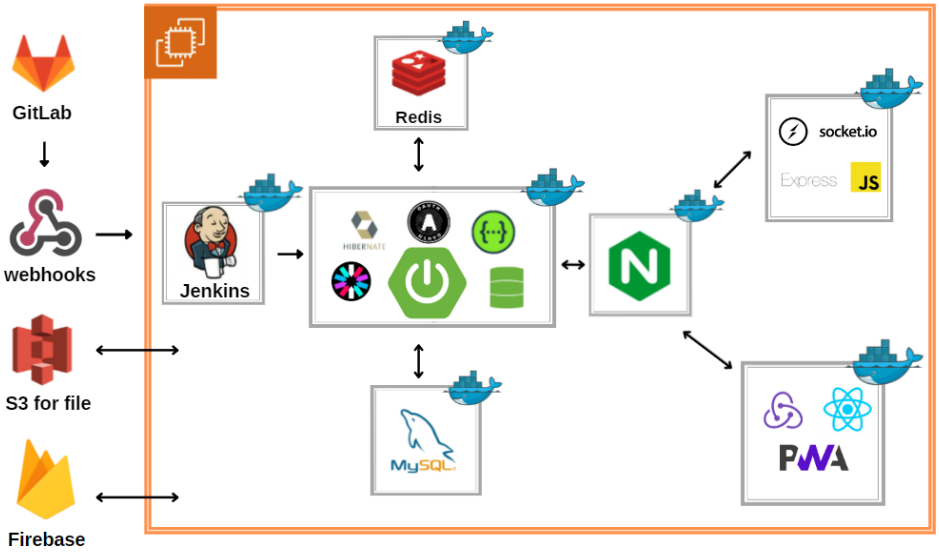
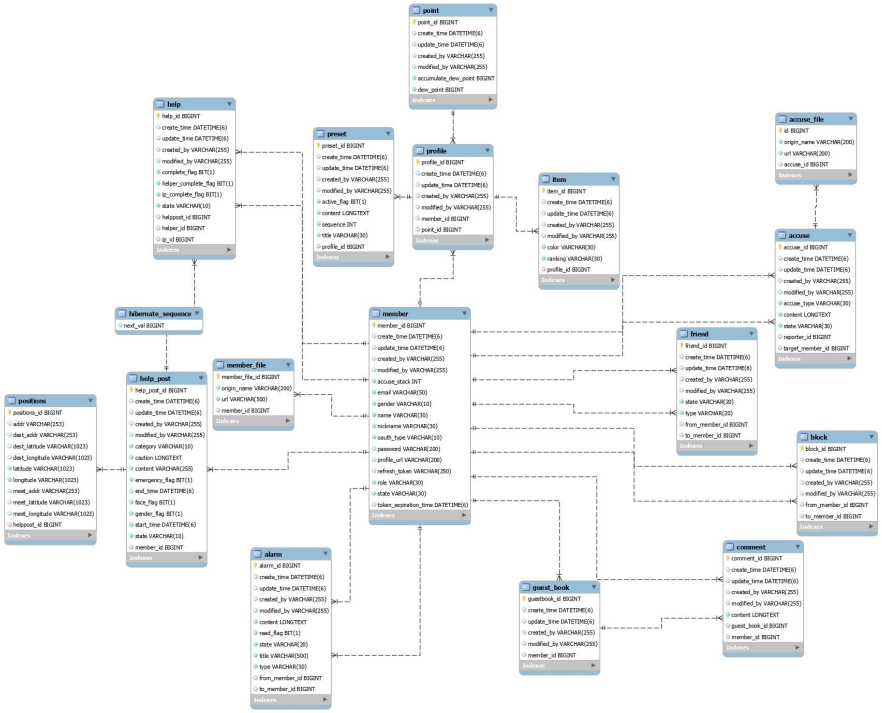
포팅매뉴얼

