



Porting Manual



삼성청년 SW 아카데미 서울캠퍼스 9기 특화 프로젝트 포팅 매뉴얼

1. 프로젝트 주제 : 피움 (FI:UM)
2. 프로젝트 기간 : 2023.08.28. - 2023.10.06.
3. 개발 인원 : 김영우(팀장), 이승준, 김승우, 남완희, 신기정, 노태균
4. 담당 코치 : 이치현

피움(FI:UM) 포팅 매뉴얼 목차

1. 프로젝트 기술 스택

a. FE (Front-End)

- i. 기술 스택
- ii. 개발 환경

b. BE (Back-End)

- i. 기술 스택
- ii. 개발 환경

2. 프로퍼티 설정

a. Jenkinsfile

b. Dockerfile

c. Flask server

d. .gitignore

e. FE property

f. BE property

g. 사용 포트

2. 도구 설치 및 설정

a. AWS EC2 서버 설정

- i. 방화벽 설정

b. MySQL 설치 및 설정

- i. AWS EC2 내에 MySQL 설치
- ii. MySQL 로그인
- iii. Database & Table 생성
- iv. Listen IP 대역폭 변경
- v. MySQL 재시작
- vi. Workbench 연결

c. Nginx 설치 및 설정

- i. 설치
- ii. 상태 확인
- iii. 실행 시작 / 중지
- iv. 환경 설정
- v. SSL 설정

d. Docker 설치 및 설정

- i. 기존 버전 삭제
- ii. apt repository 셋업
- iii. Docker Engine 설치
- iv. 설치 확인
- v. 명령어 모음

e. Jenkins 설치 및 설정

- i. Docker 내 Jenkins Container 실행

- ii. Jenkins 접속 및 라이브러리 설치
- iii. Jenkins 계정 설정
- iv. Plugin 설치
- v. Credential 등록
- vi. Item 생성 및 빌드 설정
- vii. GitLab Webhook 등록
- viii. 빌드 확인

3. 빌드 방법

- a. BE 빌드 방법
- b. FE 빌드 방법

1. 프로젝트 기술 스택

A. FE

- 기술 스택
 - React 18.2.0
 - TypeScript 5.0.2
 - Vite 4.4.5
 - Scss 1.66.1
 - Recoil 0.7.7
 - Jest 29.6.4
- 개발 환경
 - Visual Studio Code 1.80.1

B. BE

- 기술 스택
 - Java Open-JDK zulu 11.0.20
 - SpringBoot 2.7.15
 - Gradle 8.2.1
 - jbcrypt 0.3
 - jjwt 0.11.2
 - Swagger 2.9.2
 - lombok
 - JPA
 - Redis 7.2.1
 - MySQL 8.1.0
 - Nginx 1.18.0
 - Flask 2.3.3
 - finance-datareader 0.9.50
 - pandas-datareader 0.10.0
 - PyMySQL 1.1.0
 - pyportfoliopt 1.5.5
- 개발 환경
 - IntelliJ 2021.2.4
 - Visual Studio Code 1.80.1
 - MobaXterm 23.2
 - Postman 10.16.3

D. etc.

- 기술 스택
 - Jenkins 2.422
 - Docker 24.0.6
- 개발 환경
 - AWS EC2 Ubuntu 20.04.3 LTS
 - GitLab 16.0.5
 - MatterMost 7.8.6

2. 프로퍼티 정의

A. Jenkinsfile

Back-end 서버

```
pipeline {
    agent any
    tools {

        // If Jenkins has a Gradle installation, you can specify it here
        gradle 'Gradle' // Ensure you have a Gradle tool with this name or adjust accordingly
    }

    stages {
        stage('gitlab clone') {
            steps {
                git branch: 'develop_backend', // Assuming 'develop_backend' is your backend branch
                  credentialsId: 'g',
                  url: 'https://lab.ssafy.com/s09-fintech-finance-sub2/S09P22A308.git'
            }
        }

        stage('Check Java and Gradle') {
            steps {
                sh 'java -version'
                // If Jenkins has a Gradle installation specified above, you can check its version like this:
                sh 'gradle -v'
            }
        }

        stage('Build Spring Boot App') {
            steps {
                dir('dev/BE/'){ // Adjust the directory path accordingly
                    sh 'gradle clean build'
                }
            }
        }

        stage('Build and Run Spring Boot Container') {
            steps {
                dir('dev/BE/'){
                    script {
                        // Build the Docker image
                        sh 'docker build -t spring-boot-app -f Dockerfile.spring .'

                        // Stop and remove old container if it exists
                        sh 'docker rm -f my-spring-boot-container || true'

                        // Run a new container from the new image
                        sh 'docker run -d -p 8000:8000 -v spring_vol:/app/images --name my-spring-boot-container spring-boot-app'
                    }
                }
            }
        }
    }
}
```

