



# 포팅 매뉴얼

## 개발 환경

### Front-End

- Node.js - v14.15.1
- Next.js - 12.1.5
- npm - v6.14.8

### Back-End

- Java - openjdk v1.8.0\_301
- Spring Boot - 2.6.6
- JPA
- Mysql - 8.0.27
- Swagger - v3

### Infra

- AWS EC2
- Jenkins - jenkins:lts 2.332.2
- Nginx - 1.18.0 (ubuntu)
- Cerbot - 1.27.0
- Docker - 20.10.14

### Port

| App     | EC2 Port | 컨테이너 Port |
|---------|----------|-----------|
| Spring  | 8081     | 8081      |
| Next.js | 3000     | 3000      |
| MySQL   | 3306     | 3306      |
| Jenkins | 8080     | 8080      |

### DB 접속 정보

- USER : climb\_dev
- PW : weclimb401

## 1. 서버 시간 설정

### 현재 시간 설정 확인

date

## 목록에 서울 시간 있는지 확인

```
timedatectl list-timezones | grep Seoul
```

## 서울 시간으로 변경

```
sudo timedatectl set-timezone Asia/Seoul
```

## Docker 실행 시 서울 시간으로 설정 옵션

- 모든 컨테이너 run 시 해당 옵션 붙여준다.

```
-e TZ=Asia/Seoul
```

## 2. 도커 설치

### 설치 가능한 패키지 리스트를 최신화

```
sudo apt update
```

### HTTP 패키지 설치

```
~$ sudo apt install apt-transport-https  
~$ sudo apt install ca-certificates  
~$ sudo apt install curl
```

- ca-certificates : 인증서 적용을 위한 패키지
- curl : 특정한 웹사이트에서 데이터를 다운받을때 사용

### Docker 공식 GPG 키 등록

```
~$ sudo apt install software-properties-common  
~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

- curl 명령어로 Docker의 GPG key를 불러와서 apt에 추가한다. 이 과정을 통해 apt로 docker 패키지를 설치할 수 있게 된다.

### 저장소 설정

```
~$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable"
```

### 도커 설치

```
~$ apt-cache policy docker-ce  
~$ sudo apt install docker-ce
```

### 설치된 도커 버전 확인

```
sudo docker version
```

### 3. MySQL 설정

#### MySQL 컨테이너 생성

```
docker run -d -p 3306:3306 --name mysql -e MYSQL_ROOT_PASSWORD=ssafytest1234 mysql:8.0.27
```

- mysql의 경우 docker 에 이미지가 존재하기 때문에 가능

#### MySQL 컨테이너 접속

```
~$docker exec -it mysql /bin/bash  
root@a1286eac2ea0:/# mysql -u root -p
```

```
ubuntu@ip-172-26-7-250:~/S06P12A506/backend/target$ docker exec -it mysql /bin/bash  
root@a1286eac2ea0:/# mysql -u root -p  
Enter password:  
Welcome to the MySQL monitor. Commands end with ; or \g.  
Your MySQL connection id is 2095  
Server version: 8.0.28 MySQL Community Server - GPL  
  
Copyright (c) 2000, 2022, Oracle and/or its affiliates.  
  
Oracle is a registered trademark of Oracle Corporation and/or its  
affiliates. Other names may be trademarks of their respective  
owners.  
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
mysql> █
```

- mysql 접속 성공!

#### 사용자 생성

```
create user 'testuser'@'%' identified by 'password';
```

- 사용자 생성

```
grant all privileges on . to 'testuser'@'%';
```

- 외부에서 접속 할 수 있도록 권한 부여

```
flush privileges;
```

- 변경된 권한을 적용

#### MySQL :: Download MySQL Installer (Archived Versions)

Please note that these are old versions. New releases will have recent bug fixes and features! To download the latest release of MySQL Installer, please visit MySQL Downloads. MySQL open source software is provided under the GPL License.

 <https://downloads.mysql.com/archives/installer/>

- 접속 하여 window에 mysql 다운로드

#### workbench connection setting

Setup New Connection

Connection Name:  Type a name for the connection

Connection Method:  Method to use to connect to the RDBMS

Parameters SSL Advanced

Hostname:  Port:  Name or IP address of the server host - and TCP/IP port.

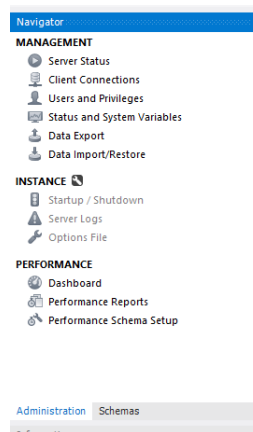
Username:  Name of the user to connect with.

Password:   The user's password. Will be requested later if it's not set.

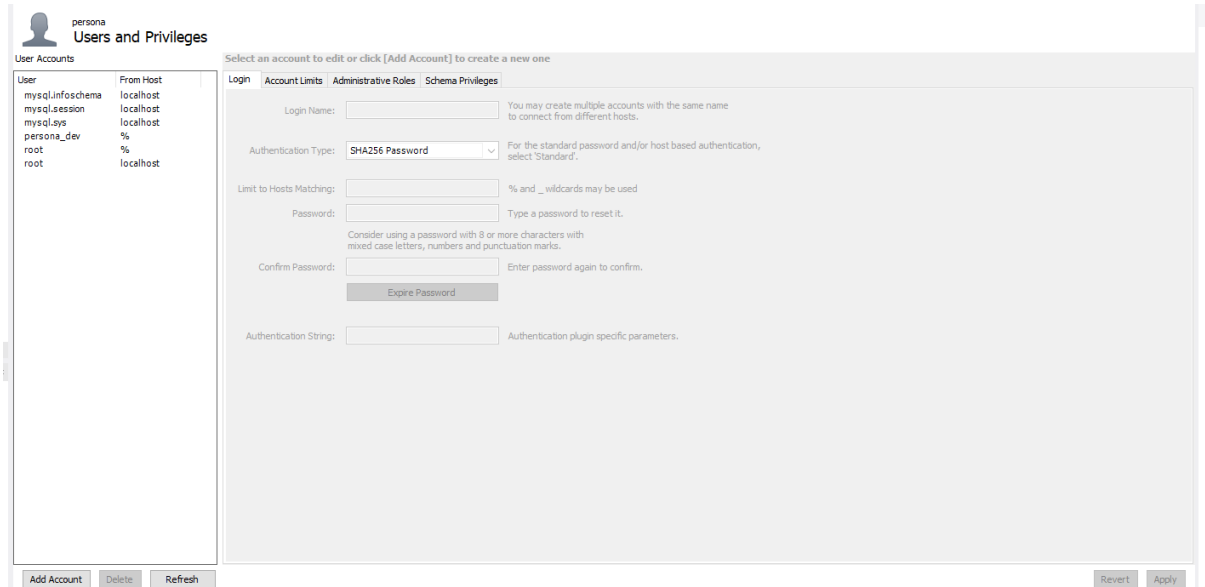
Default Schema:  The schema to use as default schema. Leave blank to select it later.

- Hostname: EC2(서버)의 public IP 주소/도메인
- Port: 3306
- Username: climb\_dev
- Password: weclimb401

## 권한 설정



- 접속하여 좌측 Administration 클릭 후 Users and Privileges 클릭



- 적절한 권한을 지닌 계정을 생성하여 불필요한 root 계정 사용을 막을 수 있습니다.

## 4. Jenkins 설치 및 설정

### Jenkins 이미지 다운로드

