포팅 메뉴얼

Index

---

1. 프로젝트 사용 도구

- 이슈 관리 : Jira
- 형상 관리 : GitLab
- 커뮤니케이션 : Notion, Mattermost
- 테스트 : Postman, Swagger
- UI/UX : Figma

2. 프로젝트 개발 환경

Frontend

- Visual Studio Code : `1.85.1`
- React : `18.2.56`
- React-redux : `9.1.0`
- React-dom : `18.2.0`
- Typescript : `5.2.2`
- Node.js : `20.10.0`
- npm: `10.4.0`
- Vite : `5.1.5`

Backend

- IntelliJ : `2023.03`
- Java : `17`
- SpringBoot : `3.2.3`

- SpringSecurity : `3.2.3`
- JPA : `3.2.3`
- Lombok : `1.18.30`
- Python : `3.10.11`

DB

- MySQL : `8.3.0`
- Redis : `7.2.4`

Service

- NginX : `1.18.0`
- Jenkins : `2.451`
- Docker : `25.0.5`

Server

- Ubuntu : `20.04`

3. 외부서비스

- Kakao Login API
- AWS S3
- AWS RDS

4. 빌드

Frontend

```
npm i
npm run build
```

Backend (Spring Boot Server)

```
Gradle -> build
```

Recommend (Fast Api Server)

```
uvicorn main:app --reload --host=0.0.0.0 --port=${PORT}
```

5. 프로젝트 환경 변수

Backend (Spring Boot Server)

application.yml

```
spring:
  profiles:
    active: prod
```

application-prod.yml

```
spring:
  config:
    activate:
      on-profile: prod
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: ${DATABASE_URL}
    username: ${DATABASE_USERNAME}
    password: ${DATABASE_PASSWORD}
  jpa:
    hibernate:
      ddl-auto: none
```

```
    show-sql: true
    properties:
      hibernate:
        format_sql: true
        show_sql: true
        jdbc:
          time_zone: Asia/Seoul
  data:
    redis:
      host: ${REDIS_HOST}
      port: ${REDIS_PORT}
  # Kakao Login config
  security:
    oauth2:
      client:
        registration:
          kakao:
            client-id: ${KAKAO_CLIENT_ID}
            client-secret: ${KAKAO_CLIENT_SECRET}
            redirect-uri: ${KAKAO_REDIRECT_URL}
            client-authentication-method: client_secret_post
            authorization-grant-type: authorization_code
            scope:
              - account_email
            client-name: Kakao
        provider:
          kakao:
            authorization-uri: https://kauth.kakao.com/oauth/
            token-uri: https://kauth.kakao.com/oauth/token
            user-info-uri: https://kapi.kakao.com/v2/user/me
            user-name-attribute: id
# log levels
logging:
  level:
    org.springframework.data.redis: debug
    org.springframework.security: debug
# JWT config
jwt:
```

```
  secret: ${JWT_SECRET}
  access:
    expiration: ${JWT_EXPIRATION_TIME}
    header: Authorization
  refresh:
    expiration: ${JWT_REFRESH_EXPIRATION_TIME}
    header: Authorization-refresh
#S3 config
cloud:
  aws:
    s3:
      bucket: ${S3_BUCKET_NAME}
      base-url: ${S3_URL}
    credentials:
      access-key: ${S3_ACCESS_KEY}
      secret-key: ${S3_PRIVATE_KEY}
    region:
      static: ap-northeast-2
      auto: false
    stack:
      auto: false

#Port
server:
  port: ${SERVER_PORT}

# WebClient Url
api:
  recommendUrl: ${RECOMMENE_SERVER_URL}:${RECOMMENE_SERVER_PO
```

Frontend

.env

```
VITE_API_BASE_URL = ${API_BASE_URL}
VITE_SOKCET_BROKER_URL = ${WEBSOCKET_BASE_URL}
```

Data + Recommend (Fast Api Server)

