포팅 메뉴얼

목차

---

1. 개발환경

1-1 프론트엔드

- **패키지 관리 및 빌드 도구**
    - vite: ^5.2.0
- **상태 관리**

    - zustand: ^4.5.2
- **HTTP 클라이언트**
    - axios: ^1.6.8
- **라우팅**
    - react-router-dom: ^6.23.0
- **코드 스타일링 및 품질 관리**
    - prettier: ^3.2.5
    - postcss: ^8.4.38
- **유틸리티**
    - qs: ^6.12.1
- **타입스크립트**
    - typescript: ^5.2.2
    - @types/qs: ^6.9.15
    - @types/sockjs-client: ^1.5.4
    - @types/stompjs: ^2.3.9
    - @types/react: ^18.2.66
    - @types/react-dom: ^18.2.22
    - @types/node: ^20.12.11
    - @typescript-eslint/eslint-plugin: ^7.2.0
    - @typescript-eslint/parser: ^7.2.0
- **API 클라이언트**
    - @aws-sdk/client-s3: ^3.572.0
    - aws-sdk: ^2.1617.0
- **CSS 프레임워크 및 유틸리티**
    - tailwindcss: ^3.4.3
    - tailwind-scrollbar-hide: ^1.1.7
    - @iconify/tailwind: ^0.1.4
    - flowbite: ^2.3.0

- flowbite-react: ^0.9.0

- **UI 컴포넌트**
  - @material-tailwind/react: ^2.1.9
- **애니메이션**
  - framer-motion: ^11.1.7
  - react-transition-group: ^4.4.5
- **웹소켓 및 실시간 통신**
  - @stomp/stompjs: ^7.0.0
  - socketjs-client: ^1.0.2
  - stomp: ^0.1.1
  - event-source-polyfill: ^1.0.31
- **이미지 슬라이더**
  - swiper: ^11.1.1
- **쿠키 관리**
  - react-cookie: ^7.1.4
- **Toast 메시지**
  - react-hot-toast: ^2.4.1
- **Babel 플러그인**
  - babel-plugin-transform-define: ^2.1.4

- **Project Running Process**
  - 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

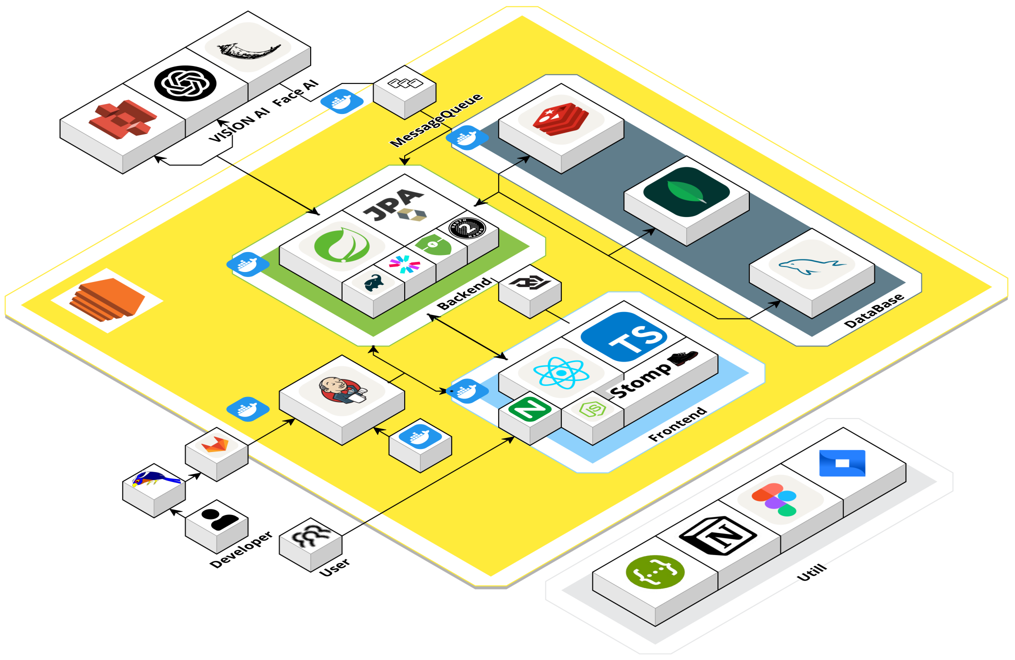
```
#server.port=3000
server.servlet.context-path=/api

#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.springframe
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**

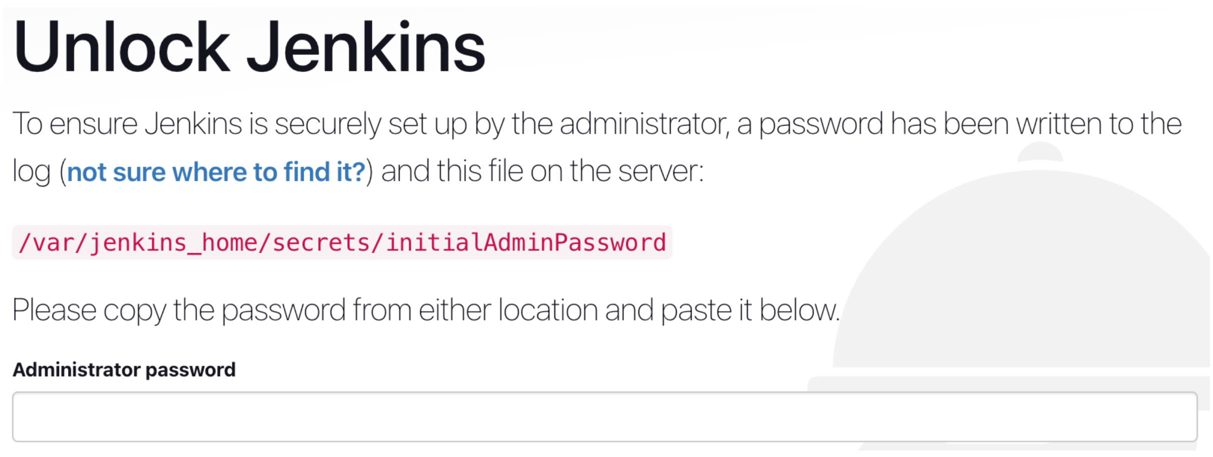
```
cd /home/ubuntu && mkdir jenkins-data

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/


Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

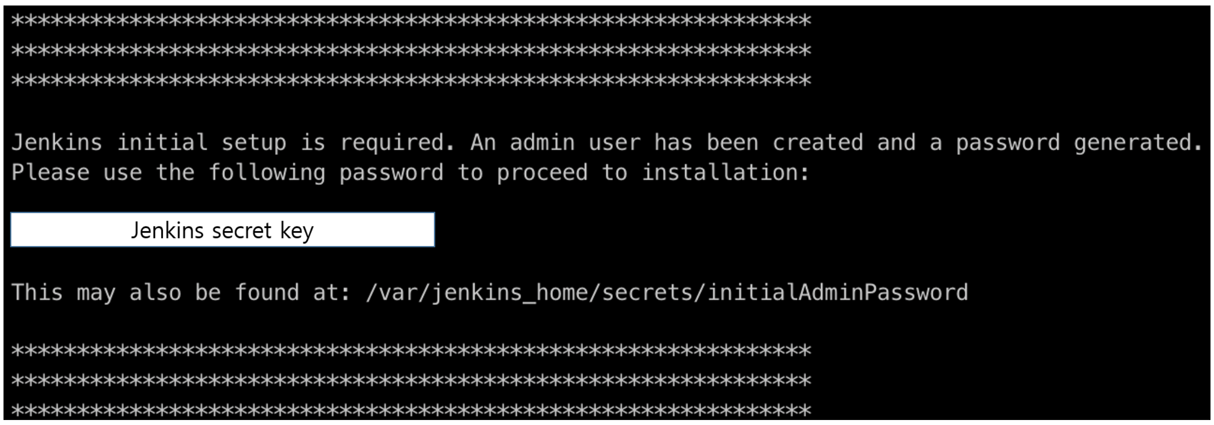
/var/jenkins_home/secrets/initialAdminPassword

Please copy the password from either location and paste it below.

