

포팅 메뉴얼

개발 환경

Application

- Flutter 3.10.0-3.1.pre.27
- Dart 3.0.0

Front-end

- Node.js 16.18.1
- react 18.2.0
- vite 4.3.6

Back-End

- JDK Azul Zulu Community 11.0.18
- Springboot 2.7.10
- MariaDB 10.11.2
- MongoDB 6.0.5
- Redis 7.0.11
- Python(FastAPI) 3.9.16

ETC

- Nginx 1.23.4
- Docker 23.0.4
- Jenkins 2.401
- AWS S3
- Firebase
- gitlab

서버 설정 및 사전 작업

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", "서버 호스트"]
```

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/r
enewal-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.

설정 파일 및 환경 변수 정보

서버 설정파일 디렉토리 구조

```
ubuntu
├── A605
│   ├── settings
│   │   ├── jenkins # jenkins 마운트 디렉토리
│   │   ├── maria # mariaDB 마운트 디렉토리
│   │   └── mongo # mongoDB 마운트 디렉토리
│   ├── properties
│   │   ├── .env # S3 .env
│   │   ├── application.yml # Springboot 설정 파일
│   │   └── firebasekey.json # firebase 키 파일
│   ├── r_proxy
│   │   └── nginx.all.conf # Springboot, FastAPI
│   ├── redis
│   │   └── data # redis 마운트 디렉토리
│   ├── .env # docker-compose에서 사용할 .env파일(docker.sock)
│   ├── docker-compose.yml # nginx, DB, jenkins docker-compose
│   ├── Dockerfile.jenkins # jenkins Dockerfile
│   ├── ninucco_front
│   │   └── docker-comopse.yml # react vite
│   ├── ninucco_back
│   │   ├── deploy.sh # blue-green 무중단 배포 쉘 스크립트
│   │   ├── docker-comopse.blue.yml # blue 컨테이너 docker-compose.yml
│   │   └── docker-comopse.green.yml # green 컨테이너 docker-compose.yml
│   ├── ninucco_model
│   │   ├── deploy.sh # blue-green 무중단 배포 쉘 스크립트
│   │   ├── docker-comopse.blue.yml # blue 컨테이너 docker-compose.yml
│   │   └── docker-comopse.green.yml # green 컨테이너 docker-compose.yml
```

Nginx

- Reverse Proxy Nginx

```
# 1: blue 2: green

upstream model {
    server 서버 호스트:포트번호;
}
upstream back-end {
    server 서버 호스트:포트번호;
}

server {

    location / {
        proxy_pass 서버 호스트:포트번호;
        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/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 /predict {
        proxy_pass http://model/predict;
        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/서버 호스트/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/서버 호스트/privkey.pem; # managed by Certbot
    client_max_body_size 5M;
}

server {
    if ($host = 서버 호스트) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

    listen 80;
    server_name 서버 호스트;
    return 404; # managed by Certbot
}
```

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
  session:
    timeout: 5m
  port: '포트번호'
  tomcat:
    connection-timeout: -1

# 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'
    pathmatch:
      matching-strategy: ant_path_matcher
  # for SPA-2 end

  # JPA start
  jpa:
    generate-ddl : true
    hibernate:
      naming:
        implicit-strategy: org.springframework.boot.orm.jpa.hibernate.SpringImplicitNamingStrategy
        physical-strategy: org.springframework.boot.orm.jpa.hibernate.SpringPhysicalNamingStrategy
      ddl-auto: update
    show-sql: false
    properties:
      hibernate:
        format_sql: true
        dialect: org.hibernate.dialect.MySQL57Dialect

  data:
    web:
      pageable:
        one-indexed-parameters: 'true'
    mongodb:
      host: 서버 호스트
      port: 포트번호
      database: DB 이름
      username: 이름
      password: 비밀번호
    datasource:
      url: jdbc:mysql://서버호스트:포트번호/DB 이름?useUnicode=yes&characterEncoding=UTF-8
      hikari:
        username: 이름
        password: 비밀번호
      # driver-class-name: com.mysql.cj.jdbc.Driver
  # JPA end

  # Redis start
  redis:
    host: 서버 호스트
    port: 포트번호
    password: 비밀번호

  # Redis end

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

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

  #logging
  #logging:
  #  level:
  #    com:
  #      hibernate:
  #        SQL: DEBUG
  #        type:
  #          descriptor:
  #            sql:
  #              BasicBinder: TRACE
  amazonaws:
    util:
      EC2MetadataUtils: ERROR
  root: INFO
  file:
    name: ./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'

# S3
cloud:
  aws:
    region:
      static: ap-northeast-2
    stack:
      auto: false
    credentials:
      access-key: S3 ACCESS-KEY
      secret-key: S3 SECRET-KEY
    s3:
      bucket: S3 버킷 이름
      endPoint: S3 엔드포인트
ai:
  stability:
    url: stability url
    key: stability api key
  model:
    url: AI Model 서버 url

