빌드 배포 환경

Front

- React 18.2
- Electron 28.2.6
- Vite 5.1.4
- Tailwind CSS 3.4.1
- Redux 9.1

Backend

1) Django (5.0.3)

- Python (3.12.2)
- KoNLPy (0.6.0)
- Scikit Learn (1.4.1)
- NumPy (1.26.4)
- Pandas (2.2.1)
- SciPy (1.12.0)
-

2) Spring Boot(3.2.3)

- Java 17
- JWT(0.11.2)
- JAVA Spring Data JPA
- Spring Security(6.2.2)
- OAuth(6.2.2)

3) DB

- MySQL(8.0.33) CI/CD

4) CI/CD

- AWS EC2(Ubuntu 20.04.6 LTS)
- Jenkins(2.449)
- Nginx(1.25.4)

- AWS RDS(8.0.35)
- Docker(25.0.4)
- Docke-Compose(1.27.4)

도커 컴포즈 파일

```
version: "3"
services:
  backend:
    container_name: springboot
    build: ./backend/springboot
    image: back_image:0.0
    ports:
      - "8080:8080"

    networks:
      - backend_frontend

  frontend:
    container_name: frontend
    build: ./frontend
    image: front_image:0.0
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - /etc/letsencrypt:/etc/letsencrypt
      - /var/lib/letsencrypt:/var/lib/letsencrypt
      - /home/ubuntu/nginx/conf.d:/etc/nginx/conf.d
      - /home/ubuntu/nginx/sites/sites-enabled:/etc/nginx/sites-enabled

    networks:
      - backend_frontend

networks:
  backend_frontend:
    driver: bridge
```

도커파일

Front

```
FROM node:latest as builder

WORKDIR /app
COPY package.json .
COPY package-lock.json .
COPY . .
RUN npm install
RUN npm run build

# nginx 이미지를 사용합니다. 뒤에 tag가 없으면 latest 를 사용합니다.
FROM nginx

COPY --from=builder /app/dist /usr/share/nginx/html

# container 실행 시 자동으로 실행할 command. nginx 시작함
CMD ["nginx", "-g", "daemon off;"]
```

Backend

```
FROM eclipse-temurin:17-jdk
VOLUME /tmp
ARG JAR_FILE=build/libs/springboot-0.0.1-SNAPSHOT.jar
COPY ${JAR_FILE} app.jar

ENTRYPOINT ["java","-jar","/app.jar"]
```

젠킨스 파이프라인

```
node {

    try {

        stage('GitLab') {
            git branch: 'dev', credentialsId: 'jenkins', url: 'https://lab.ssafy.com/s10-bigdata-recom-sub2/S10P22D109.git'
        }

        stage('Gradle Build') {
            dir('backend/springboot') {
                sh "chmod +x gradlew"
                sh "./gradlew clean build"
            }
        }


        stage('Deploy') {

            sh 'docker-compose down --remove-orphans'
            sh 'docker-compose build --no-cache'
            sh 'docker-compose up -d --force-recreate'


        }
```

```
        // 성공한 경우의 후처리
        def Author_ID = sh(script: 'git show -s --pretty=%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.BUILD_NUMBER} by ${Author_ID}(${Author_Name})₩n(<${env.BUILD_URL}|Details>)",
                       endpoint: 'https://meeting.ssafy.com/hooks/q8d8cwossjbwprucjouu8bafdy',
                       channel: 'D109Build')
    } catch (Exception e) {
        // 실패한 경우의 후처리
        def Author_ID = sh(script: 'git show -s --pretty=%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>)",
                       endpoint: 'https://meeting.ssafy.com/hooks/q8d8cwossjbwprucjouu8bafdy',
                       channel: 'D109Build')
        throw e // 예외를 다시 던져서 빌드 실패 상태를 유지
    }


}
```

DB환경(AWS RDS)

```
spring:
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://savior.c7cmcuk2k97d.ap-northeast-
2.rds.amazonaws.com/savior?useSSL=false&serverTimezone=Asia/Seoul&character
Encoding=UTF-8
    username: root
    password: root1234
```

>실행순서

1. EC2서버에서 도커를 깔고 젠킨스를 설치한다.
2. 젠킨스에서 깃랩과 웹훅으로 연결한다.
3. 젠킨스 파이프라인 코드를 넣는다
4. 프로젝트 최상단에 docker-compose.yml을 넣는다
5. 프론트/백엔드 상단에 dockerfile을 넣는다
6. 카카오 로그인을 위해 카카오 디벨로퍼에서 설정을 해준다
7. DB설정을 위해 AWS RDS에서 DB를 하나 만들어서 스프링부트에 설정해준다
8. DI/CD 끝!