빌드 및 배포 방법

개발환경

ONU_BACKEND(Business Server)

- Gradle : 7.6.1
- InteliJ : 2022.3.1
- SpringBoot 2.7.11
- Spring Security 2.7.11
- Spring Data JPA 2.7.11
- OAuth2 : 2.7.11
- Swagger2 : 2.9.2
- Json Web Token : 0.9.1
- jdk : Azul zulu 1.8
- MySQL : 8.0.31
- lombok : 1.18.26
- Redis : 7.0.11

ONU_SEARCH(SearchServer)

- Gradle : 7.6.1
- InteliJ : 2022.3.1
- Spring
  - SpringBoot 2.7.11
  - Spring Security 2.7.11
  - Spring Data JPA 2.7.11
- Swagger2 : 2.9.2
- jdk : Azul zulu 1.8
- MySQL : 8.0.31
- lombok : 1.18.26

ONU_BIGDATA(BigData Server)

- Django: 3.2.13
- Python: 3.11
- Pandas: 2.0.1
- Numpy: 1.24.3
- Scikit-learn: 1.2.2

ONU_FRONTEND

코어

- React.js: 18.2.0
- Next.js: 13.3.1

- TypeScript: 5.0.4

상태관리

- Zustand: 4.3.7

서버통신

- React-query: 4.29.5
- Axios: 1.3.6

스타일

- Tailwindcss: 3.3.2
- styled-components: 5.3.10
- daisyUI: 2.51.6
- headlessUI: 1.7.14
- emotion: 11.10.6

기타

- react-cookie: 4.1.1
- react-calendar: 4.2.1
- dayjs: 1.11.7
- react-loader-spinner: 5.3.4
- sweetalert2: 11.7.3

SERVER

- AWS
  - EC2
    - 플랫폼: Ubuntu 20.04.5 LTS
- Docker : 23.0.5
- Nginx : 1.18.0
- Jenkins
- SSL

EC2 배포 환경 설정

Docker 설치

```
# 최신 패키지 리스트 업데이트
sudo apt update

# docker 다운로드를 위해 필요한 https 관련 패키지 설치
sudo apt install apt-transport-https ca-certificates curl software-properties-common

# docker repository 접근을 위한 GPG key 설정
curl -fsSL [https://download.docker.com/linux/ubuntu/gpg](https://download.docker.com/linux/ubuntu/gpg) | sudo apt-key add -

# docker repository 등록
sudo add-apt-repository "deb [arch=amd64] [https://download.docker.com/linux/ubuntu](https://download.docker.com/linux/ubuntu) focal s

# 등록한 docker repository까지 포함하여 최신 패키지 리스트 업데이트
sudo apt update

# docker 설치
sudo apt install docker-ce

# docker 실행 중임을 확인
sudo systemctl status docker
```

```
# 도커 버전 확인
docker -v
```

MySQL 설치

```
# docker 이미지 다운
docker pull mysql:8.0.31

# onu_backend MySQL
docker run -d --name mysql-container -p [외부 접속 포트번호]:3306 -e MYSQL_ROOT_PASSWORD=비밀번호 -e MYSQL_DATABASE=onu_db -v ~/mysqldata/:/

# onu_search MySQL
docker run -d --name mysql-search-container -p [외부 접속 포트번호]:3306 -e MYSQL_ROOT_PASSWORD=비밀번호 -e MYSQL_DATABASE=search_db -v ~/se
```

jenkins 설치

```
# dockere로 jenkins 설치
docker run -d --name jenkins -p [외부 접속 포트번호]:8080 -p 50000:50000 -v /home/deploy/jenkins_v:/var/jenkins_home -v /var/run/docker.so
```

Redis 설치

- Redis 설치(설치 시 비밀번호 생성)

```
# redis 이미지 pull
docker pull redis

# docker로 실행
docker run -d --name redis-container -p [외부 접속 포트번호]:6379 redis --requirepass 비밀번호
docker exec -it redis-container /bin/bash
# 시작
docker start redis-container
```

Nginx 설치 및 설정

```
//Nginx 설치
sudo apt-get install nginx
//설치 확인
sudo nginx -v
//Nginx 중지
sudo systemctl stop nginx

# conf 파일 생성
sudo vim /etc/nginx/nginx.conf
```

nginx.conf 파일

```
#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; # 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;

        # gzip_vary on;
        # gzip_proxied any;
        # gzip_comp_level 6;
        # gzip_buffers 16 8k;
        # gzip_http_version 1.1;
        # gzip_types text/plain text/css application/json application/javascript text/xml application/xml application/xml+rss text/jav

        ##
        # Virtual Host Configs
        ##

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

/etc/nginx/sites-available/onu.conf 파일 생성

```
# onu.conf
server {
        # 프론트 연결(포트 번호는 본인의 프론트 포트번호를 입력)
        location / {
                proxy_pass http://localhost:[외부 접속 포트번호];
        }

        # 백엔드 연결(포트 번호는 본인의 백엔드 포트번호를 입력)
        location /api {
                proxy_pass http://localhost:[외부 접속 포트번호]/api;
        }

