



How to Deploy

- [1. 배포란](#)
- [2. Local에서 Test](#)
 - [2.1 Git clone](#)
 - [2.2 Front test](#)
 - [2.3 Backend test](#)
- [3. Server에서 Test](#)
 - [3.1 서버 접속 & git clone](#)
 - [3.2 개발 환경 준비](#)
 - [3.3 Front test](#)
 - [3.4 Backend test](#)
- [4. Nginx deploy](#)
 - [4.1 Nginx install](#)
 - [4.2 Front](#)
 - [4.2.1 Front build](#)
 - [4.2.2 Nginx setting 수정](#)
 - [4.3 Backend](#)
 - [4.3.1 Backend build](#)
 - [4.3.2 Nginx setting 수정](#)
 - [4.3.3 Nginx용 Spring Boot 실행](#)
- [5. 백엔드 HA구성](#)
 - [5.1 Reverse proxy 설정 및 upstream 추가, proxy_pass 수정](#)
 - [5.2 Server 실행](#)
 - [5.3 Backend & Frontend 최신버전 반영하기](#)
 - [5.3.1 기존 프로세스 종료](#)
 - [5.3.1 Git clone](#)
 - [5.3.2 Build](#)
 - [5.3.3 Server 실행](#)
- [6. Jenkins CI/CD 구성](#)
 - [6.1 Jenkins설치](#)
 - [6.1.1 EC2에 Docker 설치](#)
 - [6.1.2 Docker container에 Jenkins설치 및 진행](#)
 - [6.1.3 Jenkins blue ocean 설치및 확인](#)
 - [6.2 Gitlab 플러그인 설치 및 연동](#)
 - [6.2.1 Gitlab API Token 발급](#)
 - [6.2.2 Gitlab 플러그인 설치 및 연결 설정](#)
 - [6.3 Maven, NodeJS 플러그인 설치](#)
 - [6.3.1 플러그인 설치](#)
 - [6.3.2 플러그인 설정](#)
 - [6.4 Jenkins project 구성하고 배포하기](#)
 - [6.4.1 Freestyle project 생성](#)
 - [6.4.2 Project에 gitlab 연동](#)
 - [6.4.3 Build stage 구성](#)
 - [6.4.3 Deploy stage 구성](#)

1. 배포란

| 소프트웨어를 사용자가 접근할 수 있는 환경에 배치시키는 일

- **컴파일**: 작성된 코드를 컴퓨터가 이해할 수 있는 언어로 번역하는 일
- **빌드**: 컴파일된 코드를 실제 실행할 수 있는 상태로 만드는 일
- **배포**: 빌드가 완성된 실행 가능한 파일을 사용자가 접근할 수 있는 환경에 배치시키는 일

배포는 [컴파일 → 빌드 → 배포] 의 과정을 거쳐야 한다.

보통 [컴파일 → 빌드] 의 과정을 빌드 한다 라고 표현한다.

Spring-boot를 예로 들자면 컴파일을 포함해 war, jar 등의 실행 가능한 파일을 뽑아 내기까지의 과정을 빌드한다고 표현한다.

이클립스에서 Java로 코딩을 한다고 생각해보자

코드를 짜고 나서 Run 버튼을 눌러 코드를 실행시킨다: [컴파일 → 실행]

정상적으로 실행되면 이것을 jar 파일로 뽑아서 웹서버에 올린다: [빌드 → 배포]

2. Local에서 Test

| Local에서 테스트를 진행하지 않고 배포하는것은 의미가 없다.
반드시 테스트 후 배포를 진행할 것

2.1 Git clone

```
git clone https://lab.ssafy.com/s04-webmobile2-sub1/skeleton-project.git
```

2.2 Front test

```
> cd frontend  
> npm install  
> npm run serve
```

- 아래 주소로 들어가서 확인한다.

<http://localhost:8080/>

- 참고

[Vue.js 단위 테스트](#)

[Jest와 Vue Test Utils\(VTU\)로 Vue 컴포넌트 단위테스트](#)

[Vue test 알아보기](#)

2.3 Backend test

```
cd backend  
  
# windows  
.\mvnw spring-boot:run
```

- 아래 url로 들어가서 확인한다.

<http://localhost:8080/swagger-ui.html>

- 참고

[STS 메이븐 프로젝트 jar 패키지 빌드](#)

3. Server에서 Test

3.1 서버 접속 & git clone

```
# 서버 접속  
ssh -i cert.pem ubuntu@i4c10x.p.ssafy.io  
  
# git clone  
git clone https://lab.ssafy.com/s04-webmobile2-sub1/skeleton-project.git
```

3.2 개발 환경 준비

- **DB 서버:** EC2에 DB를 설치하고 DB를 공유하여 개발하기
- <https://www.notion.so/Windows-ssh-AWS-1360a899220646628a2a6984c2f53bbb>
- **openjdk, maven, npm** 설치

```
# install openjdk-8  
sudo apt-get install openjdk-8-jdk  
java -version  
  
# install maven  
sudo apt install maven  
  
# install npm  
sudo apt install npm
```

