

# Back-end, Infra 포팅메뉴얼

## 버전 정보



### Back-end

- IntelliJ: 2023.3
- JAVA : JDK 17.0.9
- Spring : 6.1.3
- spring boot: 3.2.2
- spring security: 6.2.1
- mysql: 8.3.0



### Infra

- Nginx: 1.18.0
- Docker: 25.0.1
- Jenkins: 2.426.3

## 1. NginX

### Nginx 설치

```
sudo apt-get install nginx
```

cerbot 설치 및 SSL 인증서 발급

```
sudo apt update
sudo apt upgrade
sudo add-apt-repository ppa:certbot/certbot
sudo apt install python3-certbot-nginx

sudo certbot --nginx -d i10b111.p.ssafy.io
```

### Nginx /etc/nginx/sites-available/default 파일 설정

```
// 필요한 최소한의 설정들만 남기고 다 들어냈다.
server {
// 이 서버의 이름을 설정한다. cerbot 설정에서 필요한 이름은 이 이름을 따라간
    server_name i10b111.p.ssafy.io;

// :80/ 포트로 들어온 요청을 :3000/으로 보낸다. 마지막에 /를 꼭 붙여야 한다
// 프론트 서버이다.
    location / {
        proxy_pass http://localhost:3000/;
    }

// /api/로 들어온 요청을 :8728/로 보낸다.
// 백엔드 메인서버이다.
    location /api/ {
        proxy_pass http://localhost:8728/;
    }

// /dev/api/로 들어온 요청을 :8729/로 보낸다.
// 백엔드 개발 서버이다.
    location /dev/api/ {
        proxy_pass http://localhost:8729/;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
    }
}
```

```

}

// cerbot이 만들어준 코드이다. :80으로 들어오면 :443으로 보낸다.
listen 443 ssl; # managed by Certbot
ssl_certificate /etc/letsencrypt/live/i10b111.p.ssafy.io/fullchain.pem; # managed by Certbot
ssl_certificate_key /etc/letsencrypt/live/i10b111.p.ssafy.io/privatekey.pem; # managed by Certbot
include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot

}

server {
    if ($host = i10b111.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

    listen 80;
    server_name i10b111.p.ssafy.io;
    return 404; # managed by Certbot
}

```

이렇게만 한다고 끝난 것이 아니다.

/etc/nginx/nginx.conf 의 설정을 보면 /etc/nginx/sites-available/default를 직접 접근하는 것이 아니라

/etc/nginx/sites-enabled/default를 접근하는 것을 알 수 있다.

그래서 /etc/nginx/sites-enabled/에 /etc/nginx/sites-available/default를 링크 파일을 만들어 줘야한다.

반복되는 설정에 이 부분의 alias를 만들었다.

```
// 기존에 있던 링크 파일을 지운다.
alias nginx='sudo rm -rf /etc/nginx/sites-enabled/default;
// 다시 링크 파일을 설정한다.
sudo ln -s /etc/nginx/sites-available/default /etc/nginx/sites-enabled/default
// Nginx를 리로드한다.
sudo systemctl reload nginx'
```

## Docker

### 도커 설치

도커 공식 홈페이지 Ubuntu(<https://docs.docker.com/engine/install/ubuntu>)을 참고해서 설치하였다.

```
// 도커의 공식 GPG키를 추가한다.
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/gpg -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

// apt를 설정한다.
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc \
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update

// apt를 이용하여 도커를 설치한다.
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin

// 도커가 잘 설치됐는데 hello-world 이미지를 실행시켜본다.
sudo docker run hello-world
```

# Jenkins

## Jenkins Docker image 설치 및 실행

```
// jenkins 이미지를 최신으로 받고 실행 시킨다
// --name으로 컨테이너 이름을 설정해준다.
// -p로 포트를 설정해준다.
// -d로 백그라운드에서 실행시킨다.
// -v로 바인드 마운트를 하여 컨테이너가 없어져도 데이터를 저장할 수 있게한다.
docker run --name jenkins -d -p 8080:8080 -v /home/jenkins:/var,
```

## Jenkins 주소

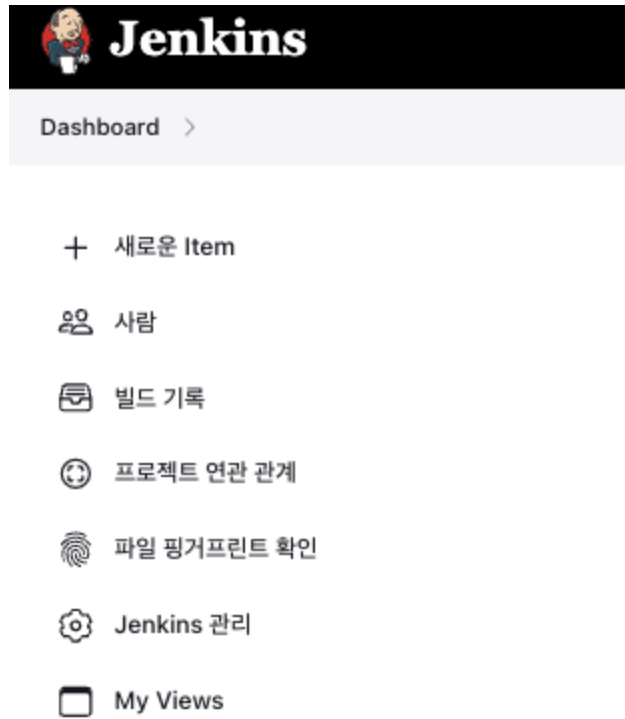
```
{{도메인명}}:8080
```

## Jenkins 초기 비밀번호 확인

```
docker logs jenkins-docker
```

## 파이프라인 구축 방법

1. 새로운 아이템 추가




왼쪽 위에 보이는 **+ 새로운 Item** 을 누른다


## 2. 아이템 이름 및 유형 설정

Enter an item name


» 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.