```
docker pull jenkins/jenkins:lts
```

### Jenkins 컨테이너 생성 및 실행

```
docker run --name jenkins -e TZ=Asia/Seoul -d -p 8080:8080 -v /home/ubuntu/jenkins:/var/jenkins_home -u root jenkins/jenkins:lts
```

- **-d** : 백그라운드 실행
- **-p** : 컨테이너와 호스트 PC 간 연결을 위해 내부 포트와 외부 포트 매핑, **로컬포트 : 컨테이너 포트** 의미
  - 8080:8080 옵션은 로컬의 8080 요청을 8080 컨테이너로 연결 (젠킨스 웹 서버 포트)
- **-v** : 이미지의 /var/jenkins\_home 디렉터리를 호스트 PC 내에 마운트, Jenkins 설치 시 ssh 키값 생성, 저장소 참조 등을 용이하게 하기 위해 설정
- **-e TZ=Asia/Seoul** : 우리나라 시간으로 설정

### 컨테이너 설정

```
sudo ufw allow 8080
```

- 8080포트 방화벽 허용 설정

### Jenkins 접속

```
http://k6a401.p.ssafy.io:8080/
```

### 초기 비밀번호 확인

## 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

Continue

- 초기 비밀번호 입력 요구가 뜨는데 해당 부분에서 확인 가능

