**1개발 환경**

Server : Ubuntu 20.04.6 LTS

JDK : OpenJDK17

Nginx : nginx/1.18.0

MySQL : 9.0.1

IntelliJ : 2024.2.0.2

Yolo : v8

React : 18

설정 및 환경 변수

CLIENT\_ID=901a36598e2f99dd264cdcd3700f0483; CLIENT\_SECRET=kTr5DvLPUmKknhMmVlg3t4QDSw3CTPKV; DB\_PASSWORD=ssafy; DB\_USER=root; MYSQL\_PASSWORD=rkdrlaqkr; MYSQL\_SERVER=3.36.63.119:3306; MYSQL\_USERNAME=joohan; SERVER\_URL=<http://localhost:8080/api/v1>; API\_KEY=48d83eda8cc14139a81c7c8bad489f23

**2. 빌드시 사용되는 환경변수**

**Ngnix 설치**

sudo apt update

sudo apt install nginx –y

sudo systemctl start nginx

sudo systemctl enable nginx

ngnix 파일

**server {**

**listen 80;**

**server\_name j11a503.p.ssafy.io;**

**client\_max\_body\_size 16777216;**

**location / {**

**return 301 https://$host$request\_uri; # HTTP를 HTTPS로 리다이렉트**

**}**

**}**

**server {**

**listen 443 ssl;**

**server\_name j11a503.p.ssafy.io;**

**ssl\_certificate /etc/letsencrypt/live/j11a503.p.ssafy.io/fullchain.pem; # SSL 인증서 경로**

**ssl\_certificate\_key /etc/letsencrypt/live/j11a503.p.ssafy.io/privkey.pem; # SSL 키 경로**

**#server\_name j11a503.p.ssafy.io;**

**location /api/v1/ {**

**client\_max\_body\_size 16777216;**

**proxy\_pass http://localhost:8080/api/v1/; # Docker 컨테이너의 포트**

**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-Proto $scheme;**

**}**

**location /ai/v1/ {**

**client\_max\_body\_size 16777216;**

**proxy\_pass http://localhost:8000/ai/v1/; # Docker 컨테이너의 포트**

**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-Proto $scheme;**

**}**

**# 정적 파일 및 프론트엔드 요청 처리**

**location / {**

**proxy\_pass http://localhost:5173;**

**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-Proto $scheme;**

**# 프록시 서버에서 404 응답시 /index.html로 내부 리다이렉션**

**proxy\_intercept\_errors on;**

**error\_page 404 = /index.html;**

**}**

**# JavaScript 파일 MIME 타입 설정**

**location ~\* \.js$ {**

**proxy\_pass http://localhost:5173;**

**add\_header Content-Type application/javascript;**

**}**

**# 기타 정적 파일 처리**

**location ~\* \.(css|png|jpg|jpeg|gif|ico|svg)$ {**

**proxy\_pass http://localhost:5173;**

**expires 1y;**

**add\_header Cache-Control "public, max-age=31536000, immutable";**

**}**

**# listen 443 ssl; # managed by Certbot**

**# ssl\_certificate /etc/letsencrypt/live/j11a503.p.ssafy.io/fullchain.pem; # managed by Certbot**

**# ssl\_certificate\_key /etc/letsencrypt/live/j11a503.p.ssafy.io/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**

