배포 가이드

기술스택

- 형상관리 : Gitlab
- 이슈관리 : Jira
- 커뮤니케이션 : Mattermost
- 디자인 : Figma
- OS : Windows 10
- DB : MySQL 5.7.39
- AWS Cloud EC2

- Ubuntu 20.04.2 LTS
- Docker 20.10.18
- Docker-compose 1.29.2

Back-end [Web & API]

- Springboot
- SpringSecurity
  - JWT
- IDE
  - 이클립스 STS
- JAVA 8
- Gradle
- ORM : Spring Data JPA
- SQL : mysql 5.7.39
- Infra: AWS EC2 Ubuntu Server
  - Tools: Mobaxterm, Putty, Gitbash
  - Docker
  - CI/CD: Jenkins

Back-end [Data]

- Python 3.9
  - Fast-api
  - pandas
  - Scikit-learn

Front-end

- vue 3.2.37
  - vue-router 4.1.2
  - vuex 4.0.2
  - axios 0.27.2
  - vue-axios 3.4.1
  - lodash 4.17.21
- core-js 3.23.4
- element-plus 2.2.9

서버

- AWS
- Docker
- nginx

배포 과정

1. 서버 시간 설정

```
# 1. 서버 시간 확인
date

# 2. 서버 시간 변경
sudo ln -sf /usr/share/zoneinfo/Asia/Seoul /etc/localtime
```

2. 도커 설치

```
#1. apt update
sudo apt update

#2. docker 설치
sudo apt install docker.io

#3. 도커 권한 설정
sudo chmod 666 /var/run/docker.sock
```

3. 도커 컴포즈 설치

```
#1. 다운받을 폴더 생성
mkdir -p ~/.docker/cli-plugins/

#2. 다운받기
curl -SL https://github.com/docker/compose/releases/download/v2.9.0/docker-compose-linux-x86_64 -o ~/.docker/cli-plugins/docker-com
pose

#3. 실행권한 설정
chmod +x ~/.docker/cli-plugins/docker-compose

#4. 링크파일 생성
ln -s ~/.docker/cli-plugins/docker-compose /usr/bin/docker-compose
```

4. mysql 설정

```
#1. 도커 mysql 5.7 이미지 pull
sudo docker pull mysql:5.7

#2. 도커 MYSQL 설치
sudo docker run -d -p 3306:3306 -v ~/mysql:/var/lib/mysql -e MYSQL_ROOT_PASSWORD='3Wooju0!?4' --name mysql5.7 mysql:5.7 --character
-set-server=utf8mb4 --collation-server=utf8mb4_unicode_ci

#3. 도커 mysql 접속
sudo docker exec -it mysql5.7 mysql -u root -p

#4. mysql root 이름 변경
update user set user='wooju_a304' where user='root';


#### 여기서부터 안한것들이긴함

#5. 테이블 생성
CREATE DATABASE moweb DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci;

#6. 계정 생성
create user 'user_a304'@localhost identified by '3Wooju0!?4';
create user 'user_a304'@'%' identified by '3Wooju0!?4';

#7. schema 생성
create database moweb;

#8. moweb 데이터베이스 권한 부여
grant all privileges on moweb.* to 'user_a507'@localhost identified by '3Wooju0!?4';
grant all privileges on moweb.* to 'user_a507'@'%' identified by '3Wooju0!?4';

#9. 권한 확인
show grants for 'user_a304'@localhost;
show grants for 'user_a304'@'%';

#10. 시간 설정
set time_zone='Asia/Seoul';
set global time_zone='Asia/Seoul';
```

```
#11. 디비 변경사항 메모리에 반영
flush privileges;
```

5. nginx 설치

```
#1. apt udpate
sudo apt update

#2. nginx 설치
sudo apt install nginx

#3. nignx 시작
sudo service nginx start
```

6. letsencrypt certbot ssl 발급 및 https 설정

certbot 설치 및 실행

```
#1. apt update
sudo apt udpate

#2. cerbot, certbot nginx 플러그인 설치
sudo apt install certbot python3-cerbot-nginx

#3. nginx 설정에 서버 도메인네임 입력
#3-1. 설정 파일 열기
sudo vi /etc/nginx/sites-enabled/wooju

#3-2. 서버 도메인 네임 입력
server_name j7a304.p.ssafy.io;

#4. nginx 재시작
sudo service nginx restart

#5. certbot 실행
sudo certbot --nginx
```

