포팅 메뉴얼

0. 목차

---

1. 사용 프로그램 버전

프로그램	버전
JVM	17
Spring Boot	3.2.1
Spring Security	6.2.1
Thymeleaf	3.2.1
gson	2.10.1
Spring cloud	3.1.0
JPA	6.4.1.Final
MariaDB	11.2.3

Redis	7.2.4
Jenkins	2.441
Docker	25.0.2
Vue	3.3.11
Vite	5.0.12
Tailwind	3.4.1
TypeScript	5.3.0
Node.js	20.11.0
npm	10.2.4
tiptap/vue-3	2.2.2
pinia	2.1.17

---

2. 배포 포트

BackEnd

- Spring Boot Application : `8089:8080`

FrontEnd

- NginX (vue 빌드파일) : `8083:80`

NginX (서버)

- 백엔드 서버 : `8082:8089`
- 프론트엔드 서버 : `443:8083`

서버

- MariaDB : `3306:3306`
- Redis : `6379:6379`
- Jenkins : `8081:8080`

3. 환경변수 파일

BackEnd

- application.yml

```
server:
  servlet:
    encoding:
      charset: UTF-8
      force: true
aladinApiConfig:
  serviceKey: ttbkimsw28261657008
  bestSellerCallBackUrl: http://www.aladin.co.kr/ttb/api/Item
  searchCallBackUrl: http://www.aladin.co.kr/ttb/api/ItemSear
jwt:
  secret: SecretKeyForOurTeamA801AndAllForThoseWhoUseOurServi
kakao:
  key: 2471aa02851a522693aaaed5e9875a3e
  redirectUri: https://i10a801.p.ssafy.io/kakaologin

spring:
  datasource:
    url: jdbc:mariadb://i10a801.p.ssafy.io:3306/polaris?serve
    username: root
    password: a801ssafy
  jpa:
    open-in-view: false
    generate-ddl: true
    show-sql: true
    hibernate:
      ddl-auto: update
    properties: # property 사용 설정
      hibernate: # hibernate property 설정
```

```
        format_sql: true
  data:
    redis:
      host: i10a801.p.ssafy.io
      port: 6379
  mail:
    host: smtp.gmail.com # SMTP 서버 호스트
    port: 587 # SMTP 서버 포트
    username: hjh70902000
    password: ocsgtjzcqrddcudm
    properties:
      mail:
        smtp:
          auth: true # 사용자 인증 시도 여부
          timeout: 180000 # Socket Read Timeout 시간 (설정은 3분
          starttls:
            enable: true # 7
  servlet:
    multipart:
      max-file-size: 10MB
      max-request-size: 10MB
  cloud:
    aws:
      credentials:
        access-key: AKIART7UDDCJ6FRXPOEF
        secret-key: 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
      s3:
        enabled: true
        region: ap-northeast-2
      region:
        static: ap-northeast-2
```

4. 배포 환경 구축 (CI/CD)

Jenkins

- 하나의 repository를 사용하기 때문에 backend 메인 브랜치와 frontend 메인브랜치의 웹훅을 받아 자동으로 스크립트 실행

> ❗ ***workspace로 쓸 폴더 내에 /env 디렉토리를 만들어, `application.yml`, `.evn` 파일을 넣어줘야 한다.***

