



Porting manual

배포

EC2 초기 설정

1. `sudo apt update`: APT 패키지 관리자가 사용하는 로컬 패키지 리스트를 최신 버전으로 업데이트하는 명령어입니다. 시스템에 설치된 패키지를 최신 상태로 유지하기 위해 필요한 업데이트가 있는지 확인할 수 있습니다.
2. `sudo apt upgrade`: 시스템에 설치된 모든 패키지를 최신 버전으로 업그레이드합니다. 업그레이드 할 패키지의 목록이 표시되며, 업그레이드를 계속할 것인지 묻는 메시지가 표시됩니다.
3. `sudo apt install build-essential`: C/C++ 컴파일러를 비롯하여 빌드 과정에서 필요한 다양한 도구와 라이브러리를 설치하는 명령어입니다. C/C++ 프로그램을 컴파일하거나 라이브러리를 빌드하는 데 필요한 패키지들이 자동으로 설치됩니다. `build-essential` 패키지와 의존성 패키지들이 설치됩니다.

```
$ sudo apt update
$ sudo apt upgrade
$ sudo apt install build-essential
```

한국으로 시간 설정

```
$ sudo ln -sf /usr/share/zoneinfo/Asia/Seoul /etc/localtime
# 시간 확인
$ date
```

jdk 설치

```
$ sudo apt install openjdk-17-jdk
$ sudo update-java-alternatives --list
$ sudo update-java-alternatives --set java-1.17.0-openjdk-amd64
```

Docker 설치

1. 기본 설정, 사전 설치
 - 이 명령어는 HTTPS를 사용하여 소프트웨어를 안전하게 다운로드하고, 인증서를 관리하여 보안성을 높이며, 소프트웨어 저장소를 관리할 수 있도록 필요한 패키지를 모두 설치합니다.

```
$ sudo apt install apt-transport-https ca-certificates curl software-properties-common
```

2. 자동 설치 스크립트 활용

- 리눅스 배포판 종류를 자동으로 인식하여 Docker 패키지를 설치해주는 스크립트를 제공

```
$ sudo wget -qO- https://get.docker.com/ | sh
```

3. Docker 서비스 실행하기 및 부팅 시 자동 실행 설정

```
$ sudo systemctl start docker
$ sudo systemctl enable docker
```

4. Docker 그룹에 현재 계정 추가

```
$ sudo usermod -aG docker ${USER}
$ sudo systemctl restart docker
```

- sudo를 사용하지 않고 docker를 사용할 수 있다.
- docker 그룹은 root 권한과 동일하므로 꼭 필요한 계정만 포함
- 현재 계정에서 로그아웃한 뒤 다시 로그인

5. Docker 설치 확인

```
$ docker -v
```

Jenkins 설치

```
$ mkdir jenkins-docker
$ cd jenkins-docker
$ vi Dockerfile
```

Dockerfile

```
FROM jenkins/jenkins:latest

USER root

RUN apt-get update \
  && apt-get -y install lsb-release \
  && curl -fsSL https://download.docker.com/linux/debian/gpg | gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg \
  && echo "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/debian $(lsb_release \
  && apt-get update \
  && apt-get -y install docker-ce docker-ce-cli containerd.io

RUN usermod -aG docker jenkins

USER jenkins
```

```
$ docker build -t my-jenkins:0.1 .
$ docker run -d --name jenkins -v /var/run/docker.sock:/var/run/docker.sock -v jenkins:/var/jenkins_home -p 9090:8080 my-jenkins:0.1
$ docker exec -it jenkins bash
```

비밀번호 확인 후 9090포트로 jenkins에 들어갈 수 있다.

```
jenkins$ docker exec jenkins cat /var/jenkins_home/secrets/initialAdminPassword
```

Jenkins 내부 java 버전 설정

17로 올림

JDK

JDK Installations ✎ Edited

JDK Installations
List of JDK installations on this system

[Add JDK](#)

JDK Name

docker-java

JAVA_HOME

/usr/lib/jvm/java-1.17.0-openjdk-amd64

☐ Install automatically ⓘ

[Add JDK](#)

