



포팅 메뉴얼

개발 환경

Front-End

- React 18.2.0
- Node.js LTS 18.15.0
- Visual Studio code 1.77.0

Back-End

- IntelliJ
 - Build 223.8214.52, built on December 20, 2022
- Java 11
 - zulu-11 java version "11.0.18"
- Springboot 2.7.7
- MySQL 8.0.32
- Flask

ETC

- Docker 23.0.1
- Jenkins 2.397
- Nginx 1.23.4 (ubuntu)
- AWS EC2 Ubuntu 20.04 LTS
- AWS S3
- GitLab
- Webhooks

설정 파일 및 환경 변수 정보

Nginx

- Reverse Proxy Nginx

```
server {  
    location /{  
        proxy_pass http://front-end:3000;  
        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;  
    }  
  
    location /api/ {  
        proxy_pass http://back-end:8080/api/;  
        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;  
    }  
}
```

```

    }

    location /disease/ {
        proxy_pass http://disease-ai:5000/;
        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;
    }

    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/j8a801.p.ssafy.io/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/j8a801.p.ssafy.io/privkey.pem; # managed by Certbot
}

server {
    if ($host = j8a801.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

    listen 80;
    server_name j8a801.p.ssafy.io;
    return 404; # managed by Certbot
}

```

- Front-end Nginx

```

server {
    listen 3000;

    location / {
        root /home/A801/build;
        index index.html;
        try_files $uri $uri/ /index.html;
    }
}

```

React

- .env

```

REACT_APP_KAKAO_MAP_API_KEY= 카카오맵 API 키
REACT_APP_KAKAO_MAP_REST_API_KEY= 카카오맵 REST API 키

```

Spring

- application.yml

```

build:
  date: '@build.date@'

# Server setting
server:
  address: 0.0.0.0
  servlet:
    encoding:
      charset: UTF-8
      enabled: 'true'
      force: 'true'
    contextPath: /api
  port: '8080'

# AWS
cloud:
  aws:
    # AWS S3
    s3:
      bucket: moemoe
    region:
      static: ap-northeast-2
    stack:
      auto: 'false'

```

```

# AWS IAM
credentials:
  secret-key: S3 secret key
  access-key: S3 access key

# for SPA-1
spa:
  default-file: /dist/index.html

spring:
  # for SPA-2 start
  web:
    resources:
      static-locations: classpath:/dist/
      add-mappings: 'false'
  mvc:
    throw-exception-if-no-handler-found: 'true'
  # for SPA-2 end

# JPA start
jpa:
  hibernate:
    naming:
      implicit-strategy: org.springframework.boot.orm.jpa.hibernate.SpringImplicitNamingStrategy
      physical-strategy: org.springframework.boot.orm.jpa.hibernate.SpringPhysicalNamingStrategy
    properties:
      hibernate:
        dialect: org.hibernate.dialect.MySQL57Dialect
  data:
    web:
      pageable:
        one-indexed-parameters: 'true'
  datasource:
    url: jdbc:mysql://서버 도메인:3306/nyang?useUnicode=yes&characterEncoding=UTF-8
    hikari:
      username: DB 사용자 이름
      password: 비밀번호
      driver-class-name: com.mysql.cj.jdbc.Driver
  # JPA end

# Multipart start
servlet:
  multipart:
    max-request-size: 100MB
    max-file-size: 100MB
  # Multipart end

# page auto load
devtools:
  livereload:
    enabled: 'true'

#login
logging:
  level:
    org:
      springframework:
        boot: DEBUG
        security: DEBUG
      apache:
        tiles: INFO
      springframework:
        web: DEBUG
    com:
      samsung:
        security: DEBUG
      amazonaws:
        util:
          EC2MetadataUtils: ERROR
    root: INFO
  file:
    name: ./ssafy-web.log

# gzip compression start
compression:
  mime-types: application/json,application/xml,text/html,text/xml,text/plain,application/javascript,text/css
  enabled: 'true'
  # gzip compression end

# for health check
management:
  health:

