

# 포팅 메뉴얼

## ▼ 1. 프로젝트 기술스택

- 1) 이슈관리 : Jira
- 2) 형상관리 : Git
- 3) 커뮤니케이션 : Mattermost, Webex, notion, Google Sheets, Figma,
- 4) 개발환경 :

✓ OS : Window 10

✓ IDE :

- IntelliJ
- VSCode
- HeidiSQL
- VIM

✓ Database :

- DBMS: Mariadb 10.3.38
- SearchEngine: ElasticSearch 8.6.2

✓ Server: AWS EC2

✓ OS: Ubuntu 20.04 LTS (Focal Fossa)

✓ File Server: AWS S3

✓ CI/CD: Jenkins, Docker, Nginx

### 5) 상세 기술

✓ Front-End

- axios : 1.3.4
- chart.js : 4.2.1
- css-loader : 6.7.3
- eslint : 8.36.0
- eslint-config-airbnb : 19.0.4
- eslint-config-airbnb-typescript : 17.0.0
- eslint-config-prettier : 8.7.0
- eslint-plugin-prettier : 4.2.1
- lint : 0.8.19
- lottie-react : 2.4.0
- node-sass : 8.0.0
- prettier : 2.8.5
- react : 18.2.0
- react-chartjs-2 : 5.2.0
- react-dom : 18.2.0
- react-icons : 4.8.0
- react-kakao-maps-sdk : 1.1.7
- react-redux : 8.0.5
- react-router-dom : 6.9.0
- react-scripts : 5.0.1
- react-slick : 0.29.0
- redux : 4.2.1
- redux-saga : 1.2.3
- sass-loader : 13.2.0

- slick-carousel : 1.8.1
- style-loader : 3.3.1
- swiper : 9.2.0
- tailwindcss : 3.2.7
- typescript : 4.9.5
- v6 : 0.0.0
- web-vitals : 2.1.0 @

#### ✓ Back-End

- spring-boot-starter-web
- spring-boot-starter-security
- spring-boot-starter-oauth2-client
- spring-boot-starter-data-jpa
- mariadb-java-client
- lombok
- springfox-swagger-ui:3.0.0
- querydsl-jpa
- querydsl-apt
- json:20220320
- jjwt:0.9.1
- jaxb-runtime:2.3.2
- spring-cloud-starter-aws:2.0.1.RELEASE
- bom:2.15.0
- s3
- gson:2.8.6
- spring-boot-starter-data-elasticsearch:2.6.2
- okhttp:3.14.9

## ▼ 2. 서버 세팅

### 1) Jenkins 설치

```
apt-get update -y
apt-get upgrade -y
apt-get install ca-certificates curl software-properties-common \
apt-transport-https gnupg lsb-release docker.io nginx docker-compose -y
docker pull jenkins/jenkins:lts
sudo docker run -d -p 8080:8080 -v /jenkins:/var/jenkins_home \
-v /var/run/docker.sock:/var/run/docker.sock \
-v /usr/local/bin/docker-compose:/usr/local/bin/docker-compose \
--name jenkins \
-u root jenkins/jenkins:lts \
```

### 2) 개발환경 버전에 맞게 설치 (nodejs, python)

```
docker exec -it jenkins bash
cd /home/
apt-get install libbz2-dev wget tar
wget https://www.python.org/ftp/python/3.9.10/Python-3.9.10.tgz
tar -xvzf Python-3.9.10.tgz
cd ./Python-3.9.10
./configure
make -j4
make install
```

node js 는 이 링크를 따라한다. <https://github.com/nodesource/distributions>

### 4) nginx

- nginx 설치 및 세팅

```
sudo apt-get install nginx -y
service start nginx
service status nginx
```

- nginx 설정 파일 찾기(위치가 버전 및 OS마다 조금씩 다름)

```
find / -name nginx.conf
```

- certbot 설치 (설치 도중, 이메일 입력 등을 통해 설정한다)

```
sudo snap install certbot --classic
certbot --nginx
```

