Bloomer

감정을 꽃 피우다

포팅 매뉴얼

포팅 매뉴얼

개발환경

1. Front-End
    a. Visual Studio Code 1.75.1
    b. React 18.2.0
        - Redux 8.0.5
        - styled-components 5.3.6
        - axios 1.2.3
    c. React-Native 0.71.3
        a. react-native-webview 11.26.0
    d. Android
    e. Modeling
        a. Blender
        b. three.js

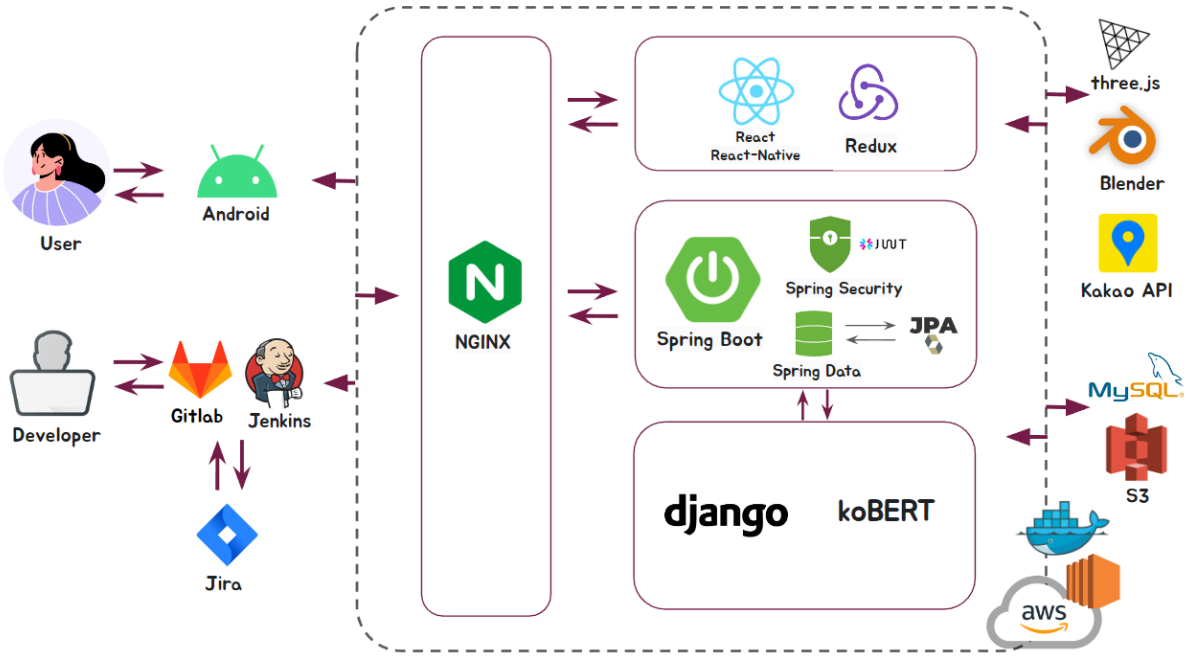
2. Back-End
    a. IntelliJ IDEA 2022.3.1
    b. SpringBoot Gradle 2.7.9
        - JAVA 11.0.16.14
        - Spring Security
        - Spring Data JPA
        - JWT 0.11.5
        - QueryDSL
        - HikariCP 4.0.3
    c. Test
        a. JUnit5
        b. Mockito
        c. Jacoco toolVersion0.8.7
        d. JMeter

3. DataBase
    a. MySQL 8.0.31

4. Domain
    a. Django 3.2.18
    b. koBERT

5. CI/CD
    a. AWS EC2
        - Ubuntu 20.04
        - Docker 20.10.23
        - Nginx 1.18.0
        - Jenkins 2.378.1

6. 공통
    - Gitlab
    - Jira
    - Mattermost, Notion
    - Postman 10.9.4



EC2 세팅

1. Docker

1-1. Docker 설치

apt-get 업데이트, 관련 패키지 설치

```
sudo apt-get update
sudo apt-get install \
    ca-certificates \
    curl \
    gnupg \
    lsb-release
```

