포팅 매뉴얼

🎨시스템 환경 및 구성

- **OS** : Ubuntu 20.04 LTS
- **Backend**
    - IDE : Intellij IDEA 2023.1.3
    - Spring Boot 2.7.14
    - jdk : 1.8
    - MySql : 8.0.34
    - Docker : 24.0.5
    - Docker-compose : 1.25.0
    - Jenkins : 2.401.3
    - Nginx : 1.25.1
- **AI Backend**
    - OS : Ubuntu 20.04 LTS
    - GPU : NVIDIA L4
    - Python : 3.9 (Virtual Environment)
    - Flask : 2.3.2
    - Pytorch : 2.0.1
    - Redis : 5.0.7
    - FFMPEG : 4.2.7
- **Frontend**
    - React : 9.5.1
    - Node.js : 18.16.1
    - ES6
    - HTML5
    - CSS3
    - Router
- **Cloud service**
    - AWS EC2
    - AWS S3
    - GCP(Google Cloud Platform)

설정 파일 및 환경 변수 정보

Spring

- application.yaml

```
spring:
  profiles.active: local
  # 데이터 소스 설정
  datasource:
    driverClassName: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://${DB_URL}/ssaout?characterEncoding=UTF-8&serverTimezone=UTC
    username: root
    password: ${DB_PASSWORD}
    hikari:
      pool-name: jpa-hikari-pool
      maximum-pool-size: 5
      jdbc-url: ${spring.datasource.url}
      username: ${spring.datasource.username}
      password: ${spring.datasource.password}
      driver-class-name: ${spring.datasource.driver-class-name}
      data-source-properties:
        rewriteBatchedStatements: true
  # JPA 설정
  jpa:
    generate-ddl: true
    hibernate:
      ddl-auto: update
    show-sql: true
    properties:
      hibernate:
        dialect: org.hibernate.dialect.MySQL8Dialect
        hbm2ddl.import_files_sql_extractor: org.hibernate.tool.hbm2ddl.MultipleLinesSqlCommandExtractor
        current_session_context_class: org.springframework.orm.hibernate5.SpringSessionContext
        default_batch_fetch_size: ${chunkSize:100}
        jdbc.batch_size: 20
        order_inserts: true
        order_updates: true
        format_sql: true
  profiles:
    include: oauth
  servlet:
    multipart:
      enabled: true
      max-file-size: 100MB
      max-request-size: 100MB

# cors 설정
cors:
  allowed-origins: 'https://i9e203.p.ssafy.io'
  allowed-methods: GET,POST,PUT,DELETE,OPTIONS
  allowed-headers: '*'
  max-age: 3600

# jwt secret key 설정
jwt:
  secret: ${JWT_SECRET_KEY}

# 토큰 관련 secret Key 및 RedirectUri 설정
app:
  auth:
    tokenSecret: ${AUTHENTICATION_TOKEN_SECRET}
    tokenExpiry: 1800000
    refreshTokenExpiry: 604800000
  oauth2:
    authorizedRedirectUris:
      - https://i9e203.p.ssafy.io/oauth/redirect

# AWS S3
cloud:
  aws:
    s3:
      bucket: ssarout
    credentials:
      access-key: ${S3_ACCESS_KEY}
      secret-key: ${S3_SECRET_KEY}
    region:
      static: ap-northeast-2
      auto: false
    stack:
      auto: false

kakao:
  admin: ${KAKAO_API_ADMIN_KEY}
```

application-oauth.yml

```
# Security OAuth
spring:
  security:
    oauth2.client:
      registration:
        google:
          clientId: "${GOOGLE_CLIENT_ID_KEY}"
          clientSecret: "${GOOGLE_CLIENT_SECRET_KEY}"
          scope:
            - email
            - profile
        kakao:
          clientId: "${KAKAO_CLIENT_ID_KEY}"
          clientSecret: "${KAKAO_CLIENT_SECRET_KEY}"
          clientAuthenticationMethod: post
          authorizationGrantType: authorization_code
          redirectUri: "https://i9e203.p.ssafy.io/login/oauth2/code/kakao"
          scope:
            - profile_nickname
            - profile_image
            - account_email
          clientName: Kakao
      # Provider 설정
      provider:
        kakao:
          authorizationUri: https://kauth.kakao.com/oauth/authorize
          tokenUri: https://kauth.kakao.com/oauth/token
          userInfoUri: https://kapi.kakao.com/v2/user/me
          userNameAttribute: id
```

