포팅 매뉴얼

안녕하세요!

합격기원108배 팀의 " 세상에 나쁜 보호자는 있다 " 포팅 매뉴얼입니다! 🐶

1. 개발 환경

- **Server : Ubuntu 20.04 LTS**
- **SpringBoot : 3.1.5**
- **Spring Security : 6**
- **JDK : OpenJDK 17**
- **Nginx : nginx/1.18.0 (Ubuntu)**
- **MariaDB : MariaDB Server 11.1**
- **Jenkins : 2.430 (latest)**
- **Docker : 24.0.7 (latest)**
- **Docker-Compose : v2.20.2**
- **Unity : 2022.3.11 LTS**
- **IntelliJ : allowed any version**
- **Redis : allowed any version**
- **JWT : no version**

2. 설정 파일 및 환경 변수 정보

(1) Application.yml

```
server:
  port:
    8080
springdoc:
```

```
  packages-to-scan: com.senabo
  default-consumes-media-type: application/json;charset=UTF-8
  default-produces-media-type: application/json;charset=UTF-8
  swagger-ui:
    tags-sorter: alpha
    operations-sorter: alpha
  api-docs:
    path: /api-docs/json
    groups:
      enabled: true
  cache:
    disabled: true
spring:
  datasource:
    driver-class-name: org.mariadb.jdbc.Driver
    url: jdbc:mariadb://ssafy-db:3306/SENABO?characterEncoding=UTF-8&serverTimezone=UTC
    username: {username}
    password: {password}

  jpa:
    hibernate:
      ddl-auto: update
    properties:
      hibernate:
        format_sql: true
        show_sql: true
  data:
    redis:
      host: docker-redis
      port: 6379

logging:
  level:
    org.hibernate:
      type.descriptor.sql: trace
      SQL: DEBUG

fcm:
  service-account-file: './senabo-account-key.json'

jwt:
  secret: {secret_key}
```

(2) build.gradle

```
plugins {
    id 'java'
    id 'org.springframework.boot' version '3.1.5'
    id 'io.spring.dependency-management' version '1.1.3'
}

group = 'com.senabo'
version = '0.0.1-SNAPSHOT'

java {
    sourceCompatibility = '17'
}

repositories {
    mavenCentral()
}

dependencies {
    // JPA
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'

    // redis
    implementation 'org.springframework.boot:spring-boot-starter-data-redis'

    // security 설정
    implementation 'org.springframework.boot:spring-boot-starter-security'
    testImplementation 'org.springframework.security:spring-security-test'

    // jwt 설정
    implementation 'io.jsonwebtoken:jjwt-api:0.11.2'
    implementation 'io.jsonwebtoken:jjwt-impl:0.11.2'
    implementation 'io.jsonwebtoken:jjwt-jackson:0.11.2'
```

```
    // lombok
    compileOnly 'org.projectlombok:lombok'
    annotationProcessor 'org.projectlombok:lombok'

    // spring
    implementation 'org.springframework.boot:spring-boot-starter-web'
    developmentOnly 'org.springframework.boot:spring-boot-devtools'

    // mariaDB
    runtimeOnly 'org.mariadb.jdbc:mariadb-java-client'

    // swagger를 위한 springdoc dependency 추가
    implementation 'org.springdoc:springdoc-openapi-starter-webmvc-ui:2.0.2'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'

    // firebase fcm 설정
    implementation 'com.google.firebase:firebase-admin:9.2.0'
    implementation 'com.squareup.okhttp3:okhttp:4.11.0'

    // QueryDSL 설정
    implementation 'com.querydsl:querydsl-jpa:5.0.0:jakarta'
    annotationProcessor "com.querydsl:querydsl-apt:${dependencyManagement.importedProperties['querydsl.version']}:jakarta"
    annotationProcessor "jakarta.annotation:jakarta.annotation-api"
    annotationProcessor "jakarta.persistence:jakarta.persistence-api"
}

tasks.named('test') {
    useJUnitPlatform()
}

// querydsl 추가 설정 (선택 사항)
def querydslDir = "$buildDir/generated/querydsl"

// java source set에 Q클래스 적용
sourceSets {
    main.java.srcDirs += [querydslDir]
}

// Q클래스 location 위치 적용
tasks.withType(JavaCompile).configureEach {
    options.getGeneratedSourceOutputDirectory().set(file(querydslDir))
}

// gradle clean task 실행시 Q클래스 삭제
clean {
    delete file(querydslDir)
}
```

(3) Nginx 설치

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install nginx
sudo apt-get -y remove --purge nginx nginx-full nginx-common //삭제
```

(4) Nginx Conf

/etc/nginx/conf.d/default.conf

