



포팅 매뉴얼

Index

- [서버 설정](#)
 - [기본 설정](#)
 - [배포 자동화 설정](#)
 - [Docker 설정](#)
 - [Jenkins 설정](#)
 - 애플리케이션 배포 설정
 - [SpringBoot 배포 설정](#)
 - [FastAPI](#)
 - [React](#)
- [데이터베이스 설정](#)
 - [MySQL](#)
 - [AWS RDS 인스턴스 생성](#)
 - [DB 인스턴스 VPC에 인바운드 규칙 설정](#)
 - [데이터베이스 파라미터 그룹 설정](#)
 - [MySQLWorkbench에 접속](#)
 - [기본 데이터 저장](#)
 - [Redis](#)
- [OpenAPI 설정](#)
 - [카카오 로그인](#)
- [버전 정보](#)
 - [Back-end](#)
 - [SpringBoot](#)
 - [FastAPI](#)
 - [Front-end](#)
 - [React](#)

포트

port	설명
22	ssh
80	http
443	https
9090	jenkins

서버 설정

기본 설정

1. AWS EC2 접속

```
ssh -i J8B104T.pem ubuntu@j8b104.p.ssafy.io
```

2. Nginx 설치

```
sudo apt update
sudo apt install nginx
sudo systemctl start nginx # 서비스 시작
sudo systemctl status nginx # 서비스 상태 확인
```

3. 방화벽 설정

```
sudo apt install ufw
sudo ufw default deny incoming # 들어오는 패킷 차단
sudo ufw default allow outgoing # 나가는 패킷 허용

# 기본 포트 설정
sudo ufw allow ssh
sudo ufw allow http
sudo ufw allow https

# 방화벽 설정
sudo ufw enable
```

4. SSL 인증서 툴 설치

```
# 인증서 설치를 위한 Certbot 설치
sudo apt update
sudo apt-get install letsencrypt -y
sudo apt install certbot python3-certbot-nginx
```

5. Nginx 초기 설정

```
sudo cp /etc/nginx/sites-available/default /etc/nginx/sites-available/dodo.conf
#####

## dodo.conf
server {
    listen 80;
    listen [::]:80;

    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 j8b104.p.ssafy.io;

    location / {
        # First attempt to serve request as file, then
        # as directory, then fall back to displaying a 404.
        try_files $uri $uri/ =404;
    }
}

#####

sudo ln -s /etc/nginx/sites-available/dodo.conf /etc/nginx/sites-enabled/dodo.conf
```

6. SSL 인증서 발급

```

sudo certbot --nginx -d j8b104.p.ssafy.io

vim /etc/nginx/sites-available/dodo.conf
#####

## dodo.conf
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 j8b104.p.ssafy.io;

    location / {
        # First attempt to serve request as file, then
        # as directory, then fall back to displaying a 404.
        try_files $uri $uri/ =404;
    }

    listen [::]:443 ssl ipv6only=on; # managed by Certbot
    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/j8b104.p.ssafy.io/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/j8b104.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 ($host = j8b104.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

    listen 80;
    listen [::]:80;

    server_name j8b104.p.ssafy.io;
    return 404; # managed by Certbot
}
}

```

배포 자동화 설정

1. 도커 설정

a. 도커 gpg키 추가 및 레포지토리 셋업

```

sudo apt-get update

sudo apt-get install -y ca-certificates \
> curl \
> software-properties-common \
> apt-transport-https \
> gnupg \
> lsb-release

sudo mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

# OK 출력 시 정상 작동
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

echo \
> "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \
> $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

```

b. 도커 설치

```

sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin

```

c. 도커 설치 확인

```
# 도커 엔진 설치 확인
sudo docker run hello-world

# 버전 확인
sudo docker -v
```

d. 도커 실행

```
sudo systemctl enable docker

# 실행중인 도커 서비스 정보 확인
docker info

# 도커 권한 적용 확인
docker ps
```

