포팅 메뉴얼

목차

---

1. 개발환경

1-1 프론트엔드

- yarn : ^1.22.19
- React : ^18.2.0
- Typescript : ^5.2.2

- Vite : ^5.1.4
- react-router-dom : ^6.22.3
- Tailwindcss : ^3.4.1
- 상태관리
  - recoil : ^0.7.7
- AI
  - @tensorflowtfjs : ^4.17.0
- API 통신
  - axios : ^1.6.8
- 지도
  - react-kakao-maps-sdk : ^1.1.26
- 차트
  - cytoscape : ^3.28.1
  - cytoscape-cose-bilkent : ^4.1.0
- Project Setup

```
yarn install
```

- Compile and Minify for Production

```
yarn build
```

- Compile and Hod-Reload for Development

```
yarn run dev
```

1-2 백엔드

- IntelliJ : 2023.3.2

- Spring Boot : 3.2.3
- JDK : 21
- MySQL : 8.0.22
- Redis : 7.2.4
- MongoDB : 7.0.8

1-3 서버 및 인프라

- Server : Ubuntu 20.04.6 LTS
- Jenkins : 2.449
- wsl : 2.0

2. 설정파일 및 환경 변수 정보

Spring

application.properties

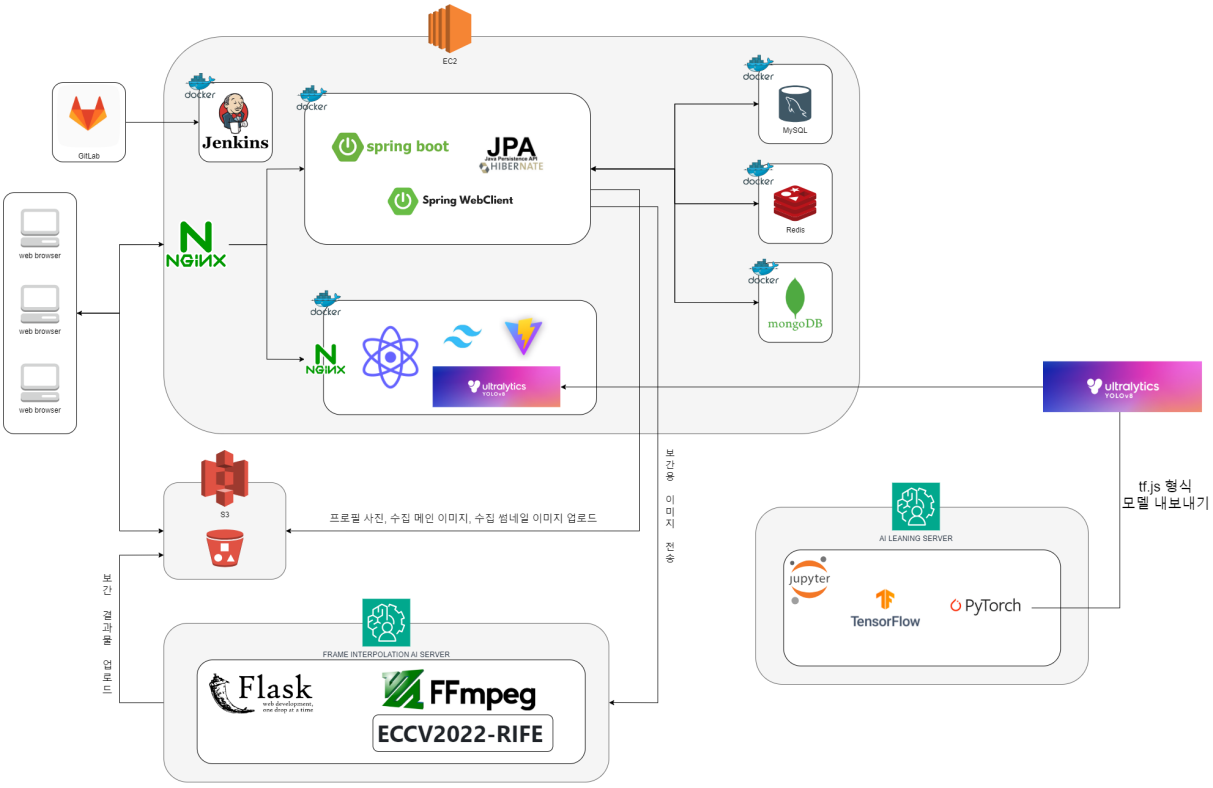
```
#server.port=3000
server.servlet.context-path=/api

#SpringSecurity
#kakao_registration
spring.security.oauth2.client.registration.kakao.client-name=
spring.security.oauth2.client.registration.kakao.client-id={i
spring.security.oauth2.client.registration.kakao.client-secre
spring.security.oauth2.client.registration.kakao.redirect-uri
spring.security.oauth2.client.registration.kakao.authorizatio
spring.security.oauth2.client.registration.kakao.scope={scope
#kakao_provider
spring.security.oauth2.client.provider.kakao.authorization-ur
spring.security.oauth2.client.provider.kakao.token-uri=https:
spring.security.oauth2.client.provider.kakao.user-info-uri=ht
spring.security.oauth2.client.provider.kakao.user-name-attrib
spring.security.oauth2.client.registration.kakao.client-authe



#Email setting
```

```
spring.mail.host={host}
spring.mail.port=587
spring.mail.username={username}
spring.mail.password={password}
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.starttls.enable=true
spring.mail.properties.mail.smtp.starttls.required=true
spring.mail.properties.mail.smtp.connectiontimeout=5000
spring.mail.properties.mail.smtp.timeout=5000
spring.mail.properties.mail.smtp.writetimeout=5000
# 30min
spring.mail.auth-code-expiration-millis=1800000

#mysql
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url={usr}
spring.datasource.username={username}
spring.datasource.password={password}
#CamelCase
spring.jpa.hibernate.naming.implicit-strategy=org.springframew
spring.jpa.hibernate.ddl-auto=none

#redis
spring.data.redis.host=j10d104.p.ssafy.io
spring.data.redis.port={port}
spring.data.redis.password={password}

#JWT
spring.jwt.secret={secret}

#S3 key
cloud.aws.credentials.accessKey={accessKey}
cloud.aws.credentials.secretKey={secretKey}
#S3 bucketName
cloud.aws.s3.bucketName={bucketName}
#S3 location
cloud.aws.region.static={static}
#cloud formation ??? ???? ?? ??.
```

```
cloud.aws.stack.auto=false
#fileUploadMaxSpeed
spring.servlet.multipart.max-file-size=-1
spring.servlet.multipart.max-request-size=-1
#30min accessToken
spring.jwt.access=1800000
#60min refreshToken
spring.jwt.refresh=3600000

#mongoDB
spring.data.mongodb.uri={uri}
```

