|  |
| --- |
|  |



|  |
| --- |
| **Lighthouse** |
| **포팅 메뉴얼** |
|  |
|  |
|  |
|  |
|  |

목차

[I. 개요 2](#_Toc96072459)

[1. 프로젝트 사용 도구 2](#_Toc96072461)

[2. 개발환경 2](#_Toc96072462)

[3. 외부 서비스 2](#_Toc96072463)

[4. 환경변수 처리한 핵심 키들 3](#_Toc96072464)

[II. 빌드 3](#_Toc96072465)

[1. 환경변수 형태 3](#_Toc96072466)

[2. 빌드하기 5](#_Toc96072467)

[3. 배포하기 5](#_Toc96072468)

[4. 서비스 이용 방법 6](#_Toc96072469)

[가) Google 클라우드 플랫폼 6](#_Toc96072470)

[나) S3와 Cloudfront 연동 7](#_Toc96072471)

# 개요

## 프로젝트 사용 도구

이슈 관리 : JIRA

형상 관리 : Gitlab

커뮤니케이션 : Notion, Mattermost

디자인 : Figma

UCC : Movavi

CI/CD : Jenkins, Docker

## 개발환경

VS Code : 1.81.1,

IntelliJ : IDEA 2023.1.3 (Runtime version: 17.0.7+10-b829.16 amd64, VM: OpenJDK 64-Bit Server VM by JetBrains s.r.o)

JVM : 1.8.0\_372

Node.js : 18.17.0

SERVER : AWS EC2 Ubuntu 20.04 LTS

DB : MariaDB (azure 10.3.23-MariaDB), MongoDB

Kafka : 3.1.0

## 외부 서비스

Google OAuth : application.properties 에 해당 내용 있음

AWS S3 : application.properties 에 해당 내용 있음

AWS CloudFront : .env 에 해당 내용 있음.

## 환경변수 처리한 핵심 키들

React: .env

Spring: 서버 내 ~/config/.env 파일에 존재

# 빌드

## 환경변수 형태

**.env:**

// MariaDB

DATASOURCE\_URL={DB 주소}

DATASOURCE\_USERNAME= {DB 호스트 유저 이름}

DATASOURCE\_PASSWORD= {{DB 비밀번호}

// MondoDB

MONGODB\_URI= {DB 주소}

SALT= {JWT Salt}

// S3

AWS\_ACCESSKEY= {S3 접근 키}

AWS\_SECRETKEY= {S3 암호 키}

// Google Oauth

GOOGLE\_CLIENT\_ID= {구글 클라이언트 아이디}

GOOGLE\_CLIENT\_SECRET= {구글 시크릿 키}

REDIRECT\_URI={리다이렉트 주소}

// React

REACT\_APP\_API\_URL= {백엔드 서버 주소}

REACT\_APP\_GOOGLE\_CLIENT\_ID={구글 클라이언트 아이디}

KAFKA\_BROKER= {카프카 서버 주소}

SSL\_PASSWORD= {SSL 비밀번호}

## 빌드하기

1) Front: React

npm i

npm start

2) Back: Spring Boot

Gradle 실행

Bootjar 실행

## 배포하기

**Jenkinsfile**

pipeline {

agent any

environment {

VERSION = "latest"

DOCKERHUB\_REPOSITORY\_BACK = {도커허브이름/백엔드저장소이름}

DOCKERHUB\_REPOSITORY\_FRONT = {도커허브이름/프론트저장소이름}

DOCKERHUB\_CREDENTIAL = credentials('dockerhub-imsong')

CONTAINER\_NAME\_BACK = {백엔드 컨테이너 이름}

CONTAINER\_NAME\_FRONT = {프론트 컨테이너 이름}

SSH\_CONNECTION = {서버 주소}

ENV\_DIR = {서버 내 환경변수 저장 위치}

PORT\_BACK = {백엔드 포트}

PORT\_FRONT = "{프론트 포트}"

}

stages {

stage('Build Backend') {

steps {

dir('backend/lighthouse') {

sh "chmod +x gradlew"

sh "./gradlew clean compileJava bootJar"

