포팅메뉴얼

1. Project Skill Stack

1.1 Backend API Server

```
Java                    17.0.8
Spring Boot             3.1.5
Spring data Jpa         3.1.5
Querydsl                5.0.0
gradle                  8.3
Swagger                 5.2.0
```

1.2 Frontend

```
react                   18
next                    13.5.6
typescript              5
node                    18.18.0
Storybook
SWR
Recoil, Recoil-persist
```

1.3 ChatBot Server

```
Flask                   3.0.0
Python                  3.8.7
```

1.4 INFRA

```
AWS EC2 (ubuntu 20.04 LTS) Memory 16GB, Storage 311GB
Docker                  24.0.6
Redis                   7.x.x
MySQL                   8.0.33
Nginx                   1.18.0
Jenkins                 2.428
```

---

2. Project Environment File

각 값들을 배포 환경에 맞게 알맞게 변경하여 사용합니다.

2.1 Backend Production yaml file (application-dev.yml)

```
spring:
  datasource:
    url: jdbc:mysql://YOUR_DATA_BASE_SERVER_URL
    driver-class-name: com.mysql.cj.jdbc.Driver
    username: YOUR_DATA_BASE_USER_NAME
    password: YOUR_DATA_BASE_USER_PASSWORD

  jpa:
    hibernate:
      ddl-auto: none
    properties:
      hibernate:
        default_batch_fetch_size: 100
        format_sql: true
        jdbc:
          time_zone: Asia/Seoul
    show-sql: true

  redis:
    host: YOUR_REDIS_HOST
    port: YOUR_REDIS_PORT
    password: YOUR_REDIS_PASSWORD
    lettuce:
      pool:
        max-active: 10
        max-idle: 10
        min-idle: 2
```

```
security:
  jwt:
    token:
      secret-key: YOUR_JWT_SIGN_KEY
      expire-length:
        access: 1800000
        refresh: 604800000
  encrypt:
    key: 12345678910111213

springdoc:
  api-docs:
    path: /docs
    enabled: true

  swagger-ui:
    path: /swagger-ui
    enabled: true

server:
  forward-headers-strategy: FRAMEWORK
```

YOUR_DATA_BASE_SERVER_URL: MySQL 주소

YOUR_DATA_BASE_USER_NAME: MySQL 유저명

YOUR_DATA_BASE_USER_PASSWORD: MySQL 유저 패스워드

YOUR_REDIS_HOST: Redis 호스트
YOUR_REDIS_PORT: Redis 포트번호
YOUR_REDIS_PASSWORD: Redis 패스워드
YOUR_JWT_SIGN_KEY: jwt 서명에 사용될 키 값을 입력합니다. `hmacSha256` 를 사용하므로 64바이트
이상의 키값을 입력해야합니다.

2.2 Backend Test yaml file (application-test.yml)

```
spring:
  datasource:
    url: jdbc:h2:mem:db;MODE=MYSQL;DB_CLOSE_DELAY=-1;DB_CLOSE_ON_EXIT=FALSE
    driver-class-name: org.h2.Driver
    username: sa
```

```
  jpa:
    hibernate:
      ddl-auto: auto
    properties:
      hibernate:
        default_batch_fetch_size: 100
        format_sql: true
        jdbc:
          time_zone: Asia/Seoul
    show-sql: true

  redis:
    host: YOUR_REDIS_HOST
    port: YOUR_REDIS_PORT
    password: YOUR_REDIS_PASSWORD
    lettuce:
      pool:
        max-active: 10
        max-idle: 10
        min-idle: 2

security:
  jwt:
    token:
      secret-key: YOUR_JWT_SIGN_KEY
      expire-length:
        access: 1800000
        refresh: 604800000
  encrypt:
    key: 12345678910111213

springdoc:
  api-docs:
    path: /docs
    enabled: true

  swagger-ui:
    path: /swagger-ui
    enabled: true

server:
  forward-headers-strategy: FRAMEWORK
```

YOUR_REDIS_HOST: Redis 호스트
YOUR_REDIS_PORT: Redis 포트번호
YOUR_REDIS_PASSWORD: Redis 패스워드

YOUR_JWT_SIGN_KEY: jwt 서명에 사용될 키 값을 입력합니다. `hmacSha256` 를 사용하므로 64바이트

이상의 키값을 입력해야합니다.