3. AWS 서버 설정

3-1 프로젝트 구조



3-2 Docker 설치

1. **apt 업데이트**

   `apt-get update`

2. **도커 설치**

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

3. **도커 설치 확인**

   `docker run hello-world`

4. **도커 버전 확인**

   `docker -v`

5. **도커 데몬 실행**

   `systemctl start docker`

3-3 Jenkins 설치 및 설정

1. **jenkins container 생성 및 구동**

```
cd /home/ubuntu && mkdir jenkins-data

sudo ufw allow *8080*/tcp
sudo ufw reload
sudo ufw status

sudo docker run -d -p 8080:8080 -v /home/ubuntu/jenkins-dat
a:/var/jenkins_home --name jenkins jenkins/jenkins:lts

sudo docker logs jenkins

sudo docker stop jenkins
sudo docker ps -a
```

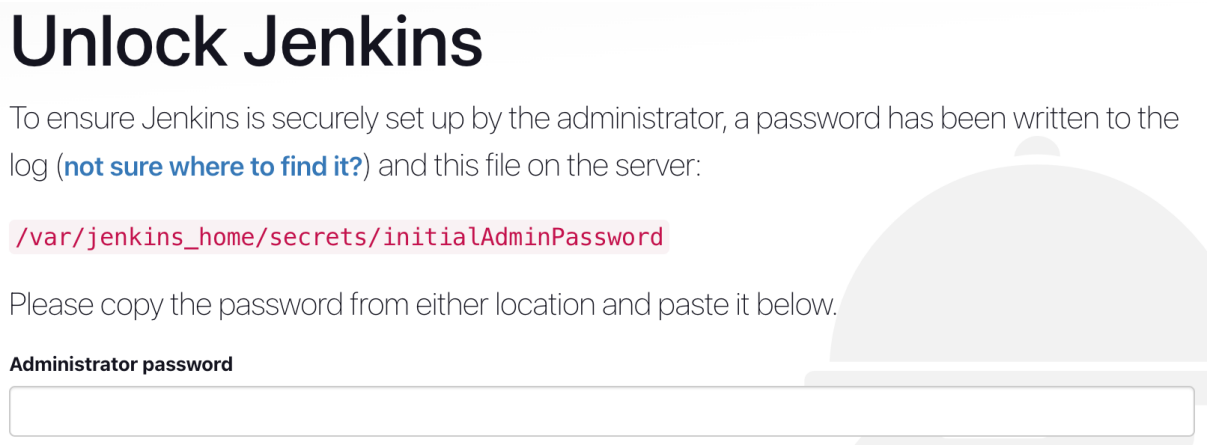
2. **환경 변수 설정**

```
cd /home/ubuntu/jenkins-data

mkdir update-center-rootCAs

wget https://cdn.jsdelivr.net/gh/lework/jenkins-update-center
```

```
sudo sed -i 's#https://updates.jenkins.io/update-center.json#

sudo docker restart jenkins
```