```

```

db:
  enabled: 'true'
diskspace:
  enabled: 'true'
default:
  enabled: 'true'
servlet:
  context-path: /manage

springboot:
  jwt:
    secret: jwt secret key

# Swagger
springfox:
  documentation:
    swagger:
      use-model-v3: 'false'

```

서버 설정 및 사전 작업

서버 타임존 설정

KST로 변경

```
timedatectl set-timezone Asia/Seoul
```

변경된 시간 확인

```
date
```

방화벽 설정

- SSH 포트 오픈 설정

```
sudo ufw allow ssh
```

- 방화벽 활성화

```
sudo ufw enable
```

도커 설치

1. 패키지 관리자 업데이트

```
sudo apt-get update
```

2. 필요한 패키지 설치

```

sudo apt-get install \
  apt-transport-https \
  ca-certificates \
  curl \
  gnupg \
  lsb-release

```

3. Docker GPG 키 추가

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
```

4. Docker 레포지토리 추가

```
echo \  
"deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu \  
$(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

5. 패키지 관리자 업데이트

```
sudo apt-get update
```

6. Docker 설치

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

7. Docker 실행

```
sudo systemctl start docker
```

8. Docker Group 설정

```
sudo usermod -aG docker 유저이름  
  
# 그룹 수정 후 도커 재시작 해야함  
sudo systemctl docker restart  
  
# 서버에 로그인도 다시 해야함  
# ⇒ 서버 세션 종료 후 다시 로그인 하기
```

Certbot - SSL 인증서 발급

인증서 저장할 볼륨 생성

```
sudo docker volume create nginx-certs
```

Dockerfile.certbot 작성

```
FROM certbot/certbot  
  
RUN which certbot  
  
# Certbot 인증서 발급 스크립트를 실행  
ENTRYPOINT ["/usr/local/bin/certbot", "certonly", "--standalone", "--non-interactive", "--agree-tos", "-d", "j8a801.p.ssafy.io"]
```

이미지 빌드

Dockerfile이 있는 디렉토리에서

```
sudo docker build -t certbot .
```

⇒ certbot이라는 이미지 생성한 것

여러개의 도커파일이 있다면

```
sudo docker build -t certbot -f DockerFile.certbot
```

과 같이 -f 속성을 통해 특정할 수 있다.

컨테이너 실행

```
sudo docker run -p 80:80 -v nginx-certs:/etc/letsencrypt certbot
```

```
ubuntu@:~/test$ sudo docker run -p 80:80 -v nginx-certs:/etc/letsencrypt certbot
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Requesting a certificate for 서버 도메인

Successfully received certificate.
Certificate is saved at: /etc/letsencrypt/live/서버 도메인/fullchain.pem
Key is saved at: /etc/letsencrypt/live/서버 도메인/privkey.pem
This certificate expires on 2023-06-17.
These files will be updated when the certificate renews.
NEXT STEPS:
- The certificate will need to be renewed before it expires. Certbot can automatically renew the certificate in the background, but you may need to take steps to enable that functionality. See https://certbot.org/renewal-setup for instructions.

-----
If you like Certbot, please consider supporting our work by:
* Donating to ISRG / Let's Encrypt: https://letsencrypt.org/donate
* Donating to EFF: https://eff.org/donate-le
```

도커 볼륨 접근권한 변경

도커 볼륨 디렉토리(/var/lib/docker)에 접근권한이 root(소유자):root(그룹)인 경우

개발자를 docker를 사용할 수 있는 그룹에 넣고

chown으로 그룹만을 docker그룹으로 바꿔서 접근가능하게 해줘야 한다.

chmod로 그룹에 대한 권한도 별도로 지정해야한다.

chown -R 명령어로 특정 디렉토리 하위의 모든 디렉토리 및 폴더에도 동시에 적용 가능하다.