빌드/ 배포 가이드

1. MySQL 설치

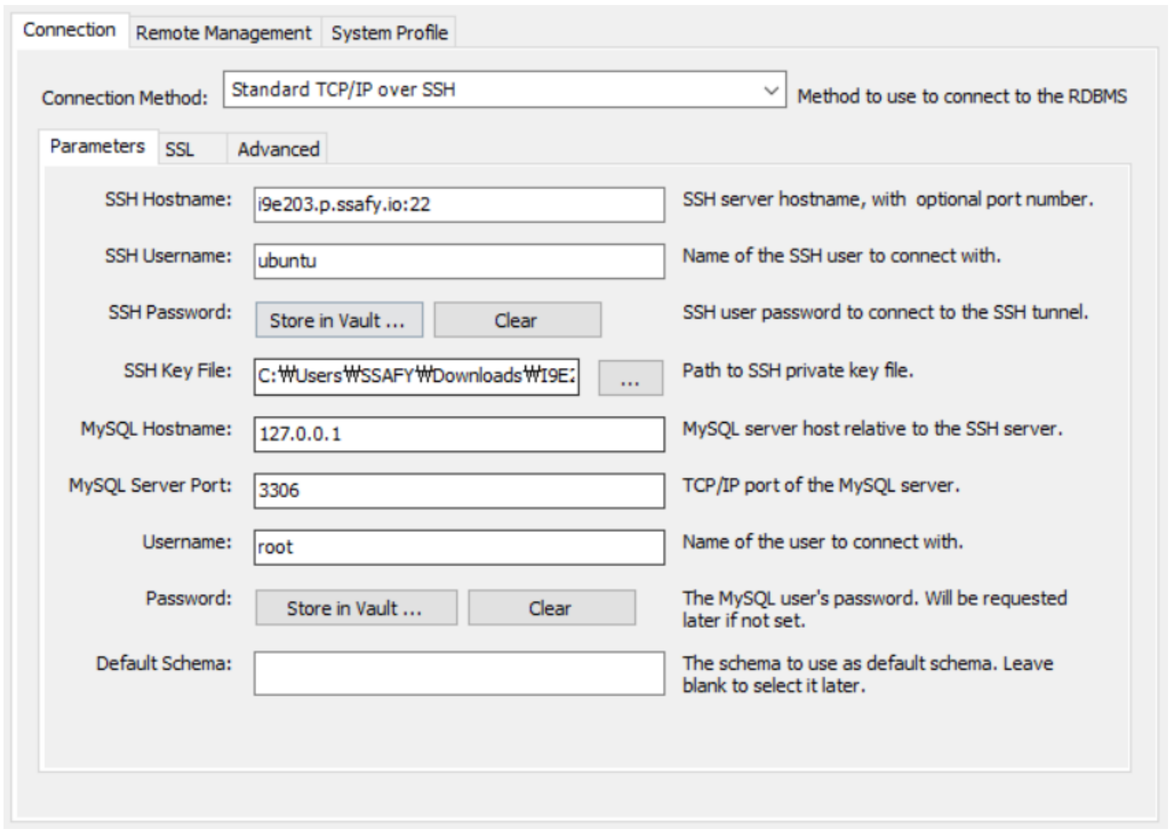
1. MySQL 설치

   ```
   # Update apt package index and install packages
   sudo apt-get update

   sudo apt install mysql-server

   sudo ufw allow 3306
   ```

2. 로컬에서 MySQL Workbench를 통해 접속


Connection Method: Standard TCP/IP over SSH — Method to use to connect to the RDBMS

SSH Hostname:	i9e203.p.ssafy.io:22	SSH server hostname, with optional port number.
SSH Username:	ubuntu	Name of the SSH user to connect with.
SSH Password:	Store in Vault ... / Clear	SSH user password to connect to the SSH tunnel.
SSH Key File:	C:₩Users₩SSAFY₩Downloads₩I9E2	Path to SSH private key file.
MySQL Hostname:	127.0.0.1	MySQL server host relative to the SSH server.
MySQL Server Port:	3306	TCP/IP port of the MySQL server.
Username:	root	Name of the user to connect with.
Password:	Store in Vault ... / Clear	The MySQL user's password. Will be requested later if not set.
Default Schema:		The schema to use as default schema. Leave blank to select it later.