Front-end 서버

```
pipeline {
    agent any
    tools {
        nodejs 'node'
    }

    stages {
        stage('gitlab clone') {
            steps {
```

```

        git branch: 'develop_frontend',
        credentialsId: 'g',
        url: 'https://lab.ssafty.com/s09-fintech-finance-sub2/S09P22A308.git'
    }
}

stage('Check npm') {
    steps {
        sh 'echo $PATH'
        sh 'node -v'
    }
}

stage('Build React App') {
    steps {
        dir('dev/FE/'){
            sh 'npm install'
            sh 'npm run build'
        }
    }
}

stage('Build and Run Nginx Container') {
    steps {
        dir('dev/FE/'){
            script {
                // Build the Docker image
                sh 'docker build -t nginx -f Dockerfile.nginx .'

                // Stop and remove old container if it exists
                sh 'docker rm -f my-nginx-container || true'

                // Run a new container from the new image
                sh 'docker run -d -v /path/on/host/nginx-config/nginx.conf:/etc/nginx/nginx.conf -p 80:80 --name my-nginx-container'
            }
        }
    }
}
}
}

```

B. Dockerfile

프로젝트의 `dev/BE` 디렉토리에 `Dockerfile.spring` 이름으로 작성

```

# Use an official OpenJDK runtime as a parent image
FROM openjdk:11-jre-slim

# Set the working directory inside the container
WORKDIR /app

# Copy the JAR file into the image
COPY ./build/libs/my-spring-boot-app.jar /app

# Set the command to run your Spring Boot application
ENTRYPOINT ["java", "-jar", "my-spring-boot-app.jar"]

# Expose port 8000
EXPOSE 8000

```

프로젝트의 `dev/FE` 디렉토리에 `Dockerfile.nginx` 이름으로 작성

```

#Dockerfile
FROM nginx:latest
COPY ./dist /usr/share/nginx/html
# Optionally copy over a custom Nginx config
# COPY ./

```

프로젝트의 `dev/Data/Flask` 디렉토리에 작성

```
# Use an official Python runtime as a parent image
FROM python:3.11-slim

# Set the working directory in docker
WORKDIR /usr/src/app

# Copy the content of the local src directory to the working directory
COPY . .

# Install any needed packages specified in requirements.txt
RUN pip install --no-cache-dir -r requirements.txt

# Make port 80 available to the world outside this container
EXPOSE 4000

# Run app.py when the container launches
CMD ["python", "app.py"]
```

C. Flask server

터미널에서 명령어 입력

```
sudo docker build -t flask1 .

sudo docker run -d -p 4000:4000 --name flask1 flask1
```

D. gitignore

프로젝트의 `root` 디렉토리에 작성

```
### Java ###
# Compiled class file
*.class

# Log file
*.log

# BlueJ files
*.ctxt

# Mobile Tools for Java (J2ME)
.mtj.tmp/

# Package Files #
*.jar
*.war
*.nar
*.ear
*.zip
*.tar.gz
*.rar

# virtual machine crash logs, see http://www.java.com/en/download/help/error_hotspot.xml
hs_err_pid*
replay_pid*

### react ###
.DS_*
logs
**/*.backup.*
**/*.back.*

node_modules
bower_components

*.sublime*

psd
thumb
```

```
sketch

# End of https://www.toptal.com/developers/gitignore/api/java,react
```

프로젝트의 `dev/BE` 디렉토리에 작성

```
HELP.md
.gradle
build/
!gradle/wrapper/gradle-wrapper.jar
!**/src/main/**/build/
!**/src/test/**/build/

### STS ###
.appt_generated
.classpath
.factorypath
.project
.settings
.springBeans
.sts4-cache
bin/
!**/src/main/**/bin/
!**/src/test/**/bin/

### IntelliJ IDEA ###
.idea
*.iws
*.iml
*.ipr
out/
!**/src/main/**/out/
!**/src/test/**/out/

### NetBeans ###
/nbproject/private/
/nbbuild/
/dist/
/nbdist/
/.nb-gradle/

### VS Code ###
.vscode/
```

프로젝트의 `dev/FE` 디렉토리에 작성

```
# Logs
logs
*.log
npm-debug.log*
yarn-debug.log*
yarn-error.log*
pnpm-debug.log*
lerna-debug.log*

node_modules
dist
dist-ssr
*.local

# Editor directories and files
.vscode/*
!.vscode/extensions.json
.idea
.DS_Store
*.suo
*.ntvs*
*.njsproj
*.sln
*.sw?
```


E. FE property

Nginx default 세팅 방법 : 3-c 참고

```
# FE/src/constants/url.ts
export const BASEURL = 'https://i9a308.p.ssafy.io/api/';
```