**Multi-configuration project**


다양한 환경에서의 테스트, 플레폼 특성 빌드, 기타 등등 처럼 다수의 서로다른 환경설정이 필요한 프로젝트에 적합함.


**Folder**

Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.



**Multibranch Pipeline**

Creates a set of Pipeline projects according to detected branches in one SCM repository.


**Organization Folder**

Creates a set of multibranch project subfolders by scanning for repositories.

If you want to create a new item from other existing, you can use this option:


**Copy from**

Type to autocomplete

OK

Freestyle Projects는 사용하기는 쉽지만 커스텀이 어렵고  
Pipeline은 사용하기는 어려울 수 있어도 커스텀이 쉽다.

### 3. 원하는 플러그인 추가

왼쪽 위의 Dashboard > Jenkins 관리 > Plugins 클릭

Dashboard > Jenkins 관리 > Plugins

Plugins

Updates11

Available plugins

Installed plugins

Advanced settings

Download progress

Search plugin updates

Update

<input type="checkbox"/>	이름 ↓	Released	설치됨
<input type="checkbox"/>	Branch API 2.1148.vce12cfcdf090 <a href="#">Library plugins (for use by other plugins)</a> This plugin provides an API for multiple branch based projects.	3 days 4 hr ago	2.1144.v1425d1c3d5a_7
<input type="checkbox"/>	Credentials 13119.v7eb_51b_3a_c97b_ <a href="#">credentials</a> <a href="#">Library plugins (for use by other plugins)</a> This plugin allows you to store credentials in Jenkins.	20 days ago	1311.vcf0a_900b_37c2
<input type="checkbox"/>	Durable Task 550.v0930093c4b_a_6 <a href="#">Build Tools</a> <a href="#">Library plugins (for use by other plugins)</a> Library offering an extension point for processes which can run outside of Jenkins yet be monitored.	8 days 19 hr ago	543.v262f6a_803410
<input type="checkbox"/>	GitHub 1.38.0 <a href="#">External Site/Tool integrations</a> <a href="#">github</a> This plugin integrates GitHub to Jenkins.	13 days ago	1.37.3.1
<input type="checkbox"/>	Gradle 2.10 <a href="#">Build Tools</a> This plugin allows Jenkins to invoke Gradle build scripts directly.	3 days 0 hr ago	2.9
<input type="checkbox"/>	Joda Time API 2.12.7-29.v5a_3a_82269a_ <a href="#">Library plugins (for use by other plugins)</a> This plugin provides the Joda Time APIs (v2.12.7) for other plugins.	7 days 0 hr ago	2.12.6-21.vca_5d74418fb_7
<input type="checkbox"/>	JUnit 1259.v65ffcf24a_s8 <a href="#">Build Reports</a> Allows JUnit-format test results to be published.	14 days ago	1256.v002534a_5f33e
<input type="checkbox"/>	OkHttp 4.11.0-172.vda_da_7feeb_cde This plugin provides <a href="#">OkHttp</a> for other plugins.	17 days ago	4.11.0-157.v6852a_a_3a_ec11
<input type="checkbox"/>	Pipeline: API 1291.v51fd2a_625da_7 <a href="#">Library plugins (for use by other plugins)</a> <a href="#">Miscellaneous</a> Plugin that defines Pipeline API.	17 days ago	1283.v99c10937efcb_
<input type="checkbox"/>	Pipeline: Nodes and Processes 1331.vcb2fed35334 <a href="#">pipeline</a> <a href="#">Miscellaneous</a>	6 days 22 hr ago	1317.v5337e0c1fe28

그러면 이런 화면이 나올 것이다.

여기서 GitLab Plugin과 Mattermost Notification Plugin을 설치한다.

#### 4. 플러그인 설정 추가

Dashboard > Jenkins 관리 > System 클릭

GitLab 탭



#### GitLab

☒ Enable authentication for 'project' end-point ?

##### GitLab connections

Connection name ?  
A name for the connection

sonmandu\_gitlab

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

+ Add

고급

Success

Test Connection

추가

최종적으로 이렇게 채워 넣을 것이다. 하나씩 따라해보자.

Connection name은 이 연결의 이름이니까 헛갈리지 않게 고유한 이름을 적으면 된다.

GitLab host URL은 자신이 이용하고 있는 GitLab 서버주소를 입력하면 된다.

**Credentials은 GitLab에서 토큰을 받아와야 한다.**

Add를 누르고 Kind에 GitLab API token을 선택한다.

#### Jenkins Credentials Provider: Jenkins

##### Add Credentials

Domain

Global credentials (unrestricted)

Kind

GitLab API token

Scope ?

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

API token

ID ?

Description ?

Add

Cancel

그러면 이런 화면이 나올 것이다.

여기서 API 토큰을 가져오기 위해 GitLab으로 접속한다.