Administrator password

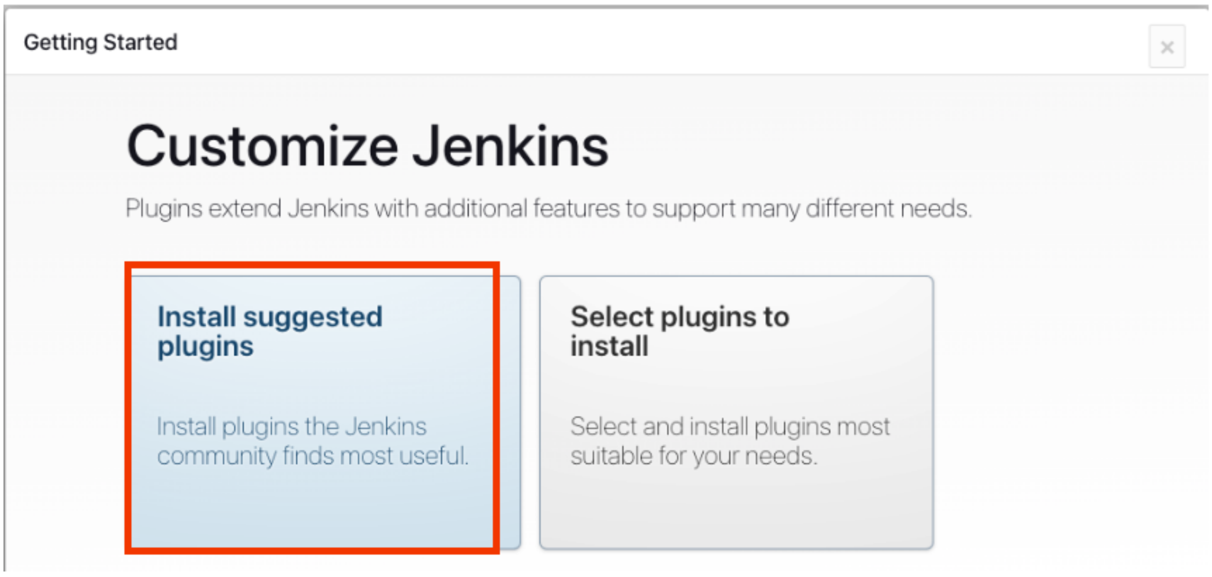
4. 젠킨스 접속 키 확인

```
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


Getting Started

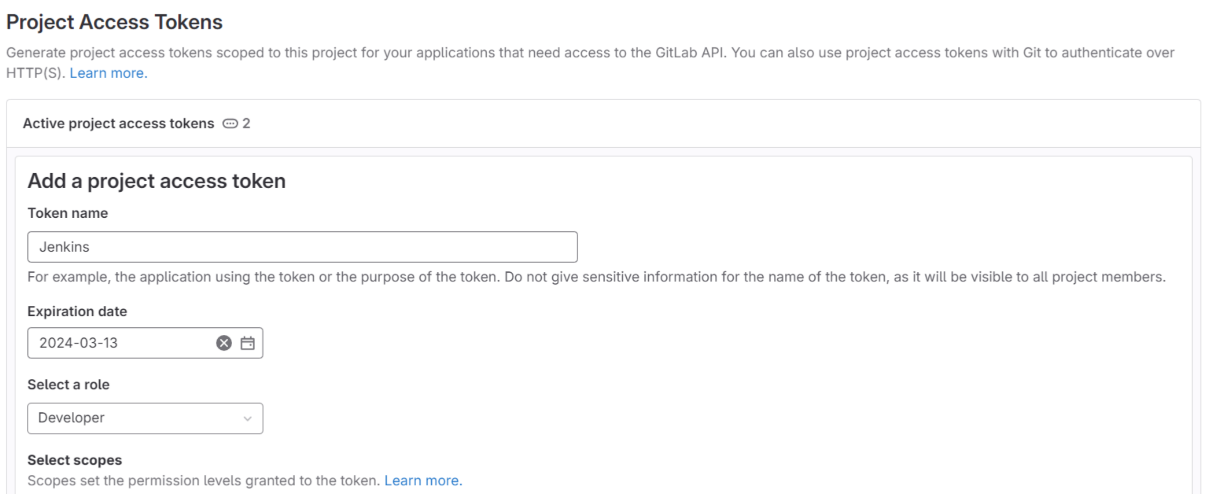
Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

Install suggested plugins
Install plugins the Jenkins community finds most useful.
Select plugins to install
Select and install plugins most suitable for your needs.

6. **GitLab 연동 설정**

깃랩 api 토큰 생성


Project Access Tokens

Generate project access tokens scoped to this project for your applications that need access to the GitLab API. You can also use project access tokens with Git to authenticate over HTTP(S). Learn more.

Active project access tokens 2

Add a project access token

Token name: Jenkins

For example, the application using the token or the purpose of the token. Do not give sensitive information for the name of the token, as it will be visible to all project members.

