포팅 메뉴얼

날짜	@2023년 10월 5일
사람	김승규

FNS 포팅 메뉴얼

1. Environment

1.1. Front-end

- Node.js 18.17.1
- React 18.2.0
  - react-redux 8.1.2
  - react-router-dom 6.15.0
  - react-scripts 5.0.1
- axios 1.5.0
- chart.js 4.4.0
- redux 4.2.1
  - redux-persist 6.0.0

1.2. Back-end

- Java OpenJDK 17
- Spring boot 2.7.15
  - Spring Data JPA

    - Spring Security
    - Lombok
- gradle 8.1.1
- Python 3.11
- FastAPI
    - SQLAlchemy 2.0.21

1.3. Server

- Ubuntu 20.04 LTS
- Docker 24.0.6
- docker compose 2.21.0
- nginx 1.18.0(ubuntu)

1.4. Database

- MySQL 8.0
- Redis
- H2

1.5. IDE

- Visual Studio Code
- IntelliJ IDEA
- PyCharm
- DBeaver

1.6. Version Control

- Git
- GitLab

2. EC2 Setting

2.1. 접속

- .pem 키가 있는 디렉토리에서 아래 커맨드 입력

```
ssh -i J9A403T.pem ubuntu@j9a403.p.ssafy.io
```

2.2. 도커 설치

```
# 도커 설치 ========================
# 1. 패키지 업뎃
sudo apt-get update

sudo apt-get install ca-certificates curl gnupg

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

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

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

sudo docker run hello-world

# sudo 없이 도커 실행
# 7. 현재 사용자를 docker group에 포함
sudo usermod -aG docker ${USER}

# 8. 터미널 재시작
# 껐다 키기
```

2.3. docker-compose.yml 파일 작성

```
version: "3.0"
services:
  jenkins:
    build: .
    image: custom-jenkins:latest
    user: root
    ports:
      - 8080:8080
    volumes:
      - /jenkins:/var/jenkins_home
      - /var/run/docker.sock:/var/run/docker.sock
    depends_on:
      - redis

  nginx:
    image: nginx
    ports:
      - 80:80
      - 443:443
    volumes:
      - ./proxy/nginx.conf:/etc/nginx/nginx.conf
      - /etc/letsencrypt:/etc/letsencrypt
      - ./data/certbot/www:/var/www/certbot
    depends_on:
      - jenkins
  redis:
    image: redis:latest
    ports:
      - 6379:6379
    volumes:
      - ./redis/data:/data
      - ./redis/conf/redis.conf:/usr/local/conf/redis.conf
    labels:
      - "name=redis"
      - "mode=standalone"
    restart: always
    command: redis-server /usr/local/conf/redis.conf
```

2.4. Jenkins Dockerfile 작성

```
FROM jenkins/jenkins:lts

USER root

RUN apt-get update && \
    apt-get install -y python3 python3-pip python3-venv pkg-config libmariadb-dev && \
    rm -rf /var/lib/apt/lists/*
```

- Python3 배포를 위해 Jenkins 커스터마이징

2.5. docker compose 실행

```
# up 커맨드
docker compose up -d

# down 커맨드
docker compose down

# docker container 목록 확인
docker ps
docker ps -a
```

```
# docker container 로그 확인
docker logs 컨테이너이름
```

2.6. nginx.conf 작성

```
user  nginx;
worker_processes  auto;
error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;
events {
    worker_connections  1024;
}
http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;
    upstream server {
        server 13.124.188.144:8081;
        keepalive 1024;
    }

    upstream client {
        server 13.124.188.144:3000;
    }

    server {
        listen 80;

        location /api/ {
            proxy_pass  http://13.124.188.144:8081;
        }

        location / {
            proxy_pass  http://13.124.188.144:3000;
            proxy_connect_timeout  300s;
            proxy_read_timeout 600s;
            proxy_send_timeout 600s;
            proxy_buffers 8 16k;
            proxy_buffer_size 32k;
        }

        location = /favicon.ico {
            return 204;
            access_log     off;
            log_not_found  off;
        }
    }

    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;
    keepalive_timeout  65;
    include /etc/nginx/conf.d/*.conf;
}
```

