알고리즘 스터디 프로젝트 Aldy 포팅 매뉴얼

팀: C206 ALDY

팀원: 박세호(팀장), 조성민, 홍석호 (BE), 문서희, 박세은, 이동주 (FE)

프로젝트 기간 : 2022년 10월 10일 ~ 2022년 11월 18일 (6주)

**1. 프로젝트 기술 스택**

1.1 Backend

* SpringBoot
* Gradle
* Spring Data JPA
* Spring Batch
* SMTP
* MariaDB
* Redis
* Spring Security

1.2 Frontend

* React
* Recoil
* Scss

1.3 형상 관리

* Gitlab
* Git

1.4 이슈 관리

* Jira

1.5 커뮤니케이션

* Mattermost
* Discord
* Webex
* Notion

1.6 데이터베이스

* MariaDB

1.7 서버

* AWS EC2
* Ubuntu 20.04 LTS

1.8 배포

* Docker
* Jenkins
* Nginx

**2. 개발환경**

2.1 운영체제

* 윈도우

2.2 IDE

* IntelliJ
* Visual Studio Code

**3. 배포**

백엔드 브랜치, 프론트엔드 브랜치 각각을 젠킨스 아이템과 연동 후 각각의 브랜치에 푸시, 머지 이벤트 발생 시 빌드하여 도커 이미지화 후 컨테이너로 실행

따라서 각각의 브랜치를 깃랩 웹훅을 이용하여 젠킨스와 연결하는 것이 우선.

3.1 도커 설치 명령어

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

3.2 젠킨스 설치 및 실행 명령어

docker run --privileged -d -p 8085:8080 -p 50000:50000 -v /jenkins:/var/jenkins\_home -e TZ=Asia/Seoul -v /var/run/docker.sock:/var/run/docker.sock -u root --name jenkins jenkins/jenkins:lts

포트번호는 8085, 시간대는 서울, 컨테이너 이름은 jenkins로 설정. 이후 젠킨스 컨테이너 내에서 docker 설치해주어야 함.

3.3 젠킨스 컨테이너 내에서 실행하는 도커 설치 명령어

apt-get remove docker docker-engine docker.io containerd runc

apt-get update apt-get install \ ca-certificates \ curl \ gnupg \ lsb-release mkdir -p /etc/apt/keyrings curl -fsSL <https://download.docker.com/linux/debian/gpg> | gpg --dearmor -o /etc/apt/keyrings/docker.gpg

echo \ "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] <https://download.docker.com/linux/debian> \ $(lsb\_release -cs) stable" | tee /etc/apt/sources.list.d/docker.list > /dev/null

apt-get update apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin

3.4 마리아 디비 설치 및 실행 명령어

docker run --name mariadb -d -p 3306:3306 --restart=always -e MYSQL\_ROOT\_PASSWORD=c206 mariadb

비밀번호는 c206으로 backend 폴더 내 main/resources/application.yml 파일 내 mariadb password 부분과 일치하여야 함.

3.5 레디스 설치 및 실행 명령어

docker run -d --name redis -p 6379:6379 redis --requirepass c206

비밀번호는 c206, --requirepass 뒤에 오는 문자열이 비밀번호로써 설정 안할 시 해킹 위험 있음.

Backend 폴더 내 main/resources 의 application.yml 파일에 redis의 password 부분과 일치하여야 레디스 정상 작동함.

3.6 젠킨스 백엔드설정 파일 Backend Jenkinsfile

pipeline {

agent any // 아무 에이전트에서 해당 파이프라인 또는 stage를 실행함.

stages {

stage('git clone'){

steps{

echo 'git clone'

git branch: ‘[원하는 백엔드 브랜치 입력]’, credentialsId: ‘[젠킨스 자격증명 키 입력]’, url: ‘[깃랩 혹은 깃허브 주소 입력]’

}

}

stage('Build Gradle') {

steps {

echo 'Build Gradle'

dir ('./backend/Aldy'){

sh 'chmod +x gradlew'

sh 'chmod +x dev.sh'

sh './gradlew clean build'

}

}

post {

success {

echo 'Successfully Build Gradle'

}

failure {

error 'This pipeline stops here...'

}

}

}

stage('Deploy') {

steps {

echo 'hello start'

sshagent (credentials: ‘[EC2 서버 SSH 연결을 위한 자격증명 ID 입력]’) {

sh """

ssh -o StrictHostKeyChecking=no [아이디@호스트] '

cd /jenkins/workspace/ssafy/backend/Aldy

docker stop spring

docker rm spring

docker rmi spring-server

docker build -t spring-server .

docker run -d -p 8081:8080 --name=spring spring-server

'

"""

}

}

}

}

}

3.7 프론트엔드 젠킨스 설정파일 FrontEnd Jenkinsfile

pipeline {

agent any // 아무 에이전트에서 해당 파이프라인 또는 stage를 실행함.

stages {

stage('git clone'){

steps{

echo 'git clone'

git branch: ‘[프론트엔드 브랜치 이름 입력]’, credentialsId: '[자격증명 키 이름 입력]', url: '[깃랩 혹은 깃허브 주소 입력]'

}

}

stage("build"){

steps{

dir ('./aldy'){

script{

try {

sh 'docker stop front'

sh 'docker rm front'

sh 'docker rmi front'

} catch (e) {

sh 'echo "front stop중 실패"'

}

try{

sh 'docker build -t front .'

}catch(e){

sh 'echo "docker 이미지빌드중 실패"'

}

}

}

}

post {

success{

sh 'echo build 성공'

}

failure{

sh 'echo build 실패'

}

}

}

stage("run"){

steps {

sh 'docker run -itd -p 3000:3000 --name front front'

}

}

}

}

백엔드 브랜치와 프론트엔드 브랜치에 포함되어 있는 Dockerfile 로 각각 빌드되어 배포됨.

그 전에 백엔드 브랜치에서 사용하는 레디스, MariaDB 관련 이미지 풀 후 컨테이너로 실행 필요

3.8 Nginx 설치 및 실행 명령어

sudo apt install nginx –y

sudo service nginx start

3.9 https 적용하기 위해 certbot 설치 및 실행

wget https://dl.eff.org/certbot-auto

sudo snap install core

sudo apt remove certbot

sudo snap install --classic certbot

ln -s /snap/bin/certbot /usr/bin/certbot

sudo certbot --nginx

필요에 따라 nginx 설정할 것.

현재 프론트엔드 코드는 백엔드 서버에 https로 요청을 보내고 있음.

백엔드 서버에 https가 적용되어 있기 때문. 그렇기 때문에 nginx를 고려해서 포팅해야 함.

3.9 Nginx 설정파일 aldy.conf 내용

server {

server\_name k7c206.p.ssafy.io

location /api {

proxy\_pass http://localhost:8081;

}

location / {

proxy\_pass http://localhost:3000;

}

listen 443 ssl; # managed by Certbot

ssl\_certificate /etc/letsencrypt/live/www.aldy.co.kr/fullchain.pem; # managed by Certbot

ssl\_certificate\_key /etc/letsencrypt/live/www.aldy.co.kr/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 {

listen 8086;

location / {

proxy\_pass http://localhost:8085;

proxy\_redirect off;

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-Host $server\_name;

}

}

server {

if ($host = k7c206.p.ssafy.io) {

return 301 https://$host$request\_uri;

} # managed by Certbot

listen 80;

server\_name k7c206.p.ssafy.io;

return 404; # managed by Certbot

}

전체 배포 순서

도커 설치 -> 젠킨스 설치 -> 깃랩과 연동 -> 레디스, 마리아디비 설치 -> nginx 설치 및 https 적용