싸브란스병원

1:1 건강검진 파트너 깐부

삼성 청년 SW 아카데미 대전캠퍼스 7기

자율 프로젝트 (2022.10.11. ~ 2022.11.21)

포팅 매뉴얼

전승준(팀장), 송다경(부팀장), 박찬, 이한기, 임웅균, 장정훈

1. 프로젝트 기술 스택

1. 이슈 관리 : Jira
2. 형상 관리 : GitLab
3. 커뮤니케이션 : Mattermost, Notion, Discord
4. 개발 환경
    1) OS : Window 10, linux(Ubuntu 18.04)
    2) IDE
        A. IntelliJ 2022.1.3, Visual Studio Code 1.69.X
        B. UI/UX : Figma
    3) Database : MySQL 8.0.29 CE
    4) Server : AWS EC2 (MobaXterm)
        A. Ubuntu 20.04.4 LTS
        B. Docker 20.10.18
        C. Jenkins 2.361.1
        D. Nginx 1.22.0
5. 상세 내용
    1) Backend
        A. Java Open-JDK zulu 17
        B. SpringBoot Gradle 2.7.2
            A. Spring Data JPA
            B. Swagger 2.9.2
        C. Fast API
            A. Azure
            B. OpenCV
    2) Frontend
        A. Vue2
            A. vuex
            B. SockJS
            C. Roslib
            D. Tensorflow-lite
    3) IoT
        A. SW
            A. ROS melodic
            B. Gazebo
            C. ROS Bridge
        B. HW

A. TurtleBot3
B. OpenCR
C. JetsonNano
D. LiDAR

2. 빌드 시 사용되는 환경 변수

1. ec2-back - env.properties 파일 추가

```
USER_NAME = root
PASSWORD = b309iot!
COOLSMS_API_KEY = {coolsms api key}
COOLSMS_SECRET = {coolsms secret key}
COOLSMS_PHONE = {coolsms에서 인증받은 번호}
BACKEND_ADDRESS = k7b309.p.ssafy.io:8081
```

2. gganbu-front - .env 파일 추가

```
VUE_APP_WEATHER_API_KEY={인증받은 weather api key}
```

3. ec2-front - .env 파일 추가

```
VUE_APP_API_SERVER="https://k7b309.p.ssafy.io/api”
VUE_APP_WEATHER_API_KEY={인증받은 weather api key}
```

3. 배포

1. 관리자 계정 활성화

```
su passwd root
```

2. 사전 패키지 업데이트

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

3. 도커 관련 패키지 설치

```
sudo apt-get install -y ca-certificates \
    curl \
    software-properties-common \
    apt-transport-https \
    gnupg \
    lsb-release
```

4. gpg key 다운로드

```
sudo mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg |
sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
```

```
echo \
    "deb [arch=$(dpkg --print-architecture)
signed-by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/ubuntu \
    $(lsb_release -cs) stable" | sudo tee
/etc/apt/sources.list.d/docker.list > /dev/null
```

5. Docker 설치

```
sudo apt install -y docker-ce docker-ce-cli containerd.io
docker-compose
```

6. 관리자가 아닌 일반계정도 도커를 실행할 수 있도록 설정

```
# docker permission
$ sudo groupadd docker
$ sudo usermod -aG docker $USER

# docker-compose permission
$ sudo chmod 666 /var/run/docker.sock
```

7. 젠킨스 컨테이너 실행을 위한 도커컴포즈 작성

~/docker-compose.yml

```
version: '3'

services:
    jenkins:
        image: jenkins/jenkins:lts
        container_name: jenkins
        volumes:
            - /var/run/docker.sock:/var/run/docker.sock
            - /jenkins:/var/jenkins_home
            - /home/ubuntu:/home/ubuntu
        ports:
```

```
            - "9090:8080"
        privileged: true
        user: root
```

8. 젠킨스 컨테이너 내부에 도커 설치

```
docker exec -it jenkins bash
```

다음 명령어로 접속 후 위에 나와있는 도커설치법을 따라 설치해준다.

9. 프론트 컨테이너 내부에서 사용할 letsencrypt파일 설정

```
sudo 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
```

4. 배포 과정

1. 사전에 작성한 docker-compose파일을 실행하여 젠킨스 띄우기

```
sudo docker-compose up -d
```

2. http://{서버주소}:9090/으로 들어가 젠킨스 설치 및 환경세팅팅 진행
   1) 젠킨스 비밀번호는 docker logs jenkins로 확인할 수 있음
   2) 젠킨스 필수 플러그인들 설치 후 freestyle 프로젝트를 gganbu로 생성
   3) 해당 프로젝트에 깃랩 연결 후 해당 프로젝트를 한번 빌드해준다.
   4) 위 과정을 거치면 내 서버 /jenkins/workspace/에 gganbu폴더가 생겼을것이다.
3. DB로 사용할 mysql 컨테이너 생성 및 세팅팅
   1) 서버에서 해당 명령어로 mysql 컨테이너 생성

```
docker run --name gganbu_DB -e
MYSQL_ROOT_PASSWORD=ssafy -d mysql
```