이메일 입력

dannba1@naver.com

서비스 이용 동의

yes

이메일 수신 동의

yes

발급받을 도메인 네임 선택 or 입력

j7a304.p.ssafy.io

nginx https redirect 자동설정

yes

7. nginx 설정

설정 파일에 아래 프록시 설정 입력

```
server {
        location /{ # /로 시작하는 url은 http://localhost:5173으로 중계(프론트엔드)
                proxy_pass http://localhost:5173;
                proxy_http_version 1.1;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection "Upgrade";
                proxy_set_header Host $host;
```

```
        }
        location /api { # /api로 시작하는 url은 http://localhost:8181/api으로 중계(백엔드)
                proxy_pass http://localhost:8181;
        }
        location /fastapi { # /fastapi로 시작하는 url은 http://localhost:8082/fastapi로 중계(fastapi)
                proxy_pass http://localhost:8082;
        }
        listen 443 ssl;
        server_name j7a304.p.ssafy.io;
        ssl_certificate /etc/letsencrypt/live/j7a304.p.ssafy.io/fullchain.pem;
        ssl_certificate_key /etc/letsencrypt/live/j7a304.p.ssafy.io/privkey.pem;
                                # include /etc/letsencrypt/options-ssl-nginx.conf;
        # ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;
}

server {
        if ($host = j7a304.p.ssafy.io) {
                return 301 https://$host$request_uri;
        }

                listen 80;
                server_name j7a304.p.ssafy.io;
        return 404;
}
```

nginx 재시작

```
# nginx 재시작
sudo service nginx restart
```

8. 프로젝트 git clone

```
git clone https://lab.ssafy.com/s07-bigdata-recom-sub2/S07P22A304
```

9. docker-compose.yml

```
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
```

10. Jenkins 설정

```
# Jenkins 각각의 item 만든 후 모두 돌아가게 설정
```

wooju_back

```
# wooju_back item 생성

# 소스 코드 관리
# Git
# Repository URL
https://lab.ssafy.com/s07-bigdata-recom-sub2/S07P22A304.git

# Credentials
dannba1@naver.com ****** (비밀번호)

# Branches to build
  master
------
```

```
# 빌드 유발

# Build when a change is pushed to GitLab. GitLab webhook URL: http://j7a304.p.ssafy.io:9090/project/wooju_back 체크
# Push Events 체크
# Open Merge Request Events 체크

# 고급 체크

# Secret token 의 Generate 버튼 클릭 후 나오는 토큰 값 저장

# GitLab 의 Webhook에서 URL , Secret token 연결

------
# build steps
# Invoke Gradle script 선택
# User Gradle Wrapper 체크
# Make gradlew executable 체크

# Wrapper location
  ${WORKSPACE}/backend

# 고급 체크
# Root Build script
  ${WORKSPACE}/backend

# build File
  build.gradle

------
# Execute shell

cd ${WORKSPACE}/backend

docker ps -q --filter name=wooju_back | grep -q . && docker stop wooju_back && docker rm wooju_back

docker ps -a -q --filter name=wooju_back | grep -q . && docker rm wooju_back

docker build -t wooju_back_image .
docker run -d -p 8181:8181 --name wooju_back wooju_back_image
```

wooju_fastapi

```
# wooju_fastapi item 생성

# 소스 코드 관리
# Git
# Repository URL
https://lab.ssafy.com/s07-bigdata-recom-sub2/S07P22A304.git

# Credentials
dannba1@naver.com ****** (비밀번호)

# Branches to build
  master

------
# 빌드 유발

# Build when a change is pushed to GitLab. GitLab webhook URL: http://j7a304.p.ssafy.io:9090/project/wooju_fastapi 체크
# Push Events 체크
# Open Merge Request Events 체크

# 고급 체크

# Secret token 의 Generate 버튼 클릭 후 나오는 토큰 값 저장

# GitLab 의 Webhook에서 URL , Secret token 연결

------

# Execute shell

cd ${WORKSPACE}/BePython/

docker ps -q --filter name=wooju_fastapi | grep -q . && docker stop wooju_fastapi && docker rm wooju_fastapi

docker ps -a -q --filter name=wooju_fastapi | grep -q . && docker rm wooju_fastapi

docker build -t wooju_fastapi_image .
docker run -e DBUSER=root -e DBPASSWORD=3Wooju0!?4 -e DBPORT=3306 -e DBNAME=wooju -d -p 8082:8082 --name wooju_fastapi wooju_fastapi_i
```

