

**E104 : 개츠비**

삼성SW청년아카데미 부울경캠퍼스 7기

특화프로젝트(7주: 2022.10.10 ~ 2022.11.21)

**포팅 매뉴얼**

담당 컨설턴트 : 김신일

김수진(팀장), 권민용, 배준식, 석민형, 윤호준, 이연의

**목차**

1. 프로젝트 기술 스택 -------------------------------------- 3p

2. 주요 환경 변수 ------------------------------------------- 4p

3. 도커 이미지 빌드 및 실행 ------------------------------- 7p

4. Jenkins 쉘 스크립트 -------------------------------------- 9p

5. Docker 파일 ----------------------------------------------- 10p

6. 배포 특이사항 ---------------------------------------------12p

7. 외부 서비스 ----------------------------------------------- 14p

**1. 프로젝트 기술 스택**

가. 이슈 관리: Jira

나. 형상 관리: Gitlab

다. 커뮤니케이션: Notion, Mattermost

라. 개발 환경

1) OS: Windows 10

2) IDE

가) IntelliJ 2021.3.2

나) Visual Studio Code 1.70.1

다) UI/UX: Figma

3) Database:

가) MySQL 8.0.31

나) Redis 7.0.5

다) Firebase 9.1.0 (외부)

4) Server: AWS EC2 Ubuntu 20.04 LTS

5) Dev-Ops

가) Docker 20.10.21

나) Jenkins 2.361.2

마. 상세 사용

1) Frontend

가) HTML5, CSS3, JavaScript(ES6)

나) React 17.0.2, Redux 4.2.0

다) Node.js 16.16.0

라) React-wordcloud 1.2.7

2) Backend

가) Spring boot 2.7.5

나) Open JDK 11

다) Gradle 7.5.1

라) Querydsl 5.0

마) Selenium 4.5.3

바) Jwt 0.11.5

**2. 주요 환경변수**

# db  
spring.datasource.url=[DB 주소]  
spring.datasource.username=**[DB 호스트명: gease]**  
spring.datasource.password=**[DB 비밀번호: g2b1s1l2]**

spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect spring.jpa.hibernate.ddl-auto=update   
spring.jpa.properties.hibernate.format\_sql=true

# 로깅

logging.level.org.hibernate.SQL=warn  
logging.level.org.springframework=warn  
logging.level.org.springframework.web=warn  
logging.level.org.springframework.security=warn

# jwt  
jwt.header=Authorization   
jwt.access-token-validity-in-seconds=86400

jwt.refresh-token-validity-in-seconds=604800

app.auth.token.secret-key=**[jwt 시크릿 키]**  
app.auth.token.refresh-cookie-key=refresh

# OAuth (kakao)  
spring.security.oauth2.client.provider.kakao.authorization-uri= https://kauth.kakao.com/oauth/authorize  
spring.security.oauth2.client.provider.kakao.token-uri= https://kauth.kakao.com/oauth/token  
spring.security.oauth2.client.provider.kakao.user-info-uri= https://kapi.kakao.com/v2/user/me  
spring.security.oauth2.client.provider.kakao.user-name-attribute= id  
spring.security.oauth2.client.registration.kakao.client-id=**[Kakao Client Id]**  
spring.security.oauth2.client.registration.kakao.client-secret=**[Kakao Secret Key]**  
spring.security.oauth2.client.registration.kakao.redirect-uri= {baseUrl}/oauth2/callback/kakao  
spring.security.oauth2.client.registration.kakao.scope= profile\_image  
spring.security.oauth2.client.registration.kakao.authorization-grant-type= authorization\_code  
spring.security.oauth2.client.registration.kakao.client-authentication-method= POST  
spring.security.oauth2.client.registration.kakao.client-name= Kakao  
  
