빌드 및 배포

프로젝트 기간 : 2022.07.11 ~ 08.19

삼성SW청년아카데미 서울캠퍼스 7기

민경욱 이주영 김찬영 정건우 이지나 채윤선

1. 기술스택

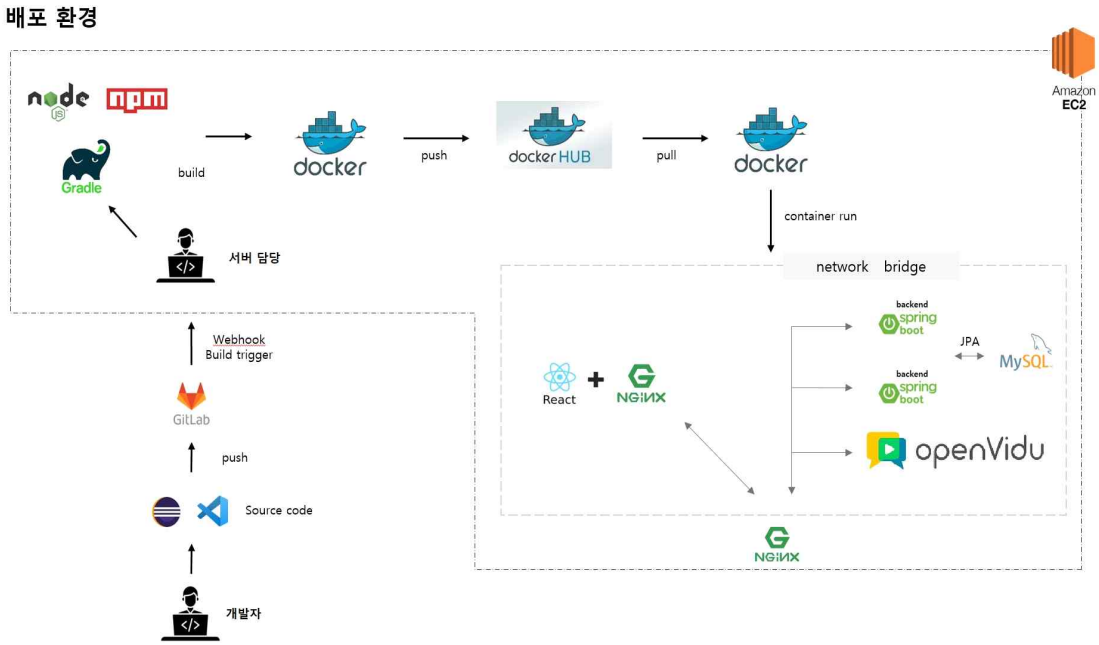
구분	기술스택	상세내용	버전
공통	형상관리	Gitlab	-
	이슈관리	Jira	-
	커뮤니케이션	Mattermost, Notion	-
BackEnd	DB	MySQL	5.7
		JPA	-
	Java	Zulu	8.33.0.1
	Spring	Spring	5.3.6
		Spring Boot	2.4.5
		Spring Security	-
	IDE	STS	3.9.14
	클라우드 스토리지	AWS S3	-
	Build	Gradle	7.3.2
	API Docs	Swagger2	3.0.0
FrontEnd	React	React	18.2.0
		Redux	8.0.2
		Redux-toolkit	1.8.3
	WebRTC	openvidu-browser	2.22.0
	WebSocket	stomp.js	6.1.2
	mui-material		5.9.0
	axios		0.14.0
	react-canvas-draw		1.2.0
	three		0.142
	tensorflow-model/facemesh		0.0.5
	tensorflow/tfjs		3.19.0
	react-xarrows		2.0.2
	react-countdown-circle-timer		3.0.9
	IDE	Visual Studio Code	1.63.2
Server	서버	AWS EC2	-
	플랫폼	Ubuntu	20.04.3 LTS
	배포	Docker	20.10.17

2. 상세내용

□ 개요

아래 그림은 Avatime 서비스의 배포 환경 및 CI/CD 배포 흐름도입니다. 팀원들이 각자 작성한 프로젝트를 GitLab에 push 하면, 서버 담당자가 FrontEnd, BackEnd, OpenVidu App을 빌드하게 됩니다.

<CI/CD 배포환경>



각 프로젝트를 빌드 한 후에는 Docker 이미지를 만들고 이를 Docker Hub 에 push 한 후, 이로부터 서비스에 필요한 이미지를 받아와 컨테이너로 띄웁니다.

서버의 경우 SSAFY에서 지원받은 AWS EC2 싱글 인스턴스로 인프라를 구축하였습니다. 이때, 추후 서비스화를 위해 Nginx는 리버스 프록시 서버로 설정하였습니다. 를 이용하여 8080포트를 Backend 서버로 설정하여 Load Balancing이 가능하도록 구축하였습니다.