.env

```
# database credentials
DATABASE_USER=${DATABASE_USER}
DATABASE_PASSWORD=${DATABASE_PASSWORD}

# database connection
DATABASE_HOST=${DATABASE_HOST}
DATABASE_PORT=${DATABASE_PORT}
DATABASE_NAME=${DATABASE_NAME}
DATABASE_URL="mysql+pymysql://${DATABASE_USER}:${DATABASE_PAS

# musixmatch
MUSIX_TOKEN=${MUSIXMATCH_API_TOKEN}
MUSIX_URL="https://apic-desktop.musixmatch.com/ws/1.1/macro.s
MUSIX_AUTHORITY="apic-desktop.musixmatch.com"
MUSIX_COOKIE="x-mxm-token-guid="

# spotify
#SPOTIFY_CLIENT_ID=${SPOTIFY_CLIENT_ID}
#SPOTIFY_CLIENT_SECRET=${SPOTIFY_CLIENT_SECRET}

# youtube
YOUTUBE_API_KEY=${YOUTUBE_API_KEY}

# papago
PAPAGO_TRANSLATION_URL="https://naveropenapi.apigw.ntruss.com
PAPAGO_LANGUAGE_DETECTION_URL="https://naveropenapi.apigw.ntr
PAPAGO_CLIENT_ID=${PAPAGO_CLIENT_ID}
PAPAGO_CLIENT_SECRET=${PAPAGO_CLIENT_SECRET}

# chromedriver
CHROMEDRIVER_PATH=${PATH_TO_CHROMEDRIVER}

# model path
MODEL_PATH=${PATH_TO_MODEL}

# redis
```

```
REDIS_HOST=${REIS_HOST}
REDIS_PORT=${REDIS_PORT}
REDIS_DB=0
```

.gitignore

```
# config
**/.env
**/application-prod.yml
```

배포

목차

1. Docker/Jenkins 설치

1.1 Docker 설치

```
sudo apt-get -y install apt-transport-https ca-certificates
curl gnupg-agent software-properties-common | curl -fsSL ht
tps://download.docker.com/linux/ubuntu/gpg | sudo apt-key a
dd - | sudo add-apt-repository "deb [arch=amd64] https://do
wnload.docker.com/linux/ubuntu $(lsb_release -cs) stable" |
```

```
sudo apt-get -y install docker-ce docker-ce-cli containerd.
io
```

1.2 Jenkins 설치

```
docker pull jenkins/jenkins:jdk17 | docker run -d --restart
always --env JENKINS_OPTS=--httpPort=<포트번호> -v /etc/local
time:/etc/localtime:ro -e TZ=Asia/Seoul -p <포트번호>:<포트번
호>-v /jenkins:/var/jenkins_home -v /var/run/docker.sock:/v
ar/run/docker.sock -v /usr/local/bin/docker-compose:/usr/lo
cal/bin/docker-compose --name jenkins -u root jenkins/jenki
ns:jdk17
```

1.3 Jenkins 내부 Docker 패키지 설치

```
apt-get update && apt-get -y install apt-transport-https ca
-certificates curl gnupg2 software-properties-common && cur
l -fsSL https://download.docker.com/linux/$(. /etc/os-relea
se; echo "$ID")/gpg > /tmp/dkey; apt-key add /tmp/dkey && a
dd-apt-repository "deb [arch=amd64] https://download.docke
r.com/linux/$(. /etc/os-release; echo "$ID") $(lsb_release
-cs) stable" && apt-get update && apt-get -y install docker
-ce
```

2. NginX 설정

2-1. SSL 설정

```
sudo snap install --classic certbot | sudo certbot --nginx
-d <등록할 도메인 주소>
```

2-2. 리버스 프록시 설정

1) nginx.conf

- 파일 위치 : etc/nginx/nginx.conf

```
user root;
worker_processes auto;
pid /run/nginx.pid;
include /etc/nginx/modules-enabled/*.conf;

events {
        worker_connections 768;
}

http {
        sendfile on;
        tcp_nopush on;
        tcp_nodelay on;
        keepalive_timeout 65;
        types_hash_max_size 2048;
        include /etc/nginx/mime.types;
        default_type application/octet-stream;

        ssl_protocols TLSv1 TLSv1.1 TLSv1.2 TLSv1.3;
        ssl_prefer_server_ciphers on;

        access_log /var/log/nginx/access.log;
        error_log /var/log/nginx/error.log;

        gzip on;
                application/xml application/xml+rss text/ja
vascript;

        include /etc/nginx/sites-enabled/*;
}
```