        # 백엔드 서치 연결(포트 번호는 본인의 검색 백엔드 포트번호를 입력)
        location /search {
                proxy_pass http://localhost:[외부 접속 포트번호]/search;
        }

       # 백엔드 빅데이터 연결(포트 번호는 본인의 검색 빅데이터 포트번호를 입력)
      location /bigdata {
          proxy_pass http://localhost:[외부 접속 포트번호];  # Gunicorn이 실행 중인 호스트와 포트로 변경하세요
          proxy_set_header Host $host;
          proxy_set_header X-Real-IP $remote_addr;
          proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }
      # Swagger
        location ~ ^/(swagger|webjars|configuration|swagger-resources|v2|csrf) {
              proxy_pass http://localhost:[외부 접속 포트번호];
              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;
        }

    listen 443 ssl; # managed by Certbot
    # 도메인 이름을 써줘야함
```

```
    ssl_certificate /etc/letsencrypt/live/[도메인 주소:o-nu.com]//fullchain.pem; # managed by Certbot
    # 도메인 이름을 써줘야함
    ssl_certificate_key /etc/letsencrypt/live/[도메인 주소:o-nu.com]/privkey.pem; # managed by Certbot
    # include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    # ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot
}

server {
    # 도메인 이름을 입력
    if ($host = [도메인 주소:o-nu.com]) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

        listen 80;
        server_name [도메인 주소:o-nu.com];
    return 404; # managed by Certbot
}
```

파일 연동 및 테스트

```
# 파일명:onu
sudo In -s /etc/nginx/sites-available/onu.conf /etc/nginx/sites-enabled/onu.conf
sudo nginx -t
//Nginx 재시작
sudo systemctl restart nginx
//Nginx 상태 확인
sudo systemctl status nginx
```

SLL 인증키 발급

```
# Let's Encrypt 설치
sudo apt-get install letsencrypt
# 인증서 적용 및 .pem 키 발급
sudo letsencrypt certonly --standalone -d o-nu.com
# 발급 경로 확인
sudo cd /etc/letsencrypt/live/o-nu.com
```

ONU_BACKEND 배포

git clone

```
git clone -b backend https://lab.ssafy.com/s08-final/S08P31A703.git
```

backend 폴더로 이동

```
cd backend
```

application.yml 작성

```
#application.yml
#port
server:
  port: [onu_backend 포트번호]
  servlet:
    context-path: /
    encoding:
      charset: utf-8
      enabled: true
      force: true

#database
spring:
  jpa:
    generate-ddl : true
    hibernate:
      ddl-auto: update
```

```
    show-sql: true
    properties:
      hibernate:
        format_sql: true
  #swagger spring version over 2.6
  mvc:
    pathmatch:
      matching-strategy: ant_path_matcher
  profiles:
    include: secret

#logging
logging:
  level:
    com.ssafy.onu: debug
```

application-secret.yml 작성

```
#application-secret
spring:
  datasource:
    url: jdbc:mysql://[도메인 주소:o-nu.com]:[mysql-container 외부 접속 포트번호]/onu_db?characterEncoding=UTF-8&serverTimezone=Asia/Seoul
    username: 사용자명
    password: 비밀번호
    driver-class-name: com.mysql.cj.jdbc.Driver

  redis:
    host: [도메인 주소:o-nu.com]
    port: [redis 외부 접속 포트번호]
    password: 비밀번호

  security:
    oauth2:
      client:
        registration:
          kakao:
            client-id: 카카오 id
            client-secret: 카카오 비밀키
            redirect-uri: [로그인 리다이렉트 url 주소]
            authorization-grant-type: authorization_code
            client-authentication-method: POST
            client-name: Kakao
            scope:
              - profile_nickname

        provider:
          kakao:
            authorization-uri: https://kauth.kakao.com/oauth/authorize
            token-uri: https://kauth.kakao.com/oauth/token
            user-info-uri: https://kapi.kakao.com/v2/user/me
            user-name-attribute: id

app:
  auth:
    tokenSecret: 토큰 비밀키
    tokenExpirationMsec: 1000000
    refreshTokenExpiry: 1209600000
  oauth2:
    # After successfully authenticating with the OAuth2 Provider,
    # we'll be generating an auth token for the user and sending the token to the
    # redirectUri mentioned by the client in the /oauth2/authorize request.
    # We're not using cookies because they won't work well in mobile clients.
    authorizedRedirectUris:
      - [로그인 리다이렉트 url 주소]
      - [로그인 리다이렉트 url 주소]
#value
cool:
  key: 키
  secret: 비밀키
  from: 전송 할 전화번호
```

DockerFile 작성

- build.gradle이 있는 파일 위치에 DockerFile 생성

```
FROM openjdk:8-jre
COPY build/libs/*.jar app.jar
ENTRYPOINT ["java", "-jar", "app.jar"]
```

빌드

