포팅메뉴얼

1. 개발환경

1.1. Frontend

- Node JS 20.13.1
- React 18.3.1
- zustand 4.5.5
- Axios 1.7.7
- Tailwind CSS 3.4.12

1.2. Backend

- Database:
  - MySQL Driver: 8.0.33
- AWS S3: AWS Java SDK S3 1.11.1000
- Other Libraries:
  - Guava: 29.0-jre
  - JSON: org.json 20210307
  - Jsoup: 1.7.2
- requirements.txt

```
torch==2.2.1
torchvision==0.17.1
torchaudio==2.2.1 --index-url https://download.pytorch.org
fastapi
uvicorn
numpy
python-multipart
opencv-python==4.8.0.74
torchvision
insightface
onnxruntime-gpu
wandb
python-dotenv
sqlalchemy
mlflow
pymysql
```

1.3. Server

- Ubuntu 20.04 LTS
- Docker-compose 2.6.1
- Nginx 1.27.0
- Docker 27.2.1
- Jenkins 2.452.3
- mlflow
- minio

1.5. Database

- MySQL 9.0.1

1.6. UI/UX

- Figma

1.6. IDE

- Visual Studio Code 1.91.1
- IntelliJ IDEA 2024.01
- PyCharm 2024.01
- Android Studio Koala 17.0.11

1.7. 형상/이슈관리

- GitLab
- Jira

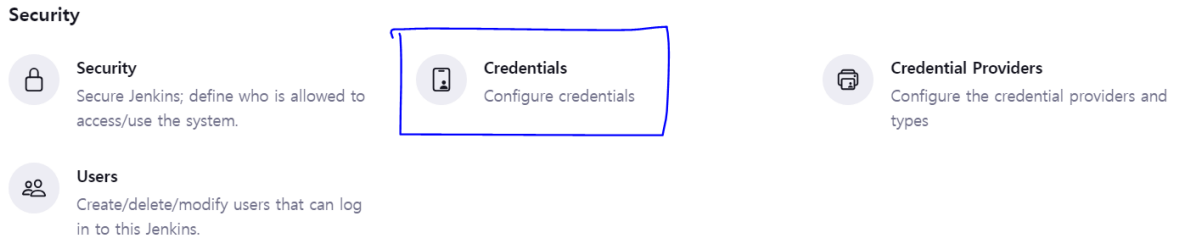
1.8. 기타 툴

- Postman 11.6.2
- Termius 9.2.0

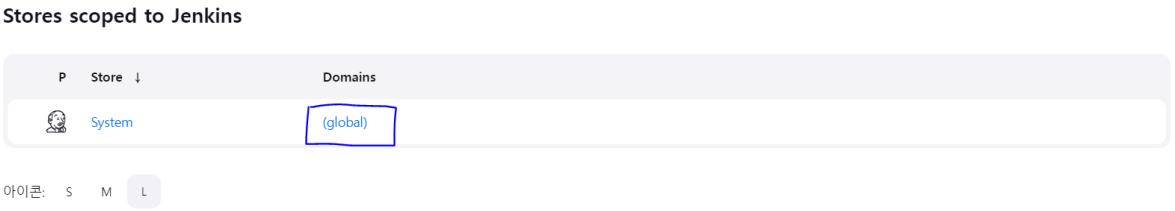
2. 환경변수

2.1. 민감 환경변수 관리

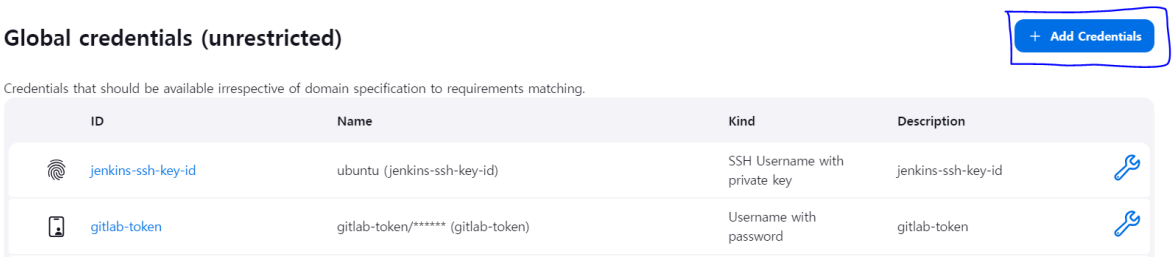
Jenkins credentials로 민감 환경변수 설정파일 수동 저장 및 관리(.gitignore에 추가하여 GitLab에 푸시되는 일이 없도록 함)



Jenkins 설정의 Credentials로 간다.