젠킨스 jdk 버전 11로 올리는 방법

배경 이번 프로젝트의 경우에는 jdk8이 문제가 없던 상황에서 잘 쓰다가 jdk11이 필요한 상황이었다. 예를 들면 rabbitMQ 나 sonarqube를 사용하는데 11 이상이 필요한 것을 확인했다. dev 서버, prod 서버, 프론트 서버, 디러닝 학습 서버,,, 이번 프로젝트 안에서 기존에 젠킨스 설정을 잡아둔게 너무 많아 새로 젠킨스를 설치할 수는 없었고, 결국 버전을 바꾸기로

🔗 <https://www.blog.ecsimsw.com/entry/젠킨스-jdk-버전-11로-올리는-방법>

Name

Required

☒ Install automatically

☒ Install Oracle Java SE Development Kit from the website

Version

Java SE Development Kit 9.0.4

☐ I agree to the Java SE Development Kit License Agreement

Installing JDK requires Oracle account. Please enter your username/password

Oracle Java SE 11 is not available for business, commercial or production use without a commercial license. Public updates for Oracle Java SE 8 released after January 2019 will not be available for business, commercial or production use without a commercial license.

[Oracle Java SE Licensing FAQ](#)

Docker-compose 설치

jenkins 안에서 진행

```
$ docker exec -itu 0 jenkins bash
```

```
jenkins$ curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
jenkins$ chmod +x /usr/local/bin/docker-compose
jenkins$ docker-compose --version
```

Plugin 설치

Dashboard > Jenkins 관리 > Plugin Manager

Updates

Available plugins

Installed plugins

Advanced settings

Download progress

Plugins

Q gitlab

이름 ↓

GitLab Plugin 1.7.8

This plugin allows [GitLab](#) to trigger Jenkins builds and display their results in the GitLab UI.
[Report an issue with this plugin](#)

사용가능

프로젝트에 파일들 추가

./start.sh

```
docker-compose -f docker-compose.yml pull

COMPOSE_DOCKER_CLI_BUILD=1 DOCKER_BUILDKIT=1 docker-compose -f docker-compose.yml up --build -d

docker rmi -f $(docker images -f "dangling=true" -q) || true
```

./docker-compose.yml

```
version: '3.3'

services:
  db:
    container_name: ttarawa_db_1
    image: mariadb
    volumes:
      - db-data:/var/lib/mysql
    environment:
      - MYSQL_ROOT_PASSWORD=ssafy605
      - MYSQL_DATABASE=ssafy605
      - MYSQL_USER=ssafy605
      - MYSQL_PASSWORD=ssafy605
    ports:
      - "3306:3306"
    networks:
      - ttarawa-net
  redis:
    container_name: ttarawa_redis_1
    image: redis:6.2.6-alpine
    ports:
      - "172.26.6.48:6379:6379"
      - "127.0.0.1:6379:6379"
    volumes:
      - /host/system/path:/container/path
      # command: redis-server --requirepass ssafy605
    networks:
      - ttarawa-net

volumes:
  db-data:

networks:
  ttarawa-net:
```

비밀번호로 접근

```
export REDIS_PASSWORD=ssafy605
docker exec -it ttarawa_redis_1 redis-cli -a $REDIS_PASSWORD
```

./ttarawa/Dockerfile

```
FROM openjdk:17-jdk-slim as builder

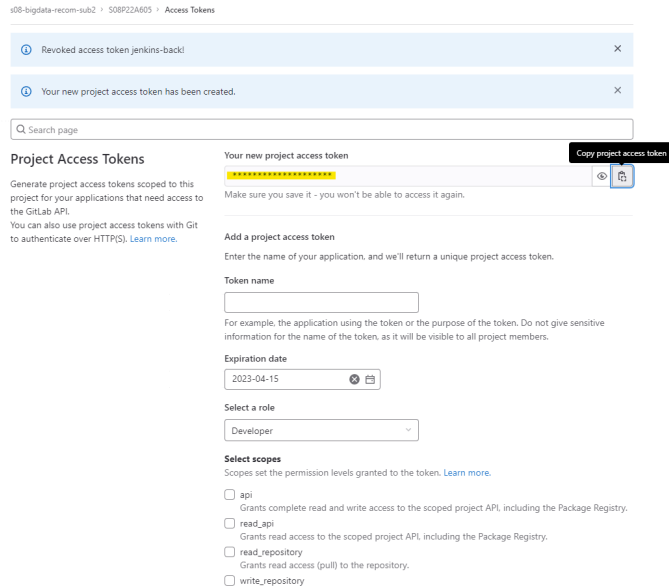
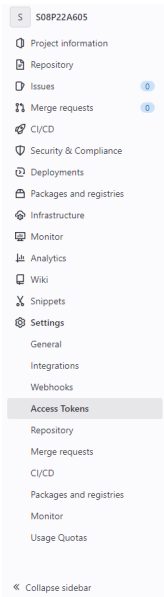
COPY gradlew .
COPY gradle gradle
COPY build.gradle .
COPY settings.gradle .
COPY src src
RUN chmod +x ./gradlew
RUN ./gradlew bootJar

FROM openjdk:17-jdk-slim
COPY --from=builder build/libs/*.jar app.jar
EXPOSE 8080

ENTRYPOINT ["java", "-Duser.timezone=Asia/Seoul", "-jar", "/app.jar"]
```