```
docker logs jenkins
```

- 혹은 로그를 통해 확인

### 초기 비밀번호 입력 후 continue

## Customize Jenkins

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

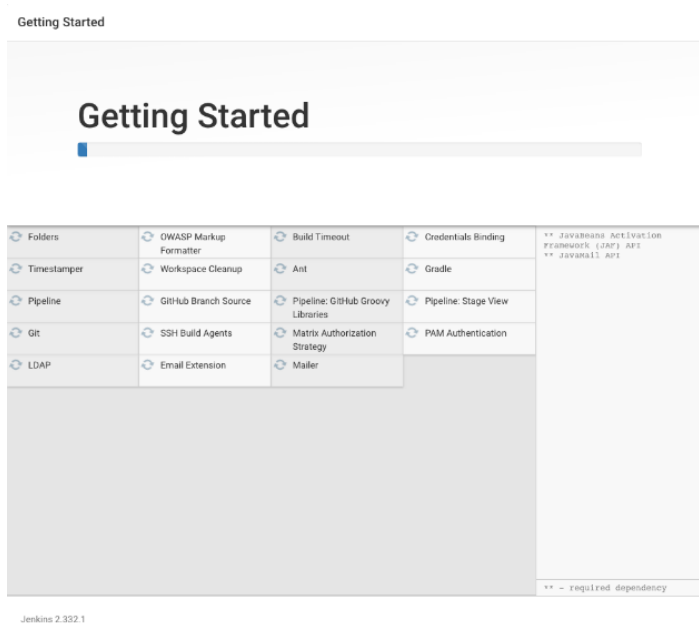
### Install suggested plugins

Install plugins the Jenkins community finds most useful.

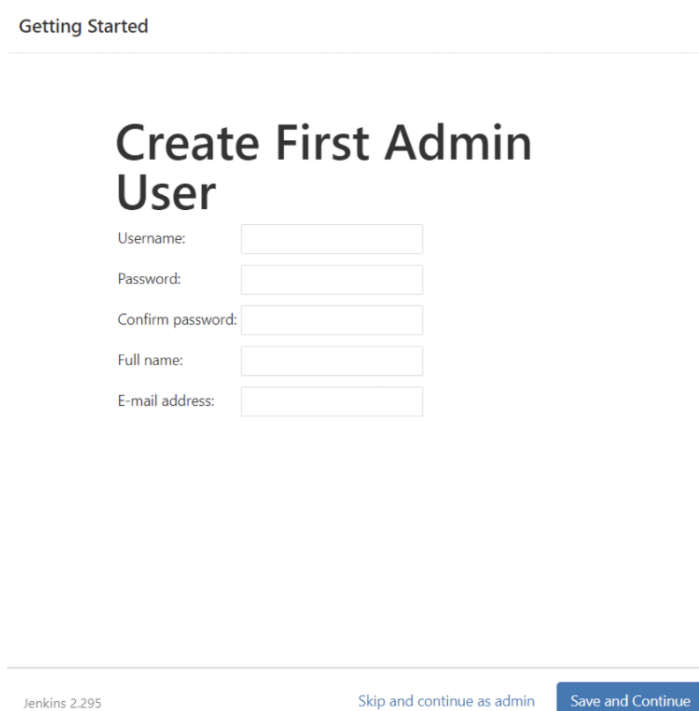
### Select plugins to install

Select and install plugins most suitable for your needs.

### install suggested plugins 선택



## 계정 생성



## URL 설정

## Instance Configuration

Jenkins URL:

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the `DU:LD_URL` environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Jenkins 2.295

Not now

Save and Finish

- EC2의 퍼블릭 IP 입력

## 5. Jenkins - GitLab 연결

### GitLab plugin 설치

Dashboard
> Plugin Manager

대시보드로 돌아가기
Jenkins 관리
Update Center

### Plugin Manager

업데이트된 플러그인 목록
설치 가능
설치된 플러그인 목록
고급

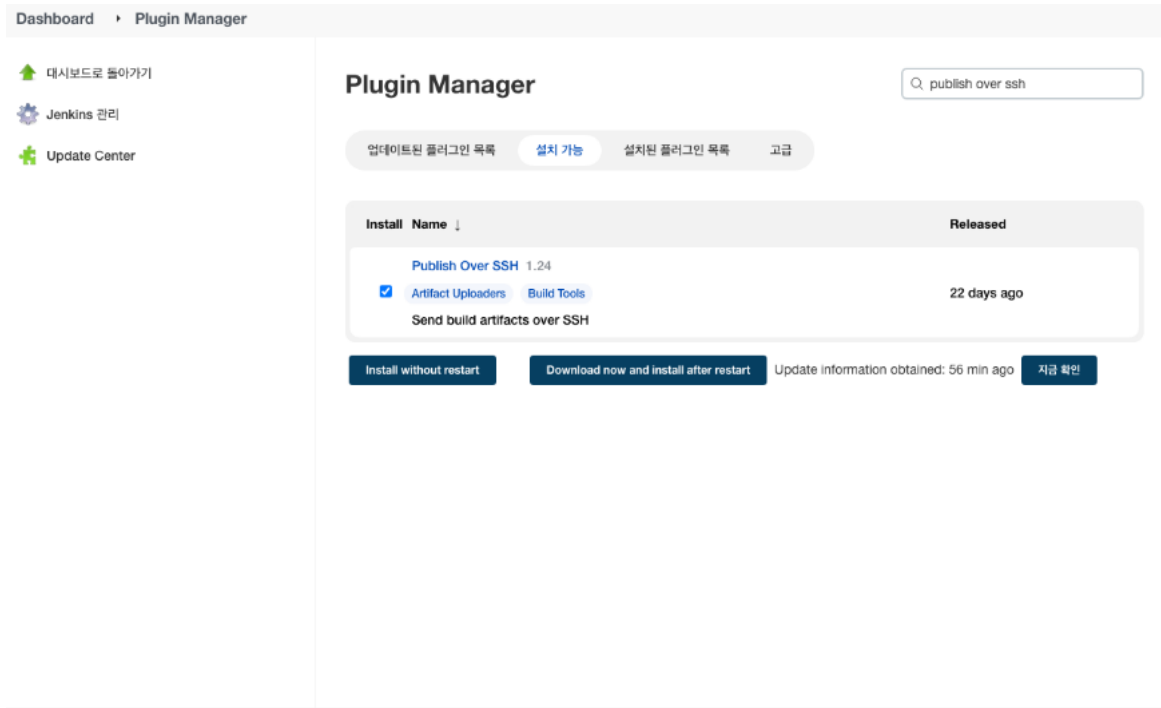
| Install                             | Name  | Released         |
|-------------------------------------|---|------------------|
| <input checked="" type="checkbox"/> | GitLab 1.5.29<br>Build Triggers<br>This plugin allows GitLab to trigger Jenkins builds and display their results in the GitLab UI.<br>This plugin is up for adoption! We are looking for new maintainers. Visit our Adopt a Plugin initiative for more information.   | 5 days 16 hr ago |
| <input type="checkbox"/>            | Generic Webhook Trigger 1.83<br>notification github webhook Build Parameters gitlab Build Triggers<br>bitbucket bitbucket-server jira<br>Can receive any HTTP request, extract any values from JSON or XML and trigger a job with those values available as variables. Works with GitHub, GitLab, Bitbucket, Jira and many more.                                | 1 mo 21 days ago |
| <input type="checkbox"/>            | GitLab Authentication 1.13<br>Authentication and User Management<br>This is the an authentication plugin using gitlab OAuth.<br>Warning: This plugin version may not be safe to use. Please review the following security notices:<br><ul style="list-style-type: none"> <li>Open redirect vulnerability</li> <li>Client Secret stored in plain text</li> </ul> | 2 mo 5 days ago  |

Install without restart
Download now and install after restart
Update information obtained: 35 min ago
지금 확인

- Jenkins 관리 - Plugin Manager - 설치 가능 탭
- GitLab 플러그인 선택 후 install

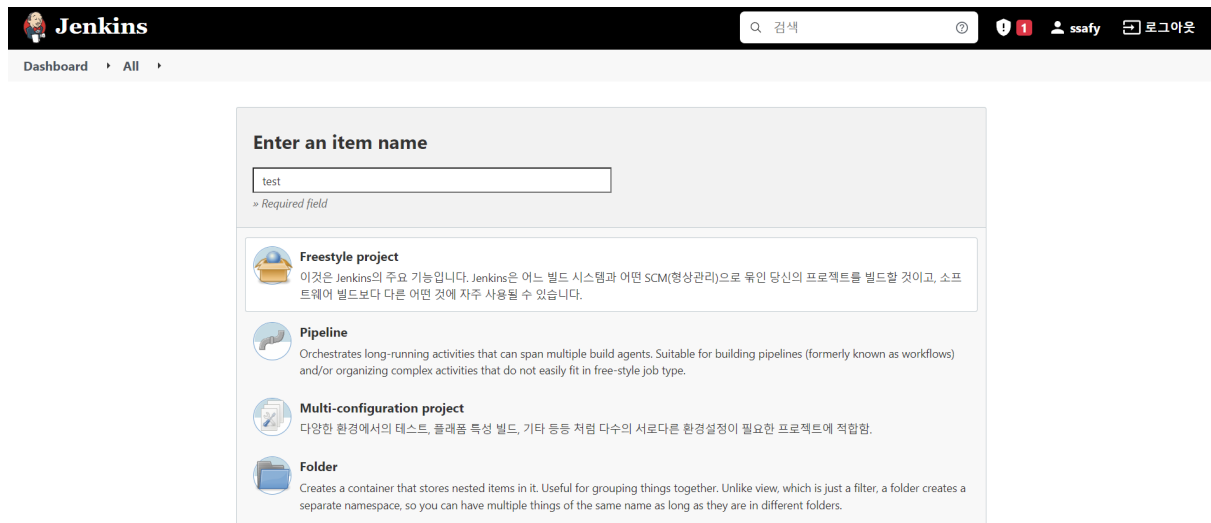
### publish over SSH 플러그인 설치





## Jenkins item 생성

### freestyle project 선택



## Jenkins SSH Key로 GitLab Webhook 설정 [repository]

- public repository라면 url만 입력해도 바로 연동이 되지만 private repository인 경우에는 ssh key를 등록하여야 정상적으로 git 반영이 가능하다.

### ssh 키 생성

General 소스 코드 관리 빌드 유발 빌드 환경 Build 빌드 후 조치

### 빌드 유발

☐ 빌드를 원격으로 유발 (예: 스크립트 사용) ?

☐ Build after other projects are built ?

☐ Build periodically ?

☒ Build when a change is pushed to GitLab. GitLab webhook URL: <http://k6a401.p.ssafy.io:8080/project/test> ?

**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

☒ Approved Merge Requests (EE-only)

☒ Comments

**Comment (regex) for triggering a build ?**

Jenkins please retry a build

고급...

- 구성 → 빌드 유발 → Build when a change is pushed to GitLab. 선택

General 소스 코드 관리 빌드 유발 빌드 환경 Build 빌드 후 조치

☒ Enable [ci-skip]

☒ Ignore WIP Merge Requests

**Labels that forces builds 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

Clear

- Advanced (고급) 클릭 - **Secret token** 메뉴의 Generate 클릭
- 토큰 복사 후 저장 클릭

**General**   소스 코드 관리   빌드 유발   빌드 환경   Build   빌드 후 조치

☐ Use alternative credential  
☐ This build requires lockable resources  
☐ 이 빌드는 매개변수가 있습니다 ?  
☐ Throttle builds ?  
☐ 빌드 안함 ?  
☐ 필요한 경우 concurrent 빌드 실행 ?

**소스 코드 관리**

☒ None  
☐ Git ?

**빌드 유발**

☐ 빌드를 원격으로 유발 (예: 스크립트 사용) ?  
☐ Build after other projects are built ?  
☐ Build periodically ?  
☒ Build when a change is pushed to GitLab. GitLab webhook URL: `http://k6a401.p.ssafy.io:8080/project/test` ?

**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**

Build when a change is pushed to GitLab. 옆에 있는 GitLab webhook URL: 뒤의 **url**을 복사  
`http://k6a401.p.ssafy.io:8080/project/item`

