운동을 기록하고 공유하는

GROUND

D103 : GROUND

SSAFY 구미캠퍼스 7기

공통 프로젝트 [2022.07.11 ~ 2022.08.19]

포팅 매뉴얼

담당 컨설턴트 : 서성수

한유빈(팀장), 김주영, 배시현, 박종욱, 박주현, 조인후

목차

1. 기술 스택

1. 이슈관리: Jira
2. 형상관리: Git
3. 커뮤니케이션: Mattermost, Webex, notion
4. 개발환경
   A. OS: Window 10
   B. IDE
      i. IntelliJ IEDA 2022.1.4
      ii. Visual Studio Code 1.67
5. UI/UX: Figma
6. Database
   A. Server: AWS RDS
   B. DBMS: MySQL 8.0.28
7. Server: AWS EC2
   A. OS: Ubuntu 20.04 LTS (GNU/Linux 5.4.0-1018-aws x86_64)
   B. SSH: MobaXterm
8. File Server: Firebase Storage
9. CI/CD: Jenkins, Docker, Nginx

10. 상세 기술

A. Backend

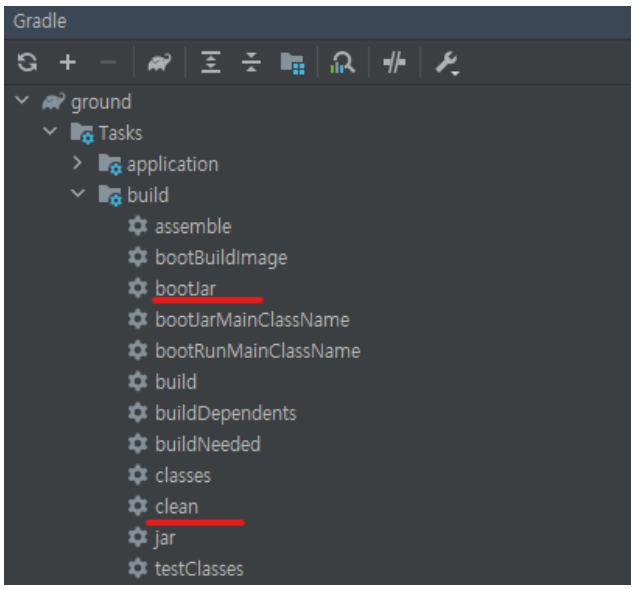
i. JDK: 11
ii. Spring Boot: 2.7.2
iii. Gradle 7.5
iv. Spring Data JPA
v. Springfox Swagger UI: 2.9.2
vi. Lombok
vii. Logger
viii. Json Web Token
ix. QLRM: 3.0.1

B. Frontend

i. HTML5, CSS3, JavaScript(ES6)
ii. React.js : 18.2.0
iii. Node.js : 16.16.0 LTS
iv. npm : 8.11.0
v. sass : 1.54.0
vi. Material-UI : 5.9.1
vii. http-proxy-middleware : 2.0.6
viii. axios : 0.27.2
ix. moment : 2.29.4
x. firebase : 9.9.2
xi. redux : 4.2.0
xii. redux-persist : 6.0.0
xiii. react-redux : 8.0.2
xiv. react-router-dom : 6.3.0
xv. react-calendar-heatmap : 1.9.0
xvi. react-frappe-charts : 4.1.0
xvii. react-hook-form : 7.33.1
xviii. react-loading : 2.0.3
xix. slick-carousel : 1.8.1

2. 빌드 상세 내용

1. Backend
   A. Gradle-Tasks-build



      i. clean으로 기존 jar파일 삭제
      ii. bootJar으로 현재 버전 jar 파일 생성
      iii. [Shift + F10] run 실행

