포팅 메뉴얼

[개발환경]

client

Backend

Spring boot	3.3.5
S3	2.2.6
jdk	17
ORM	JPA(Hibernate)
FastAPI	python 3.9-slim
intellij	2024.1.4(ultimate)

Server

Ec2	Ubuntu 20.04LTS
Nginx	1.27.2(Ubuntu)
Jenkins	2.462.3
docker	27.3.1
mysql	9.1.0

1. docker install

공식문서 참고 : https://docs.docker.com/engine/install/ubuntu/

```
sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gp
g -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/ap
t/keyrings/docker.asc] https://download.docker.com/linux/ub
untu \
  $(. /etc/os-release && echo "$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
```

2. 전체 docker-compose 코드

```
version: '3.3'

services:
  jenkins:
    build:
      context: /home/ubuntu/data/jenkins
      dockerfile: Dockerfile
    container_name: jenkins
    user: root
    environment:
      - TZ=Asia/Seoul
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - /home/ubuntu/data/jenkins:/var/jenkins_home
    ports:
      - '8000:8080'
    restart: on-failure
    networks:
      - scare_network

  redis:
    image: redis:latest
    container_name: redis
    hostname: redis
    command: redis-server --port 6379
    ports:
      - "6379:6379"
    volumes:
```

```
      - ./redis/redis_data:/data
      - ./redis:/usr/local/etc/redis/redis.conf
    networks:
      - scare_network

  nginx:
    image: nginx:latest
    container_name: nginx
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - ./nginx/nginx.conf:/etc/nginx/nginx.conf:ro
      - ./nginx/data/certbot/conf:/etc/letsencrypt
      - ./nginx/data/certbot/www:/var/www/certbot
    depends_on:
      - certbot
    networks:
      - scare_network

  certbot:
    image: certbot/certbot
    volumes:
      - ./nginx/data/certbot/conf:/etc/letsencrypt
      - ./nginx/data/certbot/www:/var/www/certbot

  mysql:
    image: mysql:latest
    restart: always
    container_name: mysql
    environment:
      MYSQL_ROOT_PASSWORD: rootpassword
      MYSQL_DATABASE: bugtopia
      MYSQL_USER: bugtopia
      MYSQL_PASSWORD: bugtopia123456
      TZ: Asia/Seoul
    ports:
      - "3306:3306"
```

```
    volumes:
      - ./mysql:/var/lib/mysql
    networks:
      - scare_network
    command:
      - --character-set-server=utf8mb4
      - --collation-server=utf8mb4_unicode_ci

networks:
  scare_network:
    external: true
```

3.Nginx

3-1. Nginx, letsencrypt, cerbot install

SSL 인증서 설치

```
## 설치
sudo apt-get install letsencrypt
sudo apt install certbot python3-certbot-nginx

# nginx 연결
sudo certbot --nginx -d j11a205.p.ssafy.io

# docker 이용시
## docker compose 명령어 nginx 적혀있는 곳에서 실행해서 key 발급 e
mail 등록 필수
sudo docker compose run --rm certbot certonly --webroot --w
ebroot-path=/var/www/certbot -d k11a303.p.ssafy.io
```

3-2. Nginx conf

```
user nginx;
worker_processes auto;
```

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

events {
    worker_connections 1024;
    multi_accept on;
}

http {
    include /etc/nginx/mime.types;
    default_type application/octet-stream;
    log_format main '$remote_addr - $remote_user [$time_local
                    '$status $body_bytes_sent "$http_referer"
                    '"$http_user_agent" "$http_x_forwarded_fo
    access_log /var/log/nginx/access.log main;
    sendfile on;
    keepalive_timeout 65;

    upstream backend {
        server backend:8080; # Spring Boot 애플리케이션 서버 주소
    }

    upstream fastapi {
        server fastapi:8001; # FastAPI 애플리케이션 서버 주소
    }

    map $http_upgrade $connection_upgrade {
        default upgrade;
        '' close;
    }

    server {
        listen 80;
        server_name k11a303.p.ssafy.io;

        location /.well-known/acme-challenge/ {
            root /var/www/certbot;
        }
```

