Coffeeing Porting Manual

1. Project Skill Stack

1.1 Backend API Server

```
Spring Boot 2.7.15
Java 17
Query DSL 5.0
gradle 8.3
jacoco 0.8.10
```

1.2 Frontend

```
Node 18.x.x
React 18.2.0
Tailwind CSS 3.3.3
TypeScript 4.9.5
Redux 8.1.2
Axios 1.5.0
env-cmd 10.1.0
```

1.3 Recommand Server

```
Python 3.11.4
pipenv
fastapi 0.103.1
numpy 2.6.3
pandas 2.1.1
scikit-learn 1.3.1
sqlalchemy 2.0.21
uvicorn 0.23.2
python-dotenv 1.0.0
scikit-surprise 1.1.3
```

1.4 INFRA

```
AWS EC2 (ubuntu 20.04 LTS) Memory 16GB, Storage 311GB
MySQL 8.0.34
Redis 7.2.1
Docker Community 24.0.6
Jenkins 2.414.1
Sonar Qube Community EditionVersion 9.9.2
Nginx 1.18.0
```

---

2. Project Environment File

각 값들을 배포 환경에 맞게 알맞게 변경하여 사용합니다.

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

```yaml
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

  mvc:
    pathmatch:
      matching-strategy: ant_path_matcher

  security:
    oauth2:
      client:
        registration:
          google:
            client-id: YOUR_GOOGLE_CLIENT_ID
            client-secret: YOUR_GOOGLE_CLIENT_SECRET
            scope: profile, email

cloud:
  aws:
    s3:
      bucket: coffeeing
      objectKey: postImage
      expire-in: 300000
    credentials:
      access-key: YOUR_S3_ACCESS_KEY
      secret-key: YOUR_S3_SECRET_KEY
    region:
      static: ap-northeast-2
    stack:
      auto: false

jwt:
  header: Authorization
  grant-type: Bearer
  secret: YOUR_JWT_SIGN_KEY
  access-token-claim-key: email
  access-token-duration: 7200000
  refresh-token-duration: 1209600000

server:
  servlet:
    context-path: /YOUR_PRODUCTION_API_PREFIX

logging:
  level:
    com:
      amazonaws:
        util:
          EC2MetadataUtils: error
front:
  redirect-url: YOUR_FRONT_DEPLOY_URL/oauth
```

```
fast-api:
  baseUrl: "YOUR_RECOMMEND_SERVER_URL/rec"
  recByParamUrl: "/collab"
  recByProductUrl: "/content"
```

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_S3_ACCESS_KEY: AWS 에서 발급받은 S3 액세스 키
YOUR_S3_SECRET_KEY: AWS 에서 발급받은 S3 시크릿 키

YOUR_GOOGLE_CLIENT_ID: 구글 클라우드 콘솔에서 발급받은 클라이언트 ID

YOUR_GOOGLE_CLIENT_SECRET: 구글 클라우드 콘솔에서 발급받은 클라이언트 Secret

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

YOUR_PRODUCTION_API_PREFIX: API 서버의 prefix를 입력합니다. (미사용시 제거)
YOUR_RECOMMEND_SERVER_URL: 추천 서버 URL

YOUR_FRONT_DEPLOY_URL: OAuth 로그인 후 리다이렉트될 프론트엔드 주소

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

  security:
    oauth2:
      client:
        registration:
          google:
            client-id: 620126596578-lho8n03abmubidf1pgq4tpitbt39mvmg.apps.googleusercontent.com
            client-secret: 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
            scope: profile, email

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

  mvc:
    pathmatch:
      matching-strategy: ant_path_matcher
  redis:
    host: YOUR_REDIS_HOST
    port: YOUR_REDIS_PORT
    password: YOUR_REDIS_PASSWORD

  mvc:
    pathmatch:
      matching-strategy: ant_path_matcher

  security:
    oauth2:
      client:
        registration:
          google:
            client-id: YOUR_GOOGLE_CLIENT_ID
            client-secret: YOUR_GOOGLE_CLIENT_SECRET
            scope: profile, email

cloud:
  aws:
    s3:
      bucket: coffeeing
      objectKey: postImage
      expire-in: 300000
    credentials:
      access-key: YOUR_S3_ACCESS_KEY
      secret-key: YOUR_S3_SECRET_KEY
    region:
      static: ap-northeast-2
    stack:
      auto: false

jwt:
  header: Authorization
  grant-type: Bearer
  secret: YOUR_JWT_SIGN_KEY
  access-token-claim-key: email
  access-token-duration: 7200000
  refresh-token-duration: 1209600000

server:
  servlet:
    context-path: /YOUR_PRODUCTION_API_PREFIX

logging:
  level:
    com:
      amazonaws:
        util:
          EC2MetadataUtils: error
front:
  redirect-url: YOUR_FRONT_DEPLOY_URL/oauth