Docker 공식 GPG-Key 추가

```
sudo mkdir -m 0755 -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
```

Repostory 설정

```
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

JDK 설치

```
$ sudo apt-get update
$ sudo apt-get install openjdk-8-jdk
```

2. Jenkins

2-1. Jenkins container 설치

```
docker run -d -p 8999:8080 -p 50500:50000 -v jenkins-data:/var/jenkins_home -v /var/run/docker.sock:/var/run/docker.sock -u root --nam
```

2-2. Jenkins 컨테이너 내부에 Docker 설치

```
sudo docker exec -u 0 -it jenkins bash

# 패키지 업데이트
apt-get update

# sudo 패키지 설치
apt-get install sudo

# Docker 설치
sudo apt-get install docker.io
```

2-3. Jenkins 플러그인 설치

Jenkins 관리 → 플러그인 관리 → Available plugins → **NodeJs Plugin 설치 & GitLab Plugin 설치**

2-4. GitLab 연결 설정

Jenkins 관리 → 시스템 설정

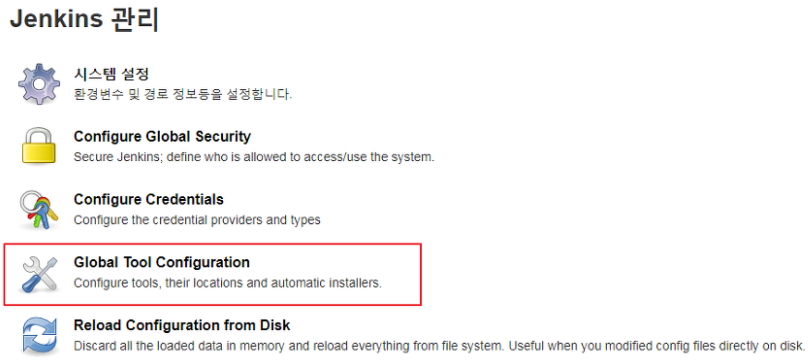
Gitlab 항목에서 **Enable authentication for '/project' end-point** 체크

- Connection name : gitlab-connection
- Gitlab host URL : https://lab.ssafy.com
- Credentials
  - +ADD 클릭하여 Jenkins 클릭
    - Kind : GitLab API Token
    - Scope : Global
    - API token :
    - ID : gitlab-access-token

- 드롭박스에서 GitLab API token 선택하여 적용

2-5. Node 설치

Jenkins 관리 → Global Tool Configuration



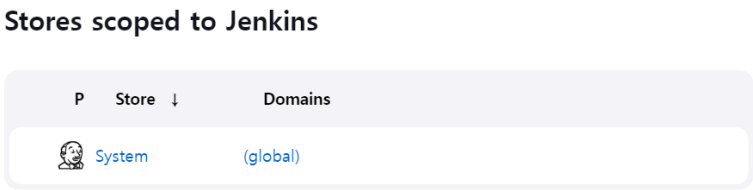
NodeJS 항목에서 **NodeJS installations** 클릭

- name : 18.12.1-LTS
- version : NodeJs 18.12.1

2-6. Username with password Credentials 추가

Jenkins 관리 → Manage Credentials

Domains의 (global) 클릭하여 +Add Credentials



- Kind : Username with password
- Scope : Global
- Username :
- Password :
- ID : gitlab-access-account

2-7. Item 생성

Enter an item name :

Pipeline 으로 생성

Configure 항목 클릭

- Git Repository와 연결 설정
  - GitLab Connetion 설정
    - gitlab-connection 선택
    - Use alternative credential 체크
      - Credential을 GitLab API token 선택
  - Build Trigger
    - **Build when a change is pushed to GitLab. GitLab webhook URL: http://j8a205.p.ssafy.io:8999/project/blommer** 체크
    - 고급 버튼 클릭
      - 하단의 Secret token 부분에서 Generate 버튼 클릭하여 토큰 생성
    - webhook URL 과 Secret token은 GitLab에서 Webhook 설정 시 필요
    - 매개변수 설정해서 .env 파일 shell script를 생성

2-8. 스크립트 작성

Pipeline

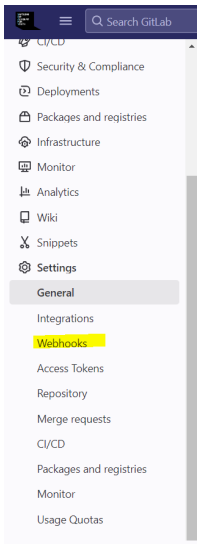
```
pipeline {
   agent any

