****

**알쏭달쏭**

**(ALSONG DALSONG)**

“감정 기반 음악 추천 일기 서비스”

**SSAFY 특화 프로젝트**

**포팅 매뉴얼**

* D204 -

**팀장** 박주현

**팀원** 김효선 박소정 신혜연 조경수 홍석현

1. **목 차**

**Ⅰ. 프로젝트 기술 스택**·······························································································3

**Ⅱ. 백엔드 빌드 방법**····································································································5

**Ⅲ. 프론트엔드 빌드 방법**···························································································6

**Ⅳ. CI/CD 설정 및 명령어 정리**···············································································7

**Ⅴ. DB 설정**·····················································································································15

**Ⅵ. Google Console 설정**·························································································17

**Ⅵ. Kakao Console 설정**···························································································19

**Ⅶ. AWS S3 사용 방법**·······························································································21

**Ⅷ. 환경 변수 설정**······································································································29

**Ⅸ. gitignore** ·················································································································30

**Ⅰ. 프로젝트 기술 스택**

* Backend
  + - python 3.7
    - Djangorestframework
    - Django 3.2.12
    - Djangorestframework-simplejwt
    - Oauth2
    - Amazone S3
* FrontEnd
  + - React 18.2.0
    - moment.js 2.29.4
    - react-bootstrap 2.5.0
    - react-router-dom 6.4.0
    - react-dom 18.2.0
    - styled-components 5.3.5
* DATA
  + - Jupyter notebook
    - Python 3.7
    - Django 3.2.12
* 배포
  + - AWS EC2
    - Nginx
    - Docker
    - Jenkins

**Ⅱ. 백엔드 빌드 방법**

1. **Django 프로젝트 폴더 이동**
2. **가상환경 설정**

python –m venv <가상환경이름>

-Windows : source <가상환경이름>/Scripts/activate

-Mac OS : source <가상환경이름>/bin/activate

1. **패키지 설치**

pip install requirements.txt

1. **Migration**

python manage.py makemigrations (Migration 생성)

python manage.py migrate (Migration 적용)