   2) 서버의 덤프파일을 DB에 넣어주기위해 해당 명령어를 실행

```
docker cp {덤프파일이 있는 위치} gganbu_DB:/home
```

   3) gganbu_DB 내부로 들어가기

```
docker exec -it gganbu_DB bash
```

4) mysql안에 덤프파일 임포트시키기

```
mysql -hlocalhost -uroot -pssafy < /home/{덤프파일명}
```

4. 젠킨스 빌드 후 생긴 ec2-back/gganbu폴더 내부에 도커파일, .env파일 넣어주기

1) ec2-back/gganbu안에 들어갈 파일목록

A. Dockerfile

```
FROM openjdk:17-jdk-alpine AS builder

COPY gradlew .
COPY gradle gradle
COPY build.gradle .
COPY settings.gradle .
COPY src src

RUN chmod +x /gradlew

RUN ./gradlew bootJar

FROM openjdk:17-jdk-alpine

COPY --from=builder build/libs/gganbu-0.0.1-SNAPSHOT.jar app.jar

ENTRYPOINT ["java","-jar","/app.jar"]
```

B. env.properties

```
USER_NAME = root
PASSWORD = b309iot!
COOLSMS_API_KEY = {coolsms api key}
COOLSMS_SECRET = {coolsms secret key}
COOLSMS_PHONE = {coolsms에서 인증받은 번호}
BACKEND_ADDRESS = k7b309.p.ssafy.io:8081
```

2) ec2-front안에 들어갈 파일목록

A. Dockerfile

```
FROM node:16.15.0 as build-stage
WORKDIR /var/jenkins_home/workspace/gganbu/ec2-front
COPY package*.json ./
RUN npm install
COPY . .
```

```
RUN npm run build
FROM nginx:stable-alpine as production-stage

RUN mkdir /app
WORKDIR /app
RUN mkdir ./dist

COPY --from=build-stage /var/jenkins_home/workspace/gganbu/ec2-front/dist
./dist
COPY ./nginx.conf /etc/nginx/conf.d

EXPOSE 80
CMD ["nginx", "-g","daemon off;"]
```

B. .env

```
VUE_APP_API_SERVER="https://k7b309.p.ssafy.io/api"
```

C. deploy_conf/nginx.conf

```
server {
    listen 3000;
    location / {
    root    /app/dist;
    index   index.html;
    try_files $uri $uri/ /index.html;
}
}
```

3) 젠킨스에 접속해서 gganbu의 빌드드설정을 마저 해준다.

4) Build Steps에 Excute shell 추가

```
docker image prune -a --force
mkdir -p /var/jenkins_home/images_tar
cd /var/jenkins_home/workspace/gganbu/ec2-front/
docker build -t vue .
docker save vue > /var/jenkins_home/images_tar/vue.tar

cd /var/jenkins_home/workspace/gganbu/ec2-back/gganbu
docker build -t springbackend .
docker save springbackend >
/var/jenkins_home/images_tar/springbackend.tar

ls /var/jenkins_home/images_tar
```

```
docker load < /var/jenkins_home/images_tar/vue.tar
docker load < /var/jenkins_home/images_tar/springbackend.tar

if (docker ps | grep "vue"); then docker stop vue; fi
if (docker ps | grep "springbackend"); then docker stop
springbackend; fi

docker run -it -d --rm -p 3000:3000 --name vue vue
echo "Run Front"
docker run -it -d --rm -p 8081:8081 -v
/tmp/gganbu:/tmp/gganbu --name springbackend springbackend
echo "Run Back"
```

5. 이후 재빌드하게되면 https://{서버주소}로 접속 가능하다.

5. Nginx 설정과 ssl 인증서 발급 및 적용

1. Nginx 다운로드

```
# nginx 다운
sudo apt-get install nginx
# 설치 확인 및 버전 확인
nginx -v
```

2. /etc/nginx/sites-available 로 이동 후 설정파일 생성

```
server{

        location / {
                proxy_pass http://localhost:3000;
        }

        location /api {
                proxy_pass http://localhost:8081/api;
        }

        listen 443 ssl;
        ssl_certificate
/etc/letsencrypt/live/k7b309.p.ssafy.io/fullchain.pem;
        ssl_certificate_key
```

```
/etc/letsencrypt/live/k7b309.p.ssafy.io/privkey.pem;
}

server{
        if ($host = k7b309.p.ssafy.io) {
                return 301
https://k7b309.p.ssafy.io$request_uri;
        }

        listen 80;
        server_name k7b309.p.ssafy.io;
        return 404;
}
```

3. 다음 명령어 차례로 실행

```
sudo ln -s /etc/nginx/sites-available/[파일명]
/etc/nginx/sites-enabled/[파일명]
# 다음 명령어에서 successful이 뜨면 nginx를 실행할 수 있다.
sudo nginx -t sudo systemctl restart nginx
```