3. Jenkinsfile & Dockerfile

3.1. Back-end(Java) Jenkinsfile

```
pipeline{
    agent any

    tools {
        gradle 'gradle-8.1.1'
        jdk 'jdk-17'
        dockerTool 'docker'
    }

    stages{
        stage("Clear current directory"){
            steps{
                sh'''
                    rm -rf *
                '''
            }
        }

        stage('Pull from GitLab') {
            steps {
                git url: 'https://lab.ssafy.com/s09-bigdata-recom-sub2/S09P22A403.git',
                    branch: 'be/develop',
                    credentialsId: 'a432e361-21de-400a-a3c2-8f8860f53b7f'
            }
        }

        stage('Apply application.yml files') {
            steps {
                withCredentials([file(credentialsId: 'application-secret', variable: 'secretFile')]) {
                    script {
                        sh 'cp $secretFile backend/fns/src/main/resources/application-secret.yml'
                    }
                }
            }
        }

        stage('Build Backend') {
            steps {
                dir('backend/fns') {
                    sh'''
                        gradle wrapper
                        chmod +x gradlew
                        ./gradlew clean build -x test --stacktrace
                    '''
                }
            }
        }

        stage('Delete existing Docker images and containers') {
            steps {
                sh'''
                    if docker container inspect fns_server >/dev/null 2>&1; then
                        echo "container exists locally"
                        docker stop fns_server
                        docker rm fns_server
                    else
                        echo "container does not exist locally"
                    fi
                    if docker image inspect server >/dev/null 2>&1; then
                        echo "Image exists locally"
                        docker rmi server
                    else
                        echo "Image does not exist locally"
                    fi
                '''
            }
        }

        stage('Build and Deploy Docker') {
            steps {
                dir('backend/fns') {
                    sh'''
                        echo [BE] Build Docker Image!
                        docker build -t server .
                        echo [BE] Run Docker Container!
                        docker run -dp 8081:8081 --name fns_server server
                    '''
                }
            }
        }

    }
    post {
        success {
            script {
                def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: true).trim()
                def Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout: true).trim()
                def GIT_COMMIT_MSG = sh(script: 'git log -1 --pretty=%B ${GIT_COMMIT}', returnStdout: true).trim()
                mattermostSend(color: 'good', message: "✅ 빌드 & 배포 성공: ${env.JOB_NAME} (<${env.BUILD_URL}|#${env.BUILD_NUMBER}>)\n브
            }
        }
        failure {
            script {
                def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: true).trim()
                def Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout: true).trim()
                def GIT_COMMIT_MSG = sh(script: 'git log -1 --pretty=%B ${GIT_COMMIT}', returnStdout: true).trim()
                mattermostSend(color: 'danger', message: "❌ 빌드 & 배포 실패: ${env.JOB_NAME} (<${env.BUILD_URL}|#${env.BUILD_NUMBER}>)\
            }
        }
    }
}
```

3.2. Back-end(Python) Jenkinsfile

```
pipeline {
    agent any

    tools {
        dockerTool 'docker'
    }

    stages {
        stage("Clear current directory") {
            steps {
                sh 'rm -rf *'
            }
        }

        stage('Pull from GitLab') {
            steps {
                git url: 'https://lab.ssafy.com/s09-bigdata-recom-sub2/S09P22A403.git',
                    branch: 'rec/develop',
                    credentialsId: 'a432e361-21de-400a-a3c2-8f8860f53b7f'
            }
        }
        stage('Setup Python Environment') {
            steps {
                sh 'python3 -m venv myenv'

                sh '. myenv/bin/activate'
            }
        }

        stage('Install Dependencies') {
            steps {
                sh 'ls -al'

                sh 'myenv/bin/pip install -r recommend/requirements.txt'
            }
        }

        stage('Delete existing Docker images and containers') {
            steps {
                sh '''
                    if docker container inspect fastapi_server >/dev/null 2>&1; then
                        echo "container exists locally"
                        docker stop fastapi_server
                        docker rm fastapi_server
                    else
                        echo "container does not exist locally"
                    fi
                    if docker image inspect fastapi_image >/dev/null 2>&1; then
                        echo "Image exists locally"
                        docker rmi fastapi_image
                    else
                        echo "Image does not exist locally"
                    fi
                '''
            }
        }

        stage('Build and Deploy Docker') {
            steps {
                sh '''
                    echo [BE] Build Docker Image!
                    docker build -t fastapi_image -f recommend/Dockerfile .
                    echo [BE] Run Docker Container!
                    docker run -dp 8083:8083 --name fastapi_server fastapi_image
                '''
            }
        }
    }

    post {
        success {
            script {
                def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: true).trim()
                def Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout: true).trim()
                def GIT_COMMIT_MSG = sh(script: 'git log -1 --pretty=%B ${GIT_COMMIT}', returnStdout: true).trim()
                mattermostSend(color: 'good', message: "✅ Build & Deployment succeeded: ${env.JOB_NAME} (<${env.BUILD_URL}|#${env.BUI
            }
        }
        failure {
            script {
                def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: true).trim()
                def Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout: true).trim()
                def GIT_COMMIT_MSG = sh(script: 'git log -1 --pretty=%B ${GIT_COMMIT}', returnStdout: true).trim()
                mattermostSend(color: 'danger', message: "❌ Build & Deployment failed: ${env.JOB_NAME} (<${env.BUILD_URL}|#${env.BUILI
            }
        }
    }
}
```