1. **서버 구동**

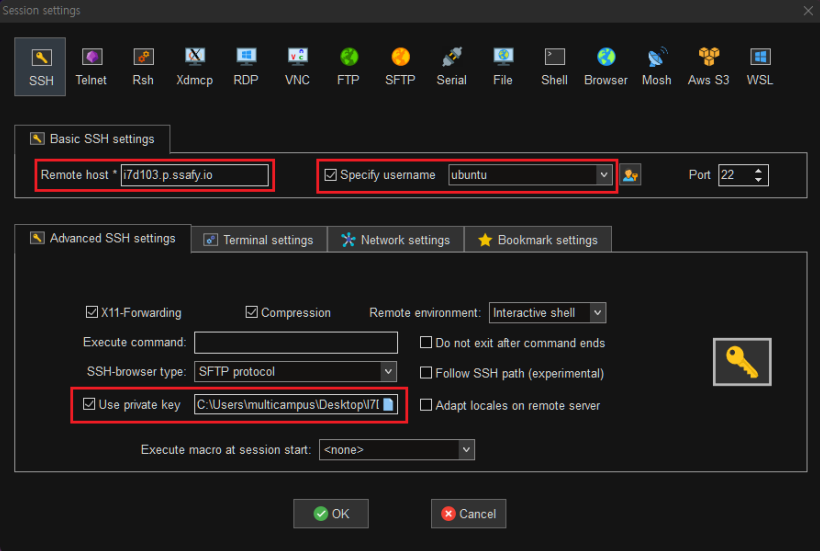
python manage.py runserver

**Ⅲ. 프론트엔드 빌드 방법**

* + **npm install**
  + **npm run build**

**Ⅳ. CI/CD 설정 및 명령어 정리**

1. **EC2 접속을 위한 SSH 설정**

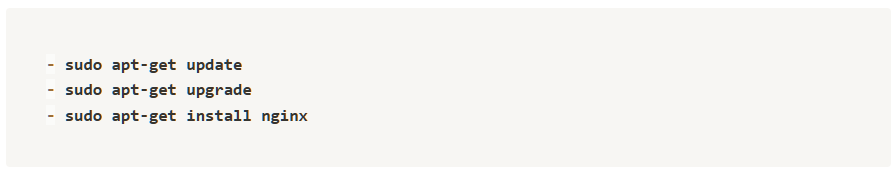
****

- Remote host : AWS EC2 ip 주소 또는 연결된 도메인

- username : 사용자명

- Use private key 체크, 발급받은 key 파일 설정

1. **Nginx 설치**



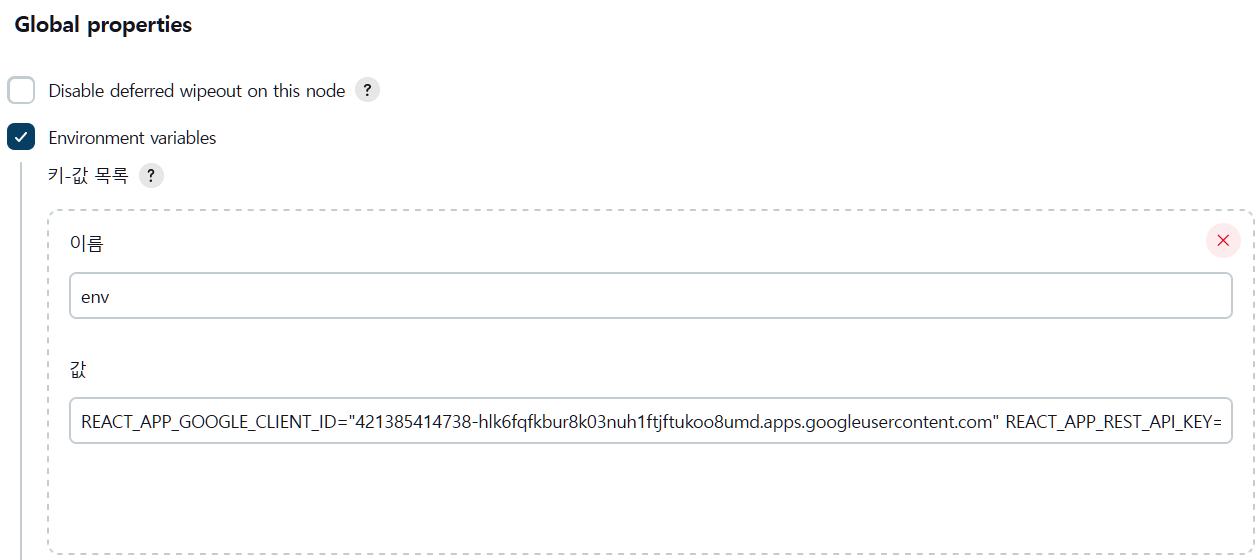
1. **docker 설치**

****

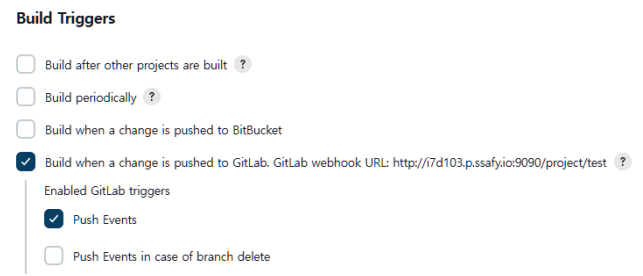
1. **Jenkins 컨테이너 실행**



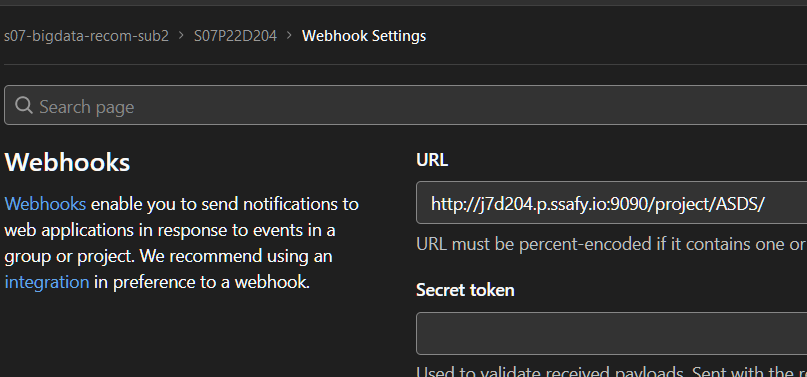
1. **Jenkins 실행 후 빌드 파라미터 등록**



1. **Webhook 등록**
   1. **Jenkins**



* 1. **GitLab**



1. **Jenkins pipeline script**

pipeline {

        agent none

        stages {

                stage('Create .env') {

                        agent any

                        steps {

                                sh 'echo "${env}" > .env'

                                sh 'cat .env'

                                sh 'ls -al'

                                sh 'cp .env frontend/music-diary'

                                sh 'rm -rf backend/.config\_secrets'

                                sh 'mkdir backend/.config\_secrets'

                                // sh 'rm secrets.json'

                                sh 'echo "${secrets\_json}" > secrets.json'

                                sh 'cat secrets.json'

                                sh 'ls -al'

                                sh 'mv secrets.json backend/.config\_secrets'

                                // sh 'rm settings\_common.json'

                                sh 'echo "${settings\_common}" > settings\_common.json'

                                sh 'cat settings\_common.json'

                                sh 'ls -al'

                                sh 'mv settings\_common.json backend/.config\_secrets'

                        }

                }

                stage('Docker build') {

                        agent any

                        steps {

                                sh 'docker build -t backimg ./backend'

                                sh 'docker build -t frontimg ./frontend/music-diary'

                                sh 'echo hello2'

                        }

                }

                stage('Docker run') {

                        agent any

                        steps {

                                sh 'docker ps -f name=front -q \

                                        | xargs --no-run-if-empty docker container stop'

                                sh 'docker ps -f name=back -q \

                                        | xargs --no-run-if-empty docker container stop'

                                sh 'docker container ls -a -f name=front -q \

                                        | xargs -r docker container rm'

                                sh 'docker container ls -a -f name=back -q \

                                        | xargs -r docker container rm'

                              sh 'docker run -d --name front -p 80:80 frontimg'

                                sh 'docker run -d --name back -p 8080:8080 backimg'

                                sh 'echo hello3'