2) include 된 /etc/nginx/sites-enabled/default

```
server {
    listen [::]:443 ssl ipv6only=on;
    listen 443 ssl;
```

```
    ssl_certificate /etc/letsencrypt/live/j10a106.p.ssafy.i
o/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/j10a106.p.ssa
fy.io/privkey.pem;
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

    # Springboot (/api)
    location ^~ /api {
        rewrite ^/api/(.*)$ /$1 break;
        proxy_pass http://172.19.0.3:8081;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwa
rded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        add_header Content-Security-Policy "default-src 'se
lf'; connect-src 'self' http://localhost:5173 http://localh
ost:5174 http://localhost:5175; script-src 'self' 'unsafe-i
nline'; object-src 'none';";
    }

    # React
    location / {
        proxy_pass http://127.0.0.1:5173;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwa
rded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    # Swagger
    location ~ ^/(swagger|webjars|configuration|swagger-res
ources|v2|v3/api-docs|csrf) {
        proxy_pass http://172.19.0.3:8081;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
```

```
        proxy_set_header X-Forwarded-For $proxy_add_x_forwa
rded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header X-Forwarded-Prefix /api;
    }

    # WebSocket
    location /ws-stomp {
        proxy_pass http://172.19.0.3:8081;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwa
rded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_read_timeout 7200s;
    }
}
```

3. Redis 설치

3.1 Redis 컨테이너 생성

```
sudo docker pull redis | docker run -d --restart always -e
TZ=Asia/Seoul -p 포트번호:포트번호 --name redis redis
```

4. Backend - API 서버(Spring Boot) 배포

4.1 Spring Dockerfile

Dockerfile

```
FROM docker
COPY --from=docker/buildx-bin:latest /buildx /usr/libexec/d
ocker/cli-plugins/docker-buildx

FROM openjdk:17-slim
EXPOSE 443

RUN apt-get update && apt-get install -y redis-tools

ADD ./build/libs/englising-be-0.0.1-SNAPSHOT.jar app.jar
ENTRYPOINT ["java", "-jar", "/app.jar"]
```

4.2 Jenkins 파이프라인 작성

```
pipeline {
    agent any
    environment {
        imageName = <이미지 이름>
        registryCredential = <docker 계정 Credential>
        dockerImage = ''

        releaseServerAccount = <서버 계정>
        releaseServerUri = <서버네임>
        releasePort = <포트번호>
        container_name = <컨테이너 이름>
    }
    stages {
        stage('Git Clone') {
            steps {
                git branch: <브랜치 명>,
                credentialsId: <gitlab 계정 Credential>,
                url: <gitlab url>
            }
        }
        stage('Application-prod.yml Download') {
            steps {
                dir ('englising-be') {
```

```
                    withCredentials([file(credentialsId:  <
properties Credential>, variable: 'appConfigFile')]) {
                        script {
                            sh 'rm -f src/main/resources/ap
plication.yml || true'
                            sh 'cp $appConfigFile src/main/
resources/application.yml'
                        }
                    }
                    withCredentials([file(credentialsId: 'b
e-prod-properties', variable: 'prodConfigFile')]) {
                        script {
                            sh 'cp $prodConfigFile src/mai
n/resources/application-prod.yml'
                        }
                    }
                }
            }
        }
        stage('Jar Build') {
            steps {
                dir ('englising-be') {
                    sh 'chmod +x ./gradlew'
                    sh './gradlew clean bootJar'
                }
            }
        }
        stage('Image Build & DockerHub Push') {
            steps {
                script {
                    docker.withRegistry('', registryCredent
ial) {
                        dir('englising-be') {
                            sh "docker buildx create --use
--name mybuilder"
                            sh "docker buildx build --platf
orm linux/amd64,linux/arm64 -t $imageName:$BUILD_NUMBER --p
ush ."
```

```
                            sh "docker buildx build --platf
orm linux/amd64,linux/arm64 -t $imageName:latest --push ."
                        }
                    }
                }
            }
        }
        stage('DockerHub Pull') {
            steps {
                sshagent(credentials: [<Ubuntu 계정 Credenti
al>]) {
                    sh "ssh -o StrictHostKeyChecking=no $re
leaseServerAccount@$releaseServerUri 'sudo docker pull $ima
geName:latest'"
                }
            }
        }
        stage('Service Start') {
            steps {
                sshagent(credentials: [<Ubuntu 계정 Credenti
al>]) {
                    sh '''
                        ssh -o StrictHostKeyChecking=no $re
leaseServerAccount@$releaseServerUri "\
                        sudo docker stop $container_name ||
true && \
                        sudo docker rm $container_name || t
rue && \
                        sudo docker run -d -e TZ=Asia/Seoul
\
                        --network englising-net \
                        --name $container_name -p $releaseP
ort:$releasePort \
                        $imageName:latest"
                    '''
                }
            }
        }
```

```
    }

    post {
        always {
            echo 'Cleaning up...'
        }
        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>)",
                endpoint: <MatterMost Url>,
                channel: <MatterMost Channel Name>
                )
            }
        }
        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.B
UILD_NUMBER} by ${Author_ID}(${Author_Name})\n(<${env.BUILD
_URL}|Details>)",
                endpoint: <MatterMost Url>,
                channel: <MatterMost Channel Name>
                )
            }
        }
    }
}
```

