포팅메뉴얼

jenkins 컨테이너 실행 (docker-compose.yml)

```
version: '3.7'
services:
 jenkins:
  build:
   context: .
  container_name: jenkins
  user: root
  privileged: true
  ports:
   - 9090:8080
   - 50000:50000
  volumes:
   - ./jenkins_home:/var/jenkins_home
   - /var/run/docker.sock:/var/run/docker.sock
```

mysql 컨테이너 실행

mysql 계정 생성 및 권한 설정

jenkins gitlab과 연결

프로젝트 git과 연동 후 백, 프론트 각각 받아온다.

백엔드프로젝트 (foodtruck)과 프론트엔드 프로젝트 (frontend)에 각각 도커파일을 생성한다.

백엔드 도커파일

```
FROM openjdk:8-jdk-alpine AS build
COPY gradlew .
COPY gradle gradle
COPY build.gradle .
COPY settings.gradle .
COPY src src
RUN chmod +x gradlew
RUN ["./gradlew", "bootJar"]
FROM openjdk:8-jdk-alpine
COPY --from=build build/libs/*.jar app.jar
EXPOSE 8085
ENTRYPOINT ["java", "-jar", "/app.jar"]
```

프론트엔드 도커파일

```
# build stage

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


#production stage

FROM nginx:stable-alpine as production-stage
COPY --from=build-stage /app/dist /usr/share/nginx/html
COPY ./nginx.conf /etc/nginx/nginx.conf
EXPOSE 3000
CMD ["nginx", "-g", "daemon off;"]
```

프론트엔드 nginx.conf 파일

```
user  nginx;
worker_processes  1;
error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;
events {
worker_connections  1024;
}
http {
include       /etc/nginx/mime.types;
default_type  application/octet-stream;
log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
'$status $body_bytes_sent "$http_referer" '
'"$http_user_agent" "$http_x_forwarded_for"';
access_log  /var/log/nginx/access.log  main;
sendfile        on;
keepalive_timeout  65;
server {
listen       3000;
server_name  localhost;
location / {
root   /usr/share/nginx/html;
index  index.html;
try_files $uri $uri/ /index.html;
}
error_page   500 502 503 504  /50x.html;
location = /50x.html {
root   /usr/share/nginx/html;
}
}
}
```

프록시서버 도커파일

💡 Dockerfile(client)

```
#nginx 이미지를 사용
FROM nginx
#work dir
WORKDIR .
#work dir 에 build 폴더 생성 : /home/blog/build
RUN mkdir ./build
#host pc 의 nginx.conf 를 복사
COPY ./nginx.conf /etc/nginx/nginx.conf
#80 포트 오픈
EXPOSE 80 443
#container 실행 시 자동으로 실행할 command. nginx 시작
CMD ["nginx", "-g", "daemon off;"]
```

프록시서버 nginx.conf 파일

```
user nginx;
worker_processes auto;

error_log       /var/log/nginx/error.log warn;
pid             /var/run/nginx.pid;

events {
worker_connections 1024;
}

http {
client_max_body_size 100M;
include         /etc/nginx/mime.types;
default_type    application/octet-stream;
log_format main '$remote_addr - $remote_user [$time_local] "$request" '
'$status $body_bytes_sent "$http_referer" '
'"$http_user_agent" "$http_x_forwarded_for"';
access_log      /var/log/nginx/access.log main;
sendfile on;
keepalive_timeout 65;

    upstream docker-back {
            server back:8080;
    }

    upstream docker-front{
            server front:3000;
    }

     server {
     listen 80;
     server_name k7b206.p.ssafy.io;
     server_tokens off;

            location /.well-known/acme-challenge/ {
                    root /var/www/certbot;
            }

            location / {
                    return 301 https://$host$request_uri;
            }
    }
```

```
server {
listen 443 ssl;
server_name k7b206.p.ssafy.io;
server_tokens off;

    ssl_certificate /etc/letsencrypt/live/k7b206.p.ssafy.io/fullchain.pem;
    ssl_certificate_key
/etc/letsencrypt/live/k7b206.p.ssafy.io/privkey.pem;
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

    location /api {
        proxy_pass      <http://docker-back/api>;
        proxy_redirect  off;
        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-Host $server_name;
    }

    location / {
        proxy_pass      <http://docker-front/>;
        proxy_redirect  off;
        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-Host $server_name;
    }
  }

}
```

Jenkins excute shell

```
docker ps -a -q --filter "name=foodtruck-front-1" | grep -q . && docker stop foodtruck-front-1 && docker rm foodtruck-front-1 || true
docker ps -a -q --filter "name=foodtruck-back-1" | grep -q . && docker stop foodtruck-back-1 && docker rm foodtruck-back-1 || true
docker ps -a -q --filter "name=foodtruck-nginx-1" | grep -q . && docker stop foodtruck-nginx-1 && docker rm foodtruck-nginx-1 || true

cd /var/jenkins_home/workspace/FoodTruck/foodtruck
docker build -t foodtruck_be .
cd /var/jenkins_home/workspace/FoodTruck/frontend
docker build -t foodtruck_fe .
cd /var/jenkins_home/workspace/FoodTruck/proxynginx
docker build -t nginxproxy .
cd /var/jenkins_home/workspace/FoodTruck
```

```
version: "3"

services:
nginxproxy:
image: nginxproxy:latest
ports:
- "80:80"
- "443:443"
restart: unless-stopped
volumes:
- ./data/certbot/conf:/etc/letsencrypt
- ./data/certbot/www:/var/www/certbot
command: "/bin/sh -c 'while :; do sleep 6h & wait $${!}; nginx -s reload;
done & nginx -g \"daemon off;\""


back:
    depends_on:
        - nginxproxy
    image: foodtruck_be:latest
    ports:
        - "8080:8080"
    restart: always
    volumes:
        - myapp:/home/app
front:
    depends_on:
        - nginxproxy
    image: foodtruck_fe:latest
    ports:
        - "3000:3000"
    restart: always
certbot:
    image: certbot/certbot
    restart: unless-stopped
    volumes:
        - ./data/certbot/conf:/etc/letsencrypt
                                                19,1
```

docker-compose up -d 명령어 입력