(global) 도메인으로 만든다.



Add Credentials로 새로운 Crdential을 만든다.

Secret 파일에 민감한 환경변수 파일을 넣고, 파이프라인 구성시 필요한 환경변수 파일을 복사해서 이미지를 빌드하는 형식으로 진행했다.

3. EC2 세팅

3.1. Docker 설치

```
# 1. 리눅스 업데이트
sudo apt update -y && sudo apt upgrade -y

# 1.1 필수 패키지 설치
sudo apt-get install -y ca-certificates curl gnupg lsb-releas

# 2. Docker의 공식 GPG 키를 추가할 디렉토리 생성
sudo mkdir -p /etc/apt/keyrings

# 3. Docker의 GPG 키 다운로드 및 바이너리 형식으로 변환하여 저장
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sud

# 4. Docker 저장소를 추가
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list
```

```
# 5. 패키지 목록 업데이트
sudo apt-get update

# 6. Docker 패키지 설치
sudo apt-get install -y docker-ce docker-ce-cli containerd.io

# 7. Docker 데몬을 시작하고 부팅 시 자동으로 시작하도록 설정
sudo systemctl start docker
sudo systemctl enable docker
```

3.2. Docker-compose 설정

```
version: '3.8'

services:
  nginx:
    image: nginx:latest
    restart: always
    environment:
      TZ: Asia/Seoul
    volumes:
      - ./data/nginx/nginx.conf:/etc/nginx/nginx.conf
      - ./data/certbot/conf:/etc/letsencrypt
      - ./data/certbot/www:/var/www/certbot
    ports:
      - "80:80"
      - "443:443"
    command: "/bin/sh -c 'while :; do sleep 320h & wait $${!}
  certbot:
    image: certbot/certbot
    restart: unless-stopped
    environment:
      TZ: Asia/Seoul
    volumes:
      - ./data/certbot/conf:/etc/letsencrypt
      - ./data/certbot/www:/var/www/certbot
    entrypoint: "/bin/sh -c 'trap exit TERM; while :; do cert
```

```
jenkins:
  image: jenkins/jenkins:lts
  restart: unless-stopped
  volumes:
    - jenkins_home:/var/jenkins_home
    - ./data/jenkins:/var/jenkins_shared
    - /var/run/docker.sock:/var/run/docker.sock  # Docker 소
    - /usr/local/bin/docker-compose:/usr/local/bin/docker-c
    - /usr/bin/docker:/usr/bin/docker  # Docker CLI 바이너리
    - /home/ubuntu/dpg:/home/ubuntu/dpg  # 호스트의 프로젝트 디
    - ./data/jenkins/log:/var/jenkins_home/logs  # 로그 디렉토
    - /home/ubuntu/dpg/compose:/var/compose
  environment:
    JENKINS_OPTS: --prefix=/jenkins
    JAVA_OPTS: -Djava.util.logging.config.file=/var/jenkins_
    TZ: Asia/Seoul

mysql:
  image: mysql:latest
  restart: unless-stopped
  environment:
    MYSQL_ROOT_PASSWORD: elqrkeldjswm
    MYSQL_DATABASE: deep
    MYSQL_USER: deep
    MYSQL_PASSWORD: elqrkeldjswm
    TZ: Asia/Seoul
  volumes:
    - ./data/mysql:/var/lib/mysql
  expose:
    - "3306"
  ports:
    - "3306:3306"
pythond:
  container_name: disrupt
  image: python-app-d
  environment:
    - TZ=Asia/Seoul
```

```
    volumes:
      - /home/ubuntu/model/:/home/ubuntu/model/
      - /home/ubuntu/detector/deepfake-detector/:/home/ubuntu
      - /home/ubuntu/attack/:/home/ubuntu/attack/
      - /usr/bin/bash:/usr/bin/bash
    deploy:
      resources:
        reservations:
          devices:
            - capabilities: [gpu]
    runtime: nvidia
  python:
     container_name: detect
     image: python-app
     environment:
       - TZ=Asia/Seoul
     volumes:
       - /home/ubuntu/model/:/home/ubuntu/model/
       - /home/ubuntu/detector/deepfake-detector/:/home/ubunt
       - /home/ubuntu/attack/:/home/ubuntu/attack/
       - /usr/bin/bash:/usr/bin/bash
  minio:
    image: minio/minio:RELEASE.2024-01-18T22-51-28Z
    container_name: minio
    ports:
      - 9000:9000
      - 9001:9001
    environment:
      MINIO_ROOT_USER: minio
      MINIO_ROOT_PASSWORD: minio1234
    command: server /data/minio --console-address :9001
    healthcheck:
      test: ["CMD", "mc", "ready", "local"]
      interval: 5s
      timeout: 5s
      retries: 5
  mlflow:
    build:
```

```
      context: .
      dockerfile: Dockerfile
    container_name: mlflow
    ports:
      - 5000:5000
    environment:
      - MLFLOW_TRACKING_URI=http://mlflow:5000
    volumes:
      - /home/ubuntu/data/disrupt/model/models:/mlflow/model
    depends_on:
      - mysql


volumes:
  jenkins_home:
```