```

Dockerfile

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:16.18.1

# 이미지 내부 작업공간 설정
WORKDIR /home/A605

COPY ./package.json .

RUN npm install

COPY . .

CMD ["npm", "run", "dev"]
```

Back-end

```
FROM azul/zulu-openjdk:11

# 깃에서 가져온 소스 코드 이미지 내부 경로로 복사
COPY . /home/A605

# 이미지 내부 작업 공간 설정
WORKDIR /home/A605

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

RUN mv ./build/libs/ninucco-0.0.1-SNAPSHOT.jar ./ninucco.jar

CMD ["nohup", "java", "-jar", "ninucco.jar", "&"]
```

Model

```
FROM python:3.9

WORKDIR /model

COPY . /model

RUN pip install --upgrade pip
RUN pip install tensorflow
RUN pip install tensorflow-estimator
RUN pip install tensorflow-intel
RUN pip install tensorflow-io-gcs-filesystem

RUN pip install --no-cache-dir --upgrade -r /model/requirements.txt

CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "포트번호"]
```

Docker-compose 파일

리버스 프록시, mariaDB, mongoDB, redis, jenkins

```
version: '3'

services:
  nginx:
    image: nginx:latest
    container_name: r_proxy
    ports:
      - "80:80"
      - "443:443"
```

```

volumes:
  - nginx-certs:/etc/letsencrypt
restart: always
maria:
  image: mariadb:latest
  volumes:
    - ./maria:/var/lib/mysql
  ports:
    - "외부포트:내부포트"
  environment:
    - MYSQL_ROOT_PASSWORD=비밀번호
  restart: always
mongo:
  image: mongo:latest
  volumes:
    - ./mongo:/data/db
  ports:
    - "외부포트:내부포트"
  restart: always
redis:
  image: redis:latest
  volumes:
    - ./redis/data:/data
    - ./redis/redis.conf:/usr/local/etc/redis/redis.conf
  ports:
    - "외부포트:내부포트"
  command: redis-server --requirepass 비밀번호
  restart: always
jenkins:
  build:
    context: .
    dockerfile: Dockerfile.jenkins
  env_file:
    - ./env
  ports:
    - "외부포트:내부포트"
  environment:
    - TZ=Asia/Seoul
  volumes:
    - ./jenkins:/var/jenkins_home
    - ./properties:/var/jenkins_home/properties
    - /var/run/docker.sock:/var/run/docker.sock
  restart: always

volumes:
  nginx-certs:
    external: true

```

초기 구성시에는 `docker cp ./r_proxy/nginx.all.conf r_proxy:/etc/nginx/conf.d/default.conf` 명령어로 .conf파일을 넣어 준다.

이후에는 무중단 배포가 되면서 스스로 변동사항을 반영한다.

프론트엔드 docker-compose.yml

```

version: '3'

services:
  front-end:
    image: ninucco_front:latest
    container_name: ninucco_front
    ports:
      - "외부포트:내부포트"
    restart: always

```

백엔드 docker-compose.blue.yml

```

version: '3'

services:
  back-end:
    image: ninucco_back:latest
    container_name: ninucco_back_blue
    ports:
      - "외부포트:내부포트"
    restart: always

```


백엔드 docker-compose.green.yml

```
version: '3'

services:
  back-end:
    image: ninucco_back:latest
    container_name: ninucco_back_green
    ports:
      - "외부포트:내부포트"
    restart: always
```

MariaDB 설정

```
cd /home/ubuntu/A605/settings
```

```
mkdir maria
```

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

MariaDB 컨테이너 내부로 이동

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

MariaDB 접속

```
mysql -u root -p
```

docker-compose.yml에서 지정해주었던 비밀번호를 입력한다.

Database 생성

```
create database ninucco;
```

사용자 생성

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

권한 설정

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

변경 사항 적용