2. 도커를 이용해 Jenkins 실행

a. jenkins home 폴더 생성

```
cd ~
mkdir jenkins
mkdir home
```

b. 배포를 위한 설정 폴더 생성

```
mkdir ~/jenkins/be/spring/resources
mkdir ~/jenkins/be/python/resources
```

c. Docker Compose 작성

```
cd ~/jenkins
vim docker-compose.yaml

### docker-compose.yaml
version: '3.3'
services:
  jenkins:
    image: jenkins/jenkins:lts-jdk11
    container_name: jenkins
    environment:
      - TZ=Asia/Seoul
    user: root
    privileged: true
    ports:
      - 9090:8080
      - 50000:50000
    volumes:
      - ./home:/var/jenkins_home
      - /var/run/docker.sock:/var/run/docker.sock
      - /usr/bin/docker:/usr/bin/docker
      - /home/ubuntu/jenkins/be/spring/resources:/var/jenkins_home/workspace/dodo-spring/be/spring/dodo/res
      - /home/ubuntu/jenkins/be/python/resources:/var/jenkins_home/workspace/dodo-fastapi/be/python/resourc
```

d. docker compose 실행

```
sudo docker-compose up -d
```

3. Jenkins 설정

a. `http://j8b104.p.ssafy.io:9090` 접속

b. Administrator Password 입력

- EC2에서 아래 명령어로 확인

```
sudo docker logs jenkins
```

c. Install suggested plugin 설치

d. 설치 완료 후 → 플러그인 관리에서 아래 플러그인들 설치

- GitLab
- BlueOcean
- SSH Agent

e. Global Credentials 설정

- 깃랩 레포지토리에서 API키 생성 후 Jenkins에 `GitLab API token` 등록
 - id: gitlab-api-token
- `Username with password` 로 깃랩 아이디와 패스워드 등록
 - id: gitlab-username
- jenkins에서 EC2접속을 위한 `SSH Username with private key` 등록
 - id: ssh-agent
 - username: ubuntu
 - Private Key → Enter directly → key에 EC2 접속을 위한 pem키 내용 입력
- `Secret text` 에 jenkins에서 도커 이미지를 올릴 도커 허브 패스워드 등록
 - id: dockerHubPwd

f. jenkins 관리 → 시스템 설정 → GitLab connection 설정

Gitlab

☒ Enable authentication for '/project' end-point

GitLab connections

Connection name
A name for the connection

dodo

Gitlab host URL
The complete URL to the Gitlab server (e.g. `http://gitlab.mydomain.com`)

https://lab.ssafy.com

Credentials
API Token for accessing Gitlab

GitLab API token

Add +

Remove -

Test Connection

g. 배포 설정

- SpringBoot 서버 배포 설정
 1. GitLab Connection: dodo 선택

2. Build Trigger 설정

Build Triggers

☐ Build after other projects are built ?

☐ Build periodically ?

☒ Build when a change is pushed to GitLab. GitLab webhook URL: <http://j8b104.p.ssafy.io:9090/project/dodo-spring> ?

Enabled GitLab triggers

☒ Push Events

☐ Push Events in case of branch delete

☐ Opened Merge Request Events

☒ Build only if new commits were pushed to Merge Request ?

☐ Accepted Merge Request Events

☐ Closed Merge Request Events

Rebuild open Merge Requests

Never

☒ Approved Merge Requests (EE-only)

☒ Comments

Comment (regex) for triggering a build ?

Jenkins please retry a build

고급 ▾ Edited

3. Build Trigger의 고급 설정

- Secret token을 생성하고 생성된 token으로 깃랩 웹훅 생성
- 깃랩 웹훅 설정의 Secret token에 a 단계에서 생성된 token 입력
- 깃랩 웹훅 설정의 URL에 <http://j8b104.p.ssafy.io:9090/project/dodo-spring> 입력
- trigger은 Push events 선택
- add webhook

2. Pipeline 설정

- Definition: Pipeline script from SCM 선택
- SCM: Git 선택
- Repository URL: 깃랩 레포지토리 주소(<https://lab.ssafy.com/s08-bigdata-recom-sub2/S08P22B104>)
- Credentials에 Global Credentials에서 설정한 깃랩 Username with password 선택
- Branch Specifier에 spring(빌드할 브랜치) 입력
- Script Path에 be/spring/dodo/Jenkinsfile(Jenkinsfile 경로) 입력