                        }

                }

        }

}

**1) stage (”Create .env”)**

A. Jenkins 에 등록한 환경변수를 사용해 .env 파일과 secrets.json 파일

B. 긱 파일을 frontend 디렉토리로 복사

**2) stage (Docker build”)**

A. backend 폴더에 있는 Dockerfile 을 backimg 라는 이름으로 이미지화

B. frontend 폴더에 있는 Dockerfile frontimg 라는 이름으로 이미지화

**3) stage(”Docker run”)**

A. front 라는 이름의 컨테이너가 있으면 stop

B. back 이라는 이름의 컨테이너가 있으면 stop

C. front 라는 이름의 컨테이너가 있으면 remove

D. back 이라는 이름의 컨테이너가 있으면 remove

E. 빌드된 frontimg 컨테이너를 80 포트에서 실행

F. 빌드된 backimg 컨테이너를 8080 포트에서 실행

**8. Dockerfile 설정**

1. **Backend**

****

- 패키지 설치

- Django 서버 8080 포트에서 실행

1. **Frontend**



- /usr/share/nginx/html/\* 삭제

- build 폴더를 /usr/share/nginx/html 로 복사

1. **nginx 세팅**

- /etc/nginx/conf.d/default.conf 삭제

- ./nginx/deploy.conf 파일을 /etc/nginx/conf.d/deploy.conf 복사

**- deploy.conf**

****

- 80 포트를 기본 포트로 설정: 포트 입력 없으면 80 포트로 연결

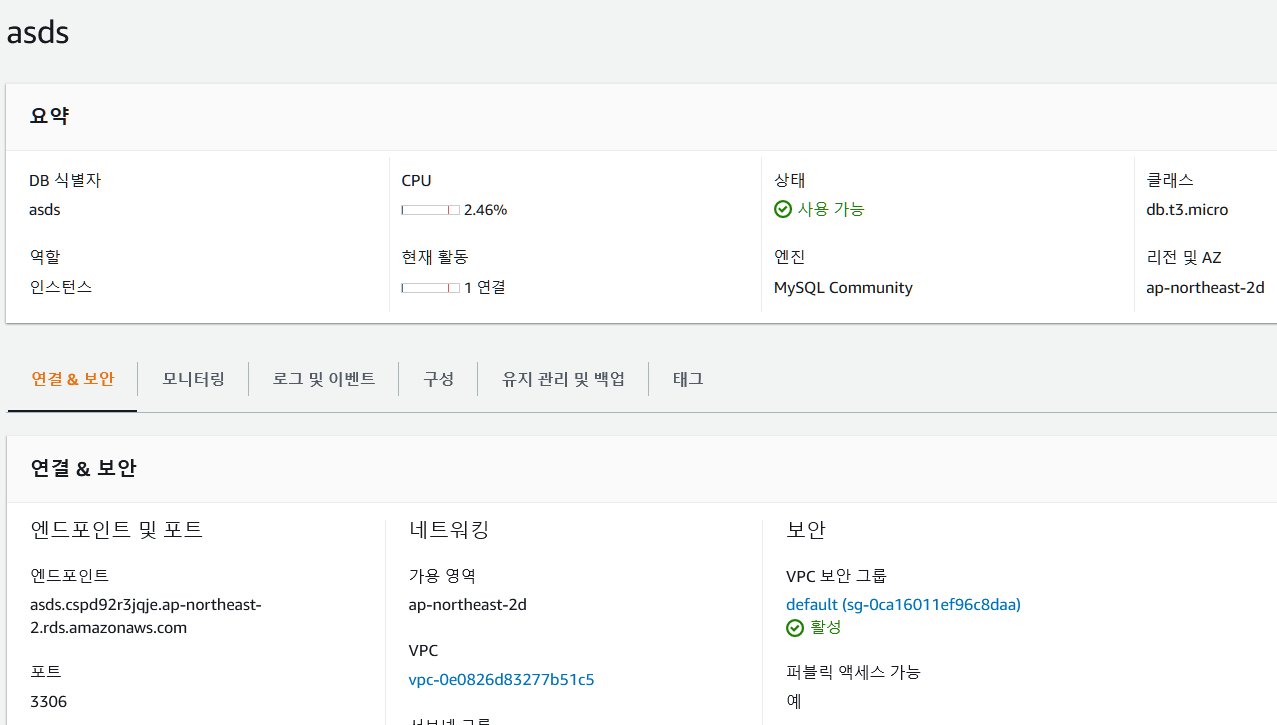
- 기본 URI 에서 user/share/nginx/html 에 있는 index.html 을 전달

- CORS Header 추가

- 프록시 설정을 통해 /rest 로 시작하는 uri 를 8080 포트로 연결

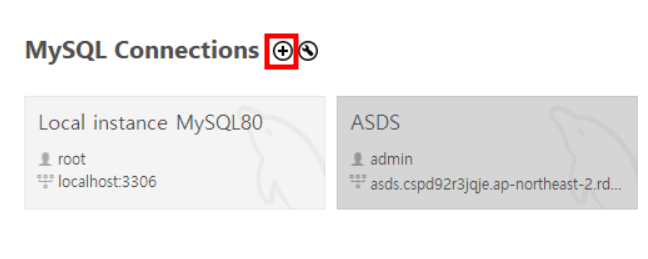
**Ⅴ. DB 설정**

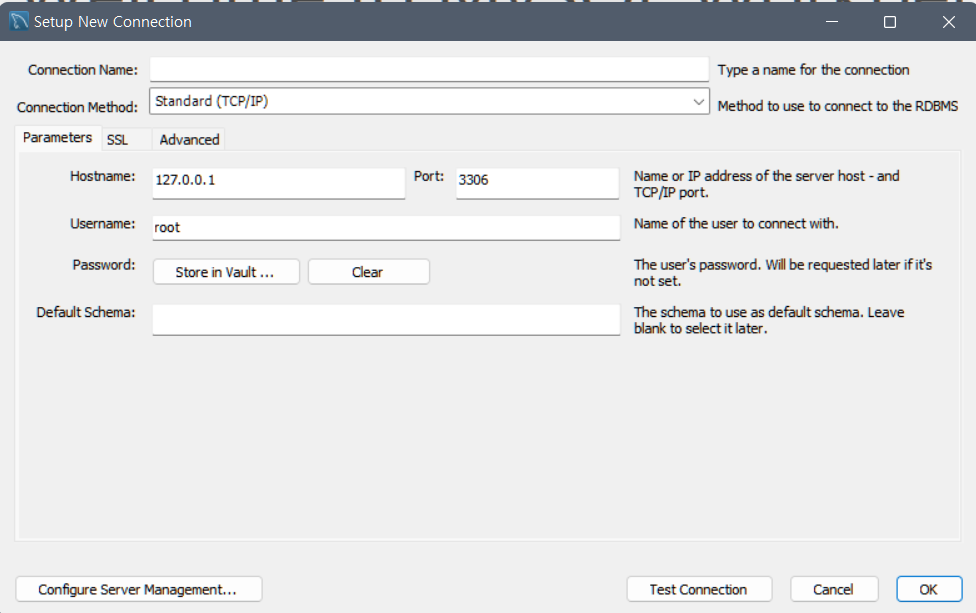
1. **AWS RDS에서 엔진으로 MySQL 선택**

****

1. **Django Projcet와 연동**
   * + - **(장고에서 mysql 연결한 사진)**
2. **MySQL Workbench 설정**