```
        location / {
            return 301 https://$server_name$request_uri;
        }
    }

    server {
        listen 443 ssl http2;
        server_name k11a303.p.ssafy.io;

        ssl_certificate /etc/letsencrypt/live/k11a303.p.ssafy
        ssl_certificate_key /etc/letsencrypt/live/k11a303.p.s

        # Spring Boot Backend reverse proxy
        location /api {
            proxy_pass http://backend;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_for
            proxy_set_header X-Forwarded-Proto $scheme;
            proxy_buffering off;
            error_page 502 503 504 = @backend_unavailable;
        }

        # FastAPI reverse proxy
        location /fastapi {
            proxy_pass http://fastapi;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_for
            proxy_set_header X-Forwarded-Proto $scheme;
            proxy_buffering off;
            error_page 502 503 504 = @backend_unavailable;
        }

        location @backend_unavailable {
            return 503 '{"error": "Backend service is current
        }
```

```
    }
}
```

3-3. docker-compose.yml

```
  nginx:
    image: nginx:latest
    container_name: nginx
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - ./nginx/nginx.conf:/etc/nginx/nginx.conf:ro
      - ./nginx/data/certbot/conf:/etc/letsencrypt
      - ./nginx/data/certbot/www:/var/www/certbot
    depends_on:
      - certbot
    networks:
      - scare_network

  certbot:
    image: certbot/certbot
    volumes:
      - ./nginx/data/certbot/conf:/etc/letsencrypt
      - ./nginx/data/certbot/www:/var/www/certbot
```

4. Jenkins

4-1. Jenkins docker file

```
FROM jenkins/jenkins:lts
USER root
# 필요한 패키지 설치 및 Docker 설정
RUN apt-get update && apt-get install -y \
    ca-certificates \
    curl \
```

```
    gnupg \
    lsb-release \
    git \
    unzip \
    xz-utils \
    libglu1-mesa \
    && rm -rf /var/lib/apt/lists/*

# Docker의 공식 GPG 키 추가
RUN install -m 0755 -d /etc/apt/keyrings \
    && curl -fsSL https://download.docker.com/linux/debian/gp
    && chmod a+r /etc/apt/keyrings/docker.asc

# Docker 리포지토리를 APT 소스에 추가
RUN echo \
    "deb [arch=$(dpkg --print-architecture) signed-by=/etc/ap
    $(. /etc/os-release && echo "$VERSION_CODENAME") stable"
    tee /etc/apt/sources.list.d/docker.list > /dev/null

# Docker 패키지 설치
RUN apt-get update && apt-get install -y \
    docker-ce-cli \
    docker-buildx-plugin \
    docker-compose-plugin \
    && rm -rf /var/lib/apt/lists/*

# jenkins 사용자를 docker 그룹에 추가
RUN groupadd -f docker && usermod -aG docker jenkins
```

4-2. docker-compose.yml

```
  jenkins:
    build:
      context: /home/ubuntu/data/jenkins
      dockerfile: Dockerfile
    container_name: jenkins
    user: root
```

```
    environment:
      - TZ=Asia/Seoul
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - /home/ubuntu/data/jenkins:/var/jenkins_home
    ports:
      - '8000:8080'
    restart: on-failure
    networks:
      - scare_network
```

4-3. Jenkins Script

```
pipeline {
   agent any

   environment {
       BACKEND_IMAGE_NAME = "scare-backend"
       BACKEND_CONTAINER_NAME = "backend"
       FASTAPI_IMAGE_NAME = "bugtopia-ai"
       FASTAPI_CONTAINER_NAME = "fastapi"
       NETWORK_NAME = "scare_network"
   }

   stages {
       stage('Checkout') {
           steps {
               dir('./') {
                   git branch: 'develop', url: 'https://lab.s
               }
               script {
                   GIT_AUTHOR_ID = sh(script: "git show -s --
                   GIT_AUTHOR_NAME = sh(script: "git show -s
               }
           }
       }
```