3. **젠킨스 접속**

   http://j10d104.p.ssafy.io:8080/

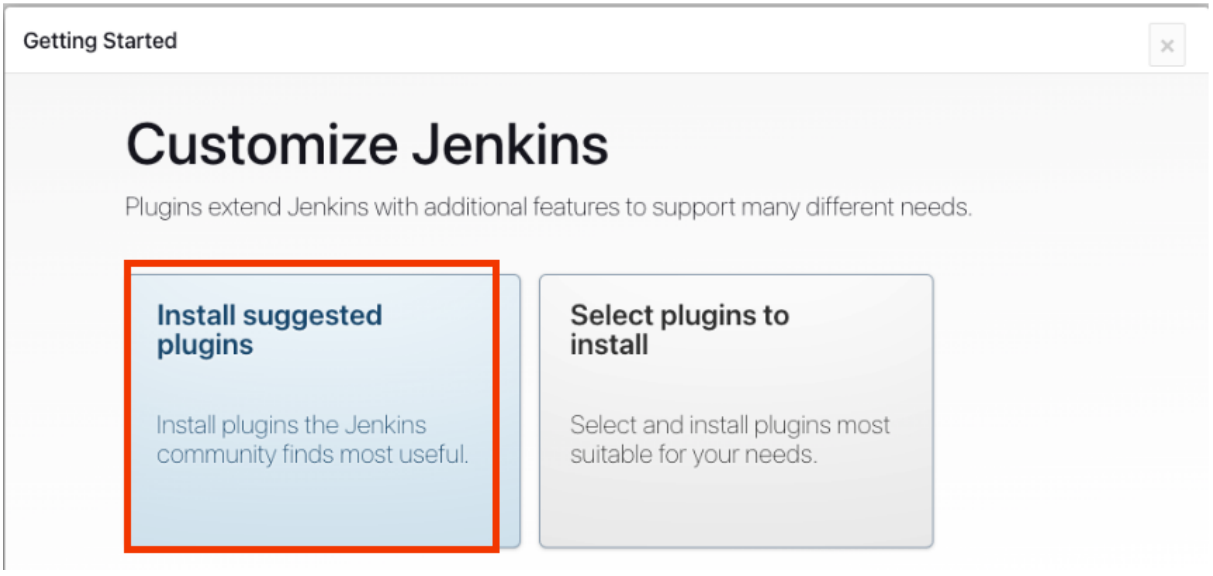


4. **젠킨스 접속 키 확인**

```
sudo docker logs jenkins
```

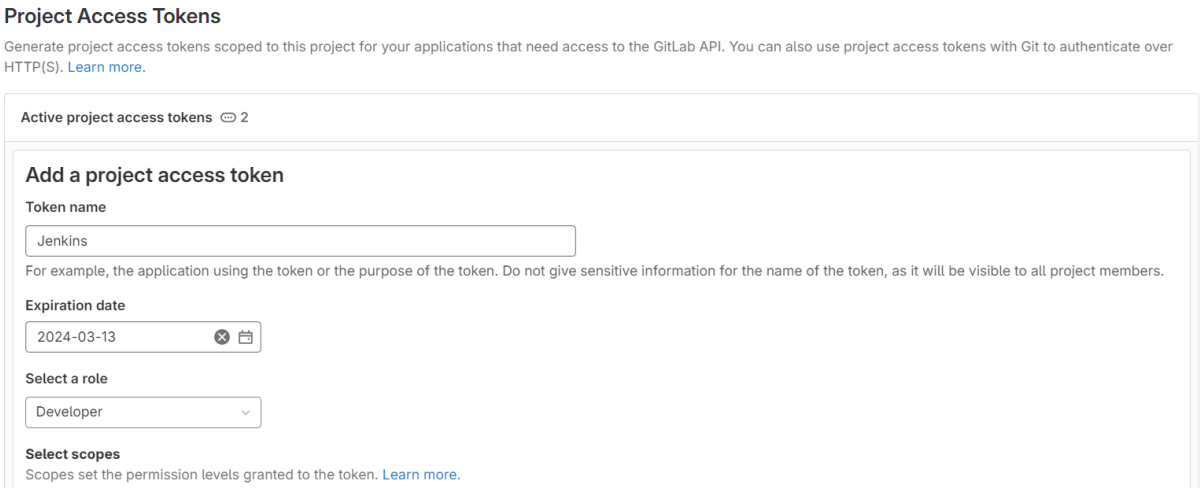
```
*************************************************************
*************************************************************
*************************************************************

Jenkins initial setup is required. An admin user has been created and a password generated.
Please use the following password to proceed to installation:

        Jenkins secret key

This may also be found at: /var/jenkins_home/secrets/initialAdminPassword

*************************************************************
*************************************************************
*************************************************************
```

