RasingDust 포팅메뉴얼

1. 개발환경

1.1 기술스택

Frontend

- React 18.2.0
- Node.js 20.11.0
- NPM 10.2.4
- Vite 5.0.8
- Typescript 5.2.2
- React Query 5.17.15

- React Zustand 4.5.0
- React Router DOM 6.21.3
- Axios 1.6.7
- Tailwind CSS 3.4.1

Backend

- SpringBoot 3.2.1
- Java 21
- Mysql 5.7
- Spring Security
- Spring Data JPA
- JWT
- OAuth2
- Swaager

Tool

- Notion
- GitLab / Git
- JIRA
- Postman
- Mattermost

1.2 설정 파일

Frontend

- .env

```
VITE_OPENAI_API_KEY={api key}
VITE_API_BASE_URL='https://j10d103.p.ssafy.io/api'
VITE_BASE_URL=http://j10d103.p.ssafy.io/api
```

```
VITE_DEV_USER_TOKEN={개발용 JWT}
VITE_PAGE_SIZE=12
```

Backend

- application-key.yaml

```
my-db:
  url: 로컬 테스트용 db
  username: db 아이디
  password: db 비밀번호
  driver-class-name: com.mysql.cj.jdbc.Driver

encryptor:
  key: {Jasypt 암호화 키}
```

Infra

- Jenkins 파이프라인

```
pipeline {
    tools {
        jdk("JDK21")
        nodejs("20.11.1")
    }
    agent any

    stages {
        stage('Clone') {
            steps {
                git branch: 'develop',
                credentialsId: 'rasingdust',
                url: 'https://lab.ssafy.com/s10-ai-image-su
b2/S10P22D103.git'
            }
        }
        stage('Build') {
            parallel {
```

```
                stage('Build FrontEnd') {
                    steps {
                        catchError(buildResult: 'SUCCESS',
stageResult: 'FAILURE') {
                            echo 'get credential files...'
                            withCredentials([file(credentia
lsId: 'react-env', variable: 'reactKey')]) {
                                script {
                                    sh 'cp $reactKey fronte
nd/.env'
                                    sh 'chown jenkins:jenki
ns frontend/.env'
                                    sh 'chmod 644 fronten
d/.env'
                                }
                            }

                            echo 'executing npm...'
                            dir('frontend') {
                                sh "rm -rf dist"
                                sh "rm -rf node_modules"
                                sh "rm -rf package-lock.jso
n"
                                sh 'npm install'
                                sh 'npm run build'
                                }
                        }
                    }
                }
                stage('Build BackEnd') {
                    steps {
                            echo 'get credential files...'
                            withCredentials([file(credentia
lsId: 'spring-key', variable: 'appKey')]) {
                                script {
                                    sh 'cp $appKey backend/
src/main/resources/application-key.yaml'
                                    sh 'chown jenkins:jenki
```

```
ns backend/src/main/resources/application-key.yaml'
                                    sh 'chmod 644 backend/s
rc/main/resources/application-key.yaml'

                                }
                            }

                            echo 'executing gradle...'
                            withGradle() {
                                sh 'chmod +x ./backend/grad
lew'
                                sh './backend/gradlew clean
build -Dscan -p ./backend -Dspring.profiles.active=server'
                            }


                    }
                }
            }
        }
        stage('Docker Delete') {
            parallel {
                stage('Delete Old front') {
                    steps {
                        sh "docker rm -f dust_frontend"
                        sh "docker rmi -f dust_frontend"
                    }
                }
                stage('Delete Old back') {
                    steps {
                        sh "docker rm -f dust_backend"
                        sh "docker rmi -f dust_backend"
                    }
                }
            }
        }
        stage('Docker Build') {
            parallel {
```

```
                stage('docker build front') {
                    steps {
                        sh 'docker build -t dust_frontend
./frontend '
                    }
                }
                stage('docker build back') {
                    steps {
                        sh 'docker build -t dust_backend ./
backend '
                    }
                }
            }
        }
        stage('Deploy') {
            steps {
                sh 'docker-compose -f compose.yml up -d'
            }
        }
    }
    post {
        success {
            script {
                def Author_ID = sh(script: "git show -s --p
retty=%an", returnStdout: true).trim()
                def Author_Name = sh(script: "git show -s -
-pretty=%ae", returnStdout: true).trim()
                mattermostSend (color: 'good',
                message: "빌드 성공: ${env.JOB_NAME} #${env.B
UILD_NUMBER} by ${Author_ID}(${Author_Name})\n(<${env.BUILD
_URL}|Details>)",
                )
            }
        }
        failure {
            script {
                def Author_ID = sh(script: "git show -s --p
retty=%an", returnStdout: true).trim()
```

```
                def Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout: true).trim()
                mattermostSend (color: 'danger',
                message: "빌드 실패: ${env.JOB_NAME} #${env.BUILD_NUMBER} by ${Author_ID}(${Author_Name})\n(<${env.BUILD_URL}|Details>)",
                )
            }
        }
    }
}
```