   tools {nodejs "18.12.1-LTS"}

   stages {
       stage('Gitlab') {
           steps {
               git branch: 'main', credentialsId: 'gitlab-access-account', url: 'https://lab.ssafy.com/s08-bigdata-recom-sub2/S08P22A2
           }
       }
       stage('SpringBootBuild') {
           steps {
               dir('back-end') {
                   sh "./gradlew bootJar"
               }
           }
       }
       stage('ReactBuild') {
           steps {
               dir('front-end') {
                   sh "npm install --legacy-peer-deps"
               }
           }
       }
       stage('Build') {
            steps {
                sh 'docker build -t dolearn-front ./front-end/'
                sh 'docker build -t dolearn-back ./back-end/'
            }
        }
        stage("Stop and Remove Front Container") {
            steps {
                script {
                    def result = sh(returnStdout: true, script: "docker ps -q --filter name=front")
                    if (result.trim().length() > 0) {
                        sh "docker stop front"
                        sh "docker rm front"
                        echo "Container named 'front' has been stopped and removed."
```

```
                    } else {
                        echo "No container named 'front' was found."
                    }
                }
            }
        }
        stage("Stop and Remove Back Container") {
            steps {
                script {
                    def result = sh(returnStdout: true, script: "docker ps -q --filter name=back")
                    if (result.trim().length() > 0) {
                        sh "docker stop back"
                        sh "docker rm back"
                        echo "Container named 'back' has been stopped and removed."
                    } else {
                        echo "No container named 'back' was found."
                    }
                }
            }
        }
       stage('Finish') {
            steps{
                sh 'docker images -qf dangling=true | xargs -I{} docker rmi {}'
            }
        }
    }
}
```

2-9. Webhook 설정

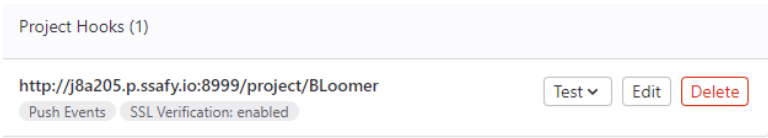
gitlab 접속하여 **S08P22A205** 프로젝트 클릭

`Settings` → `Webhook` 클릭

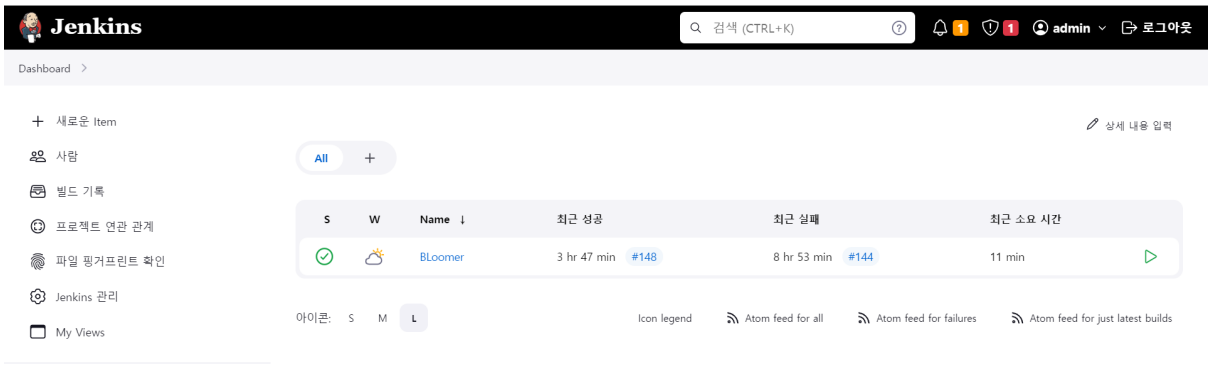


- `URL` : http://j8a205.p.ssafy.io:8999/project/Blommer
- `Secret token` : Configure에서 generate한 토큰
- `Trigger`
  - `Push events` : main

누르고 하단에 생성된 Webhook 확인



2-10. Jenkins에서 빌드테스트



3. 도메인

Dockerfile