- /etc/nginx/nginx.conf 변경하기

```
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: P00DLE
    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/j

    ##
    # Virtual Host Configs
    ##

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

    server {
        server_name sanneomeo.site www.sanneomeo.site j8a301.p.ssafy.io j8a301.p.ssafy.io;

        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forward-For $proxy_add_x_forwarded_for;
```

```

proxy_set_header X-Forward-Proto $scheme;

location / {
    proxy_pass http://localhost:3000/;
}

location /api/ {
    rewrite ^/api/(.*)$ /$1 break;
    proxy_pass http://localhost:9090/;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
}

location /recommend/ {
    rewrite ^/recommend/(.*)$ /$1 break;
    proxy_pass http://localhost:5000/;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
}

error_page 500 502 503 504 /50x.html;
location = 50x.html {
    root /usr/share/nginx/html;
}

listen 443 ssl; # managed by Certbot
ssl_certificate /etc/letsencrypt/live/sanneomeo.site/fullchain.pem; # managed by Certbot
ssl_certificate_key /etc/letsencrypt/live/sanneomeo.site/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 ($host = www.sanneomeo.site) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

    if ($host = sanneomeo.site) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

    if ($host = j8a301.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

    listen 80;
    server_name sanneomeo.site www.sanneomeo.site j8a301.p.ssafy.io;
    return 404; # managed by Certbot

}
}

```

- nginx 재실행(service nginx restart는 껏다 켜져서 사용중이면 불편을 끼침)

```

nginx -t
service nginx reload

```

### ▼ 3. 빌드 상세내용

#### 1) Spring DockerFile

```

FROM azul/zulu-openjdk:11
WORKDIR /spring
COPY ./build/libs/[프로젝트명]-0.0.1-SNAPSHOT.jar server.jar
ENTRYPOINT ["java", "-jar", "server.jar"]

```

#### 2) Flask DockerFile

```
FROM python:3.9
COPY . /flask
WORKDIR /flask
RUN pip3 install -r requirements.txt
RUN chmod 777 /flask/wsgi.py
CMD ["python3.9", "wsgi.py"]
```

### 3) React DockerFile

```
FROM node:18
WORKDIR /react
COPY . .
RUN npm install -g serve
CMD serve -s build
```

### 4) docker-compose.yml

```
version: "3"

services:
  spring:
    container_name: spring
    build: ./backend/sanneomeo
    ports:
      - "9090:9090"
    volumes:
      - /spring:/image
    restart: on-failure
  flask:
    container_name: flask
    build: ./backend/flask
    ports:
      - "5000:5000"
    volumes:
      - /flask:/image
    restart: on-failure
  react:
    container_name: react
    build: ./frontend/overmountain
    ports:
      - "3000:3000"
    volumes:
      - /react:/image
    restart: on-failure
```

### 5) Pipeline

```
pipeline {
  agent any
  stages {
    stage('Init') {
      steps {
        // catchError {
        //   deleteDir()
        // }
        sh "ls"
      }
    }
    stage('GitHub Repository Clone') {
      steps {
        git branch: 'develop', credentialsId: '9f75da46-3be4-40b6-964c-bfbd81c57fb0', url: 'https://lab.ssfy.com/s08-b
        sh "cp -rf /var/jenkins_home/application.yml ./backend/sanneomeo/src/main/resources/"
        sh "cp -rf /var/jenkins_home/DBInfo.py ./backend/flask/DBInfo/"
        sh "cp -rf /var/jenkins_home/env ./frontend/overmountain/.env"
      }
    }
    stage('Spring docker') {
      steps {
        dir("./backend/sanneomeo"){
          echo "Spring"
          sh "chmod +x gradlew"
          sh "./gradlew clean build --exclude-task test"
          sh "ls"
        }
      }
    }
  }
}
```

```

    }
    stage('Flask docker') {
    steps {
    dir("./backend/flask"){
        echo "Flask"
        sh "python3.9 -m pip install --disable-pip-version-check --upgrade pip"
        sh "rm requirements.txt"
        //sh "cp -rf /var/jenkins_home/requirements.txt ./"
        sh "pip3 freeze > requirements.txt"
        sh "pip3 install -r requirements.txt"
    }
    }
    }
    stage('React Docker') {
    steps {
    dir("./frontend/overmountain"){
        echo "React"
        sh "npm i"
        sh "npm run build --production"
    }
    }
    }
    stage('Docker compose down') {
    steps {
    catchError {
        echo 'docker compose down'
        sh 'docker-compose -f docker-compose.yml down'
    }
    }
    }
    stage('docker-compose') {
    steps {
    dir("./"){
        echo 'docker compose'
        sh "docker-compose -f docker-compose.yml up -d --build"
    }
    }
    }
    }
}

```

## ▼ 4. 외부 서비스

### 1) 날씨 API

<https://openweathermap.org/> 를 이용한다.