서버 설정파일 디렉토리 구조

```

Ubuntu
├── A801
│   └── pre
│       ├── properties
│       │   ├── sql
│       │   │   ├── moemoe.sql          # 더미 데이터
│       │   │   └── r_proxy
│       │   └── default.conf            # Nginx 리버스 프록시 conf
│       └── model                      # 질병분석 모델들
│           ├── segmentation
│           │   ├── A6_model.h5
│           │   ├── A5_model.h5
│           │   ├── A4_model.h5
│           │   ├── A3_model.h5
│           │   ├── A2_model.h5
│           │   └── A1_model.h5
│           └── classification
│               └── model_A1_A2_A3_A4_A5_A6.h5
├── front.conf                        # Nginx 프론트엔드 conf
├── application.yml                  # Springboot 설정파일
├── .env                            # 프론트엔드 .env
├── mysql                          # mysql 컨테이너 마운트 포인트
├── jenkins                        # jenkins 컨테이너 마운트 포인트
├── Dockerfile.jenkins             # jenkins 도커파일
├── Dockerfile.certbot             # certbot 도커파일
├── docker-compose.yml             # jenkins,mysql용 도커 컴포즈 파일
├── .env                            # 도커 컴포즈 환경변수 .env
└── docker-compose.yml            # 배포용 도커 컴포즈 파일(미리 생성할 필요 없음)
    => 젠킨스에서 빌드 완료 후 전송된다.
  
```

배포를 위한 Dockerfile 작성

Dockerfile.jenkins

```

FROM jenkins/jenkins:latest
ARG HOST_GID
USER root

# 도커 설치
RUN apt-get update && \
    apt-get install -y apt-transport-https \
  
```

```

        ca-certificates \
        curl \
        gnupg2 \
        software-properties-common && \
    curl -fsSL https://download.docker.com/linux/debian/gpg | apt-key add - && \
    add-apt-repository \
        "deb [arch=amd64] https://download.docker.com/linux/debian \
        $(lsb_release -cs) \
        stable" && \
    apt-get update && \
    apt-get install -y docker-ce docker-ce-cli containerd.io && \
    rm -rf /var/lib/apt/lists/*

# 설정파일 디렉토리 생성
RUN mkdir /var/jenkins_home/properties

# 배포 디렉토리 생성
RUN mkdir /var/jenkins_home/deploy

# 도커 실행 권한 부여
RUN groupadd -g $HOST_GID docker-sock && \
    usermod -aG docker-sock jenkins

USER jenkins

```

Front-end

```

# 베이스 이미지
FROM node:18.15.0 as builder

# 깃에서 가져온 소스 코드 이미지 내부 경로로 복사
COPY . /home/A801

COPY .env /home/A801/.env

# 이미지 내부 작업공간 설정
WORKDIR /home/A801

# 앱 의존성 설치
COPY package*.json yarn.lock ./

RUN yarn install

# 앱 빌드
RUN yarn run build

# nginx 이미지
FROM nginx:alpine

# react 빌드파일 외부와 마운트된 디렉토리로 복사
COPY --from=builder /home/A801/build /home/A801/build

COPY front.conf /etc/nginx/conf.d/default.conf

# 컨테이너 시작 시 nginx 실행
CMD ["nginx", "-g", "daemon off;"]

```

Back-end

```

FROM azul/zulu-openjdk:11

# 깃에서 가져온 소스 코드 이미지 내부 경로로 복사
COPY . /home/A801

# 이미지 내부 작업 공간 설정
WORKDIR /home/A801

# gradle 빌드
RUN chmod +x gradlew
RUN ./gradlew clean build

RUN mv ./build/libs/moemoe-0.0.1-SNAPSHOT.jar ./nyang.jar

CMD ["nohup", "java", "-jar", "nyang.jar", "&"]

```

AI

```
FROM python:3.8-slim-buster

COPY . /home/A801

# 작업 디렉토리 생성
WORKDIR /home/A801

# 필요한 패키지 설치
RUN pip install -r requirements.txt

RUN apt-get update && apt-get install -y libgl1-mesa-glx libgl1-mesa-dev

# 컨테이너 시작 시 실행할 명령어
CMD [ "python", "app.py" ]
```