3.3. Front-end Jenkinsfile

```
pipeline {

    agent any

    tools {
        nodejs 'node-18'
        dockerTool 'docker'
    }


    stages {
        stage('Clear current directory') {
            steps {
                sh'''
                    rm -rf *
                '''
            }
        }

        stage('Pull from GitLab') {
            steps {
                git url: 'https://lab.ssafy.com/s09-bigdata-recom-sub2/S09P22A403.git',
                    branch: 'fe/develop',
                    credentialsId: 'a432e361-21de-400a-a3c2-8f8860f53b7f'
            }
        }

        stage('Build Frontend') {
            steps {
                dir('frontend/React') {
                    sh'''
                        npm install
                        npm run build
                    '''
                }
            }
        }

        stage('Delete existing Docker images and containers') {
            steps {
                sh'''
                    if docker container inspect fns_client >/dev/null 2>&1; then
                        echo "container exists locally"
                        docker stop fns_client
                        docker rm fns_client
                    else
                        echo "container does not exist locally"
                    fi
                    if docker image inspect client >/dev/null 2>&1; then
                        echo "Image exists locally"
                        docker rmi client
                    else
                        echo "Image does not exist locally"
                    fi
                '''
            }
        }

        stage('Build and Deploy Docker') {
            steps {
                dir('frontend/React') {
                    sh'''
                        echo [FE] Build Docker Image!
                        docker build -t client .
                        echo [FE] Run Docker Container!
                        docker run -dp 3000:3000 --name fns_client client
                    '''
                }
            }
        }
    }

    post {
        success {
            script {
                def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: true).trim()
                def Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout: true).trim()
                def GIT_COMMIT_MSG = sh(script: 'git log -1 --pretty=%B ${GIT_COMMIT}', returnStdout: true).trim()
                mattermostSend(color: 'good', message: "✅ 빌드 & 배포 성공: ${env.JOB_NAME} (<${env.BUILD_URL}|#${env.BUILD_NUMBER}>)\n
            }
        }
        failure {
            script {
                def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: true).trim()
                def Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout: true).trim()
                def GIT_COMMIT_MSG = sh(script: 'git log -1 --pretty=%B ${GIT_COMMIT}', returnStdout: true).trim()
                mattermostSend(color: 'danger', message: "❌ 빌드 & 배포 실패: ${env.JOB_NAME} (<${env.BUILD_URL}|#${env.BUILD_NUMBER}>)\
            }
        }
    }
}
```

3.4. Back-end(Java) Dockerfile

```
FROM openjdk:17-alpine
ARG JAR_FILE=build/libs/*-SNAPSHOT.jar
COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java","-Dspring.profiles.active=prod","-jar","/app.jar"]
```

3.5. Back-end(Python) Dockerfile

```
FROM python:3.9WORKDIR /app
COPY recommend ./recommend
RUN pip install --no-cache-dir -r recommend/requirements.txt
EXPOSE 8083
CMD ["uvicorn", "recommend.app:app", "--host", "0.0.0.0", "--port", "8083"]
```

3.6. Front-end Dockerfile

```
FROM nginx
WORKDIR /app
RUN mkdir ./build
ADD ./build ./build
RUN rm -rf /etc/nginx/nginx.conf
COPY ./react-nginx.conf /etc/nginx/nginx.conf
EXPOSE 3000
CMD ["nginx", "-g", "daemon off;"]
```

4. Jenkins Dashboard

- `domain:8080` 으로 접속

4.1. Plugins

- Gradle Plugin
- Docker
- Gitlab
  - Gitlab API
  - Gitlab Authentication
  - Gitlab Branch Source
  - Gitlab Merge Request Builder