fast-api:
  baseUrl: "YOUR_RECOMMEND_SERVER_URL/rec"
  recByParamUrl: "/collab"
  recByProductUrl: "/content"
```

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

YOUR_S3_ACCESS_KEY: AWS 에서 발급받은 S3 액세스 키
YOUR_S3_SECRET_KEY: AWS 에서 발급받은 S3 시크릿 키

YOUR_GOOGLE_CLIENT_ID: 구글 클라우드 콘솔에서 발급받은 클라이언트 ID

YOUR_GOOGLE_CLIENT_SECRET: 구글 클라우드 콘솔에서 발급받은 클라이언트 Secret

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

YOUR_PRODUCTION_API_PREFIX: API 서버의 prefix를 입력합니다. (미사용시 제거)
YOUR_RECOMMEND_SERVER_URL: 추천 서버 URL

YOUR_FRONT_DEPLOY_URL: OAuth 로그인 후 리다이렉트될 프론트엔드 주소

2.3 Frontend Env File (.env.production)

```
REACT_APP_BASE_URL=YOUR_DEPLOY_SERVER_URL
REACT_APP_BASE_API_URL=YOUR_API_SERVER_URL
```

2.4 Frontend Dev Server Env File (.env.development, .env.local)

```
REACT_APP_MODE=development
REACT_APP_BASE_URL=YOUR_DEPLOY_SERVER_URL
REACT_APP_BASE_API_URL=YOUR_API_SERVER_URL
```

YOUR_DEPLOY_SERVER_URL: 프론트엔드 배포 주소

YOUR_API_SERVER_URL: API 서버 주소

2.5 Fast API Env File (.env)

```
DB_URL=YOUR_DATA_BASE_SERVER_HOST_PORT
DB_SCHMEA=YOUR_DATA_BASE_SCHEMA
DB_USER=YOUR_DATA_BASE_USER_NAME
DB_PWD=YOUR_DATA_BASE_USER_PASSWORD
```

YOUR_DATA_BASE_SERVER_HOST_PORT: 데이터베이스 호스트 & 포트번호

YOUR_DATA_BASE_SCHEMA: 접근할 테이블

YOUR_DATA_BASE_USER_NAME: 데이트베이스 유저명

YOUR_DATA_BASE_USER_PASSWORD: 데이터베이스 유저 패스워드

---

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 dev
```

각 환경설정 파일에 맞게 npm run dev, npm run local, npm run prd로 개발서버로 실행시킨다.

배포 시 npm build prd로 빌드후, 빌드 결과물을 배포될 위치로 위치시킨다.

3.3 FastAPI

로컬 환경에서 실행시 (python 3.11, pip, pipenv 설치 필수)

```
pipenv install
pipenv shell
uvicorn app.main:app
```

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
  tools {
    gradle('gradle8.3')
  }

  stages {
    stage('Git Pull') {
      steps {
        git branch: 'REPLACE_BRANCH_NAME', credentialsId: 'accessToken', url: 'REPLACE_YOUR_SVM_URL'
      }
    }
    stage('Pre Build Clean up') {
      steps {
        script {
          if (fileExists('backend/coffeeing/build')) {
            echo 'Build directory exists. REMOVING'
            fileOperations([folderDeleteOperation('backend/coffeeing/build')])
          }
        }
      }
    }
    stage('Copy Property Files') {
      steps {
        sh 'cp /REPLACE_YOUR_YAML_FILE_PATH/application-dev.yml /REPLACE_YOUR_JENKINS_WORKSPACE_PATH/backend/coffeeing/src/main/resources'
        sh 'cp /REPLACE_YOUR_YAML_FILE_PATH/application-test.yml /REPLACE_YOUR_JENKINS_WORKSPACE_PATH/backend/coffeeing/src/main/resources'
      }
    }
    stage('Gradlew Test') {
      steps {
        script {
          sh '''
            cd "${WORKSPACE}"/backend/coffeeing
            ./gradlew clean test
          '''
        }
      }
    }
    stage('SonarQube') {
      steps{
        withSonarQubeEnv(credentialsId: 'sonar_token', installationName: 'CoffeeingSonar') {
          sh '''
            cd "${WORKSPACE}"/backend/coffeeing
            ./gradlew sonar
          '''
        }
      }
    }
    stage('build jar') {
      steps{
        sh '''
          cd "${WORKSPACE}"/backend/coffeeing
          ./gradlew bootjar
        '''
      }
    }
    stage('Dockerize'){
      steps{
        script {
          sh '''
            sudo docker rm -f CONTAINER_NAME || true
            cd './backend/coffeeing'
            sudo docker build -t IMAGE_NAME .
            sudo docker image prune -f
            sudo docker run --name CONTAINER_NAME -d --network host -e SPRING_PROFILES_ACTIVE=dev IMAGE_NAME
          '''
        }
      }
    }
  }
}
```

4. 2 Frontend CI / CD Script