## GitLab repository에 Jenkins SSH Key 등록

**Webhook**

URL: `http://k6a401.p.ssafy.io:8080/project/whyweclimb`

Secret token:

Trigger:

☒ Push events  
 develop  
 URL is triggered by a push to the repository

☐ Tag push events  
 URL is triggered when a new tag is pushed to the repository

☐ Comments  
 URL is triggered when someone adds a comment

☐ Confidential comments  
 URL is triggered when someone adds a comment on a confidential issue

☐ Issues events  
 URL is triggered when an issue is created, updated, closed, or reopened

☐ Confidential issues events  
 URL is triggered when a confidential issue is created, updated, closed, or reopened

☐ Merge request events  
 URL is triggered when a merge request is created, updated, or merged

☐ Job events  
 URL is triggered when the job status changes

☐ Pipeline events  
 URL is triggered when the pipeline status changes

- jenkins와 연결할 gitlab의 repository에서 Settings - Webhooks
- URL : 위에서 복사해두었던 url
- Secret token : 위에서 복사해두었던 토큰값
- Trigger - Push events : push event를 발생시킬 branch (비워두면 전체 branch)

## 6. 프로젝트 자동 clone 설정

### Access Token 등록 [repository]

## GitLab Access Token 생성

- GitLab의 연결할 repository - Settings - Access Tokens 생성

- access token이 새로 생성된 것을 확인하고, 복사

## Access Token으로 GitLab에 연결

Jenkins 관리 - 시스템 설정 - Gitlab - Credentials - Add 클릭

Dashboard > 환경설정

Usage Statistics

☒ Help make Jenkins better by sending anonymous usage statistics and crash reports to the Jenkins project. ?

**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 ?  
gitlab

Description ?

Add Cancel

저장 Apply

Test Connection

- Kind : GitLab API token 선택
- API token : 위에서 생성한 GitLab Access Token
- ID : Credentials를 구분하는 고유값 (아무거나 상관 X)

Dashboard > 환경설정

Add

List of default health metrics for Folders

Gitlab

☒ Enable authentication for '/project' end-point

GitLab connections

Connection name  
lab-ssafy-506P31A401  
A name for the connection

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

Credentials  
GitLab API token Add  
API Token for accessing Gitlab

추가

고급...

Test Connection

삭제

Add 클릭 후

- Connection name : Gitlab repository를 구분할 수 있는 고유값 (아무거나 상관 X)
- global host URL : 도메인 주소까지만 (ex : https://lab.ssafy.com)
- Credentials : 위에서 Access Token을 이용해 만든 credential 선택

## Credential 추가 [개인 계정]

Jenkins 관리 - Manage Credentials - (global) 클릭 - Add Credentials 클릭

The screenshot shows the Jenkins 'Add Credentials' form for a global credential. The breadcrumb trail at the top is 'Dashboard > Credentials > System > Global credentials (unrestricted)'. On the left sidebar, there are links for 'Back to credential domains' and 'Add Credentials'. The main form has the following fields: 'Kind' (set to 'Username with password'), 'Scope' (set to 'Global (Jenkins, nodes, items, all child items, etc)'), 'Username' (set to 'kwwoo516@naver.com'), a checkbox for 'Treat username as secret' (unchecked), 'Password' (masked with asterisks), 'ID' (empty), and 'Description' (empty). An 'OK' button is at the bottom.

- Username : GitLab 로그인 시 사용하는 이메일 주소
- Password : GitLab 비밀번호

## Jenkins의 Item에 GitLab repository 연결

Jenkins의 item - 구성 - 소스 코드 관리 - Git 선택

### 빌드할 코드 등록

item Config (구성) - Source Code Management (소스 코드 관리) - Git 선택

The screenshot shows the 'Source Code Management' configuration in Jenkins. The 'General' tab is selected. Under 'Repositories', the 'Git' option is chosen. The 'Repository URL' is set to 'https://lab.ssafy.com/s06-final/S06P31A401.git'. The 'Credentials' dropdown is set to 'kwwoo516@naver.com/\*\*\*\*\*', with an 'Add' button next to it. There is an 'Add Repository' button at the bottom left and a '고급...' (Advanced) button at the bottom right.

- Repository URL : GitLab repository clone 주소 입력
- Credentials : 위에서 만든 GitLab 이메일 주소와 비밀번호로 만든 Credential 선택
- Branches to build : 빌드할 브랜치 입력

## GitLab Connection 연결

item Config (구성) - General - GitLab Connection

위에서 GitLab Access Token으로 Jenkins 환경설정에 등록해둔 GitLab Connection으로 선택

## 7. AWS 에서 명령어 실행할 수 있도록 설정

### EC2 SSH Key 를 Jenkins Docker Container 에 전송

개인 PC → EC2 홈 으로 pem key 전송