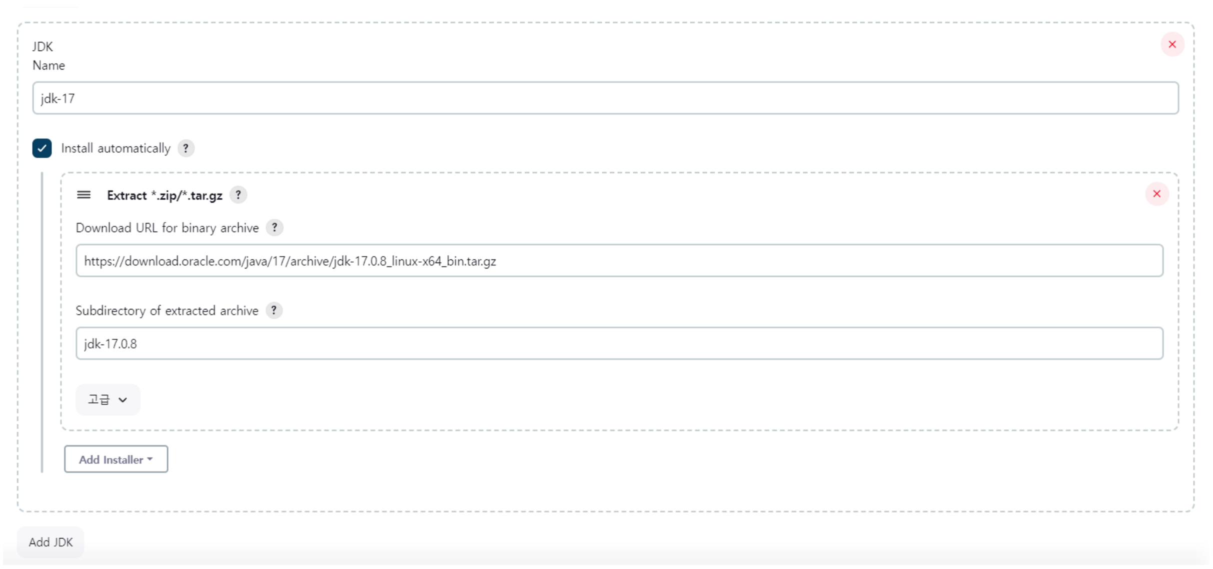
- Generic Webhook Trigger
- NodeJs

4.2. Credentials

- Token
  - GitLab API token
  - GitLab Personal Access Token
- Dependency
  - application-prod.yml
  - application-secret.yml
  - settings.py

4.3. Tools

- JDK



JDK
Name
jdk-17

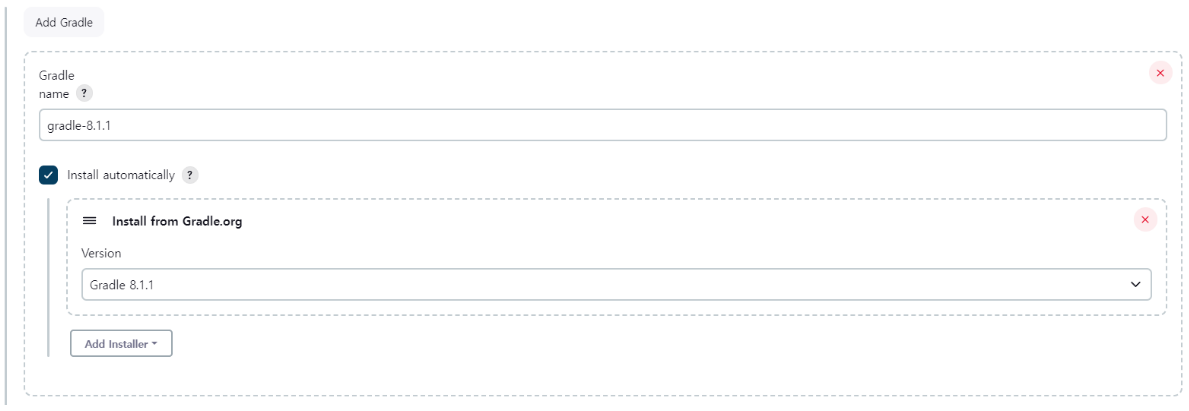
Install automatically

Extract *.zip/*.tar.gz

Download URL for binary archive
https://download.oracle.com/java/17/archive/jdk-17.0.8_linux-x64_bin.tar.gz