□ FrontEnd

◦ Docker 이미지 생성을 위한 Dockerfile (해당 파일은 프로젝트 내에 이미 작성되어 있습니다)

```
# Dockerfile
# nginx 이미지를 사용합니다. 뒤에 tag가 없으면 latest 를 사용합니다.
FROM nginx
# root 에 app 폴더를 생성
RUN mkdir /app
# work dir 고정
WORKDIR /app
# work dir 에 build 폴더 생성 /app/build
RUN mkdir ./build
# host pc의 현재경로의 build 폴더를 workdir 의 build 폴더로 복사
ADD ./build ./build
# nginx 의 default.conf 를 삭제
RUN rm /etc/nginx/conf.d/default.conf
# host pc 의 nginx.conf 를 아래 경로에 복사
COPY ./nginx.conf /etc/nginx/conf.d
# 80 포트 오픈
EXPOSE 80
# container 실행 시 자동으로 실행할 command. nginx 시작함
CMD ["nginx", "-g", "daemon off;"]
```

◦ docker 이미지 생성

```
docker build -t livemose/avatime:front-release .
```

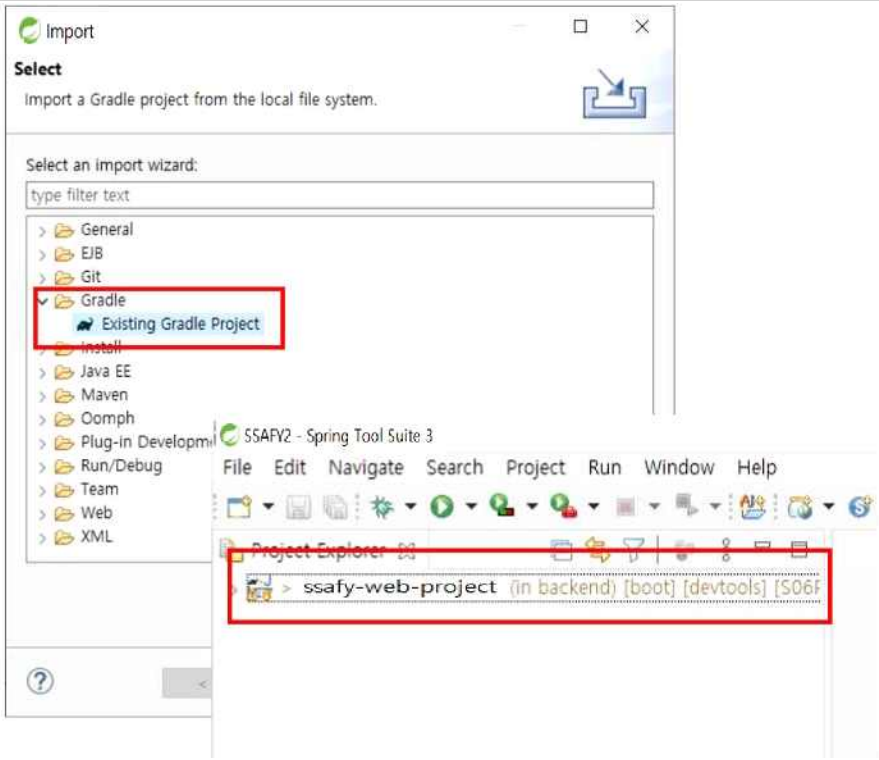
◦ 이미지 컨테이너 실행

```
docker container run --name front -d -p 3000:80 livemose/avatime:front-release
```

□ BackEnd

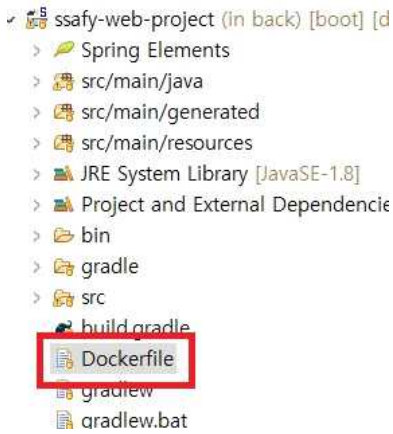
◦ Eclipse에서 backend 폴더를 Gradle로 import 합니다.