Docker-compose 파일 작성

리버스 프록시, 프론트, 백, AI 서버 docker-compose.yml

```
version: '3'

services:
  nginx:
    image: nginx:latest
    container_name: r_proxy
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - ../pre/properties/r_proxy:/etc/nginx/conf.d/
      - nginx-certs:/etc/letsencrypt
    depends_on:
      - front-end
      - back-end
    restart: always

  front-end:
    image: nyang_front:latest
    container_name: nyang_front
    ports:
      - "3000:3000"
    restart: always

  back-end:
    image: nyang_back:latest
    container_name: nyang_back
    ports:
      - "8080:8080"
    restart: always

  disease-ai:
    image: nyang_disease_ai:latest
    container_name: nyang_disease_ai
    ports:
      - "5000:5000"
    restart: always

volumes:
  nginx-certs:
    external: true
```

도커 그룹 지정을 위한 .env 파일

젠킨스에서 Dood방식을 사용하기 위하여 젠킨스 컨테이너의 사용자를 docker.sock파일의 그룹에 넣기 위해 .env파일을 docker-compose.yml파일과 같은 경로에 넣어주어야 한다.

```
# docker.sock의 그룹 아이디를 가져와 HOST_GID에 저장한다.
HOST_GID=$(stat -c '%g' /var/run/docker.sock)
```

젠킨스, mysql docker-compose.yml

```
version: '3'

services:
  mysql:
    image: mysql:8.0.32
    volumes:
      - ./mysql:/var/lib/mysql
    ports:
      - "3306:3306"
    environment:
      - MYSQL_ROOT_PASSWORD= MYSQL_ROOT 계정 비밀번호
    restart: always

  jenkins:
    build:
      context: .
      args:
        HOST_GID: ${HOST_GID}
      dockerfile: Dockerfile.jenkins
    ports:
      - "9090:8080"
    volumes:
      - ./jenkins:/var/jenkins_home
      - ./properties:/var/jenkins_home/properties
      - /var/run/docker.sock:/var/run/docker.sock
    restart: always
```

MySQL, Jenkins 컨테이너 실행 및 설정

서버 설정파일 디렉토리 구조를 참고하여

mysql 디렉토리를 생성하고, .sql파일을 업로드한다.

(docker-compose.yml 파일이 있는 위치에서) mysql, jenkins용 docker-compose 파일을 실행시킨다.

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

```
920d1f781c33  pre_jenkins  "/usr/bin/tini -- /u..." 42 hours ago Up 42 hours 50000/tcp, 0
0.0.0.0:9090→8080/tcp, :::9090→8080/tcp pre_jenkins_1
63f931f24530  mysql:8.0.32 "docker-entrypoint.s..." 2 weeks ago Up 2 weeks 0.0.0.0:3306
→3306/tcp, :::3306→3306/tcp, 33060/tcp test_db_1
ubuntu@ip-172-26-4-169:~$
```

```
docker ps
```

 명령어로 MySQL 컨테이너의 이름을 확인한다

MySQL 설정

MySQL 컨테이너 내부로 이동

```
docker exec -it MySQL컨테이너명 /bin/bash
```

.sql 파일 실행

```
source /소스파일경로/소스파일.sql
```

(docker-compose.yml 파일에서 /var/lib/mysql 디렉토리에 마운트 했기 때문에 해당 경로의 소스파일을 실행시킨다.) ⇒

```
/var/lib/mysql/moemoe.sql
```

`use 스키마명` 명령어 입력한다.

사용자 생성

```
create user '사용자 이름'@'%' identified by '비밀번호';
```

권한 설정

```
grant all privileges on *.* to '사용자 이름'@'%';
```

변경 사항 적용

```
flush privileges;
```

Jenkins 설정

```
docker-compose exec jenkins cat /var/jenkins_home/secrets/initialAdminPassword
```

명령어로 실행중인 컨테이너 쉘에 접속하여 jenkins 접속 비밀번호를 알아낸 뒤, jenkins가 실행중인 주소로 접속하여 위 비밀번호를 입력한다.

Jenkins 플러그인 설치

기본 플러그인 설치 상태에서





gitlab과 git plugin이 설치되어있는지 확인하고 안되어있으면 설치한다.

publish over ssh 플러그인도 설치한다.