- AWS RDS에 연결



- Connection Name: ASDS

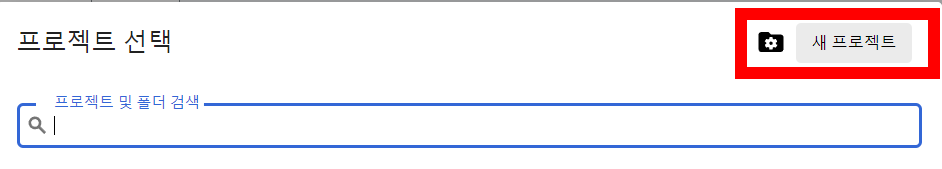
- Hostname: AWS RDS URL

- Username: 사용자 계정

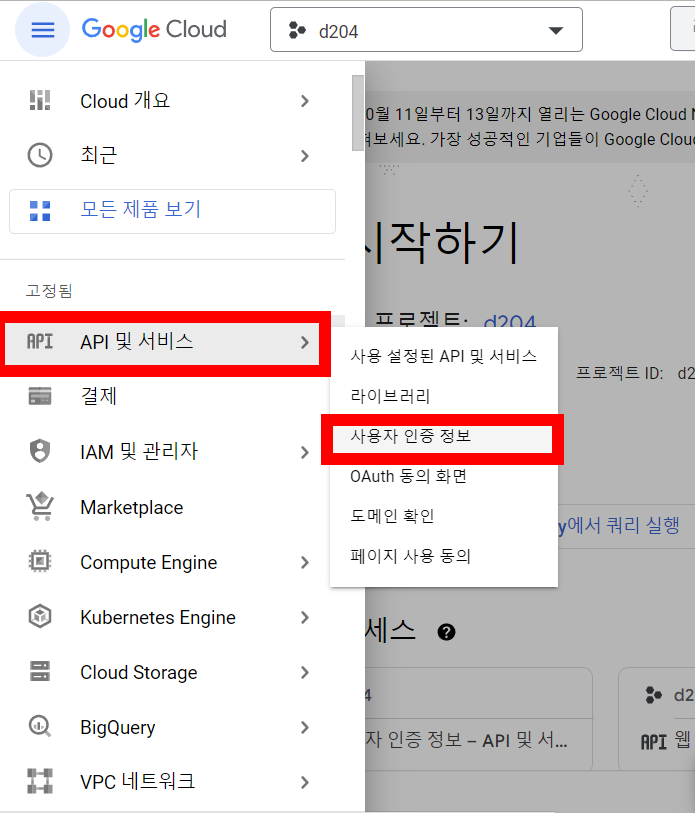
- Password: 사용자 비밀번호

**Ⅵ. Google console 설정**

1. [**https://console.cloud.google.com/**](https://console.cloud.google.com/) **접속 및 로그인**
2. **프로젝트 생성**



