빌드 및 배포 방법

버전

BACKEND

- IntelliJ : 2022.3.1
- Gradle : 7.6
- Spring
  - Spring Boot : 2.7.7
  - Spring Security : 2.3.2
  - Spring Data JPA : 2.7.6
  - Spring Cloud Starter AWS : 2.2.1
- Swagger2 : 2.9.2
- Json Web Token : 0.9.1
- jdk : zulu 8
- MariaDB : 10.6.5

FRONTEND

- Node.js : v16.19.0
- npm : v8.19.3
- React.js : v18.2.0

IoT

- H/W
  - Raspberry Pi 4 Model B
  - Arduino Uno Rev 3
  - HC-06 Bluetooth Module
  - Joystick Module
- S/W
  - CUPS: 2.3.3
  - Node.js : v16.19.0
  - npm : v8.19.3
  - React.js : v18.2.0
  - Electron : v26.0.5
  - Electron Forge : v6.0.4

AI

- Pycharm 2022.3.2
- Flask : 2.2.2
- Tensorflow : 2.11.0
- Pytorch : 1.13.1+cu117

SERVER

- AWS
  - S3
  - CloudFront
  - EC2
    - 플랫폼: Ubuntu 20.04.5 LTS
- Docker : 23.0.1
- Nginx : 1.18.0(Ubuntu)
- Jenkins : 2.375.3
- Jupyter Notebook : 6.5.2

빌드&배포

본 프로젝트의 빌드와 배포는 Jenkins(Docker Container)와 Docker를 통해 이루어져 있으며, Git Lab과 Jenkins가 연동되어있음을 전제로 한다.

또한, Docker 명령어 수행을 위해 Jenkins Container 실행시

-v /home/jenkins:/var/jenkins_home 옵션과

-v /var/run/docker.sock:/var/run/docker.sock 옵션으로

호스트와 컨테이너를 볼륨 매핑 해야한다.

BACKEND

주의사항

gitlab에는 application.properties파일이 포함되어있지 않으므로 다음의 파일을 세팅해야 빌드가 가능하다.

```
spring.datasource.driver-class-name=org.mariadb.jdbc.Driver

spring.datasource.url=jdbc:mariadb://{DB의 주소}/{DB명}
spring.datasource.username=
spring.datasource.password=
springboot.jwt.secret=

#운영시에는 validate나 none 사용. 개발시에 create나 update사용.
spring.jpa.hibernate.ddl-auto=validate
spring.jpa.show-sql=true
```

```
spring.jpa.properties.hibernate.format_sql=true
spring.mvc.pathmatch.matching-strategy=ant_path_matcher

#security 세팅
spring.security.oauth2.client.registration.google.client-id=
spring.security.oauth2.client.registration.google.client-secret=
spring.security.oauth2.client.registration.google.redirect-uri=
spring.security.oauth2.client.registration.google.scope[]= email,profile
#Kakao
## registration
spring.security.oauth2.client.registration.kakao.client-id=
spring.security.oauth2.client.registration.kakao.client-secret=
spring.security.oauth2.client.registration.kakao.scope = profile_nickname, account_email
spring.security.oauth2.client.registration.kakao.client-name = Kakao
spring.security.oauth2.client.registration.kakao.authorization-grant-type = authorization_code
spring.security.oauth2.client.registration.kakao.redirect-uri =
spring.security.oauth2.client.registration.kakao.client-authentication-method = POST
## provider
spring.security.oauth2.client.provider.kakao.authorization-uri =
spring.security.oauth2.client.provider.kakao.token-uri =
spring.security.oauth2.client.provider.kakao.user-info-uri =
spring.security.oauth2.client.provider.kakao.user-name-attribute = id
#Naver
## registration
spring.security.oauth2.client.registration.naver.client-id=
spring.security.oauth2.client.registration.naver.client-secret=
spring.security.oauth2.client.registration.naver.redirect-uri=
spring.security.oauth2.client.registration.naver.authorization-grant-type=authorization_code
spring.security.oauth2.client.registration.naver.scope=name, email, mobile
spring.security.oauth2.client.registration.naver.client-name=Naver
## provider
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

# S3
cloud.aws.credentials.accessKey=
cloud.aws.credentials.secretKey=
cloud.aws.s3.bucket=
cloud.aws.region.static=ap-northeast-2
cloud.aws.stack.auto-=false


spring.servlet.multipart.maxFileSize=1MB
spring.servlet.multipart.maxRequestSize=1MB
```

ec2 서버의 home/jenkins/workspace/{Jenkins Freestyle project 명}/behind-back/build/libs