**}**

**젠킨스 접속**

**sudo docker exec -it jenkins bash**

**apt-get update**

**node.js 설정**

**curl -sL https://deb.nodesource.com/setup\_14.x | bash -**

**apt-get install -y nodejs**

**npm 설치**

**curl -L https://www.npmjs.com/install.sh | sh**

**젠킨스에 python 3.10 설치 코드**

apt-get update

apt-get install -y software-properties-common

**add-apt-repository ppa:deadsnakes/ppa**

**apt-get update**

**apt-get install -y python3.10**

**AI  
pipeline {**

**agent any**

**environment {**

**SERVER\_URL = 'http://j11a503.p.ssafy.io'**

**GITLAB\_URL = 'https://lab.ssafy.com/s11-fintech-finance-sub1/S11P21A503.git'**

**CRE\_SONG\_GITLAB = 'songGitlab'**

**REMOTE\_HOST = 'ip-172-26-15-220'**

**REMOTE\_USER = 'ubuntu'**

**FLASK\_APP\_PATH = '/home/ubuntu/jenkins-data/workspace/AI/AI/flask'**

**DEPLOY\_DIR = '/home/ubuntu/AI'**

**DOCKER\_IMAGE\_NAME = 'ai\_flask'**

**}**

**stages {**

**stage('Git Clone') {**

**steps {**

**script {**

**git branch: 'AI/develop',**

**url: "${GITLAB\_URL}",**

**credentialsId: "${CRE\_SONG\_GITLAB}"**

**}**

**}**

**}**

**stage('Setup Python Environment') {**

**steps {**

**script {**

**dir('AI/flask') {**

**sh '''**

**python3 -m venv venv**

**. venv/bin/activate**

**which python**

**python --version**

**pip install --upgrade pip**

**pip install Flask-CORS**

**pip install wheel setuptools**

**pip install -r requirements.txt**

**nohup python main.py &**

**'''**

**}**

**}**

**}**

**}**

**stage('Copy Dist Files') {**

**steps {**

**sshagent(credentials: ['ssh-key']) {**

**script {**

**sh """**

**ssh -o StrictHostKeyChecking=no ${REMOTE\_USER}@${REMOTE\_HOST} '**

**mkdir -p ${DEPLOY\_DIR} && # 디렉토리 생성**

**sudo chown -R ${REMOTE\_USER}:${REMOTE\_USER} ${DEPLOY\_DIR} # 소유권 변경**

**'**

**"""**

**dir('AI/flask') {**

**sh """**

**scp -o StrictHostKeyChecking=no -r Dockerfile darknet.py docker-compose.yml main.py model requirements.txt utils.py temp ${REMOTE\_USER}@${REMOTE\_HOST}:${DEPLOY\_DIR}/**