General → 소스 코드 관리

gitlab에서 토큰 복사



빨간색 부분에 붙여넣기

Configure

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

소스 코드 관리

- None
- Git

Repositories

Repository URL ?

https://s08-bigdata-recom-sub-@labassafy.com/s08-bigdata-recom-sub2/S08P22A605.git

Credentials ?

- none -

+ Add

고급

Add Repository

Branches to build

Branch Specifier (blank for 'any') ?

/s08P22A605

Add Branch

Configure

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

Build Steps

Execute shell ?

Command

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

`chmod +x start.sh`

고급 ▾

Execute shell ?

Command

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

`./start.sh`

고급 ▾

Webhook 설정

Configure

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

☐ Build after other projects are built ?

☐ Build periodically ?

☒ Build when a change is pushed to GitLab. GitLab webhook URL: <http://8a605-passatfio9090/project/jenkins-back> ?

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 ▾

url 복사

Configure

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

그룹 ^

- ☒ Enable [ci-skip]
- ☒ Ignore WIP Merge Requests

Labels that forces builds if they are added (comma-separated)

- ☒ Set build description to build cause (eg. Merge request or Git Push)
- ☐ Build on successful pipeline events

Pending build name for pipeline ?

- ☐ Cancel pending merge request builds on update

Allowed branches

- ☒ Allow all branches to trigger this job ?
- ☐ Filter branches by name ?
- ☐ Filter branches by regex ?
- ☐ Filter merge request by label

Secret token ?

Generate

Clear

생성된 token 복사 후 gitlab project에 붙여넣기

- S 508P22A605
- Project information
- Repository
- Issues
- Merge requests
- CI/CD
- Security & Compliance
- Deployments
- Packages and registries
- Infrastructure
- Monitor
- Analytics
- Wiki
- Snippets
- Settings
 - General
 - Integrations
 - Webhooks
 - Access Tokens
 - Repository

s08-bigdata-recom-sub2 > 508P22A605 > Webhook Settings

Hook executed successfully: HTTP 200

Search page

Webhooks

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

Branch name or wildcard pattern to trigger on (leave blank for all)

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.

📄 목차로 가기

무중단 배포

Nginx 설치

nginx-Dockerfile

```
FROM nginx:1.11

RUN rm -rf /etc/nginx/conf.d/default.conf

COPY ./conf.d/app.conf /etc/nginx/conf.d/app.conf
COPY ./conf.d/nginx.conf /etc/nginx/nginx.conf

VOLUME ["/data", "/etc/nginx", "/var/log/nginx"]

WORKDIR /etc/nginx

CMD ["nginx"]
```

./conf.d/app.conf

```

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

    server_name "";

    access_log off;

    location / {
        proxy_pass http://docker-app;
        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 http;
        proxy_max_temp_file_size 0;

        proxy_connect_timeout 150;
        proxy_send_timeout 100;
        proxy_read_timeout 100;

        proxy_buffer_size 8k;
        proxy_buffers 4 32k;
        proxy_busy_buffers_size 64k;
        proxy_temp_file_write_size 64k;
    }
}