```
sudo vim /etc/nginx/conf.d/default.conf
```

```
upstream backend {
  server 0.0.0.0:8080;
}
server {
  listen 80;
  server_name 3.34.49.175 t0908.p.ssafy.io;

  location / {
```

```
    return 301 $scheme://senabo.co.kr$request_uri;
  }
}
server {
  listen 80;
  server_name senabo.co.kr www.senabo.co.kr;

  location /api {
    rewrite ^/api(/.*)$ $1 break;
    proxy_pass http://backend;
    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;
  }

  location / {
    proxy_pass http://frontend;
    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;
  }
}
```

저장 후 종료 `:wq!`

(5) Docker 설정

설치하기 및 시작

```
/ 를 기점으로 1줄 씩 입력 하세요
sudo apt-get -y install apt-transport-https ca-certificates curl gnupg-agent software-properties-common /
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) s
table" /
sudo apt-get update && sudo apt-get install docker-ce docker-ce-cli containerd.io /
docker -v
docker version
```

(5) Docker - Compose 설정

설치하기

```
/ 를 기점으로 1줄 씩 입력 하세요
sudo apt install jq /
DCVERSION=$(curl --silent https://api.github.com/repos/docker/compose/releases/latest | jq .name -r) /
DCDESTINATION=/usr/bin/docker-compose /
sudo curl -L https://github.com/docker/compose/releases/download/${DCVERSION}/docker-compose-$(uname -
s)-$(uname -m) -o $DCDESTINATION /
sudo chmod 755 $DCDESTINATION /
docker-compose -v
```

(6) 도커 네트워크 설정

```
docker network create deploy
docker network inspect $(docker network ls -q) // 네트워크 확인
docker container inspect ssafy-db //컨테이너 상세 정보 확인
```

3. 방화벽

UFW 설정하기 및 존재하지 않는다면 설치

```
sudo apt install ufw
```

```
sudo ufw default deny incoming // 모든 인바운드 연결 차단
sudo ufw default allow outgoing // 모든 아웃바운드 연결 허용
sudo ufw allow ssh // 22번 포트 허용
sudo ufw allow http // 80번 포트 허용
sudo ufw allow https // 443번 포트 허용

sudo ufw enable // 방화벽 켜기
```

---

4. 빌드

> **Back-spring**
> **Gradle 실행**
> **Bootjar 실행**

(1) Dockerfile - BE

> **들여쓰기 주의**

```
# Stage 1: Build with Gradle
FROM gradle:8.3-jdk17 as builder
WORKDIR /workspace
COPY build.gradle settings.gradle /workspace/
COPY src /workspace/src/
RUN gradle build -x test --no-daemon
ENV TZ Asia/Seoul

# Stage 2: Create a minimal JRE-based image for running the application
FROM eclipse-temurin:17-jdk-jammy
WORKDIR /app
COPY --from=builder /workspace/build/libs/*.jar app.jar
COPY --from=builder /workspace/src/main/resources/senabo-account-key.json .
CMD ["java", "-jar", "app.jar"]
```

(2) docker-compose.yml

> **들여쓰기 주의**

```
version: "3.8"
services:
  redis-docker:
    image: redis
    container_name: docker-redis
    volumes:
      - docker-redis:/data
    ports:
      - 6379:6379
    networks:
      - deploy

  application:
    build:
      context: /var/jenkins_home/workspace/senabo/senabo-spring/
      dockerfile: Dockerfile
```

```
    environment:
      SPRING_DATASOURCE_URL: jdbc:mariadb://ssafy-db:3306/SENABO?useUnicode=true
      SPRING_DATASOURCE_USERNAME: {username}
      SPRING_DATASOURCE_PASSWORD: {password}
      TZ: "Asia/Seoul"
    ports:
      - 8080:8080
    networks:
      - deploy

networks:
  deploy:
    external: true

volumes:
  docker-redis:
```

5. 자동 배포

Jenkins 설치 후 설정

(1) Jenkins 설정

Jenkins 설치(Windows 환경)(1)

우선 Jenkins는 흔히 말하는 CI/CD 중 지속적 통합(Continuous Integration)을 구현하기 위한 서비스이다. 개발 중인 저장소(git, svn 등)에 업로드된 소스를 테스트, 빌드, 빌드 후 작업등을 자동 동작하게 해 주어 (이 자체가 지속적 통합) 그만큼 개발자의 리소스 소모가 줄어든다. 1. Installer Download https://www.jenkins.io/download/ Jenkins

https://lock.tistory.com/2

(2) Swap 메모리 선언