3.2.1 Docker-compose (disrupt-train)

```
services:
  disrupt-train:
    container_name: disrupt-train
    image: disrupt-train-image
    environment:
      - TZ=Asia/Seoul
      - DB_USER=deep
      - DB_PASSWORD=elqrkeldjswm
      - DB_HOST=mysql
      - DB_PORT=3306
      - DB_NAME=deep
      - PYTHONPATH=/workspace
      - WANDB_API_KEY={wandb_api_key}
    volumes:
      - /home/ubuntu/model/:/home/ubuntu/model/
      - /usr/bin/bash:/usr/bin/bash
      - /home/ubuntu/data/disrupt/model/models/:/mlflow/model
      - /home/ubuntu/data/disrupt:/home/ubuntu/data/disrupt
```

```
    deploy:
      resources:
        reservations:
          devices:
            - capabilities: [gpu]
    runtime: nvidia
    networks:
      - dpg_default
networks:
  dpg_default:
    external: true
    driver: bridge
```

3.2.2 Docker-compose (disrupt-server)

```
services:
  disrupt-server:
    container_name: disrupt-server
    image: disrupt-server-image
    environment:
      - TZ=Asia/Seoul
      - DISRUPT_MODEL=/var/model/disrupt/unet_epoch_2.pth
      - DEEPFAKE_MODEL=/var/model/disrupt/inswapper_128.onnx
    volumes:
      - /home/ubuntu/fastapi_disrupt/app/model:/var/model
      - /usr/bin/bash:/usr/bin/bash
    networks:
      - dpg_default
    deploy:
      resources:
        reservations:
          devices:
            - capabilities: [gpu]
    runtime: nvidia
networks:
  dpg_default:
```

```
    external: true
    driver: bridge
```

3.2.3 Docker-compose (preprocessing)

```
services:
  fastapi:
    container_name: fast
    image: fast-image
    environment:
      - TZ=Asia/Seoul
      - DB_USER=deep
      - DB_PASSWD=elqrkeldjswm
      - DB_HOST=mysql
      - DB_PORT=3306
      - DB_NAME=deep
    volumes:
      - /home/ubuntu/model/:/home/ubuntu/model/
      - /usr/bin/bash:/usr/bin/bash
      - /home/ubuntu/data/disrupt:/home/ubuntu/data/disrupt
    networks:
      - dpg_default
networks:
  dpg_default:
    external: true
    driver: bridge
```

3.3. Nginx 설정

```
user  nginx;
worker_processes  1;

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;

    log_format main escape=json
        '{'
            '"remote_addr": "$remote_addr", '
            '"remote_user": "$remote_user", '
            '"time_local": "$time_local", '
            '"request": "$request", '
            '"status": "$status", '
            '"body_bytes_sent": "$body_bytes_sent", '
            '"http_referer": "$http_referer", '
            '"http_user_agent": "$http_user_agent", '
            '"http_x_forwarded_for": "$http_x_forwarded_for",
            '"host": "$host", '
            '"server_name": "$server_name", '
            '"request_uri": "$request_uri", '
            '"uri": "$uri", '
            '"request_body": "$request_body", '
            '"args": "$args", '
            '"upstream_addr": "$upstream_addr", '
            '"upstream_status": "$upstream_status", '
            '"request_time": "$request_time"'
        '}';

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

    sendfile        on;
    keepalive_timeout  65;
    # 요청 제한 설정
    # limit_req_zone $binary_remote_addr zone=mylimit:10m rat

# HTTP 설정
server {
    listen 80;
```

```
    server_name anti-deepfake.kr truthguard.site;
    server_tokens off;

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

    location / {
        return 301 https://$host$request_uri;
    }
}

# HTTPS 설정 - anti-deepfake.kr
server {
    listen 443 ssl;
    server_name anti-deepfake.kr;
    server_tokens off;

    ssl_certificate /etc/letsencrypt/live/anti-deepfake.kr/fu
    ssl_certificate_key /etc/letsencrypt/live/anti-deepfake.k
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

    client_max_body_size 50M;

        # 모델 시간 걸리는 것을 위한 타임아웃 설정
    client_body_timeout 120s;
    client_header_timeout 120s;
    proxy_read_timeout 120s;
    proxy_connect_timeout 120s;

    location / {
        proxy_pass http://react:80;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forward
        proxy_set_header X-Forwarded-Proto $scheme;
```

```
        add_header Cache-Control "no-store, no-cache, must-re
        add_header Pragma "no-cache";
        add_header Expires "0";

        root /usr/share/nginx/html;
        index index.html;
    }


    location /disrupt {
        proxy_pass http://disrupt-server:8000;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forward
        proxy_set_header Host $http_host;
    }

    location /detect {
        proxy_pass http://fastapi-detect:8000;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forward
        proxy_set_header Host $http_host;

        rewrite ^/detect/(.*) /$1 break;
    }

    location /jenkins {
        proxy_pass http://jenkins:8080/jenkins;
        proxy_set_header Host $host:443;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forward
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header X-Forwarded-Port 443;

        add_header Cache-Control "no-store, no-cache, must-re
        add_header Pragma "no-cache";
        add_header Expires "0";

        proxy_http_version 1.1;
```

```
    }

    location /api {
        proxy_pass http://fast:8000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forward
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header X-Original-Method $request_method;

        add_header Cache-Control "no-store, no-cache, must-re
        add_header Pragma "no-cache";
        add_header Expires "0";
    }

    location /disrupt-train {
        proxy_pass http://disrupt-train:8000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forward
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header X-Original-Method $request_method;

        add_header Cache-Control "no-store, no-cache, must-re
        add_header Pragma "no-cache";
        add_header Expires "0";
    }

}
```