2.3 Frontend Env File (.env.production)

```
NEXT_PUBLIC_API_URL=YOUR_API_SERVER_URL
```

YOUR_API_SERVER_URL: API 서버 주소

---

3. Build

3.1 Backend

로컬 환경에서 실행 시 (Java 17 설치 필수)

```
./gradlew clean test
./gradlew build
cd /build/libs
nohup java -jar 빌드파일명 &
```

도커 컨테이너 기반 실행

```
docker build -t 이미지명 .
docker image prune -f
docker run --name 컨테이너명 -d --network host -e SPRING_PROFILES_ACTIVE=dev 이미지명
```

3.2 Frontend

로컬 환경에서 실행 시 (node, npm 설치 필수)

```
npm install
npm run build
npm run start
```

도커 컨테이너 기반 실행

```
docker build -t 이미지명 .
docker image prune -f
docker run --name 컨테이너명 -d --network host 이미지명
```

3.3 Flask

도커 컨테이너 기반 실행

```
docker build -t 이미지명 .
docker image prune -f
docker run --name 컨테이너명 -d --network host 이미지명
```

---

4. Jenkins CI / CD Script

4.1 Backend API Server CI / CD Script

```
pipeline {
  agent any
    stages {
        stage('git pull') {
          steps {
            cleanWs()
            git branch: 'dev/be', credentialsId: 'omrdptoken102', url: 'https://lab.ssafy.
com/s09-final/S09P31A102'
          }
        }

        stage('set properties') {
```

```
          steps {
            sh """
                cp ${JENKINS_HOME}/workspace/properties/application.yml ${WORKSPACE}/backe
nd/omr/src/main/resources
                cp ${JENKINS_HOME}/workspace/properties/application-release.yml ${WORKSPAC
E}/backend/omr/src/main/resources
            """
          }
        }

        stage('build jar') {
          steps{
            sh """
              cd ${WORKSPACE}/backend/omr
              ./gradlew bootjar
            """
          }
        }

        stage('run container'){
          steps{
            script {
              sh """
                docker rm -f omr-backend || true
                cd ${WORKSPACE}/backend/omr
                docker build -t omr .
                docker image prune -f
                docker run --name omr-backend -d --network host -e SPRING_PROFILES_ACTIVE=
release omr
              """
            }
          }
        }
    }
}
```

4. 2 Frontend Server CI / CD Script

```
pipeline {
  agent any
    stages {
        stage('git pull') {
          steps {
            cleanWs()
            git branch: 'dev/fe', credentialsId: 'omrdptoken102', url: 'https://lab.ssafy.
com/s09-final/S09P31A102'
          }
        }
```

```
        stage('set properties') {
          steps {
            sh "cp ${JENKINS_HOME}/workspace/properties/.env ${WORKSPACE}/frontend/omr"
          }
        }

        stage('run container') {
          steps {
            sh """
                docker rm -f omr-frontend || true
                cd ${WORKSPACE}/frontend/omr
                docker build -t omr-fe .
                docker image prune -f
                docker run --name omr-frontend -d --network host omr-fe
            """
          }
        }
    }
}
```

5. Nginx Config

```
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;

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

```
        ##
        # SSL Settings
        ##

        ssl_protocols TLSv1 TLSv1.1 TLSv1.2 TLSv1.3; # Dropping SSLv3, ref: POODLE
        ssl_prefer_server_ciphers on;

        ##
        # Logging Settings
        ##

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

        ##
        # Gzip Settings
        ##

        gzip on;
        ##
        # Virtual Host Configs
        ##
        include /etc/nginx/conf.d/*.conf;
        include /etc/nginx/sites-enabled/*;
}
```

nginx.conf

```
server {
    server_name www.omrcs.com;

    location / {
        rewrite %(/.*)$ $1 break;
        proxy_pass http://127.0.0.1:3000;
        proxy_redirect off;
    }

    location /omr-api/ {
        rewrite ^/omr-api(/.*)$ $1 break;
        proxy_pass http://127.0.0.1:8080;
        proxy_set_header X-Forwarded-Prefix /omr-api;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header Host $http_host;
        proxy_redirect off;
```

```
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    location /omr-chatbot/ {
        rewrite ^/omr-chatbot(/.*)$ $1 break;
        proxy_pass http://127.0.0.1:5000;
        proxy_set_header X-Forwarded-Prefix /omr-chatbot;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header Host $http_host;
        proxy_redirect off;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    listen [::]:443 ssl ipv6only=on; # managed by Certbot
    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/www.omrcs.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/www.omrcs.com/privkey.pem;

    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot

    location /jenkins {
        include /etc/nginx/proxy_params;
        proxy_pass http://localhost:9090/jenkins;
        proxy_redirect default;
        proxy_http_version 1.1;
    }

}

server {
    if ($host = www.omrcs.com) {
        return 301 https://$host$request_uri;
    }


    listen 80 default_server;
    listen [::]:80 default_server;

    server_name www.omrcs.com;
    return 404; # managed by Certbot
}
```

/sites-enable/default

1. ssl 인증서는 sertbot을 통해 발급받은 후, nginx의 pem키의 path를 발급받은 키값의 위치로 변경