아키텍처



ERD



1. 빌드 및 배포

1. 개발 환경

- Server : AWS EC2 Ubuntu 20.04.6 LTS
- Visual Studio Code : 1.70.1
- IntelliJ IDEA : 2022.1.3(Ultimate Edition)
- java: 11.0.20
- JVM : 11.0.20+8-post-Ubuntu-1ubuntu120.04
- Docker : 24.0.5
- Node.js : 18.17.0
- MySQL : Ver 8.1.0 for Linux on x86_64
- Redis : 7.0.12
- Nginx : 1.25.1
- Jenkins : 2.346.2
- Socket.io: 2.3.0

2. 설정 파일 목록

React

- .env : /jenkins/workspace/frontend/frontend
- Dockerfile : /jenkins/workspace/frontend/frontend
- deploy.sh : /jenkins/workspace/frontend/frontend

Spring

- application.yml: /jenkins/workspace/backend/BE/src/main/resources
- application-prod.yml: /jenkins/workspace/backend/BE/src/main/resources
- Dockerfile : /jenkins/workspace/backend/BE
- deploy.sh : /jenkins/workspace/backend/BE

Docker

- docker-compose.yml : /home/ubuntu

Nginx

- app.conf : /home/ubuntu/nginx/conf.d

3. 설정 파일 및 환경 변수 정보

React

- .env.production (프론트가 가릴거 가려주셈)

```
REACT_APP_KAKAO_RESTAPI_KEY = [KAKAO Application key]
REACT_APP_KAKAO_JS_KEY = [KAKAO Map Key]
REACT_APP_KAKAO_OAUTH_REDIRECT_URI = [도메인]/user/login/oauth
REACT_APP_KAKAO_OUATH__PATH = /api/v1/oauth/kakao/callback
REACT_APP_OPENROUTESERVICE_KEY = [openrouteservice키]

WDS_SOCKET_PORT = 0
```

```
REACT_APP_API_KEY = [Firebase Project Application key]
REACT_APP_MESSAGING_SENDER_ID = [Firebase Caller ID]
REACT_APP_APP_ID = [Application ID]
REACT_APP_MEASUREMENT_ID = [Mesurement ID]
REACT_APP_VAPID_KEY = [Project Valid key]

REACT_APP_SERVER = [도메인]
```

- Dockerfile

```
FROM node:18.17.0-alpine
WORKDIR /usr/src/app
COPY ./ /usr/src/app/
RUN npm install
RUN npm run build
COPY ./ /usr/src/app/
EXPOSE 3000
CMD ["npm","run","start"]
```

- deploy.sh

```
echo '캐시없이 도커파일 이미지 빌드'
docker build --no-cache -t reactapp .

echo '컨테이너 중지'
docker stop reactapp

echo '컨테이너 삭제'
docker rm reactapp

echo '컨테이너 실행'
docker run -p 3000:3000 --name reactapp -d reactapp

echo '중지된 컨테이너 사용하지 않는 이미지 모두 강제 삭제'
docker system prune -a -f
```

Express

- deploy.sh

```
echo '캐시없이 도커파일 이미지 빌드'
docker build --no-cache -t expressapp .

echo '컨테이너 중지'
docker stop expressapp

echo '컨테이너 삭제'
docker rm expressapp

echo '컨테이너 실행'
docker run -p 3478:3478 --name expressapp -d expressapp

echo  '중지된 컨테이너 사용하지 않는 이미지 모두 강제 삭제'
docker system prune -a -f
```

- Dockerfile

```
FROM node:18.17.0-alpine
WORKDIR /usr/src/app
COPY ./ /usr/src/app/
RUN npm install
COPY ./ /usr/src/app/
EXPOSE 3478
CMD ["npm","run","start"]
```

Spring

- .env

```
MYSQL_DATABASE={DB이름}
MYSQL_USER={DB user 아이디}
MYSQL_PASSWORD={DB user 비밀번호}

SPRING_DATASOURCE_DRIVER=com.mysql.cj.jdbc.Driver
SPRING_DATASOURCE_URL=jdbc:mysql://{mysql docker 컨테이너 이름}:3306/danbi?useSSL=false&useUnicode=true&characterEncoding=utf8&allowPubli
SPRING_DATASOURCE_USERNAME={DB user 아이디}
SPRING_DATASOURCE_PASSWORD={DB user 비밀번호}
SPRING_REDIS=redis

PASSWORD={jasypt 비밀번호}
```