Jenkins Credentials 설정

Dashboard > Jenkins 관리 > Credentials

Credentials

| T | P | Store ↓ | Domain | ID | Name |
|---|---|---------|----------|--------------------------------------|-----------------------------|
|  |  | System | (global) | 9978f8da-b78e-4a66-aaeb-5a3fc5eb2196 | tmdgus908@naver.com/***** |
|  |  | System | (global) | test-project | GitLab API token (테스트 프로젝트) |

Stores scoped to Jenkins

| P | Store ↓ | Domains |
|---|---------|----------|
|  | System | (global) |

Jenkins System 설정

Jenkins관리 - System

Jenkins Location

Jenkins URL ?

[REDACTED]

System Admin e-mail address ?

address not configured yet <nobody@nowhere>

jenkins 컨테이너 URL 입력

Gitlab 설정

Gitlab

☒ Enable authentication for '/project' end-point

GitLab connections

Connection name

A name for the connection

tmdgus908

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 (nyang)

Add

그룹

Success

Test Connection

하단의 Test Connection 눌렀을 때 Success가 나오면 성공

public over ssh 설정

호스트 서버에서 ssh키를 생성한다.

```
ssh-keygen -t rsa -m pem
```

`cat id_rsa.pub` 명령어로 내용을 출력후 복사한 뒤,

배포할 원격 서버의 authorized_keys 파일에 내용을 추가해준다.

⇒ 이 프로젝트에서는 하나의 서버만 사용하므로 호스트 서버의 authorized_keys 파일에 내용을 추가한다.

Publish over SSH

Jenkins SSH Key ?

Passphrase ?

 Concealed

Path to key ?

Key ?

```
-----BEGIN RSA PRIVATE KEY-----
MIIG4wIBAAKCAIEA7NBTv9308iKtdQNZVNaHzhHjzARfjAhA5lrAlAvRyVQAJdNg
jBUuqjMok3W6PGM6HbcsLkcC5ns/Pjo6ra2OHeGl4/QLshTVokuJmT9q7UISkemi
```

Key 항목에

id_rsa 파일의 내용을 붙여넣기한다.

SSH Servers를 설정

SSH Servers

SSH Server

Name ?

Hostname ?

Username ?

Remote Directory ?