F. BE property

`application.properties` 내용

```
server.port=8000

spring.servlet.multipart.max-file-size=5MB
spring.servlet.multipart.max-request-size=10MB

server.servlet.context-path=/api/v1
spring.datasource.url= jdbc:mysql://j9a308.p.ssafy.io:3306/backend?serverTimezone=Asia/Seoul&useUnicode=true&characterEncoding=utf8&use
spring.datasource.driver-class-name= com.mysql.cj.jdbc.Driver
spring.datasource.username= root
spring.datasource.password= 1q2w3e4r1!@

spring.jpa.open-in-view=false
spring.jpa.hibernate.ddl-auto = none
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5InnoDBDialect
spring.jpa.show-sql= false
spring.datasource.hikari.maximum-pool-size=20

#spring.redis.lettuce.pool.max-active=
#spring.redis.lettuce.pool.max-idle=
#spring.redis.lettuce.pool.min-idle=

spring.redis.port=6379
spring.redis.host = j9a308.p.ssafy.io
#spring.redis.password= a308
spring.mvc.pathmatch.matching-strategy = ANT_PATH_MATCHER
```

G. 사용 포트

- Nginx 포트 : 80
- Docker SpringBoot 포트 : 8000
 - SpringBoot 포트 : 8000
- Docker Jenkins 포트 : 8080
 - Jenkins 포트 : 8080
- MySQL 포트 : 3306
- Redis 포트 : 6379
- Flask 포트 : 4000

3. 도구 설치 및 설정

A. AWS EC2 설정

방화벽 설정 - ufw

- ufw 활성화 / 비활성화

```
$ sudo ufw enable
$ sudo ufw disable
```

- ufw 상태 확인

```
$ sudo ufw status
```

- ufw 상태 및 등록된 rule 확인

```
$ sudo ufw status numbered
```

- 사용할 포트 허용

```
$ sudo ufw allow [PORT]
```

- 등록된 포트 조회

```
$ sudo ufw show added
```

- 등록된 포트 삭제

(중요) 삭제한 정책은 반드시 enable을 수행해야 적용된다.

```
$ sudo ufw delete [PORT]
```

B. MySQL 설치 및 설정

1. AWS EC2 내에 MySQL 설치

```
# apt(ubuntu에서 사용하는 package 관리 모듈) upgrade
$ sudo apt update
# MySQL 설치
$ sudo apt install mysql-server
# MySQL 버전 확인
$ mysql --version
```

2. MySQL 로그인

```
$ sudo mysql

# 아이디 [id], 비밀번호 [password]로 계정 생성
$ create user 'id'@'%' IDENTIFIED BY 'password';

# 'id' 에 권한 부여. '%'은 모든 외부 IP를 의미
```

```
# 앞 * : DB 이름, 뒤 * : 권한 내용
$ grant all privileges on *.* to 'id'@'%' with grant option;

# mysql 나가기
$ exit

# 방금 만든 admin account로 로그인하기
$ sudo mysql -u [id] -p [password]
```

3. Database & Table 생성

```
# database 생성
$ create database practice default CHARACTER SET UTF8;

# database 생성되었는 지 확인
$ show databases; / show schemas;

# database 접속
$ use practice;
```

```
# table 생성
CREATE TABLE menus (
    menu_id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    menu_name VARCHAR(20) NOT NULL,
    menu_description TEXT NOT NULL
);

# table에 데이터 기입
INSERT INTO menus
    (menu_name, menu_description)
    VALUES
    ("날라날라 바닐라", "베리베리스트로베리 친구");

INSERT INTO menus
    (menu_name, menu_description)
    VALUES
    ("복숭아 아이스티", "내가 제일 좋아하는 티");

INSERT INTO menus
    (menu_name, menu_description)
    VALUES
    ("카페라떼", "Latte is horse");

# table 내용 확인
select * from menus;

# sql 나가기
exit
```

4. Listen IP 대역폭 변경

```
# 현재 mysql은 localhost에서만 접속 가능하므로,
# 모든 IP에서 원격접속할 수 있도록 수정할 예정

$ sudo nano /etc/mysql/mysql.conf.d/mysqld.cnf
```

```
# 127.0.0.1 ← aws의 localhost / 같은 IP인 경우에만 접근이 가능한 상황
# bind-address : local-host 수정

# 주의! 모든 IP에 대해 접근을 허용하므로, 보안 상 문제 --> 나중에 조치를 취할 것

bind-address : 0.0.0.0
```

```
# 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 = 127.0.0.1
#
# * Fine Tuning
#
key_buffer_size = 16M
# max_allowed_packet = 64M
# thread_stack = 256K
#
# thread_cache_size = -1
```

5. MySQL 재시작

```
$ sudo service mysql restart
```

6. Workbench 연결

Setup New Connection

Connection Name: Type a name for the connection

Connection Method: Method to use to connect to the RDBMS

Parameters ☒ SSL ☐ Advanced

Hostname: Port: Name or IP address of the server host - and TCP/IP port.

Username: Name of the user to connect with.

Password: The user's password. Will be requested later if it's not set.

Default Schema: The schema to use as default schema. Leave blank to select it later.