5. Frontend - React Vite App 배포

5.1 React Dockerfile

Dockerfile

```
FROM node:alpine as build

WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
RUN npm run build
FROM nginx:alpine
RUN rm -rf /etc/nginx/conf.d/default.conf
COPY nginx.conf /etc/nginx/conf.d
COPY --from=build /app/dist /usr/share/nginx/html
EXPOSE 5173
CMD ["nginx", "-g", "daemon off;"]
```

5.2 NginX 설정

nginx.conf

- 파일 위치 : /etc/nginx/conf.d/nginx.conf

```
server {
            listen <포트번호>;
            location / {
                 root   /usr/share/nginx/html;
                 index  index.html index.htm;
                try_files $uri $uri/ /index.html;
                 }

            error_page   500 502 503 504  /50x.html;
            location = /50x.html {
                root   /usr/share/nginx/html;
```

```
                }
}
```

5.3 Jenkins 파이프라인 작성

```
pipeline {
    agent any
    environment {
        container_name = <이미지 이름>
        imageName = <이미지 이름>
        registryCredential = <docker 계정 Credential>
        dockerImage = ''

        releaseServerAccount = <서버 계정>
        releaseServerUri = <서버네임>
        releasePort = <포트번호>
    }

    stages {
        stage('Git Clone') {
            steps {
                git branch: <브랜치 명>,
                credentialsId: <gitlab 계정 Credential>,
                url: <gitlab url>
            }
        }
        stage('.env Insert') {
            steps {
                dir ('englising-fe') {
                    withCredentials([file(credentialsId:  <
fe-env Credential>, variable: 'configFile')]) {
                        script {
                            sh 'rm -f .env || true'
                            sh 'cp $configFile .env'
                        }
                    }
                }
            }
```

```
        }
        stage('Install dependencies') {
            steps {
                dir('englising-fe'){
                     sh 'npm install'
                }
            }
        }
        stage('Build') {
            steps {
                dir('englising-fe'){
                     sh 'npm run build'
                }
            }
        }
        stage('Image Build & DockerHub Push') {
            steps {
                script {
                    docker.withRegistry('', registryCredent
ial) {
                        dir('englising-fe') {
                            sh "docker buildx create --use
--name mybuilder"
                            sh "docker buildx build --platf
orm linux/amd64,linux/arm64 -t $imageName:$BUILD_NUMBER --p
ush ."
                            sh "docker buildx build --platf
orm linux/amd64,linux/arm64 -t $imageName:latest --push ."
                        }
                    }
                }
            }
        }
        stage('DockerHub Pull') {
            steps {
                sshagent(credentials: [<Ubuntu 계정 Credenti
al>]) {
                    sh "ssh -o StrictHostKeyChecking=no $re
```

```
leaseServerAccount@$releaseServerUri 'sudo docker pull $ima
geName:latest'"
                }
            }
        }
        stage('Service Start') {
            steps {
                sshagent(credentials: [<Ubuntu 계정 Credenti
al>]) {
                    sh '''
                        ssh -o StrictHostKeyChecking=no $re
leaseServerAccount@$releaseServerUri "\
                        sudo docker stop $container_name ||
true && \
                        sudo docker rm $container_name || t
rue && \
                        sudo docker run -d -e TZ=Asia/Seoul
\
                        --name $container_name -p $releaseP
ort:$releasePort \
                        $imageName:latest"
                    '''
                }
            }
        }
    }

    post {
        always {
            echo 'Cleaning up...'
        }
        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>)",
                endpoint: <MatterMost Url>,
                channel: <MatterMost Channel Name>
                )
            }
        }
        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.B
UILD_NUMBER} by ${Author_ID}(${Author_Name})\n(<${env.BUILD
_URL}|Details>)",
                endpoint: <MatterMost Url>,
                channel: <MatterMost Channel Name>
                )
            }
        }
    }
}
```