3. EC2 서버에 SpringBoot 설정파일 저장

- EC2 서버 접속

```
ssh -i J8B104T.pem ubuntu@j8b104.p.ssafy.io
```

- 환경설정 파일 저장

```
vim ~/jenkins/be/spring/resources/application.yml

### application.yml
spring:
  profiles:
    default: product

vim ~/jenkins/be/spring/resources/application-product.yml

### application-product.yml
app:
  auth:
    jwt:
```

```

        secret-key: {jwt_secret_key}
        redirect-url: "https://j8b104.p.ssafy.io/login/"
spring:
  mvc:
    pathmatch:
      matching-strategy: ant_path_matcher
  datasource:
    url: "jdbc:mysql://{db_url}:{db_port}/{db_name}?serverTimezone=Asia/Seoul&characterEncoding=UTF-8"
    username: "{db_username}"
    password: "{db_password}"
    driver-class-name: com.mysql.cj.jdbc.Driver
  servlet:
    multipart:
      enabled: true
      max-file-size: 20MB
      max-request-size: 20MB

  jpa:
    database:
      hibernate:
        ddl_auto: none
    properties:
      hibernate:
        "globally_quoted_identifiers": "true"
        format_sql: true
        show_sql: true

    database-platform: org.hibernate.dialect.MySQL8Dialect
  security:
    oauth2:
      client:
        registration:
          kakao:
            client-id: {kakao_client_id}
            client-secret: {kakao_secret_key}
            redirect-uri: https://j8b104.p.ssafy.io/api/login/oauth2/code/kakao
            client-authentication-method: POST
            authorization-grant-type: authorization_code
            scope: account_email, profile_image
            client-name: Kakao
        provider:
          kakao:
            authorization-uri: https://kauth.kakao.com/oauth/authorize
            token-uri: https://kauth.kakao.com/oauth/token
            user-info-uri: https://kapi.kakao.com/v2/user/me
            user-name-attribute: id

  mail:
    host: smtp.gmail.com
    port: 587
    username: {google_id} # 구글 아이디
    password: {app_password} # 앱 비밀번호
    properties:
      mail:
        smtp:
          auth: true
          timeout: 3000
          starttls:
            enable: true

  redis:
    host: localhost
    port: 6379

  cloud:
    aws:
      credentials:
        accessKey: {aws_iam_access_key} # AWS IAM AccessKey 적기
        secretKey: {aws_iam_secret_key} # AWS IAM SecretKey 적기
      s3:
        bucket: dodo-walk-bucket
      region:
        static: ap-northeast-2
      stack:
        auto: false

```

6. 저장

- FastAPI 서버 배포 설정
 1. GitLab Connection: dodo 선택
 2. Build Trigger 설정

Build Triggers

☐ Build after other projects are built ?

☐ Build periodically ?

☒ Build when a change is pushed to GitLab. GitLab webhook URL: <http://j8b104.p.ssafy.io:9090/project/dodo-fastapi> ?

Enabled GitLab triggers

☒ Push Events

☐ Push Events in case of branch delete

☒ Opened Merge Request Events

☐ Build only if new commits were pushed to Merge Request ?

☐ Accepted Merge Request Events

☐ Closed Merge Request Events

Rebuild open Merge Requests

Never

☒ Approved Merge Requests (EE-only)

☒ Comments

Comment (regex) for triggering a build ?

Jenkins please retry a build

고급

3. Build Trigger의 고급 설정

- Secret token을 생성하고 생성된 token으로 깃랩 웹훅 생성
- 깃랩 웹훅 설정의 Secret token에 a 단계에서 생성된 token 입력
- 깃랩 웹훅 설정의 URL에 <http://j8b104.p.ssafy.io:9090/project/dodo-fastapi> 입력
- trigger은 Push events 선택
- add webhook

2. Pipeline 설정

- Definition: Pipeline script from SCM 선택
- SCM: Git 선택
- Repository URL: 깃랩 레포지토리 주소(<https://lab.ssafy.com/s08-bigdata-recom-sub2/S08P22B104>)
- Credentials에 Global Credentials에서 설정한 깃랩 Username with password 선택
- Branch Specifier에 spring(빌드할 브랜치) 입력
- Script Path에 be/python/fastapi/Jenkinsfile(Jenkinsfile 경로) 입력

3. EC2에 FastAPI 설정 파일 저장

- EC2 접속

```
ssh -i J8B104T.p.ssafy.io ubuntu@j8b104.p.ssafy.io
```

- 환경설정 파일 저장

```
vim ~/jenkins/be/python/resources/.env

### .env
SECRET={jwt_secret_key}
ALGORITHM=HS512
MYSQL_URL=mysql+pymysql://{username}:{user_password}@{db_url}:{db_port}/{db_name}?charset=utf8mb4
REDIS_HOST=localhost
REDIS_PORT=6379
REDIS_DATABASE=1
```

6. 저장

- React 배포 설정

1. GitLab Connection: dodo 선택

2. Build Triggers 설정

Build Triggers

☐ Build after other projects are built ?

☐ Build periodically ?

☒ Build when a change is pushed to GitLab. GitLab webhook URL: <http://j8b104.p.ssafy.io:9090/project/dodo-react> ?