5. **필수 플러그인 설치**



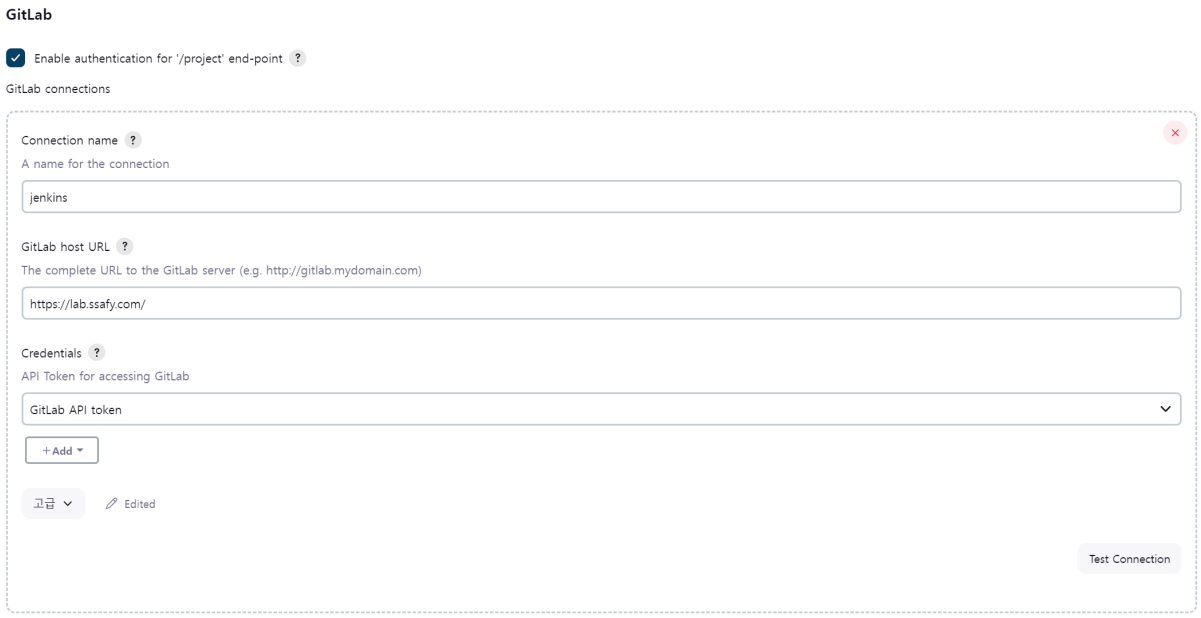
6. **GitLab 연동 설정**

깃랩 api 토큰 생성



Jenkins credential 추가

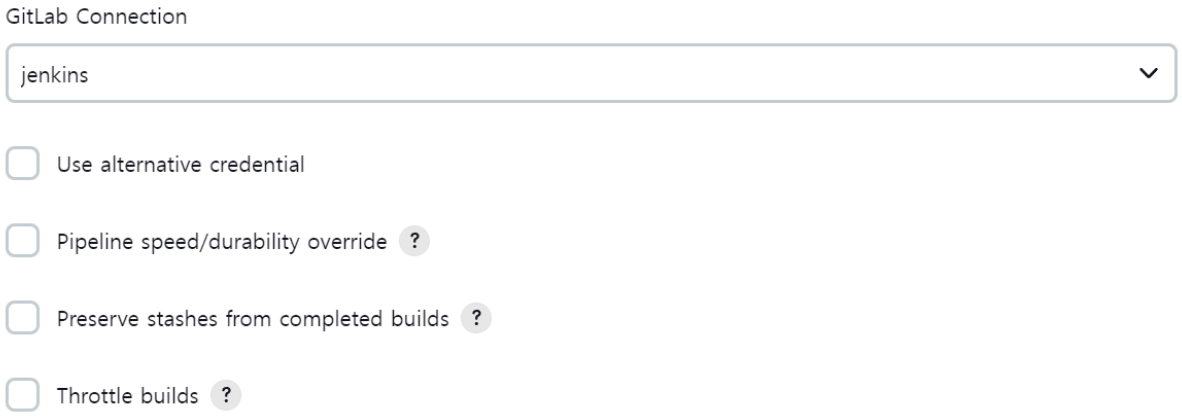
system 설정 → GitLab 설정



7. 백앤드 CI/CD 파이프라인

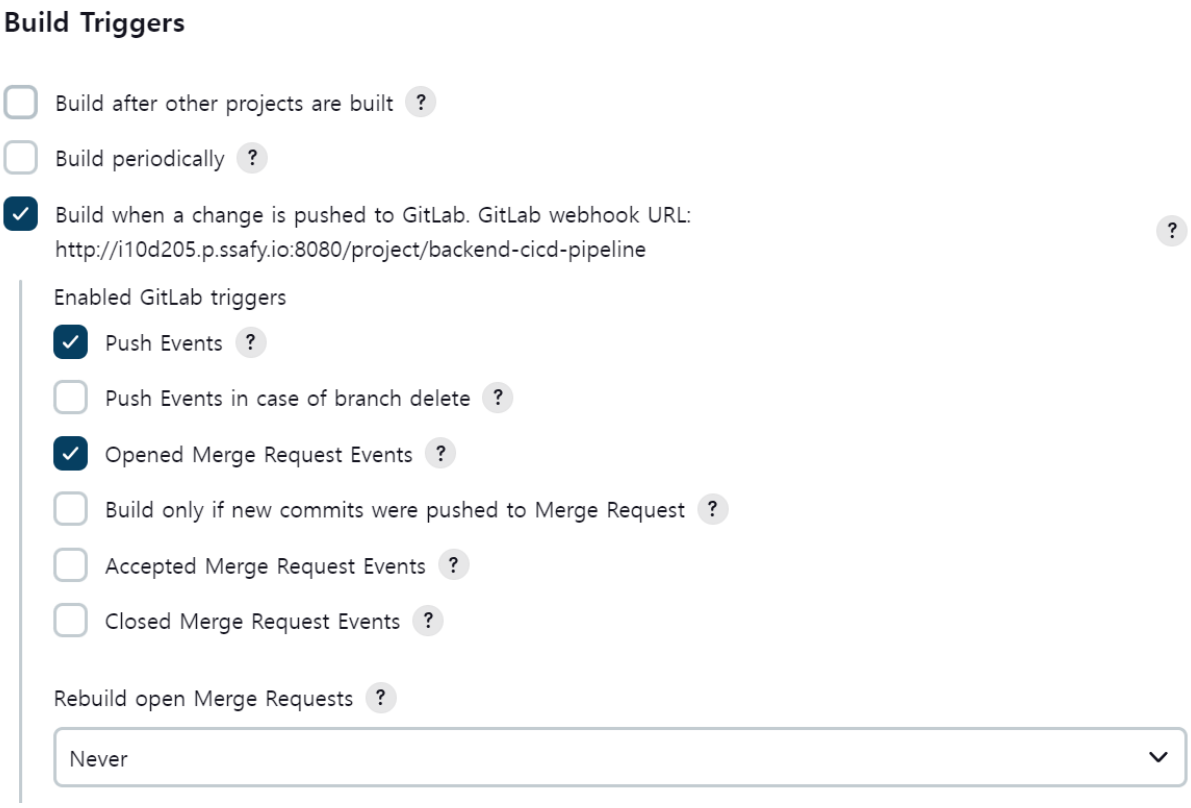
새로운 Item을 Pipeline으로 생성

미리 만들어둔 Gitlab 연동 할당

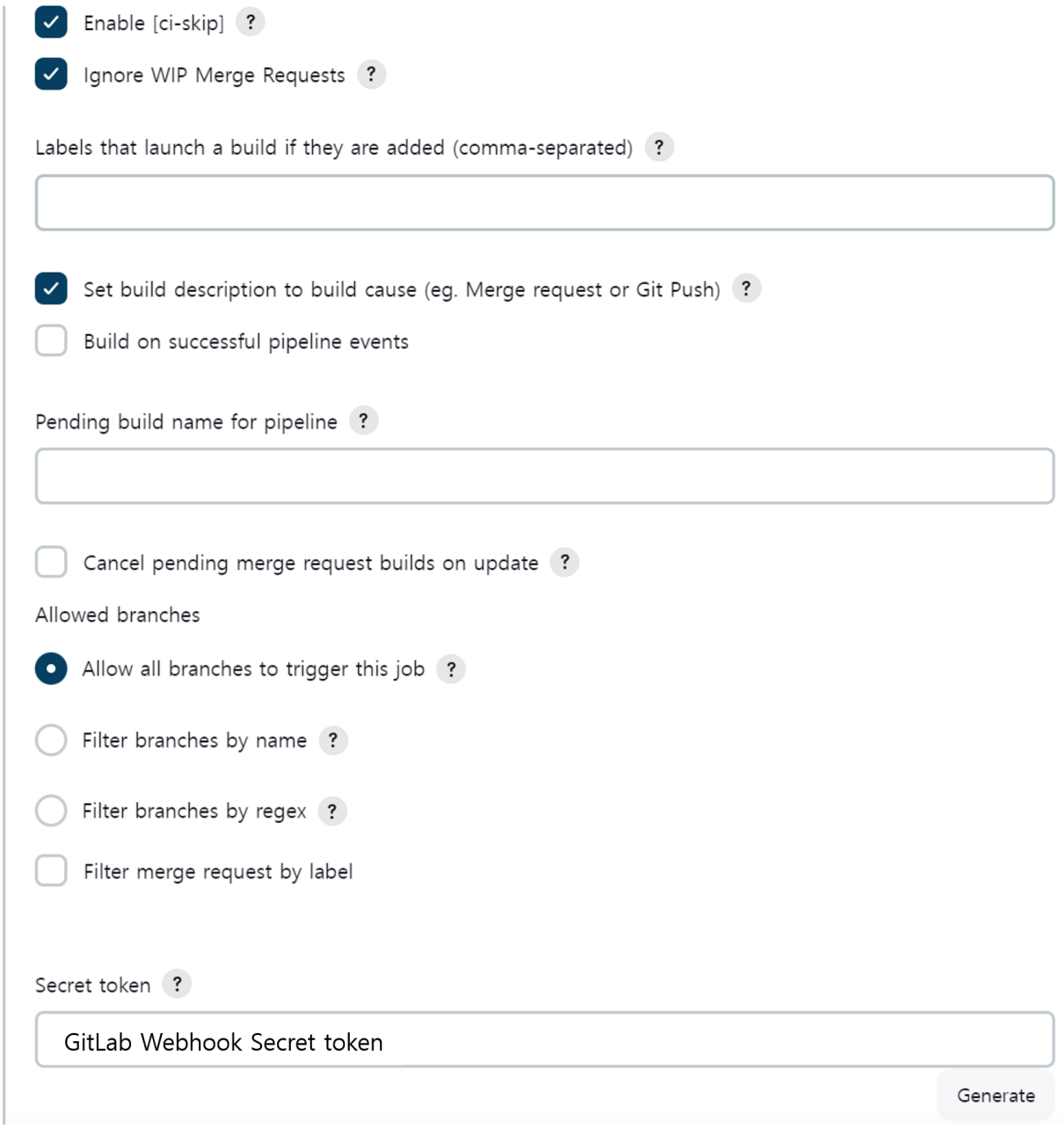


Build Triggers 설정

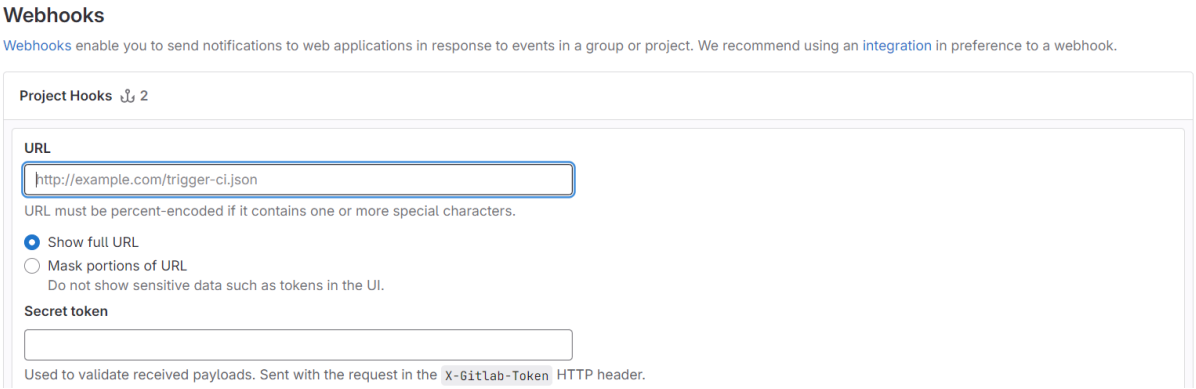
- 백엔드, 프론트엔드 브런치에 푸쉬 이벤트 발생 시 빌드 유발하도록 설정


Build Triggers
Build after other projects are built
Build periodically
Build when a change is pushed to GitLab. GitLab webhook URL:
http://i10d205.p.ssafy.io:8080/project/backend-cicd-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

GitLab 웹훅 설정


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
GitLab Webhook Secret token
Generate