2. Frontend
   A. frontend 디렉토리에 .env 파일 생성

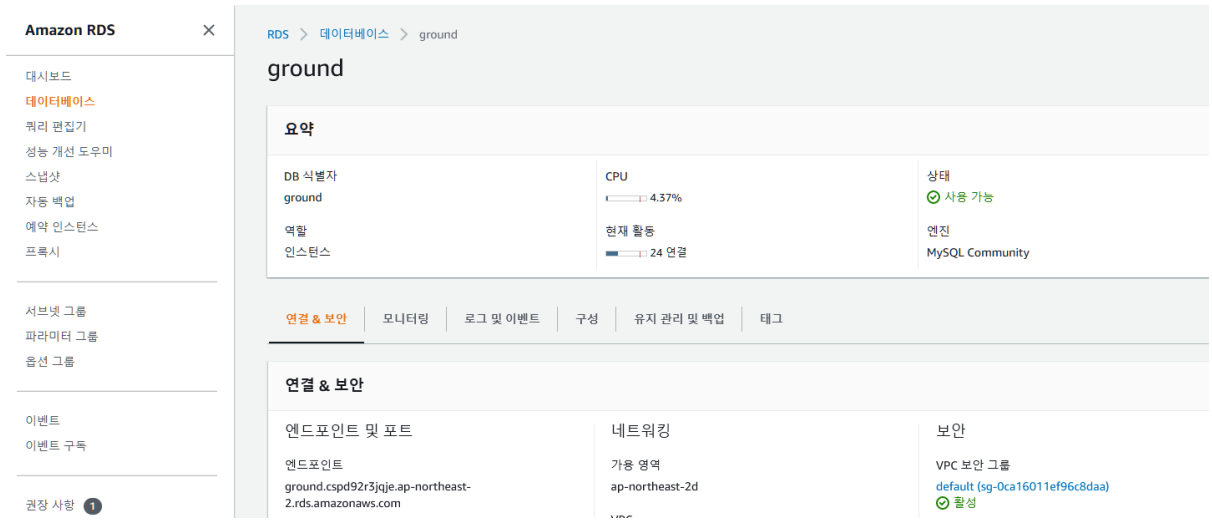
```
REACT_APP_KAKAO_REST_API_KEY=발급받은 키 값
REACT_APP_KAKAO_REDIRECT_URI=발급받은 키 값

REACT_APP_GOOGLE_CLIENT_ID=발급받은 키 값
REACT_APP_GOOGLE_REDIRECT_URI=발급받은 키 값

REACT_APP_FB_API_KEY=발급받은 키 값
REACT_APP_FB_AUTH_DOMAIN=발급받은 키 값
REACT_APP_FB_PROJECT_ID=발급받은 키 값
REACT_APP_FB_STORAGE_BUCKET=발급받은 키 값
REACT_APP_FB_MESSAGE_ID=발급받은 키 값
REACT_APP_FB_APP_ID=발급받은 키 값
```