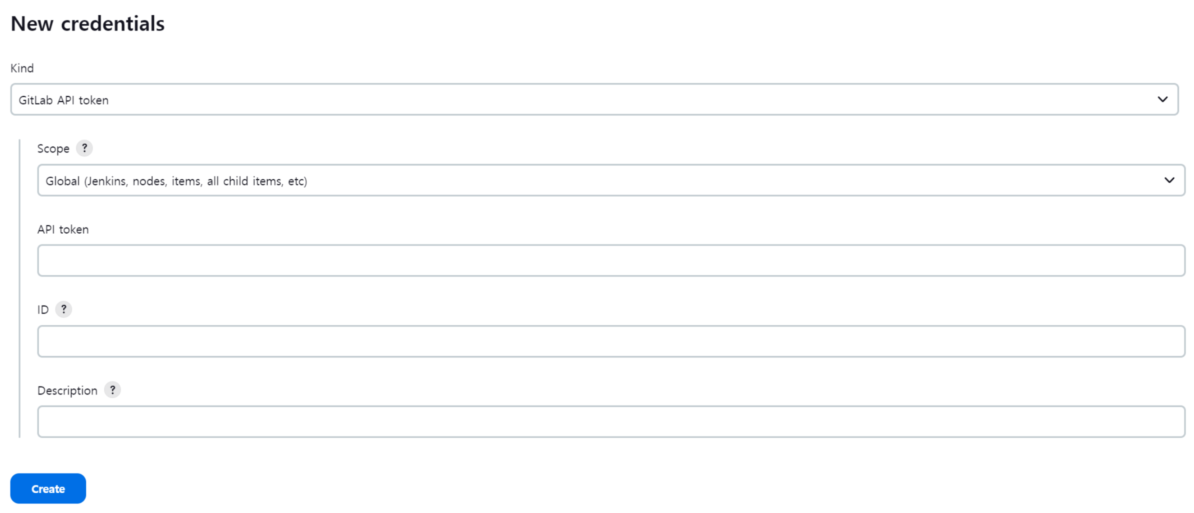
Expiration date: 2024-03-13

Select a role: Developer

Select scopes

Scopes set the permission levels granted to the token. Learn more.

Jenkins credential 추가


New credentials

Kind

GitLab API token

Scope

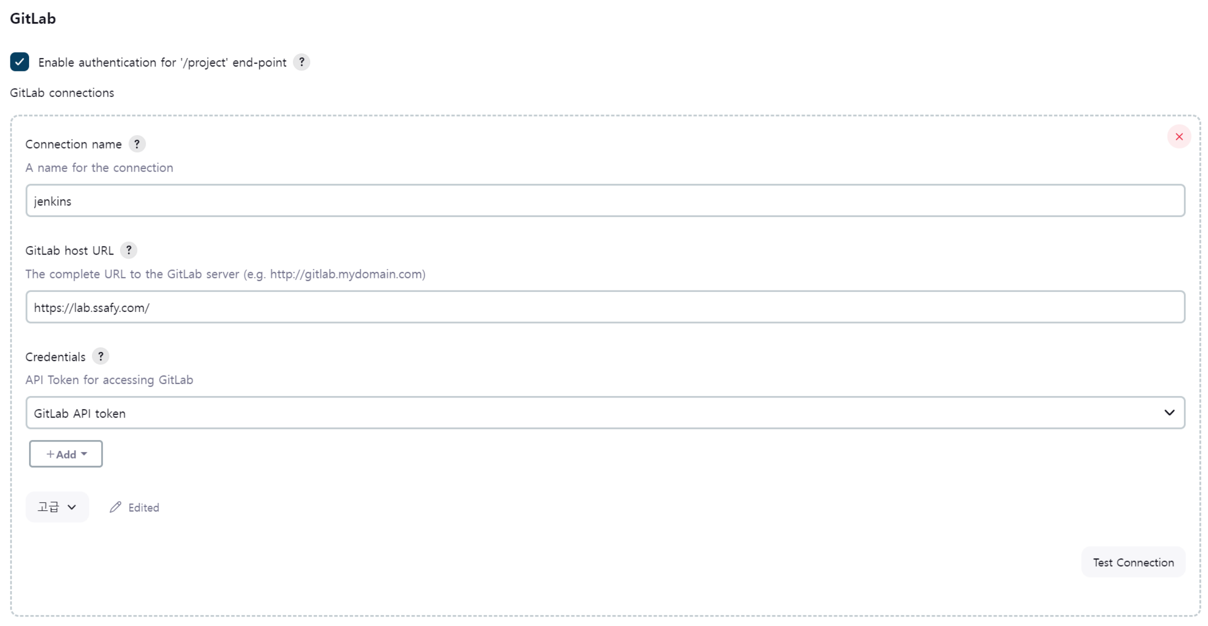
Global (Jenkins, nodes, items, all child items, etc)

API token

ID

Description

system 설정 → GitLab 설정


GitLab

☑ Enable authentication for '/project' end-point

GitLab connections

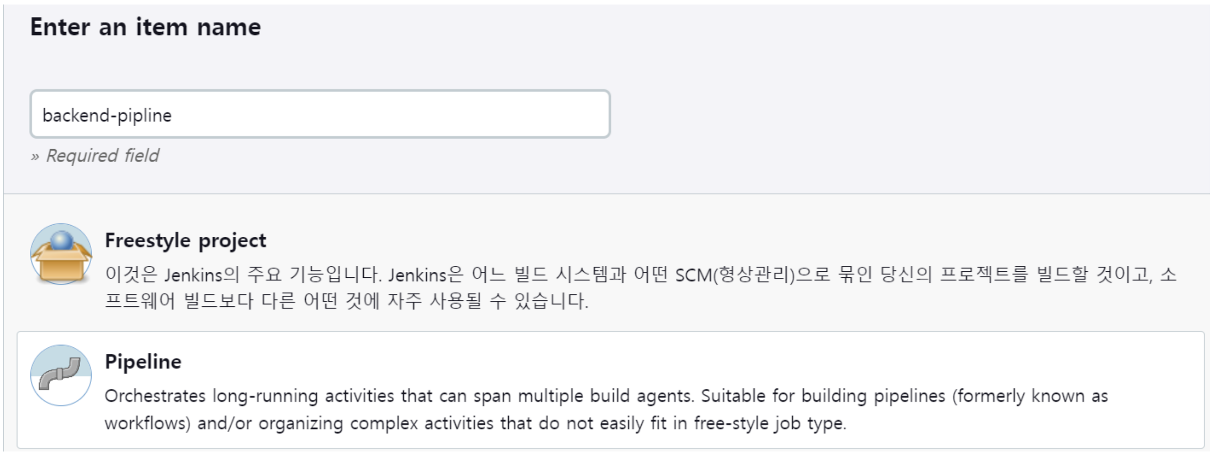
Connection name
A name for the connection
jenkins

GitLab host URL
The complete URL to the GitLab server (e.g. http://gitlab.mydomain.com)
https://lab.ssafy.com/

Credentials
API Token for accessing GitLab
GitLab API token

7. 백앤드 CI/CD 파이프라인

새로운 Item을 Pipeline으로 생성


Enter an item name

backend-pipline

» Required field

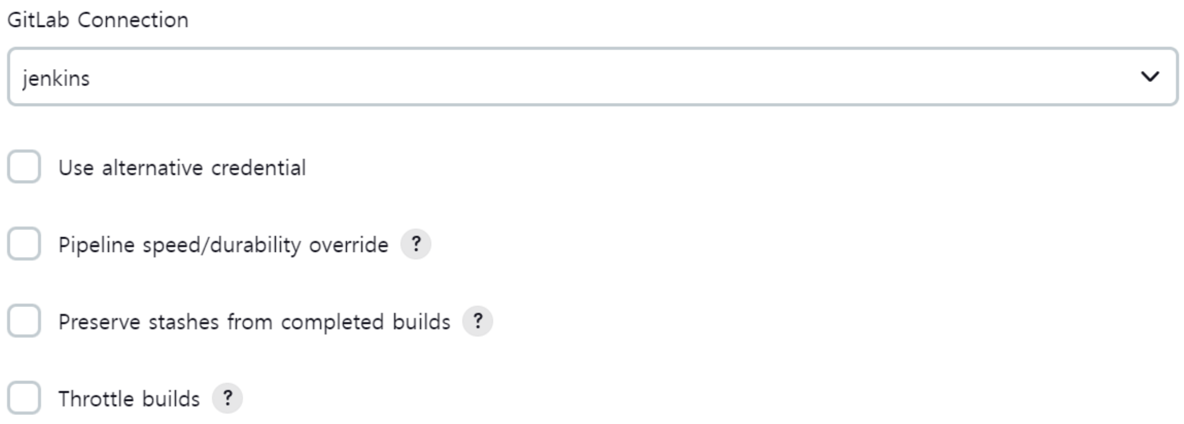