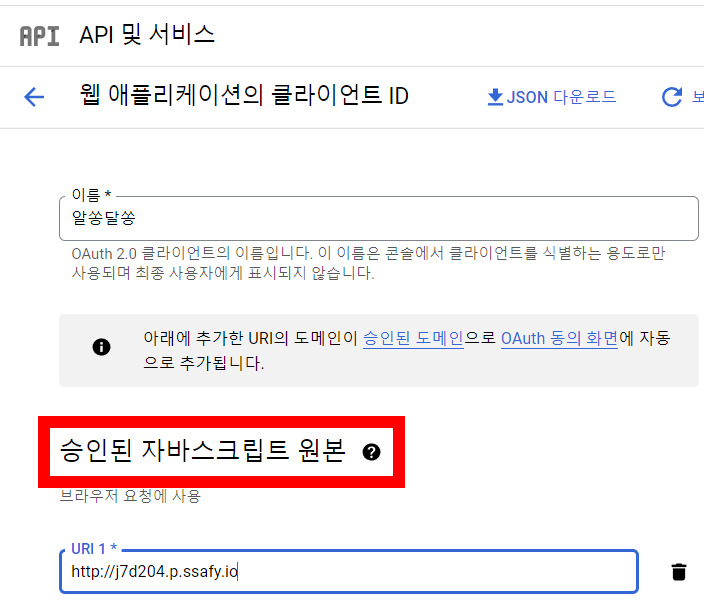
1. **사용자 인증 정보 만들기**



1. **Oauth 클라이언트 ID 생성**

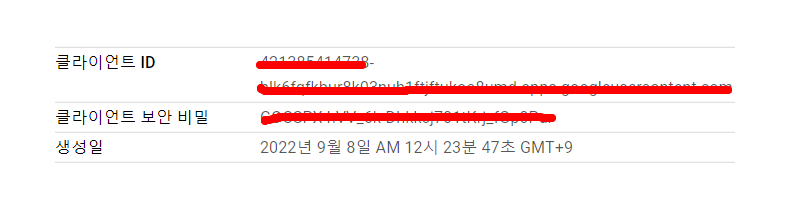


1. **동의 화면 구성**
2. **Oauth 클라이언트 ID 설정**



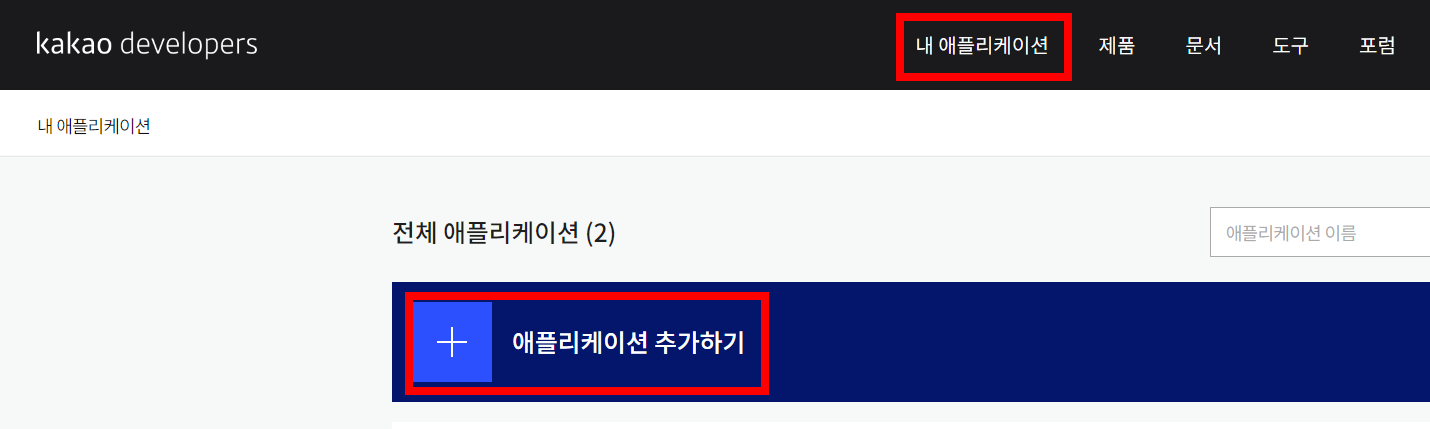


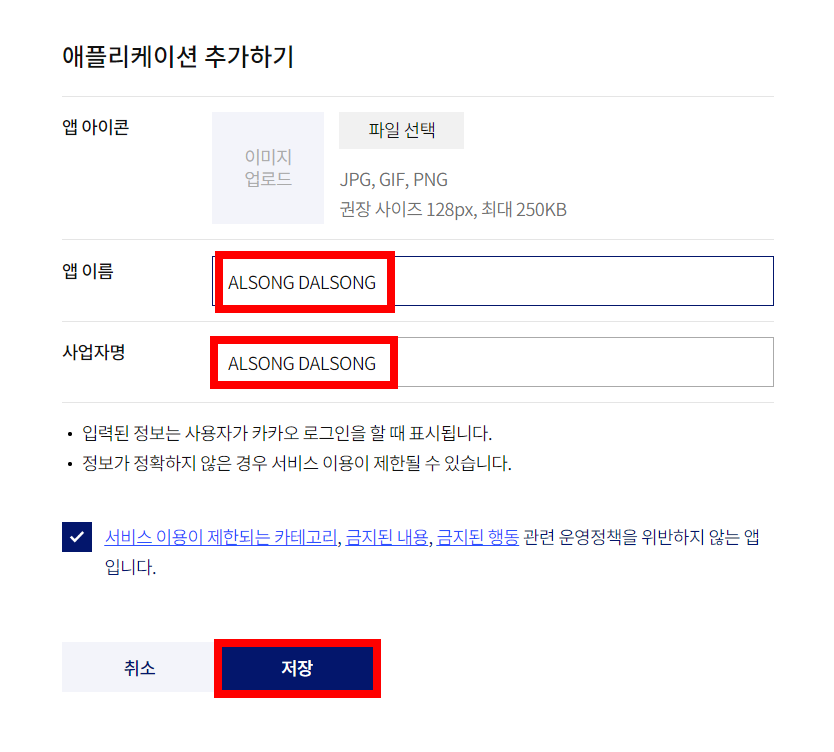
1. **클라이언트 ID & Secret key 발급**



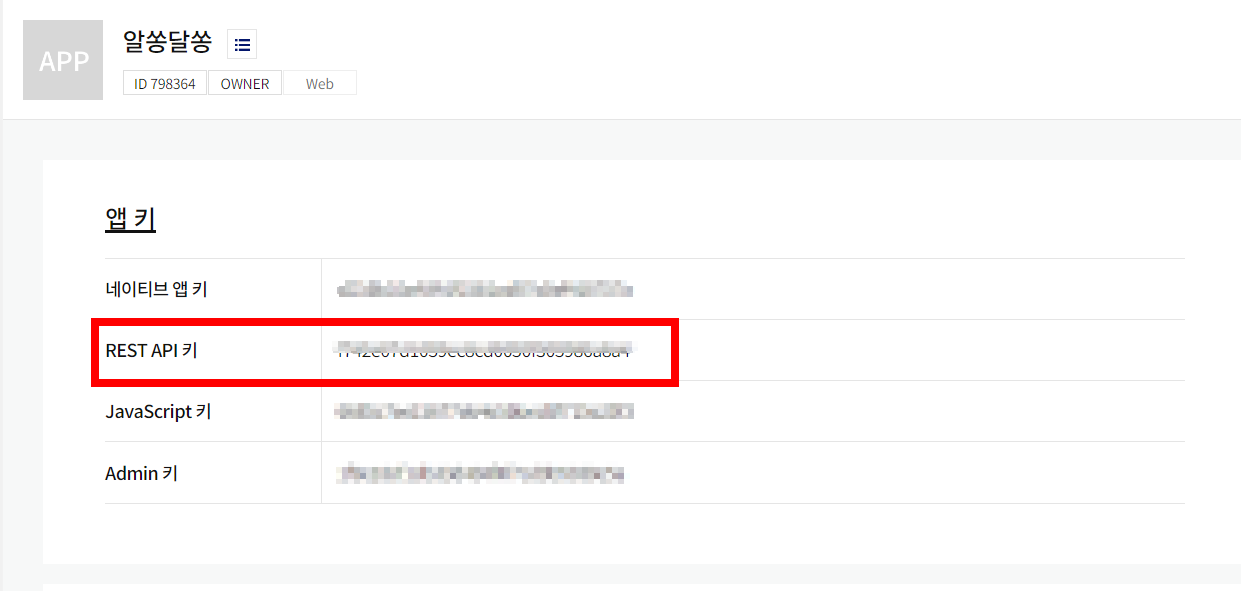
**Ⅵ. Kakao console 설정**

1. [**https://developers.kakao.com/**](https://developers.kakao.com/) **접속 및 로그인**
2. **내 애플리케이션 > 애플리케이션 추가하기**