```
scp -i .ssh/K6A401T.pem .ssh/K6A401T.pem ubuntu@k6a401.p.ssafy.io:/home/ubuntu
```

scp <upload\_path> <username>@<IP>:<download\_path>

- **-i** : 이 명령에 대한 권한을 부여하기 위해 사용
- **upload\_path** : 기존 파일이 존재하는 경로를 입력
- **username** : 사용자 계정 아이디를 입력
- **IP** : 복사하려고 하는 목적지 IP주소 또는 도메인 이름을 지정
- **download\_path** : 파일이 생성될 원하는 목적지의 파일 저장 경로를 지정

EC2에 추가된 pem key를 확인할 수 있다.

### EC2 → docker container 로 pem key 전송

```
docker cp K6A401T.pem jenkins:/var/jenkins_home/
```

docker cp 복사할파일 컨테이너명(ID):위치

### Jenkins 관리 - 시스템 설정 - Publish over SSH

**Publish over SSH**

**Jenkins SSH Key** ?

**Passphrase** ?

Concealed Change Password

**Path to key** ?

K6A401T.pem

**Key** ?

☐ Disable exec ?

**SSH Servers**

SSH Server

**Name** ?

k6a401

**Hostname** ?

k6a401.p.ssafy.io

**Username** ?

ubuntu

**Remote Directory** ?

/home/ubuntu

- **Path to key** : 위에서 pem key를 복사해둔 위치가 jenkins 홈 디렉터리이기 때문에 key 이름만 적어줘도 된다.
- SSH Servers add 클릭해서 서버 설정 추가
- **Name** : 접속할 서버를 식별할 수 있는 이름
- **Hostname** : 접속할 서버 주소나 도메인명
- **Username** : 접속할 서버에서 사용할 사용자명
- **Remote Directory** : 접속할 서버에서 사용할 폴더

**Jenkins**

Dashboard ▾ ▶ 환경설정

새로운 Item
사람
빌드 기록
프로젝트 연관 관계
파일 핑거프린트 확인
Jenkins 관리

**홈 디렉터리** ?
  
/var/jenkins\_home

**시스템 메시지** ?
  
  
[Plain text] [미리보기](#)

- jenkins 홈 디렉터리는 현재 페이지 제일 윗 부분에서 확인할 수 있다.

## 8. Spring Boot Dockerfile 작성

### Dockerfile



## 위치

- S06P31A401
  - backend
    - Dockerfile

## 파일 내용