Freestyle project

이것은 Jenkins의 주요 기능입니다. Jenkins은 어느 빌드 시스템과 어떤 SCM(형상관리)으로 묶인 당신의 프로젝트를 빌드할 것이고, 소프트웨어 빌드보다 다른 어떤 것에 자주 사용될 수 있습니다.

Pipeline

Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

미리 만들어둔 Gitlab 연동 할당

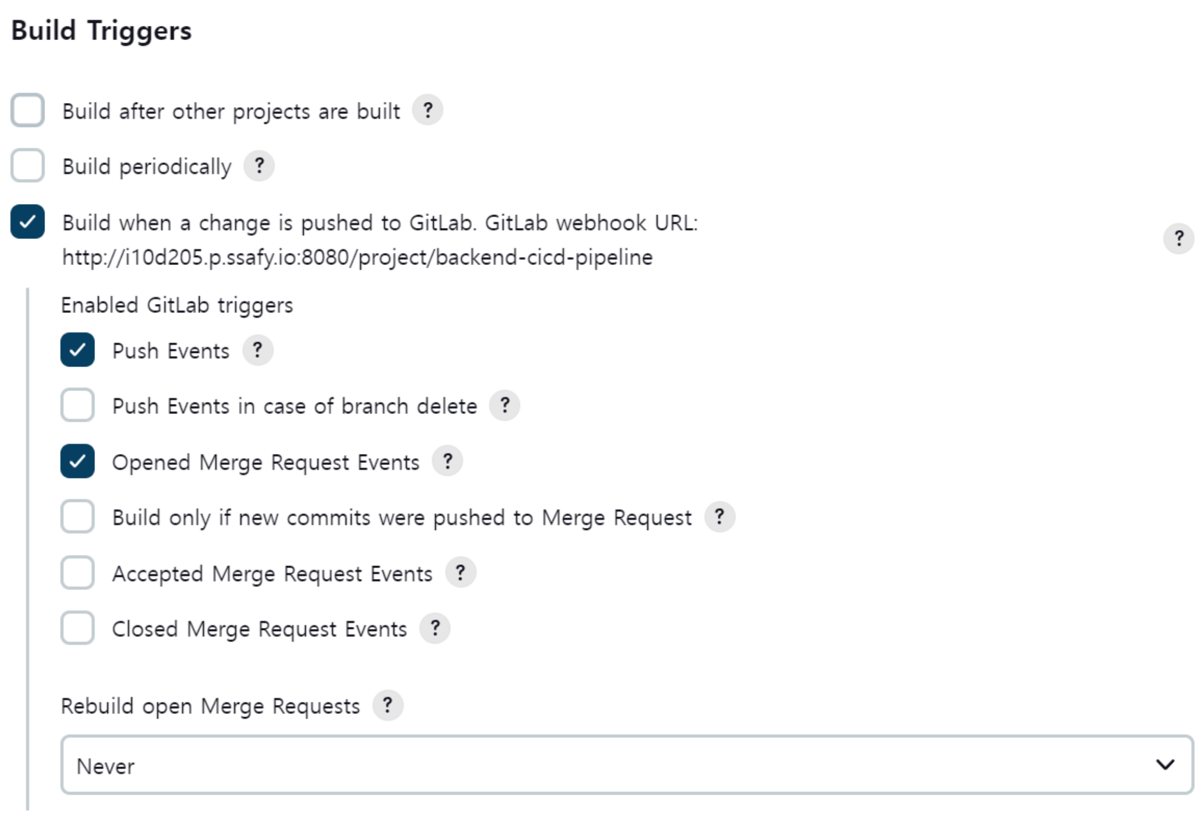

GitLab Connection

jenkins

- [ ] Use alternative credential
- [ ] Pipeline speed/durability override
- [ ] Preserve stashes from completed builds
- [ ] Throttle builds

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 ?

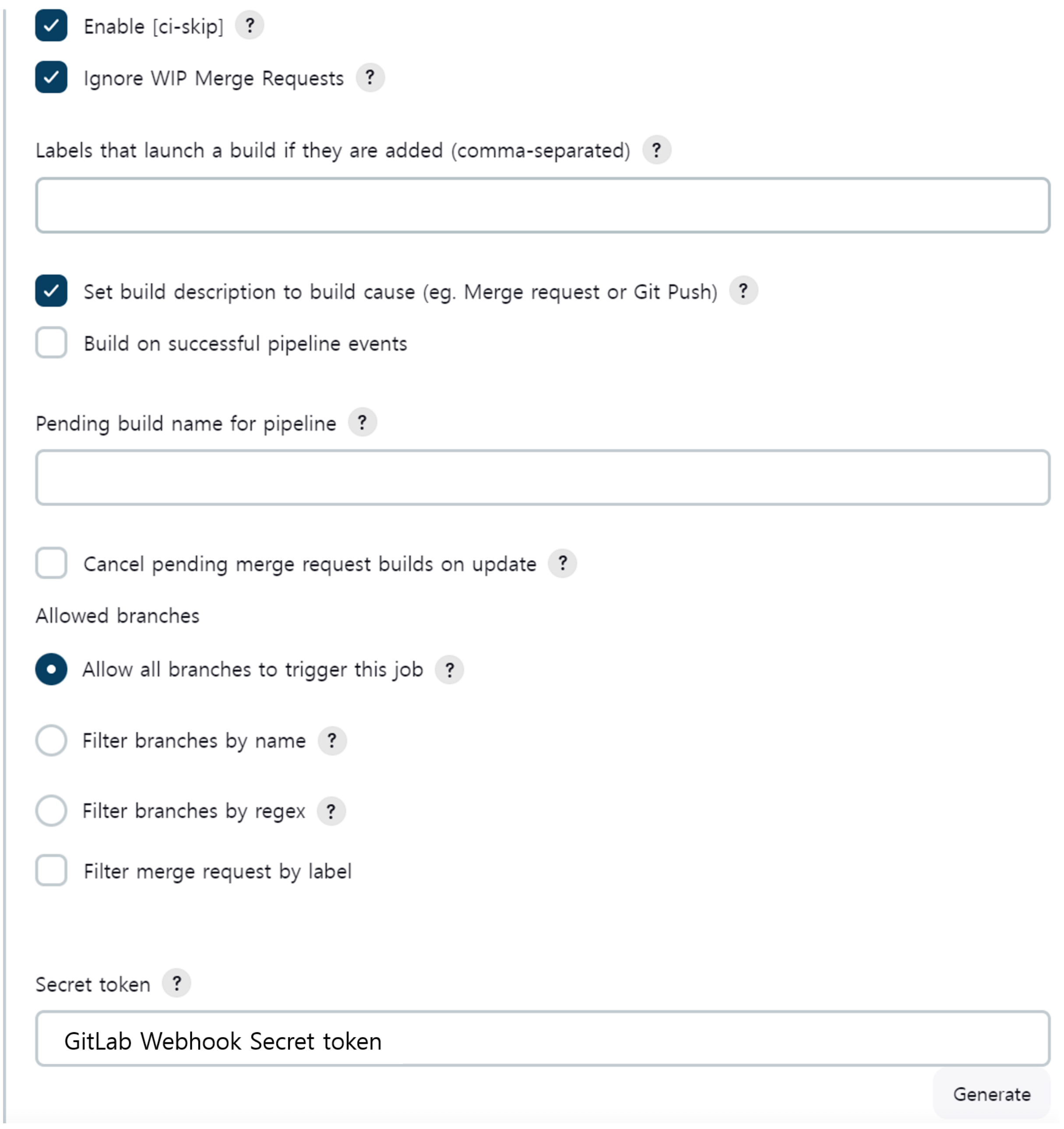
Enabled GitLab triggers

- [✓] 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 ?

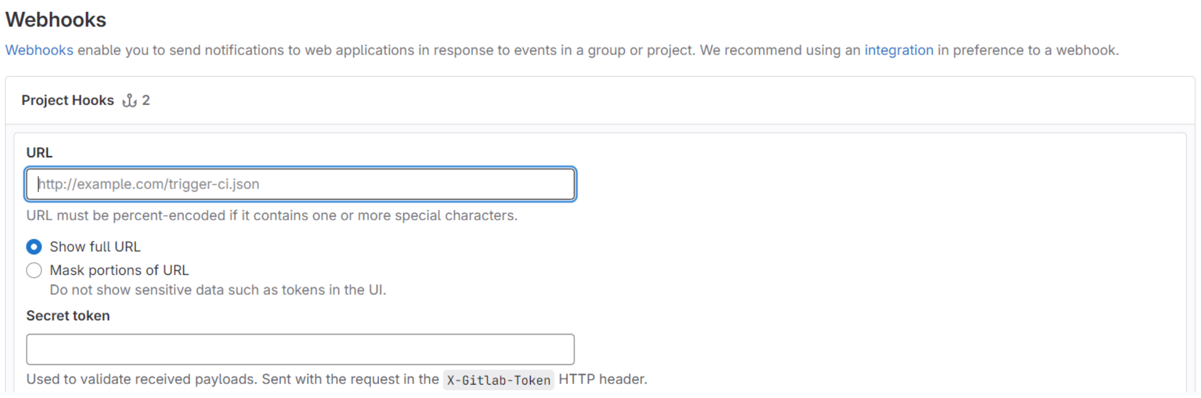
Allowed branches

- ● 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

GitLab 웹훅 추가


Webhooks

Webhooks enable you to send notifications to web applications in response to events in a group or project. We recommend using an integration in preference to a webhook.