3.4. EC2 Port

Port 번호	내용
22	SSH
80	HTTP (HTTPS로 redirect)
443	HTTPS

4. CI/CD 구축

4.1. Jenkins 도커 이미지 + 컨테이너

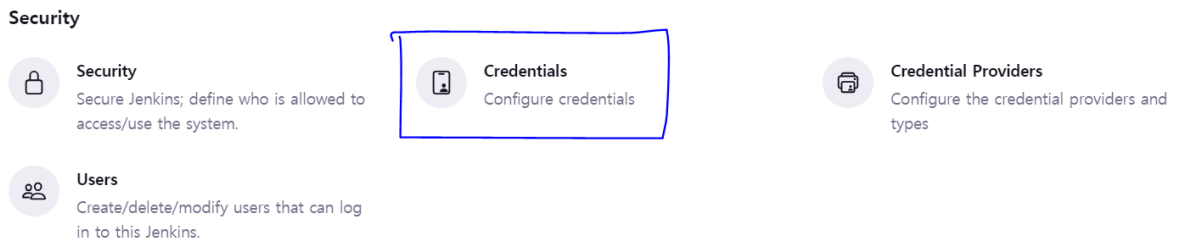
docker-compose.yml 내부

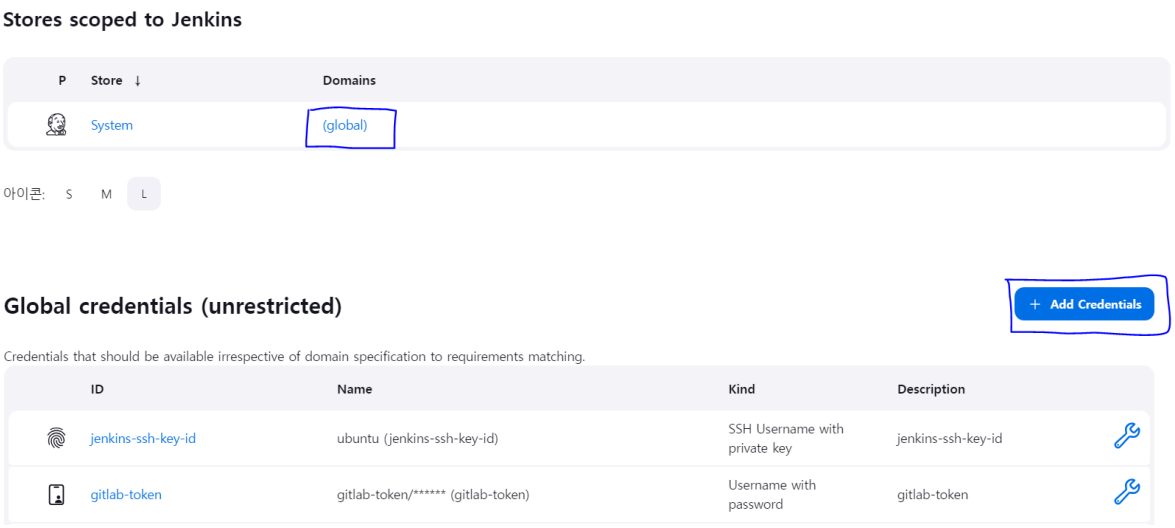
```
    jenkins:
        image: jenkins/jenkins:lts
        restart: unless-stopped
        volumes:
            - jenkins_home:/var/jenkins_home
            - ./data/jenkins:/var/jenkins_shared
            - /var/run/docker.sock:/var/run/docke
            - /usr/local/bin/docker-compose:/usr/
            - /usr/bin/docker:/usr/bin/docker  #
            - /home/ubuntu/newzy:/home/ubuntu/new
            - ./data/jenkins/log:/var/jenkins_hom

        environment:
            JENKINS_OPTS: --prefix=/jenkins
            JAVA_OPTS: -Djava.util.logging.config
            TZ: Asia/Seoul
```

