



# [포팅 메뉴얼]

## 프로젝트 기술 스택

- 가. 이슈관리 : Jira
- 나. 형상관리 : GitLab
- 다. 커뮤니케이션 : Mattermost, Notion, Webex
- 라. 개발환경

- 1. OS : Windows11
- 2. IDE
- 3. Database : MySQL Workbench 8.0 CE
- 4. Server :
- 5. 상세 내용

## 백엔드 빌드 방법

- 1. Command 빌드 방법
- 2. IntelliJ 빌드 방법

## 프론트엔드 빌드 방법

- 1. node\_modules를 위한 기본 install
- 2. 현재 상태로 빌드하기

## 배포 메뉴얼

- EC2 서버 구조
- MobaXterm을 이용해서 Ubuntu 접속
- mySQL 서버 구축하기
- Docker 설치
- Jenkin 설정
- 자동 빌드를 위한 Jenkins, Gitlab Webhook 설정
- SpringBoot 배포
- Nginx를 활용한 React 배포
- certbot을 이용한 SSL 설정
- Nginx 설정

## Spark 설정

- PlantGo! 서비스에서 Spark의 활용
- Project 생성
- SimpleApp
- 프로젝트 빌드
- Cluster 서버에서 실행 및 스케줄러 설정

## S3 세팅

### MySQL 워크벤치 추가하기

### Kakao Dev 설정

- 가. REST-API Key 등록
- 나. 카카오 플랫폼 도메인 등록
- 다. Redirect URI 등록

### Naver Dev 설정

- 가. Client ID, Client Secret 설정
- 나. 로그인 오픈 API 서비스 환경 설정

### Google Dev 설정

- 가. Client ID, Client Secret 설정
- 나. Redirection URI 설정

## 프로젝트 기술 스택

### 가. 이슈관리 : Jira

## 나. 형상관리 : GitLab

## 다. 커뮤니케이션 : Mattermost, Notion, Webex

## 라. 개발환경

### 1. OS : Windows11

### 2. IDE

- 인텔리제이 : IntelliJ IDEA 2022.1.3 (Ultimate Edition)
- visual studio code : 1.7.0

### 3. Database : MySQL Workbench 8.0 CE

### 4. Server :

### 5. 상세 내용

- Backend
  - Springboot : `'org.springframework.boot' version '2.5.5'`
  - Java : 8
  - JDK : 1.8
  - Gradle : `gradle-7.5-bin`
  - Java Data JPA : `spring-boot-starter-data-jpa`
  - OAuth2 : `spring-boot-starter-oauth2-client`
  - Jwt : `jjwt-api:0.11.2`
  - Swagger : `springfox-boot-starter:3.0.0`
- Frontend
  - node.js : 16.16.0(LTS)
  - npm : 8.11.0
  - HTML5, CSS3, JavaScript(ES6)
  - react : ^18.2.0
  - redux : ^4.2.0
  - redux-thunk : ^2.4.1
  - scss : ^0.2.4
  - typescript : ^4.8.3
- 배포 및 서버
  - Jenkins
  - Nginx
  - S3

## 백엔드 빌드 방법

## 1. Command 빌드 방법

- git clone "GIT REPOSITORY"
- cd S07P22A703/backend
- 빌드
  - (Window) ./gradlew.bat → ./gradlew build
  - (MacOS) ./gradlew build
- cd build/libs
- 서버 실행 : java -jar plantgo-0.0.1-SNAPSHOT.jar
- 서버 중지
  - (Window) ctrl + c
  - (MacOS) ls -arllh
- 빌드 삭제
  - (Window) ./gradlew.bat clean build
  - (MacOS) ./gradlew clean build

## 2. IntelliJ 빌드 방법

- git clone "GIT REPOSITORY"
- cd S07P22A703/backend
- backend 디렉토리 오른쪽 버튼 클릭 → Open Folder as IntelliJ IDEA Project
- src/main/resources/application.yml 추가
- ▼ application.yml

```
spring:
  datasource:
    url: jdbc:mysql://54.180.156.173:3306/plantgo
    username: 'plantgo'
    password: 'a703pg7'

  jpa:
    show-sql: true
    hibernate:
      ddl-auto: update
      naming-strategy: org.hibernate.cfg.ImprovedNamingStrategy
    properties:
      hibernate:
        dialect: org.hibernate.dialect.MySQL5InnoDBDialect
  security:
    oauth2:
      client:
        registration:
          naver:
            clientId: r0MZbZsfVQ3G0SLFYn7g
            client-secret: _UXmsitDL2
            clientAuthenticationMethod: post
            redirect-uri: "http://j7a703.p.ssafy.io:8080/login/oauth2/code/NAVER"
            authorization-grant-type: authorization_code
            scope:
              - nickname
              - email
              - profile_image
            client-name : Naver
        google:
            clientId: 416742949716-pao515mh3u0itqv4givjfp1jg9e16rbv.apps.googleusercontent.com
            clientSecret: GOCSPX-dH5lpAuaTA471Nvra0kc940IEg
            redirect-uri: "http://j7a703.p.ssafy.io:8080/login/oauth2/code/GOOGLE"
```

```

scope:
  - email
  - profile
kakao:
  clientId: 07efa27cef29c1f8650069e247c5b553
  clientSecret: dXGoKqRxWyc4Eibc0BxpX2evsEL2JTHH
  clientAuthenticationMethod: post
  authorizationGrantType: authorization_code
  redirectUri: "http://j7a703.p.ssafy.io:8080/login/oauth2/code/KAKAO"
  scope:
    - profile_nickname
    - profile_image
    - account_email
  clientName: Kakao
# Provider 설정
provider:
  naver:
    authorizationUri: https://nid.naver.com/oauth2.0/authorize
    tokenUri: https://nid.naver.com/oauth2.0/token
    userInfoUri: https://openapi.naver.com/v1/nid/me
    userNameAttribute: response
  kakao:
    authorizationUri: https://kauth.kakao.com/oauth/authorize
    tokenUri: https://kauth.kakao.com/oauth/token
    userInfoUri: https://kapi.kakao.com/v2/user/me
    userNameAttribute: id

# cors 설정
cors:
  allowed-origins: 'https://j7a703.p.ssafy.io'
  allowed-methods: GET, POST, PUT, DELETE, OPTIONS
  allowed-headers: '*'
  max-age: 3600

# jwt secret key 설정
jwt.secret: '8sknj103NPTBqo319DHLNqsQAfRJEdKsET0ds'

# 토큰 관련 secret Key 및 RedirectUri 설정
app:
  auth:
    tokenSecret: 926D96C90030DD58429D2751AC1BDBBC
    tokenExpiry: 1800000
    refreshTokenExpiry: 60480000
  oauth2:
    authorizedRedirectUris:
      - https://j7a703.p.ssafy.io/oauth/redirect

cloud:
  aws:
    credentials:
      accessKey: AKIA4X0CG6JP6F0PIBS7
      secretKey: I9CHL2qoQBod1CLzb3FAy2uGAm6GyGCK1oz+zB5e
  s3:
    bucket: plantgo
    dir: /photocard
  region:
    static: ap-northeast-2
  stack:
    auto: false