**"""**

**}**

**}**

**}**

**}**

**}**

**stage('Build Docker Image') {**

**steps {**

**sshagent(credentials: ['ssh-key']) {**

**sh """**

**ssh -o StrictHostKeyChecking=no ${REMOTE\_USER}@${REMOTE\_HOST} '**

**cd ${DEPLOY\_DIR} &&**

**docker build -t ${DOCKER\_IMAGE\_NAME} .**

**'**

**"""**

**}**

**}**

**}**

**stage('Deploy with Docker Compose') {**

**steps {**

**sshagent(credentials: ['ssh-key']) {**

**sh """**

**ssh -o StrictHostKeyChecking=no ${REMOTE\_USER}@${REMOTE\_HOST} '**

**cd ${DEPLOY\_DIR} &&**

**docker-compose down &&**

**docker-compose up -d**

**'**

**"""**

**}**

**}**

**}**

**}**

**}  
docker   
# 베이스 이미지 선택**

**FROM python:3.10.12-slim**

**# 작업 디렉토리 설정**

**WORKDIR /app**

**# 시스템 패키지 설치**

**RUN apt-get update && apt-get install -y \**

**libgl1-mesa-glx \**

**libglib2.0-0 \**

**&& rm -rf /var/lib/apt/lists/\***

**# requirements.txt 복사 및 의존성 설치**

**COPY requirements.txt .**

**RUN pip install --upgrade pip**

**RUN pip install -r requirements.txt**

**# 애플리케이션 코드 복사**

**COPY . .**

**# Flask 애플리케이션 실행**

**CMD ["python", "main.py"]  
  
docker-compose  
services:**

**flask-app:**

**image: ai\_flask**

**build:**

**context: .**

**dockerfile: Dockerfile**

**ports:**

**- "8000:8000"**

**volumes:**

**- .:/app**

**environment:**

**- FLASK\_ENV=development  
  
  
Front**

**Prpeline  
pipeline {**

**agent any**

**environment {**

**SERVER\_URL = 'http://j11a503.p.ssafy.io'**

**GITLAB\_URL = 'https://lab.ssafy.com/s11-fintech-finance-sub1/S11P21A503.git'**

**CRE\_SONG\_GITLAB = 'songGitlab'**

**REMOTE\_HOST = 'ip-172-26-15-220'**

**REMOTE\_USER = 'ubuntu'**

**DOCKERFILE\_PATH = '/home/ubuntu/jenkins-data/workspace/Frontend/FE/god-life-routine'**

**DEPLOY\_DIR = '/home/ubuntu/frontend'**

**DOCKER\_IMAGE\_NAME = 'god-life-routine-fe'**

**}**

**stages {**

**stage('Git Clone') {**

**steps {**

**script {**

**git branch: 'FE/develop',**

**url: "${GITLAB\_URL}",**

**credentialsId: "${CRE\_SONG\_GITLAB}"**

**}**

**}**

**}**

**stage('Build') {**

**steps {**

**script {**

**dir('FE/god-life-routine') {**

**sh 'npm install' // 의존성 설치**

**sh 'npm run build' // React 앱 빌드**

**}**

**}**

**}**

**}**

**stage('Copy Dist Files') {**

**steps {**

**sshagent(credentials: ['ssh-key']) {**

**script {**

**sh """**

**ssh -o StrictHostKeyChecking=no ${REMOTE\_USER}@${REMOTE\_HOST} '**

**mkdir -p ${DEPLOY\_DIR} && # 디렉토리 생성**

**sudo chown -R ${REMOTE\_USER}:${REMOTE\_USER} ${DEPLOY\_DIR} # 소유권 변경**

**'**

**"""**

**dir('FE/god-life-routine/dist') {**

**sh 'scp -o StrictHostKeyChecking=no -r ./ ${REMOTE\_USER}@${REMOTE\_HOST}:${DEPLOY\_DIR}/'**

**}**

**dir('FE/god-life-routine') {**

**sh 'scp -o StrictHostKeyChecking=no -r Dockerfile ${REMOTE\_USER}@${REMOTE\_HOST}:${DEPLOY\_DIR}/'**

**sh 'scp -o StrictHostKeyChecking=no -r docker-compose.yml ${REMOTE\_USER}@${REMOTE\_HOST}:${DEPLOY\_DIR}/'**

**sh 'scp -o StrictHostKeyChecking=no -r nginx.conf ${REMOTE\_USER}@${REMOTE\_HOST}:${DEPLOY\_DIR}/'**

**}**

**}**

**}**

**}**

**}**

**stage('Build Docker Image') {**

**steps {**

**sshagent(credentials: ['ssh-key']) {**

**sh """**

**ssh -o StrictHostKeyChecking=no ${REMOTE\_USER}@${REMOTE\_HOST} '**

**cd ${DOCKERFILE\_PATH} &&**

**docker build -t ${DOCKER\_IMAGE\_NAME} .**

**'**

**"""**

**}**

**}**

**}**

**stage('Deploy with Docker Compose') {**

**steps {**

**sshagent(credentials: ['ssh-key']) {**

**sh """**

**ssh -o StrictHostKeyChecking=no ${REMOTE\_USER}@${REMOTE\_HOST} '**

**cd ${DEPLOY\_DIR} &&**

**docker-compose down &&**

**docker-compose up -d**

**'**

**"""**

**}**

**}**

**}**

**}**

**}  
  
docker-compos  
services:**

**god-life-routine-fe:**

**image: god-life-routine-fe**

**container\_name: god-life-routine-fe**

**build:**

**context: .**

**dockerfile: Dockerfile**

**ports:**

**- "5173:80" # Nginx의 80 포트를 외부에 노출**

**environment:**

**- NODE\_ENV=production**

**restart: unless-stopped**

**networks:**

**- app-network**

**networks:**

**app-network:**

**driver: bridge  
  
# Use Nginx to serve the React app**

**FROM nginx:alpine**

**# Copy the build files from the previous stage**

**COPY ./dist /usr/share/nginx/html**

**# Copy custom Nginx config**

**COPY nginx.conf /etc/nginx/conf.d/default.conf**

**# Expose port 80**

**EXPOSE 80**

**# Start Nginx**

**CMD ["nginx", "-g", "daemon off;"]  
  
ngnix  