GitLab에서 profile > Preference > Access Tokens 를 클릭

여기서 Add new token을 누르고

**Personal Access Tokens**

You can generate a personal access token for each application you use that needs access to the GitLab API. You can also use personal access tokens to authenticate against Git over HTTP. They are the only accepted password when you have Two-Factor Authentication (2FA) enabled.

Active personal access tokens 4

### Add a personal access token

Token name

For example, the application using the token or the purpose of the token.

Expiration date

2024-03-17

Select scopes

Scopes set the permission levels granted to the token. [Learn more.](#)

- ☒ **api**  
Grants complete read/write access to the API, including all groups and projects, the container registry, the dependency proxy, and the package registry.
- ☒ **read\_api**  
Grants read access to the API, including all groups and projects, the container registry, and the package registry.
- ☒ **read\_user**  
Grants read-only access to the authenticated user's profile through the /user API endpoint, which includes username, public email, and full name. Also grants access to read-only API endpoints under /users.
- ☐ **create\_runner**  
Grants create access to the runners.
- ☐ **k8s\_proxy**  
Grants permission to perform Kubernetes API calls using the agent for Kubernetes.
- ☒ **read\_repository**  
Grants read-only access to repositories on private projects using Git-over-HTTP or the Repository Files API.
- ☐ **write\_repository**  
Grants read-write access to repositories on private projects using Git-over-HTTP (not using the API).
- ☐ **ai\_features**  
Grants access to GitLab Duo related API endpoints.

Create personal access token Cancel

기본적으로 읽는 권한을 다 체크해준다.

토큰 이름과 만료기한은 자신이 원하는대로 설정하면 된다.

✓ Your new personal access token

.....

Make sure you save it - you won't be able to access it again.

이렇게 생기는 토큰을 잘 가지고 Jenkins로 돌아가서 API token 자리에 넣는다.

ID는 이 키의 고유한 이름이므로 헛갈리지 않게 잘 설정한다.

그렇게 Credentials를 추가하고 오른쪽 아래에 Test Connction을 눌렀을 때 잘되면 설정을 잘 저장한다.

그리고 아래쪽으로 내려간다.

## Global Mattermost Notifier Settings

Global Mattermost Notifier Settings

Endpoint ?  
https://meeting.ssafy.com/hooks/uweg7axj1pdr7ptstcm78im1sw

Channel ?

Icon to use ?