```

- src/main/java/PlantgoApplication 실행

## 프론트엔드 빌드 방법

### 1. node\_modules를 위한 기본 install

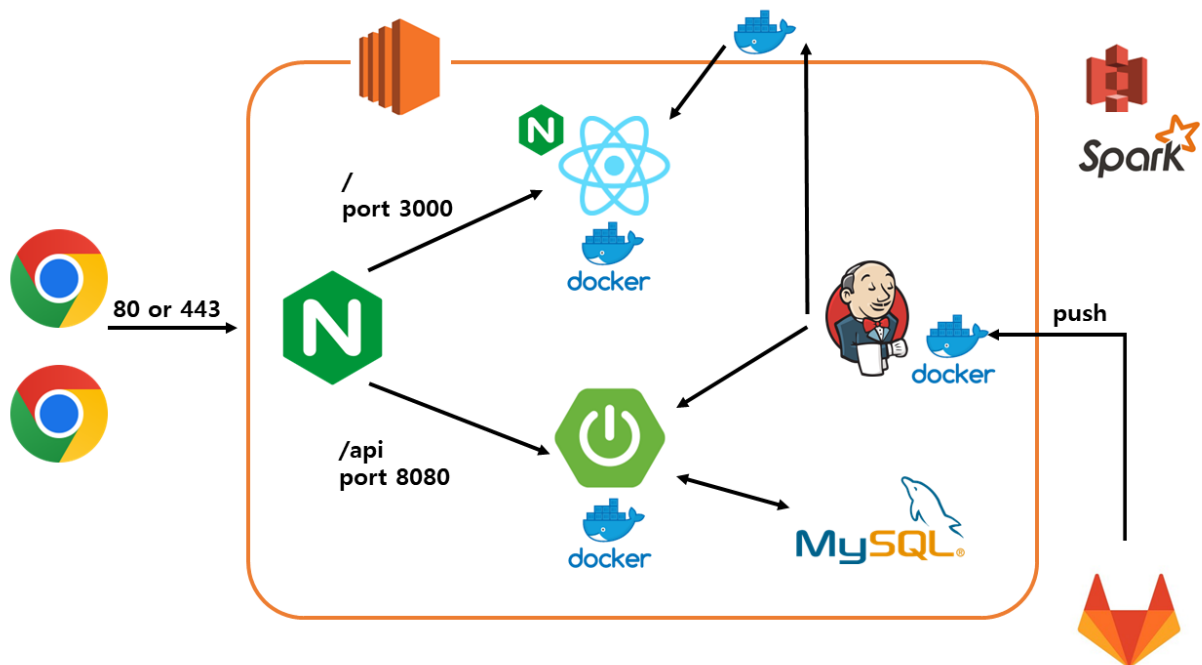
```
npm install
```

## 2. 현재 상태로 빌드하기

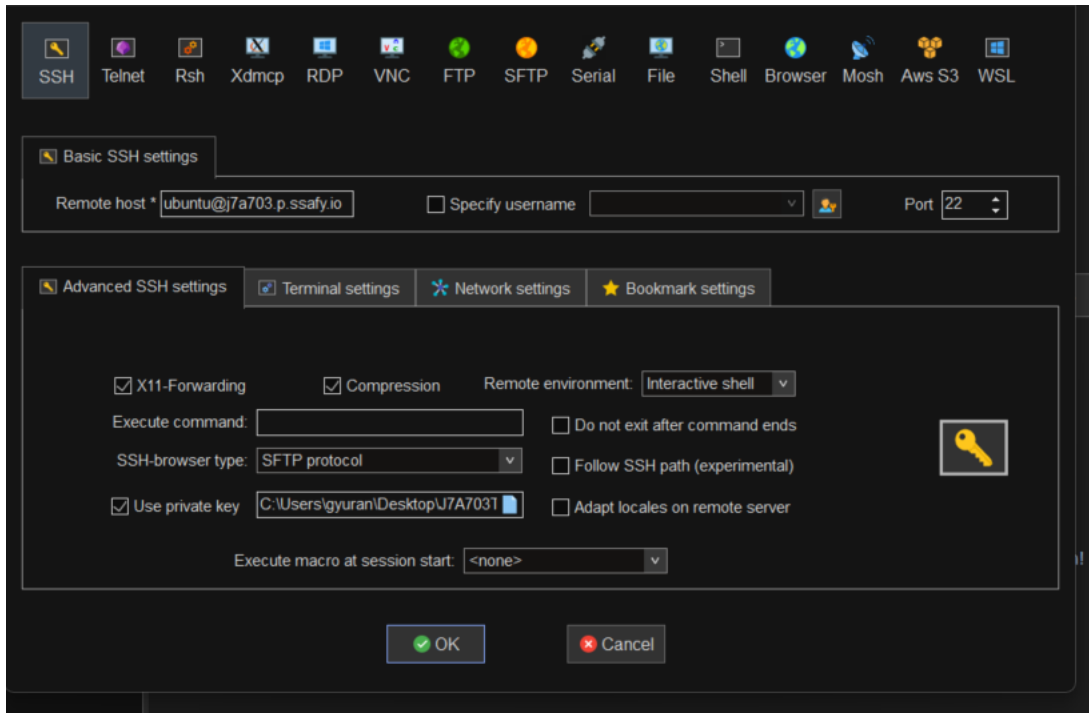
```
npm run build
```

## 배포 메뉴얼

### EC2 서버 구조



### MobaXterm을 이용해서 Ubuntu 접속



- openjdk, maven, npm 설치

```
# install openjdk-8
sudo apt-get install openjdk-8-jdk
java -version

# install maven
sudo apt install maven

# install npm
sudo apt install npm
```