2. Docker 설치

> 프로젝트는 Docker를 기반으로 실행됩니다.
> * Ubuntu 20.04, Docker 24.0.5, Docker-compose 1.25.0를 기준으로 작성했습니다.
>
> Docker 공식문서를 기준으로 진행했습니다.(링크)

1. 저장소 설정

```
# Update apt package index and install packages
sudo apt-get update
sudo apt-get install ca-certificates curl gnupg

# Add Docker's official GPG key
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
sudo chmod a+r /etc/apt/keyrings/docker.gpg

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

# Update apt package index
sudo apt-get update
```

2. Docker와 플러그인 설치

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

3. Jenkins 설치

1. Jenkins lts image 다운로드

   ```
   sudo docker pull jenkins/jenkins:lts
   ```

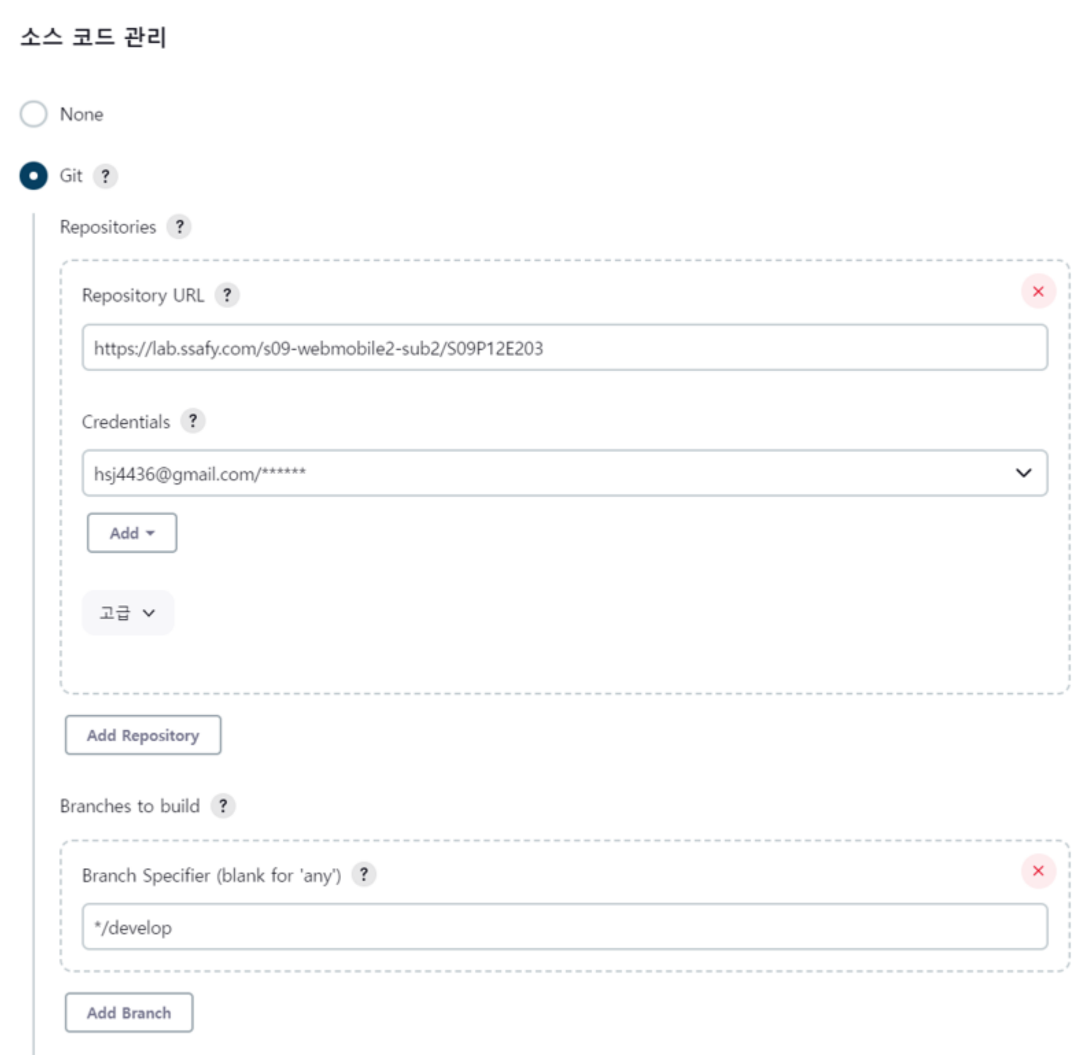
2. 이미지 실행

   ```
   sudo docker run -d -p 7070:8080 \
   -v /var/jenkins:/var/jenkins_home \
   -v /var/run/dokcer.sock:/var/run/docker/sock \
   --name jenkins -u root jenkins/jenkins:lts
   ```

3. 플러그인 설치
   젠킨스 기본 플러그인 설치 후, GitLab Plugin 추가 설치

4. 젠킨스 환경변수 설정
   Jenkins 관리 → System → Global properties 탭의 Environment variables 선택 후 환경변수 설정
   - 환경변수 목록
     - DB_PASSWORD
     - DB_URL
     - DOCKER_HUB_ID
     - DOCKER_HUB_PASSWORD
     - GOOGLE_CLIENT_ID
     - GOOGLE_CLIENT_SECRET
     - JWT_SECRET
     - KAKAO_API_ADMIN_KEY
     - KAKAO_CLIENT_ID
     - KAKAO_CLIENT_SECRET
     - S3_ACCESS_KEY
     - S3_SECRET_KEY
     - TOKEN_SECRET

