



포팅 매뉴얼

- 1. 프로젝트 기술 스택
- 2. 빌드 상세 내용
 - 2.1. JVM, 웹서버, WAS 종류 및 설정값, 버전
 - 2.2. 빌드 주요 내용
 - 2.3. 배포 시 특이사항
 - 2.4. 프로젝트(ERD)에 활용되는 주요 계정 및 프로퍼티가 정의된 파일
- 3. 외부 서비스
 - 3.1. Ncloud sms API
 - 3.2 카카오 로그인

1. 프로젝트 기술 스택

- 이슈관리: Jira
- 형상관리: Gitlab
- 커뮤니케이션: Mattermost, Webex, Notion
- 개발환경
 - OS: Window 10
 - IDE
 - IntelliJ IEDA
 - Visual Studio Code: 1.67
 - UI/UX: Figma
 - Database
 - Server: AWS RDS
 - DBMS: MySQL 8.0.28
 - Server: AWS EC2
 - OS: Ubuntu 20.04 LTS (GNU/Linux 5.4.0-1018-aws x86_64)
 - SSH: MobaXterm
 - CI/CD: Jenkins, Docker, Nginx
- 상세 기술
 - Backend
 - local
 - JDK: 11
 - Spring Boot: 2.7.3
 - docker : 20.10.12
 - server (ec2)
 - docker : 20.10.12

- jenkins : 2.346.2
- nginx : 1.23.1
- mariaDB : 10.8.3
- Frontend
 - HTML5, CSS3, JavaScript(ES6)
 - React.js : 18.0
 - Node.js : 16.17.0 LTS
 - npm : 8.15.0
 - sass : 1.54.9
 - Material-UI : 5.10.7
 - react-redux : 8.0.2
 - react-router-dom : 6.4.0
 - axios : 0.27.2
 - redux : 4.2.0
 - storybook: 6.5.12

2. 빌드 상세 내용

2.1. JVM, 웹서버, WAS 종류 및 설정값, 버전

- JVM : openjdk:11-jdk (back/iljungitjung/Dockerfile 참조)
- IntelliJ : 2022.2
- node.js : 16.16.0
- docker : 20.10.12
- webserver : nginx 1.23.1 (Docker)
 - docker-compose 를 통해 image를 생성하고 container로 띄움

```
ubuntu@ip-172-26-8-189:~/docker-volume/ssl$ ls
cert.pem  chain.pem  fullchain.pem  key.p12  privkey.pem
```

- /home/ubuntu/docker-volume/webserver/conf/default.conf 내용

```

server{
    listen      443 ssl;
    server_name k7d106.p.ssafy.io;
    ssl_certificate /ssl/fullchain.pem;
    ssl_certificate_key /ssl/privkey.pem;

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

    location /api {
        proxy_pass http://host.docker.internal:9091;
        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;
    }
}

server{
    listen      443 ssl;
    server_name localhost;

    ssl_certificate /ssl/fullchain.pem;
    ssl_certificate_key /ssl/privkey.pem;

    location /api {
        proxy_pass http://host.docker.internal:9091;
        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;
    }
}

```

- database : mariadb 10.8.3 (Docker)
 - docker-compose 를 통해 image를 생성하고 container로 띄움

```

volumes:
  - "/home/ubuntu/docker-volume/mariadb/conf.d:/etc/mysql/conf.d"
  - "/home/ubuntu/docker-volume/mariadb/data:/etc/lib/mysql"

```

◦

```

[mariadb]
skip-host-cache
skip-name-resolve

!includedir /etc/mysql/mariadb.conf.d/
!includedir /etc/mysql/conf.d/

lower_case_table_names = 1
collation-server = utf8_unicode_ci
init-connect='SET NAMES utf8'
character-set-server = utf8

```

- 'ijjj' database 생성

```
MariaDB [ijij]> show tables;
+-----+
| Tables_in_ijij |
+-----+
| category       |
| schedule       |
| user           |
+-----+
3 rows in set (0.000 sec)
```

- redis (Docker)
 - docker-compose 를 통해 image를 생성하고 container로 띄움
 - /home/ubuntu/docker-volume/redis/redis.conf 내용