## mysql 서버 구축하기

- 패키지 관리 도구 최신화

```
$ sudo apt update && sudo apt-get -y update
```

- mysql 서버 설치

```
$ sudo apt-get install mysql-server
```

- mysql 접속, 초기에는 비밀번호 없으므로 enter 눌러서 접속

```
$ sudo mysql -u root -p
```

- root 계정 비밀번호 설정

```
> use mysql
> select Host, User, authentication_string from user;
> alter user 'root'@'localhost' identified with mysql_native_password by '비밀번호';
> flush privileges;
> exit
```

```
mysql> select Host, User, authentication_string from user;
```

Host	User	authentication_string
localhost	debian-sys-maint	\$A\$005\$W~1*A
localhost	mysql.infoschema	\$A\$005\$THISISACOMBINATIONOFINVALIDSALTANDPASSWORDTHATMUSTNEVERBRBEUSED
localhost	mysql.session	\$A\$005\$THISISACOMBINATIONOFINVALIDSALTANDPASSWORDTHATMUSTNEVERBRBEUSED
localhost	mysql.sys	\$A\$005\$THISISACOMBINATIONOFINVALIDSALTANDPASSWORDTHATMUSTNEVERBRBEUSED
localhost	root	\$A\$005\$THISISACOMBINATIONOFINVALIDSALTANDPASSWORDTHATMUSTNEVERBRBEUSED

```
5 rows in set (0.00 sec)
```

- 원격에서 접근 가능하도록 설정 변경하기

```
$ sudo vi /etc/mysql/mysql.conf.d/mysqld.cnf > bind-address 0.0.0.0으로 변경
```

```
# If MySQL is running as a replication slave, this should be
# changed. Ref https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_tmpdir
# tmpdir = /tmp
#
# Instead of skip-networking the default is now to listen only on
# localhost which is more compatible and is not less secure.
bind-address = 0.0.0.0
mysqlx-bind-address = 0.0.0.0
#
```

- 설정 후 MySQL 재시작

```
$ sudo service mysql restart
```

- 외부 접속을 위한 계정 생성

```
> create user 'plantgo'@'%' identified by 'a703pg7';
> grant all privileges on *.* to plantgo@'%';
```

## Docker 설치

- 사전 패키지 설치

```
> sudo apt update
> sudo apt-get install -y ca-certificates \
  curl \
  software-properties-common \
  apt-transport-https \
  gnupg \
  lsb-release
```

- gpg 키 다운로드

```
> curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add
```

- Docker 설치

```
> sudo apt update
> sudo apt install docker-ce docker-ce-cli containerd.io docker-compose
```

```
> docker -v
```

## Jenkin 설정

- `$ vim docker-compose.yml` 파일 생성

```
version: '3'

services:
  jenkins:
    image: jenkins/jenkins:lts
    container_name: jenkins
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - /jenkins:/var/jenkins_home
    ports:
      - "9090:8080"
    privileged: true
    user: root
```

```
# 컨테이너 생성
> sudo docker-compose up -d

# 생성된 컨테이너 확인
> sudo docker ps
```

```
ubuntu@ip-172-26-3-134:~$ sudo docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                    NAMES
5ee5d8a080d9   jenkins/jenkins:lts   "/usr/bin/tini -- /u..." 2 hours ago   Up 2 hours   50000/tcp, 0.0.0.0:9090→8080/tcp, :::9090→8080/tcp   jenkins
ubuntu@ip-172-26-3-134:~$
```

- <http://도메인주소/> 접속



# Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log (**not sure where to find it?**) and this file on the server:

```
/var/jenkins_home/secrets/initialAdminPassword
```

Please copy the password from either location and paste it below.