   B. 다음 커맨드 실행

```
npm run build
```

DB 설정

- AWS RDS에서 엔진으로 MySQL 선택

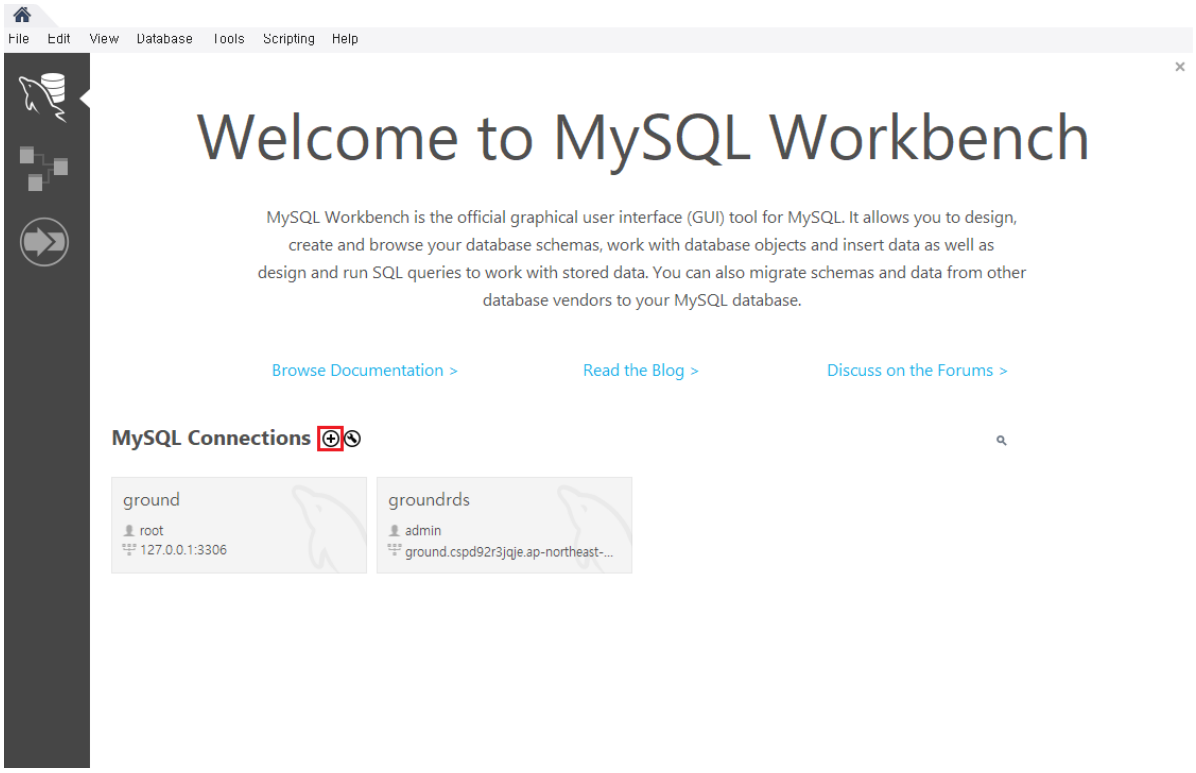


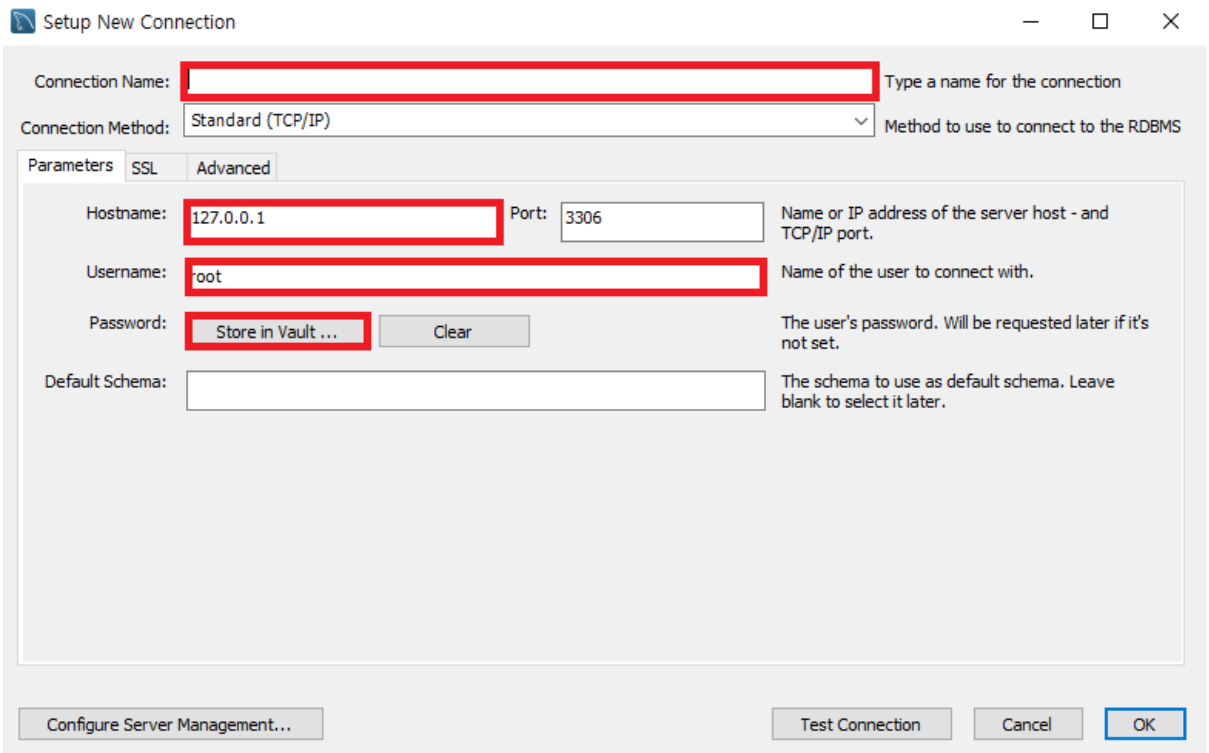
- Spring Boot Proejct와 연동

```
// aplication.yaml

datasource:
    url: [AWS RDS URL]
    username: [사용자 계정]
    password: [사용자 비밀번호]
    driver-class-name: com.mysql.cj.jdbc.Driver
```