4.2. Jenkins 설정

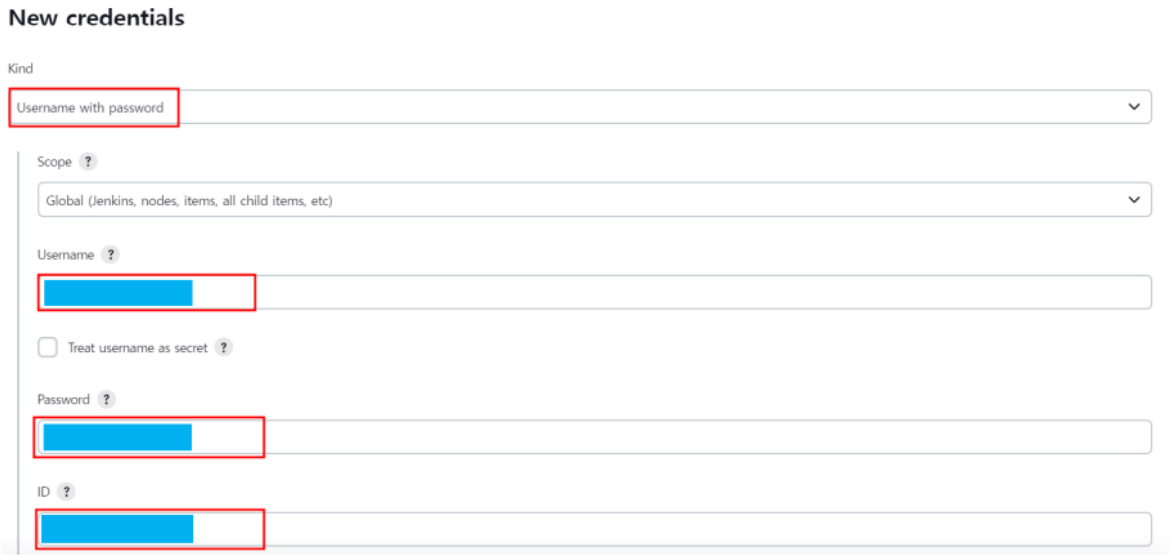
4.2.1 GitLab Credentials 설정

1. 아이디 → “Credentials” 클릭
2. “Store : System” → “(global)” → “+ Add Credentials” 클릭





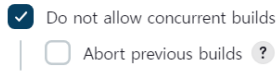
3. “Kind”에 “Username with password” 입력 → “Username”에 GitLab ID 혹은 원하는 ID 입력(gitlab-token) → “Password”에 Gitlab Personal Access Tokens 입력 → “ID”에 임의 아이디 입력(gitlab-token) → 생성
   *** Personal Access Token은 Gitlab > User Settings > Access Tokens에서 생성