6. Recommend - 추천 서버(Fast Api) 배포

6-1. Python Dockerfile 작성

Dockerfile

```
FROM python:3.10

WORKDIR /app

COPY . /app
```

```
RUN apt-get update \
    && apt-get install -y wget unzip \
    && pip install --no-cache-dir -r requirements.txt


RUN python -m nltk.downloader punkt averaged_perceptron_tag
ger wordnet stopwords
CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port",
"8083"]
```

6-2. Jenkins 파이프라인 설정

```
pipeline {
    agent any
    environment {
        container_name = <이미지 이름>
        imageName = <이미지 이름>
        registryCredential = <docker 계정 Credential>
        dockerImage = ''

        releaseServerAccount = <서버 계정>
        releaseServerUri = <서버네임>
        releasePort = <포트번호>
    }
    stages {
        stage('Git Clone') {
            steps {
                git branch: <브랜치 명>,
                credentialsId: <gitlab 계정 Credential>,
                url: <gitlab url>
            }
        }
        stage('.env Insert') {
            steps {
                dir ('englising-recommend') {
                    withCredentials([file(credentialsId: <d
ata-env Credential>, variable: 'configFile')]) {
```

```
                        script {
                            sh 'rm -f .env || true'
                            sh 'cp $configFile .env'
                        }
                    }
                }
            }
        }
        stage('Image Build & DockerHub Push') {
            steps {
                script {
                    docker.withRegistry('', registryCredent
ial) {
                        dir('englising-recommend') {
                            sh "docker buildx create --use
--name mybuilder"
                            sh "docker buildx build --platf
orm linux/amd64 -t $imageName:$BUILD_NUMBER --push ."
                            sh "docker buildx build --platf
orm linux/amd64 -t $imageName:latest --push ."
                        }
                    }
                }
            }
        }
        stage('DockerHub Pull') {
            steps {
                sshagent(credentials: [<Ubuntu 계정 Credenti
al>]) {
                    sh "ssh -o StrictHostKeyChecking=no $re
leaseServerAccount@$releaseServerUri 'sudo docker pull $ima
geName:latest'"
                }
            }
        }
        stage('Service Start') {
            steps {
                sshagent(credentials: [<Ubuntu 계정 Credenti
```

```
al>]) {
                    sh '''
                        ssh -o StrictHostKeyChecking=no $re
leaseServerAccount@$releaseServerUri "\
                        sudo docker stop $container_name ||
true && \
                        sudo docker rm $container_name || t
rue && \
                        sudo docker run -d -e TZ=Asia/Seoul
\
                        --network englising-net \
                        -v /home/ubuntu/englising/model:/ap
p/word_model \
                        --name $container_name -p $releaseP
ort:$releasePort \
                        $imageName:latest"
                    '''
                }
            }
        }
    }
    post {
        always {
            echo 'Cleaning up...'
        }
    }
}
```

7. Data - 데이터 크롤링 컨테이너 배포

7-1. Python Dockerfile 작성 (Selenium 포함)