MySQL Workbench 설정

- AWS RDS에 연결





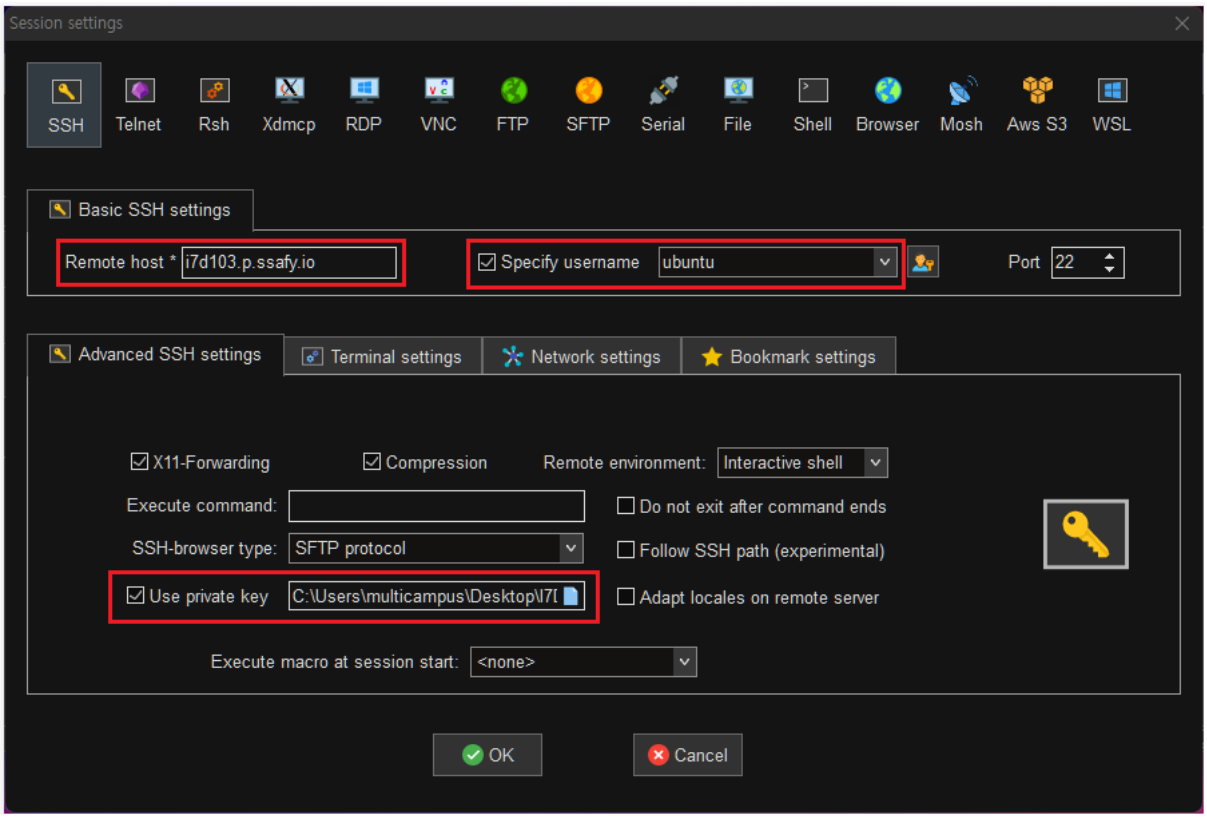
- Connection Name: ground-rds
- Hostname: AWS RDS URL
- Username: 사용자 계정
- Password: 사용자 비밀번호