- **Connection Name** : 내가 원하는 이름으로 설정
- **Hostname** : AWS EC2의 `public IPv4(or DNS)` 주소를 기입
- **Username** : 2번에서 작성한 `id` 로 수정
- **Password** : `Store in Vault ...` 를 클릭해서, 2번에서 작성한 `password` 를 기입

C. Nginx 설치 및 설정

1. Nginx 설치

```
$ sudo apt install nginx
```

2. Nginx 상태 확인

```
$ sudo systemctl status nginx
```

3. Nginx 실행 시작 / 중지

```
$ sudo systemctl start nginx  
$ sudo systemctl stop nginx
```

4. Nginx 환경 설정

```
$ sudo vi /etc/nginx/sites-available/default
```

```
server {  
  
    # SSL configuration  
    #  
    # listen 443 ssl default_server;  
    # listen [::]:443 ssl default_server;  
    #  
    # Note: You should disable gzip for SSL traffic.  
    # See: https://bugs.debian.org/773332  
    #  
    # Read up on ssl_ciphers to ensure a secure configuration.  
    # See: https://bugs.debian.org/765782  
    #  
    # Self signed certs generated by the ssl-cert package  
    # Don't use them in a production server!  
    #  
    # include snippets/snakeoil.conf;  
  
    # root /var/www/html;  
    # Front-End build 파일 위치 기입  
    root /var/jenkins_home/workspace/GitLab_CI/dev/FE/dist;  
  
    # Add index.php to the list if you are using PHP  
    index index.html index.htm index.nginx-debian.html;  
  
    # EC2 주소 기입  
    server_name i9a104.p.ssafy.io;  
  
    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://localhost:3000;  
        # proxy_set_header Host $host;  
        # 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;  
        try_files $uri $uri/ /index.html;  
    }  
  
    # Back-End 연결 (10002 포트)  
    location /api/sse/ {  
        proxy_http_version 1.1;  
        proxy_set_header Connection "";  
        proxy_pass http://i9a104.p.ssafy.io:10002/;  
        proxy_hide_header Access-Control-Allow-Origin;  
        add_header 'Access-Control-Allow-Origin' '';  
        proxy_buffering off;  
        proxy_read_timeout 3600000;  
        send_timeout 3600000;  
    }  
}
```

```

        location /api/ {
            proxy_http_version 1.1;
            proxy_set_header    Connection "";
            proxy_pass http://i9a104.p.ssafy.io:10002/;
        }
    }
}

```

5. SSL 설정 (https) - Let's Encrypt & Certbot

```

# Let's Encrypt 설치
$ sudo apt-get install letsencrypt

# Certbot 설치
$ sudo apt-get install certbot python3-certbot-nginx

# Certbot 동작
$ sudo certbot --nginx

## 추가 : 방화벽 기본 포트 설정
$ sudo ufw allow ssh
$ sudo ufw allow http
$ sudo ufw allow https

```

D. Docker 설치 및 설정

Docker Engine 설치

1. 오래된 버전 삭제

```
$ for pkg in docker.io docker-doc docker-compose podman-docker containerd runc; do sudo apt-get remove $pkg; done
```

2. apt repository 셋업

```

# Update the apt package index and install packages to allow apt to use a repository over HTTPS
$ sudo apt-get update
$ sudo apt-get install ca-certificates curl gnupg

# Add Docker's official GPG key
$ sudo install -m 0755 -d /etc/apt/keyrings
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
$ sudo chmod a+r /etc/apt/keyrings/docker.gpg

# Set up the repo
$ echo \
"deb [arch=$(dpkg --print-architecture)] signed-by=/etc/apt/keyrings/docker.gpg https://download.docker.com/linux/ubuntu \
$(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

# Update apt package index
$ sudo apt-get update

```

3. Docker Engine 설치

```
$ sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

4. 설치 확인

```
$ sudo docker run hello-world
```

Docker 설정

- 컨테이너 찾기

```
$ sudo docker ps -a
```

- 컨테이너 접속

```
$ sudo docker exec -it [컨테이너 이름] /bin/bash
```

- 컨테이너 접속종료

```
# exit
```

- 컨테이너 중지

```
$ sudo docker stop [컨테이너 이름]
```

- 컨테이너 삭제

```
$ sudo docker rm -f [컨테이너 이름]
```

- 이미지 생성 (Dockerfile이 있는 디렉토리에서)

```
$ sudo docker build -t [이미지 이름] .
```

- 이미지 찾기

```
$ sudo docker images
```

- 이미지 실행

```
$ sudo docker run --name [생성할 컨테이너 이름] -p [host 포트]:[docker 포트] [이미지 이름]
```

- 이미지 삭제

```
$ sudo docker rmi [이미지 이름]
```

E. Jenkins 설치 및 설정

1. Docker 활용 Jenkins 실행

```
$ sudo docker run -d --name jenkins --restart=on-failure \
-p 8083:8080 \
-v /var/jenkins_home:/var/jenkins_home \
-v /var/run/docker.sock:/var/run/docker.sock \
-e TZ=Asia/Seoul \
-u root \
jenkins/jenkins
```

2. Jenkins 접속 및 라이브러리 설치

<http://i9a104.p.ssafy.io:8083/>

Getting Started

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

password를 찾기 위해, jenkins docker container 내부에 들어가야 한다