3.3 Front test

```
cd frontend
npm install
npm run serve

# test:
```

- 아래 주소로 들어가서 확인한다
<http://i4c10x.p.ssafy.io:8080> (자신의 ec2 주소에 맞게 수정)
- Invalid Host header 발생시
skeleton-project/frontend/vue.config.js에 아래 내용 추가

```
module.exports = {
  configureWebpack: {
    // other webpack options to merge in ...
  },
  // devServer options dont belong into `configureWebpack`
  devServer: {
    host: "0.0.0.0",
    hot: true,
    disableHostCheck: true
  },
};
```

3.4 Backend test

```
mvn spring-boot:run
```

- 아래 주소로 들어가서 확인한다
<http://i4c10x.p.ssafy.io:8080/swagger-ui.html> (자신의 ec2 주소에 맞게 수정)

4. Nginx deploy

- 웹 서버: Nginx
- Nginx로 Reverse-Proxy 서버 만들기

4.1 Nginx install

```
sudo apt install nginx
nginx -v
```

4.2 Front

4.2.1 Front build

```
cd front
npm run build
```

4.2.2 Nginx setting 수정

```
sudo vim /etc/nginx/sites-enabled/default

#웹서버 루트 기존 설정은 주석처리, front를 build한 위치로 변경
-----
#root /var/www/html;
root /home/ubuntu/skeleton-project/frontend/dist;
...
-----

# 설정 변경 후 syntax 검사 필수
sudo nginx -t

# 설정 변경 후 Nginx 재시작
sudo service nginx restart
```

- 아래 주소로 들어가서 확인한다.
<http://i4c10x.p.ssafy.io> (자신의 ec2 주소에 맞게 수정)

4.3 Backend

4.3.1 Backend build

```
cd backend
mvn package
```

4.3.2 Nginx setting 수정

```
sudo vim /etc/nginx/sites-enabled/default

-----
server {
    listen 80;
    ...
    ...

    location / {
        # First attempt to serve request as file, then
        # as directory, then fall back to displaying a 404.
        try_files $uri $uri/ /index.html;
    }

    ### backend reverse proxy 설정 추가 ###
    location /api {
        proxy_pass http://localhost:8080;
        proxy_http_version 1.1;
        proxy_set_header Connection "";

        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header X-Forwarded-Host $host;
    }
}
```

```

        proxy_set_header X-Forwarded-Port $server_port;
    }
    ### 여기까지 ###
    ...
}
-----

```

```

# 설정 변경 후 syntax 검사 필수
sudo nginx -t

```

```

# 설정 변경 후 Nginx 재시작
sudo service nginx restart

```

4.3.3 Nginx용 Spring Boot 실행

```
cd backend
```

```

# 보안상 외부에서 8080으로 접근 불가하게 실행하려면 --server.address=127.0.0.1 옵션 추가
java -jar target/*.jar --server.servlet.context-path=/api

```

- 아래 주소로 들어가서 확인한다.

<http://i4c10x.p.ssafy.io/api/swagger-ui.html> (자신의 ec2 주소에 맞게 수정)

5. 백엔드 HA구성

| High Availability: 고 가용성

- HA란?

5.1 Reverse proxy 설정 및 upstream 추가, proxy_pass 수정

```
sudo vim /etc/nginx/sites-enabled/default
```

```

-----
# backend upstream 설정
upstream backend {
    server localhost:8080;
    server localhost:8081;
}

server {
    listen 80;
    ...

    location / {
        # First attempt to serve request as file, then
        # as directory, then fall back to displaying a 404.
        try_files $uri $uri/ /index.html;
    }

    location /api {
        # 이 부분 upstream으로 변경
        proxy_pass http://backend;
        proxy_http_version 1.1;
        proxy_set_header Connection "";

        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }
}

```

```

    proxy_set_header X-Forwarded-Proto $scheme;
    proxy_set_header X-Forwarded-Host $host;
    proxy_set_header X-Forwarded-Port $server_port;
}
}
-----

```

```

# 설정 변경 후 syntax 검사 필수
sudo nginx -t

```

```

# 설정 변경 후 Nginx 재시작
sudo service nginx restart

```

- Reverse Proxy

Reverse Proxy란?

Forward Proxy와 Reverse Proxy의 차이

5.2 Server 실행

- 터미널에서 backend 프로그램 실행 후 빠져나오면 프로세스가 종료됨 ⇒ `nohup, &`

```

# 첫번째 서버 실행
nohup java -jar target/*.jar \
--server.servlet.context-path=/api \
--server.address=127.0.0.1 \
--server.port=8080 &

```

```

# 두번째 서버 실행
nohup java -jar target/*.jar \
--server.servlet.context-path=/api \
--server.address=127.0.0.1 \
--server.port=8081 &

```

```

# process 확인
ps -ef | grep java

```

- Linux ps (프로세스 확인하기).