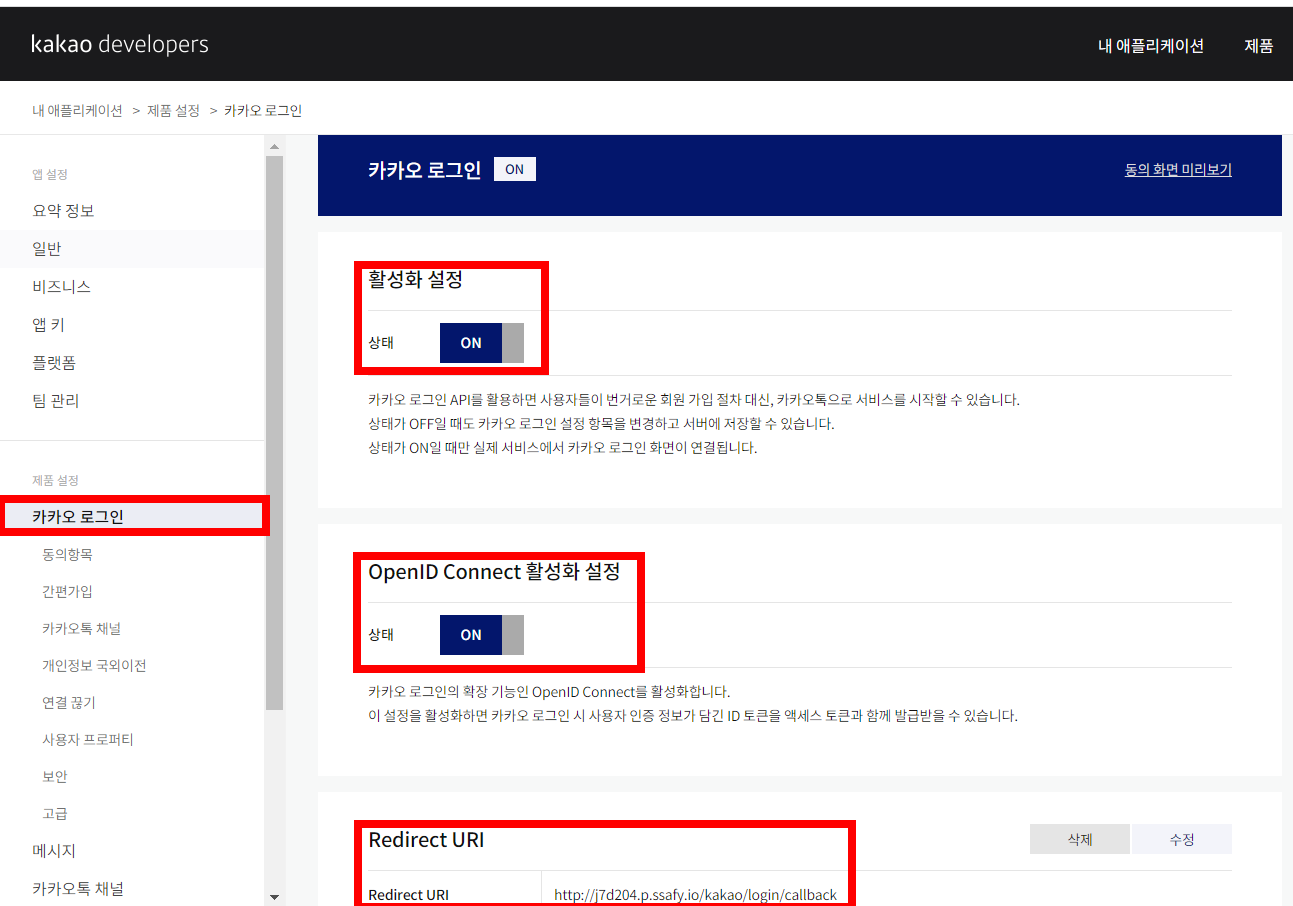
1. **REST API 키 발급**



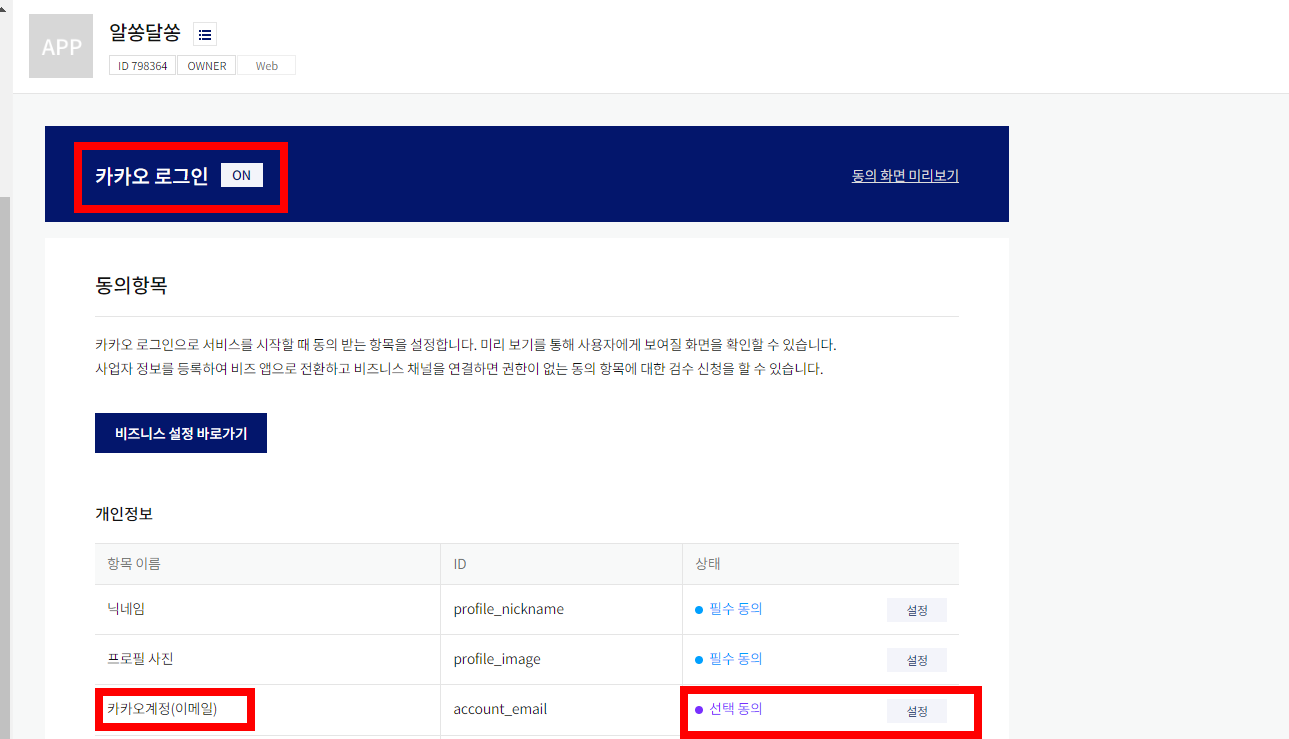
1. **앱 설정 > 플랫폼 > Web > 도메인 등록**



1. **제품 설정 > 카카오 로그인 활성화 & Redirect URI 설정**

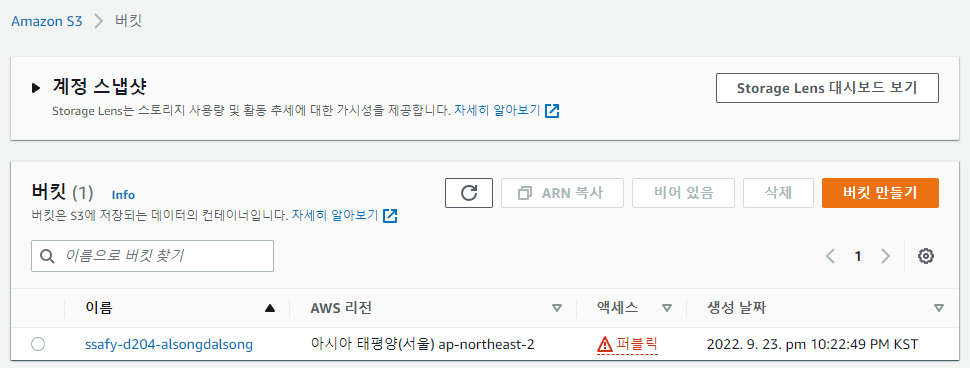


1. 카카오 로그인 동의 항목 설정



**Ⅶ. AWS S3 사용 방법**

1. AWS 로그인 후 <https://s3.console.aws.amazon.com/s3/> 접속
2. 버킷 만들기



* 1. AWS region을 서울로 지정한다.

