포팅메뉴얼

EC2 접속

```
# PEM 키가 있는 위치에서 아래의 명령어 수행
 ssh -i I10A405T.pem ubuntu@i10a405.p.ssafy.io
```

EC2 내부 설치

```
sudo apt-get update

sudo apt-get upgrade

# 편집을 위한 vim 설치
sudo apt-get install vim
```

TimeZone 설정

```
sudo timedatectl set-timezone Asia/Seoul
```

방화벽 확인

```
# 접속을 원하는 포트를 열어준다.
sudo ufw allow 8081

# 원하는 포트가 열려있는지 확인한다.
sudo ufw status

# 열려있어야 하는 포트번호 22 (ssh) , 80(http), 443(https)
```

포트포워딩을 위한 NGINX 설정

/etc/nginx/nginx.conf

```
http {

        ##
        # Basic Settings
        ##
```

```
        sendfile on;
        tcp_nopush on;
        tcp_nodelay on;
        keepalive_timeout 65;
        types_hash_max_size 2048;
        # server_tokens off;

        # server_names_hash_bucket_size 64;
        # server_name_in_redirect off;

        include /etc/nginx/mime.types;
        default_type application/octet-stream;

        ##
        # SSL Settings
        ##

        ssl_protocols TLSv1 TLSv1.1 TLSv1.2 TLSv1.3; # Dropping SSLv3, ref: POODLE
        ssl_prefer_server_ciphers on;

        ##
        # Logging Settings
        ##

        access_log /var/log/nginx/access.log;
        error_log /var/log/nginx/error.log;

        ##
        # Gzip Settings
        ##

        gzip on;

        # gzip_vary on;
        # gzip_proxied any;
        # gzip_comp_level 6;
        # gzip_buffers 16 8k;
        # gzip_http_version 1.1;
        # gzip_types text/plain text/css application/json application/javascript text/xml
application/xml application/xml+rss text/javascript;
```

```
        ##
        # socket
        ##
    map $http_upgrade $connection_upgrade {
       default upgrade;
       ''      close;
    }

    ##
    # Virtual Host Configs
    ##

    include /etc/nginx/conf.d/*.conf;
    include /etc/nginx/sites-enabled/*;
}
```

/etc/nginx/sites-available/default

```
server {

    root /var/www/html;

    # Add index.php to the list if you are using PHP
    index index.html index.htm index.nginx-debian.html;
  server_name i10a405.p.ssafy.io; # managed by Certbot


    location / {
        # 프론트엔드 서버 주소
        proxy_pass http://localhost:3000;
    }

    location /api {
        # 백엔드 서버 주소
        proxy_pass http://localhost:8080;

    }

    location /ws {
        proxy_pass http://localhost:8080;

        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;
        }


    listen [::]:443 ssl ipv6only=on; # managed by Certbot
    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/i10a405.p.ssafy.io/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/i10a405.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 ($host = i10a405.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


        listen 80 ;
        listen [::]:80 ;
    server_name i10a405.p.ssafy.io;
    return 404; # managed by Certbot

}
```

도커 권한 설정

```
# 도커는 항상 루트 권한으로 실행되기 때문에 현재 유저에게 도커 실행권한을 줘야한다.
sudo usermod -aG docker $USER
```

소스코드 받기

```
# gitlab으로 부터 소스 코드를 가져온다.
git clone  https://lab.ssafy.com/s10-webmobile1-sub2/S10P12A405.git

cd ./S10P12A405
```

프론트엔드 빌드 및 배포

```
# node 설치
curl -sL https://deb.nodesource.com/setup_20.11 | sudo -E bash -
sudo apt install nodejs npm

# node 버전 확인  (v20.10.0)이상
node -v

#yarn 설치  및 확인
npm install -g yarn
yarn --version


# frontend 폴더로 이동
cd ./fe

# 패키지 설치
yarn

# 빌드
yarn build

# 웹서버 역할의 nginx 컨테이너를 띄운다.
docker run -d -v $(pwd)/dist:/app -p 3000:80 --name frontend nginx:1.18

# 접속 후 업데이트 및 vim 설치
docker exec -it frontend bash
apt-get update
apt-get upgrade
apt-get install vim

# nginx 설정 파일 수정 후 reload
vim /etc/nginx/conf.d/default/conf
nginx -s reload
```

웹 서버 Nginx 설정

```
server {
   # 생략
    location / {
        root           /app;
        index    index.html;
    }
```

```
    #생략
}
```

백엔드 빌드 및 배포

```
# docker network를 생성한다. (backend, mysql, redis 통신을 위한)
docker network create db-network

# redis 컨테이너를 띄운다.
docker run --name redis -d -e TZ=Asia/Seoul --net db-network redis

# MYSQL 컨테이너를 띄운다. volume mount를 통해 로컬에 db를 저장한다.
# 네트워크 연결을 통해 통신이 가능하지만 포트를 열어둔 이유는 직접 제어를 해야할 경우가 있기 때문
docker run --name mysql -d -p 3307:3306 -e MYSQL_ROOT_PASSWORD=<root 패스워드> -e TZ=Asia/Seoul -v /otc/lib/mysql:/var/lib/mysql --net db-network mysql:8.0.35



# build를 위한 OpenJDK17 설치
sudo apt install openjdk-17-jdk
java --version

# Backend 폴더로 이동 후 빌드
cd ~/S10A405/be

# application.yaml 파일을 불러온다.
# 빌드
chmod +x ./gradlew
gradlew build -x test

#Backend 서버를 띄운다.
docker run -d -v $(pwd):/app -p 8080:8080 --name backend -e TZ=Asia/Seoul --net db-network openjdk:17

# 컨테이너 실행
docker exec -it backend bash

# jar파일 실행  프로그램의 systemout과 error를 logfile에 기록
java -jar /app/build/lib/<project snapshot>.jar  > /app/logfile.log 2>&1
```