Project Hooks 2

URL

http://example.com/trigger-ci.json

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.

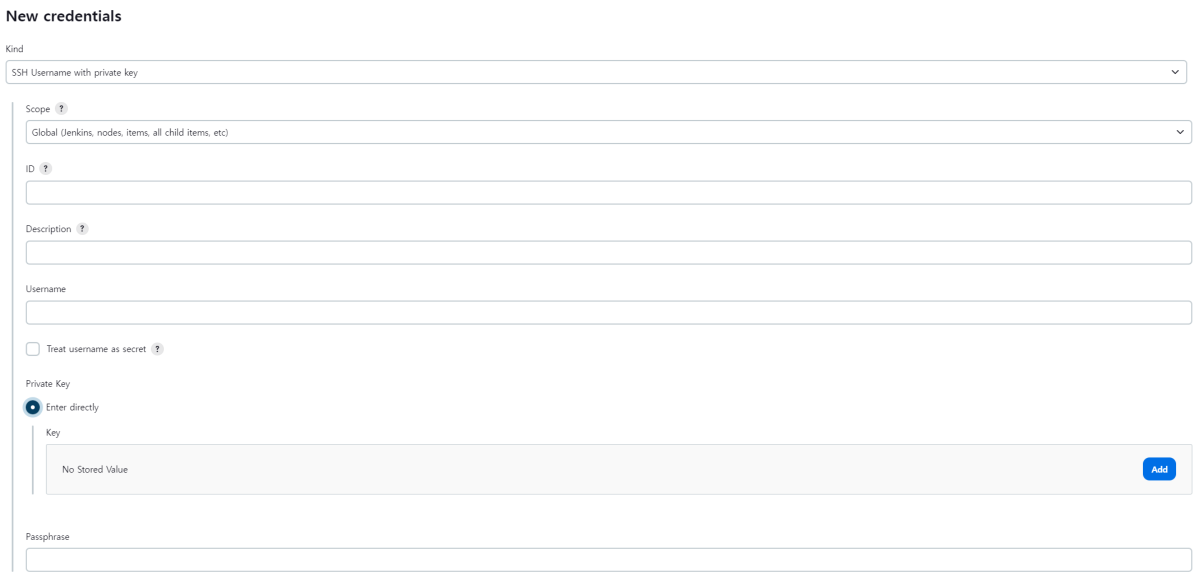
Secret token

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

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


New credentials

Kind: SSH Username with private key

Scope: Global (Jenkins, nodes, items, all child items, etc)

ID

Description

Username

Treat username as secret

Private Key

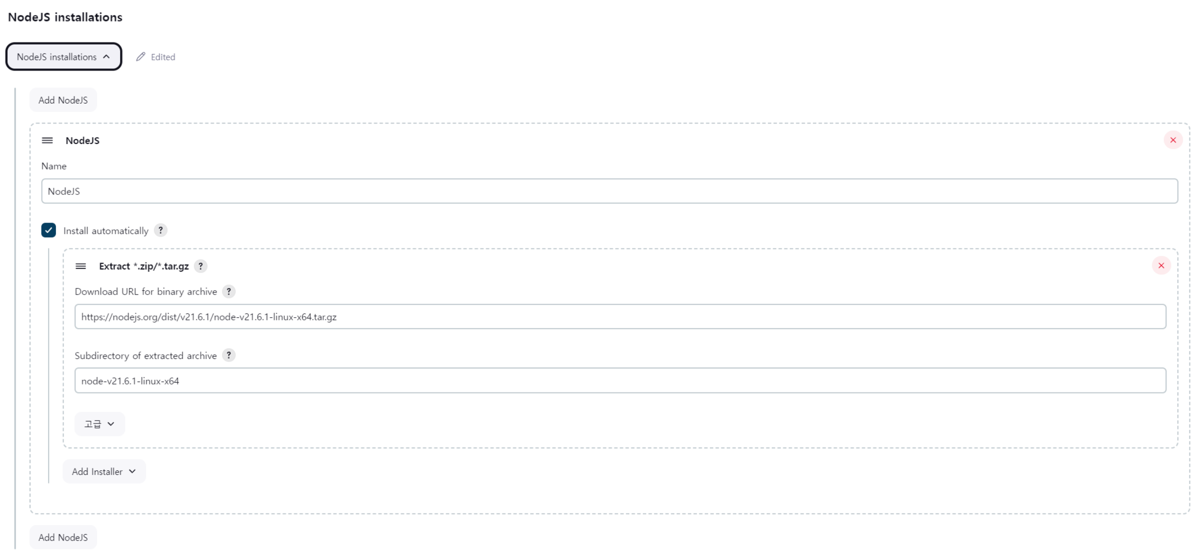
[✓ Enter directly]

Key: No Stored Value

Passphrase

- key 에 SSH pem 파일 정보 복사 후 붙여넣기

9. **NodeJs 설정**



NodeJS

Name: NodeJS

[✓] Install automatically

Extract *.zip/*.tar.gz

Download URL for binary archive: https://nodejs.org/dist/v21.6.1/node-v21.6.1-linux-x64.tar.gz

Subdirectory of extracted archive: node-v21.6.1-linux-x64

10. 백엔드 파이프라인 스크립트