wooju_front

```
# wooju_front item 생성

# 소스 코드 관리
# Git
# Repository URL
https://lab.ssafy.com/s07-bigdata-recom-sub2/S07P22A304.git

# Credentials
dannba1@naver.com ****** (비밀번호)

# Branches to build
  master

------
# 빌드 유발

# Build when a change is pushed to GitLab. GitLab webhook URL: http://j7a304.p.ssafy.io:9090/project/wooju_front 체크
# Push Events 체크
# Open Merge Request Events 체크

# 고급 체크

# Secret token 의 Generate 버튼 클릭 후 나오는 토큰 값 저장

# GitLab 의 Webhook에서 URL , Secret token 연결

------

# Execute shell

cd ${WORKSPACE}/frontend/

docker ps -q --filter name=wooju_front | grep -q . && docker stop wooju_front && docker rm wooju_front

docker ps -a -q --filter name=wooju_front | grep -q . && docker rm wooju_front

docker build -t wooju_front_image .
docker run -d -p 5173:5173 --name wooju_front wooju_front_image
```

11. 전체 설정파일

1. nginx 설정파일

```
server {
        location /{ # /로 시작하는 url은 http://localhost:5173으로 중계(프론트엔드)
                proxy_pass http://localhost:5173;
                proxy_http_version 1.1;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection "Upgrade";
                proxy_set_header Host $host;
        }
        location /api { # /api로 시작하는 url은 http://localhost:8181/api으로 중계(백엔드)
                proxy_pass http://localhost:8181;
        }
        location /fastapi { # /fastapi로 시작하는 url은 http://localhost:8082/fastapi로 중계(fastapi)
                proxy_pass http://localhost:8082;
        }
        listen 443 ssl;
        server_name j7a304.p.ssafy.io;
        ssl_certificate /etc/letsencrypt/live/j7a304.p.ssafy.io/fullchain.pem;
        ssl_certificate_key /etc/letsencrypt/live/j7a304.p.ssafy.io/privkey.pem;
                            # include /etc/letsencrypt/options-ssl-nginx.conf;
        # ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;
}

server {
        if ($host = j7a304.p.ssafy.io) {
                return 301 https://$host$request_uri;
        }

                listen 80;
                server_name j7a304.p.ssafy.io;
```

```
        return 404;
}
```

2. 환경변수

```
# 백엔드 api swagger 진입
http://j7a304.p.ssafy.io/api/swagger-ui/

# 루트 서버 url
https://j7a304.p.ssafy.io/

# 젠킨스 url
http://j7a304.p.ssafy.io:9090/

# 백엔드 fastapi url
https://j7a304.p.ssafy.io/fastapi/

# 데이터베이스 이름
db_dbname = wooju

# 데이터베이스 패스워드
db_password = 3Wooju0!?4

# 데이터베이스 url
db_url = localhost:3306

# 데이터베이스 아이디
db_username = root
```