OpenWeather Weather in your city Guide API Dashboard Marketplace Pricing Maps Our Initiatives Partners Blog For Business Sign in Support

## Weather API Home / Weather API

Please, [sign up](#) to use our fast and easy-to-work weather APIs. As a start to use OpenWeather products, we recommend our [One Call API 3.0](#). For more functionality, please consider our products, which are included in [professional collections](#).

### One Call API 3.0 NEW

[API doc](#) [Subscribe](#)

Make one API call and receive all essential weather data in one response:

- Minute forecast for 1 hour
- Hourly forecast for 48 hours
- Daily forecast for 8 days
- Historical data for 40+ years back by timestamp
- National weather alerts

Read more about this API and subscription plan in the [FAQ](#).

**Pay as you call**

**1,000 API calls per day for free**  
**0.0012 GBP** per API call over the daily limit

[Subscribe to One Call by Call](#)

This is a separate subscription plan, which include only One Call API.

### Professional collections

For professionals and specialists with middle sized project, we recommend our Professional collections, which included [Current & Forecasts collection](#), [Historical weather data collection](#), [Weather Maps collection](#) and other APIs.

For Enterprise level projects we provide Enterprise license, which is included all forecast products and current state, along with alerts, maps, and other products. [Learn more](#)

You can read the [How to Start](#) guide and enjoy using our powerful weather APIs right now.