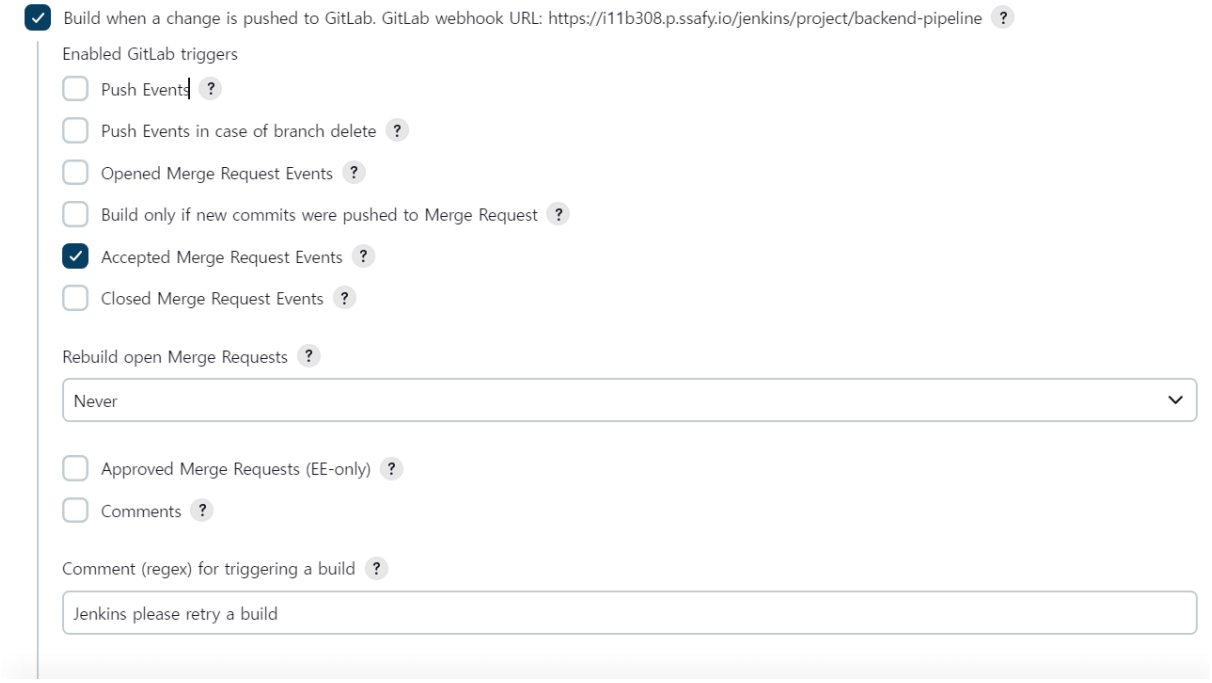
4.2.2 Jenkins Item 생성

1. “새로운 Item” 클릭
2. “Enter an item name”에 임의 Item 이름 입력 → “Pipeline” 클릭

3. “General” → “Do not allow concurrent builds” 클릭
   (한 빌드를 진행중이면 동시에 빌드를 진행하지 않게 한다)



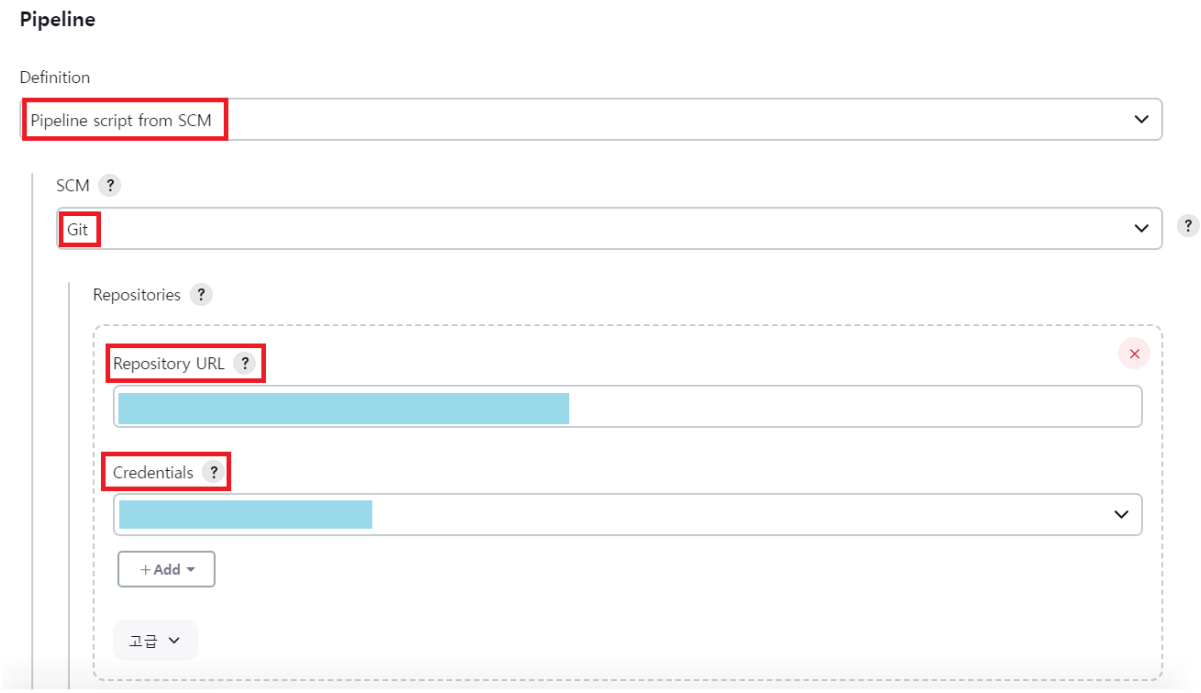
4. “Build Triggers” → “Build when a change is pushed to GitLab” 클릭
   (WebHook 설정 : GitLab 특정 브랜치 merge 시 자동 빌드 + 배포 설정)
   (해당 URL 복사 → WebHook 설정 시 사용 예정)