```
pipeline {
    agent any

    tools {
        gradle 'gradle'
    }

    stages {
        stage('Clone') {
            steps {
                git branch: 'BE', credentialsId: 'gitlab', ur
            }
        }
        stage('Build') {
            steps{
                dir("./back/ukiki"){
                    sh 'java -version'
                    sh 'chmod +x gradlew && ./gradlew clean b
                }
            }
        }
        // stage('sonarqube') {
```

```
        //     steps{
        //         dir("./back/ukiki"){
        //             withSonarQubeEnv('sonarqube'){
        //                 sh './gradlew sonarqube'
        //             }
        //         }
        //     }
        // }
        stage('Deploy') {
            steps {
                sshagent(credentials: ['ssh_key']) {
                    sh 'ssh -o StrictHostKeyChecking=no ubunt
                    sh 'pwd'
                                              //빌드파일을 서비
                    sh 'scp /var/jenkins_home/workspace/BE/bac
                                              //컨테이너 생성을
                    sh 'ssh ubuntu@k10d202.p.ssafy.io "cd /hom
                }
            }
        }
    }
    post {
        success {
            script {
                def Author_ID = sh(script: "git show -s --pre
                def Author_Name = sh(script: "git show -s --p
                mattermostSend (color: 'good',
                message: "[${env.JOB_NAME} #${env.BUILD_NUMBE
                )
            }
        }
        failure {
            script {
                def Author_ID = sh(script: "git show -s --pre
                def Author_Name = sh(script: "git show -s --p

                mattermostSend (color: 'danger',
                message: "[${env.JOB_NAME} #${env.BUILD_NUMBE
```

```
                    )
                }
            }
        }

}
```

11. **백엔드 DockerFile**

```
FROM openjdk:21

EXPOSE 5000

COPY ukiki-0.0.1-SNAPSHOT.jar back.jar

ENTRYPOINT ["java", "-jar", "-Dencryptor.key=I*EsAlz4Hwjnhd+j
```

12. **백엔드 run.sh**

```
echo "start build image"
echo "=====================================================
sudo docker build -t backend/app . #이미지 빌드
echo "=====================================================
echo "complete built"
echo "=====================================================
echo "is any same name container running?" #같은 이름으로 실행중인
sudo docker ps -a --filter "name = backend" | grep -q . && su
echo "=====================================================
echo "docker run"
sleep 5
sudo docker run -p 5000:5000 -d --name=backend -e TZ=Asia/Seo
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', u
            }
        }
        stage('build') {
            steps{
                dir('front'){
                    nodejs(nodeJSInstallationName: 'nodeJS'){
                          sh 'node --version'
                          sh 'npm install -g yarn'
                          sh 'yarn install && yarn build'
                    }
                }
            }
        }
    //     stage("sonarqube") {
    //      steps{
    //              script{
    //              def scannerHome = tool 'sonar-scanner';
    //                 withSonarQubeEnv(installationName:'son
    //                      sh "${scannerHome}/bin/sonar-scan
    //              }
    //              }
    //      }
    //      }
        stage('tar') {
            steps {
                dir('front'){
                    sh 'tar -cvf dist.tar dist' //빌드 파일 압축
                }
            }
        }
        stage('ssh') {
         steps {
            dir('front'){
                sshagent(credentials: ['ssh_key']) {
                    sh 'ls'
                    sh 'ssh -o StrictHostKeyChecking=no ubunt
                                        //빌드 파일 서버 전송
                    sh 'scp dist.tar ubuntu@k10d202.p.ssafy.i
                }
            }
         }
       }
       stage('unpack'){
           steps {
               sshagent(credentials: ['ssh_key']) {
                                        //빌드 파일 압축 해제
                    sh 'ssh ubuntu@k10d202.p.ssafy.io "cd /ho
                }
           }
       }
       stage('run.sh'){
           steps {
               sshagent(credentials: ['ssh_key']) {
                                        //컨테이너 생성을 위한 쉘
                    sh 'ssh ubuntu@k10d202.p.ssafy.io "cd /ho
                }
           }
       }
   }
   post {
       success {
           script {
               def Author_ID = sh(script: "git show -s --pre
                def Author_Name = sh(script: "git show -s --p
                mattermostSend (color: 'good',
                message: "[${env.JOB_NAME} #${env.BUILD_NUMBE
                )
            }
        }
        failure {
            script {
                def Author_ID = sh(script: "git show -s --pre
                def Author_Name = sh(script: "git show -s --p

                mattermostSend (color: 'danger',
                message: "[${env.JOB_NAME} #${env.BUILD_NUMBE
                )
            }
        }
    }
}
```

14. **프론트엔드 DockerFile**

```

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/conf.d/
# 8082 포트 오픈
EXPOSE 8082
```

15. **프론트엔드 run.sh**

```


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

13. **MQ 파이프라인 스크립트**

```
pipeline {
    agent any

    tools {
        gradle 'gradle'
    }

    stages {
        stage('Clone') {
            steps {
                git branch: 'MQ', credentialsId: 'gitlab', ur
            }
        }
        stage('Build') {
            steps{
                dir("./MQ/ukkikki"){
                    sh 'pwd'
                    sh 'java -version'
                    sh 'chmod +x gradlew && ./gradlew clean b
                }
            }
        }
        stage('Deploy') {
            steps {
                sshagent(credentials: ['ssh_key']) {
                    sh 'ssh -o StrictHostKeyChecking=no ubunt
                    sh 'pwd'
                                                //빌드파일을 서
                    sh 'scp /var/jenkins_home/workspace/MQ/MQ
                                                //컨테이너 생성
                    sh 'ssh ubuntu@k10d202.p.ssafy.io "cd /ho
                }
            }
        }
    }
    post {
        success {
            script {
                def Author_ID = sh(script: "git show -s --pre
                def Author_Name = sh(script: "git show -s --p
                mattermostSend (color: 'good',
                message: "[${env.JOB_NAME} #${env.BUILD_NUMBE
                )
            }
        }
        failure {
            script {
                def Author_ID = sh(script: "git show -s --pre
                def Author_Name = sh(script: "git show -s --p

                mattermostSend (color: 'danger',
                message: "[${env.JOB_NAME} #${env.BUILD_NUMBE
                )
            }
        }
    }

}
```