Build Server URL ?  
http://f10b111.p.ssafy.io:8080/

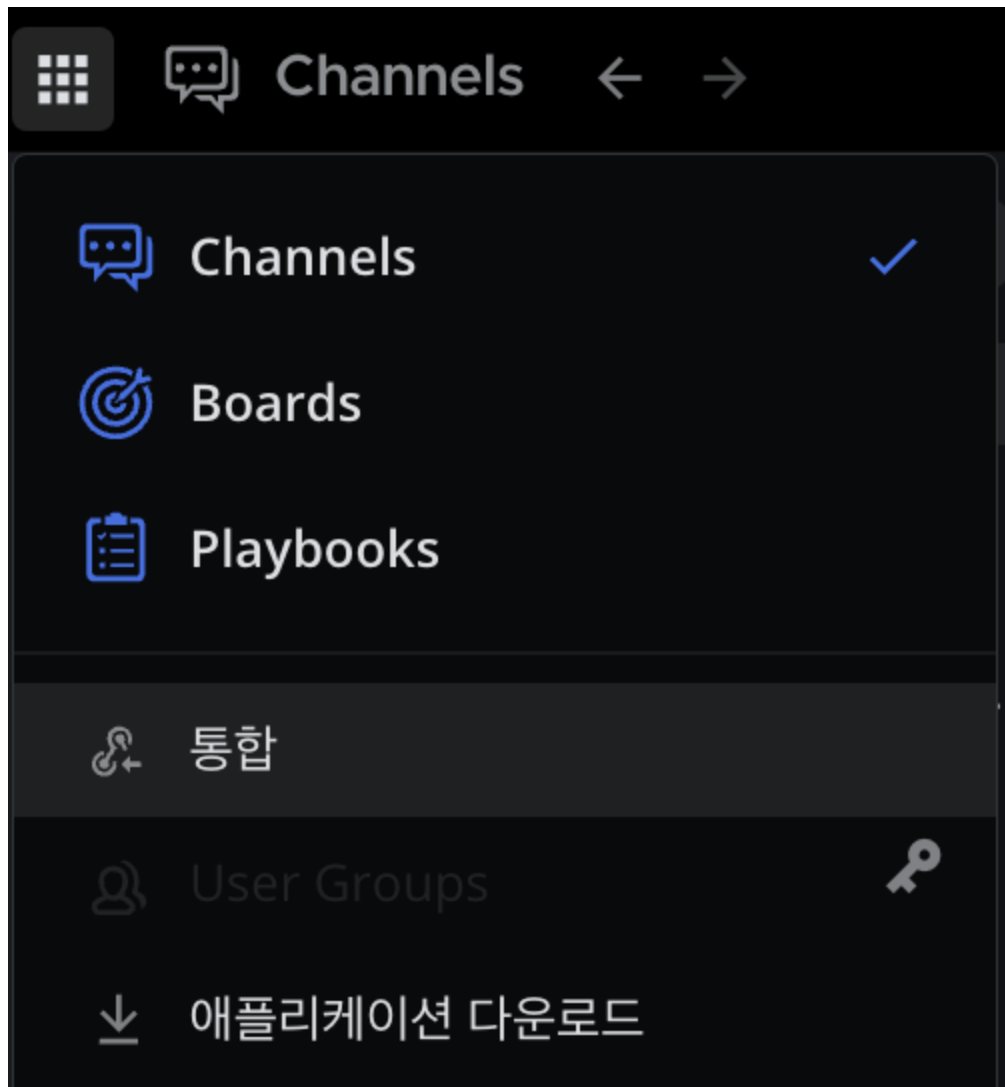
Mattermost Custom Proxy Settings ▾

Success

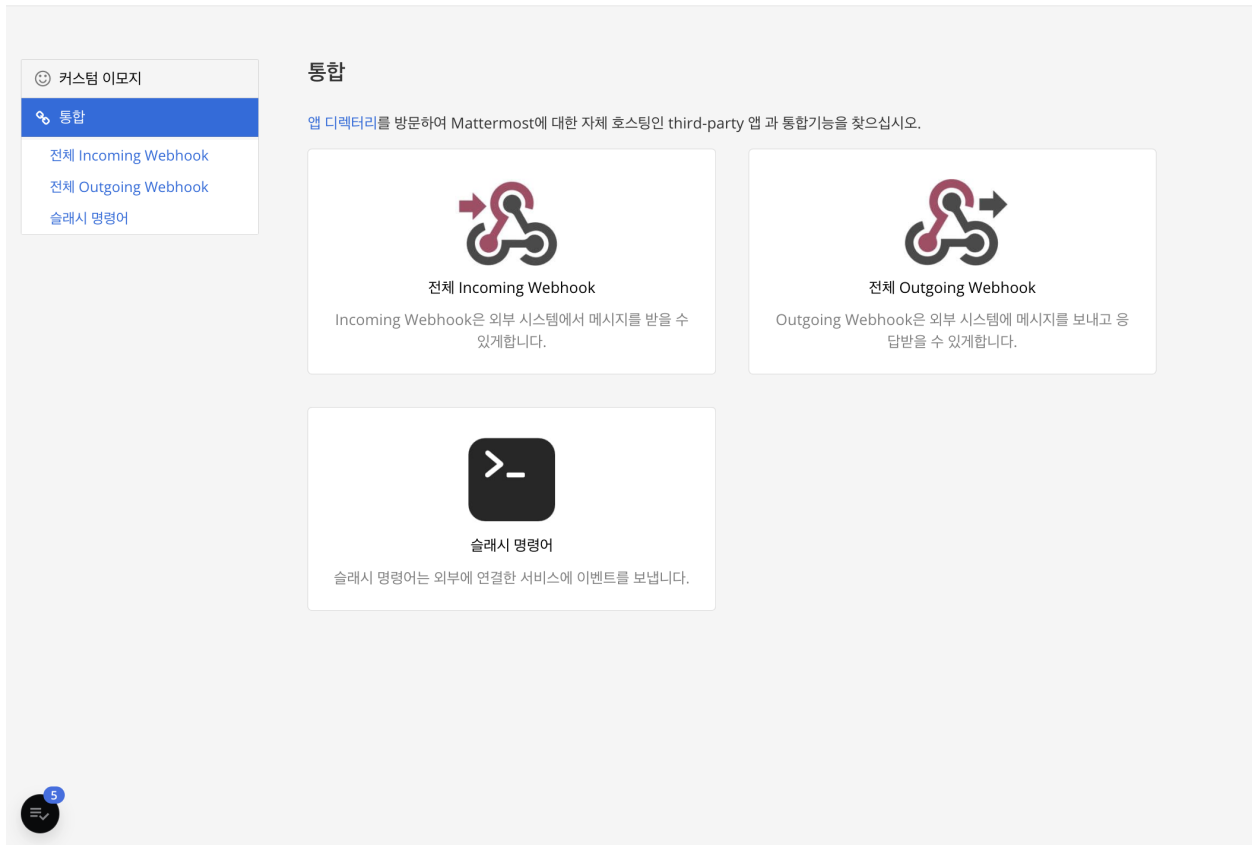
Test Connection

Jenkins pipeline에서 mattermost를 사용하기 위해 설정할 것이다.

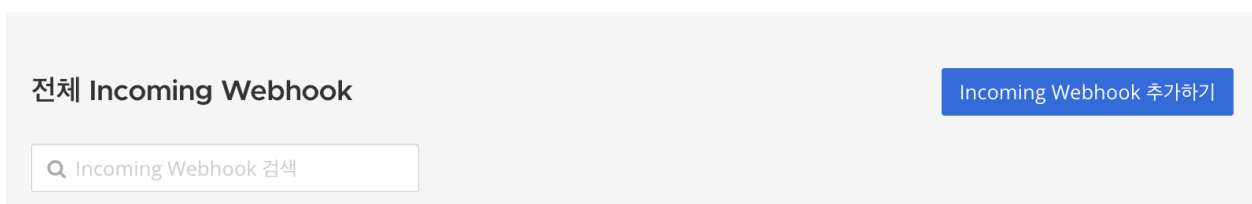
mattermost로 가서 왼쪽 위를 확인한다.



통합을 클릭한다.



전체 Incoming Webhook을 클릭한다.



오른쪽 위의 Incoming Webhook 추가하기를 클릭한다.

## Incoming Webhooks > 추가

제목

웹훅 설정 페이지에 대해 최대 64자의 제목을 지정합니다.

설명

웹훅에 대한 설명을 입력하세요.