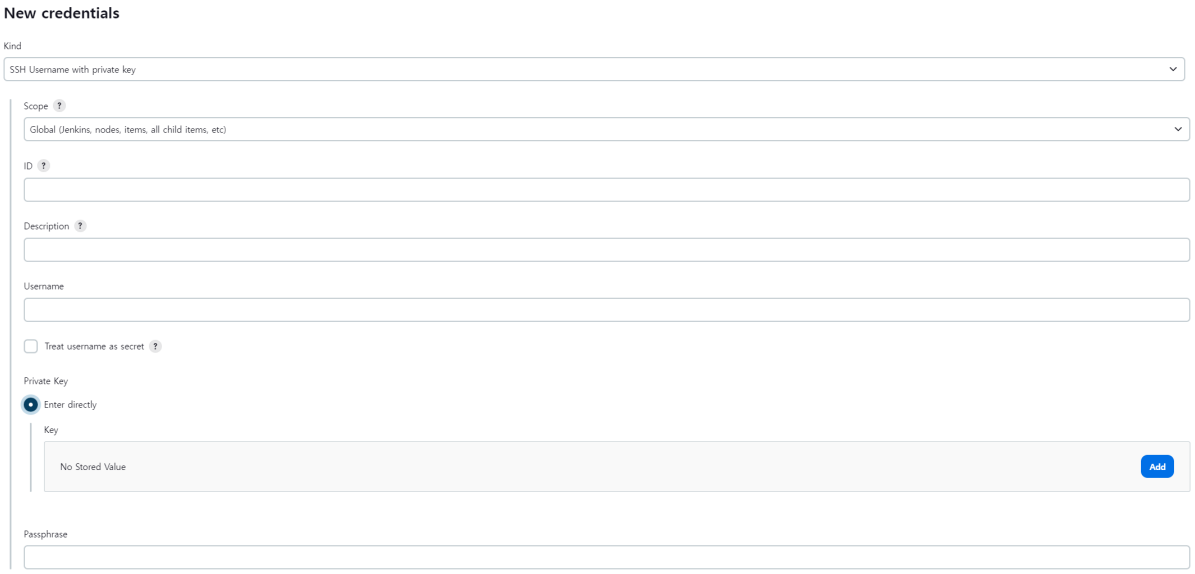
GitLab 웹훅 추가



8. **SSH 설정**

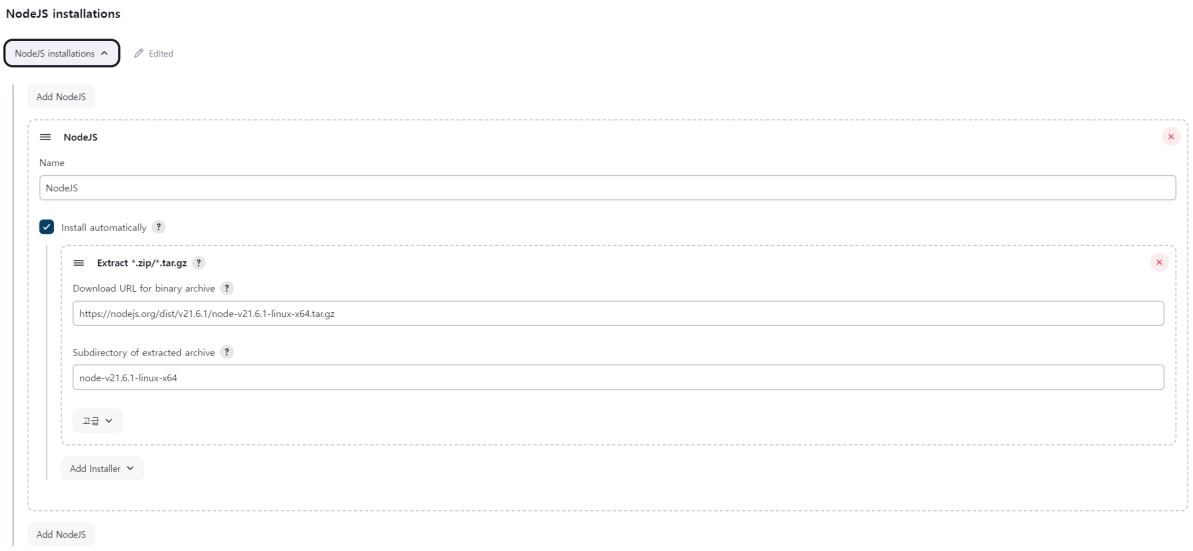
Jenkins Pipeline 에서 AWS 서버에 빌드 파일 전송 및 실행을 위한 SSH 설정

- Jenkins Plugin 에서 SSH Agent Plugin 설치
- Credentials 추가



- key 에 SSH pem 파일 정보 복사 후 붙여넣기

9. **NodeJs 설정**



10. 백엔드 파이프라인 스크립트

```
pipeline {
    agent any
    tools {
        gradle 'gradle'
    }
    stages {
        stage('clone') {
            steps {
                git branch: 'BE', credentialsId : 'gitlab', u
            }
        }
        stage('BE-Build') {
            steps {
                dir("./grabpic-BE") {
                    sh "chmod +x gradlew && ./gradlew clean b
                }
            }
        }
        stage('Deploy') {
           steps {
               dir('grabpic-BE/build/libs') {
                   sh "chmod 777 ./*"
```

```
                    sshagent(credentials: ['pem']) {
                         sh 'ssh -o StrictHostKeyChecking=no u
                                                  //빌드파일을 서버
                         sh 'scp grabpic-0.0.1-SNAPSHOT.jar ub
                                                  //서버에서 도커
                         //sh 'scp ../../Dockerfile ubuntu@j10
                                                  //컨테이너 생성을
                         sh 'ssh ubuntu@j10d104.p.ssafy.io "cd
                    }
                }
            }
        }
    }
}
```

11. **백엔드 DockerFile**

```
FROM openjdk:17-alpine
EXPOSE 5000
ARG JAR_FILE=BE-0.0.1-SNAPSHOT.jar
COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java", "-jar", "/app.jar"]
```

12. **백엔드 run.sh**

```
echo "start build image"
echo "=========================================================
sudo docker build -t be/app . #이미지 빌드
echo "=========================================================
echo "complete built"
echo "=========================================================
echo "is any same name container running?" #같은 이름으로 실행중인
sudo docker ps -a --filter "name = be-app" | grep -q . && sud
echo "=========================================================
echo "docker run"
```

```
sleep 5
sudo docker run -p 5000:8080 -d --name=be-app be/app #컨테이너
echo "=========================================================
echo "rmi process running" #같은 이름의 이미지가 있다면 삭제
sudo docker rmi -f $(sudo docker images -f "dangling=true" -q
```

13. **프론트엔드 파이프라인 스크립트**

```
pipeline {
   agent any

   stages {
      stage('Clone') {
          steps {
              git branch: 'FE', credentialsId : 'gitlab', url
          }
      }
      stage('Build'){
          steps {
              dir('client'){
                nodejs(nodeJSInstallationName: 'NodeJS'){
                  sh 'node --version'
                  sh 'yarn install && yarn build'
                }
              }
          }
      }
       stage('tar') {
            steps {
                dir('client'){
                    sh 'tar -cvf dist.tar dist' //빌드 파일 압축
                }
            }
       }
      stage('ssh') {
        steps {
            dir('client'){
```

```
                sshagent(credentials: ['pem']) {
                    sh 'ls'
                    sh 'ssh -o StrictHostKeyChecking=no ubunt
                                        //빌드 파일 서버 전송
                    sh 'scp dist.tar ubuntu@j10d104.p.ssafy.i
                }
            }
        }
    }
    stage('unpack'){
        steps {
            sshagent(credentials: ['pem']) {
                                        //빌드 파일 압축 해제
                    sh 'ssh ubuntu@j10d104.p.ssafy.io "cd /ho
                }
        }
    }
    stage('run.sh'){
        steps {
            sshagent(credentials: ['pem']) {
                                        //컨테이너 생성을 위한 쉘
                    sh 'ssh ubuntu@j10d104.p.ssafy.io "cd /ho
                }
        }
    }
  }
}
```