```

./conf.d/nginx.conf

```

daemon off;
user www-data;
worker_processes 2;

error_log /var/log/nginx/error.log warn;
pid /var/run/nginx.pid;

events {
    worker_connections 1024;
    use epoll;
    accept_mutex off;
}

http {
    include /etc/nginx/mime.types;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

    default_type application/octet-stream;

    upstream docker-app {
        least_conn;
        server j8a605.p.ssafy.io:8085 weight=10 max_fails=3 fail_timeout=30s;
        server j8a605.p.ssafy.io:8086 weight=10 max_fails=3 fail_timeout=30s;
    }

    log_format main '$remote_addr - $remote_user [$time_local] "$request"'
        '$status $body_bytes_sent "$http_referer"'
        '"$http_user_agent" "$http_x_forwarded_for"';

    access_log /var/log/nginx/access.log main;

    sendfile on;
    #tcp_nopush

    keepalive_timeout 65;
}

```



```

client_max_body_size 300m;
client_body_buffer_size 128k;

gzip on;
gzip_http_version 1.0;
gzip_comp_level 6;
gzip_min_length 0;
gzip_buffers 16 8k;
gzip_proxied any;
gzip_types text/plain text/css text/xml text/javascript application/xml application/xml+rss application/javascript application/json;
gzip_disable "MSIE [1-6]\.";
gzip_vary on;

#리눅스환경에서 취급하는 호스팅하는 웹서버 경로
include /etc/nginx/conf.d/*.conf;
}

```

nginx 실행

```

docker build -t docker-nginx:0.1 -f nginx-Dockerfile .

docker run -d --name docker-nginx -p 80:80 docker-nginx:0.1

```

Spring Boot Application 작성

application.yml

```

spring:
  jpa:
    hibernate:
      ddl-auto: update
      generate-ddl: false
      show-sql: true
    datasource:
      # mariaDB setting
      driver-class-name: org.mariadb.jdbc.Driver
      url: jdbc:mariadb://j8a605.p.ssafy.io:3306/ssafy605
      username: ssafy605
      password: ssafy605

  server:
    port: 8080

```

docker-compose.blue.yml

```

version: '3.8'

services:
  app:
    image: app:0.1
    container_name: app_blue
    environment:
      - "spring_profiles_active=blue"
    ports:
      - "8085:8080"

```

docker-compose.green.yml

```

version: '3.8'

services:
  app:
    image: app:0.2
    container_name: app_green
    environment:
      - "spring_profiles_active=green"
    ports:
      - "8086:8080"

```

배포 스크립트 작성

jenkins 내부 execute shell

```

cd ttarawa
chmod +x ./gradlew
./gradlew bootJar
chmod +x deploy.sh
./deploy.sh

```

deploy.sh

```

#!/bin/bash

function create_docker_image_blue(){

    echo "> blue docker image 만들기"

    ./gradlew clean build

    docker build -t app:0.1 .

}

function create_docker_image_green(){

    echo "> green docker image 만들기"

    ./gradlew clean build

    docker build -t app:0.2 .

}

function execute_blue(){
    docker ps -q --filter "name=app_blue" || grep -q . && docker stop app_blue && docker rm app_blue || true

    sleep 10

    docker-compose -p app-blue -f docker-compose.blue.yml up -d

    sleep 10

    echo "GREEN:8086 종료"
    docker-compose -p app-green -f docker-compose.green.yml down

    #dangling=true : 불필요한 이미지 지우기
    docker rmi -f $(docker images -f "dangling=true" -q) || true
}

function execute_green(){
    docker ps -q --filter "name=app_green" || grep -q . && docker stop app_green && docker rm app_green || true

    echo "GREEN:8086 실행"
    docker-compose -p app-green -f docker-compose.green.yml up -d

    sleep 10

    echo "BLUE:8085 종료"
    docker-compose -p app-blue -f docker-compose.blue.yml down
}

```

```

#dangling=true : 불필요한 이미지 지우기
docker rmi -f $(docker images -f "dangling=true" -q) || true
}

# 현재 사용중인 어플리케이션 확인
# 8086포트의 값이 없으면 8085포트 사용 중
# shellcheck disable=SC2046
RUNNING_GREEN=$(docker ps -aqf "name=app_green")
RUNNING_BLUE=$(docker ps -aqf "name=app_blue")

echo ${RUNNING_GREEN}
echo ${RUNNING_BLUE}

# Blue or Green
if [ -z ${RUNNING_GREEN} ]
then
    # 초기 실행 : BLUE도 실행중이지 않을 경우
    if [ -z ${RUNNING_BLUE} ]
    then
        echo "구동 앱 없음 => BLUE 실행"

        create_docker_image_blue

        sleep 10

        docker-compose -p app-blue -f docker-compose.blue.yml up -d
    else
        # 8086포트로 어플리케이션 구동
        echo "BLUE:8085 실행 중"

        create_docker_image_green

        execute_green
    fi
else
    # 8085포트로 어플리케이션 구동
    echo "GREEN:8086 실행 중"

    echo "BLUE:8085 실행"

    create_docker_image_blue

    execute_blue
fi

# 새로운 어플리케이션 구동 후 현재 어플리케이션 종료
#kill -15 ${RUNNING_PORT_PID}

