



# 포팅 메뉴얼

1. 개발 환경 및 기술 스택
2. 설정 파일 및 환경 변수 정보
3. 빌드 및 배포
  - 1) Docker 설치
  - 2) Docker compose 설치
  - 3) Jenkins 컨테이너 실행
  - 4) docker compose를 통한 실행
  - 5) NGINX 설정
4. 외부 서비스 및 정보
5. DB dump 설명

## 1. 개발 환경 및 기술 스택

- Backend
  - JAVA 17 (17.0.9 2023-10-17 LTS)
  - Spring boot 3.2.1
  - OAUTH 2.0
  - Spring Security 6.2.1
  - Spring Data JPA
  - Mysql 8.0.35
  - Redis 5.0.7
  - RabbitMQ 3.12.12
  - OpenVidu 2.29.0
- Frontend
  - Node.js 20.10.0

- npm 10.2.3
- Vue 3.3.11
- vite 4.5.2
- YJS 13.6.11
- Y-websocket 1.5.3
- Infra
  - Ubuntu 20.04.6 LTS
  - Nginx 1.18.0
  - Jenkins 2.442
  - docker 25.0.3
- IDE
  - IntelliJ Ultimate
  - Vscode

## 2. 설정 파일 및 환경 변수 정보

### 1. Spring boot

- application.yml (로컬용)

```
spring:
  jpa:
    open-in-view: false
    defer-datasource-initialization: true
    generate-ddl: true
    hibernate:
      ddl-auto: create-drop # ddl 자동 작성 0
    properties:
      hibernate:
        format_sql: true # 하이버네이트가 실행한 SQL
        use_sql_comments: true
        show_sql: true # 하이버네이트가 실행한 SQL
      jdbc:
```

```

        batch_size: 100                # insert/update 쿼리
        default_batch_fetch_size: 100

datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver # DB 드라이버
    url: jdbc:mysql://localhost:3306/plancard?useUnicode=true
    username: ssafy                        # 데이터베이스 계정명
    password: ssafy                        # 데이터베이스 계정비밀번호

# data_테이블명.sql 관련 실행 setting
sql:
    init:
        mode: always
        data-locations:
            - 'classpath:/data_member.sql'
            - 'classpath:/data_place.sql'
            - 'classpath:/data_plan.sql'
            - 'classpath:/data_card.sql'
            - 'classpath:/data_alarm.sql'
            - 'classpath:/data_friendship.sql'
            - 'classpath:/data_plan_member.sql'

    batch:
        jdbc:
            initialize-schema: always
        job:
            enabled: false

# NoSQL setting
data:
    # Redis setting
    redis:
        host: localhost
        port: 6379

    # mongodb setting
    mongodb:
        uri: mongodb://localhost:27017/planCard

```

```

# Java Mail Sender setting (Google Mail)
mail:
  host: smtp.gmail.com
  port: 587
  username: donggeun3484@gmail.com
  password: tbplnbqivtwteeki
  properties:
    mail:
      smtp:
        auth: true
        starttls:
          enable: true

# RabbitMQ setting
rabbitmq:
  host: localhost # RabbitMQ 서버 호스트
  port: 15672 # RabbitMQ 서버 포트, 기본값은 15672
  username: guest # RabbitMQ 사용자 이름, 기본값은 guest
  password: guest # RabbitMQ 비밀번호, 기본값은 guest
  virtual-host: / # RabbitMQ 가상 호스트, 기본값은 /

# jwt setting
jwt:
  secret-key:
    access: TestAccessKey1111111222222222233333333333333aaa
    refresh: TestRefreshKey1111111222222222233333333333333a

  expired-min:
    access: 60 # 액세스 토큰 만료제한시간 60분 (1시간) (60)
    refresh: 10080 # 리프레쉬 토큰 만료제한시간 10080분 (7일) (10080)

# log 관리
logging:
  level:
    org.hibernate:
      type.descriptor.sql: trace