텍스트이(가) 표시된 사진

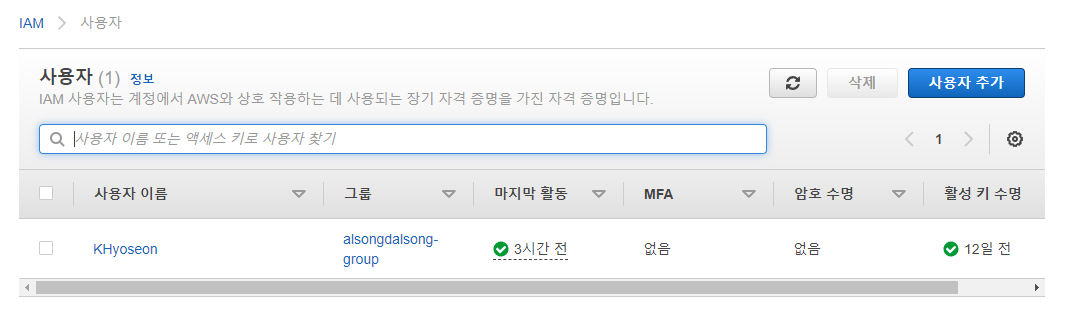
자동 생성된 설명

1. 퍼블릭 액세스 차단 설정

텍스트이(가) 표시된 사진

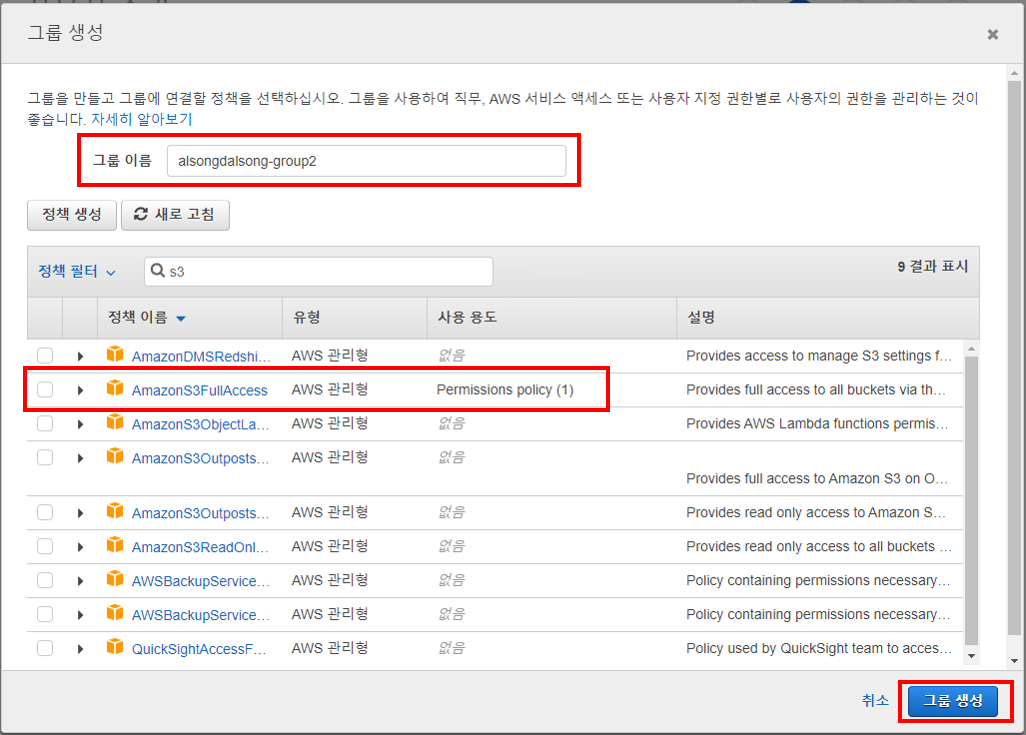
자동 생성된 설명

1. 권한 설정을 위해 <https://us-east-1.console.aws.amazon.com/iamv2> 접속
2. 사용자 추가



텍스트이(가) 표시된 사진

자동 생성된 설명



텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

위 과정에서 얻은 액세스 키와 비밀 액세스 키는 반드시 csv로 다운받거나 따로 보관을 해 두어야 한다.

1. 버킷 정책 설정

텍스트이(가) 표시된 사진

자동 생성된 설명

(첫번째 statement: KHyoseon이라는 사용자에게 모든 권한을 허용, 두번째 statement: 모든 사용자에게 객체 삭제와 조회 권한 허용)

버킷과 사용자 arn 주소를 맞게 입력한다.

1. 프로젝트 가상환경에 boto3 설치

Pip install boto3

1. Secrets.json 생성

텍스트이(가) 표시된 사진

자동 생성된 설명

1. Settings.py에 필수 변수 지정

텍스트이(가) 표시된 사진

자동 생성된 설명

1. S3 사용하는 앱에 storages.py 파일 추가

import boto3

import uuid

from django.conf import settings

class FileUpload:

    def \_\_init\_\_(self, client):

        self.client = client

    def upload(self, file):

        return self.client.upload(file)

    def delete(self, image\_id):

        return self.client.delete(image\_id)

class MyS3Client:

    def \_\_init\_\_(self, access\_key, secret\_key, bucket\_name):

        boto3\_s3 = boto3.client(

            's3',

            aws\_access\_key\_id = access\_key,

            aws\_secret\_access\_key = secret\_key

        )

        self.s3\_client = boto3\_s3

        self.bucket\_name = bucket\_name

    def upload(self, file):

        try:

            file\_id = str(uuid.uuid4())

            extra\_args = { 'ContentType' : file.content\_type }

            self.s3\_client.upload\_fileobj(

                    file,

                    self.bucket\_name,

                    file\_id,

                    ExtraArgs = extra\_args

                )

            return settings.AWS\_S3\_CUSTOM\_DOMAIN + "/" + file\_id

        except:

            return None

    def delete(self, image\_id):

      try:

        self.s3\_client.delete\_object(Bucket=self.bucket\_name, Key=image\_id)

        return "SUCCESS"

      except:

        return "FAIL"

# MyS3Client instance

s3\_client = MyS3Client(settings.AWS\_ACCESS\_KEY\_ID, settings.AWS\_SECRET\_ACCESS\_KEY, settings.AWS\_STORAGE\_BUCKET\_NAME)

1. Views.py 사용

from .storages import FileUpload, s3\_client

class ImageDetail(GenericAPIView):

    serializer\_class = ImageSerializer

    renderer\_classes = (renderers.JSONRenderer,)

    def post(self, request, format=None):

        file = request.FILES['image']

        profile\_image\_url = FileUpload(s3\_client).upload(file)

        if profile\_image\_url != None:

            return Response(profile\_image\_url, status=status.HTTP\_200\_OK)

        return Response(status=status.HTTP\_500\_INTERNAL\_SERVER\_ERROR)