Dockerfile

```
FROM python:3.10
WORKDIR /app
```

```
COPY requirements.txt ./

RUN pip install --no-cache-dir -r requirements.txt
RUN apt-get update && \
    apt-get install -y wget gnupg2 unzip && \
    apt-get clean && \
    rm -rf /var/lib/apt/lists/*
RUN wget -q -O - https://dl-ssl.google.com/linux/linux_sign
ing_key.pub | apt-key add - && \
    echo "deb [arch=amd64] http://dl.google.com/linux/chrom
e/deb/ stable main" >> /etc/apt/sources.list.d/google-chrom
e.list && \
    apt-get update && \
    apt-get install -y google-chrome-stable && \
    apt-get clean && \
    rm -rf /var/lib/apt/lists/*

ARG CHROME_DRIVER_VERSION=123.0.6312.86
RUN wget -O /tmp/chromedriver.zip https://storage.googleapi
s.com/chrome-for-testing-public/123.0.6312.86/linux64/chrom
edriver-linux64.zip && \
    unzip /tmp/chromedriver.zip -d /tmp/ && \
    mv /tmp/chromedriver-linux64/chromedriver /usr/local/bi
n/chromedriver && \
    chmod +x /usr/local/bin/chromedriver && \
    mv /usr/local/bin/chromedriver /app/chromedriver

COPY . .


CMD ["python", "./main.py"]
```

7-2. Python Dockerfile 작성 (Selenium 미포함)

Dockerfile

```
FROM python:3.10.11

WORKDIR /app
```

```
COPY . /app

COPY requirements.txt /app/requirements.txt
RUN pip install --no-cache-dir -r requirements.txt

CMD ["python", "main.py"]
```

7-3. Jenkins 파이프라인 설정

```
pipeline {
    agent any
    environment {
        container_name = <이미지 이름>
        imageName = <이미지 이름>
        registryCredential = <docker 계정 Credential>
        dockerImage = ''

        releaseServerAccount = <서버 계정>
        releaseServerUri = <서버네임>
        releasePort = <포트번호>
    }
    stages {
        stage('Git Clone') {
            steps {
                git branch: <브랜치 명>,
                credentialsId: <gitlab 계정 Credential>,
                url: <gitlab url>
            }
        }
        stage('.env Insert') {
            steps {
                dir (<프로젝트 루트 디렉토리>) {
                    withCredentials([file(credentialsId: <d
ata-env Credential>, variable: 'configFile')]) {
                        script {
                            sh 'rm -f .env || true'
                            sh 'cp $configFile .env'
                        }
```

```
                    }
                }
            }
        }
        stage('Image Build & DockerHub Push') {
            steps {
                script {
                    docker.withRegistry('', registryCredent
ial) {
                        dir('englising-data-youtube-crawle
r') {
                            sh "docker buildx create --use
--name mybuilder"
                            sh "docker buildx build --platf
orm linux/amd64 -t $imageName:$BUILD_NUMBER --push ."
                            sh "docker buildx build --platf
orm linux/amd64 -t $imageName:latest --push ."
                        }
                    }
                }
            }
        }
        stage('DockerHub Pull') {
            steps {
                sshagent(credentials: [<Ubuntu 계정 Credenti
al>]) {
                    sh "ssh -o StrictHostKeyChecking=no $re
leaseServerAccount@$releaseServerUri 'sudo docker pull $ima
geName:latest'"
                }
            }
        }
        stage('Service Start') {
            steps {
                sshagent(credentials: [<Ubuntu 계정 Credenti
al>]) {
                    sh '''
                        ssh -o StrictHostKeyChecking=no $re
```

```
leaseServerAccount@$releaseServerUri "\
                        sudo docker stop $container_name ||
true && \
                        sudo docker rm $container_name || t
rue && \
                        sudo docker run -d -e TZ=Asia/Seoul
\
                        --network englising-net \
                        --name $container_name -p $releaseP
ort:$releasePort \
                        $imageName:latest"
                    '''
                }
            }
        }
    }

    post {
        always {
            echo 'Cleaning up...'
        }
    }
}
```

외부 서비스 이용

카카오 로그인 API

1) 카카오 디벨로퍼스 → 내 애플리케이션 → 애플리케이션 추가

2) 생성한 애플리케이션 → 플랫폼 → 사이트 도메인 추가 후 앱 키 사용