5. 젠킨스 아이템 설정 및 웹훅 설정
   - 소스 코드 관리



소스 코드 관리

- ○ None
- ● Git

Repositories

Repository URL

https://lab.ssafy.com/s09-webmobile2-sub2/S09P12E203

Credentials

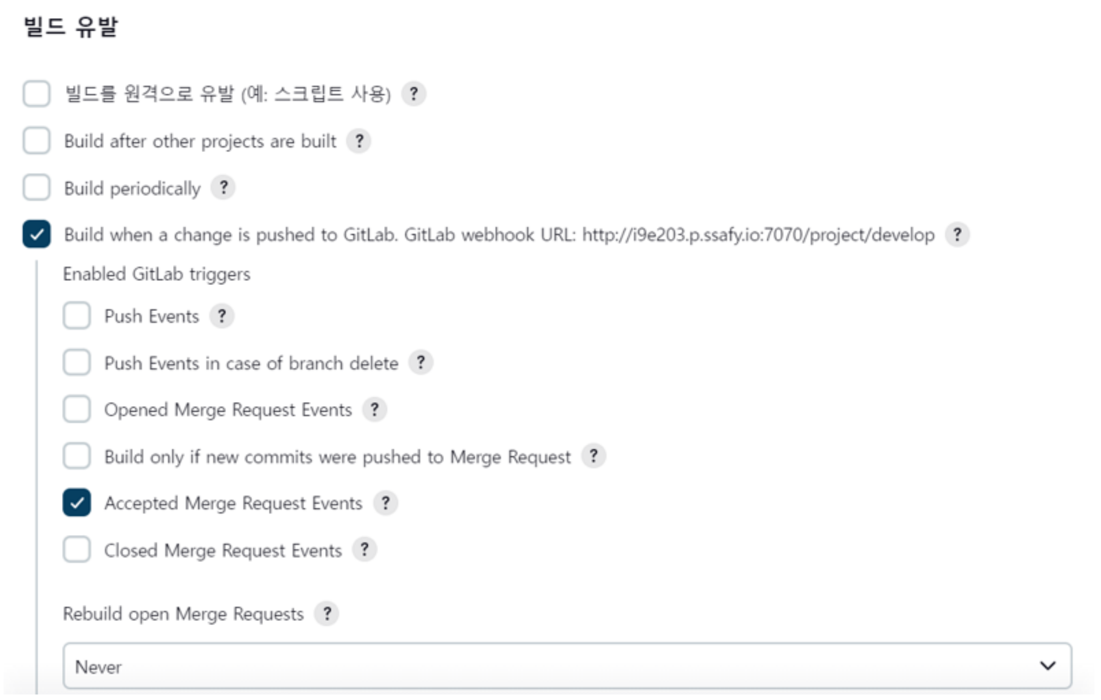
hsj4436@gmail.com/******

Branches to build

Branch Specifier (blank for 'any')