```
        stage('Copy env') {
            steps {
                dir('backend/bugar/src/main/resources') {
                    withCredentials([file(credentialsId: 'back
                        sh 'cp $YML_FILE application-private.y
                    }
                }
            }
        }

        stage('Build Backend Docker Image') {
            steps {
                dir('./backend/bugar') {
                    sh 'docker build -t ${BACKEND_IMAGE_NAME}:
                }
            }
        }

        stage('Deploy Backend') {
            steps {
                script {
                    sh "docker stop ${BACKEND_CONTAINER_NAME}
                    sh "docker rm ${BACKEND_CONTAINER_NAME} ||
                    sh "docker image prune -f"
                    sh """
                    docker run -d \
                        --name ${BACKEND_CONTAINER_NAME} \
                        --network ${NETWORK_NAME} \
                        --restart=unless-stopped \
                        -p 8080:8080 \
                        ${BACKEND_IMAGE_NAME}:${BUILD_NUMBER}
                    """
                }
            }
        }

        stage('Build FastAPI Docker Image') {
            steps {
```

```
                dir('./bugtopia_ai') {
                    sh 'docker build -t ${FASTAPI_IMAGE_NAME}:
                }
            }
        }

        stage('Deploy FastAPI') {
            steps {
                script {
                    sh "docker stop ${FASTAPI_CONTAINER_NAME}
                    sh "docker rm ${FASTAPI_CONTAINER_NAME} ||
                    sh """
                    docker run -d \
                        --name ${FASTAPI_CONTAINER_NAME} \
                        --network ${NETWORK_NAME} \
                        --restart=unless-stopped \
                        -p 8001:80 \
                        ${FASTAPI_IMAGE_NAME}:${BUILD_NUMBER}
                    """
                }
            }
        }
    }

    post {
        success {
            script {
                mattermostSend(
                    color: 'good',
                    message: "빌드 성공: ${env.JOB_NAME} #${env.
                    endpoint: 'https://meeting.ssafy.com/hooks
                    channel: 'A303_build_result'
                )
            }
        }

        failure {
            script {
```

```
                mattermostSend(
                    color: 'danger',
                    message: "빌드 실패: ${env.JOB_NAME} #${env.
                    endpoint: 'https://meeting.ssafy.com/hooks
                    channel: 'A303_build_result'
                )
            }
        }

        always {
            cleanWs()
        }
    }
}
```

4-4. Jenkins plugins

▼ plugin 리스트

```
{
  "name": "ionicons-api",
  "version": "74.v93d5eb_813d5f"
}
{
  "name": "cloudbees-folder",
  "version": "6.955.v81e2a_35c08d3"
}
{
  "name": "antisamy-markup-formatter",
  "version": "162.v0e6ec0fcfcf6"
}
{
  "name": "asm-api",
  "version": "9.7.1-97.v4cc844130d97"
}
{
  "name": "json-path-api",
```

```
  "version": "2.9.0-118.v7f23ed82a_8b_8"
}
{
  "name": "structs",
  "version": "338.v848422169819"
}
{
  "name": "workflow-step-api",
  "version": "678.v3ee58b_469476"
}
{
  "name": "token-macro",
  "version": "400.v35420b_922dcb_"
}
{
  "name": "build-timeout",
  "version": "1.33"
}
{
  "name": "bouncycastle-api",
  "version": "2.30.1.78.1-248.ve27176eb_46cb_"
}
{
  "name": "credentials",
  "version": "1384.vf0a_2ed06f9c6"
}
{
  "name": "plain-credentials",
  "version": "183.va_de8f1dd5a_2b_"
}
{
  "name": "variant",
  "version": "60.v7290fc0eb_b_cd"
}
{
  "name": "ssh-credentials",
  "version": "343.v884f71d78167"
}
```