Enabled GitLab triggers

☒ Push Events

☐ Push Events in case of branch delete

☒ Opened Merge Request Events

☐ Build only if new commits were pushed to Merge Request ?

☐ Accepted Merge Request Events

☐ Closed Merge Request Events

Rebuild open Merge Requests

Never

☒ Approved Merge Requests (EE-only)

☒ Comments

Comment (regex) for triggering a build ?

Jenkins please retry a build

고급

3. Build Triggers 고급 설정

- a. Secret token을 생성하고 생성된 token으로 깃랩 웹훅 생성
- b. 깃랩 웹훅 설정의 Secret token에 a 단계에서 생성된 token 입력
- c. 깃랩 웹훅 설정의 URL에 <http://j8b104.p.ssafy.io:9090/project/dodo-react> 입력
- d. trigger은 Push events 선택
- e. add webhook

4. Pipeline 설정

- a. Definition: Pipeline script 선택
- b. Script에 아래 내용 입력

```
pipeline {
  agent any
  stages {
    stage('Git Clone') {
      steps {
        git branch: 'fe', credentialsId: 'gitlab-username-password', url: 'https://lab.ssafy.com/s08-bigdata-recom-sub2/S08P22B104.git'
      }
    }
    stage('Build') {
      steps {
        dir('fe') {
          nodejs(nodeJSInstallationName: 'NodeJS 16.9.0') {
            sh 'rm -rf node_modules package-lock.json'
            sh 'npm install && CI=false npm run build'
          }
        }
      }
    }
  }
}
```

```

stage('Deploy'){
    steps{
        dir('fe/build') {
            sshagent (credentials: ['ssh-agent']) {
                sh 'ls'
                sh 'pwd'
                sh 'scp -r /var/jenkins_home/workspace/dodo-react/fe/build ubuntu@j8b104.p.ssafy.io:/home/
ubuntu/'
            }
        }
    }
}
}
}
}
}

```

c. Use Groovy Sandbox 체크

5. 저장

데이터베이스 설정

- MySQL
 - AWS RDS 인스턴스(프리티어) 생성
 - DB 엔진: `MySQL 8.0.31`
 - DB 인스턴스 식별자 입력: `dodo-db`
 - 마스터 사용자 이름, 패스워드 입력
 - 인스턴스 구성: `db.t3.micro`
 - 스토리지
 - 범용 SSD(gp2)
 - 할당된 스토리지 20
 - 연결 추가 구성에서 퍼블릭 액세스 가능 선택
 - 데이터베이스 인증: 암호 인증 선택
 - 생성
 - DB 인스턴스 VPC에 인바운드 규칙 설정
 - IP 버전: `IPv4`
 - 유형: `MySQL/Aurora`
 - 프로토콜: `TCP`
 - 포트 범위: `3306`
 - 데이터베이스 파라미터 그룹 설정
 - RDS > 파라미터 그룹에서 파라미터 그룹 생성
 - time_zone 변경 → `Asia/Seoul`
 - 아래 Character Set 변경 → `utf8mb4`
 - character_set_client
 - character_set_connection
 - character_set_database
 - character_set_filesystem
 - character_set_results
 - charcater_set_server
 - collation_connection
 - collation_server

- RDS > 데이터베이스 > dodo-db > 수정
 - 추가 구성 > 데이터베이스 옵션 > DB 파라미터 그룹을 새로 생성한 파라미터 그룹으로 변경
- 계속 > 수정 예약: 즉시 적용 선택 후 DB 인스턴스 수정
- MySQLWorkbench에서 RDS 인스턴스 접속
 - Hostname: 인스턴스의 엔드포인트
 - Port: 3306
 - Username: 마스터 사용자 이름
 - Password: 마스터 사용자 패스워드
- 접속 후 `ddl.sql`, `dump.sql` 파일 실행
- Redis
 - EC2 접속

```
ssh -i J8B104T.pem ubuntu@j8b104.p.ssafy.io
```

- Redis 설치

```
sudo apt update
sudo apt install redis-server

# redis 서비스 상태확인
sudo systemctl status redis-server
```

