**포팅 메뉴얼**

**구미 2반 D212**

장민성

박단비

이준수

조현우

편수지

황윤정

**Gerrit 설정**- 매뉴얼 참고

**방화벽 설정** – 매뉴얼 참고

1. **버전**

Java 17.0.10 azul

Spring boot 3

Gradle 8.5

NodeJs 20.11.0

Jenkins lts, Openvidu lts, mariaDB lts

1. **Docker-compose**

version: '3'

services:

jenkins:

image: jenkins/jenkins:lts

restart: on-failure

ports:

- "설정할 외부 포트:8080" # 젠킨스 접속 포트

- " 설정할 외부 포트:설정할 내부 포트" #api 서버

- " 설정할 외부 포트:설정할 내부 포트" #채팅 서버

volumes:

- /var/jenkins\_home:/var/jenkins\_home

- /var/run/docker.sock:/var/run/docker.sock # Docker 소켓 공유 추가

- /usr/bin/docker:/usr/bin/docker # 호스트의 Docker 바이너리를 Jenkins 컨테이너 내부로 복사

- /var/www/backend:/var/www/backend

- /var/www/frontend:/var/www/frontend

user: root

environment:

- JASYPT\_ENCRYPTOR\_PASSWORD=암호화 비밀번호

networks:

- app-network

my\_mariadb:

image: my-mariadb-image

ports:

- "설정할 포트:3306"

volumes:

- /var/lib/mysql2:/var/lib/mysql2

networks:

- app-network

networks:

app-network:

driver: bridge

1. **Jenkins pipeline**

pipeline {

agent any

environment {

APP\_NAME = "koala-0.0.1-SNAPSHOT.jar"

GRADLE\_BUILD\_PATH = "/var/www/backend"

DEPLOY\_PATH = "/opt/koala"

LOG\_PATH = "${DEPLOY\_PATH}/logs"

}

tools { nodejs "nodeJS" }

stages {

stage('env') {

steps {

script {

환경변수 설정하기

}

}

}

stage('clone') {

steps {

echo 'Start cloning fitter...'

git credentialsId: 'b6aa708b-82fb-4ab3-94b3-ff7f0b324258', url: 깃랩 주소

echo 'Clone finished!'

}

}

stage('build backend') {

steps{

echo 'Start building fitter...'

sh 'ls'

sh 'cd backend/ && chmod +x ./gradlew && ./gradlew clean build'

sh 'cp backend/build/libs/koala-0.0.1-SNAPSHOT.jar /var/www/backend'

echo 'Build finished!'

}

}

stage('build backendChat') {

steps{

echo 'Start building fitter...'

sh 'ls'

sh 'cd backendChat/ && chmod +x ./gradlew && ./gradlew clean build'

sh 'cp backendChat/build/libs/backendChat-Chat-0.0.1-SNAPSHOT.jar /var/www/backendChat'

echo 'Build finished!'

}

}

stage('publish on ssh'){

steps{

sh 'cd /var/www/backend'

sh 'ls'

sshPublisher(

publishers:

[

sshPublisherDesc(

configName: '프로젝트명’,

transfers:

[

sshTransfer(

cleanRemote: false,

excludes: '',

execCommand: 'sh /home/ubuntu/[프로젝트명]/script/init\_server.sh',

execTimeout: 0,

flatten: false,

makeEmptyDirs: true,

noDefaultExcludes: false,

patternSeparator: '[, ]+',

remoteDirectory: '/[프로젝트명]/deploy',

remoteDirectorySDF: false,

removePrefix: '/var/www/backend',

sourceFiles: '/var/www/backend/koala-0.0.1-SNAPSHOT.jar'

)

],

usePromotionTimestamp: false,

useWorkspaceInPromotion: false,

verbose: true

)

]

)

}

}

// stage('Check Server Status') {

// steps {

// script {

// // 서버 상태 확인

// def isRunning = sh(script: "ps -ef | grep '[k]oala-0.0.1-SNAPSHOT.jar' | wc -l", returnStdout: true).trim()

// if (isRunning.toInteger() > 0) {

// echo '서버가 실행 중입니다. 종료를 시도합니다.'