### Current & Forecast weather data collection

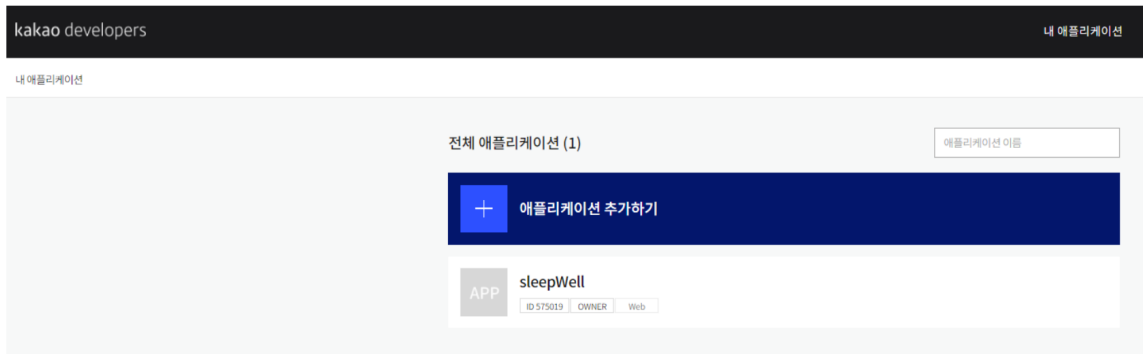
Current Weather Data

Hourly Forecast 4 days

Daily Forecast 16 days

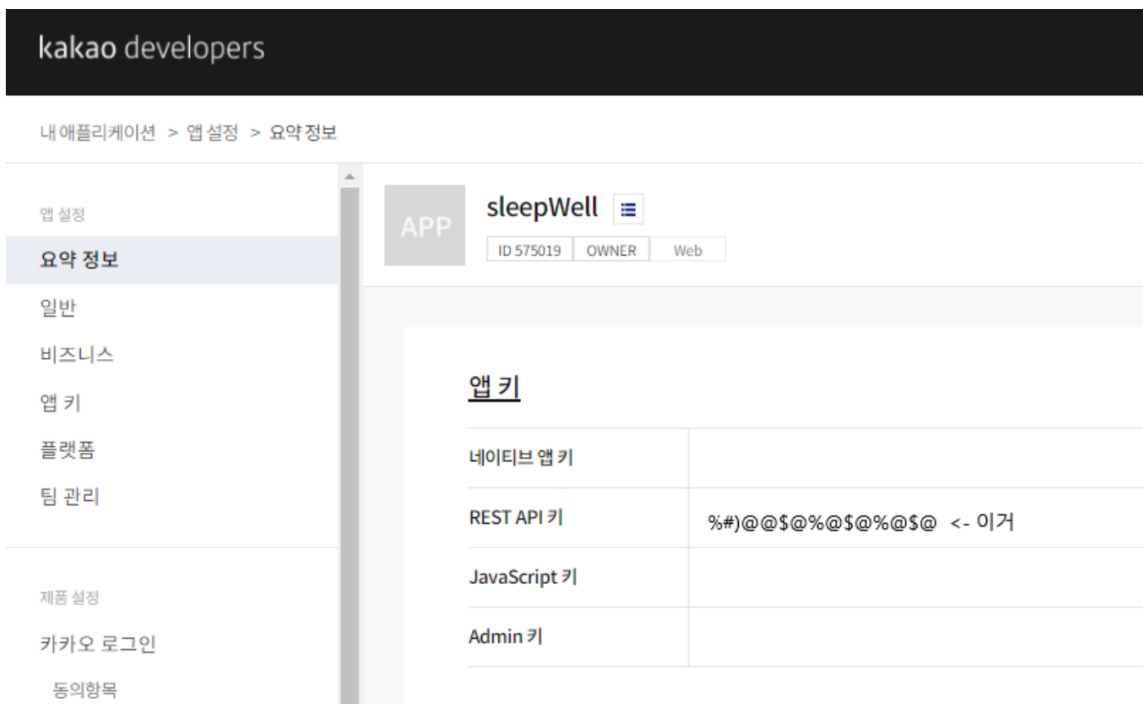
## 2) kakao API

### 애플리케이션



- cliend\_id와 redirect\_uri를 받아와서 {REST\_API\_KEY}와 {REDIRECT\_URI}에 채워주어야 한다.
- 두 가지를 얻으려면 우선 애플리케이션을 생성한다.

### Redirect Uri 추가



- cliend\_id는 kakao developers에서 내 애플리케이션을 추가했을 때 생기는 REST\_API 키를 넣어주면 되고, Redirect URI는 카카오 로그인 메뉴에 들어가서 추가를 해준다.

### Redirect URI

[삭제](#)[수정](#)

Redirect URI

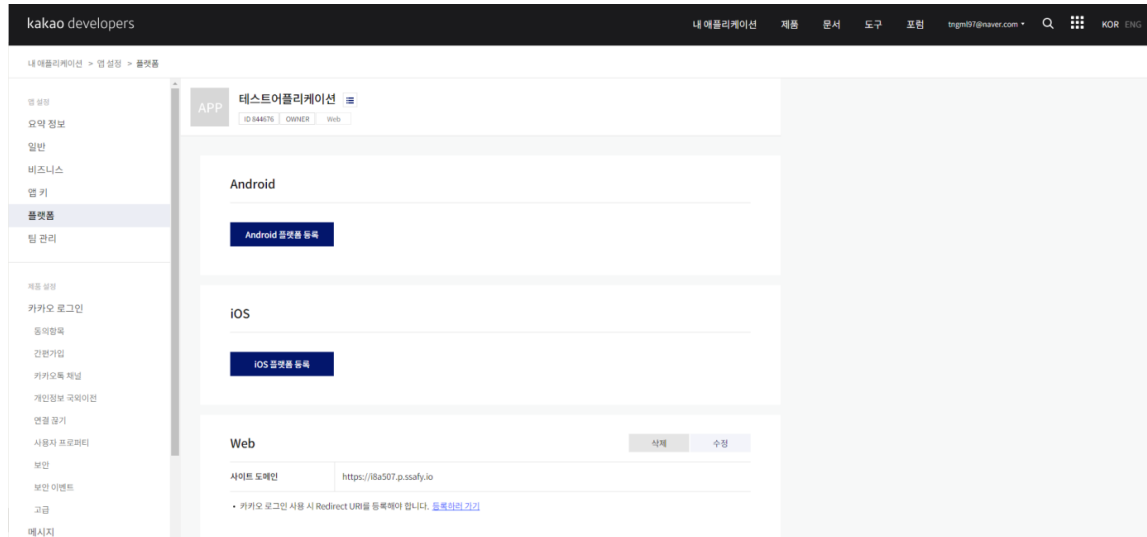
https://i8a507.p.ssafy.io  
https://i8a507.p.ssafy.io/oauth/kakao/callback  
http://localhost:3000/oauth/kakao/callback

- 카카오 로그인에서 사용할 OAuth Redirect URI를 설정합니다. (최대 10개)
- REST API로 개발하는 경우 필수로 설정해야 합니다.

- Redirect URI는 반드시 프론트에서 접근할 수 있는 Host로 지정해주어야 한다.

- 왜냐하면 여기에서 인가 코드 받고 넘기고 등등 모든 작업이 이루어져야 하는데 프론트엔드가 접근할 수 없는 Host로 지정을 해버리면 말 그대로 접근을 못하니 아무것도 할 수 없다.  
(localhost:8080 등... 대신 이건 백엔드에서 자체 테스트할 때 사용할 수 있다)

## 플랫폼 추가



- Web에서 사용할 것이기 때문에 Web 플랫폼에 사이트 도메인을 추가한다.