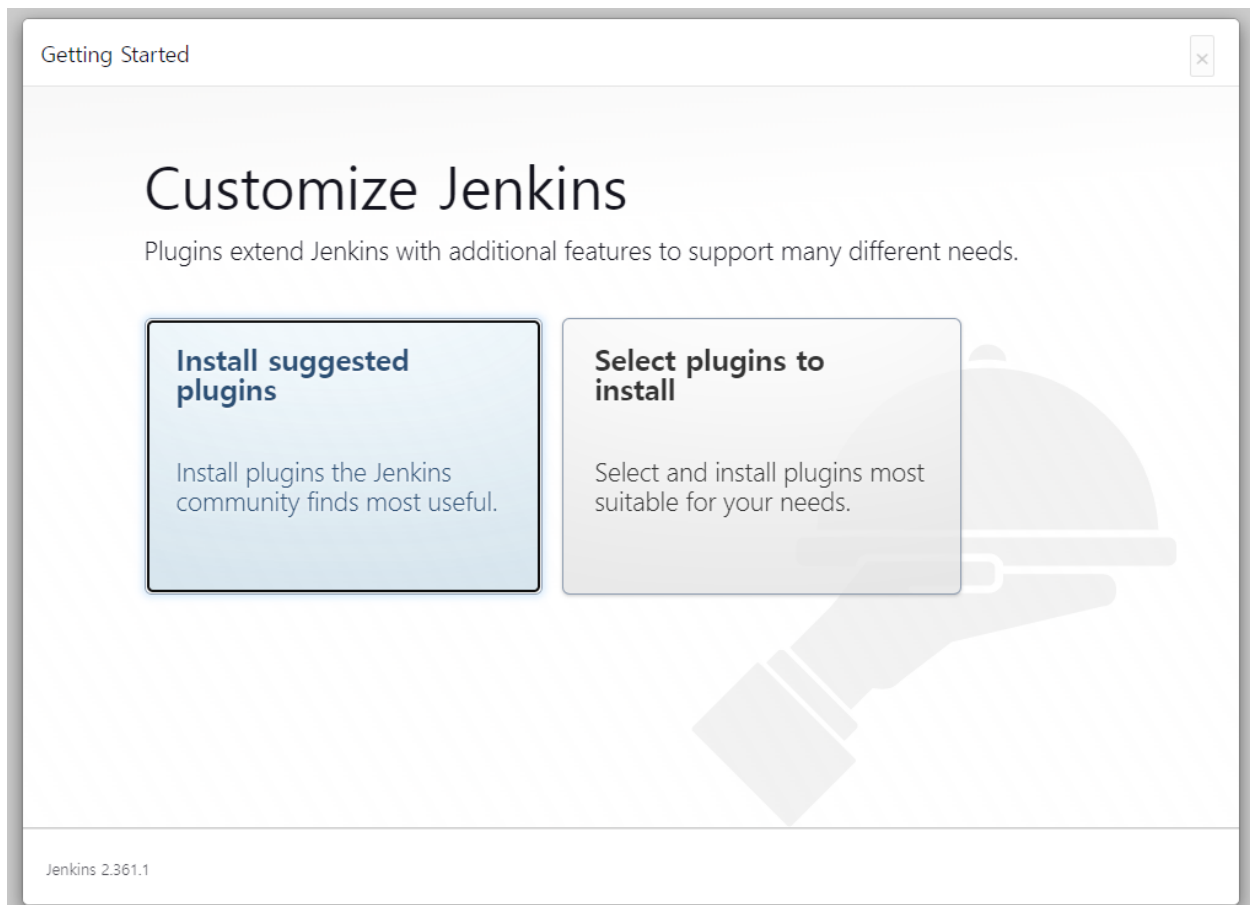
Administrator password

Continue

- 아래 명령어 입력하여 비밀번호 확인 후 입

```
> sudo docker exec -it jenkins /bin/bash // Docker container 접속
> cat /var/jenkins_home/secrets/initialAdminPassword // admin password 확인
```

- Jenkins 설치(Install suggested plugins 클릭)



- 계정 생성 후 Jenkins URL 확인
- 플러그인 설치
- 좌측에서 `Jenkins 관리` > `플러그인 관리` > `설치 가능` > `gitlab` 검색 > `Install without restart`
- 이후에 필요한 플러그인 추가 설치

## Plugin Manager

업데이트된 플러그인 목록

설치 가능

설치된 플러그인 목록

고급

Q gitlab

Install	Name ↓	Released
<input checked="" type="checkbox"/>	<div>GitLab 1.5.35</div> <div>Build Triggers</div> <p>This plugin allows <a href="#">GitLab</a> to trigger Jenkins builds and display their results in the GitLab UI.</p> <div>This plugin is up for adoption! We are looking for new maintainers. Visit our <a href="#">Adopt a Plugin</a> initiative for more information.</div>	2 mo 16 days ago
<input checked="" type="checkbox"/>	<div>Generic Webhook Trigger 1.84.1</div> <div>notification github webhook Build Parameters gitlab Build Triggers bitbucket bitbucket-server jira</div> <p>Can receive any HTTP request, extract any values from JSON or XML and trigger a job with those values available as variables. Works with GitHub, GitLab, Bitbucket, Jira and many more.</p>	4 days 4 hr ago
<input checked="" type="checkbox"/>	<div>Gitlab API 5.0.1-78.v47a_45b_9f78b_7</div> <div>Library plugins (for use by other plugins)</div> <p>This plugin provides <a href="#">GitLab API</a> for other plugins.</p>	1 mo 21 days ago
<input checked="" type="checkbox"/>	<div>GitLab Authentication 1.16</div> <div>Authentication and User Management</div> <p>This is the an authentication plugin using gitlab OAuth.</p> <div>This plugin is up for adoption! We are looking for new maintainers. Visit our <a href="#">Adopt a Plugin</a> initiative for more</div>	4 mo 28 days ago

Install without restart

Download now and install after restart

Update information obtained: 2 hr 15 min ago

지금 확인

Q docker

Install	Name ↓	Released
<input checked="" type="checkbox"/>	<div>Docker 1.2.9</div> <div>Cloud Providers Cluster Management docker</div> <div>This plugin integrates Jenkins with <b>Docker</b></div> <div>This plugin is up for adoption! We are looking for new maintainers. Visit our <a href="#">Adopt a Plugin</a> initiative for more information.</div>	4 mo 20 days ago
<input checked="" type="checkbox"/>	<div>Docker Commons 1.21</div> <div>Library plugins (for use by other plugins) docker</div> <div>Provides the common shared functionality for various Docker-related plugins.</div>	18 days ago
<input checked="" type="checkbox"/>	<div>Docker Pipeline 521.v1a_dd2073b_2e</div> <div>pipeline DevOps Deployment docker</div> <div>Build and use Docker containers from pipelines.</div>	1 mo 6 days ago
<input checked="" type="checkbox"/>	<div>Docker API 3.2.13-37.vf3411c9828b9</div> <div>Library plugins (for use by other plugins) docker</div> <div>This plugin provides <b>docker-java</b> API for other plugins.</div> <div>This plugin is up for adoption! We are looking for new maintainers. Visit our <a href="#">Adopt a Plugin</a> initiative for more information.</div>	4 mo 28 days ago

Q SSH

Install	Name ↓	Released
<input type="checkbox"/>	<div>SSH 2.6.1</div> <div>Build Wrappers</div> <div>This plugin executes shell commands remotely using SSH protocol.</div> <div>Warning: This plugin version may not be safe to use. Please review the following security notices:<ul style="list-style-type: none"><li><a href="#">CSRF vulnerability and missing permission checks allow capturing credentials</a></li><li><a href="#">Missing permission check allows enumerating credentials IDs</a></li></ul></div>	4 yr 5 mo ago
<input checked="" type="checkbox"/>	<div>Publish Over SSH 1.24</div> <div>Artifact Uploaders Build Tools</div> <div>Send build artifacts over SSH</div>	6 mo 24 days ago

## 자동 빌드를 위한 Jenkins, Gitlab Webhook 설정

- 좌측에서 새로운 item 클릭

### Enter an item name

deploy

» Required field



#### Freestyle project

이것은 Jenkins의 주요 기능입니다. Jenkins은 어느 빌드 시스템과 어떤 SCM(형상관리)으로 묶인 당신의 프로젝트를 빌드할 것이고, 소프트웨어 빌드보다 다른 어떤 것에 자주 사용될 수 있습니다.