Subdirectory of extracted archive
jdk-17.0.8

- Gradle



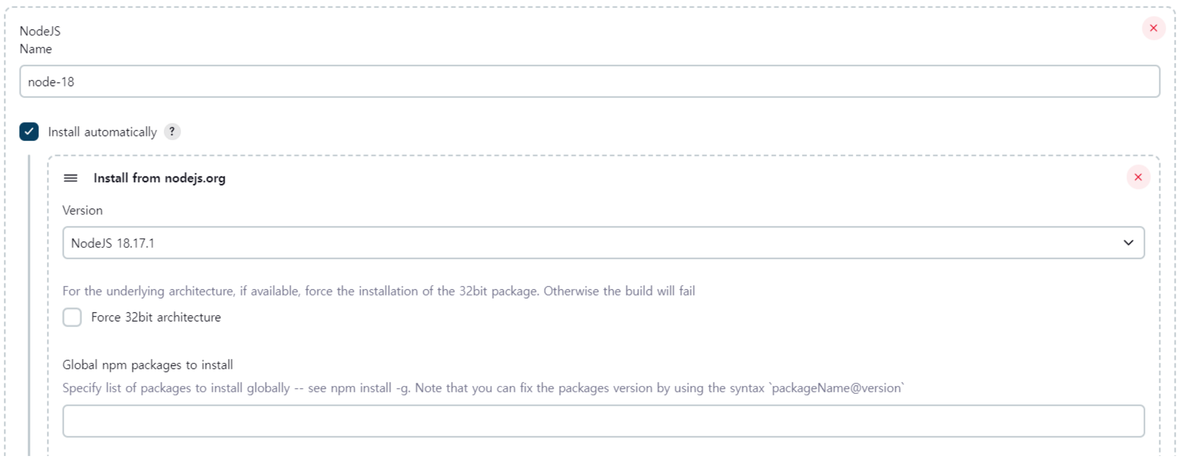
Gradle
name
gradle-8.1.1

Install automatically

Install from Gradle.org

Version
Gradle 8.1.1

- NodeJS


NodeJS
Name
node-18

✓ Install automatically

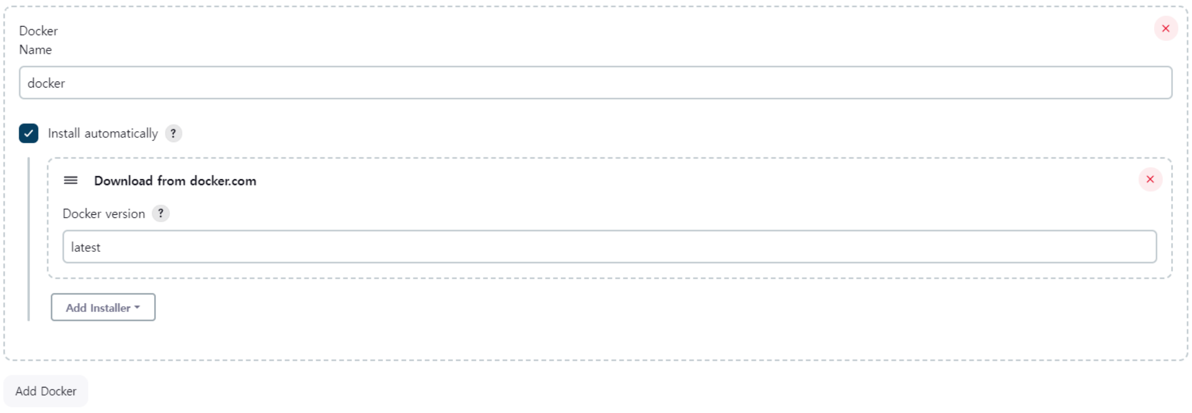
Install from nodejs.org

Version
NodeJS 18.17.1

For the underlying architecture, if available, force the installation of the 32bit package. Otherwise the build will fail
☐ Force 32bit architecture

Global npm packages to install
Specify list of packages to install globally -- see npm install -g. Note that you can fix the packages version by using the syntax `packageName@version`