```
FROM python:3.7.9
WORKDIR /usr/src/app

## Install packages
COPY requirements.txt ./
RUN pip install -r requirements.txt
RUN pip install gluonnlp pandas tqdm
RUN pip install pymysql
RUN pip install Django==3.2.18
RUN pip install scikit-learn
RUN pip install openpyxl
# copy all file
COPY . .

## Copy KoBERTall src files
COPY ./KoBERT/kobert/ /usr/local/lib/python3.7/site-packages/kobert
COPY ./emotions/music_vector.csv /usr/src/app/emotions
COPY ./emotions/tag_list.xlsx /usr/src/app/emotions
COPY ./emotions/tag_music.xlsx /usr/src/app/emotions
## Run the application on the port 8000
EXPOSE 8000

# gunicorn 배포 명령어
# CMD ["gunicorn", "--bind", "허용하는 IP:열어줄 포트", "project.wsgi:application"]
CMD ["python", "manage.py", "runserver", "0.0.0.0:8000"]
```

koBERT로 학습하고 추출된 pickle파일을 home directory밑에 train 디렉토리에 넣어놓고 docker로 실행시킬 때 바인딩 함.(코드는 docker에서 실행됨)

4. MYSQL

4-1. MYSQL 설치

도커 이미지 다운로드

```
docker pull mysql
```

컨테이너 생성

```
docker run --network=host --name mysql -e MYSQL_ROOT_PASSWORD=root -d -p 3306:3306 mysql:latest
```

4-2. MYSQL 환경설정

Bash 실행

```
docker exec -it mysql bash
```

Admin 접속

```
mysql -u root -p
// 입력 후 컨테이너 실행 시 사용했던 Password 입력
```

유저 생성 (예시 : ssafy)

```
# USER 생성, '%'는 모든 IP에서 접속 가능
mysql> CREATE USER ssafy@'%' identified by {비밀번호};
# 생성한 USER에 모든 권한 부여
mysql> GRANT ALL PRIVILEGES ON *.* to ssafy@'%';
# 변경 사항 적용
mysql> FLUSH PRIVILEGES;
mysql> exit;
```

ssafy 유저로 접속

```
mysql -u ssafy -p
// Enter password: {비밀번호}
```

Bloomer DB 생성

```
CREATE DATABASE bloomer;
SHOW DATABASES; // 확인
```

application.properties설정

```
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://j8a205.p.ssafy.io:3301/bloommer?useSSL=false&useUnicode=true&serverTimezone=Asia/Seoul
spring.datasource.username=
spring.datasource.password=
#spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update
#spring.jpa.properties.hibernate.format_sql=true
```

Nginx Default값

nginx.conf

```
user www-data;
worker_processes auto;
pid /run/nginx.pid;
include /etc/nginx/modules-enabled/*.conf;

events {
        worker_connections 768;
        # multi_accept on;
}

http {

        ##
        # Basic Settings
        ##

        sendfile on;
        tcp_nopush on;
        tcp_nodelay on;
        keepalive_timeout 65;
        types_hash_max_size 2048;
        # server_tokens off;