Failed to connect to repository : Command 'git ls-remote -h -- https://iab.ssary.com/SU7-bigdata-dist-subZ/SU7PZZA

Scope ?

Global (Jenkins, nodes, items, all child items, etc)

Username ?

gyuraniakim@naver.com

☐ Treat username as secret ?

Password ?

.....

ID ?

gyuran

Description ?

☒ Build when a change is pushed to GitLab. GitLab webhook URL: <http://j7a703.p.ssafy.io:9090/project/deploy> ?

Enabled GitLab triggers

☒ Push Events

☐ Push Events in case of branch delete

☒ Opened Merge Request Events

☐ Build only if new commits were pushed to Merge Request ?

☐ Accepted Merge Request Events

☐ Closed Merge Request Events

Rebuild open Merge Requests

Never

☒ Approved Merge Requests (EE-only)

☒ Comments

- 하단에 **Secret token** > **Generate**
- 깃랩에서 설정해줘야 하므로 토큰 값 복사

- Gitlab 레파지토리 > **Setting** > **Webhooks**

## Webhooks

Webhooks enable you to send notifications to web applications in response to events in a group or project. We recommend using an [integration](#) in preference to a webhook.

### URL

<http://example.com/trigger-ci.json>

URL must be percent-encoded if it contains one or more special characters.

### Secret token

Used to validate received payloads. Sent with the request in the **X-Gitlab-Token** HTTP header.

### Trigger

☒ Push events

Branch name or wildcard pattern to trigger on (leave blank for all)

Push to the repository.

- Jenkins 프로젝트 URL 입력 후 복사한 **Secret token** 값 입력, **Push events** 에 이벤트 발생 시 빌드할 브랜치 입력
  - Jenkins URL 예시: <http://도메인주소:9090/project/프로젝트명/>
- Mattermost Webhook 설정 방

## 빌드 후 조치

☰

Mattermost Notifications

☒ Notify Build Start

☐ Notify Aborted

☒ Notify Failure

☐ Notify Not Built

☒ Notify Success

☐ Notify Unstable

☐ Notify Back To Normal

고급...

## 전체 Incoming Webhook

[Incoming Webhook 추가하기](#)

Incoming webhook을 사용하여 외부 도구를 Mattermost에 연결 [Build Your Own](#) 또는 [앱 디렉터리](#)에 방문하여 자체 호스팅 third-party 앱과 통합기능을 찾습니다.

jenkins

[편집 - 삭제](#)

URL: <https://meeting.ssafy.com/hooks/xqc1n5k73f8szj8781b85b5s1e>

gyuraniakim(이)가 2022년 9월 23일 금요일에 생성

## SpringBoot 배포

- Dockerfile

```
FROM openjdk:8
EXPOSE 8080
CMD ["ls", "-al"]
ARG JAR_FILE=plantgo/build/libs/plantgo-0.0.1-SNAPSHOT.jar
CMD ["ls", "-al"]
COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java", "-jar", "/app.jar"]
```

- Jenkins Execute Shell

```
cd plantgo

clean build

# 도커 이미지 빌드
docker build -t plantgo -f plantgo/Dockerfile .
```

```
# 실행 중인 컨테이너 검색 후 동일한 이름의 컨테이너 삭제
docker ps -q --filter name=plantgo | grep -q . && docker stop plantgo && docker rm plantgo
# 빌드한 이미지로 컨테이너 실행
docker run -p 8080:8080 -d --name=plantgo plantgo

# 컨테이너 실행 후 불필요한 이미지 삭제
docker rmi -f $(docker images -f "dangling=true" -q) || true
```

Invoke Gradle script ?

Invoke Gradle ?

Gradle Version

gradle

Use Gradle Wrapper ?

Tasks ?

clean build

Switches ?

System properties

Root Build script ?

/var/jenkins\_home/workspace/back01/plantgo

Build File ?

Specify Gradle build file to run. Also, [some environment variables are available to the build script](#)

build.gradle

Force GRADLE\_USER\_HOME to use workspace ?

## Nginx를 활용한 React 배포

- Dockerfile



```
FROM node:lts-alpine as build-stage

WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
RUN npm run build

FROM nginx:stable-alpine as production-stage
RUN mkdir /app
RUN rm /etc/nginx/conf.d/default.conf
COPY ./nginx.conf /etc/nginx/conf.d
COPY --from=build-stage ./app/build /app

EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

- nginx.conf

```
server {
    listen 80;
    location / {
        root    /app;
        index   index.html;
        try_files $uri $uri/ /index.html;
    }
}
```

- Jenkins Execute Shell

```
cd front-plantgo
# 도커 이미지 빌드
docker build -t front-plantgo .

# 도커 허브에 이미지 push
docker tag front-plantgo:latest gyurania/front-plantgo:latest
docker push gyurania/front-plantgo:latest

# 도커 허브에서 이미지 pull 받은 후 컨테이너 실행
docker pull gyurania/front-plantgo:latest
docker ps -q -a --filter name=front-plantgo | grep -q . && docker stop front-plantgo && docker rm front-plantgo
docker run -p 3000:80 -d --name=front-plantgo gyurania/front-plantgo
docker rmi -f $(docker images -f "dangling=true" -q) || true
```

## certbot을 이용한 SSL 설정

```
> sudo apt update

> sudo apt-get install letsencrypt -y

# Nginx 중단 후 인증서 발급 받고 재실행
> sudo service nginx stop
> sudo certbot certonly --standalone -d 도메인주소
> sudo service nginx restart
```

## Nginx 설정

- Ubuntu 접속 후 Nginx 설정 파일 확인