14. **프론트엔드 DockerFile**

```
# nginx 이미지를 사용합니다. 뒤에 tag가 없으면 latest 를 사용합니다.
FROM nginx:latest

# root 에 app 폴더를 생성
RUN mkdir /app

# work dir 고정
```

```
WORKDIR /app

# work dir 에 dist 폴더 생성 /app/dist
RUN mkdir ./dist

# host pc의 현재경로의 dist 폴더를 workdir 의 dist 폴더로 복사
ADD ./dist ./dist

# nginx 의 default.conf 를 삭제
RUN rm /etc/nginx/conf.d/default.conf

# host pc 의 default.conf 를 아래 경로에 복사
COPY ./default.conf /etc/nginx/codenf.d

# 8082 포트 오픈
EXPOSE 8082
```

15. **프론트엔드 run.sh**

```
echo "start build image"
echo "=========================================================
sudo docker build -t fe/app .
echo "=========================================================
echo "complete built"
echo "=========================================================
echo "is any same name container running?"
sudo docker ps -a --filter "name = fe-app" | grep -q . && sud
echo "=========================================================
echo "docker run"
sleep 5
sudo docker run -p 8082:8082 -d --name=fe-app fe/app
echo "=========================================================
echo "rmi process running"
sudo docker rmi -f $(sudo docker images -f "dangling=true" -q
```

3-4 MySQL 컨테이너 생성

1. **MySQL Docker Image 다운로드**

   `docker pull mysql:8.0.22`

2. **다운로드 된 Docker Image 확인**

   `docker images -a`

```
REPOSITORY                    TAG         IMAGE ID       CREATED            SIZE
mysql                         8.0.22      d4c3cafb11d5   3 weeks ago        545MB
```

3. **MySQL Docker 컨테이너 생성 & 실행**

   `docker run --name mysql -e MYSQL_ROOT_PASSWORD=[패스워드] -d -p 3306:3306 mysql:8.0.22`

4. **Docker 컨테이너 리스트 확인**

   `docker ps -a`

```
CONTAINER ID   IMAGE                 COMMAND                  CREATED       STATUS          PORTS                                          NAMES
fd55b73a7a60   mysql:8.0.22          "docker-entrypoint.s…"   2 weeks ago   Up 11 days      0.0.0.0:3306->3306/tcp, :::3306->3306/tcp, 33060/tcp   mysql
```

3-5 Redis 컨테이너 생성

1. **Redis Docker Image 다운로드**

   `docker pull --platform linux/amd64 redis`

2. **다운로드 된 Docker Image 확인**

   `docker images -a`

```
REPOSITORY                    TAG         IMAGE ID       CREATED            SIZE
redis                         latest      bdff4838c172   4 weeks ago        138MB
```

3. **Redis Docker 컨테이너 생성 & 실행**

   `docker run --name redis -p 6379:6379 --network redis-network -it -d redis`

4. **Docker 컨테이너 리스트 확인**

   `docker ps -a`

```
CONTAINER ID   IMAGE                     COMMAND                  CREATED       STATUS                  PORTS
53653fb364d4   redis                     "docker-entrypoint.s…"   2 weeks ago   Up 11 days              0.0.0.0:6379->6379/tcp, :::6379->6379/tcp
                          NAMES
                          redis
```

3-6 MongoDB 컨테이너 생성

1. Mongo Docker Image 다운로드

   `docker pull mongo`

2. 다운로드 완료 Docker Image 확인

   `docker images -a`

3. MongoDB 컨테이너 생성 & 실행

   `sudo docker run --name mongo -p {port}:{port} -e MONGO_INITDB_ROOT_USERNAME={username} -e MONGO_INITDB_ROOT_PASSWORD={password} -d mongo`

4. Docker 컨테이너 리스트 확인

   `docker ps -a`

3-7 NginX 설치 및 설정

1. Nginx 설치

```
sudo apt update
sudo apt install nginx
```

2. Nginx 상태 체크

```
systemctl status nginx
```

3. HTTPS 적용

- Certbot 설치

```
sudo apt install certbot python3-certbot-nginx
```

- 인증서 발급

```
sudo certbot --nginx -d 도메인 이름 -d www.도메인 이름
```

- 옵션 선택 2번

```
Please choose whether or not to redirect HTTP traffic to HTTP
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
1: No redirect - Make no further changes to the webserver con
2: Redirect - Make all requests redirect to secure HTTPS acce
new sites, or if you're confident your site works on HTTPS. Y
change by editing your web server's configuration.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Select the appropriate number [1-2] then [enter] (press 'c' t
```

- Nginx 설정 파일 작성 및 다운로드 제한 설정

```
cd /etc/nginx/sites-available
sudo vim deploy-test.conf



server {

        location / {
                proxy_pass http://localhost:8082;
        }
```

```
        location /api/v1 {
                proxy_pass http://localhost:5000;
        }

                client_max_body_size 100M;
        listen 443 ssl;
        ssl_certificate /etc/letsencrypt/live/<도메인>/fullchai
        ssl_certificate_key /etc/letsencrypt/live/<도메인>/priv
}

server {

        if ($host = <도메인>) {
                return 301 https://$host$request_uri;
        }

        listen 80;
        server_name <도메인>;
                client_max_body_size 100M;
        return 404;
}
```

3. AI 서버

3-1. 영상 보간 서버

1. wsl 설정
   - 영상 보간 서버 설정을 위한 가상 환경 세팅(window powershell)

```
wsl --install

# 3. WSL2를 기본 버전으로 변경
wsl --set-default-version 2

# 설치된 리눅스 확인하기(-list, -version)
wsl -l -v
```