        # server_names_hash_bucket_size 64;
```

```
        # server_name_in_redirect off;

        include /etc/nginx/mime.types;
        default_type application/octet-stream;

        ##
        # SSL Settings
        ##

        ssl_protocols TLSv1 TLSv1.1 TLSv1.2 TLSv1.3; # Dropping SSLv3, ref: POODLE
        ssl_prefer_server_ciphers on;

        ##
        # Logging Settings
        ##

        access_log /var/log/nginx/access.log;
        error_log /var/log/nginx/error.log;

        ##
        # Gzip Settings
        ##

        gzip on;

        # gzip_vary on;
        # gzip_proxied any;
        # gzip_comp_level 6;
        # gzip_buffers 16 8k;
        # gzip_http_version 1.1;
        # gzip_types text/plain text/css application/json application/javascript text/xml application/xml application/xml+rss text/jav

        ##
        # Virtual Host Configs
        ##

        include /etc/nginx/conf.d/*.conf;
        include /etc/nginx/sites-enabled/*;

                #ssl_certificate /etc/letsencrypt/live/j8a205.p.ssafy.io/fullchain.pem; # managed by Cert>
                #ssl_certificate_key /etc/letsencrypt/live/j8a205.p.ssafy.io/privkey.pem; # managed by Ce>
                #include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Ce
                #ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Cert