```
// Jenkins 컨테이너 접속
# sudo docker exec -it jenkins /bin/bash

// Password 정보 확인
# cat /var/jenkins_home/secrets/initialAdminPassword
```

위에서 얻은 password를 Jenkins 홈 화면에 복사/붙여넣기 한다

이왕 들어간 김에 필요한 라이브러리도 깔아준다

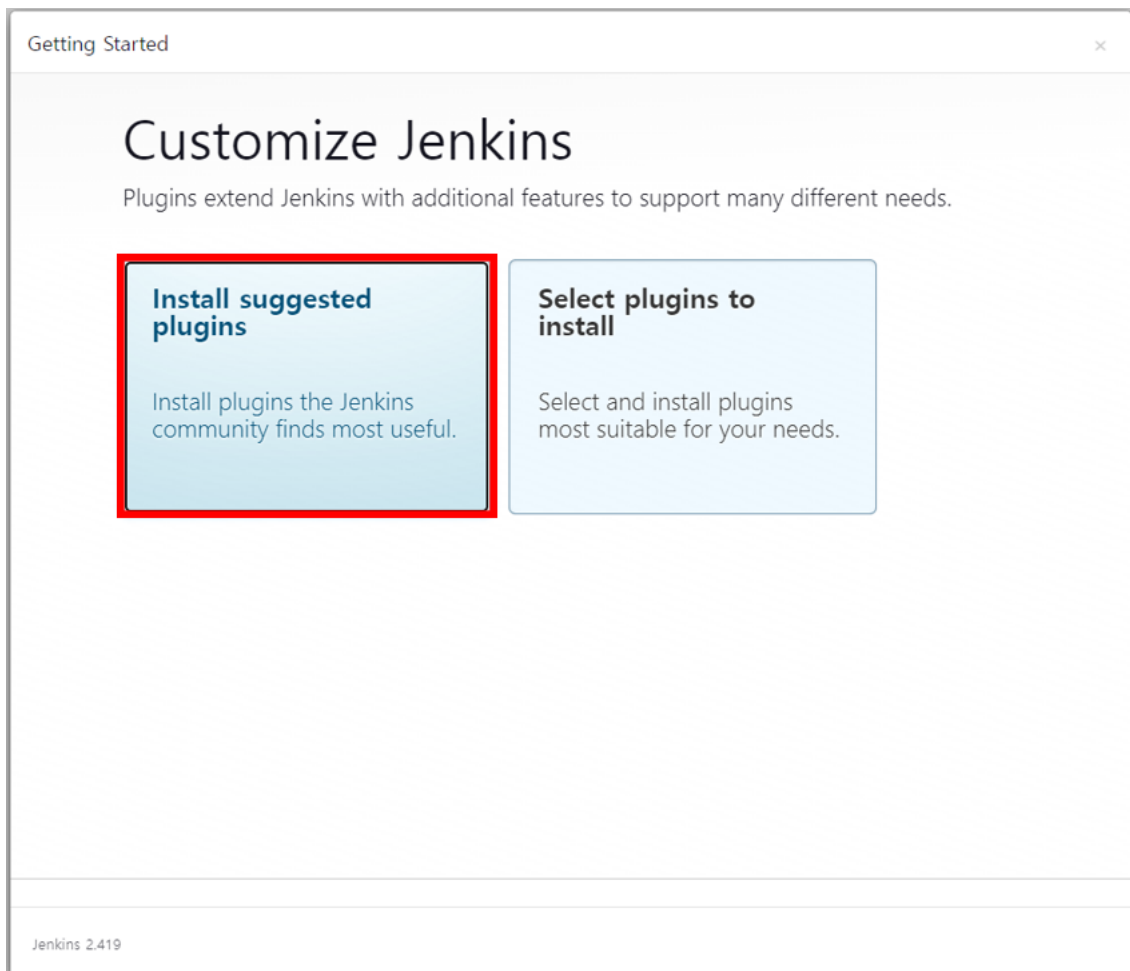

```
// node, npm 설치
# apt update
# curl -fsSL https://deb.nodesource.com/setup_20.x | bash -
# apt install -y nodejs
# npm install -g npm@9.8

// maven 설치
# apt install maven -y

// docker 설치
# curl -fsSL https://get.docker.com -o get-docker.sh
# sh get-docker.sh

