1.3.0

soundfile==0.12.1

tensorboard==2.12.0

tensorflow==2.12.0

torch==1.13.1

torchaudio==0.13.1

## 2.2. 배포 환경

배포 서버 : Ubuntu 20.04.4 LTS

Docker : 23.0.1

Docker Image 목록

**parkjonggeon/urida:board**

**parkjonggeon/urida:problem**

**parkjonggeon/urida:user**

**hungry-jones (Mysql)**

**jenkins**

## 2.3. 서버 배포를 위한 과정

**스프링**

1. <https://lab.ssafy.com/s08-ai-speech-sub2/S08P22D202.git> 를 clone 한다.

2. develop 브랜치에 머지 되는 순간 젠킨스를 이용한 자동 배포가 됨

3. 수동 배포를 위해선 각 소스코드의 gradlew 페이지가 있는 위칭ㅅ각 이미지 (모듈 ) 별로 다음과 같은 명령어를 입력하면 된다.

# user 모듈 배포 예시

**./gradlew :urida-user:build**

**Sudo docker stop user-develop**

**Sudo docker rm user-develop**

**Sudo docker build -t parkjonggeon/urida:user .**

**Sudo docker run -d -p 8081:8081 --name user-develop parkjonggeon/urida:user**

4. 관리자 페이지는 따로 없다.

**장고**

1. <https://lab.ssafy.com/luminaries1/sdfsfsdfs.git> 를 /home/ubuntu/aimodule 디렉토리에서 pull 받는다.

2. gunicorn.service 의 내용은 다음과 같고,

[Unit]

Description=gunicorn daemon

After=network.target

[Service]

User=ubuntu

Group=www-data

WorkingDirectory=/home/ubuntu/aimodule/djago\_ai/django\_ai

ExecStart=/home/ubuntu/aimodule/djago\_ai/django\_ai/venv2/bin/gunicorn \

--workers 3 \

--bind 0.0.0.0:8084 \

django\_ai.wsgi:application \

--timeout 600

[Install]