# OAuth (Naver)  
spring.security.oauth2.client.provider.naver.authorization-uri= https://nid.naver.com/oauth2.0/authorize  
spring.security.oauth2.client.provider.naver.token-uri= https://nid.naver.com/oauth2.0/token  
spring.security.oauth2.client.provider.naver.user-info-uri= https://openapi.naver.com/v1/nid/me  
spring.security.oauth2.client.provider.naver.user-name-attribute= response  
spring.security.oauth2.client.registration.naver.client-id=**[Naver Client Id]**  
spring.security.oauth2.client.registration.naver.client-secret=**[Naver Secret Key]**  
spring.security.oauth2.client.registration.naver.redirect-uri= {baseUrl}/oauth2/callback/naver  
spring.security.oauth2.client.registration.naver.authorization-grant-type= authorization\_code  
  
# OAuth (github)  
spring.security.oauth2.client.registration.github.client-id=**[Github Client Id]**  
spring.security.oauth2.client.registration.github.client-secret=**[Github Secret Key]**  
spring.security.oauth2.client.registration.github.redirect-uri= {baseUrl}/oauth2/callback/github  
spring.security.oauth2.client.registration.github.scope= user  
  
# OAuth (google)  
spring.security.oauth2.client.registration.google.client-id=**[Google Client Id]**  
spring.security.oauth2.client.registration.google.client-secret=**[Google Secret Key]**  
spring.security.oauth2.client.registration.google.redirect-uri= {baseUrl}/oauth2/callback/{registrationId}  
spring.security.oauth2.client.registration.google.scope= profile, email

# s3  
cloud.aws.stack.auto=false   
cloud.aws.region.static=**[AWS region]**  
cloud.aws.credentials.access-key=**[발급받은 엑세스 키]**  
cloud.aws.credentials.secret-key=**[발급받은 시크릿 키]**  
cloud.aws.s3.bucket=**[버킷명]**  
logging.level.com.amazonaws.util.EC2MetadataUtils=error

# 서버 설정

server.servlet.context-path=/api   
server.error.include-stacktrace=never

# redis  
spring.redis.host=**[레디스 호스트 주소]**  
spring.redis.port=**[레디스 포트 번호]**  
spring.redis.password=**[레디스 비밀번호]**

# ssl  
security.require-ssl=true  
server.ssl.key-store=classpath:spring\_key.p12  
server.ssl.key-store-type=PKCS12  
server.ssl.key-store-password=**[ssl 인증서 비밀번호]**  
server.ssl.enabled=true

**3. 도커 이미지 빌드 및 실행**

**가) Docker 설치**

$ sudo apt-get remove docker docker-engine [docker.io](http://docker.io/) containerd runc

$ sudo apt-get update

$ sudo apt-get install apt-transport-https ca-certificates curl gnupg-agent software-properties-common

$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

$ sudo apt-key fingerprint 0EBFCD88

$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable"

$ sudo apt-get update

$ sudo apt-get install docker-ce docker-ce-cli containerd.io

$ sudo docker --version

**나) mysql 도커에 올리기**

$ sudo docker pull mysql

$ sudo docker images

$ sudo ufw allow 3306

$ sudo docker run -d --name mysql -e MYSQL\_ROOT\_PASSWORD=[패스워드] -p 3306:3306 mysql

$ sudo docker ps

**다) Jenkins 도커에 올리기**

$ sudo docker pull jenkins/jenkins:lts

$ sudo docker

$ sudo ufw allow

$ sudo docker run --name jenkins -d -p 8080:8080 -p 50000:50000 -v /home/jenkins:/var/jenkins\_home -v /var/run/docker.sock:/var/run/docker.sock -e TZ=Asia/Seoul -u root jenkins/jenkins:lts

$ sudo docker ps

$ sudo docker logs jenkins

**4. Jenkins 쉘 스크립트**

**가) backend**

$ cd backend

$ docker build -t backend .

$ docker ps -q --filter "name=backend" | grep -q . && docker stop backend && docker rm backend | true

$ docker run -p 8081:8080 -d -e TZ=Asia/Seoul --name=backend backend

$ docker rmi -f $(docker images -f "dangling=true" -q) || true

**나) frontend**

$ cd frontend

$ docker build -t frontend .