2.2. 빌드 주요 내용

- Dockerfile

```
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 build

VOLUME /tmp

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

EXPOSE 8080

ENTRYPOINT ["java", "-jar", "-Dspring.profiles.active=deploy", "/app.jar"]
```

- gradle을 통해 빌드한다.
- 빌드한 파일을 openjdk:11-jdk 환경에서 java -jar -Dspring.profiles.active=deploy boot/app.jar를 통해 실행시킨다.
- deploy → application-deploy.properties 파일 사용
- deploy.sh

```

docker build -t iljungitjung_server .

DOCKER_APP_NAME=iljungitjung_server
WEBSERVER_NAME=iljungitjung_webserver_1

EXIST_BLUE=$(docker-compose -p ${DOCKER_APP_NAME}-blue -f docker-compose.blue.yml ps | grep Up)

if [ -z "$EXIST_BLUE" ]; then
    echo "BLUE UP"
    docker-compose -p ${DOCKER_APP_NAME}-blue -f docker-compose.blue.yml up -d
    sleep 20
    START_PORT=9090
    TERMINATE_PORT=9091
    docker exec ${WEBSERVER_NAME} sed -i "s/${TERMINATE_PORT}/${START_PORT}/" /etc/nginx/conf.d/default.conf
    echo "nginx reload..."
    docker exec ${WEBSERVER_NAME} service nginx reload
    sleep 5
    docker-compose -p ${DOCKER_APP_NAME}-green -f docker-compose.green.yml down
else
    echo "GREEN UP"
    docker-compose -p ${DOCKER_APP_NAME}-green -f docker-compose.green.yml up -d
    sleep 20
    START_PORT=9091
    TERMINATE_PORT=9090
    docker exec ${WEBSERVER_NAME} sed -i "s/${TERMINATE_PORT}/${START_PORT}/" /etc/nginx/conf.d/default.conf
    echo "nginx reload..."
    docker exec ${WEBSERVER_NAME} service nginx reload
    sleep 5
    docker-compose -p ${DOCKER_APP_NAME}-blue -f docker-compose.blue.yml down
fi

```

- Dockerfile을 통해 iljungitjung_server라는 이름의 이미지로 빌드한다.
- docker-compose를 이용해 iljungitjung_server-blue라는 container가 돌아가는지 확인하고, 존재한다면 docker-compose를 이용해 iljungitjung_server-green container를 올리고 nginx container 내부의 9090과 9091을 바꾼 후, nginx를 reload 해준다.
- 만약 iljungitjung_server-blue가 없다면 반대로 진행해준다.
- docker-compose.yml

```

version: '3.0'

services:
  webserver:
    image: nginx
    restart: unless-stopped
    ports:
      - 443:443
    volumes:
      - "/home/ubuntu/docker-volume/ssl:/ssl"
      - "/home/ubuntu/docker-volume/webserver/conf:/etc/nginx/conf.d"
      - "/home/ubuntu/docker-volume/webserver/dist:/usr/share/nginx/html/dist"
    extra_hosts:
      - "host.docker.internal:host-gateway"

  redis:
    image: redis
    volumes:
      - "/home/ubuntu/docker-volume/redis/redis.conf:/usr/local/etc/redis/redis.conf"
    environment:
      - REDIS_PASSWORD=ijij
      - ALLOW_EMPTY_PASSWORD=no
    restart: unless-stopped
    ports:
      - 6379:6379

```

```

extra_hosts:
  - "host.docker.internal:host-gateway"

database:
  image: mariadb
  restart: unless-stopped
  ports:
    - 3306:3306
  volumes:
    - "/home/ubuntu/docker-volume/mariadb/conf.d:/etc/mysql/conf.d"
    - "/home/ubuntu/docker-volume/mariadb/data:/etc/lib/mysql"
  environment:
    MARIADB_DATABASE: ijj
    MARIADB_USER: ijj
    MARIADB_PASSWORD: ijj
    MARIADB_ROOT_PASSWORD: ijj
    - TZ=Asiacd redis
  extra_hosts:
    - "host.docker.internal:host-gateway"

```

2.3. 배포 시 특이사항

- 배포시 `deploy.sh`를 실행시키면 Dockerfile과 docker-compose 파일을 사용하여 알아서 docker image를 만들고 docker-compose 파일을 이용하여 container를 띄운다.
 - 배포시 jenkins에서 `deploy.sh`를 실행시키기 때문에 jenkins는 사용자 권한이 있어야한다.