```
{
  "name": "credentials-binding",
  "version": "681.vf91669a_32e45"
}
{
  "name": "scm-api",
  "version": "696.v778d637b_a_762"
}
{
  "name": "workflow-api",
  "version": "1336.vee415d95c521"
}
{
  "name": "commons-lang3-api",
  "version": "3.17.0-84.vb_b_938040b_078"
}
{
  "name": "timestamper",
  "version": "1.27"
}
{
  "name": "caffeine-api",
  "version": "3.1.8-133.v17b_1ff2e0599"
}
{
  "name": "script-security",
  "version": "1367.vdf2fc45f229c"
}
{
  "name": "javax-activation-api",
  "version": "1.2.0-7"
}
{
  "name": "jaxb",
  "version": "2.3.9-1"
}
{
  "name": "snakeyaml-api",
```

```
  "version": "2.3-123.v13484c65210a_"
}
{
  "name": "json-api",
  "version": "20240303-101.v7a_8666713110"
}
{
  "name": "jackson2-api",
  "version": "2.17.0-379.v02de8ec9f64c"
}
{
  "name": "commons-text-api",
  "version": "1.12.0-129.v99a_50df237f7"
}
{
  "name": "workflow-support",
  "version": "930.vf51d22b_ce488"
}
{
  "name": "plugin-util-api",
  "version": "5.1.0"
}
{
  "name": "font-awesome-api",
  "version": "6.6.0-2"
}
{
  "name": "bootstrap5-api",
  "version": "5.3.3-1"
}
{
  "name": "jquery3-api",
  "version": "3.7.1-2"
}
{
  "name": "echarts-api",
  "version": "5.5.1-4"
}
```

```
{
  "name": "display-url-api",
  "version": "2.204.vf6fddd8a_8b_e9"
}
{
  "name": "checks-api",
  "version": "2.2.1"
}
{
  "name": "junit",
  "version": "1307.vdd5b_2646279e"
}
{
  "name": "matrix-project",
  "version": "839.vff91cd7e3a_b_2"
}
{
  "name": "resource-disposer",
  "version": "0.24"
}
{
  "name": "ws-cleanup",
  "version": "0.47"
}
{
  "name": "ant",
  "version": "511.v0a_a_1a_334f41b_"
}
{
  "name": "okhttp-api",
  "version": "4.11.0-172.vda_da_1feeb_c6e"
}
{
  "name": "durable-task",
  "version": "577.v2a_8a_4b_7c0247"
}
{
  "name": "workflow-durable-task-step",
```

```
  "version": "1371.vb_7cec8f3b_95e"
}
{
  "name": "workflow-scm-step",
  "version": "427.v4ca_6512e7df1"
}
{
  "name": "workflow-cps",
  "version": "3990.vd281dd77a_388"
}
{
  "name": "workflow-job",
  "version": "1436.vfa_244484591f"
}
{
  "name": "jakarta-activation-api",
  "version": "2.1.3-1"
}
{
  "name": "jakarta-mail-api",
  "version": "2.1.3-1"
}
{
  "name": "apache-httpcomponents-client-4-api",
  "version": "4.5.14-208.v438351942757"
}
{
  "name": "instance-identity",
  "version": "201.vd2a_b_5a_468a_a_6"
}
{
  "name": "mailer",
  "version": "488.v0c9639c1a_eb_3"
}
{
  "name": "workflow-basic-steps",
  "version": "1058.vcb_fc1e3a_21a_9"
}
```