- 초기 데이터 입력

```
-- 운동 종목
insert into t_category(id, event) values (1,"헬스");
insert into t_category(id, event) values (2,"요가");
insert into t_category(id, event) values (3,"필라테스");
insert into t_category(id, event) values (4,"러닝");
insert into t_category(id, event) values (5,"홈트레이닝");
insert into t_category(id, event) values (6,"축구");
insert into t_category(id, event) values (7,"야구");
insert into t_category(id, event) values (8,"농구");
insert into t_category(id, event) values (9,"테니스");
insert into t_category(id, event) values (10,"배드민턴");
insert into t_category(id, event) values (11,"등산");
insert into t_category(id, event) values (12,"수영");
insert into t_category(id, event) values (13,"골프");
insert into t_category(id, event) values (14,"볼링");
insert into t_category(id, event) values (15,"자전거/사이클");
insert into t_category(id, event) values (16,"기타");

-- 지역
insert into t_location(id, location) values (1, "서울");
insert into t_location(id, location) values (2, "경기");
insert into t_location(id, location) values (3, "인천");
insert into t_location(id, location) values (4, "강원");
insert into t_location(id, location) values (5, "충북");
insert into t_location(id, location) values (6, "세종");
insert into t_location(id, location) values (7, "대전");
insert into t_location(id, location) values (8, "충남");
insert into t_location(id, location) values (9, "경북");
insert into t_location(id, location) values (10, "대구");
insert into t_location(id, location) values (11, "울산");
insert into t_location(id, location) values (12, "경남");
insert into t_location(id, location) values (13, "부산");
insert into t_location(id, location) values (14, "광주");
insert into t_location(id, location) values (15, "전남");
insert into t_location(id, location) values (16, "제주");
```

CI/CD 설정

- EC2 접속을 위한 SSH 설정



- Remote host : AWS EC2 ip 주소 또는 연결된 도메인
- username : 사용자명
- Use private key 체크, 발급받은 key 파일 설정