$ docker ps -q --filter "name=frontend" | grep -q . && docker stop frontend && docker rm frontend | true

$ docker run -d -p 80:80 -p 443:443 -v /home/ubuntu/certbot/conf:/etc/letsencrypt/ -v /home/ubuntu/certbot/www:/var/www/certbot --name frontend frontend

$ docker rmi -f $(docker images -f "dangling=true" -q) || true

**5. Docker 파일**

**가) backend**

FROM openjdk:11-jdk-slim as *builder*COPY gradlew .  
COPY gradle gradle  
COPY build.gradle .  
COPY settings.gradle .  
COPY src src  
COPY chrome chrome  
RUN chmod +x ./gradlew  
RUN ./gradlew bootJar  
  
  
FROM openjdk:11-jdk-slim  
COPY --from=*builder* build/libs/\*.jar app.jar  
ENTRYPOINT ["java","-jar","-Dspring.profiles.active=gcp","/app.jar"]  
EXPOSE 8081

**나) frontend**

# build stage  
FROM node:lts-alpine as build-stage  
WORKDIR /app  
COPY package\*.json ./  
RUN yarn install  
COPY . .  
RUN npm run build

# production stage   
FROM nginx:stable-alpine as production-stage   
COPY --from=build-stage /app/build /usr/share/nginx/html   
EXPOSE 80   
CMD ["nginx", "-g", "daemon off;"]

**6. 배포 특이사항**

**가) Spring boot에 SSL 적용**

1) Certbot container 생성 및 인증서 발급

|  |
| --- |
| sudo mkdir certbotcd certbotsudo mkdir conf www logs​sudo docker pull certbot/certbotsudo docker run -it --rm --name certbot -p 80:80 \-v "/home/ubuntu/certbot/conf:/etc/letsencrypt" \-v "/home/ubuntu/certbot/log:/var/log/letsencrypt" \-v "/home/ubuntu/certbot/www:/var/www/certbot" \certbot/certbot certonly |

2) SSL인증서를 spring boot에서 필요한 형식(PKCS12)로 변환

openssl pkcs12 -export -in fullchain.pem -inkey privkey.pem-out keystore.p12 -name tomcat -CAfile chain.pem -caname root

3) keystore p.12 파일을 /src/main/resources에 이동

**나) nginx SSL 설정**

1) /home/ubuntu/nginx/conf/default.conf

|  |
| --- |
| server {  listen 80;  server\_name k7e104.p.ssafy.io;  location / {  return 301 https://$host$request\_uri;  }    }  server {  listen 443 ssl;  server\_name k7e104.p.ssafy.io;  access\_log /var/log/nginx/access.log;  error\_log /var/log/nginx/error.log;    ssl\_certificate /etc/letsencrypt/live/k7e104.p.ssafy.io/fullchain.pem;  ssl\_certificate\_key /etc/letsencrypt/live/k7e104.p.ssafy.io/privkey.pem;  ssl\_protocols TLSv1 TLSv1.1 TLSv1.2 SSLv3;  ssl\_ciphers ALL;  location / {  root /usr/share/nginx/html;  index index.html index.htm  proxy\_redirect off;  charset utf-8;  try\_files $uri $uri/ /index.html;  proxy\_http\_version 1.1;  proxy\_set\_header Upgrade $http\_upgrade;  proxy\_set\_header Connection "upgrade";  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;  proxy\_set\_header X-Nginx-Proxy true;  }  } |

**7. 외부 서비스**

가) [카카오 로그인 기능](https://developers.kakao.com/docs/latest/ko/kakaologin/common)

나) [네이버 로그인 기능](https://developers.naver.com/products/login/api/api.md)

다) [깃허브 로그인 기능](https://docs.github.com/en/developers/apps/building-oauth-apps/authorizing-oauth-apps)

라) [구글 로그인 기능](https://developers.google.com/people/?hl=ko)

다) [AWS S3](https://aws.amazon.com/ko/s3/)