}
```

빌드 및 배포

1. FrontEnd

.env 파일 추가

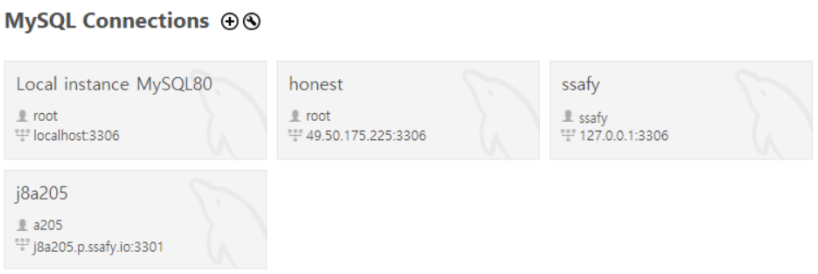
```
REACT_APP_KAKAO_API_KEY=
REACT_APP_WEATHER_API_KEY=
```

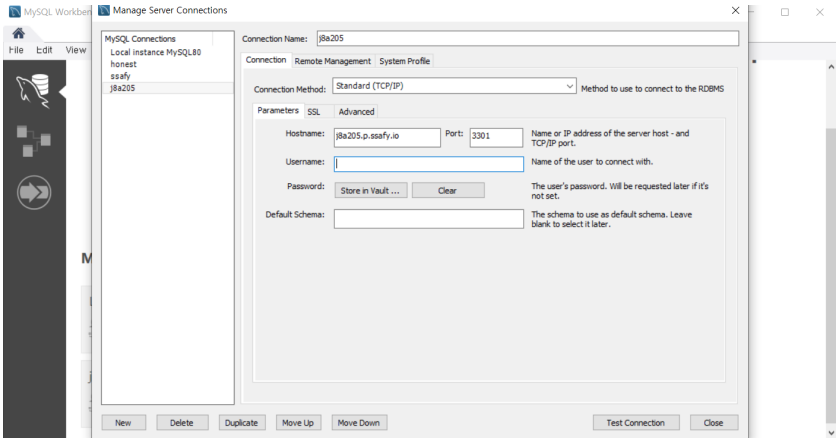
MYSQL WorkBench 사용방법

MySQL 8.0.32 설치 진행

https://dev.mysql.com/downloads/installer/

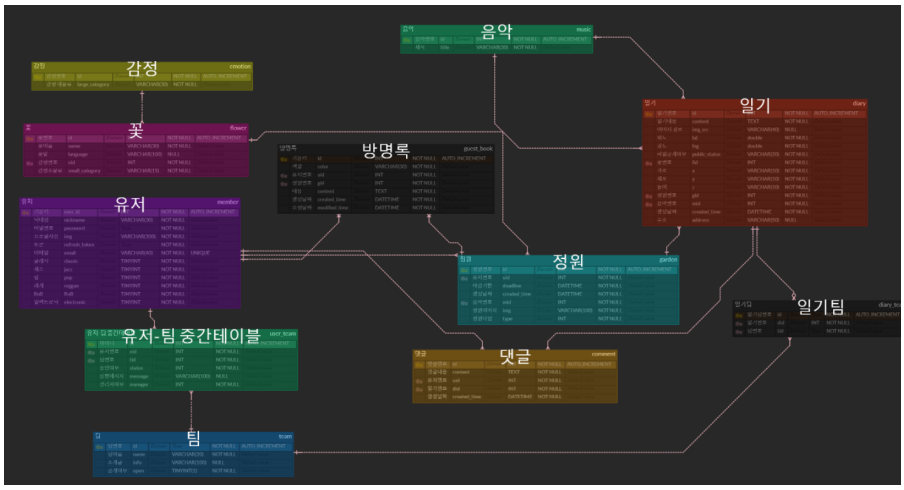
Workbench에서 Connection 생성





i8a205.p.ssafy.io:3301

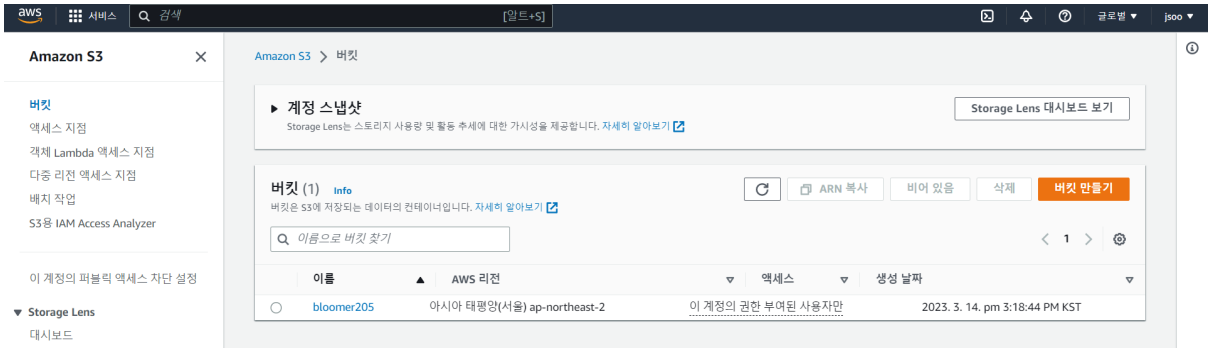
→ 계정 입력 완료 후 Test Connection에 성공하면 연결 성공



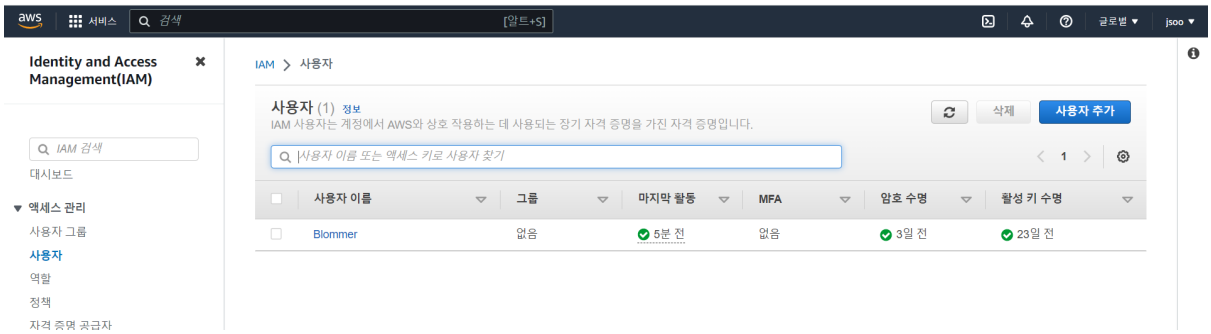
외부 서비스

1. AWS S3 Bucket

계정 생성 및 bucket 추가



보안 자격 증명 > 엑세스키 생성



BackEnd - application.properties 추가