```
{
  "name": "gradle",
  "version": "2.13.1"
}
{
  "name": "pipeline-milestone-step",
  "version": "119.vdfdc43fc3b_9a_"
}
{
  "name": "pipeline-build-step",
  "version": "540.vb_e8849e1a_b_d8"
}
{
  "name": "pipeline-groovy-lib",
  "version": "744.v5b_556ee7c253"
}
{
  "name": "pipeline-stage-step",
  "version": "312.v8cd10304c27a_"
}
{
  "name": "joda-time-api",
  "version": "2.13.0-93.v9934da_29b_a_e9"
}
{
  "name": "pipeline-model-api",
  "version": "2.2214.vb_b_34b_2ea_9b_83"
}
{
  "name": "pipeline-model-extensions",
  "version": "2.2214.vb_b_34b_2ea_9b_83"
}
{
  "name": "branch-api",
  "version": "2.1197.vfa_d0c47c267d"
}
{
  "name": "workflow-multibranch",
```

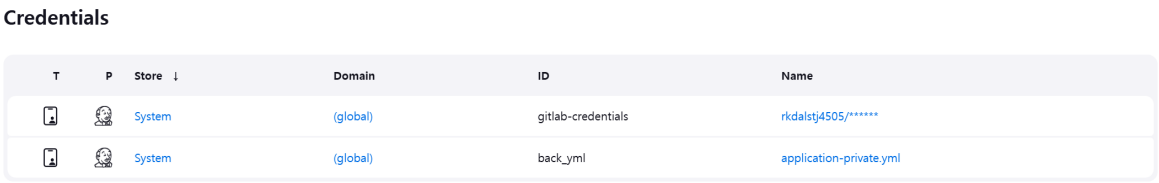
```
  "version": "795.ve0cb_1f45ca_9a_"
}
{
  "name": "pipeline-stage-tags-metadata",
  "version": "2.2214.vb_b_34b_2ea_9b_83"
}
{
  "name": "pipeline-input-step",
  "version": "495.ve9c153f6067b_"
}
{
  "name": "pipeline-model-definition",
  "version": "2.2214.vb_b_34b_2ea_9b_83"
}
{
  "name": "workflow-aggregator",
  "version": "600.vb_57cdd26fdd7"
}
{
  "name": "jjwt-api",
  "version": "0.11.5-112.ve82dfb_224b_a_d"
}
{
  "name": "github-api",
  "version": "1.321-468.v6a_9f5f2d5a_7e"
}
{
  "name": "mina-sshd-api-common",
  "version": "2.14.0-133.vcc091215a_358"
}
{
  "name": "mina-sshd-api-core",
  "version": "2.14.0-133.vcc091215a_358"
}
{
  "name": "gson-api",
  "version": "2.11.0-85.v1f4e87273c33"
}
```