BackEnd

- jenkins 스크립트

```
pipeline {
    agent any

    environment {
        imageName = "coderyard/test-repo" // docker 허브에 등록
        registryCredential = 'dockerhub-token' // docker 허브
        dockerImage = ''

        containerName = 'polaris_backend' // 서버에 등록될 conta

        releaseServerAccount = 'ubuntu' // ssh로 서버 접속 시 사
        releaseServerUri = 'https://i10a801.p.ssafy.io' // 서버
                releaseServerIPAddr = '172.26.0.13' // 서버 ip
        releasePort = '8089' // container 포트포워딩 정보

    }

    stages {
        stage('Git Clone') { // 프로젝트를 git clone
            steps {
                git branch: 'develop-backend',
                    credentialsId: 'Jinhajenkins', // GitLab
                    url: 'https://lab.ssafy.com/s10-webmobile
            }
        }
                stage('application.yml copy') { // 따로 관리 중
```

```
                    steps {
                        sh "mkdir -p ./back-end/polaris/src/m
                        sh "cp -f ../env/application.yml ./ba
                    }
                }
        stage('Jar Build') { // 프로젝트 파일 빌드
            steps {
                dir ('back-end/polaris') {
                    sh 'chmod +x ./gradlew'
                    sh './gradlew clean bootJar'
                    // sh './gradlew build'
                }
            }
        }

        stage('Image Build & DockerHub Push') { // 빌드된 파일 
            steps {
                dir('back-end/polaris') {
                    script {
                        docker.withRegistry('', registryCrede
                            sh "docker buildx create --use --
                            sh "pwd"
                            sh "docker buildx build --platfor
                            sh "docker buildx build --platfor
                        }
                    }
                }
            }
        }
        stage('Before Service Stop') { // 서비스를 다시 컨테이너로
            steps {
                sshagent(credentials: ['ubuntu-a801']) {
                    sh '''
                    if test "`ssh -o StrictHostKeyChecking=no
                    ssh -o StrictHostKeyChecking=no $releaseS
                    ssh -o StrictHostKeyChecking=no $releaseS
                    ssh -o StrictHostKeyChecking=no $releaseS
                    fi
```

```
                    '''
                }
            }
        }
        stage('DockerHub Pull') { // docker 이미지 가져옴
            steps {
                sshagent(credentials: ['ubuntu-a801']) {
                    // sh "ssh -o StrictHostKeyChecking=no $r
                    sh "ssh -o StrictHostKeyChecking=no $rele
                }
            }
        }
        stage('Service Start') { // docker 컨테이너 만들고 실행
            steps {
                sshagent(credentials: ['ubuntu-a801']) {
                    // sh '''
                    //     ssh -o StrictHostKeyChecking=no $r
                    // '''
                     sh '''
                         ssh -o StrictHostKeyChecking=no $rel
                     '''
                }
            }
        }
        stage('Service Check') { // 연결 체크
            steps {
                sshagent(credentials: ['ubuntu-a801']) {
                    sh '''
                        #!/bin/bash

                        for retry_count in \$(seq 20)
                        do
                          if curl -s "http://i10a801.p.ssafy.
                          then
                              curl -d '{"text":"Release Compl
                              break
                          fi
```

```
                            if [ $retry_count -eq 20 ]
                            then
                              curl -d '{"text":"Release Fail"}'
                              exit 1
                            fi

                            echo "The server is not alive yet.
                            sleep 5
                          done
                    '''
                }
            }
        }
    }
}
```

- Dockerfile (프로젝트 최상단에 위치)

```
FROM docker
COPY --from=docker/buildx-bin:latest /buildx /usr/libexec/doc

FROM openjdk:17-jdk
ADD ./build/libs/polaris-0.0.1-SNAPSHOT.jar app.jar
ENTRYPOINT ["java", "-jar", "/app.jar"]
```

FrontEnd

- jenkins 스크립트

```
pipeline {
    agent any
    tools {nodejs "nodejs"}

    environment {
        imageName = "coderyard/polaris-front" // docker hub의
        registryCredential = 'dockerhub-token' // docker hub
```

```
        dockerImage = ''

        releaseServerAccount = 'ubuntu' // ssh 연결 시 사용할 us
        releaseServerUri = 'i10a801.p.ssafy.io' // 서비스 url
        containerName = 'polaris_frontend' // 컨테이너 이름
        containerPort = '80' // 컨테이너 포트를
        releasePort = '8083' // 배포포트로 포트포워딩
    }

    stages {
        stage('Git Clone') { // 프로젝트 소스파일 clone
            steps {
                git branch: 'develop-frontend',
                    credentialsId: 'Jinhajenkins',
                    url: 'https://lab.ssafy.com/s10-webmobile
            }
        }
                stage('.env copy') {
                        steps {
                                sh "mkdir -p ./front-end/pola
                                sh "cp -f ../env/.env ./front
                        }
                }
        stage('Node Build') { // 프로젝트 build
            steps {
                dir ('front-end/polaris') {
                    sh 'npm install'
                    sh 'npm run build'
                }
            }
        }
        stage('Image Build & DockerHub Push') { // 빌드된 파일
            steps {
                dir('front-end/polaris') {
                    script {
                        docker.withRegistry('', registryCrede
                            sh "docker buildx create --use --
                            sh "docker buildx build --platfor
```

```
                        sh "docker buildx build --platfor
                    }
                }
            }
        }
        stage('Before Service Stop') { // 서비스 중단 전 기존 컨테
            steps {
                sshagent(credentials: ['ubuntu-a801']) {
                    sh '''
                    if test "`ssh -o StrictHostKeyChecking=no
                    ssh -o StrictHostKeyChecking=no $releaseS
                    ssh -o StrictHostKeyChecking=no $releaseS
                    ssh -o StrictHostKeyChecking=no $releaseS
                    fi
                    '''
                }
            }
        }
        stage('DockerHub Pull') { // docker hub에서 프론트엔드 이
            steps {
                sshagent(credentials: ['ubuntu-a801']) {
                    sh "ssh -o StrictHostKeyChecking=no $rele
                }
            }
        }
        stage('Service Start') { // pull된 이미지 이용하여 docker
            steps {
                sshagent(credentials: ['ubuntu-a801']) {
                    sh '''
                        ssh -o StrictHostKeyChecking=no $rele
                    '''
                }
            }
        }
        stage('Service Check') { // 연결 체크
            steps {
                sshagent(credentials: ['ubuntu-a801']) {
```

```
                    sh '''
                        #!/bin/bash

                        for retry_count in \$(seq 20)
                        do
                          if curl -s "http://$releaseServerUr
                          then
                              curl -d '{"text":"Release Compl
                              break
                          fi

                          if [ $retry_count -eq 20 ]
                          then
                            curl -d '{"text":"Release Fail"}'
                            exit 1
                          fi

                          echo "The server is not alive yet.
                          sleep 5
                        done
                    '''
                }
            }
        }
    }
}
```