```
df -h # 용량 할당
sudo fallocate -l 8G /swapfile # Swap 영역 할당 (일반적으로 서버 메모리의 2배)
sudo chmod 600 /swapfile # Swapfile 권한 수정
sudo mkswap /swapfile # Swapfile 생성
sudo swapon /swapfile # Swapfile 활성화
free -h # swap 영역이 할당 되었는지 확인
```

(3) Jenkins Nginx 설정

Reverse proxy - Nginx

Jenkins – an open source automation server which enables developers around the world to reliably build, test, and deploy their software

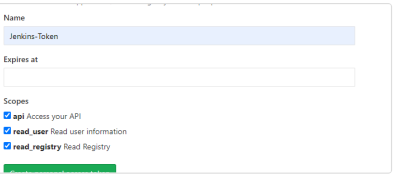
https://www.jenkins.io/doc/book/system-administration/reverse-proxy-configuration-with-jenkins/reverse-proxy-configuration-nginx/

(4) GitLab 연동

Jenkins - Git 연동 (1) - 연결

1. Git 설치 후 Jenkins 연동 https://git-scm.com/ 깃 설치 안된 상태이면, 해당 사이트 들어가서 설치. 설치 후, Jenkins 관리 -> Global Tool Configuration 들어가서 설치된 깃 정보 입력. 2. 깃 계정 AccessToken 확인 깃 Access Tokens 확인 깃 로그인 후 계정 아이콘 드롭 메뉴 -> Preferences -> 왼쪽 사이트 메뉴의 Access Tokens

https://lock.tistory.com/6



(5) Pipeilne Script

```
pipeline{
  agent any

  options{
    timeout(time: 1, unit: 'HOURS')
  }
    stages{
      stage('git clone') {
        steps {
          git url: "https://lab.ssafy.com/s09-final/S09P31A108.git",
            branch: "backend",
            credentialsId: "senabo"
            sh "ls -al"
        }
      }
      stage('set backend enviornment'){
        steps{
          dir("./senabo-spring"){
              sh '''
                cp /var/jenkins_home/util/senabo/Dockerfile /var/jenkins_home/workspace/senabo/senabo-spring/Dockerfile
                cp /var/jenkins_home/util/senabo/docker-compose.yml /var/jenkins_home/workspace/senabo/docker-compose.yml
                cp /var/jenkins_home/util/senabo/application.yml /var/jenkins_home/workspace/senabo/senabo-spring/src/main/resources/ap
                cp /var/jenkins_home/util/senabo/senabo-account-key.json /var/jenkins_home/workspace/senabo/senabo-spring/src/main/reso
              '''
              // sh "chmod +x ./gradlew"
              // sh "./gradlew clean"
              // sh "./gradlew build -x test"
          }
        }
      }
    stage('Docker down'){
      steps{
        dir("/var/jenkins_home/workspace/senabo"){
          echo "Docker compose down"
          sh "docker-compose -f docker-compose.yml down --rmi all"
          sh "docker ps -a"
        }
      }
    }
    stage('Docker build'){
      steps{
        echo "docker compose build"
        dir("/var/jenkins_home/workspace/senabo"){
          sh "docker-compose -f docker-compose.yml build --no-cache"
        }
      }
      post{
        success{
          echo "Success to build"
        }
        failure{
          echo "Docker build failed. clear unused file"
          sh "docker system prune -f"
          error 'pipeline aborted'
        }
      }
    }
    stage('Docker up'){
      steps{
        echo "docker compose up"
        sh "docker-compose -f /var/jenkins_home/workspace/senabo/docker-compose.yml up -d"
      }
    }
    stage('Docker clear'){
      steps {
        sh "docker system prune -f"
      }
    }
```

```
  }
}
```

(6) 최종 Jenkins 화면

Stage View

	git clone	set backend enviornment	Docker down	Docker build	Docker up	Docker clear
Average stage times: (Average full run time: ~1min 19s)	979ms	376ms	1s	1min 7s	7s	1s
#137 11월 16일 10:25 1 commit	883ms	370ms	1s	1min 7s	7s	1s
#136 11월 16일 10:18 1 commit	950ms	386ms	1s	1min 5s	7s	1s
#135 11월 16일 02:08 1 commit	964ms	368ms	1s	1min 9s	7s	1s
#134 11월 16일 01:37 1 commit	1s	380ms	1s	1min 6s	7s	1s
#133 11월 15일 19:45 1 commit	878ms	373ms	1s	1min 10s	7s	1s