// Docker container 나가기
# exit
```

Jenkins 화면 에서 Plugin 설치 진행



3. Jenkins 계정 설정

Getting Started

Create First Admin User

계정명

ssafy1

암호

....

암호 확인

....

이름

forTest

이메일 주소

ssafy@fighting.com|

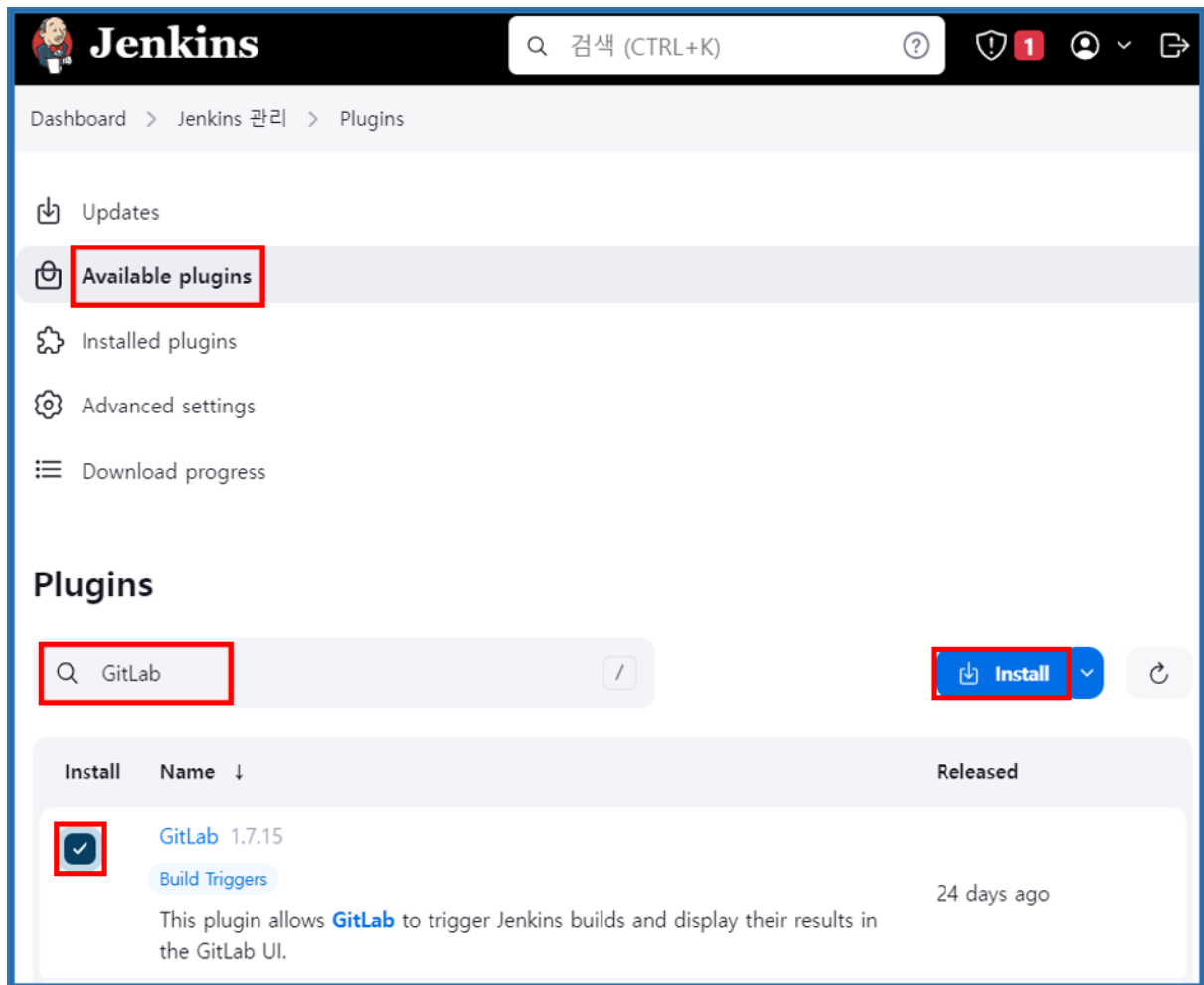
Jenkins 2.419

Skip and continue as admin

Save and Continue

4. Plugin 설치

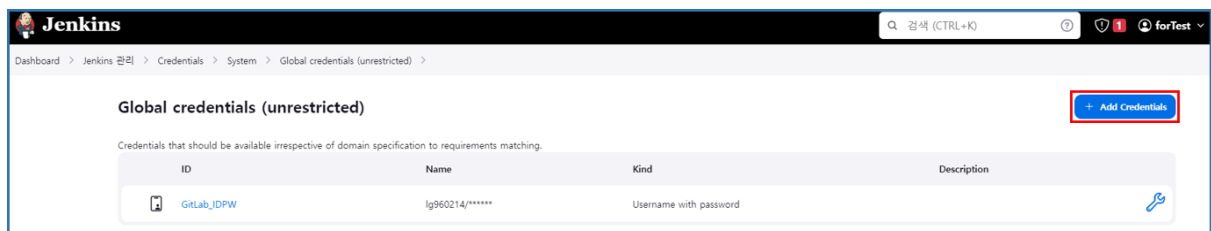
[Jenkins 관리] → [Plugins] → [Available plugins] 접속해서  Plugin 설치



5. Credential 등록 (2가지)

등록 진행하기 전, **GitLab**에서 **API Token** 을 발급받은 후 진행

Username with password 와 **GitLab API Token** 두 가지 모두 설정



New credentials

Kind
Username with password

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

Username ?
lg960214

☐ Treat username as secret ?


Password ?

ID ?
GitLab_JDPW

An internal unique ID by which these credentials are identified from jobs and other configuration. Normally left blank, in which case an ID will be generated, which is fine for jobs created using visual forms. Useful to specify explicitly when using credentials from scripted configuration.
(from [Credentials Plugin](#))

Description ?

- **Username** : GitLab 아이디
- **Password** : GitLab에서 발급받은 API Token
- **ID** : 해당 Credential 등록정보의 타이틀 (아무거나 써도 됨)
- **Description** : ID를 설명하는 세부 내용 (안 써도 됨)



Jenkins

Dashboard > Jenkins 관리 > Credentials > System > Global credentials (unrestricted) >

New credentials

Kind

GitLab API token

Scope ?

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

API token

.....

ID ?

GitLabToken

Description ?

Create

- **API token** : GitLab에서 발급받은 API Token (위와 중복)
- **ID** : 본인이 식별 가능한 타이틀로 설정 (아무거나 써도 됨)

이후 **[Jenkins 관리]** → **[System]** 에서 다음과 같이 GitLab 설정

GitLab

☒ Enable authentication for '/project' end-point ?

GitLab connections

Connection name ?

A name for the connection

GitLab_Connection

GitLab host URL ?

The complete URL to the GitLab server (e.g. http://gitlab.mydomain.com)

https://lab.ssafy.com

Credentials ?

API Token for accessing GitLab

GitLab API token

Add

고급

Success

Test Connection

저장

Apply

Test Connection 을 클릭해서 Success 메세지 확인하면, 저장 버튼 클릭

6. Item 생성 및 빌드 설정

대시보드에서, 새로운 Item 을 눌러, Pipeline 프로젝트 빌드

Jenkins Dashboard > All >

Enter an item name

GitLab_CI
» Required field

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

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
다양한 환경에서의 테스트, 플래폼 특성 빌드, 기타 종종 저렴 다수의 서로다른 환경설정이 필요한 프로젝트에 적합함.

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

Multibranch Pipeline
Creates a set of Pipeline projects according to detected branches in one SCM repository.

OK

아래에 있는 설정을 따라해야 함

General Enabled

설명

Plain text [미리보기](#)

☐ Do not allow concurrent builds

☐ Do not allow the pipeline to resume if the controller restarts

☐ GitHub project

GitLab Connection