채널

--- 채널을 선택하세요 ---

▼

웹훅 페이로드를 수신할 기본 채널(공개 혹은 비공개)입니다. 비공개 채널로 웹훅을 설정할 때에는 그 채널에 속해 있어야 합니다.

이 채널로 고정

☐

설정되면, 들어오는 웹훅은 선택된 채널에만 게시할 수 있습니다.

취소

저장

자신이 보내고 싶은 채널과 이 웹훅에 대한 제목과 설명을 추가한다.

저장하기를 누르고 나온 URL을 잘 가지고 있다.

그리고 Jenkins로 돌아와서 EndPoint에 붙여넣고 Test Connection을 클릭한다.

Success가 뜨면 저장한다.


### 4. 권한 설정

pipeline에 필요한 key들을 등록할 것이다.

GitLab Repository Clone에 필요한 키와 ec2에 ssh와 scp 명령어를 사용하기 위한 pem키를 등록할 것이다.

Dashboard > Jenkins 관리 > Credentials 를 클릭

#### Stores scoped to Jenkins

P	Store ↓	Domains
	System	(global)

여기서 (global) 클릭

오른쪽 위의 Add Credentials를 클릭한다.

#### New credentials

Kind  
Username with password ▼

Scope ?  
Global (Jenkins, nodes, items, all child items, etc) ▼

Username ?

☐ Treat username as secret ?

Password ?

ID ?

Description ?

Create

GitLab Repository Clone에 필요한 키는 Username with password이다.

Username에 아이디를

Password에 비밀번호를

Id에는 이 키의 고유한 이름을 설정해주고 Create를 누른다.

다시 Add Credentials를 클릭한다.

## 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

Passphrase

Create

ssh와 scp를 실행하기 위한 키는 SSH Username with private key이다.

여기서 ID는 이 키의 고유한 값을 입력해주고

아래 쪽에 Enter directly를 클릭하고 Add를 클릭한다.

Private Key

☒ Enter directly

Key

Enter New Secret Below

터미널로 들어가서 `cat {{pem key}}`를 해서 나온 문자열을 넣고 Create를 한다.

## 5. pipeline 구성

Dashboard > {{자신의 Item}} > Configuration을 클릭



## General

Enabled 

설명

Plain text [미리보기](#)

- ☐ Do not allow concurrent builds
- ☐ Do not allow the pipeline to resume if the controller restarts
- ☐ GitHub project

GitLab Connection

sonmandu\_gitlab

- ☐ Use alternative credential
- ☐ Pipeline speed/durability override ?
- ☐ Preserve stashes from completed builds ?
- ☐ Throttle builds ?
- ☐ 오래된 빌드 삭제 ?
- ☐ 이 빌드는 매개변수가 있습니다 ?

Build Triggers

- ☐ Build after other projects are built ?
- ☐ Build periodically ?
- ☐ Build when a change is pushed to GitLab. GitLab webhook URL: <http://10b111.p.ssafy.io:8080/project/%EB%A9%94%EB%89%B4%EC%96%BC%20%EC%9E%91%EC%84%B1> ?

그러면 이 화면이 나올 것이다.

여기서 다양한 설정들을 할 것이다.

먼저 오래된 빌드 삭제를 체크한다.

☒ 오래된 빌드 삭제 ?

Strategy

Log Rotation

빌드 이력 유지 기간(일)

공백일 경우, [보관할 최대갯수] 만큼 기록됩니다.

보관할 최대갯수

If not empty, only up to this number of build records are kept

3

고급

다양하게 커스텀을 할 수 있지만 여기서는 최근 3개의 빌드까지만 저장하고 나머지는 삭제한다.

그리고 아래 Build Triggers > Build when a change is pushed to GitLab을 체크한다.



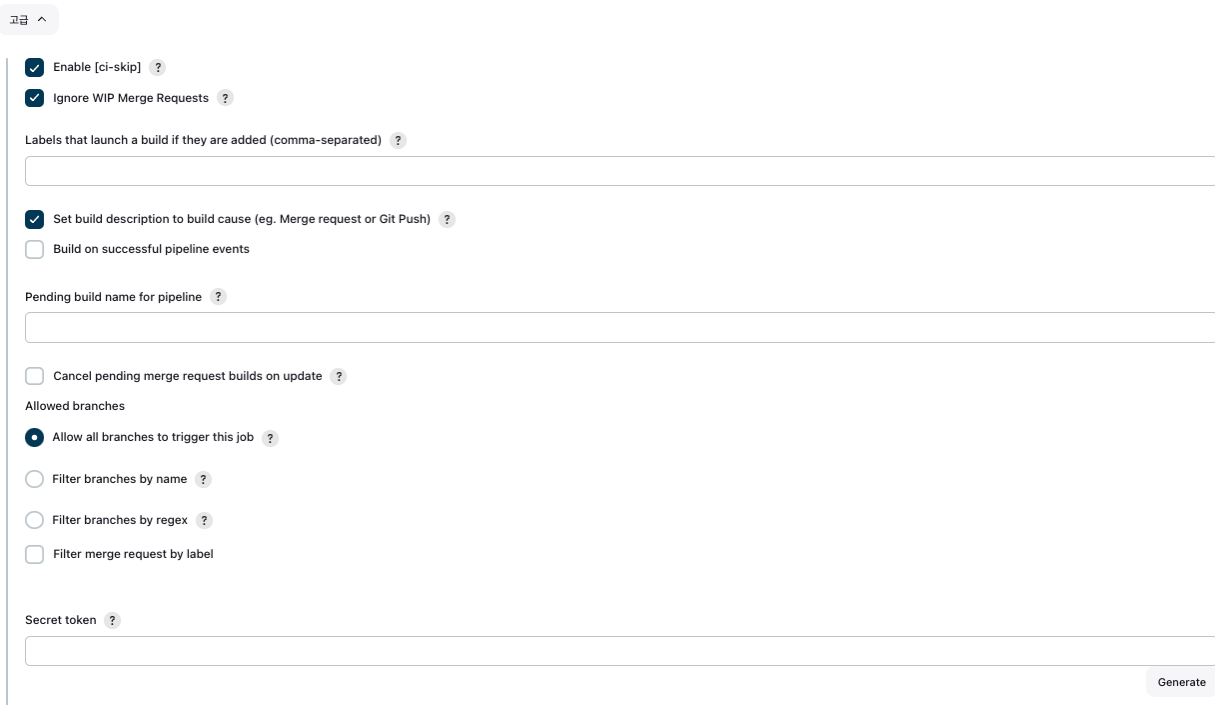
☒ Build when a change is pushed to GitLab. GitLab webhook URL: <http://f10b111.p.ssafy.io:8080/project/%EB%A9%94%EB%89%B4%EC%96%BC%20%EC%9E%91%EC%84%B1> ?