```

```
org.hibernate.SQLQuery: debug

# 공공데이터
public-data:
  key: 734364614c73656e32386579495665

stt:
  id: YrKMFxbs_H29nZ1cZs3r
  secret: -wXxmNVuqYD-F1_jNH7DRYgQ0s0SlL8Vb2W2ArJR

server:
  ssl:
    enabled: false

openvidu:
  url: http://localhost:4443/
  secret: MY_SECRET

# OAUTH2.0 Setting
oauth:
  kakao:
    client-id: 8e8a2a929b0cbf786ac9567191ac7ff7
    client-secret: 0aZuL5rd6IcxxcwDd4pgcKRnJP9afsHo
    redirect-uri: http://localhost:5173/member/loading/kak
    scope:
      - profile_nickname
      - profile_image
      - account_email
      - name

  naver:
    client-id: d5K90tbM_Jby0ZIi0G54
    client-secret: UAm7jJPmSc
    redirect_uri: http://localhost:5173/member/loading/nav
    scope:
```

```

- nickname
- name
- email
- profile_image

# firebase setting
app:
  firebase-configuration-file: classpath:serviceAccountKey
  firebase-bucket: plancard-3b8f9.appspot.com

```

- application.yml (배포용)

```

spring:
  jpa:
    open-in-view: false
    defer-datasource-initialization: true
    generate-ddl: true
    hibernate:
      ddl-auto: create-drop # ddl 자동 작성 0
    properties:
      hibernate:
        format_sql: true # 하이버네이트가 실행한 SQL
        use_sql_comments: true
        show_sql: true # 하이버네이트가 실행한 SQL
      jdbc:
        batch_size: 100 # insert/update 쿼리
        default_batch_fetch_size: 100

  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver # DB 드라이버
    url: jdbc:mysql://plancard-db.cts0808eovz6.ap-northeast-2.amazonaws.com:3306/plancard
    username: admin # 데이터베이스 계정명
    password: ssafy1234 # 데이터베이스 비밀번호

# data_테이블명.sql 관련 실행 setting
sql:
  init:
    mode: always

```

```

    data-locations:
      - 'classpath:/data_member.sql'
      - 'classpath:/data_place.sql'
      - 'classpath:/data_plan.sql'
      - 'classpath:/data_card.sql'
      - 'classpath:/data_alarm.sql'
      - 'classpath:/data_friendship.sql'
      - 'classpath:/data_plan_member.sql'

batch:
  jdbc:
    initialize-schema: always
  job:
    enabled: false

# NoSQL setting
data:
  # Redis setting
  redis:
    host: i10C110.p.ssafy.io
    port: 6379

    # mongodb setting
  mongodb:
    #      host: mongodb
    #      port: 27017
    uri: mongodb://mongodb:27017/planCard

# Java Mail Sender setting (Google Mail)
mail:
  host: smtp.gmail.com
  port: 587
  username: donggeun3484@gmail.com
  password: tbplnbqivtwteeki
  properties:
    mail:
      smtp:

```

```

        auth: true
        starttls:
            enable: true

# RabbitMQ setting
rabbitmq:
    host: rabbitmq    # RabbitMQ 서버 호스트
    port: 5672        # RabbitMQ 서버 포트, 기본값은 15672
    username: guest   # RabbitMQ 사용자 이름, 기본값은 guest
    password: guest   # RabbitMQ 비밀번호, 기본값은 guest
    virtual-host: /   # RabbitMQ 가상 호스트, 기본값은 /

# jwt setting
jwt:
    secret-key:
        access: TestAccessKey1111111222222222233333333333333aaa
        refresh: TestRefreshKey1111111222222222233333333333333a

    expired-min:
        access: 100000000 # 액세스 토큰 만료제한시간 60분 (1시간)
        refresh: 100000000 # 리프레쉬 토큰 만료제한시간 10080분 (7일)

# log 관리
logging:
    level:
        org.hibernate:
            type.descriptor.sql: trace
            org.hibernate.SQLQuery: debug

# 공공데이터
public-data:
    key: 734364614c73656e32386579495665

stt:
    id: YrKMFxbs_H29nZ1cZs3r
    secret: -wXxmNVuqYD-F1_jNH7DRYgQ0s0S1L8Vb2W2ArJR