```
{
  "name": "git-client",
  "version": "5.0.0"
}
{
  "name": "git",
  "version": "5.5.2"
}
{
  "name": "github",
  "version": "1.40.0"
}
{
  "name": "github-branch-source",
  "version": "1807.v50351eb_7dd13"
}
{
  "name": "pipeline-github-lib",
  "version": "61.v629f2cc41d83"
}
{
  "name": "pipeline-graph-analysis",
  "version": "216.vfd8b_ece330ca_"
}
{
  "name": "metrics",
  "version": "4.2.21-458.vcf496cb_839e4"
}
{
  "name": "pipeline-graph-view",
  "version": "340.v28cecee8b_25f"
}
{
  "name": "eddsa-api",
  "version": "0.3.0-4.v84c6f0f4969e"
}
{
  "name": "trilead-api",
```

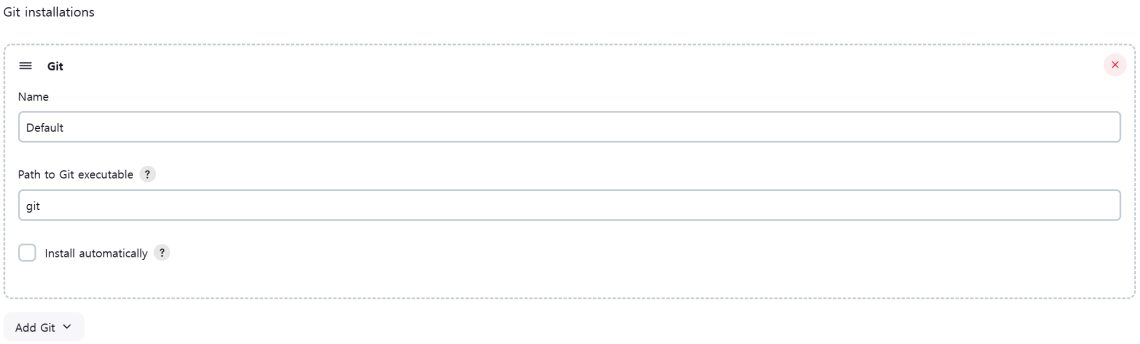
```
  "version": "2.147.vb_73cc728a_32e"
}
{
  "name": "ssh-slaves",
  "version": "2.973.v0fa_8c0dea_f9f"
}
{
  "name": "matrix-auth",
  "version": "3.2.3"
}
{
  "name": "pam-auth",
  "version": "1.11"
}
{
  "name": "ldap",
  "version": "725.v3cb_b_711b_1a_ef"
}
{
  "name": "email-ext",
  "version": "1844.v3ea_a_b_842374a_"
}
{
  "name": "theme-manager",
  "version": "262.vc57ee4a_eda_5d"
}
{
  "name": "dark-theme",
  "version": "479.v661b_1b_911c01"
}
{
  "name": "jersey2-api",
  "version": "2.44-151.v6df377fff741"
}
{
  "name": "gitlab-plugin",
  "version": "1.8.2"
}
```

```
{
  "name": "javax-mail-api",
  "version": "1.6.2-10"
}
{
  "name": "sshd",
  "version": "3.330.vc866a_8389b_58"
}
{
  "name": "mattermost",
  "version": "3.1.3"
}
{
  "name": "pipeline-rest-api",
  "version": "2.34"
}
{
  "name": "pipeline-stage-view",
  "version": "2.34"
}
{
  "name": "jdk-tool",
  "version": "80.v8a_dee33ed6f0"
}
{
  "name": "command-launcher",
  "version": "115.vd8b_301cc15d0"
}
{
  "name": "external-monitor-job",
  "version": "215.v2e88e894db_f8"
}
{
  "name": "unity3d-plugin",
  "version": "1.3"
}
```

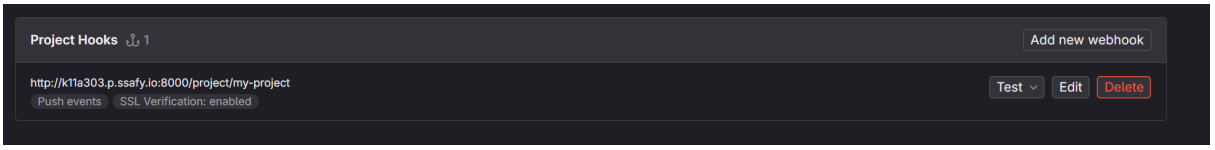
4-5. Jenkins Credentials



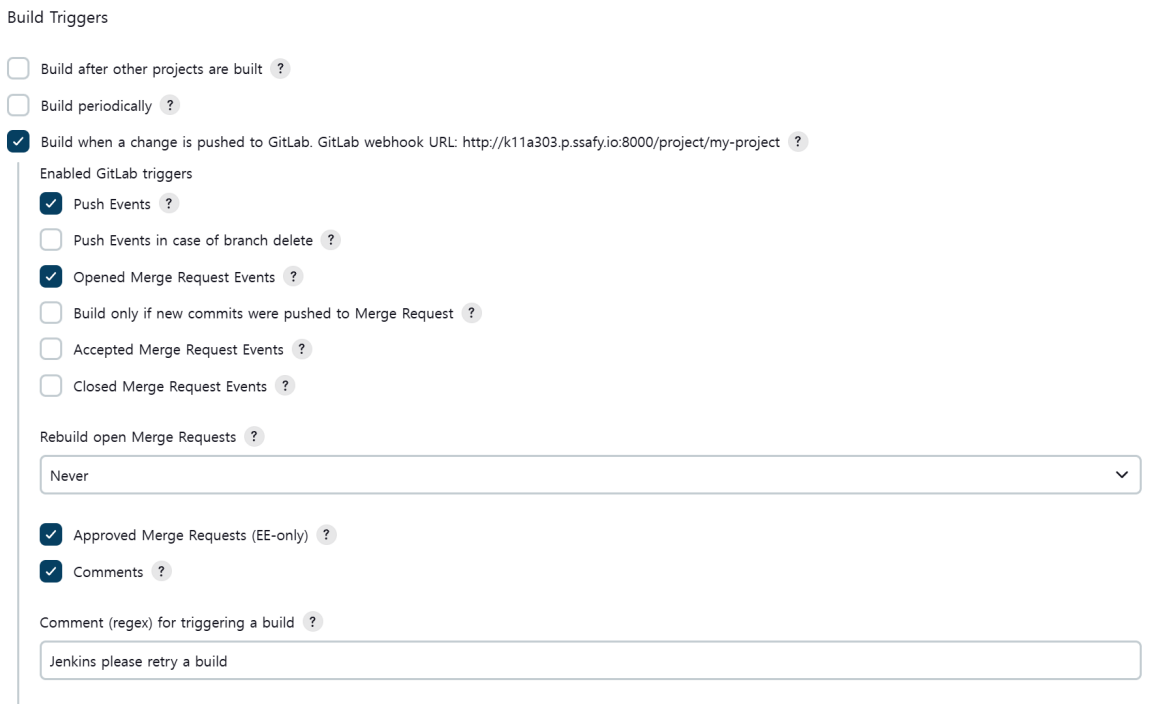
4-6. Tools setting



4-7. webhook setting



4-8. Trigger setting



5. backend server docker file

spring server

