Gitlab 소스 클론 이후 빌드/배포 시 참고사항

1. 개발 환경

- 사용한 JVM
  - BellSoft LibericaJDK-17
- 웹서버/WAS
  - Spring boot 3.4.2
- IDE 버전
  - IntelliJ IDEA 2024.3.1.1

2. 빌드 시 사용되는 환경변수

- **4. 프로퍼티 참고**

3. 배포 시 특이사항

- 배포 환경
  - 운영 환경에서는 Jenkins에서 빌드 파일을 생성하여 Docker 컨테이너로 자동 배포함
  - 로컬에서 실행 시, 수동으로 BE 빌드 파일(JAR)을 생성해야 함
  - MySQL 서버는 로컬에 직접 설치함
  - redis 서버를 필요로 하므로 docker로 배포하는 것을 권장함
- 배포 스크립트

```
cd backend
.₩gradlew clean build
cd ..₩docker
docker-compose down
docker-compose up -d
```

- BE 배포 방법

```
# MySQL, redis를 직접 설치하거나 docker로 구동 중이라고 가정한다.
cd backend
.₩gradlew clean build
java –jar build/libs/ourdoc-0.0.1-SNAPSHOT.jar
```

■ FE 배포 방법

```
# BE가 정상 배포되었다고 가정한다.
cd frontend
npm install
npm install –g yarn
yarn dev
```

4. DB 접속 정보 등 프로젝트에 활용되는 주요 계정 및 프로퍼티가 정의된 파일 목록

■ DB 접속 정보

- 배포 환경에서는 AWS RDS를 사용하여 MySQL 서버를 구성함
- 로컬에서 배포 시 MySQL을 직접 설치해야 함(localhost:3306)

■ 주요 계정

- ERD 조회는 별첨한 파일 참고

■ 프로퍼티

- 경로
  - ₩₩S12P11A405₩backend₩src₩main₩resources
- 세부 파일
  - application.yml

```
server:
  servlet:
    context-path: /api

spring:
  messages:
    basename: messages
    encoding: UTF-8
  ai:
    openai:
```

```
      api-key: ${spring-ai.gpt.api-key}
      options:
        model: ${spring-ai.gpt.model}
  application:
    name: backend
  datasource:
    url: ${database.url}
    username: ${database.name}
    password: ${database.password}
    driver-class-name: com.mysql.cj.jdbc.Driver
  data:
    redis:
      host: ${redis.host}
      port: ${redis.port}
      timeout: ${redis.timeout}
      password: ${redis.password}

  jpa:
    hibernate:
      properties:
        hibernate:
          dialect: org.hibernate.dialect.MySQLDialect
#           jdbc.batch_size: 100
      ddl-auto: validate
    show-sql: true
  config:
    import:
      - classpath:secret.yml
      - classpath:ocr.yml
  servlet:
    multipart:
      max-request-size: 10MB
      max-file-size: 10MB

aws:
  s3:
    bucket-name: ${aws.bucket-name}
    region: ${aws.region}
    access-key: ${aws.access-key}
```

```
        secret-key: ${aws.secret-key}
        upload:
          access-url: ${aws.upload.access-url}

    book:
      api-url: ${book.nl-api-url}
      cert-key: ${book.nl-cert-key}
      read-url: ${book.read-url}

    logging:
      level:
        root: info
        com.ssafy.ourdoc: debug
      file:
        name: logs/ourdoc.log

    school:
      api-url: ${school.elesch-api-url}
      api-key: ${school.elesch-api-key}

    openvidu:
      url: ${openvidu.url}
      secret: ${openvidu.secret}

    prod:
      url: ${prod.url}
      QrUrl: ${prod.QrUrl}
      ChangeQrUrl: ${prod.ChangeQrUrl}
      excluded-paths: ${prod.excluded-paths}
```

- ocr.yml

```
    ocr:
      api-url: ${api-url}
      secret-key: ${secret-key}
```

- secret.yml
  - ✓ **RDS 정보, 외부 API key 등 민감사항이 있어 해당 값은 공개하지 않음**

```
database:
  name: # 데이터베이스 계정명
  password: # 데이터베이스 계정 비밀번호
  url: # 데이터베이스 URL
  # jdbc:mysql://xxxxxx


# 네이버 OCR
api-url:
"https://mgjetysupx.apigw.ntruss.com/custom/v1/37887/fbbc17ca091ca1b8d393ca74da
1a935fd0c488756f0a14f77fc8cf2b38e7c171/general"
secret-key: # OCR API 키

# AWS S3
aws:
  bucket-name: # S3 버킷 이름
  region: # S3 리전
  access-key: # S3 access key
  secret-key: # S3 secret key
  upload:
    access-url: # S3 access-url

# SpringAI
spring-ai:
  gpt:
    api-key: # gpt api key
    model: # gpt model

# 국립중앙도서관 OpenAPI
book:
  nl-api-url: # 국립중앙도서관 ISBN 서지정보 api url
  nl-cert-key: # 국립중앙도서관 api key
  read-url: # 독서로 api url

school:
  elesch-api-url: # 나이스 학교 기본정보 api url
  elesch-api-key: # 나이스 api key

# JWT
```

```
jwt:
  secret-key: # jwt secret-key
  access-expiration: # access token 만료 시간
  refresh-expiration: # refresh toekn 만료 시간

# Redis
redis:
  host: # Redis 서버 주소
  port: # Redis 기본 포트
  timeout: # 타임아웃 (밀리초)
  password: # Redis 비밀번호

# Openvidu
openvidu:
  url: # openvidu client url
  secret: # openvidu secret

prod:
  url: # 배포 서버 url
  QrUrl: # 학급 가입 qr url
  ChangeQrUrl: # 학급 변경 qr url
  excluded-paths: # interceptor에서 token 검증을 무시하는 url 목록
```