```
chmod +x ./gradlew
./gradlew build
```

도커 이미지 생성

```
sudo docker build --tag onu_backend .
```

도커 컨테이너 생성

```
docker run -d -p [외부에서 연결할 포트번호]:[onu_backend 포트번호] --name [도커 컨테이너 이름]  -v /etc/localtime:/etc/localtime:ro -e TZ=Asia/Se
```

Jenkins 무중단 자동 배포 시 Excute shell 작성

```
# Excute shell
cd backend
chmod +x ./gradlew
./gradlew build
docker build --tag onu_backend .
docker stop onu_backend
docker rm onu_backend
docker run -d -p [외부에서 연결할 포트번호]:[onu_backend 포트번호] --name onu_backend  -v /etc/localtime:/etc/localtime:ro -e TZ=Asia/Seoul o
```

ONU_FRONTEND 배포

git clone

```
git clone -b frontend https://lab.ssafy.com/s08-final/S08P31A703.git
```

backend 폴더로 이동

```
cd frontend
```

DockerFile 작성

- package.json이 있는 파일 위치에 DockerFile 생성

```
# base image
FROM node:18-alpine

# set working directory
WORKDIR /app

# copy package.json and package-lock.json
COPY package*.json ./

# install dependencies
RUN npm install

# copy source code
COPY . .

# build Next.js app
RUN npm run build

# expose port 3000
EXPOSE 3000

# start the app
CMD ["npm", "start"]
```

Jenkins 무중단 자동 배포 시 설정 및 Excute shell 작성

- jenkins에 nodeJS plugin 설치 및 NodeJS Tool 추가

```
# Excute shell
cd frontend
npm install
npm run build
docker build --tag onu_frontend .
docker stop onu_frontend
docker rm onu_frontend
docker run -d -p [외부에서 연결할 포트번호]:3000 --name onu_frontend  -v /etc/localtime:/etc/localtime:ro -e TZ=Asia/Seoul onu_frontend
```

ONU_BIGDATA 배포

git clone

```
git clone -b bigdata https://lab.ssafy.com/s08-final/S08P31A703.git
```

backend 폴더로 이동

```
cd bigdata
```

DockerFile 작성

- requirements.txt이 있는 파일 위치에 DockerFile 생성

```
# Base image
FROM python:3.11

# Set working directory
WORKDIR /onu

# Copy requirements.txt
COPY requirements.txt .

# Install dependencies including Gunicorn
RUN pip install --no-cache-dir -r requirements.txt gunicorn

# Copy application code
COPY . .

# Expose port
EXPOSE 8000

# Run the application with Gunicorn
CMD ["gunicorn", "--bind", "0.0.0.0:8000", "onu.wsgi:application"]
```

Jenkins 무중단 자동 배포 시 Excute shell 작성

```
# Excute shell
cd onu
docker build --tag onu_bigdata .
docker stop onu_bigdata
docker rm onu_bigdata
docker run -d -p [외부에서 연결할 포트번호]:8000 --name onu_bigdata -v /etc/localtime:/etc/localtime:ro -e TZ=Asia/Seoul onu_bigdata
```

ONU_SEARCH 배포

git clone

```
git clone -b search https://lab.ssafy.com/s08-final/S08P31A703.git
```

backend 폴더로 이동

```
cd search
```

application.yml 작성

```
#application.yml
#port
server:
  port: [onu_search 포트번호]
  servlet:
    context-path: /
    encoding:
      charset: utf-8
      enabled: true
      force: true

#database

spring:
  jpa:
    generate-ddl : true
    hibernate:
      ddl-auto: update
    show-sql: true
    properties:
      hibernate:
        format_sql: true
  #swagger spring version over 2.6
  mvc:
    pathmatch:
      matching-strategy: ant_path_matcher
  profiles:
    include: secret

#logging
logging:
  level:
    com.ssafy.onu: debug
```

application-secret.yml 작성

```
#application-secret
spring:
  datasource:
    url: jdbc:mysql://[도메인 주소:o-nu.com]:[mysql-search-container 외부 접속 포트번호]/search_db?characterEncoding=UTF-8&serverTimezone=As
    username: 사용자명
    password: 비밀번호
    driver-class-name: com.mysql.cj.jdbc.Driver
```

DockerFile 작성

- build.gradle이 있는 파일 위치에 DockerFile 생성

```
FROM openjdk:8-jre
COPY build/libs/*.jar app.jar
ENTRYPOINT ["java", "-jar", "app.jar"]
```

빌드

```
chmod +x ./gradlew
./gradlew build
```

도커 이미지 생성

```
docker build --tag onu_search .
```

도커 컨테이너 생성

```
docker run -d -p [외부에서 연결할 포트번호]:[onu_search 포트번호] --name [도커 컨테이너 이름] -v /etc/localtime:/etc/localtime:ro -e TZ=Asia/Seou
```

Jenkins 무중단 자동 배포 시 Excute shell 작성

```
# Excute shell
cd search
chmod +x ./gradlew
./gradlew build
docker build --tag onu_search .
docker stop onu_search
docker rm onu_search
docker run -d -p [외부에서 연결할 포트번호]:[onu_search 포트번호] --name [도커 컨테이너 이름] -v /etc/localtime:/etc/localtime:ro -e TZ=Asia/Seou
```