```

 [목차로 가기](#)

배포 확인



Spring Actuator는 *org.springframework.boot:spring-boot-starter-actuator*

패키지를 Dependency에 추가만 해주면 바로 사용할 수 있는 기능으로, Spring boot를 사용하여 Backend를 구현할 경우 애플리케이션 모니터링 및 관리 측면에서 도움을 줄 수 있습니다.

build.gradle

```

// actuator
implementation 'org.springframework.boot:spring-boot-starter-actuator'

```

application.yml

```
management:
  endpoints:
    web:
      exposure:
        include:
          - "httpexchanges"
          - "health"
    endpoint:
      health:
        enabled: true
        show-details: always

  httpexchanges:
    recording:
      enabled: true
```

ActuatorHttpExchangesConfig

```
package com.jsdckj.ttarawa.config.actuator;

import org.springframework.boot.actuate.web.exchanges.HttpExchangeRepository;
import org.springframework.boot.actuate.web.exchanges.InMemoryHttpExchangeRepository;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class ActuatorHttpExchangesConfig {
    @Bean
    public HttpExchangeRepository httpTraceRepository() {
        return new InMemoryHttpExchangeRepository();
    }
}
```

접근 방법

{ip:port}/actuator #로 접근하면 뜨는 모든 url에 접근 가능

목차로 가기

Amazon S3

application-aws.yml

```
cloud:
  aws:
    s3:
      bucket:
        ttarawa-bucket
      credentials:
        access-key: AWS S3 ACCESS KEY
        secret-key: AWS S3 SECRET KEY
      region:
        static: ap-northeast-2
        auto: false
      stack:
        false
```

버킷 (1) [Info](#)

버킷은 S3에 저장되는 데이터의 컨테이너입니다. [자세히 알아보기](#)

이름으로 버킷 찾기

이름	AWS 리전	액세스	생성 날짜
ttarawa-bucket	아시아 태평양(서울) ap-northeast-2	객체를 퍼블릭으로 설정할 수 있음	2023. 3. 19. pm 5:27:47 PM KST

ttarawa-bucket [Info](#)

[객체](#) | [속성](#) | [권한](#) | [지표](#) | [관리](#) | [액세스 지정](#)

객체 (2)

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](#)

이름으로 객체 찾기

이름	유형	마지막 수정	크기	스토리지 클래스
history/	폴더	-	-	-
profile/	폴더	-	-	-

- ttarawa-bucket의 history폴더와 profile 폴더에 저장

[📁목차로 가기](#)

OAuth2.0 소셜 로그인

application.yml

```
spring:
  security:
    oauth2:
      client:
        registration:
          kakao:
            client-id: KAKAO CLIENT ID
            client-secret: KAKAO CLIENT SECRET
            scope:
              - profile_nickname
              - account_email
              - profile_image
            client-name: Kakao
            authorization-grant-type: authorization_code
            redirect-uri: '{baseUrl}/{action}/oauth2/code/{registrationId}'
            client-authentication-method: POST
          naver:
            client-id: NAVER CLIENT ID
            client-secret: NAVER CLIENT SECRET
            redirect-uri: '{baseUrl}/{action}/oauth2/code/{registrationId}'
            authorization-grant-type: authorization_code
            client-authentication-method: post
            scope:
              - name
              - email
              - profile_image
            client-name: Naver
          google:
            client-id: GOOGLE CLIENT ID
            client-secret: GOOGLE CLIENT SECRET
            redirect-uri: '{baseUrl}/{action}/oauth2/code/{registrationId}'
            scope:
              - email
              - profile
        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
```

```

naver:
  authorization-uri: https://nid.naver.com/oauth2.0/authorize
  token-uri: https://nid.naver.com/oauth2.0/token
  user-info-uri: https://openapi.naver.com/v1/nid/me
  user-name-attribute: response

```

Kakao Login

Redirect URI

삭제 수정

Redirect URI	http://localhost:8080/login/oauth2/code/kakao http://j8a605.p.ssafy.io/login/oauth2/code/kakao
--------------	--

- 카카오 로그인에서 사용할 OAuth Redirect URI를 설정합니다. (최대 10개)
- REST API로 개발하는 경우 필수로 설정해야 합니다.