*/develop

- 빌드 유발



빌드 유발

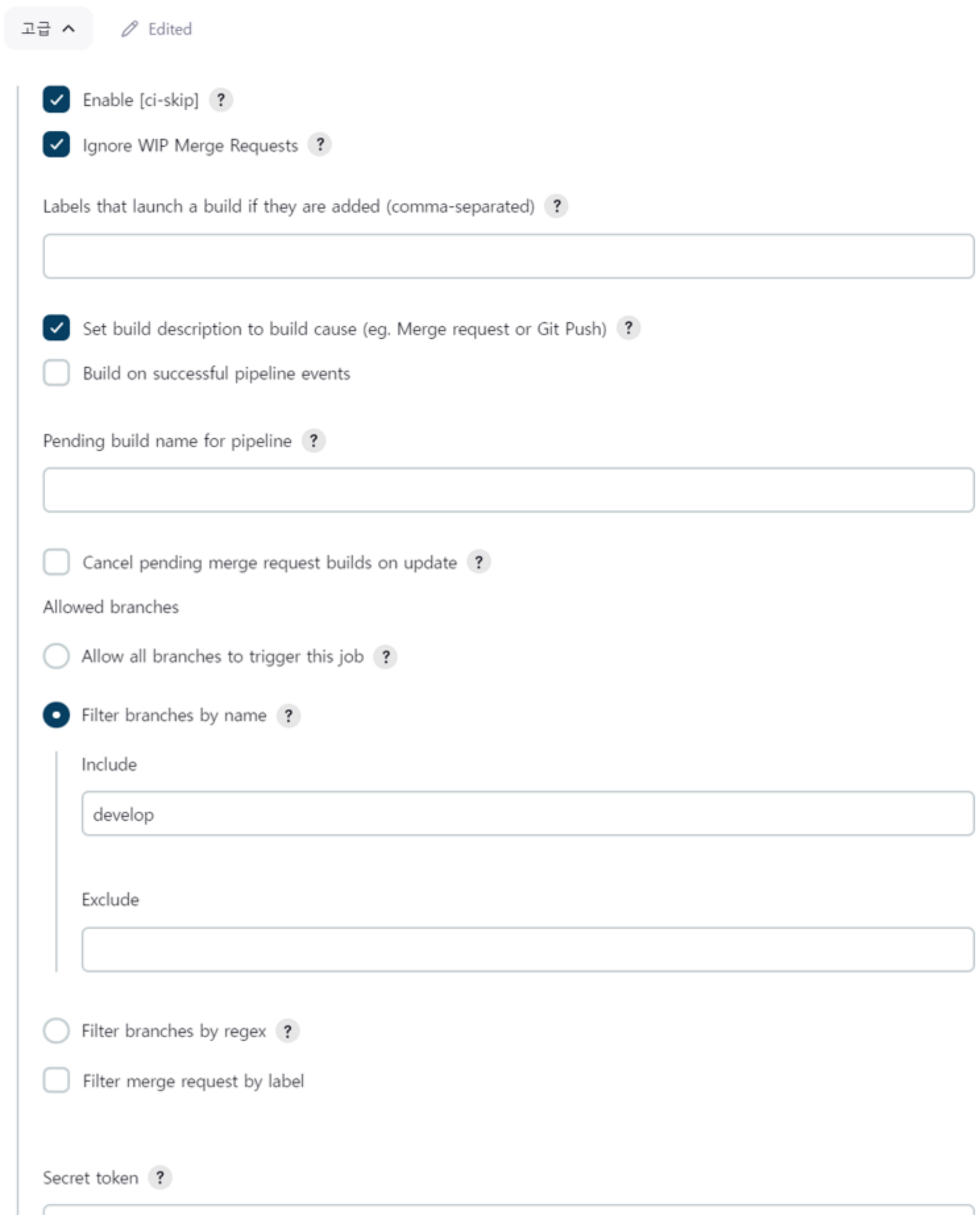
- ☐ 빌드를 원격으로 유발 (예: 스크립트 사용)
- ☐ Build after other projects are built
- ☐ Build periodically
- ☑ Build when a change is pushed to GitLab. GitLab webhook URL: http://i9e203.p.ssafy.io:7070/project/develop