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 ?

여기서 나온 URL은 잘 가지고 있고 그 아래에 어떤 행동이 있었을 때 이 pipeline을 작동할 지 설정한다.

여기서는 push와 merge의 상황을 체크하였다.



고급 ^

- ☒ 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 ?

Generate

그리고 고급을 누르고 오른쪽 아래에 Generate를 눌러서 나온 Secret token을 잘 가지고 있다.

그리고 GitLab Project Repository > Setting > Webhooks를 들어간다.

여기서 오른쪽의 Add new webhook을 클릭한다.

**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** 3

**URL**

URL must be percent-encoded if it contains one or more special characters.

☒ Show full URL

☐ Mask portions of URL

Do not show sensitive data such as tokens in the UI.

**Secret token**

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

**Trigger**

☒ Push events

☐ Tag push events

A new tag is pushed to the repository.

☐ Comments

A comment is added to an issue or merge request.

☐ Confidential comments

A comment is added to a confidential issue.

☐ Issues events

An issue is created, updated, closed, or reopened.

☐ Confidential issues events

A confidential issue is created, updated, closed, or reopened.

☐ Merge request events

A merge request is created, updated, or merged.

☐ Job events

A job's status changes.

☐ Pipeline events

A pipeline's status changes.

☐ Wiki page events

A wiki page is created or updated.

여기서 가지고 있던 URL, secret key를 입력하고 이 Repository가 어떤 상황일 때 신호를 줄지 설정하고 저장한다.

이렇게 저장한 후 다시 Jenkins로 돌아와서 Pipeline 탭으로 돌아온다.

**Pipeline**

Definition

Pipeline script

Script ?

1

try sample Pipeline...

☒ Use Groovy Sandbox ?

[Pipeline Syntax](#)

이 부분은 위의 트리거가 작동했을 때 하는 행동을 설정하는 곳이다.

이 프로젝트에서는 3개의 pipeline을 구축하였는데 각각 이러하다.

```
// be-develop

pipeline{
    agent any

    stages{
        stage("GitLab"){
            steps{
                git credentialsId: "lee-gitlab-token",
                  url: 'https://lab.ssafy.com/s10-webmobile1-sub2',
                  branch: 'be/develop'
            }
        }
        stage("Build"){
            steps{
                dir("sonmandu-be"){
```

```

        sh 'chmod +x ./gradlew; ./gradlew bootJar'
    }
}
stage("Copy to ec2 & Docker build & run"){
    steps{
        dir("sonmandu-be/build/libs"){
            sshagent(['ec2-server']){
                sh 'scp sonmandu-be-0.0.1-SNAPSHOT.jar i
                sh 'ssh ubuntu@i10b111.p.ssafy.io "cd ~,
            }
        }
    }
}
post{
    success{
        mattermostSend (
            color: "good",
            message: " :pepe_2: [백엔드 개발용 서버] 빌드 &
        )
    }
    failure{
        mattermostSend (
            color: "danger",
            message: ":suicide_pepe: [백엔드 개발용 서버] 실패
        )
    }
}
}
}

```

```
// be-master
```

```
pipeline{
```

```

agent any

stages{
    stage('Gitlab') {
        steps {

            git credentialsId: 'lee-gitlab-token',
                url: 'https://lab.ssafy.com/s10-webmobile1-9',
                branch: 'be/master'

        }
    }
    stage("Build"){
        steps{
            dir("sonmandu-be"){
                sh 'chmod +x ./gradlew; ./gradlew bootJar'
            }
        }
    }
    stage("Copy to ec2 & Docker build, run"){
        steps{
            dir("sonmandu-be/build/libs"){
                sshagent(['ec2-server']) {
                    sh 'scp sonmandu-be-0.0.1-SNAPSHOT.jar i
                    sh 'ssh ubuntu@i10b111.p.ssafy.io "cd ~,

                }
            }
        }
    }
}

post{
    success{
        mattermostSend (
            color: "good",

```

```

        message: "[백엔드 메인 서버] 빌드 & 배포 성공!!:"
    )
}
failure{
    mattermostSend (
        color: "danger",
        message: "[백엔드 메인 서버]실패.. 젠킨스를 확인해!"
    )
}
}
}
}