Naver Login

로그인 오픈 API 서비스 환경 ②

환경 추가 ▼

PC 웹

서비스 URL

<http://localhost:8080/>

서비스 URL 예시: (O) <http://naver.com> (X) <http://www.naver.com>

서비스 URL 값이 잘못 입력되어 있으면 정확한 값으로 수정하실 때까지 네이버 로그인 사용이 일시적으로 제한됩니다.

불법/음란성 사이트 등 이용약관에 위배되는 사이트의 경우, 이용이 제한될 수 있습니다.

서비스하려는 사이트 URL과 동일한 사이트 URL로 해주셔야 **네이버 로그인** 배지가 노출됩니다.

네이버 로그인

Callback URL (최대 5개)

<http://localhost:8080/login/oauth2/code/naver>

-

<http://j8a605.p.ssafy.io/login/oauth2/code/naver>

+

텍스트 폼 우측 끝의 '+' 버튼을 누르면 행이 추가되며, '-' 버튼을 누르면 행이 삭제됩니다.

Callback URL은 네이버 로그인 후 이동할 페이지 URL입니다. Callback URL 값이 잘못 입력되어 있으면 정확한 값으로 수정하실 때까지 네이버 로그인 사용이 일시적으로 제한됩니다.

입력한 주소와 다른 Callback URL로 리다이렉트 될 경우, 이용이 제한될 수 있습니다.

Google Login

Porting manual

14

이름 *
따라와

OAuth 2.0 클라이언트의 이름입니다. 이 이름은 콘솔에서 클라이언트를 식별하는 용도로만 사용되며 최종 사용자에게 표시되지 않습니다.



아래에 추가한 URI의 도메인이 [승인된 도메인](#)으로 [OAuth 동의 화면](#)에 자동으로 추가됩니다.

승인된 자바스크립트 원본

브라우저 요청에 사용

URI 1 *
`https://auth.expo.io`

+ URI 추가

승인된 리디렉션 URI


웹 서버의 요청에 사용

URI 1 *
`http://j8a605.p.ssafy.io/login/oauth2/code/google`

URI 2 *
`http://localhost:8080/login/oauth2/code/google`

URI 3 *
`https://auth.expo.io/@choisb/ttarawa`

+ URI 추가

 목차로 가기

플라스크 배포

1. 프로젝트 최상단 위치에서 필요한 pip 파일 requirements.txt 에 작성

```
Flask
Flask-SQLAlchemy
SQLAlchemy
pandas
scikit-learn
numpy
scipy
mysqlclient
mariadb==1.0.11
```

2. Dockerfile 작성

```

# 기반 이미지를 Python 3.8 버전을 사용하는 이미지로 설정합니다.
FROM python:3.8

RUN apt update && \
    apt install -y libmariadb-dev-compat libmariadb-dev && \
    apt clean && \
    rm -rf /var/lib/apt/lists/*

# 작업 디렉토리를 /app으로 설정합니다.
WORKDIR /app

# requirements.txt 파일을 작업 디렉토리로 복사합니다.
COPY requirements.txt .

# requirements.txt에 명시된 라이브러리들을 설치합니다.
RUN pip install --no-cache-dir -r requirements.txt

# 현재 디렉토리의 모든 파일을 컨테이너의 /app/ 디렉토리에 복사합니다.
COPY . /app

# gunicorn을 설치합니다.
RUN pip install gunicorn

# gunicorn으로 Flask 애플리케이션을 실행합니다.
CMD ["gunicorn", "--bind", "0.0.0.0:8000", "app:app"]


```

3. docker-compose.yml에 추가

```

cf:
  build:
    context: ./ttarawa-cf
    dockerfile: Dockerfile
  ports:
    - "5000:5000"
  volumes:
    - ./ttarawa-cf:/app
  depends_on:
    - db
    - redis
    - backend

```

 [목차로 가기](#)