<import Gradle Project>



◦ Docker 이미지 생성을 위한 Dockerfile 작성 (해당 내용은 프로젝트 내에 이미 작성 되어 있습니다)

```
FROM openjdk:8-jdk-alpine
EXPOSE 8080
ARG JAR_FILE=build/libs/*.jar
COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java","-jar","/app.jar"]
ARG DEBIAN_FRONTEND=noninteractive
ENV TZ=Asia/Seoul
```



◦ Gradle 빌드

```
gradlew clean build
```

◦ docker 이미지 생성

```
docker build --build-arg JAR_FILE=build/libs/*.jar -t livemose/avatime:back-release .
```

◦ mysql 컨테이너 실행

```
docker run -p 3306:3306 –e MYSQL_ROOT_PASSWORD=webrtcavatime77!
-e TZ=Asia/eoul –d mysql:latest
```

◦ 이미지 컨테이너 실행

```
docker container run -d -p 8080:8080 livemose/avatime:back-release
```

◦ 서버 컨테이너 로그 확인

```
docker logs <containerID>
```

□ OpenVidu

◦ 설치

```
curl <https://s3-eu-west-1.amazonaws.com/aws.openvidu.io/install_openvidu_latest.sh>
| bash
```

3. 특이사항

□ 개요

Avatime 서비스는 Docker 이미지 컨테이너를 기반으로 서비스를 배포하고 있습니다. 서비스에 문제가 발생 시, 아래 명령어를 확인하여 상태를 확인 할 수 있습니다. BackEnd, FrontEnd, OpenVidu, Mysql 프로젝트의 상태를 확인하기 위해선 각 컨테이너의 로그를 확인하는 명령어를 사용하여 log 확인이 가능합니다.

□ Nginx

◦ 상태 확인

```
sudo service nginx status
```

◦ 재실행

```
sudo service nginx restart
```

□ Docker

◦ 컨테이너 확인

```
sudo docker ps -a
```

◦ 서버 컨테이너 로그 확인

```
docker logs <containerID>
```

◦ 컨테이너 재실행

```
sudo docker restart <containerID>
```

◦ 컨테이너 삭제

```
sudo docker rm <containerID>
```

◦ 이미지 삭제

```
sudo docker rmi <imageID>
```

4. 프로퍼티 정의

□ MySQL

◦ MySQL Docker 컨테이너에서 DB 스키마를 생성해두면 SpringBoot 구동 시 자동으로 Table 생성 및 Dump Data가 삽입됩니다.

◦ Spring application.properties DB 관련 설정

spring.jpa.hibernate.naming.implicit-strategy=org.springframework.boot.orm.jpa.hibernate.SpringImplicitNamingStrategy
spring.jpa.hibernate.naming.physical-strategy=org.springframework.boot.orm.jpa.hibernate.SpringPhysicalNamingStrategy
spring.jpa.hibernate.ddl-auto=update
spring.jpa.generate-ddl=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL57Dialect
spring.data.web.pageable.one-indexed-parameters=true
spring.datasource.url=jdbc:mysql://i7a309.p.ssafy.io:3306/ssafy_web_db?useUnicode=true&characterEncoding=utf8&serverTimezone=Asia/Seoul&zeroDateTimeBehavior=convertToNull&rewriteBatchedStatements=true
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.hikari.username=root
spring.datasource.hikari.password=webrtcavatime77!

◦ 계정 생성

```
create user 'root'@'%' identified by 'webrtcavatime77!';
grant all privileges on *.* to 'root'@'%' with grant option; flush privileges;
```

◦ 스키마 생성

```
create database if not exists ssafy_web_db collate utf8mb4_general_ci;
```

□ OpenVidu

◦ 환경설정

```
$ nano .env

# OpenVidu configuration
# ----------------------
# 도메인 또는 퍼블릭IP 주소
DOMAIN_OR_PUBLIC_IP=i5a608.p.ssafy.io

# 오픈비두 서버와 통신을 위한 시크릿
OPENVIDU_SECRET=HOMEDONG

# Certificate type
CERTIFICATE_TYPE=letsencrypt

# 인증서 타입이 letsencrypt일 경우 이메일 설정
LETSENCRYPT_EMAIL=user@example.com

# HTTP port
HTTP_PORT=4442
```

```
# HTTPS port(해당 포트를 통해 오픈비두 서버와 연결)
HTTPS_PORT=4443

# 오픈 비두 실행
$ ./openvidu start
```

□ Nginx

◦ 환경설정 - etc/nginx/sites-available/test.conf

```
server {

        location /{
                proxy_pass http://i7a309.p.ssafy.io:3000;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For
$proxy_add_x_forwarded_for;
                proxy_set_header Host $http_host;

        }

        location /api {
                proxy_pass http://i7a309.p.ssafy.io:8080/api;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For
$proxy_add_x_forwarded_for;
                proxy_set_header Host $http_host;

        }
        location /ws{
                proxy_pass http://i7a309.p.ssafy.io:8080/ws;
```

```
                    proxy_http_version 1.1;
                    proxy_set_header Upgrade $http_upgrade;
                    proxy_set_header Connection "upgrade";
                    proxy_set_header Host $host;
}

    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/i7a309.p.ssafy.io/fullchain.pem;
# managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/i7a309.p.ssafy.io/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 {
    if ($host = i7a309.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

        listen 80;
        server_name i7a309.p.ssafy.io;
    return 404; # managed by Certbot
}
```