혹은 clone 받은 S08P12A404/behind-back/build/libs에

Dockerfile을 생성한다.

```
FROM openjdk:8-jdk

#JAR_FILE 변수 정의 -> 기본적으로 jar file이 2개이기 때문에 이름을 특정해야함
ARG JAR_FILE=behind-0.0.1-SNAPSHOT.jar

#JAR 파일 메인 디렉토리에 복사
COPY ${JAR_FILE} app.jar

#시스템 진입점 정의
ENTRYPOINT ["java","-jar","/app.jar"]
```

Jenkins Freestyle project의 자동 빌드, 배포를 위한 shell script

```
cd /var/jenkins_home/workspace/Behind-Back-Freestyle/behind-back
chmod +x gradlew
```

```
./gradlew build
cd build/libs
docker build -t behind-spring .
if [ $( docker ps -a | grep behind-spring | wc -l ) -gt 0 ]; then
echo "behind-spring container exists"
echo "delete container"
docker stop behind-spring
docker rm behind-spring
else
echo "behind-spring container does not exist"
fi
docker run -p 8080:8080 --name behind-spring -d behind-spring
```

FRONTEND

ec2 서버의 home/jenkins/workspace/{Jenkins Freestyle project 명}/behind-front

혹은 clone 받은 S08P12A404/behind-front에

Dockerfile과 default.conf를 생성한다.

Dockerfile

```
FROM node as builder

# 작업 폴더를 만들고 npm 설치
RUN mkdir /usr/src/app
WORKDIR /usr/src/app
ENV PATH /usr/src/app/node_modules/.bin:$PATH
COPY package.json /usr/src/app/package.json
RUN npm install --silent
RUN npm install react-scripts@2.1.3 -g --silent

# 소스를 작업폴더로 복사하고 빌드
COPY . /usr/src/app
RUN npm run build



FROM nginx:1.13.9-alpine
# nginx의 기본 설정을 삭제하고 앱에서 설정한 파일을 복사
RUN rm -rf /etc/nginx/conf.d
COPY default.conf /etc/nginx/conf.d/default.conf

# 위에서 생성한 앱의 빌드산출물을 nginx의 샘플 앱이 사용하던 폴더로 이동
COPY --from=builder /usr/src/app/build /usr/share/nginx/html

# 80포트 오픈하고 nginx 실행
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

default.conf

```
server {
  listen 80;
  location / {
    root   /usr/share/nginx/html;
    index  index.html;
    try_files $uri $uri/ /index.html;
  }
}
```

Jenkins Freestyle project의 자동 빌드, 배포를 위한 shell script

```
cd /var/jenkins_home/workspace/Behind-Front-Freestyle/behind-front
docker build -t behind-react .
if [ $( docker ps -a | grep behind-react | wc -l ) -gt 0 ]; then
  echo "behind-react container exists"
  echo "delete container"
  docker stop behind-react
  docker rm behind-react
else
  echo "behind-react container does not exist"
fi
docker run -p 3000:80 --name behind-react -d behind-react
```

데이터베이스

MariaDB

Spring Data JPA를 사용하였으므로, spring.jpa.hibernate.ddl-auto=create 사용 시 자동으로 데이터베이스가 생성된다.

ec2 서버에서 maria db를 설치하고 사용하기 위해서는 아래의 방법을 따른다.

