

**PLANTO**

**포팅 매뉴얼**

**목차**

**Ⅰ. 개요**

1. **개발환경 및 기술스택**
2. **외부 서비스**

**Ⅱ. 포팅 가이드**

1. **환경설정**
2. **EC2 설정**
3. **Docker, Docker Compose 설치**
4. **Nginx 설치 및 설정**
5. **빌드**

Ⅰ. 개요

1. 개발환경 및 기술스택
   1. **Tools**

* Notion
* Jira
* Git
* Intellij IDEA v2021.2.4
* VS CODE v1.74.2
* Gitlab
* PostMan
* Mattermost
* MySQL WorkBench 8.0CE
  1. **기술스택**

1. FrontEnd

* React v18.2.0
* Node.js v18.13.0
* JavaScript
* HTML / CSS

1. BackEnd

* Java v1.8
* SpringBoot v2.7.7
* Python v3.9.2
* Raspberrypi Pi 4 Model B Rev 1.2
* Debian GNU/Linux 11

1. DataBase

* MySQL v8.0.30
* FireBase

1. Server

* Docker v23.0.0
* Jenkins v2.375.2
* Nginx v1.18.0

1. **외부 서비스**

* Kakao Oauth (카카오톡 로그인 API)
* <https://developers.kakao.com/>
* 농사로 (식물 데이터 API)
* <https://www.nongsaro.go.kr/portal/ps/psz/psza/contentMain.ps?menuId=PS00191>
* 기상청 (날씨 데이터 API)
* <https://data.kma.go.kr/api/selectApiList.do?pgmNo=42>
* FireBase (식물 이미지 저장)
* https://firebase.google.com/?hl=ko
* Amazon EC2 (서버 배포)
* <https://aws.amazon.com>

Ⅱ. 포팅 가이드

1. **환경설정**

* **MySQL 설정**

|  |
| --- |
| backend/src/main/resources/application.yml |
| spring:  …  # mysql DB  datasource:  driver-class-name: com.mysql.cj.jdbc.Driver  <url:jdbc:mysql://>**{도메인주소}**/**{데이터베이스명}**?serverTimezone=Asia/Seoul  username: **{Id}**  password: **{Password}**  … |

|  |
| --- |
| backend/src/main/resources/env.properties |
| # MySQL  properties.mysql.url=**{URL}**  properties.mysql.username=**{유저네임}**  properties.mysql.password=**{비밀번호}**… |

* **FireBase 설정**

|  |
| --- |
| backend/src/main/resources/application.yml |
| app:  firebase-configuration-file: ./serviceAccountKey.json  firebase-bucket: **{주소}**  … |

|  |
| --- |
| backend/src/main/resources/serviceAccountKey.json |
| {  “type”: **{타입}**  “project\_id”: **{프로젝트 아이디}**  “private\_key\_id”: **{개인 키 아이디}**  “private\_key”: **{개인 키}**  “client\_email”: **{클라이언트 이메일}**  “client\_id”: **{클라이언트 아이디}**  “auth\_uri”: **{사용자 URI}**  “token\_uri”: **{토큰 URI}**  “auth\_provider\_x509\_cert\_url”: **{URL}**  “client\_x509\_cert\_url”: **{URL}**  } |

* **MQTT 설정**

|  |
| --- |
| backend/src/main/resources/env.properties |
| # Broker  java.mqtt.url=**{URL}**  java.mqtt.username=**{유저네임}**  java.mqtt.password=**{비밀번호}** |