```
flush privileges;
```

dump.sql 파일 실행

```
source sql파일 경로
```

MongoDB 설정

MongoDB 컨테이너 내부로 이동

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

MongoDB 접속

```
mongosh
```

Database 생성

```
use DB이름
```

컬렉션 생성

```
db.createCollection("testCollection");
```

유저 생성 ⇒ 권한부여

```
db.createUser(  
  {  
    user: 유저이름,  
    pwd: 비밀번호,  
    roles [ { role: 'readWrite', db: 'a605' } ]  
  }  
)
```

dump 파일 import

```
mongorestore --db db이름 /path/to/your/파일.bson
```

Redis 설정

Redis 컨테이너 내부로 이동

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

redis-cli 접속

```
redis-cli --raw
```

raw를 옵션으로 지정하면 한글 인코딩이 깨지지 않은 실제 저장했던 데이터를 볼 수 있다.

로그인

```
auth 비밀번호
```

docker-compose.yml 에서 --requirepass 옵션으로 설정해주었던 비밀번호를 입력한다.

redis-csv.csv 파일로 데이터 저장

```
cat redis-csv.csv | awk -F',' '{print " SET \""$1\"" \""$2\"" \n"}' | redis-cli --pipe -a 비밀번호
```

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

Credentials

T	P	Store ↓	Domain	ID	Name
		System	(global)	a605	GitLab API token (gitlab project)
		System	(global)	S2Hyeon	[REDACTED]

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

고급

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.

Dashboard > Jenkins 관리 > System >

Publish over SSH

Jenkins SSH Key ?

Passphrase ?

 Concealed

Path to key ?

Key ?

```
-----BEGIN RSA PRIVATE KEY-----
MIIG4wIBAAKCAYEA7NBTv9308iKtdQNzVNaHzhHjzARfjAhA5lrAlAvRyVQAJdNg
jBUuqjMok3W6PGM6HbcsLkcC5ns/Pjo6ra2OHeGl4/QLshTVokuJmT9q7Ulskemi
```

Key 항목에

id_rsa 파일의 내용을 붙여넣기한다.

SSH Servers를 설정

SSH Servers

SSH Server

Name ?

a605

Hostname ?

Username ?

Remote Directory ?

/home/ubuntu/A605

고급 ▾

추가

고급 ▾

Test Configuration 눌렀을 때 Success가 나오면 성공

gitlab-jenkins 연동

Gitlab Access token 발급

User Settings

Profile

Account

Applications

Chat

Access Tokens

Emails

Password

Notifications

SSH Keys

GPG Keys

Preferences

Active Sessions

Authentication log

Scopes set the permission levels granted to the token. [Learn more.](#)

☐ api

Grants complete read/write access to the API, including all groups and projects, the registry, and the package registry.

☐ read_api

Grants read access to the API, including all groups and projects, the container registry, and the package registry.

☐ read_user

Grants read-only access to the authenticated user's profile through the /user API. Includes username, public email, and full name. Also grants access to read-only API under /users.

☐ read_repository

Grants read-only access to repositories on private projects using Git-over-HTTP or the Repository Files API.

☐ write_repository

Grants read-write access to repositories on private projects using Git-over-HTTP (not using the API).

Create personal access token

Active personal access tokens (1)

Token name	Scopes	Created	Last Used ⓘ	Expires	Action
test	api, read_api, read_user, read_repository	21 Mar, 2023	3 hours ago	in 4 weeks	

송승현

@tmdgus908

Set status

Edit profile

Preferences

Sign out

Dashboard > Jenkins 관리 > System >

Gitlab

☒ Enable authentication for '/project' end-point

GitLab connections

Connection name ✕

A name for the connection

Gitlab host URL

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

Credentials

API Token for accessing Gitlab

아이템 생성 및 설정
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

☐ Use alternative credential

☐ Throttle builds [?](#)

☐ 오래된 빌드 삭제 [?](#)

GitLab Connection에 위에서 지정했던 임의의 이름으로 설정

소스코드 관리

소스 코드 관리

☐ None

☒ Git ?

Repositories ?

Repository URL ?

https://lab.ssfy.com/s08-final/S08P31A605.git

Credentials ?

[Redacted]

Add ▾

고급 ▾

깃랩 레포지토리 url과 credentials에 등록했던 깃랩 아이디와 패스워드를 사용

빌드할 브랜치 지정

Branches to build ?

Branch Specifier (blank for 'any') ?

refs/remotes/origin/backend

Add Branch

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://k8a605.p.ssafy.io:9090/project/A605> ?

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

merge가 accept 됐을때 이벤트를 감지한다

Secret token ?

[Redacted 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 Generage 후 복사해둔다.

Build Steps - Execute Shell

프론트엔드 빌드 쉘, 빌드 후 조치(Public Over SSH)