고급 ▾

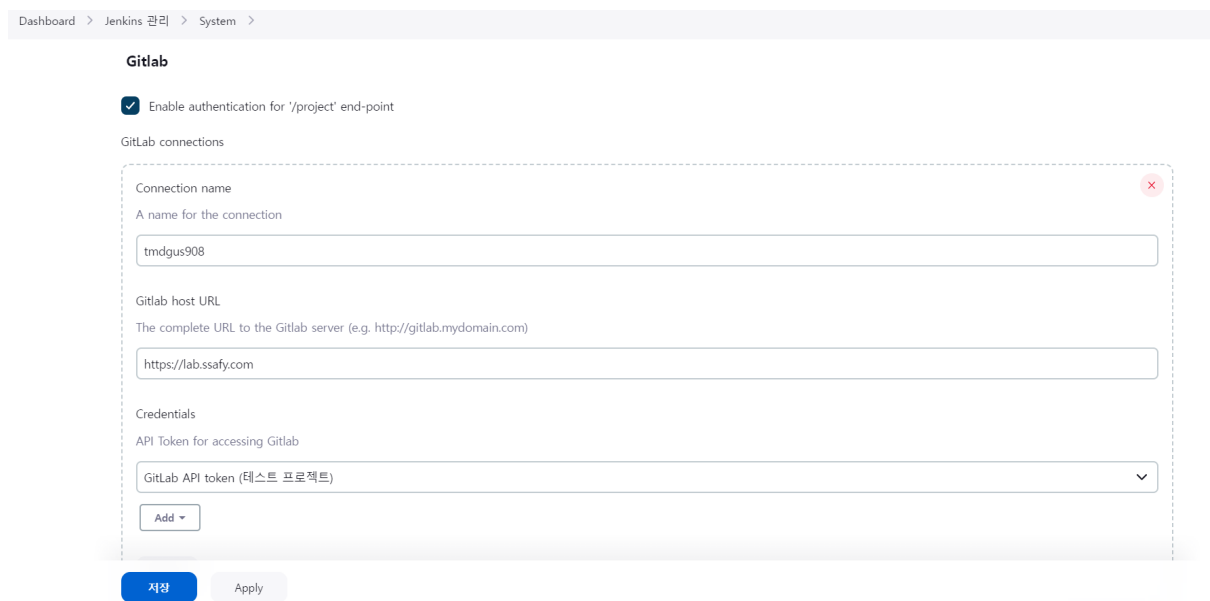
Success

Test Configuration

Test Configuration 눌렀을 때 Success가 나오면 성공

gitlab-jenkins 연동

Gitlab Access token 발급



Freestyle project로 생성한다.

Dashboard > test > Configuration

Configure

General

- 소스 코드 관리
- 빌드 유발
- 빌드 환경
- Build Steps
- 빌드 후 조치

General

Enabled

설명

[Plain text] [미리보기](#)

☐ GitHub project

GitLab Connection

tmdgus908

☐ Use alternative credential

☐ Throttle builds ?

☐ 오래된 빌드 삭제 ?

[저장](#) [Apply](#)

GitLab Connection에 위에서 지정했던 임의의 이름으로 설정

소스코드 관리

소스 코드 관리

☐ None

☒ Git ?

Repositories ?

Repository URL ?

[https://lab.ssafy.com/tmdgus908/web-hook_test_project](#)

Credentials ?

tmdgus908@naver.com/*****

[Add](#)

[고급](#)

[Add Repository](#)

Branches to build ?

[저장](#) [Apply](#)

깃랩 레포지토리 url과 credentials에 등록했던 깃랩 아이디와 패스워드를 사용

빌드할 브랜치 지정

Branches to build ?

Branch Specifier (blank for 'any') ?

*/develop

Add Branch

Repository browser ?

(자동)

Additional Behaviours

Add ▾

빌드 유발

빌드 유발

- ☐ 빌드를 원격으로 유발 (예: 스크립트 사용) ?
- ☐ Build after other projects are built ?
- ☐ Build periodically ?
- ☒ Build when a change is pushed to GitLab. GitLab webhook URL: <http://j8a801.p.ssafy.io:8081/project/test> ?

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

저장

Apply

push와 merge가 accept 됐을때 이벤트를 감지한다

Secret token ?

Generate

Clear

- ☐ GitHub hook trigger for GITScm polling ?
- ☐ Poll SCM ?

빌드 환경

- ☐ Delete workspace before build starts

고급 탭의 맨 밑에 있는 Secret token Generate

셸 스크립트 작성

Build Steps

Execute shell ?

Command

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

```
cd back-end
cp /var/jenkins_home/properties/application.yml src/main/resources/application.yml && \

IMAGE_NAME="nyang_back"
IMAGE_TAG="latest"

# Build the new image
docker build -t $IMAGE_NAME:$IMAGE_TAG .

cd ../front-end
cp /var/jenkins_home/properties/.env ../.env
cp /var/jenkins_home/properties/front.conf ../front.conf

IMAGE_NAME="nyang_front"

# Build the new image
docker build -t $IMAGE_NAME:$IMAGE_TAG .

cd ../ai
cp -r /var/jenkins_home/properties/model ../model

IMAGE_NAME="nyang_disease_ai"

# Build the new image
docker build -t $IMAGE_NAME:$IMAGE_TAG .
```

```
cd back-end
cp /var/jenkins_home/properties/application.yml src/main/resources/application.yml && \

IMAGE_NAME="nyang_back"
IMAGE_TAG="latest"

# Build the new image
docker build -t $IMAGE_NAME:$IMAGE_TAG .

cd ../front-end
cp /var/jenkins_home/properties/.env ../.env
cp /var/jenkins_home/properties/front.conf ../front.conf

IMAGE_NAME="nyang_front"
```



```
# Build the new image
docker build -t $IMAGE_NAME:$IMAGE_TAG .

cd ../ai
cp -r /var/jenkins_home/properties/model ./model

IMAGE_NAME="nyang_disease_ai"

# Build the new image
docker build -t $IMAGE_NAME:$IMAGE_TAG .
```

프로젝트 빌드 후 조치

Send build artifacts over SSH ?