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



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

Labels that launch a build 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

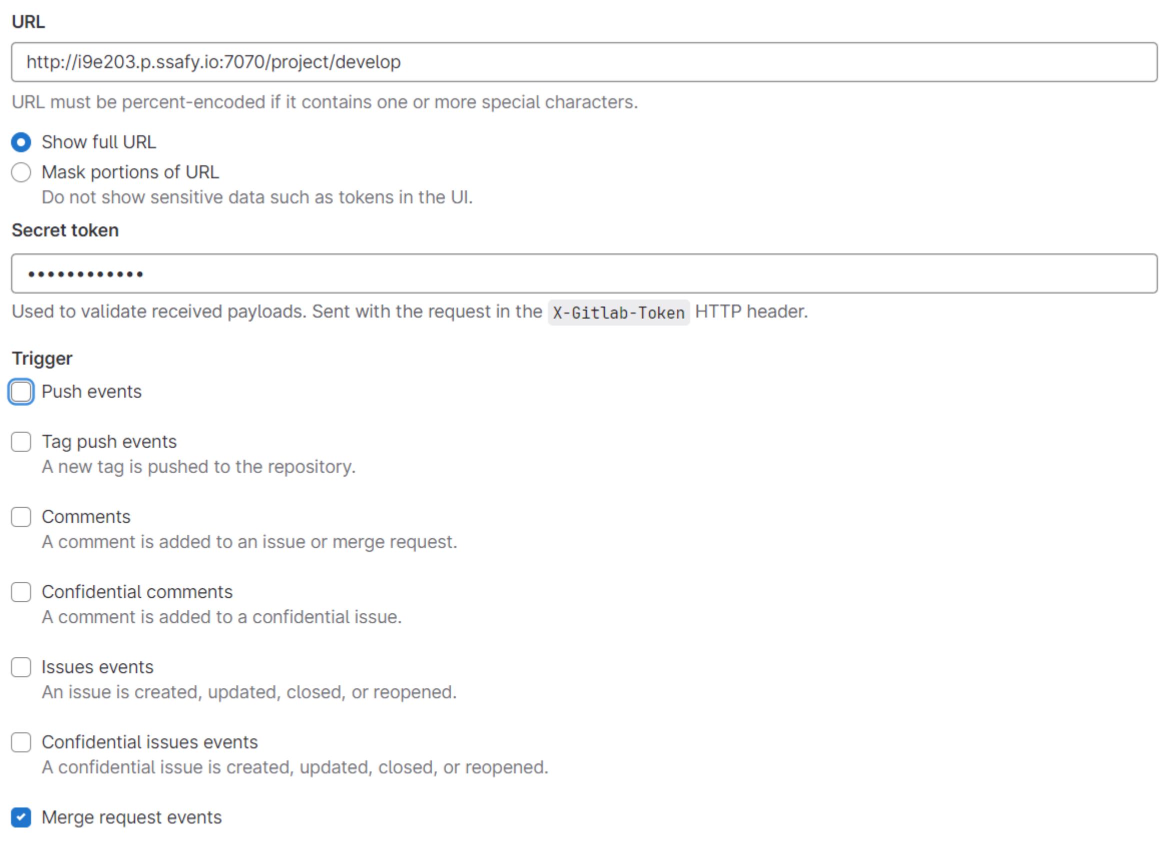
Include: develop

Exclude:

- ○ Filter branches by regex
- [ ] Filter merge request by label

Secret token

- 깃랩 웹훅 설정



URL

http://i9e203.p.ssafy.io:7070/project/develop

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

● Show full URL

○ Mask portions of URL
Do not show sensitive data such as tokens in the UI.