```

```

// fe-master

pipeline{
    agent any

    stages{
        stage("Gitlab"){
            steps{
                git credentialsId: "lee-gitlab-token",
                url: 'https://lab.ssafy.com/s10-webmobile1-sub2',
                branch: 'fe/develop'
            }
        }
        stage("Build"){
            steps{
                dir("sonmandu-fe"){
                    nodejs("NodeJS 21.0.0"){
                        sh '''
                            npm install
                            npm run build
                            rm -rf node_modules
                            tar -cvf next.tar .next
                        '''
                    }
                }
            }
        }
    }
}

```

```

    }
  }
}
stage("Copy to ec2 & Docker build & run"){
  steps{
    dir("sonmandu-fe"){
      sshagent(['ec2-server']){
        sh '''
            scp -r next.tar package.json public
            ssh ubuntu@i10b111.p.ssafy.io "cd ~,
            '''
        }
      }
    }
  }
}
post{
  success{
    mattermostSend (
      color: "good",
      message: " :gentleman_pepe:  [프론트엔드 데
    )
  }
  failure{
    mattermostSend (
      color: "danger",
      message: " :more_sad_pepe:  [프론트엔드 메
    )
  }
}
}

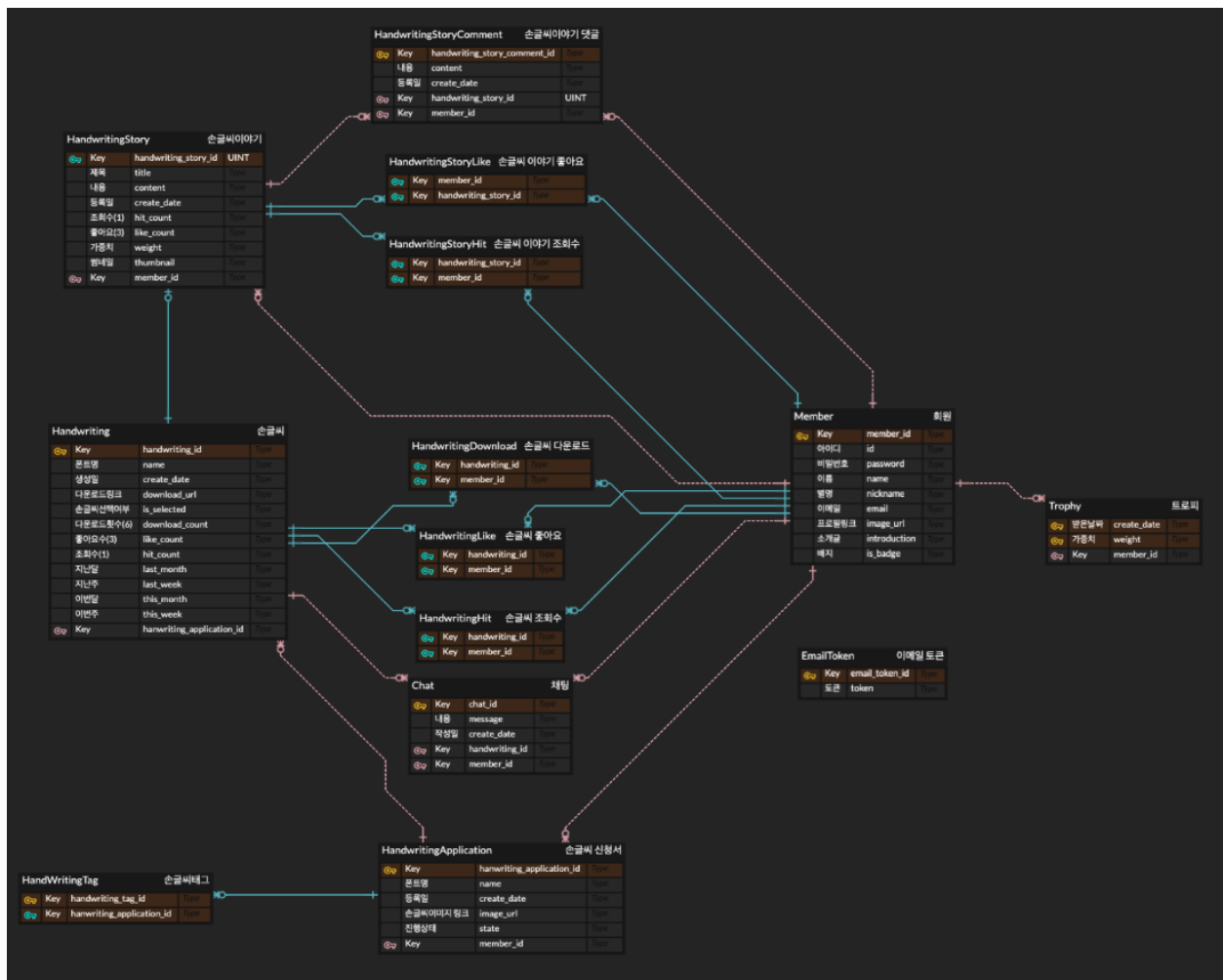
```



이와 같이 프로젝트의 목적에 맞게 script를 작성해두면 자신이 원하는대로 동작하게 할 수 있습니다.