SSH Publishers

SSH Server

Name ?

a801

Transfers

Transfer Set

Source files ?

docker-compose.yml

Remove prefix ?

Remote directory ?

Exec command ?

cd /home/ubuntu/A801
docker-compose down
docker-compose up -d

All of the transfer fields (except for Exec timeout) support substitution of [Jenkins environment variables](#)

publish over ssh에 대한 설정을 위와 같이 한다

Webhook 설정

Search page

Webhook

Webhooks enable you to send notifications to web applications in response to events in a group or project. We recommend using an [integration](#) in preference to a webhook.

URL

 http://j8a801.p.ssafy.io:9090/project/A801

URL must be percent-encoded if it contains one or more special characters.

Secret token

Used to validate received payloads. Sent with the request in the `X-Gitlab-Token` HTTP header.

Trigger

☒ Push events

 develop

Push to the repository.

☐ Tag push events

A new tag is pushed to the repository.

☐ Comments

A comment is added to an issue or merge request.

☐ Confidential comments

A comment is added to a confidential issue.

☐ Issues events

An issue is created, updated, closed, or reopened.

☐ Confidential issues events

A confidential issue is created, updated, closed, or reopened.

☒ Merge request events

A merge request is created, updated, or merged.

☐ Job events

A job's status changes.

빌드유발에서 나왔던 GitLab webhook URL을 붙여넣고,

Secret token 또한 빌드유발의 고급 탭에서 발급받았던 Secret token 값을 넣는다.

웹 훅 이벤트를 설정 완료 후 하단에서 test탭을 눌러 push event를 선택했을때 200 코드가 뜨면 성공적으로 젠킨스에 이벤트를 전달했다는것을 의미한다

젠킨스에서는 이를 감지하여 작성된 빌드 쉘 스크립트에 따라 빌드를 진행하고 빌드 완료 후에는 publish over ssh 플러그인을 사용하여 원격서버(호스트 서버)에 배포용 docker-compse.yml파일을 전송하고, 원격서버(호스트 서버) 에서는 실행중인 컨테이너들을 종료, 삭제 후 전송받은 docker-compose.yml 파일로 새로운 컨테이너들을 실행한다.

외부 서비스 정보

Amazon S3

Amazon S3 > 버킷 > moemoe

moemoe Info

퍼블릭 액세스 가능

객체 | 속성 | 권한 | 지표 | 관리 | 액세스 지점

객체 (4)

객체는 Amazon S3에 저장되어 있는 기본 엔티티입니다. Amazon S3 [인벤토리](#)를 사용하여 버킷에 있는 모든 객체의 목록을 얻을 수 있습니다. 다른 사용자가 객체에 액세스할 수 있게 하려면 명시적으로 권한을 부여해야 합니다. [자세히 알아보기](#)

🔍 접두사로 객체 찾기

| <input type="checkbox"/> | 이름 ▲ | 유형 ▼ | 마지막 수정 ▼ | 크기 ▼ | 스토리지 클래스 ▼ |
|--------------------------|-------------|------|----------|------|------------|
| <input type="checkbox"/> | 📁 board/ | 폴더 | - | - | - |
| <input type="checkbox"/> | 📁 cat/ | 폴더 | - | - | - |
| <input type="checkbox"/> | 📁 disease/ | 폴더 | - | - | - |
| <input type="checkbox"/> | 📁 feedspot/ | 폴더 | - | - | - |

Kakao Map API

Web

| | |
|---------|--|
| 사이트 도메인 | https://j8a801.p.ssafy.io http://localhost:3000 |
|---------|--|

• 카카오 로그인 사용 시 Redirect URI를 등록해야 합니다. [등록하러 가기](#)