2. 배포 방법

2.1 EC2

2.1.1 방화벽 설정

```
sudo ufw allow 80
sudo ufw allow 8080
sudo ufw allow 443
sudo ufw enable
sudo ufw status
```

2.1.2 도커 설치

```
sudo apt update
sudo apt install apt-transport-https ca-certificates curl software-properties-commonsudo apt install apt-transport-https ca-certificates curl software-properties-common
//도커 레포 접근 키 설정
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
// 도커 레포 등록
sudo add-apt-repository "deb [arch=amd64] https://download.
```

```
docker.com/linux/ubuntu bionic stable"
sudo apt update
//도커 설치
sudo apt install docker-ce
//실행 확인
sudo systemctl status docker
```

2.1.3 도커 컴포즈 설치

```
sudo curl -L "https://github.com/docker/compose/releases/do
wnload/1.28.2/docker-compose-$(uname -s)-$(uname -m)" -o /u
sr/local/bin/docker-composesudo curl -L "https://github.co
m/docker/compose/releases/download/1.28.2/docker-compose
-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
//권한 부여
sudo chmod +x /usr/local/bin/docker-compose
//심볼릭 링크
sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-co
mpose
//버전 확인
docker-compose --version
```

2.1.4 젠킨스 설치

```
// 젠킨스 이미지 실행
sudo docker run -d -p 8080:8080 -v /home/ubuntu/jenkins-dat
a:/var/jenkins_home -v /var/run/docker.sock:/var/run/docke
r.sock -v /usr/bin/docker:/usr/bin/docker -v /usr/bin/docke
r-compose:/usr/bin/docker-compose --name jenkins jenkins/je
nkins:lts

// 안되면 sudo chown -R 1000 ./jenkins-data 폴더 권한주기

//권한
sudo chmod 666 /var/run/docker.sock
sudo chmod a+rx /usr/local/bin/docker-compose
```

```
//젠킨스 비밀번호 얻기
docker exec -it jenkins bash -c "cat /var/jenkins_home/secr
ets/initialAdminPassword"
```

2.1.5 ssl 인증서발급

```
sudo apt-get install letsencrypt
//인증서 설치
sudo letsencrypt certonly --standalone -d <도메인>
```

2.1.6 MySQL 도커 컨테이너 실행

```
docker pull mysql

docker volume create mysql-volume

docker run -d --name mysql-container -p 3306:3306 -v mysql-
volume:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=비번 mysql:5.7
--character-set-server=utf8mb4 --collation-server=utf8mb4_u
nicode_ci

//접속
docker exec -it mysql-container bash

$ mysql -u root -p
mysql> CREATE DATABASE test;
mysql> SHOW DATABASES;
```

2.2 GitLab webHook

s10-ai-image-sub2 / S10P22D103 / Webhook Settings / **Webhook**

Search page

Webhook

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://j10d103.p.ssafy.io:8080/project/RasingDust

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

- [x] Push events
  - ( ) All branches
  - (•) Wildcard pattern
    develop
    Wildcards such as `*-stable` or `production/*` are supported.
  - ( ) Regular expression
- [ ] Tag push events
  A new tag is pushed to the repository.