- Nginx 설치

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install nginx
```

- docker 설치

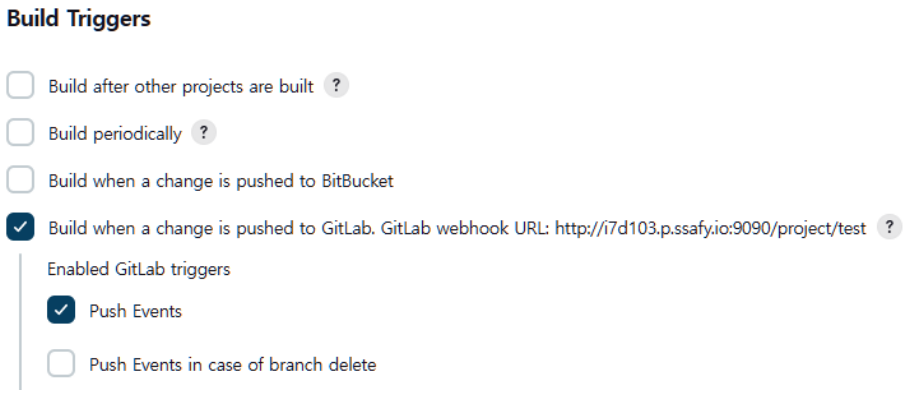
```
# 필수 패키지 설치
sudo apt-get install apt-transport-https ca-certificates
curl gnupg-agent software-properties-common
# GPG Key 인증
curl -fsSL https://download.docker.com/linux/ubuntu/gpg |
sudo apt-key add -
# docker repository 등록
sudo add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/ubuntu $(lsb_release -cs)
stable"
# 도커 설치
sudo apt-get update && sudo apt-get install docker-ce
docker-ce-cli containerd.io
# 도커 확인
sudo service docker statu
```

- Jenkins 컨테이너 실행

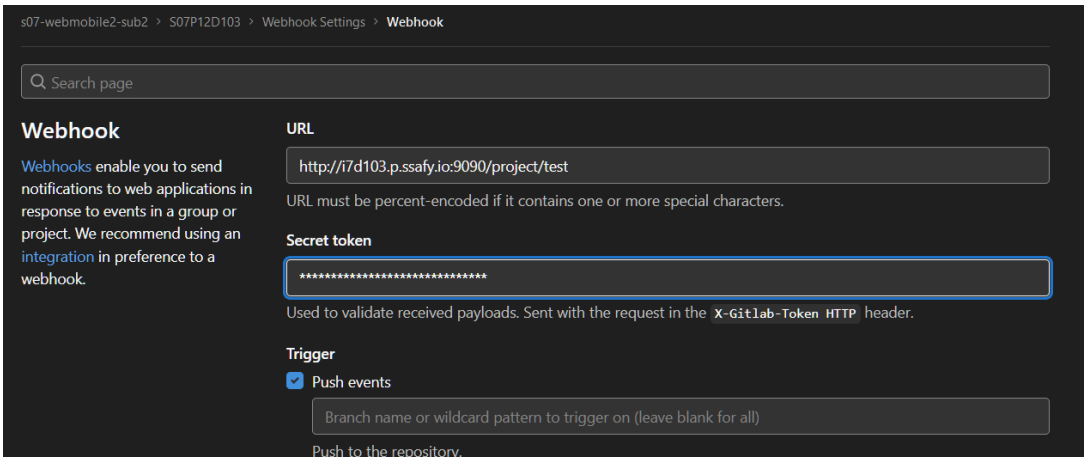
```
sudo docker run -d --name jenkins -u root --privileged \
-p '9090:8080' \
-v '/home/ubuntu/docker-volume/jenkins:/var/jenkins_home' \
-v '/var/run/docker.sock:/var/run/docker.sock' \
-v '/usr/bin/docker:/usr/bin/docker' \
jenkins/jenkins
```

- Jenkins 실행 후 빌드 파라미터 등록

- Webhook 등록
  - Jenkins



  - GitLab



- Jenkins pipleline script