- Dockerfile (프로젝트 최상단에 위치)

```
# nginx 이미지 사용
FROM nginx:latest

# root에 /app 폴더 생성
RUN mkdir /app

# work dir 고정
WORKDIR /app
```

```
# work dir에 build 폴더 생성
RUN mkdir ./build

# host pc의 현재 경로의 build 폴더를 work dir의 build 폴더로 복사
ADD ./dist ./build

# nginx의 default.conf 삭제
RUN rm /etc/nginx/conf.d/default.conf

# host pc의 nginx.conf를 아래 경로에 복사
# nginx config 파일 또한 프로젝트 최상단에 위치
COPY ./nginx.conf /etc/nginx/conf.d

# 80 포트 개방
EXPOSE 80

# container 실행 시 자동으로 실행할 command. nginx 시작함
CMD ["nginx", "-g", "daemon off;"]
```

- nginx.conf (프로젝트 최상단에 위치)

```
server {
    listen 80;
    location / {
        root    /app/build;
        index   index.html;
        try_files $uri $uri/ /index.html;
    }
}
```

NginX(Server 자체)

- /etc/nginx/nginx.conf

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

events {
        worker_connections 768;
        # multi_accept on;
}

http {

        ##
        # Basic Settings
        ##

        sendfile on;
        tcp_nopush on;
        tcp_nodelay on;
        keepalive_timeout 65;
        types_hash_max_size 2048;
        # server_tokens off;

        # server_names_hash_bucket_size 64;
        # server_name_in_redirect off;

        include /etc/nginx/mime.types;
        default_type application/octet-stream;

        ##
        # SSL Settings
        ##

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

        ##
        # Logging Settings
```

```
        ##

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

        proxy_set_header Connection '';
        proxy_http_version 1.1;
        ##
        # Gzip Settings
        ##

        gzip on;

        ##
        # Virtual Host Configs
        ##

        include /etc/nginx/conf.d/*.conf;
        include /etc/nginx/sites-enabled/*; // sites-availabl

        server {
                location /chat {
                        proxy_pass https://i10a801.p.ssafy.io
                        proxy_set_header Connection '';
                        proxy_set_header Cache-Control 'no-ca
                        proxy_set_header X-Accel-Buffering 'n
                        proxy_set_header Content-Type 'text/e
                        proxy_buffering off;
                        chunked_transfer_encoding on;
                        proxy_read_timeout 86400s;
                }
        }
}
```

- /etc/nginx/sites-available/default

```
server {
    listen 80;
```

```
    server_name i10a801.p.ssafy.io;
    return 301 https://$host$request_uri;
}

server {
    listen 443 ssl;
    server_name i10a801.p.ssafy.io;

    ssl_certificate /etc/letsencrypt/live/i10a801.p.ssafy.io/
    ssl_certificate_key /etc/letsencrypt/live/i10a801.p.ssafy

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

    location / {
        proxy_pass http://localhost:8083; # 맞게 변경한다.
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forward

    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
    }
}

server {
    listen 8082 ssl;
    server_name i10a801.p.ssafy.io;

    ssl_certificate /etc/letsencrypt/live/i10a801.p.ssafy.io/
    ssl_certificate_key /etc/letsencrypt/live/i10a801.p.ssafy

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

    location / {
        proxy_pass http://i10a801.p.ssafy.io:8089; # 맞게 변경한
```

```
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forward
    proxy_set_header X-NginX-Proxy true;
    }

    location /stomp {
        proxy_pass http://i10a801.p.ssafy.io:8089; # 맞게 변경한

        proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
        proxy_set_header Host $host;
    }
}
```

포트 설정 (ufw)

22	ALLOW	Anywhere
80	ALLOW	Anywhere
89	ALLOW	Anywhere
443	ALLOW	Anywhere
3306	ALLOW	Anywhere
8080	ALLOW	Anywhere
8081	ALLOW	Anywhere
8082	ALLOW	Anywhere
8083	ALLOW	Anywhere
8089	ALLOW	Anywhere