2. Docker 설치
3. nvidia-docker 설치

```
$ distribution=$(. /etc/os-release;echo $ID$VERSION_ID) \
   && curl -s -L https://nvidia.github.io/nvidia-docker/gp
   && curl -s -L https://nvidia.github.io/nvidia-docker/$d

$ sudo apt-get update
$ sudo apt-get install -y nvidia-docker2
```

4. Google Deep Learning Containers 설치

```
docker run --runtime=nvidia -d -p 8080:8080 -v /path/to/lo
  gcr.io/deeplearning-platform-release/{version}
```

5. ECCV2022-RIFE 프로젝트 클론

```
git clone https://github.com/hzwer/ECCV2022-RIFE.git
```

3-2. GPU 학습 서버

1. Conda 설치
3. Conda 가상환경 구축

```
conda create -n "가상 환경명"
```

4. Conda init 후 터미널 종료 후 재접속

```
conda init
```

5. Conda 가상환경 활성화

```
conda activate
```

6. Jupyter NoteBook 설치

```
pip install jupyter notebook
```

7. 설치되어있는 CUDA Version 확인

```
nvcc --version
```

8. 아래 사이트에서 버전에 맞는 PyTorch 찾기

   https://pytorch.org/get-started/locally/

9. PyTorch 설치

```
conda install pytorch==2.1.0 torchvision==0.16.0 torchaudi
```

10. Jupyter Notebook Kernel에 Conda 가상환경 추가

    a. 설치 확인

```
jupyter kernelspec list
```

    b. ipykernel 설치

```
pip install ipykernel
```

    c. Kernel에 Conda 가상환경 추가

```
python -m ipykernel install --user --name (conda가상 환경명)
```

3-3 Jupyter Note에서 AI 학습하기

1. uploda 버튼을 통해 zip 파일 업로드
2. 아래 코드를 생성하고 실행하여 압축 해제

```
import torch
!unzip -q "/경로/압축파일.zip" -d 압축이 풀리고 난 뒤의 파일명
```

3. 아래 코드를 생성하고 실행하여 AI 학습 시작

```
from ultralytics import YOLO
import os

os.environ["CUDA_DEVICE_ORDER"] = "PCI_BUS_ID"
os.environ["CUDA_VISIBLE_DEVICES"] = "6"
os.environ['KMP_DUPLICATE_LIB_OK']='True'

# Load a model
model = YOLO('yolov8n.yaml')  # build a new model from YAM
model = YOLO('yolov8n.pt')  # load a pretrained model (rec
model = YOLO('yolov8n.yaml').load('yolov8n.pt')  # build f

if __name__ == '__main__':
    # Train the model
    results = model.train(data='./datasets/data.yaml', epo
```

4. 아래 코드를 생성하고 실행하여 학습된 모델을 tfjs 형식으로 export

```
from ultralytics import YOLO

# Load the YOLOv8 model
model = YOLO('final_animal.pt')

# Export the model to tensorflow js format
model.export(format='tfjs', imgsz=640)
```

4. MySQL dump

```
-- MySQL dump 10.13  Distrib 8.0.34, for Win64 (x86_64)
--
-- Host: j10d104.p.ssafy.io    Database: grabpic
-- ------------------------------------------------------
-- Server version   8.0.22
```

```
/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESUL
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION
/*!50503 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECK
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FO
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALU
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Dumping data for table `alarm`
--

LOCK TABLES `alarm` WRITE;
/*!40000 ALTER TABLE `alarm` DISABLE KEYS */;
/*!40000 ALTER TABLE `alarm` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Dumping data for table `biology_list`
--

LOCK TABLES `biology_list` WRITE;
/*!40000 ALTER TABLE `biology_list` DISABLE KEYS */;
INSERT INTO `biology_list` VALUES (1,'식육목','개과','개속','회색
/*!40000 ALTER TABLE `biology_list` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Dumping data for table `encyclopedia`
--

LOCK TABLES `encyclopedia` WRITE;
/*!40000 ALTER TABLE `encyclopedia` DISABLE KEYS */;
INSERT INTO `encyclopedia` VALUES (1,9,13,'2024-04-04 02:04:0
```

```
/*!40000 ALTER TABLE `encyclopedia` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Dumping data for table `gallery_log`
--

LOCK TABLES `gallery_log` WRITE;
/*!40000 ALTER TABLE `gallery_log` DISABLE KEYS */;
INSERT INTO `gallery_log` VALUES (1,11,1),(2,5,4),(3,11,3),(4
/*!40000 ALTER TABLE `gallery_log` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Dumping data for table `guest_book`
--

LOCK TABLES `guest_book` WRITE;
/*!40000 ALTER TABLE `guest_book` DISABLE KEYS */;
INSERT INTO `guest_book` VALUES (1,9,5,'우와 쇠오리 어디서 수집하셨
/*!40000 ALTER TABLE `guest_book` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Dumping data for table `reports`
--

LOCK TABLES `reports` WRITE;
/*!40000 ALTER TABLE `reports` DISABLE KEYS */;
/*!40000 ALTER TABLE `reports` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Dumping data for table `subscribe`
--

LOCK TABLES `subscribe` WRITE;
/*!40000 ALTER TABLE `subscribe` DISABLE KEYS */;
```

```
INSERT INTO `subscribe` VALUES (1,11,9),(2,9,11),(5,5,9),(6,5
/*!40000 ALTER TABLE `subscribe` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Dumping data for table `user`
--

LOCK TABLES `user` WRITE;
/*!40000 ALTER TABLE `user` DISABLE KEYS */;
INSERT INTO `user` VALUES (1,'test@test.com','$2a$10$f2DBkXM4
/*!40000 ALTER TABLE `user` ENABLE KEYS */;
UNLOCK TABLES;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT *
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION *
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;

-- Dump completed on 2024-04-04 10:36:48
```