- Dockerfile

```
FROM openjdk:11-jdk
ARG JAR_FILE=build/libs/*.jar
COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java", "-Dspring.profiles.active=prod", "-jar", "-Duser.timezone=Asia/Seoul","app.jar"]
```

- deploy.sh

```
echo '빌드 시작'
./gradlew clean build -x test

echo '캐시없이 도커파일 이미지 빌드'
docker build --no-cache -t springbootapp .

echo '컨테이너 중지'
docker stop springbootapp

echo '컨테이너 삭제'
docker rm springbootapp

echo '컨테이너 실행'
docker run --env-file ./.env -p 8080:8080 --name springbootapp --network ubuntu_default -d springbootapp

echo '중지된 컨테이너 사용하지 않는 이미지 모두 강제 삭제'
docker system prune -a -f
```

Docker

- docker-compose.yml

```
version: "3.0"

services:
  redis:
    image: redis
    container_name: redis
    ports:
      - 6379:6379

  mydb:
    image: mysql
    container_name: mydb
    volumes:
      - ./db:/var/lib/mysql
    environment:
      MYSQL_ROOT_PASSWORD: {mysql root 계정 비밀번호}
      MYSQL_DATABASE: {mysql 데이터베이스 이름}
      MYSQL_USER: {mysql user 계정 아이디}
      MYSQL_PASSWORD: {mysql user 계정 비밀번호}
```

```
    ports:
      - 3306:3306

  nginx:
    image: nginx
    container_name: nginx
    ports:
      - 80:80
      - 443:443
    volumes:
      - ./nginx/conf.d:/etc/nginx/conf.d
      - /etc/letsencrypt:/etc/letsencrypt

jenkins:
    image: jenkins/jenkins:lts
    container_name: jenkins
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - /jenkins:/var/jenkins_home
    ports:
      - 9090:8080
    privileged: true
    user: root
```

Nginx

- app.conf

```
server {
  listen 443 ssl;
  server_name i9d207.p.ssafy.io;

  # ssl 인증서 적용하기
  ssl_certificate /etc/letsencrypt/live/i9d207.p.ssafy.io/fullchain.pem;
  ssl_certificate_key /etc/letsencrypt/live/i9d207.p.ssafy.io/privkey.pem;

  location / { # 프론트엔드
    proxy_pass http://i9d207.p.ssafy.io:3000;
  }

  location /api { # 백엔드
    proxy_pass http://i9d207.p.ssafy.io:8080;
    proxy_set_header Host $http_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; # https 필요

    # 웹 소켓 설정
    proxy_set_header Connection "upgrade";
    proxy_set_header Upgrade $http_upgrade;
  }

   location /room { # 백엔드
    proxy_pass http://i9d207.p.ssafy.io:3478;
    proxy_set_header Host $http_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; # https 필요

    # 웹 소켓 설정
    proxy_set_header Connection "upgrade";
    proxy_set_header Upgrade $http_upgrade;
  }

location /ws {
        proxy_pass http://i9d207.p.ssafy.io:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header Host $host;
        proxy_set_header Origin "";
    }

}
```

Java 설치

```
sudo apt-get update && sudo apt-get upgrade
sudo apt-get install openjdk-11-jdk
java -version
```

4. Docker 설치

Docker설치

```
sudo apt update # 패키지 업데이트

# https관련 패키지 설치
sudo apt install apt-transport-https ca-certificates curl software-properties-common

# docker repository 접근을 위한 gpg 키 설정
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 focal stable"

#다시 업데이트
sudo apt update

#도커 설치
sudo apt install docker-ce

#설치 확인
docker --version
```

Docker Compose설치

```
sudo curl -L
"https://github.com/docker/compose/releases/download/1.29.2/dockercompose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
```

5. SSL 인증서 발급

```
apt-get install letsencrypt # Let's Encrypt 설치
sudo letsencrypt certonly --standalone -d [도메인] # 인증서 적용 및 pem키 발급
cd /etc/letsencrypt/live/[도메인] # 발급 경로 확인
vim custom.conf # conf 파일 생성
```