server {**

**listen 80;**

**server\_name j11a503.p.ssafy.io;**

**root /usr/share/nginx/html;**

**index index.html;**

**location / {**

**try\_files $uri $uri/ /index.html;**

**}**

**# JavaScript 모듈 파일 MIME 타입 처리**

**location ~\* \.js$ {**

**add\_header Content-Type application/javascript;**

**try\_files $uri =404;**

**}**

**# 다른 정적 파일 처리**

**location ~\* \.(css|png|jpg|jpeg|gif|ico|svg)$ {**

**expires 1y;**

**add\_header Cache-Control "public, max-age=31536000, immutable";**

**}**

**}  
  
Back**

**파이프라인  
pipeline {**

**agent any**

**environment {**

**SERVER\_URL = 'http://j11a503.p.ssafy.io'**

**GITLAB\_URL = 'https://lab.ssafy.com/s11-fintech-finance-sub1/S11P21A503.git'**

**CRE\_SONG\_GITLAB = 'songGitlab'**

**REMOTE\_HOST = 'ip-172-26-15-220'**

**REMOTE\_USER = 'ubuntu'**

**JAR\_PATH = '/home/ubuntu/jenkins-data/workspace/GodLife/BE/god-life/build/libs/god-life-0.0.1-SNAPSHOT.jar'**

**DOCKERFILE\_PATH = '/home/ubuntu/jenkins-data/workspace/GodLife/BE/god-life'**

**DEPLOY\_DIR = '/home/ubuntu/app'**

**DOCKER\_COMPOSE\_FILE = '/home/ubuntu/jenkins-data/workspace/GodLife/BE/god-life/docker-compose.yml'**

**DOCKER\_IMAGE\_NAME = 'god-life-app'**

**}**

**stages {**

**stage('Git Clone') {**

**steps {**

**script {**

**git branch: 'BE/develop',**

**url: "${GITLAB\_URL}",**

**credentialsId: "${CRE\_SONG\_GITLAB}"**

**}**

**}**

**}**

**stage('Prepare Environment') {**

**steps {**

**script {**

**dir('BE/god-life') {**

**// .env 파일의 내용을 환경 변수로 설정**

**def envContent = readFile '.env'**

**envContent.split('\n').each { line ->**

**def (key, value) = line.split('=')**

**if (key && value) {**

**env."${key.trim()}" = value.trim()**

**echo "${key.trim()} is set to ${value.trim()}" // 디버깅을 위한 로그**

**}**

**}**

**}**

**}**

**}**

**}**

**stage('Build') {**

**steps {**

**script {**

**dir('BE/god-life') {**

**sh 'chmod +x ./gradlew'**

**sh './gradlew clean build'**

**}**

**}**

**}**

**}**

**stage('Copy jar Files') {**

**steps {**

**sshagent(credentials: ['ssh-key']) {**

**script {**

**sh """**

**ssh -o StrictHostKeyChecking=no ${REMOTE\_USER}@${REMOTE\_HOST} '**

**mkdir -p ${DEPLOY\_DIR} &&**

**sudo chown -R ${REMOTE\_USER}:${REMOTE\_USER} ${DEPLOY\_DIR}**

**'**

**"""**

**dir('BE/god-life/build/libs') {**

**sh 'scp -o StrictHostKeyChecking=no -r ./god-life-0.0.1-SNAPSHOT.jar ${REMOTE\_USER}@${REMOTE\_HOST}:${DEPLOY\_DIR}/'**

**}**

**// .env 파일도 복사**

**dir('BE/god-life') {**

**sh 'scp -o StrictHostKeyChecking=no .env ${REMOTE\_USER}@${REMOTE\_HOST}:${DEPLOY\_DIR}/'**

**}**

**}**

**}**

**}**

**}**

**stage('Build Docker Image') {**

**steps {**

**sshagent(credentials: ['ssh-key']) {**

**sh """**

**ssh -o StrictHostKeyChecking=no ${REMOTE\_USER}@${REMOTE\_HOST} '**

**cd ${DOCKERFILE\_PATH} &&**

**docker build -t ${DOCKER\_IMAGE\_NAME} .**

**'**

**"""**

**}**

**}**

**}**

**stage('Deploy with Docker Compose') {**

**steps {**

**sshagent(credentials: ['ssh-key']) {**

**sh """**

**ssh -o StrictHostKeyChecking=no ${REMOTE\_USER}@${REMOTE\_HOST} '**

**cd ${DEPLOY\_DIR} &&**

**docker-compose down &&**

**docker-compose up -d --build**

**'**

**"""**

**}**

**}**

**}**

**}**

**}  
  
dockerfile  
FROM openjdk:17-jdk-slim**

**WORKDIR /app**

**COPY build/libs/god-life-0.0.1-SNAPSHOT.jar app.jar**

**ENTRYPOINT ["java", "-Duser.timezone=Asia/Seoul", "-jar", "app.jar"]  
  
docker-compose  
services:**

**app:**

**image: god-life-app # 빌드된 Docker 이미지 이름**

**container\_name: god-life-container # 생성될 컨테이너 이름**

**restart: always**

**ports:**

**- "8080:8080" # 호스트:컨테이너 포트 매핑**

**environment:**

**- SPRING\_PROFILES\_ACTIVE=prod # 스프링 프로필 설정 (예시)**

**TZ: "Asia/Seoul"**

**volumes:**

**- /home/ubuntu/app/config:/app/config # 설정 파일 등을 볼륨으로 마운트**

**env\_file:**

**- .env  
  
3. 특이사항  
chrome에서 작동을 해야함**

**4.**