```
FROM openjdk:8-jdk-alpine
ARG JAR_FILE=build/libs/backend-0.0.1-SNAPSHOT.jar
COPY ${JAR_FILE} app.jar
EXPOSE 8081
ENTRYPOINT ["java", "-jar", "/app.jar"]
```

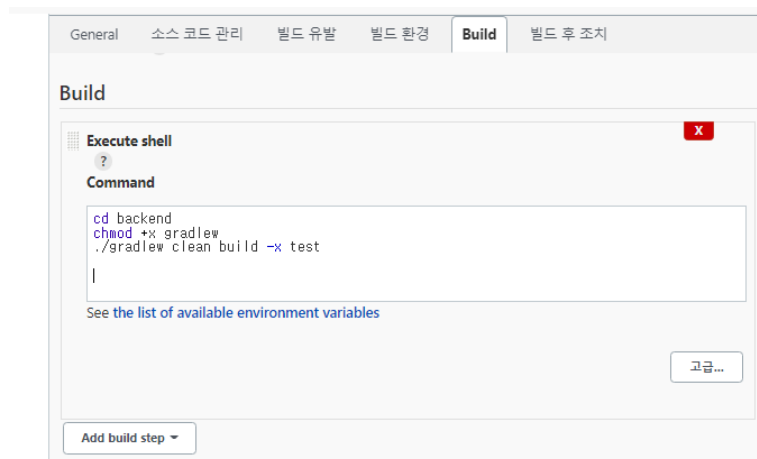
- **openjdk java 8버전**의 환경의 이미지를 받는다.
- 빌드 될 때 활용된다. **JAR\_FILE 변수에 target/\*.jar**를 담음.
- **app.jar**에 JAR\_FILE을 복사한다.
- 8081 포트를 사용한다.
- 기존 jar 파일을 단순 실행하듯이 **"java -jar /app.jar"**(jar 파일을 실행하는 명령어)를 실행하면서 스프링부트가 올라간다.

## 8. 백엔드 빌드 및 배포

### 빌드

#### item 구성 - Build - Add build step - Execute shell 선택

```
cd backend
chmod +x gradlew
./gradlew clean build -x test
```



- **+x gradlew** : gradlew 접근 권한 부여
- **-x test** : 테스트 코드 실행하지 않음

**Project whyweclimb**

작업 공간  
최근 변경사항

**고정링크**

- Last build, (#127), 8 hr 54 min 전
- Last stable build, (#127), 8 hr 54 min 전
- Last successful build, (#127), 8 hr 54 min 전
- Last failed build, (#111), 1 day 19 hr 전
- Last unstable build, (#119), 22 hr 전
- Last unsuccessful build, (#119), 22 hr 전
- Last completed build, (#127), 8 hr 54 min 전

**Build History** 추이 ^

Filter builds...

| Build Number | Timestamp            | Started by                    |
|--------------|----------------------|-------------------------------|
| #127         | 2022. 5. 13. 오전 1:09 | Started by GitLab push by 우윤석 |
| #126         | 2022. 5. 12. 오후 5:25 | Started by GitLab push by 우윤석 |
| #125         | 2022. 5. 12. 오후 4:41 | Started by GitLab push by 우윤석 |
| #124         | 2022. 5. 12. 오후 4:41 | Started by GitLab push by 우윤석 |
| #123         | 2022. 5. 12. 오후 4:03 | Started by GitLab push by 우윤석 |

- push했을 때 정상 빌드 확인

## 배포

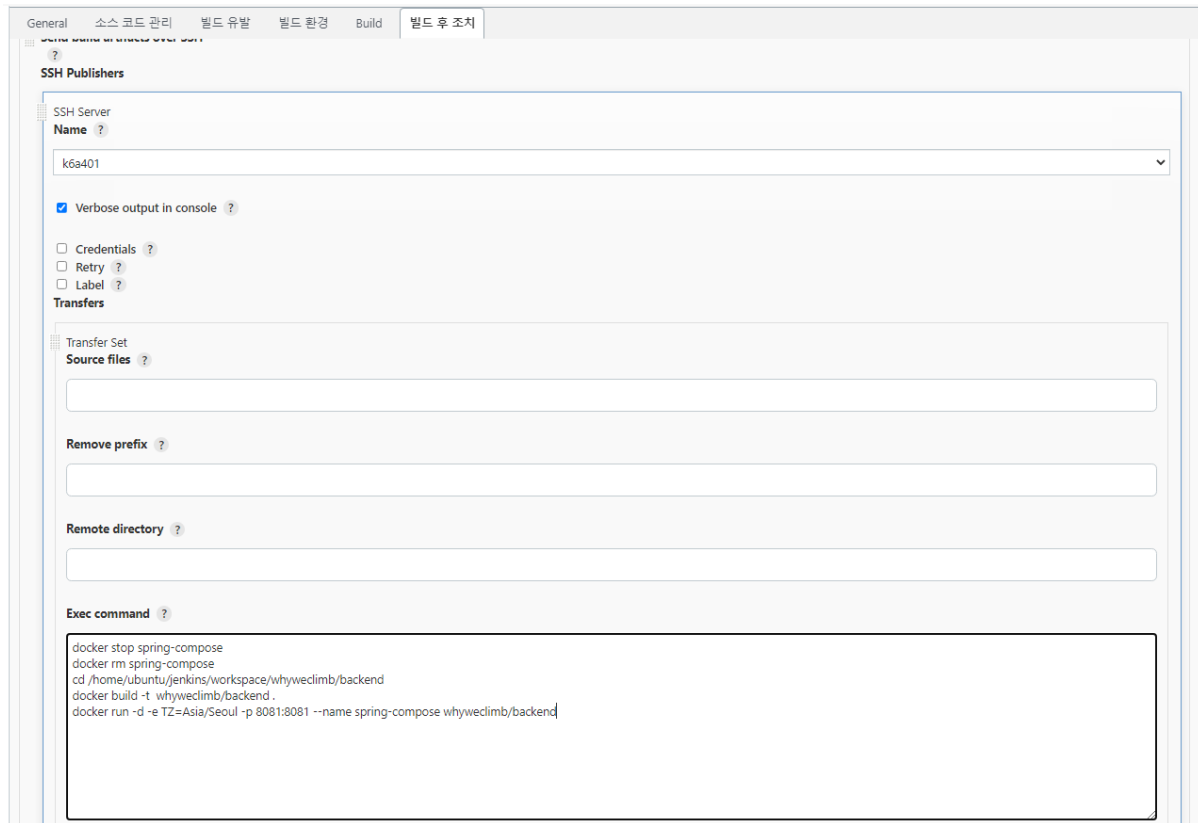
### 빌드 후 조치 - 빌드 후 조치 추가 - Send build artifacts over SSH 선택

- **SSH Server Name** : 위에서 pem키로 만들었던 server 선택
- Exec command에 아래 내용 추가
  - jenkins 처음 실행시킬 때 (도커 이미지 없을 때)