6. DB 및 Infra 배포

```
# docker-compose 작성
# docker-compose 실행
$ cd /home/ubuntu (docker-compose.yml 경로에서)
$ sudo docker-compose up –build –d (docker-compose 실행)

# 서버 콘솔로 돌아가 Jenkins 내부에 docker를 설치
$ sudo docker exec -it jenkins컨테이너명 /bin/bash
$ apt-get update -y
$ apt-get install -y
$ apt-get install docker.io -y
$ docker –v
```

Build Steps

- backend

```
echo 'jenkinsbuild started...'
pwd
cd BE
chmod +x gradlew
./deploy.sh
```

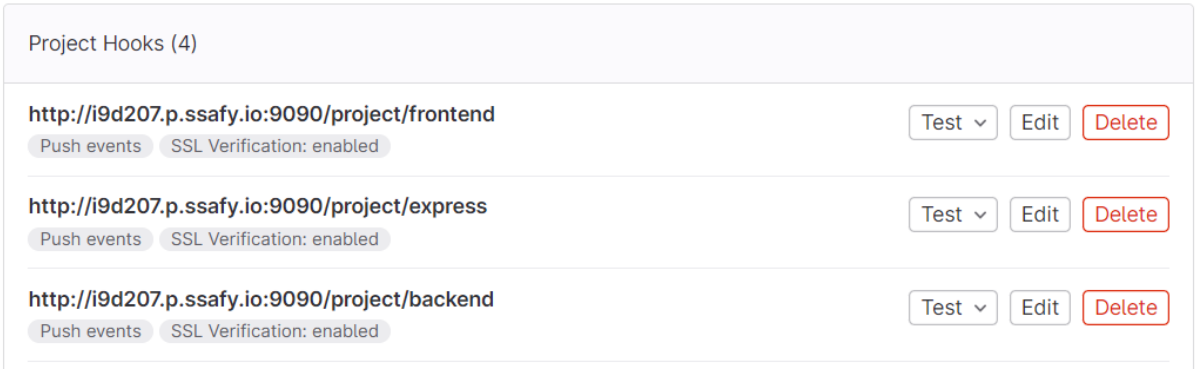
- frontend

```
echo 'jenkinsbuild start...'
pwd
cd FE/danbi
pwd
./deploy.sh
```

- express

```
echo 'jenkinsbuild start...'
pwd
cd Express
pwd
./deploy.sh
```

빌드 유발(Enabled GitLab triggers)



7. 방화벽 설정

```
# ec2 접속
ssh -i I9D207T.pem ubuntu@i9d207.p.ssafy.io

1. ufw 상태 확인
```

```
$ sudo ufw status
Status : inactive

2. 사용할 포트 허용하기 (ufw inactive 상태)
$ sudo ufw allow 22

2-1 등록한 포트 조회하기 (ufw inactive 상태)
$ sudo ufw show added
Added user rules (see 'ufw status' for running firewall):
ufw allow 22

3. ufw 활성화 하기
$ sudo ufw enable
Command may disrupt existing ssh connections. Proceed with operation (y|n)? y

3.1 ufw 상태 및 등록된 rule 확인하기
$ sudo ufw status numbered
Status: active

     To                         Action      From
     --                         ------      ----
[ 1] 22                         ALLOW IN    Anywhere
[ 2] 22 (v6)                    ALLOW IN    Anywhere (v6)

4. 새로운 터미널을 띄워 ssh 접속해 본다.
C:\> ssh -i 팀.pem ubuntu@팀.p.ssafy.io

5. ufw 구동된 상태에서 80 포트 추가하기
$ sudo ufw allow 80

5-1. 80 포트 정상 등록되었는지 확인하기
$ sudo ufw status numbered
Status: active

     To                         Action      From
     --                         ------      ----
[ 1] 22                         ALLOW IN    Anywhere
[ 2] 80                         ALLOW IN    Anywhere
[ 3] 22 (v6)                    ALLOW IN    Anywhere (v6)
[ 4] 80 (v6)                    ALLOW IN    Anywhere (v6)

5-2. allow 명령을 수행하면 자동으로 ufw에 반영되어 접속이 가능하다.

6. 등록한 80 포트 삭제 하기
$ sudo ufw status numbered
Status: active

     To                         Action      From
     --                         ------      ----
[ 1] 22                         ALLOW IN    Anywhere
[ 2] 80                         ALLOW IN    Anywhere
[ 3] 22 (v6)                    ALLOW IN    Anywhere (v6)
[ 4] 80 (v6)                    ALLOW IN    Anywhere (v6)

6-1. 삭제할 80 포트의 [번호]를 지정하여 삭제하기
      번호 하나씩 지정하여 삭제한다.
$ sudo ufw delete 4
$ sudo ufw delete 2
$ sudo ufw status numbered  (제대로 삭제했는지 조회해보기)
Status: active

     To                         Action      From
     --                         ------      ----
[ 1] 22                         ALLOW IN    Anywhere
[ 2] 22 (v6)                    ALLOW IN    Anywhere (v6)

6-2 (중요) 삭제한 정책은 반드시 enable을 수행해야 적용된다.
$ sudo ufw enable
Command may disrupt existing ssh connections. Proceed with operation (y|n)? y입력


기타
- ufw 끄기
$ sudo ufw disable

# 특정 IP 방화벽 허용
sudo ufw allow from ${IP_ADDRESS}

# 특정 IP 주소와 프로토콜, 포트 허용
sudo ufw allow from ${IP_ADDRESS} to any port 22
```