```
# 빌드 스테이지
FROM gradle:8.10.2-jdk17 AS build
WORKDIR /app
COPY build.gradle settings.gradle ./
COPY gradle ./gradle

# 의존성 다운로드
RUN gradle dependencies --no-daemon

# 소스 코드 복사
COPY src ./src

# 애플리케이션 빌드 및 JAR 파일 생성
RUN gradle clean build -x test --no-daemon

# 실행 스테이지
FROM openjdk:17-jdk-slim
```

```
WORKDIR /app

# 빌드 스테이지에서 생성된 JAR 파일을 컨테이너로 복사
COPY --from=build /app/build/libs/bugar-0.0.1-SNAPSHOT.jar ./

EXPOSE 8080
ENTRYPOINT ["java", "-jar", "app.jar"]
```

fastapi

```
# Python 3.9 slim 이미지 사용
FROM python:3.9-slim-buster

# 작업 디렉토리 설정
WORKDIR /app

# 시스템 패키지 및 Python 개발 도구 설치
RUN apt-get update && apt-get install -y \
    libgl1-mesa-glx \
    libglib2.0-0 \
    gcc \
    python3-dev \
    && rm -rf /var/lib/apt/lists/*

# pip 업그레이드
RUN pip install --upgrade pip

# requirements.txt 복사 및 패키지 설치
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

# 애플리케이션 코드 복사
COPY ./app ./app

# 모델 파일이 있는 디렉토리 생성 및 복사
RUN mkdir -p /app/app/model
COPY ./app/model/bounding_model.pt /app/app/model/
```

```
COPY ./app/model/prediction_model.pt /app/app/model/
COPY ./app/model/classes.txt /app/app/model/