14. **MQ DockerFile**

```
FROM openjdk:21

EXPOSE 5555

COPY ukkikki-0.0.1-SNAPSHOT.jar back.jar

ENTRYPOINT ["java", "-jar" , "/back.jar"]
```

15. **프론트엔드 run.sh**

```


echo "start build image"
echo "==========================================================
sudo docker build -t mq/app . #이미지 빌드
echo "==========================================================
echo "complete built"
echo "==========================================================
echo "is any same name container running?" #같은 이름으로 실행중인
sudo docker ps -a --filter "name = mq" | grep -q . && sudo do
echo "==========================================================
echo "docker run"
sleep 5
sudo docker run -p 5555:5555 -d --name=mq -e TZ=Asia/Seoul mq
echo "=====================================================
echo "rmi process running" #같은 이름의 이미지가 있다면 삭제
sudo docker rmi -f $(sudo docker images -f "dangling=true" -q
```

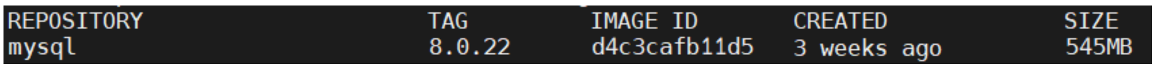
3-4 MySQL 컨테이너 생성

1. **MySQL Docker Image 다운로드**

   `docker pull mysql:8.0.22`

2. **다운로드 된 Docker Image 확인**

   `docker images -a`



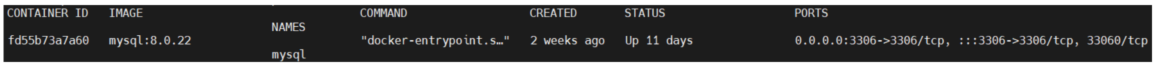
```
REPOSITORY                TAG        IMAGE ID       CREATED        SIZE
mysql                     8.0.22     d4c3cafb11d5   3 weeks ago    545MB
```

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
```

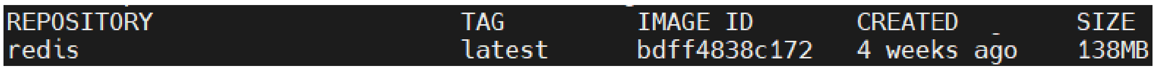
3-5 Redis 컨테이너 생성

1. **Redis Docker Image 다운로드**

   `docker pull --platform linux/amd64 redis`

2. **다운로드 된 Docker Image 확인**

   `docker images -a`



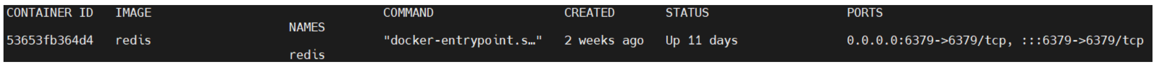
```
REPOSITORY                    TAG        IMAGE ID       CREATED       SIZE
redis                         latest     bdff4838c172   4 weeks ago   138MB
```

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
                                    NAMES
53653fb364d4   redis                         "docker-entrypoint.s…"   2 weeks ago   Up 11 days                   0.0.0.0:6379->6379/tcp, :::6379->6379/tcp
                                    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

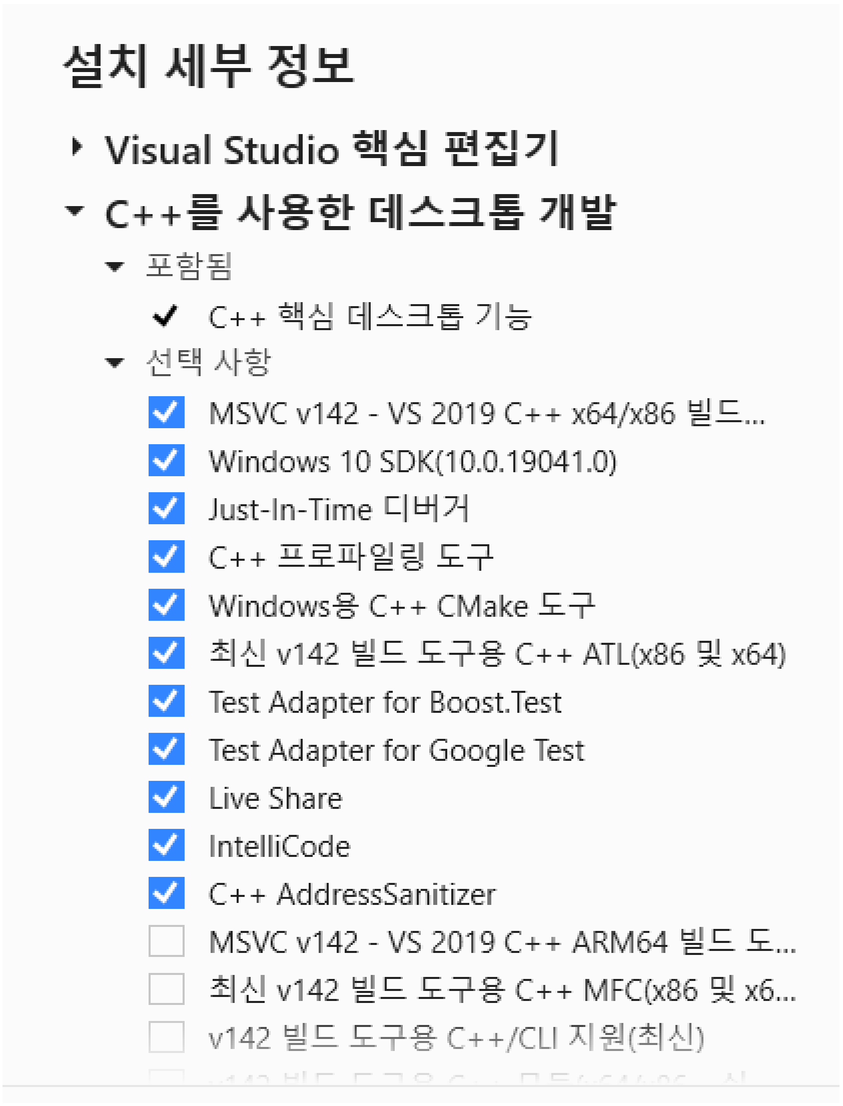
3-1. 개발 환경 설정

1. GPU 가속을 위한 CUDA 및 cudnn 설치
    a. 현재 사용중인 GPU와 dlib 요구사항을 고려하여 CUDA 10.2 / cudnn 8.0 설치

2. CMake 설치
    a. https://cmake.org/download/ 최신버전 설치

3. Visual Studio 2019 설치