GitLab_Connection

아까 설정한 **GitLab Connection**으로 변경

Build Triggers

☐ Build after other projects are built ?

☐ Build periodically ?

☒ Build when a change is pushed to GitLab. GitLab webhook URL: http://19a104.p.ssafy.io:8083/project/GitLab_CI ?

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 ?

Comment (regex) for triggering a build ?

Jenkins please retry a build

URL 기억할 것 (추후 GitLab Webhook에 적용)

GitLab trigger는 필요한 것을 설정

바로 아래에 고급 탭을 눌러 Secret token 생성 (추후 Webhook 시 사용)

Secret token ?

d0c[REDACTED]

Generate

Clear

☐ GitHub hook trigger for GITScm polling ?

☐ Poll SCM ?

☐ Quiet period ?

☐ 빌드를 원격으로 유발 (예: 스크립트 사용) ?

Advanced Project Options

Pipeline

Definition

Pipeline script from SCM

SCM ?

Git

Repositories ?

Repository URL ?

https://lab.ssfy.com/s09-webmobile3-sub2/S09P12A104

Credentials ?

lg960214/*****

Add

고급

Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ?

*/develop

- **Repository URL** : GitLab 프로젝트 주소
- **Credentials** : 위에서 등록한 Credential 정보를 가져옴
- **Branch Specifier** : 감시할 브랜치 명 기입

이후 Save 버튼을 눌러 Pipeline을 저장

7. GitLab Webhook 등록

위의 **Build Triggers** 설정 시 진행했던 **webhook URL**과 **Secret token**을 이용

s09-webmobile3-sub2 > S09P12A104 > Webhook Settings

Search page

Webhooks

Settings

General

Integrations

Webhooks

Access Tokens

Repository

Merge requests

CI/CD

Packages and registries

Monitor

Usage Quotas

URL

http://i9a104.p.ssfy.io:8083/project/GitLab_CI

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

☒ Show full URL

☐ Mask portions of URL

Do not show sensitive data such as tokens in the UI.

Secret token

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

Trigger

☒ Push events

☒ All branches

☐ Wildcard pattern

☐ Regular expression

☐ Tag push events

A new tag is pushed to the repository.

☐ Comments

SSL verification

☒ Enable SSL verification

Save changes

Test ▾

Delete

Recent events

GitLab events trigger webhooks. Use the request details of a webhook to help troubleshoot problems. [How do I troubleshoot?](#)

Status	Trigger	Elapsed time	Request time	
200	Push Hook	0.13 sec	just now	View details

아래와 같이 **Test** 진행 시, **200** 이 뜨면 signal이 jenkins에 잘 도착한 것

8. 빌드 확인

이후로 Event가 일어날 시, **Jenkinsfile**에 적은 Pipeline 대로 진행될 것이다

GitLab_CI

상세 내용 입력

프로젝트 중지하기

Stage View

Average stage times:
(Average full run time: ~4min 43s)

#1

8월 17일 13:02

No Changes

Declarative: Checkout SCM	Remove Previous SpringBoot Settings	SpringBoot Build	Remove Previous SpringBoot Docker	Spring Docker Build and Run	Remove Previous React Settings	React Build	Remove Previous React Docker	React Docker Build and Run
767ms	749ms	3min 42s	1s	3s	1s	28s	13s	6s
767ms	749ms	3min 42s	1s	3s	1s	28s	13s	6s

4. 빌드 방법

A. Back-End

1. 프로젝트에서, `dev/BE` 폴더로 이동
2. 다음 명령어를 통해 빌드