- Docker


Docker
Name
docker

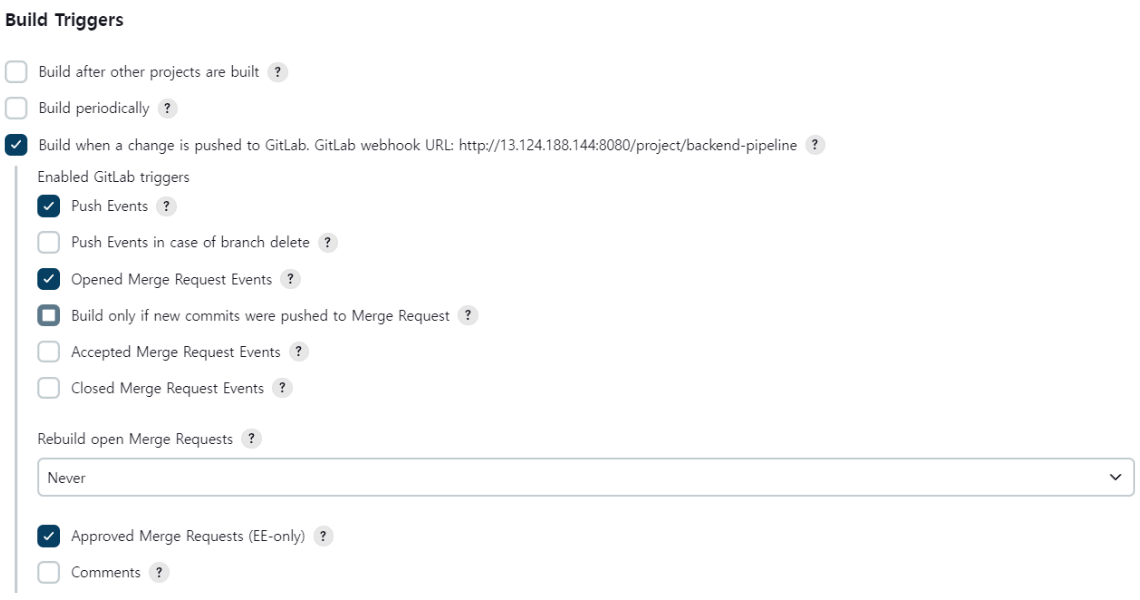
✓ Install automatically

Download from docker.com

Docker version
latest

4.4. Pipelines

- Build Triggers

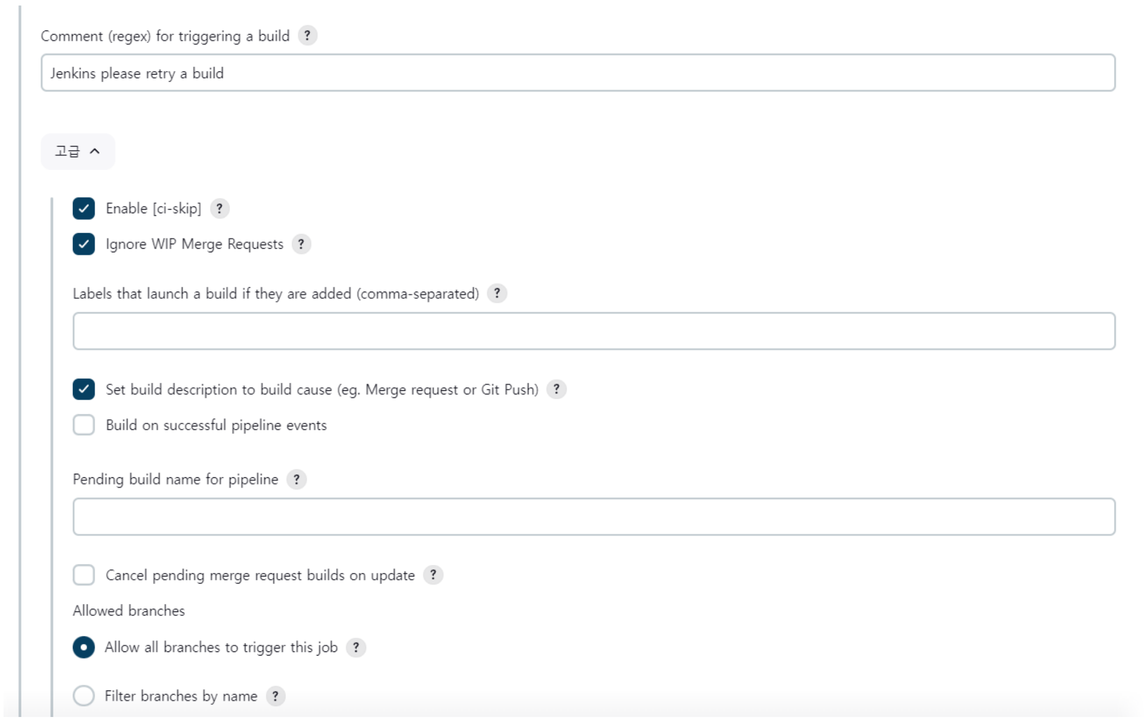

Build Triggers

☐ Build after other projects are built
☐ Build periodically
✓ Build when a change is pushed to GitLab. GitLab webhook URL: http://13.124.188.144:8080/project/backend-pipeline