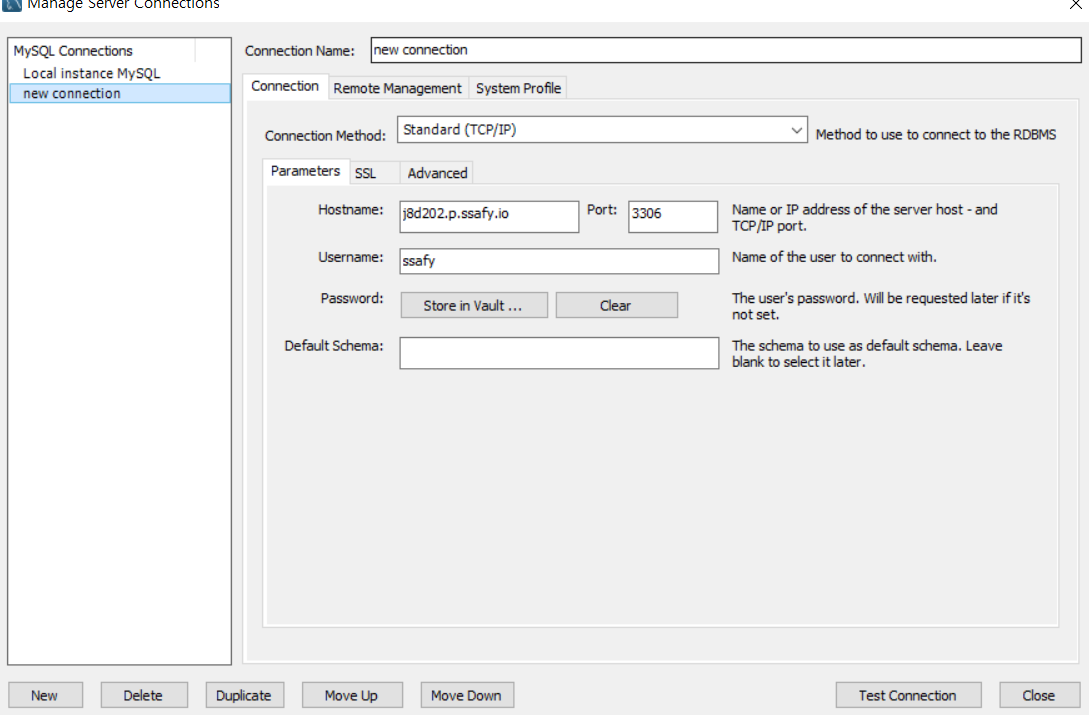
WantedBy=multi-user.target

3. 해당 파일을 수정하였다면 sudo systemctl daemon-reload

4. sudo systemctl restart gunicorn 명령어를 차례로 입력하면 배포 완료

## 2.4. DB 접속을 위한 세팅 정보

* Mysql 최신버전 이미지
* Workbench를 통한 connection 정보



* 계정정보는 ID : ssafy PW: saffy
* 기본적으로 사용하는 Database 는 urida

## 2.5. 배포시 특이사항

- ai 모듈과 BE + FE 를 위한 git repository가 따로 되어있다.

- 각 모듈별로 포트가 모두 다르다.

- 8080: Jenkins

- 8081: User (인증과 소셜로그인)

- 8082: Problem (다시 풀기를 위한 문제 정보 모듈)

- 8083: Board ( 커뮤니티를 위한 api)

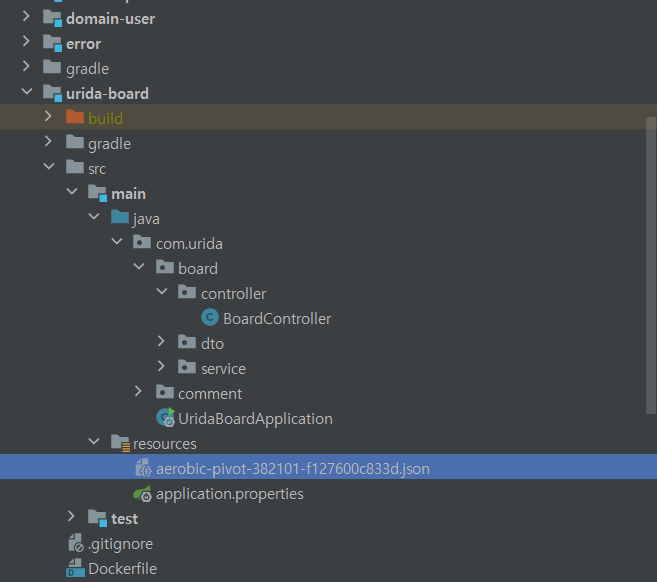
- 8084: Django (음성, 이미지 ai 모듈)

- <http://j8d202.p.ssafy.io:8081/swagger-ui.html#/> 등 8081 8082 8083 포트를 통해 swagger 를 확인 할 수 있다.

- Django 서버만 gunicron 을 통한 백그라운드 수동 실행을 통해 배포 나머지는 Jenkins를 통한 이미지 자동 배포 상태이다.

- 커뮤니티 이미지 저장을 위해 구글 cloud storage 를 저장공간으로 사용하고 있다.

- 해당 연동을 위한 계정 정보 파일의 소스 코드에서의 위치는 다음과 같다.



aerobic-pivot-382101-f127600c833d.json 의 파일 내용은

{  
 "type": "service\_account",  
 "project\_id": "aerobic-pivot-382101",  
 "private\_key\_id": "f127600c833d8cde7c4077fb81df24e6ecdb8ced",  
 "private\_key": “{키 값}”  
 "client\_email": "springboot-storage-access@aerobic-pivot-382101.iam.gserviceaccount.com",  
 "client\_id": "113983899059753147599",  
 "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": "https://www.googleapis.com/robot/v1/metadata/x509/springboot-storage-access%40aerobic-pivot-382101.iam.gserviceaccount.com"  
}

Application.properties 에서 해당 연동 정보를 기록한다.

spring.cloud.gcp.storage.credentials.location=classpath:aerobic-pivot-382101-f127600c833d.json  
spring.cloud.gcp.storage.project-id=aerobic-pivot-382101