* **OAUTH 설정**

|  |
| --- |
| Backend/src/main/resources/application.yml |
| spring:  security:  oauth2:  client:  registration:  kakao:  client-id: **{client-id : 카카오 REST API키}**  client-secret: **{client-secret키}**  redirect-uri: **{Redirect-uri 주소}**  authorization-grant-type: authorization\_code  client-name: Kakao  scope:  - profile\_nickname  - profile\_image  - account\_email  provider:  kakao:  authorization-uri: <https://kauth.kakao.com/oauth/authorize>  token-uri: <https://kauth.kakao.com/oauth/token>  user-info-uri: https://kakao.com/v2/user/me  … |

|  |
| --- |
| backend/src/main/resources/env.properties |
| # oauth  java.oauth.kakao.clientId = **{클라이언트 아이디}**  java.oauth.kakao.redirectUri = **{리다이렉트 URI}** |

* **기상청 api 설정**

|  |
| --- |
| backend/src/main/resources/env.properties |
| # weather  java.weather.secretKey= **{시크릿 키}** |

* **농사로 api 설정**

|  |
| --- |
| backend/src/main/resources/env.properties |
| # plant  java.plant.secretKey= **{시크릿 키}** |

* **Docker 설정**

|  |
| --- |
| docker-compose.yml |
| version: "3"  services:  mysql:  image: mysql:8.0.32  restart: always  ports:  - 3309:3306  cap\_add:  - SYS\_NICE  environment:  - MYSQL\_DATABASE=${MYSQL\_DATABASE}  - MYSQL\_ROOT\_PASSWORD=${MYSQL\_ROOT\_PASSWORD}  - MYSQL\_USER=${MYSQL\_USER}  - MYSQL\_PASSWORD=${MYSQL\_PASSWORD}  redis:  image: redis:alpine  container\_name: redis\_boot  hostname: redis\_boot  volumes:  - ./redis/data:/data  - ./redis/conf/redis.conf:/usr/local/conf/redis.conf  # 컨테이너에 docker label을 이용해서 메타데이터 추가  labels:  - "name=redis"  - "mode=standalone"  ports:  - 6379:6379  # 컨테이너 종료시 재시작 여부 설정  restart: always    backend:  build:  context: ./backend  dockerfile: Dockerfile  ports:  - 8080:8080  container\_name: spring-boot  depends\_on:  - mysql  - redis    front:  build:  context: ./frontend/planto  dockerfile: Dockerfile  container\_name: "nginx-react"  ports:  - 3000:3000    mosquitto:  restart: always  image: "eclipse-mosquitto"  ports:  - "1883:1883"  - "9001:9001"  volumes:  - ./eclipse-mosquitto/config/mosquitto.conf:/mosquitto/config/mosquitto.conf  - ./eclipse-mosquitto/data:/mosquitto/data  - ./eclipse-mosquitto/log:/mosquitto/log  my-jenkins:  image: jenkins/jenkins:lts  container\_name: "my-jenkins"  ports:  - 9090:8080  - 50000:50000  volumes:  - /home/opendocs/jenkins:/var/jenkins\_home  - /var/run/docker.sock:/var/run/docker.sock  user: root  nginx:  image: nginx:1.21.5-alpine  ports:  - "80:80"  volumes:  - ./nginx/nginx.conf:/etc/nginx/nginx.conf  container\_name: myweb-proxy  depends\_on:  - backend  - front |

1. **EC2 설정**

**초기 설정**

|  |
| --- |
| sudo apt update  sudo apt upgrade  sudo apt install build-essential |

**Java 설치**

|  |
| --- |
| # 설치  sudo apt-get install openjdk-8-jdk  # 버전확인  java -version |

**Timezone 설정**

|  |
| --- |
| sudo rm /etc/localtime  sudo ln -s /usr/share/zoneinfo/Asia/Seoul /etc/localtime |

**Hostname 설정**

|  |
| --- |
| sudo hostnamectl set-hostname webterview.localdomain  sudo vi /etc/hosts |

1. **Docker, Docker Compose 설치**

**3-1. Docker 설치**

1. **기본 설정, 사전 설치**

|  |
| --- |
| sudo apt update  sudo apt install apt-transport-https ca-certificates curl software-properties-common |