```
cd frontend/ninucco
docker build -t ninucco_front:latest .
```

Send build artifacts over SSH ?

SSH Publishers

SSH Server

Name ?

a605

고급

Edited

Transfers

Transfer Set

Source files ?

frontend/ninucco/docker-compose.yml

Remove prefix ?

frontend/ninucco/

Remote directory ?

ninucco_front

Exec command ?

cd A605/ninucco_front
docker-compose down
docker-compose up -d

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

백엔드 빌드 쉘, 빌드 후 조치(Public Over SSH)

```
cd backend/ninucco
cp /var/jenkins_home/properties/application.yml src/main/resources/application.yml
cp /var/jenkins_home/properties/firebasekey.json src/main/resources/firebasekey.json

IMAGE_NAME="ninucco_back"
IMAGE_TAG="latest"

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

Send build artifacts over SSH

SSH Publishers

SSH Server

Name

a605

고급

Edited

Transfers

Transfer Set

Source files

backend/ninucco/deploy/

Remove prefix

backend/ninucco/deploy/

Remote directory

ninucco_back

Exec command

cd A605/ninucco_back
chmod 764 deploy.sh
./deploy.sh

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

AI 모델 빌드 쉘, 빌드 후 조치(Public Over SSH)

```
cd backend/ninucco-model
cp /var/jenkins_home/properties/.env .env

IMAGE_NAME="ninucco_model"
IMAGE_TAG="latest"

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

Send build artifacts over SSH ?

SSH Publishers

SSH Server

Name ?

a605

고급

Edited

Transfers

Transfer Set

Source files ?

backend/ninucco-model/deploy/

Remove prefix ?

backend/ninucco-model/deploy/

Remote directory ?

ninucco_model

Exec command ?

cd A605/ninucco_model
chmod 764 deploy.sh
./deploy.sh

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

고급

Webhook 설정

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

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

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.

☐ Pipeline events
A pipeline's status changes.

☐ Wiki page events
A wiki page is created or updated.

☐ Deployment events
A deployment starts, finishes, fails, or is canceled.

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

외부 서비스 정보

Amazon S3

Amazon S3 > 버킷 > ninucco-bucket

ninucco-bucket Info

퍼블릭 액세스 가능

객체 | 속성 | 권한 | 지표 | 관리 | 액세스 지점

객체 (271)

객체는 Amazon S3에 저장되어 있는 기본 연타입입니다. 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"/>	0012f54f-467c-3952-b0ae-e575746583d9.png	png	2023. 5. 18. am 10:33:43 AM KST	50.9KB	Standard
<input type="checkbox"/>	0112d85e-7642-3bf3-b826-fb91bd61ca47.png	png	2023. 5. 18. pm 3:09:55 PM KST	410.5KB	Standard
<input type="checkbox"/>	013b3019-c855-3da1-a2fc-30d0395edb6e.png	png	2023. 5. 18. am 10:42:28 AM KST	134.2KB	Standard
<input type="checkbox"/>	02e468c4-3d3b-3342-b778-53f643d16ec2.png	png	2023. 5. 17. pm 3:15:30 PM KST	347.2KB	Standard
<input type="checkbox"/>	03da40e1-9440-351e-bedf-3b7fcc0ff49.png	png	2023. 5. 16. am 9:57:17 AM KST	603.9KB	Standard

Stability AI(Stable Diffusion)

Account

Manage your account and billing information



[Log out](#)
[Contact us](#)
[Billing guide](#)
[Prompt guide](#)

Credits

808.2 ~4,040 images

Purchase credits

\$ 10

1,000 credits ~5,000 images

API keys

KEY	DATE CREATED	
[REDACTED]	2023. 5. 1.	<input type="button" value="📄"/> <input type="button" value="👁"/> <input type="button" value="✕"/>
[REDACTED]	2023. 5. 2.	<input type="button" value="📄"/> <input type="button" value="👁"/> <input type="button" value="✕"/>

Documentation can be found at platform.stability.ai

Payments

DATE	CREDITS
23. 04. 28. 오후 03:40:01	12,000

Danger zone

Delete account
This action is permanent and cannot be undone