Build Steps

Execute shell ?

Command

See [the list of available environment variables](#)

```
cd back/iljungitjung
docker-compose up -d
```

고급...

Execute shell ?

Command

See [the list of available environment variables](#)

```
cd back/iljungitjung
ls
sh ./deploy.sh
```

- jenkins또한 docker로 띄웠기 때문에 jenkins container 내부에 docker, docker-compose 설치해주고 사용자 권한을 줘야한다.

- 해당 container는 9090:8080 or 9091:8080으로 포트 설정이 되어있으며 nginx(Docker)를 reverse_proxy로 사용하여 <https://k7d106.p.ssafy.io/api> url로 들어온 모든 요청을 해당 container로 돌린다.
- 아무런 table이 없는 DB에 초기화 데이터를 사용하기 위해선 `application-deploy.properties` 내부 설정을 변경해준다.
(1회)

```
spring.config.activate.on-profile=deploy

server.servlet.context-path=/api

spring.datasource.url=jdbc:mariadb://host.docker.internal:3306/ijij
spring.datasource.driver-class-name=org.mariadb.jdbc.Driver
spring.datasource.username=ijij
spring.datasource.password=ijij

spring.jpa.hibernate.ddl-auto=update

login.kakao.path=/oauth/authorize
login.kakao.host=kauth.kakao.com
login.kakao.rest_api_key={API_Key}
login.kakao.redirect_uri=https://k7d106.p.ssafy.io/api/oauth/kakao?client-uri=https://k7d106.p.ssafy.io/calendar/my
login.kakao.response_type=code
login.kakao.user_info_server_uri=https://kapi.kakao.com/v2/user/me
login.kakao.register_client_uri=https://k7d106.p.ssafy.io/register
oauth.kakao.grant_type=authorization_code
oauth.kakao.token.path=/oauth/token

spring.redis.host=host.docker.internal
spring.redis.port=6379
spring.redis.password=ijij

message.nccloud.service_id={Service_ID}
message.nccloud.access_key={Access_key}
message.nccloud.secret_key={Secret_Key}
message.nccloud.phone={Sender_phone}
```

- 1회만 해당 properties로 빌드 후 이후에는 원래대로 빌드해준다.
- 프론트 배포시 jenkins에서 제공해주는 nodejs 플러그인을 사용해서 빌드 후 nginx container 내부에 docker cp로 옮겨준다.

Execute NodeJS script
?

NodeJS Installation
Specify nodejs installation where npm installed packages to execute the script
16.18.0

Script
See [the list of available environment variables](#) accessible by process.env.ENV_VARIABLE.

npmrc file
- use system default -

Cache location
Default (~/.npm or %APP_DATA%\npm-cache)

Execute shell
?

Command
See [the list of available environment variables](#)

```
cd front
ls
npm install esbuild
npm run build
docker cp dist iljungitjung_webserver_1:/usr/share/nginx/html
```

2.4. 프로젝트(ERD)에 활용되는 주요 계정 및 프로퍼티가 정의된 파일

- back/iljungitjung/src/main/resources/application-deploy.properties

3. 외부 서비스

3.1. Ncloud sms API

메시지 및 알람을 전송하고 전송 현황을 실시간으로 확인할 수 있는 API로, 사용을 위해 플랫폼 access key와 secret key, service ID 발급이 필요하다.