```
- mariadb 설치하기
  - 먼저 timezone을 확인하고 변경한다.
    - 확인 : -date
    - 변경 : sudo timedatectl set-timezone 'Asia/Seoul'

  - 설치하기
    - 패키지 목록 최신화 : sudo apt update
    - 서버 설치 : sudo apt install mariadb-server
    - 마리아 db가 확실히 실행되는지 다시 체크 : sudo systemctl start mariadb.service  ==(동일)  sudo service mariadb start

  - 보안 설정
    - sudo mysql_secure_installation
    - root 사용자의 password는 설정하지 않기 때문에
      enter current password for root : 에서는 enter
      set root password? 에서 n을 입력해준다.
    - 익명 유저, test 데이터베이스를 없애고, root 원격 로그인을 막기 위해 Y를 입력해준다.

  - 새 유저 설정
    # 중요
    - MariaDB 10.3부터는 루트유저는 unix_socket을 사용하기 때문에 비밀번호를 사용하지 않는다. 문제가 생기는 것을 막기 위해 건드리지 말도록 하자.
    - sudo mysql로 root유저 접속이 가능하다.
    - 유저 생성 및 권한 주기
      ```
      USE mysql;
      //새로 유저를 만든다. 내부 접속용이면 localhost를, 외부에서 접근을 모두 허용해주려면 %를 입력해준다.
      CREATE USER '유저이름'@'접속ip' IDENTIIFED BY '비밀번호';
      //유저에게 권한을 부여해준다.
      GRANT ALL PRIVILEGES ON 데이터베이스이름.* TO '유저이름'@'접속ip';
      //설정을 저장한다.
      FLUSH PRIVILEGES;
      ```
    - unix_socket 사용을 원하지 않는다면 plugin을 공란으로 써주면 된다.
      ```
      USE mysql;
      UPDATE user SET plugin='' where user='root';
      SET password = password('비밀번호');
      FLUSH PRIVILEGES;
      ```
    - 유저 정보 변경은 update문을 이용하면 된다. flush privileges만 잊지말자
      ex) update user set password = password('새비밀번호') where user = '아이디' and host = '호스트 ip';
      flush privileges;

  - 외부에서 접근을 원한다면
```

```
    - sudo vi /etc/mysql/mariadb.conf.d/50-server.cnf
    - sudo vim /etc/mysql/mariadb.conf.d/50-server.cnf (vim 설치시)
    - 주석처리되어있는 port = 3306을 주석 해제해준다.
    - bind-address를 0.0.0.0으로 변경해준다.
```

IoT

라즈베리파이 패키지 리스트를 최신화하고 업데이트한다.

```
$ sudo apt-get update
$ sudo apt-get upgrade -y
```

RPi-Arduino 블루투스 통신 환경을 구축한다.

```
# 블루투스 패키지 설치
$ sudo apt-get install bluetooth blueman bluez
$ sudo apt-get install pi-bluetooth

# 블루투스 페어링
$ sudo bluetoothctl

power on
scan on

agent on
default-agent

pair <MAC 주소>

exit
```

라즈베리파이 인쇄 환경을 구축한다.

```
# CUPS 패키지 설치
$ sudo apt-get install cups
$ sudo apt-get install libcups2-dev

# 사용 권한 부여
$ sudo usermod -a -G lpadmin <username>
```

백그라운드 앱(app.py)에 사용된 Python 모듈, 패키지를 설치한다.

```
# 파이썬 블루투스 모듈 설치
$ sudo pip3 install pybluez

# CUPS 사용을 위한 Python bindings
$ sudo pip3 install pycups

# Python 키보드 제어 모듈
$ sudo pip3 install pynput

# GPIO 제어 라이브러리
$ wget https://github.com/joan2937/pigpio/archive/master.zip
$ unzip master.zip
$ cd pigpio-master
$ make
$ sudo make install

# Python parsing 패키지
$ sudo pip3 install parse
```

프로그램 실행을 위해 2가지 파일을 정해진 순서대로 실행한다.

```
# Background App - 물리적 제어
$ python3 app.py

# GUI App
cd behind  # 디렉터리 이동
./behind-embedded-front  # 실행
```