```
pipeline {
    agent any
    stages {
        stage('Pull dev/fe git') {
            steps {
                echo 'Pulling git'
                git branch: 'REPLACE_BRANCH_NAME', credentialsId: 'accessToken', url: 'REPLACE_YOUR_SVM_URL'
            }
        }
        stage('Build FE'){
            // install node modules
            steps{
                dir('frontend/coffeeing') {
                    sh 'npm install'
                    sh 'npm run dbuild'
                }
            }
        }
        stage("Deploy build files"){
            steps{
                dir('frontend/coffeeing/build'){
                    sh 'sudo cp -a ./. /REPLACE_YOUR_DEPLOY_PATH'
                }
            }
        }
    }
}
```

4. 3 FastAPI CI / CD Script

```
pipeline {
  agent any

  stages {
    stage('Git Pull') {
      steps {
        git branch: 'REPLACE_BRANCH_NAME', credentialsId: 'accessToken', url: 'REPLACE_YOUR_SVM_URL'
      }
    }
    stage('Copy Property Files') {
      steps {
        sh 'cp /REPLACE_YOUR_ENV_PATH/.env /REPLACE_YOUR_JENKINS_WORKSPACE_PATH/model/coffeeing'
      }
    }
    stage('Dockerize'){
      steps{
        script {
          sh '''
            sudo docker rm -f CONTAINER_NAME || true
            cd './model/coffeeing'
            sudo docker build -t IMAGE_NAME .
            sudo docker image prune -f
            sudo docker run --name CONTAINER_NAME -d --network host IMAGE_NAME
          '''
        }
      }
    }
  }
}
```

다음의 값을 배포환경에 맞게 입력합니다.

REPLACE_YOUR_SVM_URL: 형상관리 서버 리모트 레포지토리 URL

REPLACE_BRANCH_NAME: 각 소스 코드가 위치한 브랜치명

CONTAINER_NAME : 사용할 컨테이너명

IMAGE_NAME : 빌드된 이미지명

REPLACE_YOUR_JENKINS_WORKSPACE_PATH : 설정한 Jenkins Workspace

---

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/*;

        upstream jenkins {
                keepalive 32;
                server 127.0.0.1:YOUR_JENKINS_PORT;
        }

        server {
                listen 443 ssl;
                server_name YOUR_SERVER_URL;

                location /api {
                        proxy_set_header HOST $host;
                        proxy_pass http://127.0.0.1:YOUR_API_SERVER_PORT;
                        proxy_redirect off;
                        proxy_set_header X-Forwarded-Proto $scheme;
                }
                location /dev {
                        proxy_set_header HOST $host;
                        proxy_pass http://127.0.0.1:YOUR_API_TEST_SERVER_PORT;
                        proxy_redirect off;
                        proxy_set_header X-Forwarded-Proto $scheme;
                }

                location /jenkins {
                        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-Forwared-Proto $scheme;
                        proxy_pass http://jenkins;
                        proxy_redirect default;
                        proxy_http_version 1.1;
                        proxy_set_header Connection "";
                }

                location /sonarqube {
                        proxy_set_header HOST $host;
                        proxy_pass http://127.0.0.1:YOUR_SONARQUBE_PORT;
                        proxy_redirect default;
                }

                location / {
                        root /home/ubuntu/static/;
                        index index.html index.htm;
                        try_files $uri $uri/ /index.html =404;
                }

                ssl_certificate YOUR_CERTIFICATE;
                ssl_certificate_key YOUR_SSL_CERTIFICATE_KEY;
                include YOUR_SSL_NGINX_CONFIG_PATH;
                ssl_dhparam YOUR_SSL_DHPARAM_PATH;

        }


        server {
                if ($host = YOUR_SERVER_URL) {
                        return 301 https://$host$request_uri;
                } # managed by Certbot


                server_name YOUR_SERVER_URL;
                listen 80;
        }

}
```

다음의 값을 배포환경에 맞게 입력합니다.

(해당 프로젝트의 경우 letsencrypt와 Certbot를 활용해 SSL 인증 및 갱신 하는 것을 기준으로 작성되었으므로,

상황에 맞게 변경해서 사용해야 합니다.)

YOUR_SERVER_URL : 호스트 서버 URL

YOUR_JENKINS_PORT: 젠킨스 실행 포트

YOUR_API_SERVER_PORT: Backend API 서버 포트

YOUR_API_TEST_SERVER_PORT: Backend API 테스트 서버 포트

YOUR_SONARQUBE_PORT: 소나큐브 실행 포트

YOUR_CERTIFICATE: fullchain.pem 경로

YOUR_SSL_CERTIFICATE_KEY : privkey.pem 경로

YOUR_SSL_NGINX_CONFIG_PATH: 포함시킬 ssl 관련 nginx설정 파일 경로

YOUR_SSL_DHPARAM_PATH : ssl-dhparams.pem 위치 경로