    @swagger\_auto\_schema(request\_body=ImageSerializer)

    def delete(self, request, format=None):

        image\_url = request.data['image\_url']

        image\_id = image\_url.split('com/')[1]

        ret = FileUpload(s3\_client).delete(image\_id)

        if ret=="SUCCESS":

            return Response({'result': ret},status=status.HTTP\_204\_NO\_CONTENT)

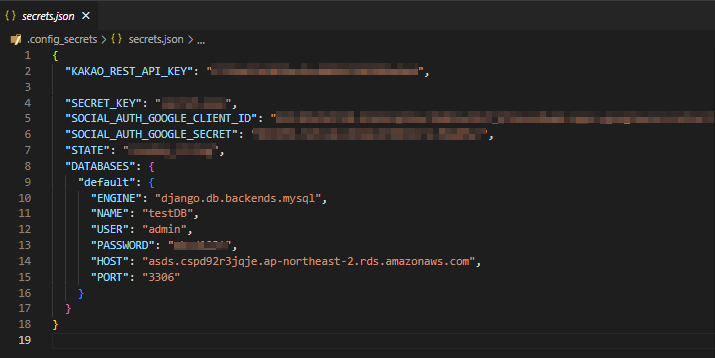
        return Response({'result': ret}, status=status.HTTP\_500\_INTERNAL\_SERVER\_ERROR)

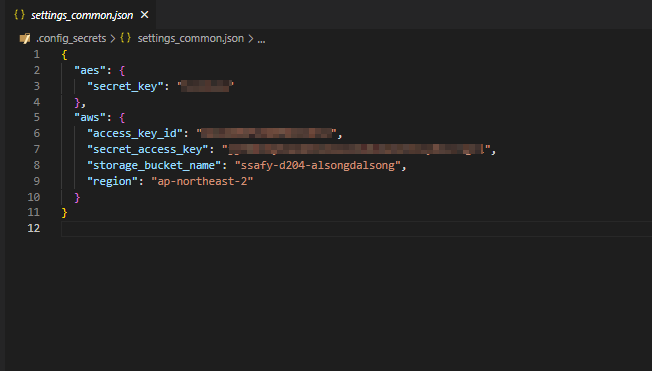
**Ⅷ. 환경 변수 설정**

* **Backend**

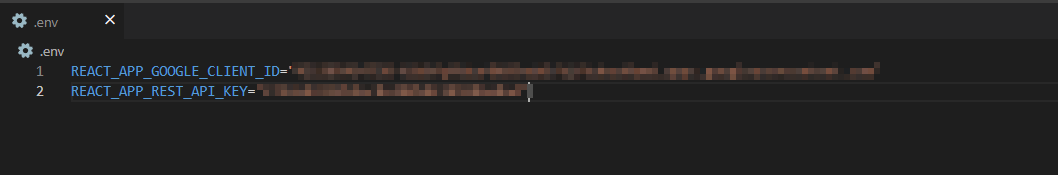
**# backend/.config\_secrets/**

* + - * **secrets.json**



* + - * **settings\_common.json**
* **FrontEnd**

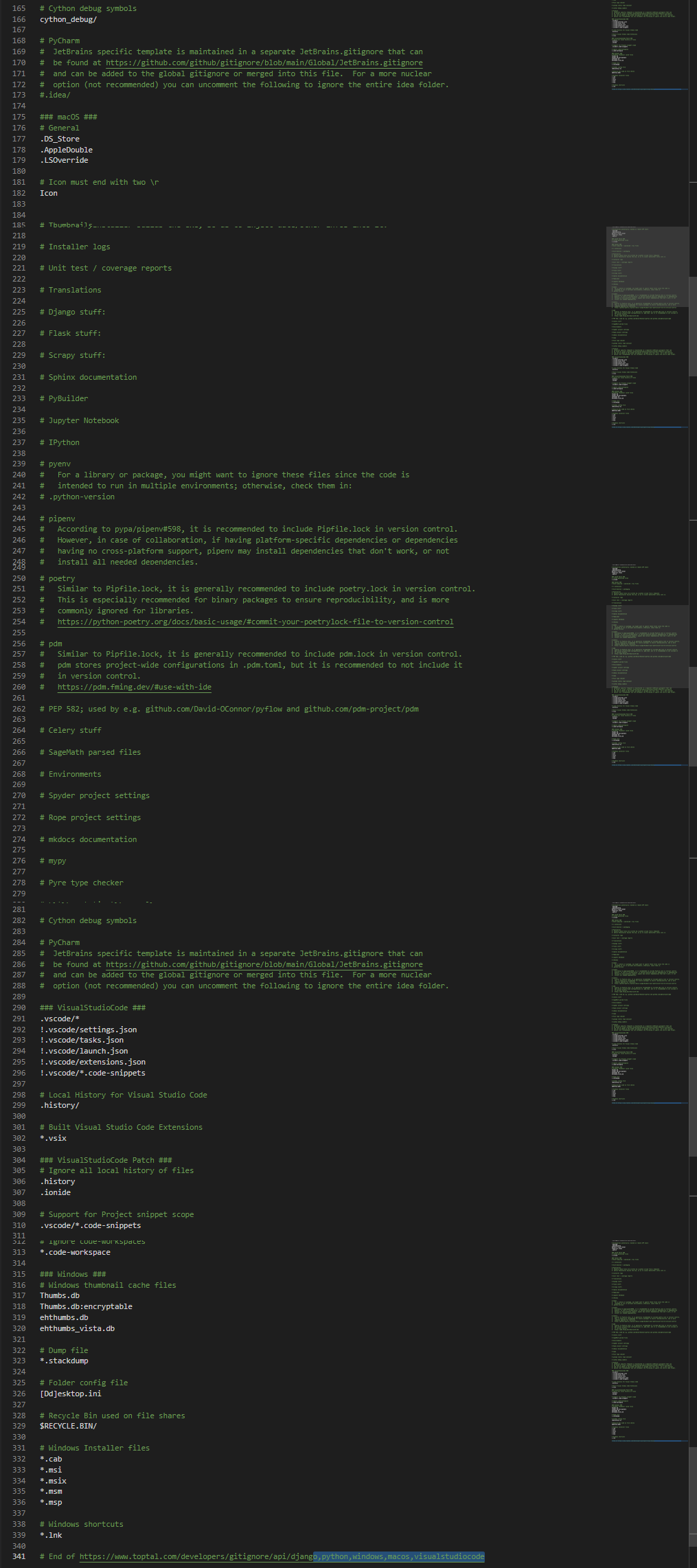
**# music-diary/.env**



**Ⅸ. gitignore**

* **BackEnd**





* **FrontEnd**