}

}

}

stage("Build Images") {

steps {

sh "docker compose build"

}

}

stage('Push Images'){

steps {

sh "echo $DOCKERHUB\_CREDENTIAL\_PSW | docker login -u $DOCKERHUB\_CREDENTIAL\_USR --password-stdin"

sh "docker push $DOCKERHUB\_REPOSITORY\_BACK:$VERSION"

sh "docker push $DOCKERHUB\_REPOSITORY\_FRONT:$VERSION"

}

}

stage('Deploy Backend Server') {

steps {

sshagent(credentials: ['ec2']) {

sh "ssh -o StrictHostKeyChecking=no $SSH\_CONNECTION 'docker rm -f $CONTAINER\_NAME\_BACK'"

sh "ssh -o StrictHostKeyChecking=no $SSH\_CONNECTION 'docker rmi -f $DOCKERHUB\_REPOSITORY\_BACK:$VERSION'"

sh "ssh -o StrictHostKeyChecking=no $SSH\_CONNECTION 'docker pull $DOCKERHUB\_REPOSITORY\_BACK:$VERSION'"

sh "ssh -o StrictHostKeyChecking=no $SSH\_CONNECTION 'echo y | docker image prune'"

sh "ssh -o StrictHostKeyChecking=no $SSH\_CONNECTION 'docker images'"

sh "ssh -o StrictHostKeyChecking=no $SSH\_CONNECTION 'docker run -d --name $CONTAINER\_NAME\_BACK --env-file $ENV\_DIR -p $PORT\_BACK:8080 $DOCKERHUB\_REPOSITORY\_BACK:$VERSION'"

sh "ssh -o StrictHostKeyChecking=no $SSH\_CONNECTION 'docker ps'"

}

}

}

stage('Deploy Frontend Server') {

steps {

sshagent(credentials: ['ec2']) {

sh "ssh -o StrictHostKeyChecking=no $SSH\_CONNECTION 'docker rm -f $CONTAINER\_NAME\_FRONT'"

sh "ssh -o StrictHostKeyChecking=no $SSH\_CONNECTION 'docker rmi -f $DOCKERHUB\_REPOSITORY\_FRONT:$VERSION'"

sh "ssh -o StrictHostKeyChecking=no $SSH\_CONNECTION 'docker pull $DOCKERHUB\_REPOSITORY\_FRONT:$VERSION'"

sh "ssh -o StrictHostKeyChecking=no $SSH\_CONNECTION 'echo y | docker image prune'"

sh "ssh -o StrictHostKeyChecking=no $SSH\_CONNECTION 'docker images'"

sh "ssh -o StrictHostKeyChecking=no $SSH\_CONNECTION 'docker run -d --name $CONTAINER\_NAME\_FRONT --env-file $ENV\_DIR -p $PORT\_FRONT:3000 $DOCKERHUB\_REPOSITORY\_FRONT:$VERSION'"

sh "ssh -o StrictHostKeyChecking=no $SSH\_CONNECTION 'docker ps'"

}

}

}

}

}

## 서비스 이용 방법

### Google 클라우드 플랫폼(GCP)

1. console.cloud.google.com에 접속 → 새 프로젝트 생성

2. OAuth 동의 화면에서 User Type 외부로 설정

3. 앱 정보 등록(앱 이름, 이메일)

4. 사용자 인증 정보 → 사용자 인증 정보 만들기 → OAuth 클라이언트 ID

5. 승인된 자바스크립트 원본에 localhost, localhost:3000, 프론트 서버 주소 추가

6. 승인된 리다이렉션 URI에 local callback API 주소와 백엔드 서버 주소 추가

### S3와 CloudFront 연동

1. S3 버킷 생성

2. CloudFront → create Distribution → 배포 방식: Web 선택

3. 배포 설정(공식문서 참고)

https://docs.aws.amazon.com/ko\_kr/AmazonCloudFront/latest/DeveloperGuide/distribution-web-values-specify.html