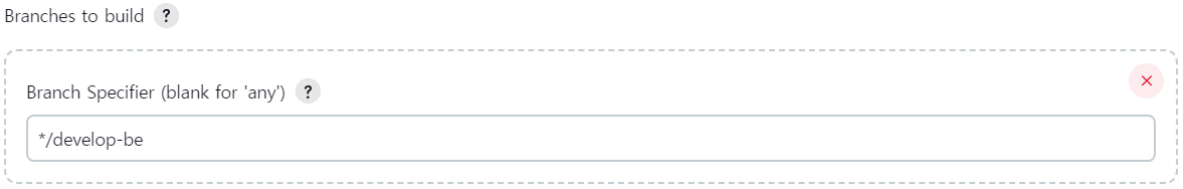
5. “Build when a change is pushed to GitLab” 하위의 “고급...” 클릭

6. 특정 브랜치에서 타겟 브랜치로 머지를 할 경우 빌드 + 배포가 진행되도록 설정
   Secret token의 “Generate” 클릭 후 생성된 토큰값 복사

7. “Pipeline” → “Definition”에 Pipeline script from SCM 설정 → “SCM”에 “Git” 설정 → “Repository URL”에 프로젝트 GitLab URL
   입력 → “Credentials”에 사전에 추가한 Credentials 입력



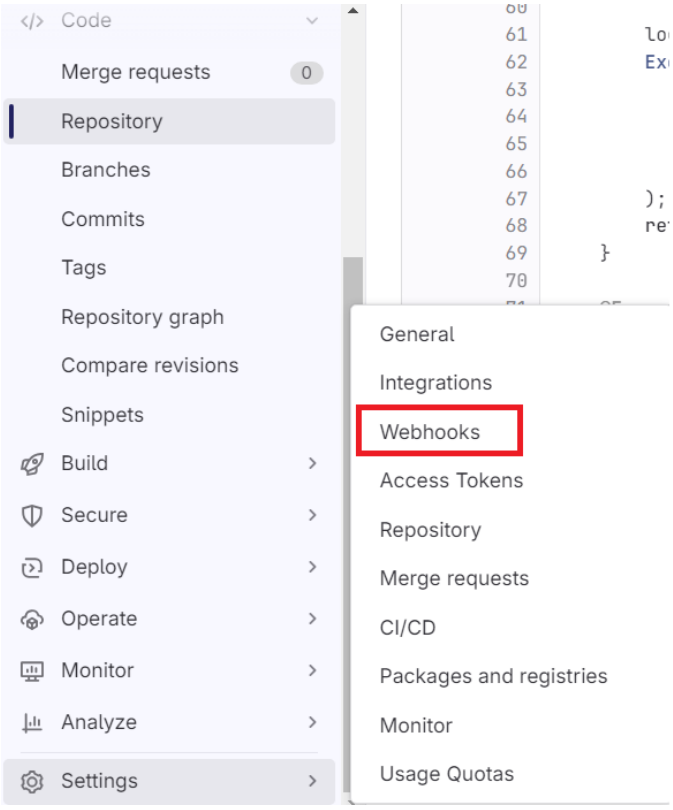
8. “Branch Specifier”에 빌드 할 브랜치명 입력 (master일 시 “*/master)