# Python 경로 설정
ENV PYTHONPATH="/app:${PYTHONPATH}"

# 포트 80 노출
EXPOSE 8001

# FastAPI 앱 실행
CMD ["uvicorn", "app.main:app", "--host", "0.0.0.0", "--port"
```

6. mysql

docker compose

```
  mysql:
    image: mysql:latest
    restart: always
    container_name: mysql
    environment:
      MYSQL_ROOT_PASSWORD: rootpassword
      MYSQL_DATABASE: bugtopia
      MYSQL_USER: bugtopia
      MYSQL_PASSWORD: bugtopia123456
      TZ: Asia/Seoul
    ports:
      - "3306:3306"
    volumes:
      - ./mysql:/var/lib/mysql
    networks:
      - scare_network
    command:
      - --character-set-server=utf8mb4
      - --collation-server=utf8mb4_unicode_ci
```

7. 환경변수

application-priavate.yml

```
spring:
  datasource:
    url: jdbc:mysql://k11a303.p.ssafy.io:3306/bugtopia?server
    username: bugtopia
    password: bugtopia123456
    driver-class-name: com.mysql.cj.jdbc.Driver

  jpa:
    hibernate:
      ddl-auto: update
    properties:
      hibernate:
        dialect: org.hibernate.dialect.MariaDBDialect
    show-sql: true


logging:
  level:
    root: INFO
    org.springframework.data.mongodb.core.MongoTemplate: DEBU


firebase:
  server:
    key: "BJuk7MoH__eRKwHE-Li5x1ThQs7dNeRNmm_zOFPIzSlz_fmw48O
    url: "https://fcm.googleapis.com/v1/projects/test-c17e0/m


cloud:
  aws:
    s3:
      bucket: "bugtopia"
    stack.auto: false
    region.static: ap-northeast-2
```

```
    credentials:
      accessKey: "AKIAY5AOGINKTREZ7BDB"
      secretKey: "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"

fastapi:
  base_url: "https://k11a303.p.ssafy.io/fastapi"
```

FastAPI requirements.txt

```
annotated-types==0.7.0
anyio==4.6.2.post1
click==8.1.7
colorama==0.4.6
exceptiongroup==1.2.2
fastapi==0.115.4
h11==0.14.0
httptools==0.6.4
idna==3.10
pydantic==2.9.2
pydantic-core==2.23.4
python-dotenv==1.0.1
sniffio==1.3.1
starlette==0.41.2
typing-extensions==4.12.2
uvicorn==0.32.0
watchfiles==0.24.0
websockets==13.1
addict==2.4.0
certifi==2024.8.30
charset-normalizer==3.4.0
cycler==0.11.0
fonttools==4.38.0
idna==3.10
importlib-metadata==6.7.0
joblib==1.3.2
kiwisolver==1.4.5
matplotlib==3.5.3
```

```
numpy==1.21.6
nvidia-cublas-cu11==11.10.3.66
nvidia-cuda-nvrtc-cu11==11.7.99
nvidia-cuda-runtime-cu11==11.7.99
nvidia-cudnn-cu11==8.5.0.96
opencv-python==4.10.0.84
packaging==24.0
pandas==1.3.5
Pillow==10.3.0
platformdirs==4.0.0
psutil==6.1.0
py-cpuinfo==9.0.0
pyparsing==3.1.4
python-dateutil==2.9.0.post0
pytz==2024.2
PyYAML==6.0.1
requests==2.31.0
scikit-learn==1.0.2
scipy==1.7.3
seaborn==0.12.2
six==1.16.0
threadpoolctl==3.1.0
tomli==2.0.1
torch==1.13.1
torchaudio==0.13.1
torchvision==0.14.1
tqdm==4.66
ultralytics==8.0.145
urllib3==2.0.7
yapf==0.40.2
zipp==3.15.0
gunicorn
```

8.사용 포트

컨테이너	포트 번호
spring	8080
fastapi	8001
jenkins	8000
mysql	3306
nginx	80, 443