OpenAPI 등록

- 카카오 로그인
 1. <https://developers.kakao.com/> 로그인
 2. 애플리케이션 등록
 3. 내 애플리케이션 > 앱 설정 > 플랫폼 > Web에 사이트 도메인 등록
 - a. <https://j8b104.p.ssafy.io>
 4. 내 애플리케이션 > 제품 설정 > 카카오 로그인
 - a. 활성화 설정 `on`
 - b. Redirect URI 등록
 - <https://j8b104.p.ssafy.io/api/login/oauth2/code/kakao>
 5. 내 애플리케이션 > 제품 설정 > 카카오 로그인 > 동의항목 설정
 - a. 프로필 사진
 - b. 카카오계정(이메일)
 6. 내 애플리케이션 > 제품 설정 > 카카오 로그인 > 보안에서 Client Secret 발급
- 구글 Gmail
 1. 구글 계정 관리 > 보안 > 2단계 인증 사용에서 사용 설정 후 메일, Windows 컴퓨터를 위한 앱 비밀번호 발급
 2. Gmail 설정 > 전달 및 POP/IMAP 탭
 - POP 다운로드 - 1.상태 : **모든 메일에 POP**를 활성화 하기 선택
 - IMAP 액세스 - 상태 : IMAP 사용 선택

버전 정보

▼ Backend

• SpringBoot

◦ version

SpringBoot	2.9.7
Java	11
IDE	2022.3.2 (Ultimate Edition)
MySql	8.0.31
Redis	5.0.7

◦ 설정 파일

■ src/main/resources/application.yml

```
spring:
  profiles:
    default: product
```

■ src/main/resources/application-product.yml

```
app:
  auth:
    jwt:
      secret-key: 4C822FC199EB6C0563A8E1CE9041BB7A1ADB2D803EB2EB6F531A1D9B0B0ECAD0
      redirect-url: "https://j8b104.p.ssafy.io/login/"
spring:
  mvc:
    pathmatch:
      matching-strategy: ant_path_matcher
  datasource:
    url: "jdbc:mysql://${db_url}:${db_port}/${db_name}?serverTimezone=Asia/Seoul&characterEncoding=UTF-8"
    username: "${db_username}"
    password: "${db_password}"
    driver-class-name: com.mysql.cj.jdbc.Driver
  servlet:
    multipart:
      enabled: true
      max-file-size: 20MB
      max-request-size: 20MB

  jpa:
    database:
    hibernate:
      ddl-auto: none
    properties:
      hibernate:
        "globally_quoted_identifiers": "true"
        format_sql: true
        show_sql: true

    database-platform: org.hibernate.dialect.MySQL8Dialect
  security:
    oauth2:
      client:
        registration:
          kakao:
            client-id: {kakao_client_id}
            client-secret: {kakao_secret_key}
            redirect-uri: https://j8b104.p.ssafy.io/api/login/oauth2/code/kakao
            client-authentication-method: POST
            authorization-grant-type: authorization_code
            scope: account_email, profile_image
            client-name: Kakao
        provider:
          kakao:
            authorization-uri: https://kauth.kakao.com/oauth/authorize
            token-uri: https://kauth.kakao.com/oauth/token
            user-info-uri: https://kapi.kakao.com/v2/user/me
            user-name-attribute: id
  mail:
    host: smtp.gmail.com
    port: 587
    username: {google_id} # 구글 아이디
    password: {app_password} # 앱 비밀번호
    properties:
      mail:
```

```

smtp:
  auth: true
  timeout: 3000
  starttls:
    enable: true

redis:
  host: localhost
  port: 6379

cloud:
  aws:
    credentials:
      accessKey: {aws_iam_access_key} # AWS IAM AccessKey 적기
      secretKey: {aws_iam_secret_key} # AWS IAM SecretKey 적기
    s3:
      bucket: dodo-walk-bucket
      region:
      static: ap-northeast-2
      stack:
      auto: false

```

- FastAPI

- version

FastAPI	0.95.0
SQLAlchemy	2.0.7
uvicorn	0.21.1
Python	3.11.3
IDE	VS code
MySql	8.0.31
Redis	5.0.7

- 설정 파일

- app/.env

```

SECRET={jwt_secret_key}
ALGORITHM=HS512
MYSQL_URL=mysql+pymysql://{username}:{user_password}@{db_url}:{db_port}/{db_name}?charset=utf8mb4
REDIS_HOST=localhost
REDIS_PORT=6379
REDIS_DATABASE=1

```

▼ Frontend

- React

- version

React	18.2.0
Redux	8.0.5
axios	1.3.4
react-dom	18.2.0
node.js	16.9.0

- 설정 파일

- .prettierrc

```

{
  "printWidth": 160,
  "tabWidth": 2,
  "useTabs": false,
  "semi": true,
  "singleQuote": false,
  "trailingComma": "all",
  "bracketSpacing": true,
  "arrowParens": "avoid",

```

```
"proseWrap": "never",  
"endOfLine": "auto"  
}
```