```
> cd /etc/nginx/sites-available
> sudo vim default
```

- default

```
server {
    listen 80 default_server;
    listen [::]:80 default_server;
    server_name j7a703.p.ssafy.io;

    location / {
        return 301 https://$host$request_uri;
    }
}

server {
    # SSL configuration

    listen 443 ssl;
    listen [::]:443 ssl ipv6only=on;
    ssl_certificate /etc/letsencrypt/live/j7a703.p.ssafy.io/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/j7a703.p.ssafy.io/privkey.pem;

    root /var/www/html;

    # Add index.php to the list if you are using PHP
    index index.html index.htm index.nginx-debian.html;

    server_name _;

    location / {
        # First attempt to serve request as file, then
        # as directory, then fall back to displaying a 404.
        # try_files $uri $uri/ =404;
        proxy_pass http://54.180.156.173:3000;
    }

    location /api {
        proxy_pass http://54.180.156.173:8080;
        proxy_redirect off;
        charset utf-8;
    }
}
```

## Spark 설정

### PlantGo! 서비스에서 Spark의 활용

PlantGo!의 랭킹 기능은 전체 회원들을 대상으로 하며, 각 회원이 수집한 포토카드 목록을 바탕으로 해당 회원이 수집한 전체 식물의 개수를 카운트하여 랭킹을 매겨주어야 합니다. 이 때, 포토카드와 회원들의 수가 늘어나게 되면 회원들이 랭킹 페이지에서 백엔드로 요청을 보낼 때마다 해당 로직을 수행해 소요 시간이 늘어나고 서버에 과부하가 걸릴 수 있으므로, 랭킹 테이블을 별도로 생성하고 주기적으로 스파크에서 Top30 랭킹을 구해 해당 테이블에 insert 해주게 됩니다.

### Project 생성

1. Maven 기반의 Java 프로젝트를 생성합니다.
2. pom.xml 파일에 다음과 같은 dependency를 추가해줍니다.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
```

```

<groupId>org.example</groupId>
<artifactId>plantgo-spark2</artifactId>
<version>1.0-SNAPSHOT</version>

<properties>
  <maven.compiler.source>8</maven.compiler.source>
  <maven.compiler.target>8</maven.compiler.target>
</properties>

<dependencies>
  <dependency>
    <!-- Spark dependency -->
    <groupId>org.apache.spark</groupId>
    <artifactId>spark-sql_2.11</artifactId>
    <version>2.4.0</version>
  </dependency>
  <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
  <dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.28</version>
  </dependency>
</dependencies>

</project>

```

## SimpleApp

```

/* SimpleApp.java */
import org.apache.spark.sql.*;

import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Properties;

import static org.apache.spark.sql.functions.desc;
import static org.apache.spark.sql.functions.lit;

public class SimpleApp {

    private static final String MYSQL_DRIVER = "com.mysql.jdbc.Driver";
    private static final String MYSQL_CONNECTION_URL = "jdbc:mysql://j7a703.p.ssafy.io:3306/plantgo?allowPublicKeyRetrieval=true&useSSL=false";
    private static final String MYSQL_DBTable = "photocard";
    private static final String MYSQL_USERNAME = "plantgo";
    private static final String MYSQL_PWD = "a703pg7";

    public static void main(String[] args) {
        //Spark세션 및 앱 설정&생성
        SparkSession spark = SparkSession.builder().appName("Simple Application").config("spark.master", "local").getOrCreate();

        //DB 연결 설정
        Properties connectionProperties = new Properties();
        connectionProperties.put("user", MYSQL_USERNAME);
        connectionProperties.put("password", MYSQL_PWD);

        Dataset<Row> photocards = spark.read()
            .jdbc(MYSQL_CONNECTION_URL, MYSQL_DBTable, connectionProperties);

        //MySQL 테이블에서 필요없는 컬럼 삭제
        Dataset<Row> collected = photocards.drop("photocard_id", "area", "latitude", "longitude", "memo", "photo_url", "plant");
        //중복되는 row 삭제
        Dataset<Row> dropped = collected.dropDuplicates();

        //회원번호를 기준으로 그룹화하여 데이터 집계, 내림차순으로 정렬, 상위 30개만 가져옴
        RelationalGroupedDataset groupedDataset = dropped.groupBy("user_seq");
        Dataset<Row> computed = groupedDataset.count().sort(desc("count")).limit(30);

        //랭킹을 생성한 시간 기록
        SimpleDateFormat format1 = new SimpleDateFormat ( "yyyy-MM-dd HH:mm:ss");
        Date time = new Date();
        String time1 = format1.format(time);
    }
}

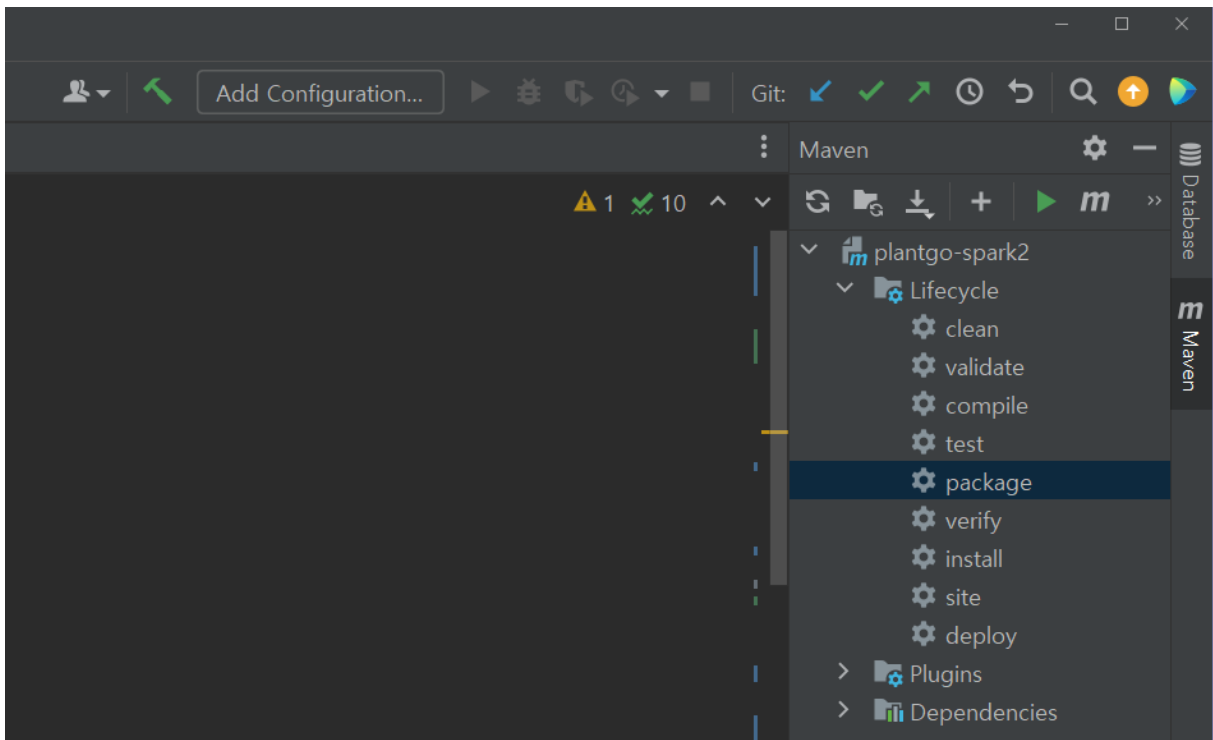
```

```
//시간데이터의 default 값을 바탕으로 칼럼 생성 및 count 칼럼 이름 바꿈
Dataset<Row> finalData = computed.withColumn("insert_time", lit(time1)).withColumnRenamed("count", "plants_collects");

//userrank 테이블에 랭크데이터를 추가
finalData.write()
    .mode("append")
    .jdbc(MYSQL_CONNECTION_URL, "userrank", connectionProperties);
//spark 종료
spark.stop();
}
}
```