```
pipeline {
      agent none
      stages {
          stage('Create .env') {
                  agent any
                  steps {
                                  sh 'echo
        "REACT_APP_KAKAO_REST_API_KEY=${REACT_APP_KAKAO_REST_API_KEY}
         \nREACT_APP_KAKAO_REDIRECT_URI=${REACT_APP_KAKAO_REDIRECT_URI}
         \nREACT_APP_GOOGLE_CLIENT_ID=${REACT_APP_GOOGLE_CLIENT_ID}
         \nREACT_APP_GOOGLE_REDIRECT_URI=${REACT_APP_GOOGLE_REDIRECT_URI}
         \nREACT_APP_FB_API_KEY=${REACT_APP_FB_API_KEY}
         \nREACT_APP_FB_AUTH_DOMAIN=${REACT_APP_FB_AUTH_DOMAIN}
         \nREACT_APP_FB_PROJECT_ID=${REACT_APP_FB_PROJECT_ID}
         \nREACT_APP_FB_STORAGE_BUCKET=${REACT_APP_FB_STORAGE_BUCKET}
         \nREACT_APP_FB_MESSAGE_ID=${REACT_APP_FB_MESSAGE_ID}
         \nREACT_APP_FB_APP_ID=${REACT_APP_FB_APP_ID}\n" > .env'
                          sh 'cp .env frontend'
                  }
              }
              stage('Docker build') {
                    agent any
                    steps {
                          sh 'docker build -t backimg ./backend/ground'
                          sh 'docker build -t frontimg ./frontend'
                    }
              }
              stage('Docker run') {
                    agent any
                    steps {
                          sh 'docker ps -f name=front -q \
                                  | xargs --no-run-if-empty docker container stop'

                          sh 'docker ps -f name=back -q \
                                  | xargs --no-run-if-empty docker container stop'

                          sh 'docker container ls -a -f name=front -q \
                                  | xargs -r docker container rm'

                          sh 'docker container ls -a -f name=back -q \
                                  | xargs -r docker container rm'
                          sh 'docker run -d --name front -p 80:80 frontimg'
                          sh 'docker run -d --name back -p 8080:8080 backimg'
                    }
              }
      }
}
```

1. **stage (”Create .env”)**

    A. Jenkins 에 등록한 환경변수를 사용해 .env 파일 생성
    B. .env 파일을 frontend 디렉토리로 복사

2. **stage (”Docker build”)**
    A. backend 폴더에 있는 Dockerfile 을 backimg 라는 이름으로 이미지화
    B. frontend 폴더에 있는 Dockerfile frontimg 라는 이름으로 이미지화

3. **stage(”Docker run”)**

    A. front 라는 이름의 컨테이너가 있으면 stop
    B. back 이라는 이름의 컨테이너가 있으면 stop
    C. front 라는 이름의 컨테이너가 있으면 remove
    D. back 이라는 이름의 컨테이너가 있으면 remove
    E. 빌드된 frontimg 컨테이너를 80 포트에서 실행
    F. 빌드된 backimg 컨테이너를 8080 포트에서 실행

- Dockerfile

1. Backend

```
FROM openjdk:11-jdk 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:11-jdk
COPY --from=builder build/libs/*.jar app.jar
```

    A. gradle bootJar 를 통해 jar 파일 빌드
    B. 빌드된 jar 파일 실행 (백엔드 서버 구동)

2. Frontend

```
FROM node:lts-alpine as build-stage
WORKDIR /app
COPY package*.json ./

RUN npm install
COPY . .
RUN npm run build

FROM nginx:stable-alpine as production-stage
RUN rm /etc/nginx/conf.d/default.conf
COPY ./nginx/deploy.conf
/etc/nginx/conf.d/deploy.conf


RUN rm -rf /usr/share/nginx/html/*
COPY --from=build-stage /app/build
/usr/share/nginx/html

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

A. /usr/share/nginx/html/* 삭제

B. build 폴더를 /usr/share/nginx/html 로 복사

C. nginx 세팅

i. /etc/nginx/conf.d/default.conf 삭제

ii. ./nginx/deploy.conf 파일을 /etc/nginx/conf.d/deploy.conf 복사

iii. deploy.conf

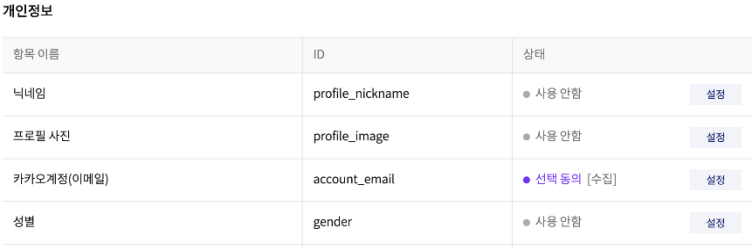
```
server {
    listen 80 default_server;
    listen [::]:80 default_server;

    index index.html;

    server_name _;

    location / {
        root /usr/share/nginx/html;
        try_files $uri $uri/ /index.html;
    }

    location /rest {
```