Secret token

••••••••••••

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

Trigger

☐ Push events

☐ 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

- Build Script 작성

```
# Backend Build and Push
cd Ssarout/BackEnd

chmod +x ./gradlew
./gradlew clean build -x test
docker ps -f name=ssarout/backend -q | xargs --no-run-if-empty docker container stop
docker container ls -a -f name=ssarout/backend -q | xargs -r docker container rm

# Docker Hub Login
docker login -u ${DOCKER_HUB_ID} -p ${DOCKER_HUB_PASSWORD}

# Build image with environment variables and Dockerfile
docker build \
--build-arg DB_PASSWORD=${DB_PASSWORD} \
--build-arg DB_URL=${DB_URL} \
--build-arg GOOGLE_CLIENT_ID=${GOOGLE_CLIENT_ID} \
--build-arg GOOGLE_CLIENT_SECRET=${GOOGLE_CLIENT_SECRET} \
--build-arg JWT_SECRET=${JWT_SECRET} \
--build-arg KAKAO_CLIENT_ID=${KAKAO_CLIENT_ID} \
--build-arg KAKAO_CLIENT_SECRET=${KAKAO_CLIENT_SECRET} \
--build-arg TOKEN_SECRET=${TOKEN_SECRET} \
-t ssarout/backend .

# Push image to dockerhub's repository
docker push ssarout/backend

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

# Frontend Build and Push
cd ../../front-end

# Build image with Dockerfile
docker build -t ssarout/frontend .

# Push image to dockerhub's repository
docker push ssarout/frontend

cd /var/jenkins_home
docker-compose rm
docker-compose up -d
docker image prune -f
```

- docker-compose 파일 작성

  docker-compose.yaml

```
version: '3'
services:

  spring:
    container_name: spring
    image: ssarout/backend:latest
    expose:
      - 8080
    environment:
      - TZ=Asia/Seoul

  react:
    container_name: react
    image: ssarout/frontend:latest
    expose:
      - 3000
    environment:
      - TZ=Asia/Seoul

  nginx:
    container_name: nginx
    image: nginx:latest
    restart: always
    volumes:
      - /etc/nginx/:/etc/nginx/
      - /etc/letsencrypt:/etc/letsencrypt
    ports:
      - 80:80
      - 443:443
    depends_on:
      - spring
      - react
    environment:
      - TZ=Asia/Seoul
```

4. Ngnix 설치

1. SSL 인증서 발급

```
yum install epel-release
yum install certbot
```

```
certbot certonly --standalone -d domain.com
```

2. Nginx image 다운로드

```
sudo docker pull nginx:latest
```

3. Nginx 설정 파일

   a. /etc/nginx/nginx.conf

```
user  nginx;
worker_processes  1;

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

events {
    worker_connections  1024;
}
```

```
http {
    include       /etc/nginx/mime.types;

    default_type  application/octet-stream;

    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     on;

    keepalive_timeout  65;

    #gzip  on;

    include /etc/nginx/conf.d/*.conf;


    server {
       listen 80;
       server_name i9e203.p.ssafy.io;
       server_tokens off;
       return 301 https://$host$request_uri;
    }

    server {
       listen 443 ssl;
       server_name i9e203.p.ssafy.io;
       ssl_certificate /etc/letsencrypt/live/i9e203.p.ssafy.io/fullchain.pem;
       ssl_certificate_key /etc/letsencrypt/live/i9e203.p.ssafy.io/privkey.pem;
       include /etc/letsencrypt/options-ssl-nginx.conf;
       client_max_body_size 20M;

       location / {
          include /etc/nginx/proxy_params;
          proxy_pass http://react:3000;
       }

       location /oauth/redirect {
          include /etc/nginx/proxy_params;
          proxy_pass http://react:3000;
       }

       location /api {
          include /etc/nginx/proxy_params;
          # proxy_pass http://i9e203.p.ssafy.io:8080;
          proxy_pass http://spring:8080;
       }

       location /oauth2 {
          include /etc/nginx/proxy_params;
          proxy_pass http://spring:8080;
       }

       location /login/oauth2/code/kakao {
          include /etc/nginx/proxy_params;
          proxy_pass http://spring:8080;
       }

       location /login/oauth2/code/google {
          include /etc/nginx/proxy_params;
          proxy_pass http://spring:8080;
       }

       location /logout {
          include /etc/nginx/proxy_params;
          proxy_pass http://spring:8080;
       }

    }

}
```