```
cd /home/ubuntu/jenkins/workspace/whyweclimb/backend
docker build -t whyweclimb/backend .
docker run -d -e TZ=Asia/Seoul -p 8081:8081 --name spring-compose whyweclimb/backend
```

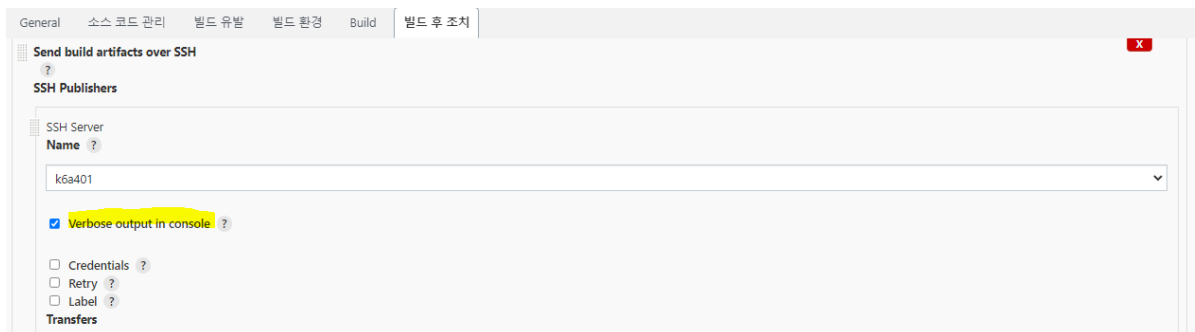
- 처음 실행이 아닐 때 (도커 이미지 만들어져있을 때)

```
docker stop spring-compose
docker rm spring-compose
cd /home/ubuntu/jenkins/workspace/whyweclimb/backend
docker build -t whyweclimb/backend .
docker run -d -e TZ=Asia/Seoul -p 8081:8081 --name spring-compose whyweclimb/backend
docker rmi $(sudo docker images -f "dangling=true" -q)
```



## SSH로 실행한 명령어 로그 출력 설정

빌드 후 조치 - SSH Publishers - 고급 - Verbose output in console



## 정상 배포 확인

```
[test] $ /bin/sh -xe /tmp/jenkins6760130193800076393.sh
+ cd backend
+ ./gradlew clean build -x test
Starting a Gradle Daemon (subsequent builds will be faster)
> Task :clean
> Task :check
> Task :compileJava
> Task :processResources
> Task :classes
> Task :bootJarMainClassName
> Task :bootJar
> Task :jar
> Task :assemble
> Task :build

BUILD SUCCESSFUL in 9s
6 actionable tasks: 6 executed
SSH: Connecting from host [5ealce974594]
SSH: Connecting with configuration [j6a301] ...
SSH: EXEC: completed after 1,001 ms
SSH: Disconnecting configuration [j6a301] ...
SSH: Transferred 0 file(s)
Finished: SUCCESS
```

console output

## 9. SSL 설정

### SSL 인증서 발급 및 적용

HTTPS 통신을 위해 certbot으로 SSL 인증서를 발급받아 nginx에 적용

#### snapd 최신 버전으로 설치

```
sudo snap install core
```

```
sudo snap refresh core
```

#### cerbot 설치

```
sudo snap install --classic certbot
```

### EC2에 nginx 설치

#### 패키지 업데이트

```
sudo apt-get update
```

#### nginx 설치

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

#### 설치 버전 확인

```
nginx -v
```

#### 인증서 발급

```
sudo certbot --nginx
```

- email address : 이메일 주소 입력
- 이용 약관 동의 : Y
- EFF의 소식 이메일 수신 여부 : N
- 도메인명 : k6a401.p.ssafy.io

#### 🔴likely firewall problem 오류

80, 443번 포트 방화벽 허용

```
sudo ufw allow 80
sudo ufw allow 443
```

#### 코드 적용 확인

```
sudo nginx -t
```

아래와 같이 나와야 정상적으로 적용 된 것 (에러 발생시 코드 재확인)

```
// nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
// nginx: configuration file /etc/nginx/nginx.conf test is successful
```

## nginx 설정

```
cd /etc/nginx/sites-available/
```

설정파일 추가

```
sudo vim myssl.conf
```

아래 내용 추가 후 저장

```
# myssl.conf

server{
    listen 80;
    listen [::]:80;
    server_name k6a401.p.ssafy.io;
    return 301 https://$host$request_uri;
}