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

[✓ 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

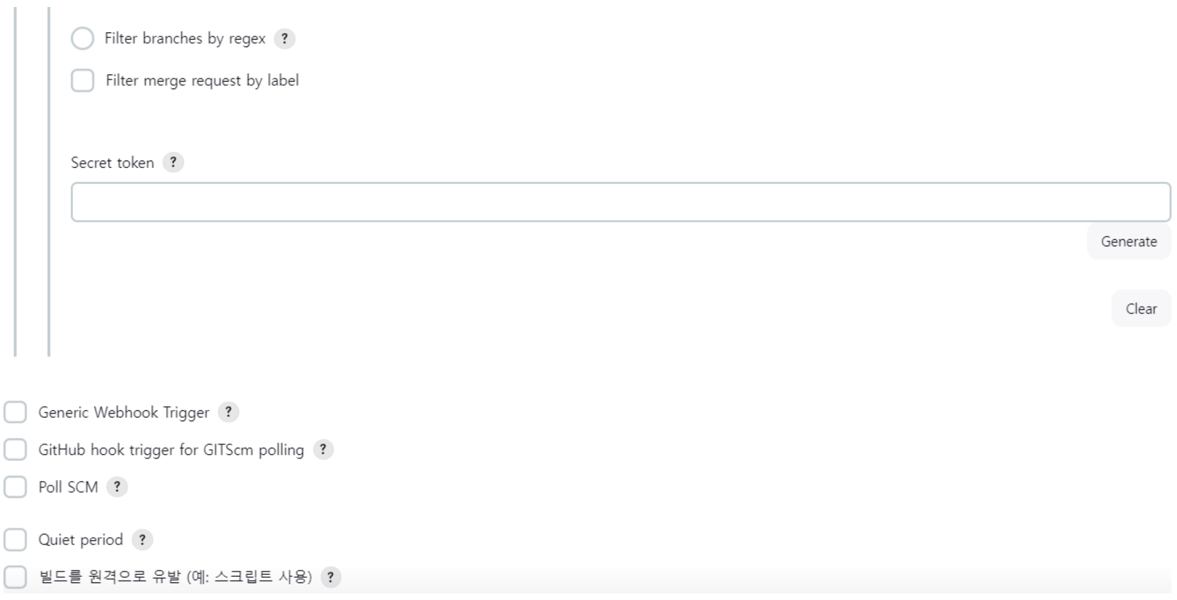
Pending build name for pipeline ?

Cancel pending merge request builds on update ?

Allowed branches

[✓ Allow all branches to trigger this job] ?

Filter branches by name ?



Filter branches by regex ?

Filter merge request by label

Secret token ?

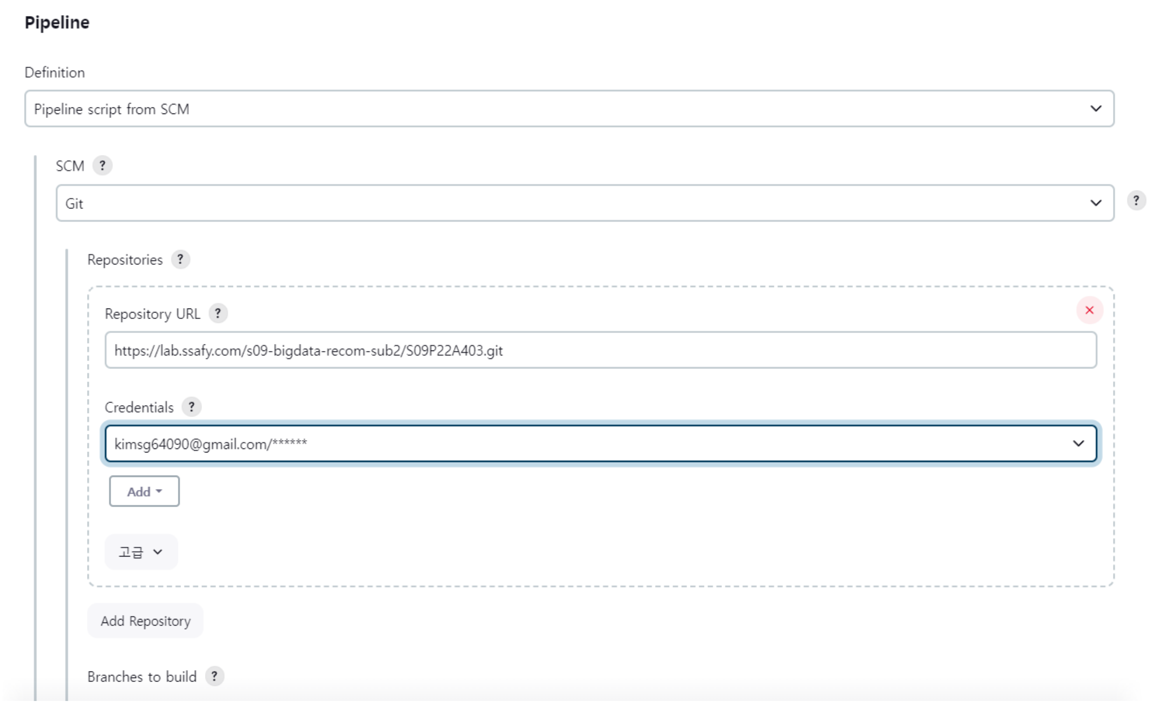
Generic Webhook Trigger ?