프론트엔드

1. npm install

```
# 패키지 설치
npm install
```

2. 개발환경 실행

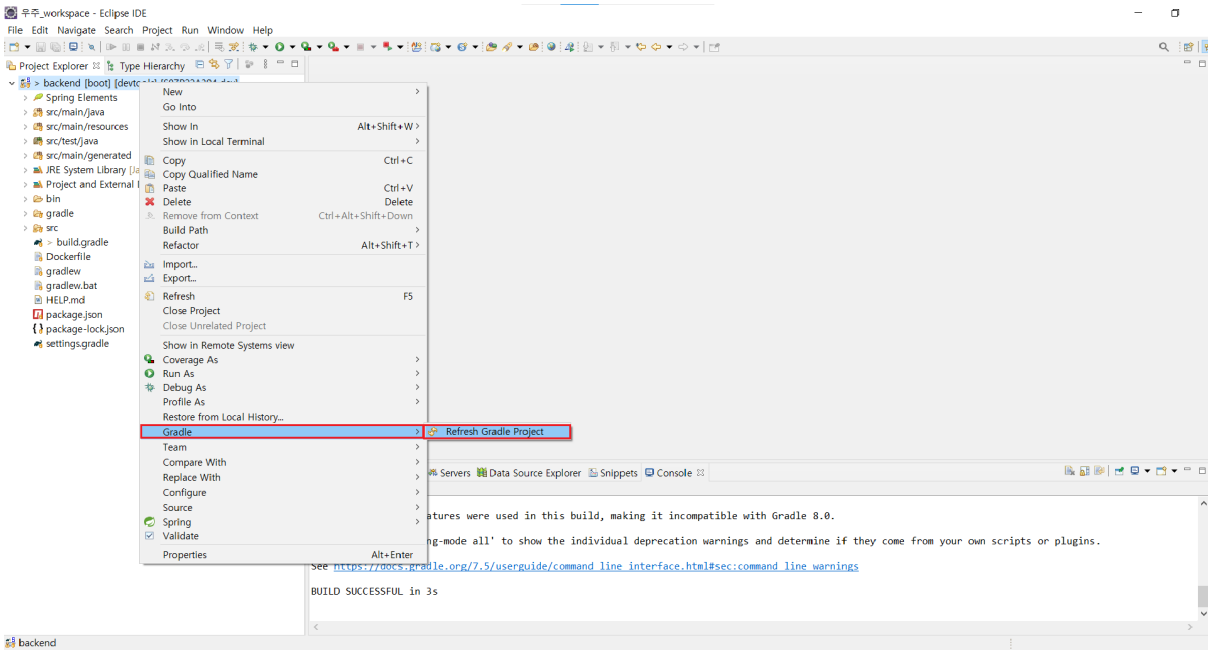
```
# 5173 포트로 실행됨
npm run dev
```

3. 빌드

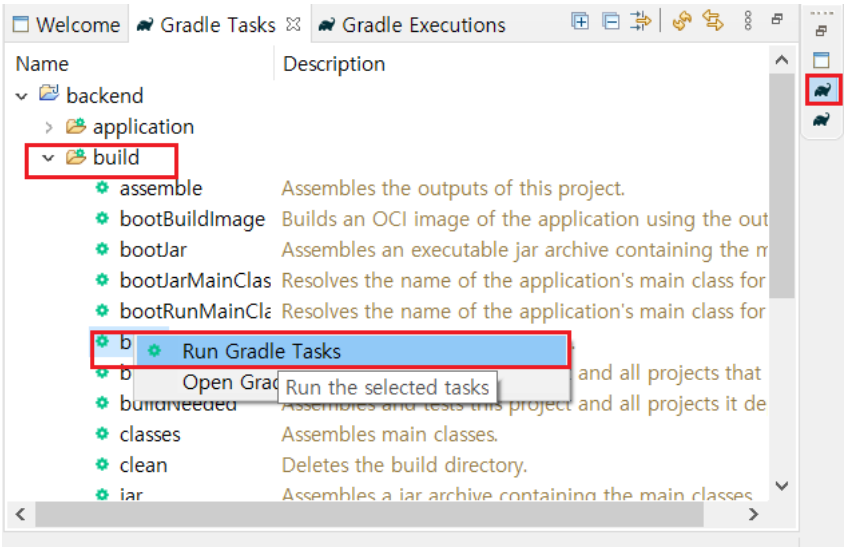
```
npm run build
```

백엔드

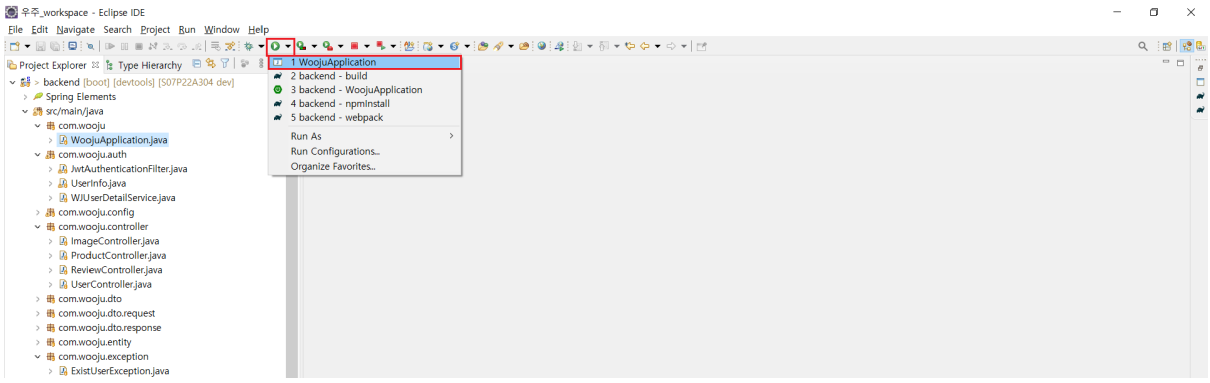
1. 개발 진행 과정



1. project 우클릭 > Gradle> Refresh GradleProject



2. 우측 Gradle 아이콘 클릭 > 프로젝트 내부 build클릭> build 우클릭> Run Gradle Tasks 실행



3. WoojuApplication.java 실행

2. command

```
# 빌드
./gradlew clean build
cd ./build/libs
ls backend-1.0-SNAPSHOT.jar
```

데이터 (Fastapi)

1. 가상환경

```
python -m venv venv
source venv/Script/activate
pip install -r requirements.txt
```

2. 실행

```
python app/main.py
```