## MySQL

### 데이터베이스 설계



### MySQL Docker image 다운 및 실행

```
// 도커 이미지로 mysql를 실행한다.
// --name으로 컨테이너 이름을 설정한다.
```

```
// -e로 환경변수를 설정한다. 여기서는 root 계정의 비밀번호를 설정하였다.  
// -d로 백그라운드에서 실행하였다.  
// -p로 포트를 설정해주었다.  
// 마지막의 mysql은 이미지의 이름이다.  
docker run --name mysql -e MYSQL_ROOT_PASSWORD=[패스워드] -d -p 3
```

## MySQL 터미널 접근 방법

: MySQL 접근할 일이 많아 alias로 등록해 두었다.

```
// -it 옵션으로 mysql 컨테이너에서 sh을 실행시킨다.  
alias mysql='sudo docker exec -it mysql sh'
```

MySQL에서 많이 사용했던 명령어들을 공유한다.

```
// 컨테이너 안에서 MySQL을 접근한다.  
mysql -u{{계정 아이디}} -p{{계정 비밀번호}}  
  
// 데이터베이스를 선택한다.  
use {{데이터베이스 이름}};  
  
// 테이블들을 확인한다.  
show tables;  
  
// 테이블의 정보를 확인한다.  
select * from {{테이블 명}};
```

평소에 root 계정을 사용하면 위험하니 User를 생성하고 특정 권한을 준다.

```
// 유저 생성 '%'는 외부접속 허용을 뜻함.  
CREATE USER {{유저 아이디}}@'%' identified by "{{유저 비밀번호}}"
```

```
// 생성한 유저에게 권한을 설정한다.
```

```
GRANT {{권한을 줄 명령어}} on {{데이터베이스명}}.{{테이블명}} {{유저}}@{
```

## Spring

---

```
// application.yml
```

```
spring:
```

```
  profiles: // 개발 환경을 설정하는 부분
```

```
    active: prod // 개발환경을 prod로 설정해서 application-prod.yml
```

```
    include: env // application-env.yml 파일도 읽어온다.
```

```
mail: // 이메일을 보내기 위한 설정이다.
```

```
  host: smtp.gmail.com // 구글 이메일을 사용한다.
```

```
  port: 587 // 구글 이메일을 사용하기 위한 포트이다.
```

```
  username: sonmandu0103@gmail.com // 보낼 때 사용하는 이메일이다.
```

```
  password: uqoqxuadzaottenr // 비밀번호는 이메일의 비밀번호가 아니라
```

```
  properties: // 이메일을 보내는 프로토콜을 열어준다.
```

```
    mail:
```

```
      smtp:
```

```
        auth: true
```

```
        starttls:
```

```
          enable: true
```

```
jpa:
```

```
  hibernate:
```

```
    ddl-auto: validate // 배포환경에서는 초기화 전략을 validate를 사
```

```
servlet:
```

```

multipart:
  enabled: true // 멀티 파트 업로드 지원 여부를 설정한다.
  file-size-threshold: 0B // 파일을 디스크에 저장하지 않고 메모리에
  location: ~/sonmando-store/temp // 업로드된 파일이 임시로 저장
  max-file-size: 15MB // 한개 파일의 최대 사이즈를 설정한다.
  max-request-size: 100MB // 한개 요청의 최대 사이즈를 설정한다.

cloud:
  aws:
    s3: // 이미지와 폰트를 저장하는데 사용할 AWS s3에 관련된 설정들이다.
    bucket: sonmando // 버킷이름을 설정한다.
    credentials: // s3에 접근하기 위한 비밀키들이 있다. applicatio
    access-key: ${s3.access-key}
    secret-key: ${s3.secret-key}
    region: // 기본적인 환경설정을 해준다.
    static: ap-northeast-2
    auto: false
  stack:
    auto: false

server: // 서버와 연결 시간을 설정한다.
tomcat:
  connection-timeout: 1800000

```

```
// application-prod.yml
```

```

spring:
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver // 데이터베이스로는
    // 외부에서도 접근할 수 있게 설정해준다.
    url: jdbc:mysql://i10B111.p.ssafy.io:3306/sonmandu?useS
    // 데이터베이스에 접근할 수 있는 계정 정보를 등록한다.
    username: ${mysql.username}
    password: ${mysql.password}

```

```
// 개발환경에서 frontend와 backend의 prefix를 정의해준다.
```

```
client:
```

```
  url: https://i10b111.p.ssafy.io
```

```
server:
```

```
  url: https://i10b111.p.ssafy.io/api
```

```
// 로그를 볼 수준을 정해준다.
```

```
logging:
```

```
  level:
```

```
    root: info
```

```
// application-env.yml
```

```
// 개발환경에서 h2 사용시 계정 정보를 등록한다.
```

```
h2:
```

```
  username: user
```

```
  password: 1234
```

```
// MySQL에 접근할 수 있는 계정 정보를 등록한다.
```

```
mysql:
```

```
  username: sonmandu
```

```
  password: son0103
```

```
// s3에 접근할 수 있는 비밀키 정보를 등록한다.
```

```
s3:
```

```
  access-key: AKIA53BIDTSEUYN2CV53
```

```
  secret-key: mtbAz/FSLfPbF6XcyMvdU6j50ed8/NqQi4RaCyX1
```

```
// jwt 비밀키 정보를 등록한다.
```

```
jwt:
```

```
  secret: dlksjfidsojfoidsjfoijsdoijsfoidsfoidjsoidfjgldfkgmls
```

```
// 이메일 발송에 사용될 이메일 정보를 등록한다.
```

sonmandu:

email: sonmandu0103@gmail.com