b. /etc/nginx/conf.d/default.conf

```
server {
    listen 3000;

    location / {

        root /usr/share/nginx/html;
        index index.html index.htm;
        try_files $uri  $uri/ /index.html;

    }
}
```

c. /etc/nginx/conf.d/service-url.inc

```
set $service_url http://localhost:8080;
```

d. /etc/nginx/proxy_params

```
proxy_set_header Host $http_host;
proxy_set_header Upgrade $http_upgrade;
proxy_set_header Connection keep-alive;
proxy_cache_bypass $http_upgrade;
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;
proxy_set_header X-Forwarded-Host $host;
proxy_set_header X-Forwarded-Server $host;
proxy_set_header X-Forwarded-Port $server_port;
proxy_set_header X-NginX-Proxy true;

client_max_body_size 256M;
client_body_buffer_size 1m;

proxy_buffering on;
proxy_buffers 256 16k;
proxy_buffer_size 128k;
proxy_busy_buffers_size 256k;

proxy_temp_file_write_size 256k;
proxy_max_temp_file_size 1024m;

proxy_connect_timeout 300;
proxy_send_timeout 300;
proxy_read_timeout 300;
proxy_intercept_errors on;
```

5. 배포

Front-End Dockerfile

```
FROM node:alpine as builder
WORKDIR /usr/src/app
COPY package.json .
RUN npm install
COPY ./ ./
RUN npm run build

FROM nginx
EXPOSE 3000
COPY ./default.conf /etc/nginx/conf.d/default.conf
COPY --from=builder /usr/src/app/build /usr/share/nginx/html
```

Back-End Dockerfile

```
FROM openjdk:8-jdk
ENV JAVA_OPTS "-XX:+UnlockExperimentalVMOptions -XX:+UseCGroupMemoryLimitForHeap -XX:MaxRAMFraction=1 -XshowSettings:vm"
EXPOSE 8080
ARG DB_PASSWORD
ARG DB_URL
ARG GOOGLE_CLIENT_ID
ARG GOOGLE_CLIENT_SECRET
ARG JWT_SECRET
ARG KAKAO_CLIENT_ID
ARG KAKAO_CLIENT_SECRET
ARG TOKEN_SECRET
ARG S3_ACCESS_KEY
ARG S3_SECRET_KEY
ARG KAKAO_API_ADMIN_KEY
ARG JAR_FILE=build/libs/ssaout-0.0.1-SNAPSHOT.jar

ENV DB_PASSWORD=$DB_PASSWORD
ENV DB_URL=$DB_URL
ENV GOOGLE_CLIENT_ID_KEY=$GOOGLE_CLIENT_ID
ENV GOOGLE_CLIENT_SECRET_KEY=$GOOGLE_CLIENT_SECRET
ENV JWT_SECRET_KEY=$JWT_SECRET
ENV KAKAO_CLIENT_ID_KEY=$KAKAO_CLIENT_ID
ENV KAKAO_CLIENT_SECRET_KEY=$KAKAO_CLIENT_SECRET
ENV AUTHENTICATION_TOKEN_SECRET=$TOKEN_SECRET
ENV S3_ACCESS_KEY=$S3_ACCESS_KEY
ENV S3_SECRET_KEY=$S3_SECRET_KEY
ENV KAKAO_API_ADMIN_KEY=$KAKAO_API_ADMIN_KEY

COPY ${JAR_FILE} ssaout.jar
ENTRYPOINT ["java","-Duser.timezone=\"Asia/Seoul\"","-jar","/ssaout.jar","&"]
```