GitHub hook trigger for GITScm polling ?

Poll SCM ?

Quiet period ?

빌드를 원격으로 유발 (예: 스크립트 사용) ?



Pipeline

Definition: Pipeline script from SCM

SCM: Git

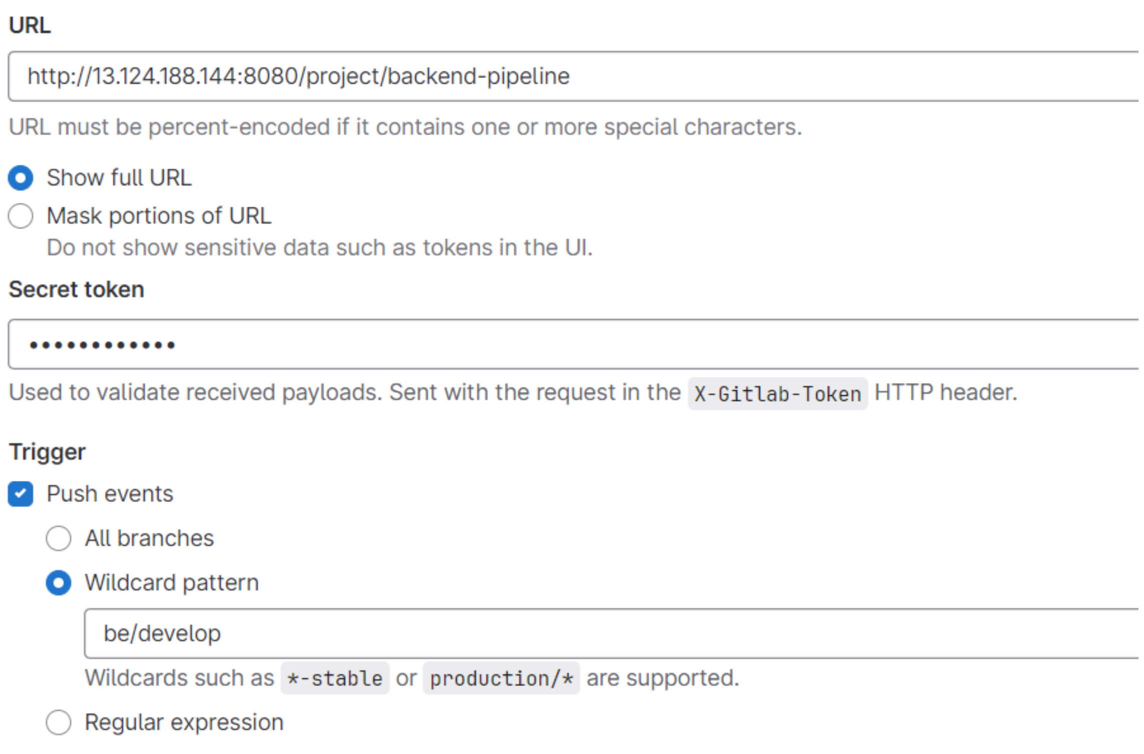
Repositories

Repository URL: https://lab.ssafy.com/s09-bigdata-recom-sub2/S09P22A403.git

Credentials: kimsg64090@gmail.com/******

- 발급받은 URL과 Secret Token을 Gitlab Webhook에 넣기

- GitLab Webhook 연결
  - Settings - Webhooks



URL

http://13.124.188.144:8080/project/backend-pipeline

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]

All branches [✓ Wildcard pattern] Regular expression

be/develop

Wildcards such as `*-stable` or `production/*` are supported.

  - Wildcard pattern에 브랜치 입력

5. Properties