# sub domain
server{
    root /var/www/html;
    index index.html index.htm index.nginx-debian.html;
    server_name whyweclimb.site www.whyweclimb.site;

    location /{
    }

}

server{
    listen 443 ssl;
    listen [::]:443 ssl;

    server_name k6a401.p.ssafy.io www.k6a401.p.ssafy.io;

    root /var/www/html;

    index index.html index.htm index.nginx-debian.html;

    proxy_connect_timeout 1d;
    proxy_send_timeout 1d;
    proxy_read_timeout 1d;

    ssl_certificate /etc/letsencrypt/live/k6a401.p.ssafy.io/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/k6a401.p.ssafy.io/privkey.pem;

    ssl_protocols TLSv1 TLSv1.1 TLSv1.2;

    location / {
        proxy_pass http://localhost:3000;
        proxy_redirect off;
    }

    location /api {
        proxy_pass http://localhost:8081;
        proxy_redirect off;
    }
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
}
```

설정파일 참고 블로그

#### Ngix 정적콘텐츠 제공

etc/nginx/ 폴더안에 sites-available, sites-enabled폴더가 있습니다. |—— sites-available # 비활성화된 설정 | |—— default |—— sites-enabled # 활성화된 설정 | |—— default -> /etc/nginx/sites-available/default # | sites-available의 파일을 심볼릭 링크로 연결 sites-available은 사용하지않는 설정파일을 저장합니다. 이 설정파일의 심볼릭 링크를 sites-enabled에 만들면, nginx는 설정을 읽고, 이를 실행합니다. 정적 파일을 제공하도록 nginx를 설정 해보겠습니다.  
<https://www.vompressor.com/setting-nginx-1/>

## 심볼릭 링크 추가

```
cd /etc/nginx/sites-enabled
sudo rm default
sudo ln -s /etc/nginx/sites-available/myssl.conf /etc/nginx/sites-enabled/myssl.conf
```

- 기존의 default 링크를 삭제하고 새로운 링크 추가

## 재시작하여 서버에 적용

```
sudo systemctl restart nginx
```

# 10. 프론트엔드 빌드 및 배포

## NVM을 사용하여 Node.js 및 npm 설치

### nvm 설치

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.1/install.sh | bash
```

터미널 재접속!

### node.js 16.13.1 설치

```
nvm install 16.13.1
```

### 설치된 node.js 와 npm 버전 확인

```
node -v
```

```
npm -v
```

### node npm 삭제 명령어

```
sudo apt-get remove --purge npm node nodejs
sudo apt purge nodejs npm
sudo apt-get autoremove
sudo rm -rf /usr/local/bin/npm /usr/local/share/man/man1/node* /usr/local/lib/dtrace/node.d ~/.npm ~/.node-gyp /opt/local/bin/node /op
sudo rm -rf /usr/local/lib/node* ; sudo rm -rf /usr/local/include/node* ; sudo rm -rf /usr/local/bin/node*
```

## Jenkins에 nodejs plugin 설치

Dashboard > Plugin Manager

대시보드로 돌아가기  
Jenkins 관리

## Plugin Manager

nodejs

업데이트된 플러그인 목록   **설치 가능**   설치된 플러그인 목록   고급

| Install                             | Name ↓  | Released        |
|-------------------------------------|---|-----------------|
| <input checked="" type="checkbox"/> | <b>NodeJS</b> 1.5.1<br>npm<br>NodeJS Plugin executes NodeJS script as a build step. | 2 mo 6 days ago |

Install without restart  
 Download now and install after restart  
 Update information obtained: 19 hr ago  
 지금 확인

## NodeJS 환경 설정

Dashboard > Global Tool Configuration

## NodeJS

NodeJS installations

Add NodeJS

NodeJS

Name

NodeJS-16.13.1

☒ Install automatically ?

Install from nodejs.org

Version

NodeJS 16.13.1

☐ Force 32bit architecture  
For the underlying architecture, if available, force the installation of the 32bit package. Otherwise the build will fail

Global npm packages to install

Specify list of packages to install globally -- see npm install -g. Note that you can fix the packages version by using the syntax 'packageName@version'

Global npm packages refresh hours

72  
Duration, in hours, before 2 npm cache update. Note that 0 will always update npm cache

Delete installer


Add Installer

## item 구성 - 빌드 환경

위에서 추가한 nodejs 선택

**빌드 환경**

☒ Delete workspace before build starts

 고급...

☐ Use secret text(s) or file(s) ?

☐ Provide Configuration files ?

☐ Send files or execute commands over SSH before the build starts ?

☐ Send files or execute commands over SSH after the build runs ?

☐ Abort the build if it's stuck

☐ Add timestamps to the Console Output

☐ Inspect build log for published Gradle build scans

☒ Provide Node & npm bin/ folder to PATH

**NodeJS Installation**

NodeJS-16.14.0

Specify needed nodejs installation where npm installed packages will be provided to the PATH

**npmrc file**

- use system default -

**Cache location**

Default (~/.npm or %APP\_DATA%\npm-cache)

## 빌드

## Dockerfile

### 위치

- S06P31A401
  - front
    - Dockerfile

### 파일 내용

```
FROM node:alpine
WORKDIR /usr/src/app
COPY package*.json ./
RUN apk --no-cache add tzdata && \
    cp /usr/share/zoneinfo/Asia/Seoul /etc/localtime && \
    echo "Asia/Seoul" > /etc/timezone
RUN npm i --force -y
COPY ./ ./
EXPOSE 3000
RUN npm run build
CMD [ "npm", "run", "start"]
```

## 배포

### 빌드 후 조치

아래 내용 추가

```
docker stop front
docker rm front
docker build -t front .
docker run -d -p 3000:3000 --name front front
```

⇒ 이것까지 추가하면 **빌드 후 조치**의 총 내용은 아래와 같음

```
docker stop spring-compose
docker rm spring-compose
cd /home/ubuntu/jenkins/workspace/whyweclimb/backend
docker build -t whyweclimb/backend .
docker run -d -e TZ=Asia/Seoul -p 8081:8081 --name spring-compose whyweclimb/backend
cd /home/ubuntu/jenkins/workspace/whyweclimb/front
docker stop front
docker rm front
docker build -t front .
```



```
docker run -d -p 3000:3000 --name front front
docker rmi $(sudo docker images -f "dangling=true" -q)
```

---