포트

```
sudo ufw allow 8080
sudo ufw deny 8080
```

Allowed Ports

```
22, SSH
80, HTTP
8080, Spring
3306, MySQL
443,HTTPS
3000, React, Nginx
3478, Express
```

2. 외부 서비스

1. 소셜 로그인

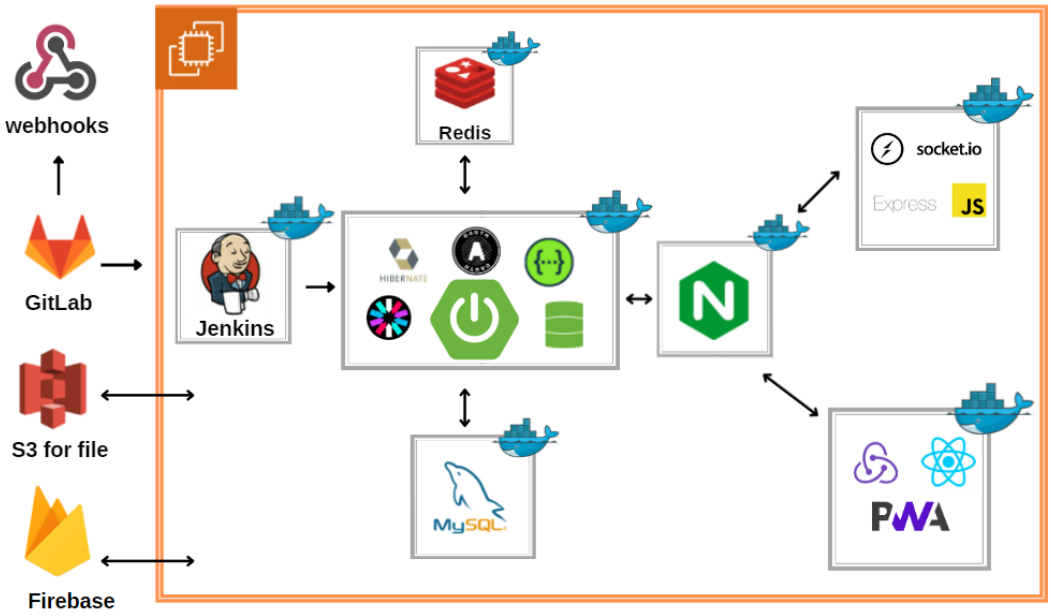
- KAKAO

2. 지도 서비스

- KAKAO MAP
- openrouteservice

3. STT / TTS

- Web Speech API


webhooks
GitLab
S3 for file
Firebase
Redis
Jenkins
HIBERNATE
MySQL
socket.io
Express
JS
PWA