```
https://visualstudio.microsoft.com/ko/thank-you-download
ing-visual-studio/?sku=Community&rel=16
```



설치 세부 정보

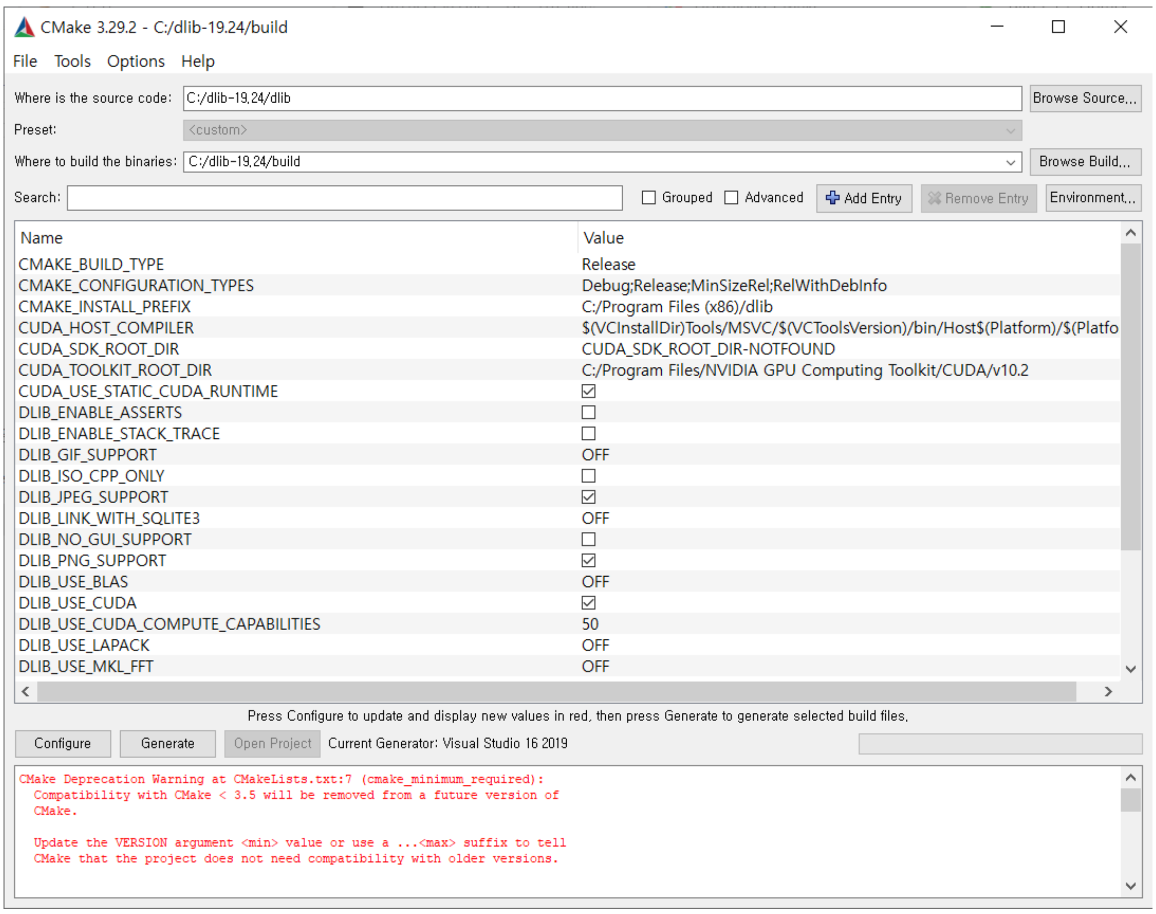
- Visual Studio 핵심 편집기
- C++를 사용한 데스크톱 개발
  - 포함됨
    - ✓ C++ 핵심 데스크톱 기능
  - 선택 사항
    - [✓] MSVC v142 - VS 2019 C++ x64/x86 빌드...
    - [✓] Windows 10 SDK(10.0.19041.0)
    - [✓] Just-In-Time 디버거
    - [✓] C++ 프로파일링 도구
    - [✓] Windows용 C++ CMake 도구
    - [✓] 최신 v142 빌드 도구용 C++ ATL(x86 및 x64)
    - [✓] Test Adapter for Boost.Test
    - [✓] Test Adapter for Google Test
    - [✓] Live Share
    - [✓] IntelliCode
    - [✓] C++ AddressSanitizer
    - [ ] MSVC v142 - VS 2019 C++ ARM64 빌드 도...
    - [ ] 최신 v142 빌드 도구용 C++ MFC(x86 및 x6...
    - [ ] v142 빌드 도구용 C++/CLI 지원(최신)

4. dlib 설치

   a. http://dlib.net/ 최신버전 설치

   b. 압축파일 해제

   c. C:\Program Files\CMake\bin\cmake-gui.exe cmake 실행



Where is the source code: C:/dlib-19.24/dlib

Preset: <custom>

Where to build the binaries: C:/dlib-19.24/build

Search:

Name	Value
CMAKE_BUILD_TYPE	Release
CMAKE_CONFIGURATION_TYPES	Debug;Release;MinSizeRel;RelWithDebInfo
CMAKE_INSTALL_PREFIX	C:/Program Files (x86)/dlib
CUDA_HOST_COMPILER	$(VCInstallDir)Tools/MSVC/$(VCToolsVersion)/bin/Host$(Platform)/$(Platfo
CUDA_SDK_ROOT_DIR	CUDA_SDK_ROOT_DIR-NOTFOUND
CUDA_TOOLKIT_ROOT_DIR	C:/Program Files/NVIDIA GPU Computing Toolkit/CUDA/v10.2
CUDA_USE_STATIC_CUDA_RUNTIME	☑
DLIB_ENABLE_ASSERTS	☐
DLIB_ENABLE_STACK_TRACE	☐
DLIB_GIF_SUPPORT	OFF
DLIB_ISO_CPP_ONLY	☐
DLIB_JPEG_SUPPORT	☑
DLIB_LINK_WITH_SQLITE3	OFF
DLIB_NO_GUI_SUPPORT	☐
DLIB_PNG_SUPPORT	☑
DLIB_USE_BLAS	OFF
DLIB_USE_CUDA	☑
DLIB_USE_CUDA_COMPUTE_CAPABILITIES	50
DLIB_USE_LAPACK	OFF
DLIB_USE_MKL_FFT	OFF

Press Configure to update and display new values in red, then press Generate to generate selected build files.

Current Generator: Visual Studio 16 2019

```
CMake Deprecation Warning at CMakeLists.txt:7 (cmake_minimum_required):
  Compatibility with CMake < 3.5 will be removed from a future version of
  CMake.

  Update the VERSION argument <min> value or use a ...<max> suffix to tell
  CMake that the project does not need compatibility with older versions.
```

- sourece code 와 build 디렉토리 지정후 Configure 버튼 클릭
- DLIB_USE_CUDA 항목으로 GPU 인식 여부 확인
- Generate 버튼 클릭
- Open Project 버튼으로 프로젝트 빌드

5. cmd 창에서 dlib이 설치된 폴더로 이동한 후 설치

```
cd dlib-19.22
```

```
python setup.py install
```

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
```

5. MQ 서버

5-1 . 개발 환경

- IntelliJ : 2023.3.2
- Spring Boot : 3.2.3
- JDK : 21

5-2. application.yaml

```yaml
server:
  port: 5555
  servlet:
    context-path: /mq

spring:
  servlet:
    multipart:
      max-file-size: -1
      max-request-size: -1
```