```
cloud.aws.credentials.access-key=
cloud.aws.credentials.secret-key=
cloud.aws.s3.bucket=
cloud.aws.stack.auto=false
logging.level.com.amazonaws.util.EC2MetadataUtils=error

spring.servlet.multipart.maxFileSize=10MB
spring.servlet.multipart.maxRequestSize=10MB
```

Front : .env 추가

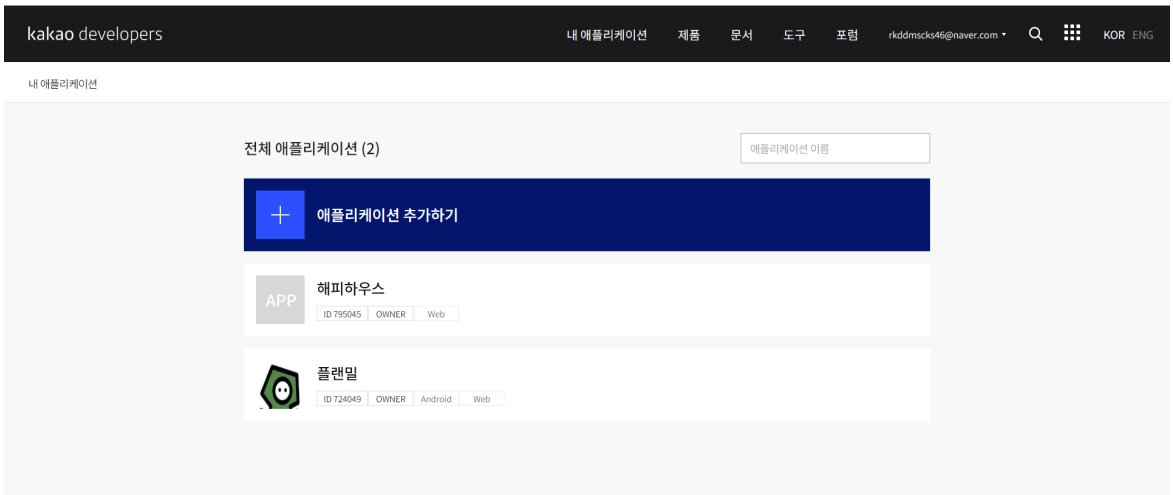
```
REACT_APP_S3_ACCESS_KEY_ID=
REACT_APP_S3_SECRET_ACCESS_KEY=
REACT_APP_S3_REGION=
```

2. Google 소셜 로그인

```
spring.security.oauth2.client.registration.google.client-id=
spring.security.oauth2.client.registration.google.client-secret=
spring.security.oauth2.client.registration.google.scope=profile,email
```

3. Kakao

- 애플리케이션 추가




```
spring.security.oauth2.client.registration.kakao.client-id=
spring.security.oauth2.client.registration.kakao.redirect-uri=
spring.security.oauth2.client.registration.kakao.client-authentication-method=POST
spring.security.oauth2.client.registration.kakao.authorization-grant-type=authorization_code
spring.security.oauth2.client.registration.kakao.scope=account_email,profile_nickname
spring.security.oauth2.client.registration.kakao.client-name=Kakao

spring.security.oauth2.client.provider.kakao.authorization-uri=https://kauth.kakao.com/oauth/authorize
spring.security.oauth2.client.provider.kakao.token-uri=https://kauth.kakao.com/oauth/token
spring.security.oauth2.client.provider.kakao.user-info-uri=https://kapi.kakao.com/v2/user/me
spring.security.oauth2.client.provider.kakao.user-name-attribute=id
```