## 프로젝트 빌드

- 인텔리제이를 사용하는 경우 : Maven → Lifecycle → package



- Cmd로 빌드하는 경우 : (Maven 설치 필수) 명령어 입력

```
mvn package
```



빌드 후 클러스터 서버에(MobaXterm으로 접속) 마우스 드래그로 로컬 프로젝트를 복사

## Cluster 서버에서 실행 및 스케줄러 설정

```
spark-submit \
--class "SimpleApp" \
--master local[*] \
plantgo-spark2/target/plantgo-spark2-1.0-SNAPSHOT.jar
```

SimpleApp 클래스를 마스터 노드에서(로컬) 모든 코어를 사용해 빌드된 프로젝트의 .jar 파일 실행

```
date
crontab -e
00 18 * * * spark-submit --class "SimpleApp" --master local[*] plantgo-spark2/target/plantgo-spark2-1.0-SNAPSHOT.jar
crontab -l

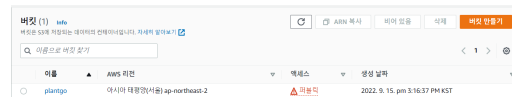
// 모든 스케줄 삭제
crontab -r

// 3분 마다 실행
crontab -e
*/3 * * * * spark-submit --class "SimpleApp" --master local[*] plantgo-spark2/target/plantgo-spark2-1.0-SNAPSHOT.jar
crontab -l
```

1. 리눅스 서버의 timezone 확인
2. crontab 작업목록 수정
3. 매일 한국 시간으로 새벽 3시, UTC 시간으로 저녁 6시에 해당 명령어 실행
4. crontab 작업목록 확인

## S3 세팅

1. 아마존 S3 버킷 생성



2. 버킷정책

```
{
  "Version": "2012-10-17",
  "Id": "Policy1664504324395",
  "Statement": [
    {
      "Sid": "Stmt1664504198466",
      "Effect": "Allow",
      "Principal": "*",
      "Action": [
        "s3:DeleteObject",
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3:::plantgo/*"
    }
  ]
}
```

3. IAM 사용자 추가

## 사용자 추가

1 2 3 4 5

### 사용자 세부 정보 설정

동일한 액세스 유형 및 권한을 사용하여 한 번에 여러 사용자를 추가할 수 있습니다. [자세히 알아보기](#)

사용자 이름\*

[+ 다른 사용자 추가](#)

### AWS 액세스 유형 선택


해당 사용자가 AWS에 액세스하는 방법을 선택합니다. 마지막 단계에서는 액세스 키와 자동 생성된 비밀번호가 제공됩니다. [자세히 알아보기](#)


- 액세스 유형\* ☒ **프로그래밍 방식 액세스**  
AWS API, CLI, SDK 및 기타 개발 도구에 대해 액세스 키 ID 및 비밀 액세스 키(를) 활성화합니다.
- ☐ **AWS Management Console 액세스**  
사용자가 AWS Management Console에 로그인할 수 있도록 허용하는 비밀번호(를) 활성화합니다.


## 사용자 추가

1 2 3 4 5

### ▼ 권한 설정

 그룹에 사용자 추가

 기존 사용자에서 권한 복사

 기존 정책 직접 연결

정책 생성




정책 필터 





1 결과 표시

**객체 소유권** [info](#) 편집  
다른 AWS 계정에서 이 버킷에 작성된 객체의 소유권 및 액세스 제어 목록(ACL)의 사용을 제어합니다. 객체 소유권은 객체에 대한 액세스를 지정할 수 있는 사용자를 결정합니다.

**객체 소유권**  
버킷 소유자 선택  
ACL이 활성화되어 이 버킷과 그 객체에 대한 액세스 권한을 부여하는 데 사용될 수 있습니다. 이 버킷에 작성된 새 객체가 bucket-owner-full-control 설정 ACL을 지정하는 경우 새 객체는 버킷 소유자가 소유합니다. 그렇지 않은 경우 객체 라이더가 소유합니다.