// // 기존 프로세스 종료

// sh 'ps -ef | grep "[k]oala-0.0.1-SNAPSHOT.jar" | awk \'{print $2}\' | xargs -r kill -SIGTERM'

// // 종료 확인

// def isTerminated = sh(script: "ps -ef | grep '[k]oala-0.0.1-SNAPSHOT.jar' | wc -l", returnStdout: true).trim()

// if (isTerminated.toInteger() == 0) {

// echo '서버 종료가 성공적으로 완료되었습니다.'

// } else {

// error '서버 종료에 실패했습니다.'

// }

// } else {

// echo '실행 중인 서버가 없습니다. 배포를 진행합니다.'

// }

// }

// }

// }

// stage('deploy backend'){

// steps{

// echo 'Start deploying...'

// script {

// // 로그 디렉토리 생성 (존재하지 않는 경우)

// sh "mkdir -p ${LOG\_PATH}"

// // 애플리케이션 배포

// echo "애플리케이션을 ${DEPLOY\_PATH}에 배포합니다."

// sh "cp -f ${GRADLE\_BUILD\_PATH}/${APP\_NAME} ${DEPLOY\_PATH}/"

// // 애플리케이션 실행

// echo "애플리케이션을 백그라운드에서 실행합니다."

// sh "nohup java -jar ${DEPLOY\_PATH}/${APP\_NAME} &"

// }

// // dir ('/var/www/backend') {

// // sh 'nohup java -jar koala-0.0.1-SNAPSHOT.jar &'

// // }

// //sh 'docker stop backend || true'

// //sh 'docker rm backend || true'

// //sh 'docker rmi backend || true'

// //sh 'docker run --name backend -d -v /var/www/backend/:/app -v /var/log/myapp:/var/log/myapp --network host openjdk:17 java -jar /app/koala-0.0.1-SNAPSHOT.jar'

// echo 'Deploy finished!'

// }

// }

stage('build frontend') {

steps{

echo 'Start building fitter...'

sh 'ls'

dir ('./frontend') {

sh 'npm install'

sh "BUILD\_PATH='/var/www/frontend' CI=false npm run build"

}

echo 'Build finished!'

}

}

}

}

1. **Nginx, conf**

server {

listen 80;

rewrite\_log on;

location / {

root /var/www/frontend;

index index.html index.htm;

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

}

location /api {

proxy\_pass 도메인 : 포트; # 백엔드 서버 주소

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;

# /api를 제거하고 백엔드로 전달

rewrite ^/api(.\*?)/?$ $1 break;

}

location ~\* ^/(cocktalk|rtc) {

proxy\_pass 도메인 : 포트 # 백엔드 서버 주소

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;

}

}

server {

listen 443 ssl http2;

server\_name 도메인 주소;

# ssl 인증서 적용하기

ssl\_certificate /etc/letsencrypt/live/[도메인주소]/fullchain.pem;

ssl\_certificate\_key /etc/letsencrypt/live/[도메인주소]/privkey.pem;

location / {

root /var/www/frontend;

index index.html index.htm;

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

}

location /api { # location 이후 특정 url을 처리하는 방법을 정의

proxy\_pass 도메인 주소 : 포트# Request에 대해 어디로 리다이렉트하는지

proxy\_http\_version 1.1;

proxy\_set\_header Connection "upgrade";

proxy\_set\_header Upgrade $http\_upgrade;

proxy\_set\_header Host $http\_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;

proxy\_set\_header X-NginX-Proxy true;

# /api를 제거하고 백엔드로 전달

rewrite ^/api(.\*?)/?$ $1 break;

}

location ~\* ^/(cocktalk|rtc) {

proxy\_pass 도메인 주소 : 포트# 백엔드 서버 주소

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\_forwarded\_for;

proxy\_set\_header X-Forwarded-Proto $scheme;

}

}

1. **배포 백엔드 실행 (수동)**
2. API 서버 실행

* Docker exec –it Jenkins-jenkins-1 bash
* cd /var/www/backend
* java –jar koala-0.0.1-SNAPSHOT.jar

1. 채팅 서버 실행

* Docker exec –it Jenkins-jenkins-1 bash
* Cd /var/www/backendChat
* Java –jar backendChat-Chat-0.0.1-SNAPSHOT.jar

1. 이후 도메인 주소로 웹으로 접속