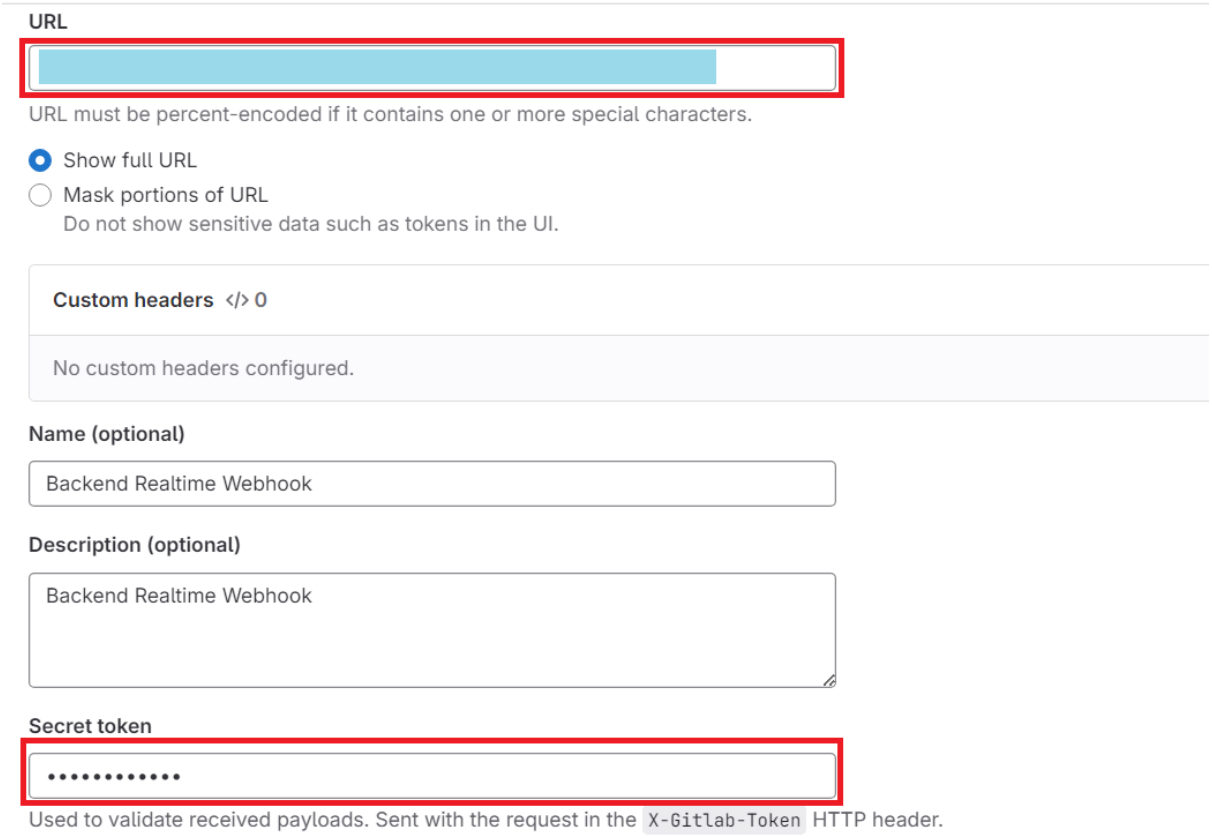
9. “Script Path”에 Jenkinsfile 경로 입력

4.2.3. GitLab Webhook 설정

1. 프로젝트 GitLab → “Settings” → “Webhooks” 클릭



2. “URL”에 사전에 복사해놓은 Jenkins URL 입력 → “Secret token”에 사전에 복사해 놓은 Secret token 입력 → “Merge request events” 클릭
   후 WebHook 적용 브랜치 입력 (Jenkins Branch Specifier과 일치하여야 함)



4.2.4. 빌드 및 배포

> Option 1. 상기 WebHook 설정한 브랜치로 merge

> Option 2. Jenkins 홈 화면 → Jenkins Item 클릭 → “지금 빌드” 클릭

5. 시연 시나리오

5.1. 생성 방해

5.1.1. 웹 사이트

- 메인 페이지에서 ‘생성 방해’ 버튼 클릭
- 노이즈를 삽입하고자 하는 이미지 삽입 후 ‘삽입하기’ 버튼 클릭
- 원본 이미지와 노이즈 씌운 이미지의 딥페이크 생성 결과 비교
- 다시 업로드 페이지로 돌아와 **크롬 익스텐션** 페이지로 이동

5.1.2. 어플리케이션

- 업로드한 사진을 서버에 저장한다는 동의 화면 → '동의' 클릭
- '업로드' 버튼 클릭 → 갤러리 접근 권한 설정
- 노이즈를 삽입할 이미지 업로드
- 미리보기로 노이즈가 삽입된 이미지 확인 후 '이미지 저장' 버튼 클릭
- 하단 바에서 'dashboard' 클릭 → 지금까지 변환된 사진 목록 조회
- 목록 중 사진 하나 클릭하여 세부 조회

5.1.3. 크롬 익스텐션

- 크롬 익스텐션 페이지 접속
- 노이즈를 삽입할 이미지 업로드
- 노이즈가 삽입된 이미지 확인 후 저장

5.2. 탐지

5.2.1. 웹 사이트

- 딥페이크 생성물인지 판단하고자 하는 이미지 삽입
- 해당 이미지가 딥페이크 생성물 같은지에 대한 사용자 의견 받기
- 딥페이크 탐지 결과 확인