```
$ mvn clean package
```

3. 명령어가 성공하면, 현재 위치한 프로젝트 내에 빌드 파일(.jar)이 위치한 `target` 폴더가 생성

```
ubuntu@ip-10-10-10-10:~/workspace/lg/S09P12A104/dev/BE$ ls
Dockerfile  mvnw  mvnw.cmd  pom.xml  src
ubuntu@ip-10-10-10-10:~/workspace/lg/S09P12A104/dev/BE$ mvn clean package
```

```
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] --- maven-jar-plugin:3.2.2:jar (default-jar) @ A104 ---
[INFO] Building jar: /home/ubuntu/workspace/lg/S09P12A104/dev/BE/target/A104-0.0.1-SNAPSHOT.jar
[INFO]
[INFO] --- spring-boot-maven-plugin:2.7.13:repackage (repackage) @ A104 ---
[INFO] Replacing main artifact with repackaged archive
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 12.792 s
[INFO] Finished at: 2023-08-15T13:26:17Z
[INFO]
ubuntu@ip-10-10-10-10:~/workspace/lg/S09P12A104/dev/BE$ ls
Dockerfile  mvnw  mvnw.cmd  pom.xml  src  target
ubuntu@ip-10-10-10-10:~/workspace/lg/S09P12A104/dev/BE$ cd target
ubuntu@ip-10-10-10-10:~/workspace/lg/S09P12A104/dev/BE/target$ ls
A104-0.0.1-SNAPSHOT.jar      generated-sources  maven-status
A104-0.0.1-SNAPSHOT.jar.original  generated-test-sources  surefire-reports
classes                    maven-archiver     test-classes
```

B. Front-End

1. 프로젝트에서, `dev/FE` 폴더로 이동
2. 다음 명령어를 통해 패키지 다운로드

```
$ npm install
```

3. 패키지 다운로드를 통해 `node_modules` 폴더가 생성됐으면, 다음 명령어를 통해 빌드

```
$ npm run build
```

4. 명령어가 성공하면, 빌드 파일이 위치한 `dist` 폴더가 생성됨

```
ubuntu@ip-172-26-9-135:~/workspace/lg/S09P12A104/dev/FE$ ls
Dockerfile  index.html      package.json    public  tailwind.config.js  tsconfig.node.json
README.md   package-lock.json  postcss.config.js  src    tsconfig.json       vite.config.ts
ubuntu@ip-172-26-9-135:~/workspace/lg/S09P12A104/dev/FE$ npm install
```

```
added 484 packages, and audited 485 packages in 14s
```

```
ubuntu@ip-172-26-9-135:~/workspace/lg/S09P12A104/dev/FE$ ls
Dockerfile  node_modules  postcss.config.js  tailwind.config.js  vite.config.ts
README.md   package-lock.json  public            tsconfig.json
index.html  package.json    src              tsconfig.node.json
ubuntu@ip-172-26-9-135:~/workspace/lg/S09P12A104/dev/FE$ npm run build
```

```
✓ 1057 modules transformed.
dist/index.html                                0.45 kB | gzip: 0.29 kB
dist/assets/Bungee-Regular-4ebf3ee5.ttf       125.12 kB
dist/assets/JejuGothic-49f919f9.ttf          2,460.77 kB
dist/assets/index-1dd6c7b4.css                23.83 kB | gzip: 5.62 kB
dist/assets/index-f19bdb71.js                 816.77 kB | gzip: 265.42 kB

(!) Some chunks are larger than 500 kBs after minification. Consider:
- Using dynamic import() to code-split the application
- Use build.rollupOptions.output.manualChunks to improve chunking: https://rollupjs.org/configuration-options/#output-manualchunks
- Adjust chunk size limit for this warning via build.chunkSizeWarningLimit.
✓ built in 7.35s
ubuntu@ip-172-26-9-135:~/workspace/lg/S09P12A104/dev/FE$ ls
Dockerfile  index.html      package.json    src              tsconfig.node.json
README.md   node_modules  postcss.config.js  tailwind.config.js  vite.config.ts
dist        package-lock.json  public          tsconfig.json
ubuntu@ip-172-26-9-135:~/workspace/lg/S09P12A104/dev/FE$ cd dist
ubuntu@ip-172-26-9-135:~/workspace/lg/S09P12A104/dev/FE/dist$ ls
assets  img  index.html  vite.svg
```