1. **자동 설치 스크립트 활용**

|  |
| --- |
| sudo wget -qO- https://get.docker.com/ | sh |

1. **Docker 서비스 실행하기 및 부팅 시 자동 실행 설정**

|  |
| --- |
| sudo systemctl start docker  sudo systemctl enable docker |

1. **Docker 그룹에 현재 계정 추가**

|  |
| --- |
| sudo usermod -aG docker ${USER}  sudo systemctl restart docker |

**3-2 Docker Compose 설치**

1. **설치**

|  |
| --- |
| sudo curl -L "https://github.com/docker/compose/releases/download/1.24.1/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose |

1. **권한 설정**

|  |
| --- |
| sudo chmod +x /usr/local/bin/docker-compose |

1. **심볼릭 링크 설정**

|  |
| --- |
| sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose |

**3-3 Docker 명령어**

|  |
| --- |
| # 현재 실행중인 컨테이너  docker ps  # 모든 컨테이너  docker ps -a  # 이미지 목록  docker images  # 컨테이너 중지  docker kill [컨테이너이름|컨테이너ID]  # 컨테이너 시작  docker start [컨테이너이름|컨테이너ID]  # 컨테이너 삭제  docker rm [컨테이너이름|컨테이너ID]  # 이미지 삭제 docker rmi [이미지이름|이미지ID]  # 실행중인 컨테이너 shell 환경으로 접속 docker exec -it [컨테이너이름|컨테이너ID] bash  # 컨테이너 로그  docker logs -f [컨테이너이름|컨테이너ID] |

1. **Nginx 설치 및 설정**

**4-1 Nginx 설치**

|  |
| --- |
| sudo apt update sudo apt install nginx |

**4-2 SSL 인증서**

1. **Certbot 설치**

|  |
| --- |
| sudo add-apt-repository ppa:certbot/certbot  sudo apt install python-certbot-nginx |

1. **SSL 인증서 가져오기**

|  |
| --- |
| # nginx 플러그인을 사용한다.  sudo certbot --nginx –d i8c202.p.ssafy.io |

**4-3 default 설정**

|  |
| --- |
| server\_name i8c202.p.ssafy.io; # managed by Certbot  location / {  # First attempt to serve request as file, then  # as directory, then fall back to displaying a 404.  #try\_files $uri $uri/ =404;  charset utf-8;  proxy\_redirect off;  proxy\_set\_header X-Real-Ip $remote\_addr;  proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;  proxy\_set\_header X-Forwarded-Proto $scheme;  proxy\_set\_header X-NginX-Proxy true; client\_max\_body\_size 10M;  proxy\_pass <http://localhost:3000/>;  }  location /api {  error\_page 405 =200 $uri; proxy\_redirect off;  charset utf-8; proxy\_set\_header X-Real-Ip $remote\_addr;  proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;  proxy\_set\_header X-Forwarded-Proto $scheme;  proxy\_set\_header X-NginX-Proxy true;  client\_max\_body\_size 10M;  proxy\_pass <http://localhost:8080/api>;  }  listen [::]:443 ssl ipv6only=on; # managed by Certbot  listen 443 ssl; # managed by Certbot  ssl\_certificate /etc/letsencrypt/live/i8c202.p.ssafy.io/fullchain.pem; # managed by Certbot  ssl\_certificate\_key /etc/letsencrypt/live/i8c202.p.ssafy.io/privkey.pem; # managed by Certbot  include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot  ssl\_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot  }  server {  if ($host = i8c202.p.ssafy.io) {  return 301 <https://$host$request_uri>;  } # managed by Certbot  listen 80 ;  listen [::]:80 ;  server\_name i8c202.p.ssafy.io;  return 404est\_uri;; # managed by Certbot  } |

1. **빌드**

**FrontEnd**

|  |
| --- |
| npm install --legacy-peer-deps  npm run build |

**BackEnd**

|  |
| --- |
| ./gradlew build |