1. Ncloud(<https://www.ncloud.com/>) 접속 및 가입
2. 마이페이지 > 계정관리 > 인증키 관리 접속하여 신규 API 인증키 생성
3. access key ID와 Secret Key 복사

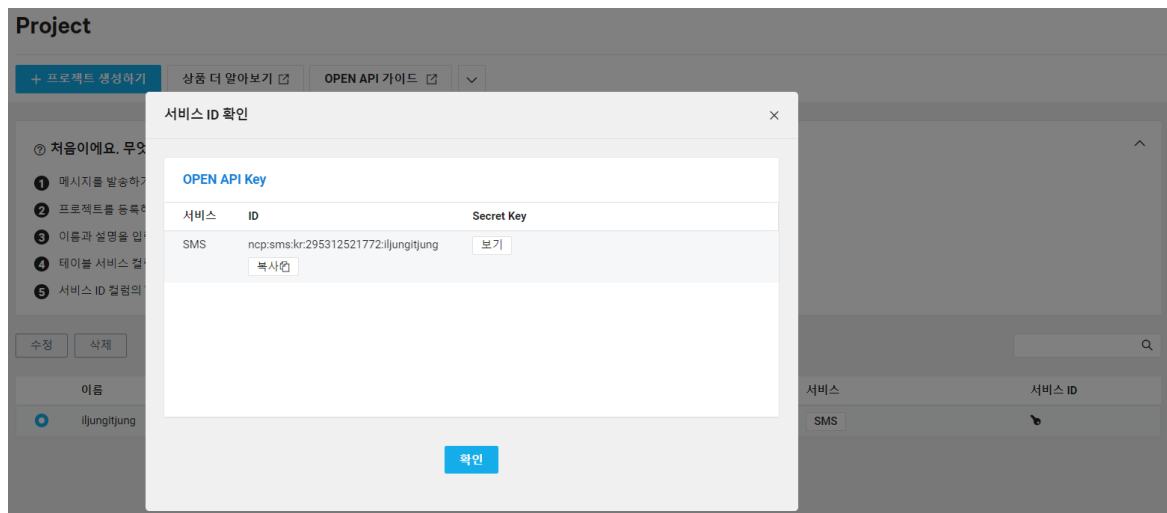
API 인증키 관리

신규 API 인증키 생성

Access Key ID	Secret Key	생성일자	상태	관리
aHxTolxJrQW0FMKaPuti	보기	2022년 10월 31일	사용 중	사용 중지
hbULeYt2uent66JVK2Q	보기	2022년 11월 19일	사용 중	사용 중지

4. 서비스 > Simple & Easy Notification Service 접속하여 이용 신청하기

5. 콘솔로 이동하여 Project에서 프로젝트 생성 후 서비스 ID 복사



6. SMS > Calling Number 접속하여 발신번호 등록

SMS Calling Number

[상품 더 알아보기](#) [OPEN API 가이드](#) [▼](#)

② SMS 발신번호는 어떻게 등록하나요 ?

- 1 발신번호 등록은 하단의 발신번호 등록 Tab을 클릭하세요
- 2 하단의 발신번호 조회 Tab에서 발신번호 등록 현황을 확인할 수 있습니다.
- 3 등록된 발신번호는 SMS 발송시에 사용할 수 있습니다.

프로젝트명 iljungitjung ▼

발신번호 조회 발신번호 등록

7. 발급받은 ID와 key, 발신번호는 [application-deploy.properties](#)에서 ncloud 환경변수로 설정(2.3 참고)

3.2 카카오 로그인

- 카카오 로그인 API 신청

앱 설정

요약 정보

일반

비즈니스

앱 키

플랫폼

팀 관리

제품 설정

카카오 로그인

동의항목


간편가입

카카오톡 채널

개인정보 국외이전

연결 끊기

사용자 프로퍼티



iljungitjung

ID 811036

OWNER

Biz

Web

앱 키

네이티브 앱 키	239d5de205a2988962d844993336df0b
REST API 키	658d16aaf6e1b553e600b3badabd1847
JavaScript 키	ff2ea0e2e8197fbaa1a61a8df9c6e531
Admin 키	82b97dd0d07b94821af3588dbb649751

플랫폼

Android	-
---------	---

- 리다이렉트 URI 등록

Redirect URI		삭제	수정
Redirect URI	http://localhost:8080/oauth http://localhost:3000 http://localhost:8080/oauth/kakao?client-uri=http://localhost:3000/calendar/my https://k7d106.p.ssafy.io/api/oauth/kakao?client-uri=https://k7d106.p.ssafy.io/calendar/my		

- application.properties에 등록

```

login.kakao.rest_api_key=658d16aaf6e1b553e600b3badabd1847
login.kakao.redirect_uri=https://k7d106.p.ssafy.io/api/oauth/kakao?client-uri=https://k7d106.p.ssafy.io/calend

```