**ACL(액세스 제어 목록)** 편집  
다른 AWS 계정에 기본 읽기/쓰기 권한을 부여합니다. [자세히 알아보기](#)

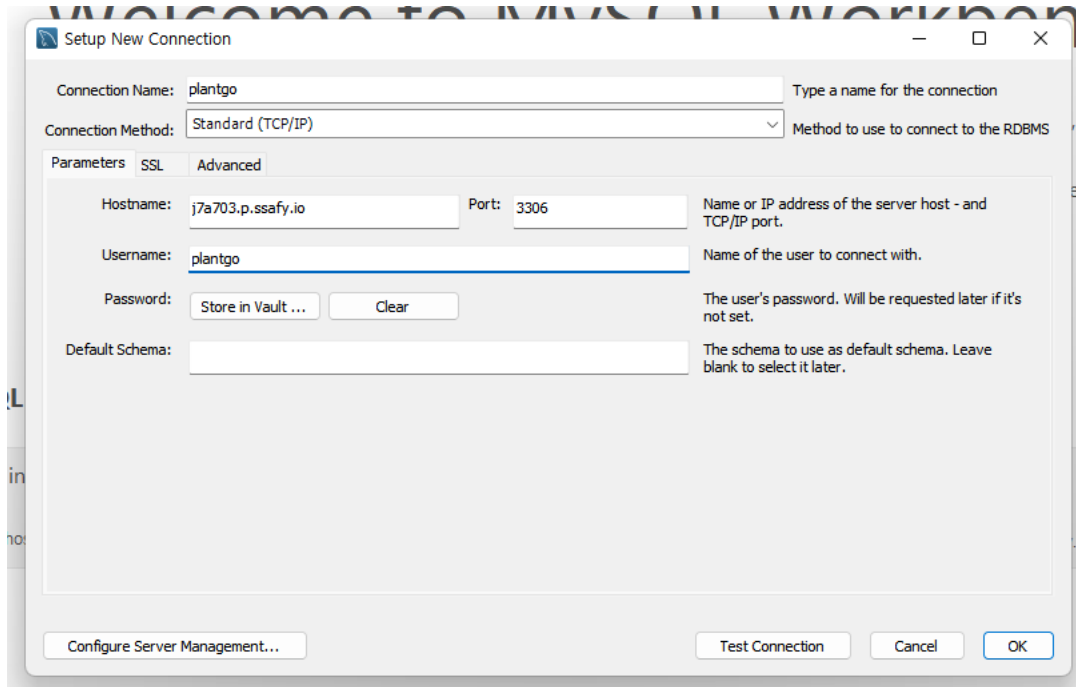
 **중복 피부여자에 대한 통합 액세스 권한 부여가 콘솔에 표시됩니다.**  
전체 ACL 목록을 보려면 Amazon S3 REST API, AWS CLI 또는 AWS SDK를 사용하세요.

피부여자	객체	버킷 ACL
버킷 소유자(AWS 계정) 정식 ID:  a414706e8b142391a4f8905ddefa70da8bebf79b6071f4170d2c7ea513c3331	나열, 쓰기	읽기, 쓰기
모든 사람(퍼블릭 액세스) 그룹:  http://acs.amazonaws.com/groups/global/AllUsers	-	-
인증된 사용자 그룹(AWS 계정이 있는 모든 사용자) 그룹:  http://acs.amazonaws.com/groups/global/AuthenticatedUsers	-	-
S3 로그 전달 그룹 그룹:  http://acs.amazonaws.com/groups/s3/LogDelivery	-	-

참고

<https://devlog-wjdrbs96.tistory.com/323>

## MySQL 워크bench 추가하기



## Kakao Dev 설정

### 가. REST-API Key 등록



plantgo



ID 799819

OWNER

Biz

Web

## 앱 키

네이티브 앱 키	e1ba26dd88505ed1dba9d50dc1a52c3e
REST API 키	07efa27cef29c1f8650069e247c5b553
JavaScript 키	cc64a004ad67f286068c848d6c47c3df
Admin 키	36802e2c616afc8501f95fec0da17d98

## 나. 카카오 플랫폼 도메인 등록

### Web

사이트 도메인	http://j7a703.p.ssafy.io
---------	--------------------------

- 카카오 로그인 사용 시 Redirect URI를 등록해야 합니다. [등록하러 가기](#)

## 다. Redirect URI 등록



## Redirect URI

Redirect URI	http://j7a703.p.ssafy.io:8080/login/oauth2/code/KAKAO
--------------	---

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

## Naver Dev 설정

### 가. Client ID, Client Secret 설정

#### 애플리케이션 정보

Client ID	<input type="text" value="rOMZbZsfVQ3G0SIFYn7g"/>
Client Secret	<input type="text" value="_UXmsitDL2"/> <input type="button" value="재발급"/>

### 나. 로그인 오픈 API 서비스 환경 설정

PC 웹
×
^

서비스 URL

http://j7a703.p.ssafy.io:8080

서비스 URL에서: (O) http://naver.com (X) http://www.naver.com

서비스 URL값이 잘못 입력되어 있으면 정확한 값으로 수정하실 때 까지 네이버 로그인 사용이 일시적으로 제한됩니다.

불법/음란성 사이트 등 이용약관에 위배되는 사이트의 경우, 이용이 제한될 수 있습니다.

서비스하려는 사이트 URL과 동일한 사이트 URL로 해주셔야 **네이버 로그인** **벤티지**가 노출됩니다.

네이버 로그인

Callback URL (최대 5개)

http://j7a703.p.ssafy.io:8080/login/oauth2/code/NAVER
+

텍스트 폼 우측 끝의 '+' 버튼을 누르면 행이 추가되며, '-' 버튼을 누르면 행이 삭제됩니다.

Callback URL은 네이버 로그인 후 이동할 페이지 URL입니다. Callback URL값이 잘못 입력되어 있으면 정확한 값으로 수정하실 때 까지 네이버 로그인 사용이 일시적으로 제한됩니다.

입력한 주소와 다른 Callback URL로 리다이렉트 될 경우, 이용이 제한될 수 있습니다.

## Google Dev 설정

### 가. Client ID, Client Secret 설정

클라이언트 ID	416742949716-pao515mh3u0itqv4gijvfp1jg9e16rbv.apps.googleusercontent.com
클라이언트 보안 비밀	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
생성일	2022년 9월 6일 AM 8시 48분 39초 GMT+9

### 나. Redirection URI 설정

## 승인된 리디렉션 URI

웹 서버의 요청에 사용

URI 1 \*

`http://j7a703.p.ssafy.io:8080/login/oauth2/code/GOOGLE`

[+ URI 추가](#)