- 80 포트를 기본 포트로 설정: 포트 입력 없으면 80 포트로 연결
- 기본 URI 에서 user/share/nginx/html 에 있는 index.html 을 전달
- 프록시 설정을 통해 /rest 로 시작하는 uri 를 8080 포트로 연결

외부 서비스

- 카카오/구글 OAuth
  1. 동작 방식 (카카오, 구글 동일)
     A. 사용자가 카카오 로그인 및 활용 동의 시 프론트엔드에서 카카오 서버에 로그인을 요청한다.
     B. 카카오 서버는 사용자 정보를 확인하고 kakao developers 의 애플리케이션에 등록된 Redirect URI 로 리다이렉트 시킴과 동시에 인가 코드를 발급한다.
     C. 프론트엔드는 Redirect URI 에 발급 받은 인가 코드를 가져와 백엔드에 전송한다.
     D. 백엔드는 해당 인가 코드를 카카오 서버로 전송함과 동시에 사용자 정보를 요청한다.
     E. 카카오 서버는 인가 코드를 확인하고 백엔드로 사용자 정보를 전송한다.
     F. 백엔드는 사용자 정보를 DB 에서 조회하고 존재 유무에 따라 서로 다른 response 를 JWT 와 함께 프론트엔드에 전송한다.
  2. 설정
     A. Kakao developers
        i. 애플리케이션 등록
        ii. 내 애플리케이션 > 제품 설정 > 카카오 로그인
        iii. Redirect URI 에 인가 코드를 받을 경로 등록

        iv. 내 애플리케이션 > 제품 설정 > 카카오 로그인 > 동의항목



v. 개인 정보 > 카카오계정(이메일) 선택 동의 설정

B. Google cloud console

i. 애플리케이션 등록

ii. 사용자 인증 정보 > OAuth 2.0 클라이언트 ID

- Firebase

1. 콘솔 > 프로젝트 설정 > 내 앱에서 API 키 및 환경변수 확인 가능
2. firebase.json

```
{
  "storage": {
    "rules": "storage.rules"
  }
}
```

3. storage.rules

```
rules_version = '2';
service firebase.storage {
  match /b/{bucket}/o {
    match /{allPaths=**} {
      allow read, write: if true;
    }
  }
}
```

gitignore 파일

- frontend

```
# See https://help.github.com/articles/ignoring-
files/ for more about ignoring files.

# dependencies
/node_modules
/.pnp
.pnp.js
src/config

# testing
/coverage

# production
/build

# misc
.DS_Store
.env
.env.local
.env.development.local
.env.test.local
.env.production.local

npm-debug.log*
yarn-debug.log*
yarn-error.log*
```

- backend

```
HELP.md
.gradle/
build/
!gradle/wrapper/gradle-wrapper.jar
!**/src/main/**/build/
!**/src/test/**/build/

### STS ###
.apt_generated
.classpath
.factorypath
.project
.settings
.springBeans
.sts4-cache
bin/
!**/src/main/**/bin/
!**/src/test/**/bin/

### IntelliJ IDEA ###
.idea
*.iws
*.iml
*.ipr
out/
!**/src/main/**/out/
!**/src/test/**/out/

### NetBeans ###
/nbproject/private/
/nbbuild/
/dist/
/nbdist/
/.nb-gradle/

### VS Code ###
.vscode/
```