포팅메뉴얼

(동화숲)

작성자  
K12B202T

편민준

목 차

1. 개발 환경

2. 환경 변수

3. 필수 소프트웨어

4. 배포 및 설치

1. 개발 환경

|  |  |
| --- | --- |
| 사용 툴 | 버전 및 사양 |
| UBUNTU | Ubuntu 24.04.2 LTS (Noble Numbat) |
|  |  |
| AWS EC2 | Vcpu 4, RAM 16GB |
|  |  |
|  |  |
| Docker | 28.0 |
| Containerd | 1.7.27 |
| Python | 3.11 |
|  |  |
| Java | 17 |
| Spring boot | 3.4.4 |
| Spring Data Redis | - |
| React | 19.0.0 |
| Redis | 8.0.1 |
| Mysql | 8.4.4 LTS |
| AWS Java SDK | V2 |
| NextJS | 15.3.2 |
| TailwindCSS | 4.1.4 |
| TypeScript | 5.8.3 |
| Zustand | 5.0.3 |
| Biome | 1.9.4 |
| Cursor | 0.50 |
| OpenAI API | 4.5 |

2. 환경 변수

**- Backend**

JWT\_ACCESS\_EXP

JWT\_HEADER

JWT\_PREFIX

JWT\_REFRESH\_EXP

JWT\_SECRET\_KEY

KAKAO\_ADMIN\_KEY

KAKAO\_APP\_ID

KAKAO\_BASE\_URL

KAKAO\_CLIENT\_ID

KAKAO\_REDIRECT\_URL

MYSQL\_DATABASE

MYSQL\_HOST

MYSQL\_PASSWORD

MYSQL\_USER

MYSQL\_PORT

REDIS\_HOST\

REDIS\_PASSWORD

REDIS\_PORT

KAFKA\_BOOTSTRAP\_SERVERS

KAFKA\_TOPIC

KAFKA\_GROUP\_ID

KAFKA\_SPRING\_GROUP

KAFKA\_RESULT\_TOPIC

DATABASE\_URL

OPENAI\_API\_KEY

AWS\_ACCESS\_KEY\_ID

AWS\_SECRET\_ACCESS\_KEY

AWS\_REGION

S3\_BUCKET\_NAME

COQUI\_TOS\_AGREED

KAKAO\_REDIRECT

KAKAO\_REDIRECT\_URL\_CALLBACK

**- Frontend**

VITE\_KAKAO\_REST\_API\_KEY

VITE\_API\_BASE\_URL

VITE\_KAKAO\_CLIENT\_ID

VITE\_KAKAO\_REDIRECT\_URI

VITE\_KAKAO\_POPUP\_REDIRECT\_URI

-docker-compose.yml

version: '3'

services:

jenkins:

build:

context: .

dockerfile: Dockerfile.jenkins

container\_name: jenkins

restart: unless-stopped

privileged: true

user: "0:0"

ports:

- "5000:8080"

- "50000:50000"

volumes:

- jenkins\_home:/var/jenkins\_home

- /var/run/docker.sock:/var/run/docker.sock

- /usr/bin/docker:/usr/bin/docker

- /home/ubuntu/nginx:/home/ubuntu/nginx

environment:

- JAVA\_OPTS=-Djenkins.install.runSetupWizard=false -Xmx2g

networks:

- jenkins-network

mysql:

image: mysql:8.0

container\_name: mysql

restart: unless-stopped

environment:

MYSQL\_ROOT\_PASSWORD: ssafy202

MYSQL\_DATABASE: donghwasoop

MYSQL\_USER: user

MYSQL\_PASSWORD: ssafy202

ports:

- "3306:3306"

volumes:

- mysql\_data:/var/lib/mysql

networks:

- jenkins-network

redis:

image: redis:7

container\_name: redis

restart: unless-stopped

ports:

- "6379:6379"

networks:

- jenkins-network

zookeeper:

image: confluentinc/cp-zookeeper:7.6.0

container\_name: zookeeper

restart: unless-stopped

environment:

ZOOKEEPER\_CLIENT\_PORT: 2181

ZOOKEEPER\_TICK\_TIME: 2000

ports:

- "2181:2181"

networks:

- jenkins-network

kafka:

image: confluentinc/cp-kafka:7.6.0

container\_name: kafka

restart: unless-stopped

depends\_on:

- zookeeper

ports:

- "9092:9092"

environment:

KAFKA\_BROKER\_ID: 1

KAFKA\_ZOOKEEPER\_CONNECT: zookeeper:2181

KAFKA\_ADVERTISED\_LISTENERS: PLAINTEXT://k12b202.p.ssafy.io:9092

KAFKA\_OFFSETS\_TOPIC\_REPLICATION\_FACTOR: 1

KAFKA\_AUTO\_CREATE\_TOPICS\_ENABLE: "true"

networks:

- jenkins-network

portainer:

image: portainer/portainer-ce

container\_name: portainer

restart: unless-stopped

ports:

- "9000:9000"

volumes:

- /var/run/docker.sock:/var/run/docker.sock

- portainer\_data:/data

networks:

- jenkins-network

elasticsearch:

image: docker.elastic.co/elasticsearch/elasticsearch:7.17.12

container\_name: elasticsearch

environment:

- discovery.type=single-node

- xpack.security.enabled=false

- "ES\_JAVA\_OPTS=-Xms1g -Xmx1g" # 힙 크기를 1GB로 제한

ports:

- "9200:9200"

volumes:

- esdata:/usr/share/elasticsearch/data

networks:

- elk

kibana:

image: docker.elastic.co/kibana/kibana:7.17.12

container\_name: kibana

ports:

- "5601:5601"

depends\_on:

- elasticsearch

environment:

- ELASTICSEARCH\_HOSTS=http://elasticsearch:9200

networks:

- elk

logstash:

image: docker.elastic.co/logstash/logstash:7.17.12

container\_name: logstash

volumes:

- ~/logstash/pipeline:/usr/share/logstash/pipeline

- ./logs:/logs

ports:

- "5044:5044"

- "5046:5046"

depends\_on:

- elasticsearch

networks:

- elk

- jenkins-network

networks:

jenkins-network:

driver: bridge

elk:

name: fariytale\_elk # 명시적으로 전체 이름 지정

driver: bridge

volumes:

jenkins\_home:

external: true

name: fariytale\_jenkins\_home

mysql\_data:

portainer\_data:

esdata:

**- Nginx**

**# ✅ HTTP 요청은 HTTPS로 리디렉션**

**server {**

**listen 80;**

**listen [::]:80;**

**server\_name donghwasoop.com www.donghwasoop.com;**

**return 301 https://$host$request\_uri;**

**}**

**server {**

**listen 80;**

**server\_name k12b202.p.ssafy.io;**

**return 301 https://$host$request\_uri;**

**}**

**# ✅ HTTPS 서버**

**server {**

**listen 443 ssl;**

**listen [::]:443 ssl;**

**server\_name donghwasoop.com www.donghwasoop.com;**

**# 🔐 SSL 인증서 경로 (Certbot이 자동으로 설정함)**

**ssl\_certificate /etc/letsencrypt/live/donghwasoop.com/fullchain.pem;**

**ssl\_certificate\_key /etc/letsencrypt/live/donghwasoop.com/privkey.pem;**

**include /etc/letsencrypt/options-ssl-nginx.conf;**

**ssl\_dhparam /etc/letsencrypt/ssl-dhparams.pem;**

**# 🔧 기본 설정**

**root /home/ubuntu/nginx;**

**index index.html;**

**client\_max\_body\_size 50M;**

**# 🔄 SPA 지원**

**location / {**

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

**}**

**# 🔁 API 프록시**

**location /api/ {**

**proxy\_pass http://localhost:8080;**

**proxy\_set\_header Host $host;**

**proxy\_set\_header X-Real-IP $remote\_addr;**

**}**

**# 🔁 WebSocket 프록시**

**location /api/v1/ws/ {**

**proxy\_pass http://localhost:8080;**

**proxy\_http\_version 1.1;**

**proxy\_set\_header Upgrade $http\_upgrade;**

**proxy\_set\_header Connection "Upgrade";**

**proxy\_set\_header Host $host;**

**proxy\_cache\_bypass $http\_upgrade;**

**proxy\_read\_timeout 3600s;**

**}**

**}**

**server {**

**listen 443 ssl;**

**server\_name k12b202.p.ssafy.io;**

**ssl\_certificate /etc/letsencrypt/live/p.ssafy.io/fullchain.pem;**

**ssl\_certificate\_key /etc/letsencrypt/live/p.ssafy.io/privkey.pem;**

**include /etc/letsencrypt/options-ssl-nginx.conf;**

**ssl\_dhparam /etc/letsencrypt/ssl-dhparams.pem;**

**client\_max\_body\_size 50M;**

**root /home/ubuntu/nginx;**

**index index.html;**

**location / {**

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

**}**

**location /api/ {**

**proxy\_pass http://localhost:8080;**

**proxy\_set\_header Host $host;**

**proxy\_set\_header X-Real-IP $remote\_addr;**

**}**

**location /api/v1/ws/ {**

**proxy\_pass http://localhost:8080;**

**proxy\_http\_version 1.1;**

**proxy\_set\_header Upgrade $http\_upgrade;**

**proxy\_set\_header Connection "Upgrade";**

**proxy\_set\_header Host $host;**

**proxy\_cache\_bypass $http\_upgrade;**

**proxy\_read\_timeout 3600s;**

**}**

**}**

**- Jenkins (front)**

pipeline {

agent any

stages {

stage('Secrets Setup') {

steps {

withCredentials([

file(credentialsId: 'env-file-fe', variable: 'EnvFile'),

]) {

sh '''

cp "$EnvFile" ./frontend/.env

chmod 644 ./frontend/.env

'''

}

}

}

stage('Build Frontend') {

steps {

sh '''

cd frontend

npm install

npm run build || true

'''

}

}

stage('Deploy to Nginx') {

steps {

sh '''

pwd

mkdir -p /home/ubuntu/nginx

rm -rf /home/ubuntu/nginx/\*

cp -r ./frontend/dist/\* /home/ubuntu/nginx/

cd /home/ubuntu/nginx/

ls

'''

}

}

}

}**- Jenkins (backend)**

pipeline {

agent any

environment {

FASTAPI\_CONTAINER\_NAME = 'fastapi-container'

FASTAPI\_DOCKER\_IMAGE = 'fastapi-app:latest'

CONTAINER\_NAME = 'backend'

DOCKER\_IMAGE = 'backend-image:latest'

RUNPOD\_HOST = 'ssh.runpod.io'

RUNPOD\_USER = '2j97ni5p02dqqk-64411020'

SSH\_CRED\_ID = 'runpod-ssh'

GIT\_REPO = 'https://gitlab.com/yourname/rvc-fastapi.git' // 변경 필요

TARGET\_DIR = '/workspace/RVC'

}

stages {

stage('Secrets Setup') {

steps {

withCredentials([

file(credentialsId: 'env-file', variable: 'EnvFile'),

]) {

sh '''

cp "$EnvFile" .env

chmod 644 .env

'''

}

}

}

stage('Build Backend') {

steps {

//백엔드 빌드

sh '''

cd ./server

chmod +x ./gradlew

# gradle 빌드 시 테스트 스킵 (-x test)

# 데몬 비활성화 (--no-daemon)

./gradlew clean build -x test --no-daemon

'''

}

}

stage('Docker Build & Deploy') {

steps {

sh '''

# Docker 이미지 빌드

docker build -t backend-image:latest ./server

# 기존 컨테이너 중지 및 삭제

docker stop backend || true

docker rm backend || true

# 새 컨테이너 실행

docker run -d \

--name backend \

--env-file .env \

--network fariytale\_jenkins-network \

--network-alias backend \

-p 8080:8080 \

backend-image:latest

'''

}

}

stage('Docker Cleanup') {

steps {

sh '''

echo "사용하지 않는 Docker 리소스 정리 중..."

# dangling 이미지 삭제 (태그가 없는 이미지)

docker image prune -f

# fastapi 관련 오래된 이미지 정리 (최신 2개 제외)

docker images --format "{{.ID}} {{.Repository}}" | grep fastapi | sort -k2 | awk 'NR>2 {print $1}' | xargs -r docker rmi -f

# backend 관련 오래된 이미지 정리 (최신 2개 제외)

docker images --format "{{.ID}} {{.Repository}}" | grep backend | sort -k2 | awk 'NR>2 {print $1}' | xargs -r docker rmi -f

# 미사용 볼륨 정리

docker volume prune -f

# 미사용 네트워크 정리

docker network prune -f

echo "Docker 리소스 정리 완료"

'''

}

}

}

}

3. **필수** 소프트웨어

Docker

Nginx

Docker-compose

4. 배포 및 설치

**1. 사전 준비 사항**

Ubuntu 22.04 이상 설치된 서버

다음 포트 오픈 필요: 80, 443, 22, 3000, 8080, 5000, 9000,3306 등

Docker / DockerCompose 환경 구성 완료

Jenkins 설치 및 docker 권한 부여 (sudo usermod -aG docker $USER / newgrp docker)

GitLab 또는 코드 저장소 준비

**2. 시작**

nginx 설정하기

jenkins에 pipeline 연결

각 script 실행하여 키기