```



```

server:
  ssl:
    enabled: false

openvidu:
  url: https://i10c110.p.ssafy.io:4443
  secret: MY_SECRET

# OAUTH2.0 Setting
oauth:
  kakao:
    client-id: 8e8a2a929b0cbf786ac9567191ac7ff7
    client-secret: 0aZuL5rd6IcxxcwDd4pgcKRnJP9afsHo
    redirect-uri: https://i10c110.p.ssafy.io/member/loading
    scope:
      - profile_nickname
      - profile_image
      - account_email
      - name

  naver:
    client-id: d5K90tbM_Jby0ZIi0G54
    client-secret: UAm7jJPmsC
    redirect-uri: https://i10c110.p.ssafy.io/member/loading
    scope:
      - nickname
      - name
      - email
      - profile_image

# firebase setting
app:
  firebase-configuration-file: classpath:serviceAccountKey
  firebase-bucket: plancard-3b8f9.appspot.com

```

## 2. Vue.js

- .env(로컬용)

```
# API URL settings for PJT
VITE_VUE_API_URL=http://localhost:8080/api/v1
VITE_KAKAO_MAP_SERVICE_KEY=ec21f2ebe162718a8bd658f5bb7a0bd

# Stomp WebSocket 주소 설정
VITE_VUE_WS_URL=ws://localhost:8080/ws

# audio 처리를 위한 WebSocket 주소 설정
VITE_VUE_AUDIO_WS_URL=ws://localhost:8080/audio

# YJS y-websocket 주소 설정
VITE_VUE_YJS_WS_URL=ws://localhost:1234/yjs

# 기본 URL 설정
BASE_URL=
```

- .env(배포용)

```
# API URL settings for PJT
VITE_VUE_API_URL=https://i10c110.p.ssafy.io/api/v1
VITE_KAKAO_MAP_SERVICE_KEY=ec21f2ebe162718a8bd658f5bb7a0bd

# Stomp WebSocket 주소 설정
VITE_VUE_WS_URL=wss://i10c110.p.ssafy.io/ws

# audio 처리를 위한 WebSocket 주소 설정
VITE_VUE_AUDIO_WS_URL=wss://i10c110.p.ssafy.io/audio

# YJS y-websocket 주소 설정
VITE_VUE_YJS_WS_URL=wss://i10c110.p.ssafy.io/yjs

# 기본 URL 설정
BASE_URL=
```

### 3. 빌드 및 배포

#### 1) Docker 설치

```
# 1. Docker 엔진의 공식 버전을 설치하기 전 충돌하는 패키지를 제거해야 한다
for pkg in docker.io docker-doc docker-compose docker-compose

# 2. Docker apt 저장소 설정
# Add Docker's official GPG key:
sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg
sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" |
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update

# 3. Docker 패키지 설치
# 최신 버전 설치
sudo apt-get install docker-ce docker-ce-cli containerd.io do

# 4. hello-world 이미지를 실행하여 Docker 엔진 설치가 성공했는지 확인
sudo docker run hello-world
```

#### 2) Docker compose 설치

```
# 설치
sudo apt update
sudo apt install docker.io
```

```
sudo apt install docker-compose
```

```
# 설치 확인
```

```
docker --version
```

```
docker-compose --version
```

### 3) Jenkins 컨테이너 실행

```
# 네트워크 생성
```

```
sudo docker network create jenkins-network
```

```
# jenkins 컨테이너 실행
```

```
sudo docker run -d --name jenkins --network jenkins-network
```

### 4) docker compose를 통한 실행

- Jenkins를 통해 docker-compose.yml파일의 구성을 실행시킨다.

#### docker-compose.yml

```
version: '3.8'

services:
  backend:
    image: pji0128/back:latest
    container_name: back
    build:
      context: ./BackEnd
      dockerfile: Dockerfile
    ports:
      - "8081:8080"
    networks:
      - jenkins-network
```

```

environment:
  RABBITMQ_HOST: rabbitmq

mongodb:
  image: mongo
  command: mongod --bind_ip 0.0.0.0
  container_name: mongodb
  ports:
    - "27017:27017"
  networks:
    - jenkins-network
  volumes:
    - mongodb_data:/data/db

frontend:
  image: pji0128/front:latest
  container_name: front
  build:
    context: ./FrontEnd
    dockerfile: Dockerfile
  ports:
    - "8083:4173"
  networks:
    - jenkins-network
  environment:
    - HOST=0.0.0.0
    - PORT=1234

y-websocket:
  image: node:latest
  container_name: y-websocket-server
  command: sh -c "npm -g install yjs y-socket && HOST=0.0.0.0"
  ports:
    - "1234:1234"
  networks:
    - jenkins-network

volumes:

```

```
mongodb_data:

networks:
  jenkins-network:
    external: true
```

## BE Jenkinsfile

```
pipeline {
  agent any
  environment {
    REPO = "s10-webmobile1-sub2/S10P12C110"
    DOCKERHUB_REGISTRY = "pji0128/back"
    DOCKERHUB_CREDENTIALS = credentials('Docker-credential')
  }
  stages {
    stage('Checkout') {
      steps {
        checkout scm
      }
    }
    stage('Setup Environment') {
      steps {
        dir("${env.WORKSPACE}/BackEnd"){
          script {
            sh "ls -al"
            sh "chmod +x ./gradlew"
          }
        }
      }
    }
    stage("Build") {
      steps {
        script {
          sh "docker build -t ${DOCKERHUB_REGISTRY}"
        }
      }
    }
  }
}
```

```

    }
}
stage("Login") {
    steps {
        sh "echo \${DOCKERHUB_CREDENTIALS_PSW} | doc
    }
}
stage("Tag and Push") {
    steps {
        script {
            withCredentials([[ $class: 'UsernamePasswo
                sh "docker push ${DOCKERHUB_REGISTRY}
            }
        }
    }
}
stage('Prune old images'){
    steps{
        script{
            sh "docker ps"
        }
    }
}
stage('Pull') {
    steps {
        script {
            withCredentials([[ $class: 'UsernamePasswo
                sh "docker stop back || true" // Ign
                sh "docker rm back || true"    // Ign
                sh "docker rmi ${DOCKERHUB_REGISTRY}|
                sh "docker pull ${DOCKERHUB_REGISTRY}
            }
        }
    }
}
stage('Up') {
    steps {
        script {

```





## 5) NGINX 설정

### nginx.conf

```
user www-data;
worker_processes auto;
pid /run/nginx.pid;
include /etc/nginx/modules-enabled/*.conf;

events {
    worker_connections 768;
    # multi_accept on;
}

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; # Droppi
    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 app.

##
# Virtual Host Configs
##

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

#mail {
#       # See sample authentication script at:
#       # http://wiki.nginx.org/ImapAuthenticateWithApachePhp
#
#       # auth_http localhost/auth.php;
#       # pop3_capabilities "TOP" "USER";
#       # imap_capabilities "IMAP4rev1" "UIDPLUS";
#

```

```
#       server {
#           listen      localhost:110;
#           protocol    pop3;
#           proxy        on;
#       }
#
#       server {
#           listen      localhost:143;
#           protocol    imap;
#           proxy        on;
#       }
#}
```

## default

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

    root /var/www/html;

    index index.html index.htm index.nginx-debian.html;

    server_name _;

    location / {
        proxy_pass http://i10c110.p.ssafy.io:8083;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        #
    }

    # Backend
    location /api/v1 {
```

```

        proxy_pass http://i10c110.p.ssafy.io:8081;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }

    location /ws {
        proxy_pass http://i10c110.p.ssafy.io:8081;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header Host $host;
    }

    location /yjs {
        proxy_pass http://i10c110.p.ssafy.io:1234;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header Host $host;
    }
}

server {
    listen 9001 ssl;
    server_name i10c110.p.ssafy.io;

    ssl_certificate /etc/letsencrypt/live/i10c110.p.ssafy
    ssl_certificate_key /etc/letsencrypt/live/i10c110.p.s

    location / {
        proxy_pass http://localhost:9000;
    }
}

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 i10c110.p.ssafy.io; # managed by Certbot


location / {

    proxy_pass http://i10c110.p.ssafy.io:8083;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
}

# Backend
location /api/v1 {

    proxy_pass http://i10c110.p.ssafy.io:8081;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
}

location /ws {
    proxy_pass http://i10c110.p.ssafy.io:8081;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
    proxy_set_header Host $host;
}

location /yjs {
    proxy_pass http://i10c110.p.ssafy.io:1234;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
    proxy_set_header Host $host;
}

```

```

listen [::]:443 ssl ipv6only=on; # managed by Certbot
listen 443 ssl; # managed by Certbot
ssl_certificate /etc/letsencrypt/live/i10c110.p.ssafy.io/
ssl_certificate_key /etc/letsencrypt/live/i10c110.p.ssafy
include /etc/letsencrypt/options-ssl-nginx.conf; # manage
ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed

}
server {
    if ($host = i10c110.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

    listen 80 ;
    listen [::]:80 ;
    server_name i10c110.p.ssafy.io;
    return 404; # managed by Certbot

}

```

## 4. 외부 서비스 및 정보

- 카카오 소셜 로그인 정보

카카오 로그인 ☐ ON[동의 화면 미리보기](#)

## 동의항목

카카오 로그인으로 서비스를 시작할 때 동의받는 항목을 설정합니다. 미리보기를 통해 사용자에게 보여질 화면을 확인할 수 있습니다.  
원본 앱보다 더 많은 동의항목을 사용할 수 있습니다. 권한이 필요한 동의항목은 원본 앱에서 심사 신청 후 사용할 수 있습니다.

## 개인정보

항목 이름	ID	상태	
닉네임	profile_nickname	● 필수 동의	<a href="#">설정</a>
프로필 사진	profile_image	● 필수 동의	<a href="#">설정</a>
카카오계정(이메일)	account_email	● 필수 동의 [수집]	<a href="#">설정</a>
이름	name	● 필수 동의 [수집]	<a href="#">설정</a>

- 네이버 소셜 로그인 정보

애플리케이션 이름 ⇄	SSAFY_Project_PlanCard <ul style="list-style-type: none"> <li>네이버 로그인할 때 사용자에게 표시되는 이름이므로 서비스 브랜드를 대표할 수 있도록 가급적 10자 이내로 간결하게 설정해주세요.</li> <li>40자 이내의 영문, 한글, 숫자, 공백문자, 쉼표(,), "/", "-", "_", 만 입력 가능합니다.</li> </ul>															
카테고리	기타 ▼															
	선택하세요. ▼															
사용 API ⇄	<p>네이버 로그인</p> <p>제공 정보 선택(이용자 식별자는 기본 정보로 제공) ①</p> <p>필수 항목은 개인정보보호법 제3조 제1항, 제16조 제1항 등에 따라 서비스 제공을 위해 필요한 최소한의 개인정보를 선택해야 합니다.</p> <table border="1"> <thead> <tr> <th>권한</th> <th>필수</th> <th>추가</th> </tr> </thead> <tbody> <tr> <td>회원이름</td> <td><input checked="" type="checkbox"/></td> <td><input type="checkbox"/></td> </tr> <tr> <td>연락처 이메일 주소</td> <td><input checked="" type="checkbox"/></td> <td><input type="checkbox"/></td> </tr> <tr> <td>별명</td> <td><input checked="" type="checkbox"/></td> <td><input type="checkbox"/></td> </tr> <tr> <td>프로필 사진</td> <td><input checked="" type="checkbox"/></td> <td><input type="checkbox"/></td> </tr> </tbody> </table>	권한	필수	추가	회원이름	<input checked="" type="checkbox"/>	<input type="checkbox"/>	연락처 이메일 주소	<input checked="" type="checkbox"/>	<input type="checkbox"/>	별명	<input checked="" type="checkbox"/>	<input type="checkbox"/>	프로필 사진	<input checked="" type="checkbox"/>	<input type="checkbox"/>
권한	필수	추가														
회원이름	<input checked="" type="checkbox"/>	<input type="checkbox"/>														
연락처 이메일 주소	<input checked="" type="checkbox"/>	<input type="checkbox"/>														
별명	<input checked="" type="checkbox"/>	<input type="checkbox"/>														
프로필 사진	<input checked="" type="checkbox"/>	<input type="checkbox"/>														

## 5. DB dump 설명

테이블들은 JPA로 자동으로 DB와 매핑하여 Spring boot 서버 내에서 자동으로 생성되게끔 설정하였습니다.

해당 db dump 데이터는 PlanCard\BackEnd\src\main\resources 폴더내에 저장되어 있습니다.

- data\_member.sql (회원 데이터)
- data\_place.sql (여행지 데이터)
- data\_plan.sql (여행 계획 데이터)
- data\_card.sql (카드 데이터)
- data\_alarm.sql (알람(친구 추가 알람, 여행 계획 참여 요청 알람) 데이터)
- data\_friendship.sql (친구 관계 데이터)
- data\_plan\_member.sql (해당 여행 계획에 같이 갈 멤버 데이터)