5.3 Backend & Frontend 최신버전 반영하기

5.3.1 기존 프로세스 종료

```
kill -9 `pgrep java`
```

- ps와 grep으로 pid를 조회하여 프로세스 종료

5.3.1 Git clone

```

# 기존 git repo 삭제
rm -rf ~/skeleton-project

# git clone
git clone https://lab.ssafy.com/s04-webmobile2-sub1/skeleton-project.git

```

- frontend와 backend의 build 폴더를 삭제하고, git pull 수행 후 다시 build 해도 상관없다.

5.3.2 Build

```
# front
cd frontend
npm install
npm run build

#backend
cd backend
mvn package
```

5.3.3 Server 실행

```
# 첫번째 서버 실행
nohup java -jar target/*.jar \
--server.servlet.context-path=/api \
--server.address=127.0.0.1 \
--server.port=8080 &

# 두번째 서버 실행
nohup java -jar target/*.jar \
--server.servlet.context-path=/api \
--server.address=127.0.0.1 \
--server.port=8081 &

# process 확인
ps -ef | grep java
```

- 아래 주소로 들어가서 확인한다.

frontend: <http://i4c10x.p.ssafy.io> (자신의 ec2 주소에 맞게 수정)

backend: <http://i4c10x.p.ssafy.io/api/swagger-ui.html> (자신의 ec2 주소에 맞게 수정)

6. Jenkins CI/CD 구성

본 문서는 SSAFY Jenkins를 사용하지 않습니다. 본 문서는 Jenkins 서버와 Deploy 서버가 같은 클라우드 서버 내에 있습니다.

현업에서는 관리하는 서버가 많습니다. API, ADMIN, CONNECTION, AUTH 등의 서버의 종류도 여러개이고 하나의 서버를 여러개 복제(scale-out)해서 배포합니다. 따라서 배포의 일정 부분을 자동화 해놓지 않으면 여러대의 모든 서버를 일일이 배포해야합니다.

배포 자동화를 진행하기 위해서 배포 전용 서버를 하나 둡니다. 서버를 두는 이유는 로컬 PC에서 배포를 진행한다면 각자의 PC 설정이 다르기 때문입니다. 이 서버에 배포에 필요한 스크립트를 작성합니다. 그 후 원격으로 트리거를 걸어서 필요한 서버의 갯수만큼 스크립트를 수행시키면 배포 작업들이 자동화 됩니다.

비유를 하자면 리모컨(배포 서버)을 하나 두고 해당 리모컨에 배포 버튼(배포 스크립트)을 만들고, 해당 버튼을 누르면(트리거) 배포 작업을 수행하는 것입니다.

스크립트에는 프로그래밍적인 요소를 담을 수 있습니다. 따라서 배포에 여러 전략들을 구사할 수 있습니다. 예를들면 1~20번의 서버가 있는데, 이중 1~10번의 서버만 새로 배포된 서버로 바뀌고, 일정시간이 지나면 11~20번의 서버를 교체하는 등의 안정화 전략을 적용할 수 있습니다.

⇒ [git clone → build → restart] 하는 작업을 배포서버의 스크립트를 실행하는 작업으로 자동화

6.1 Jenkins설치

Jenkins는 리모컨입니다. 배포 전용 서버인 Jenkins를 설치해 봅시다.

Jenkins는 JavaRuntime 위에서 동작하는 프로그램입니다. 따라서 Jenkins가 돌아가고 있는 서버에는 JavaRuntime 환경이 구축되어 있어야 합니다. 본 문서는 Jenkins 서버와 ec2 서버가 같은 클라우드 서버 내에 있으므로 환경 분리를 위해서 docker를 사용합니다.

6.1.1 EC2에 Docker 설치

- 아래의 명령을 실행한다.

```
> sudo apt update
> sudo apt upgrade
> sudo apt install apt-transport-https ca-certificates
> sudo apt install curl gnupg-agent software-properties-common
```

apt-transport-https : 패키지 관리자가 https를 통해 데이터 및 패키지에 접근할 수 있도록 한다.

ca-certificates : certificate authority에서 발행하는 디지털 서명. SSL 인증서의 PEM 파일이 포함되어 있어 SSL기반 앱이 SSL 연결이 되어 있는지를 확인할 수 있다.

curl : 특정 웹사이트에서 데이터를 다운로드 받을 때 사용

software-properties-common : PPA(Personal Package Archive)를 추가하거나 제거할 때 사용한다

- Docker의 공식 GPG 키를 추가한다.

```
> curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add
```

- stable repository를 세팅하기 위한 명령어를 실행한다.

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

add-apt-repository : PPA 저장소를 추가해준다. apt 리스트에 패키지를 다운로드 받을 수 있는 경로가 추가된다.

- 가장 최신 버전의 Docker 엔진을 설치한 후, 버전을 확인한다.

```
> sudo apt update
> sudo apt install docker-ce docker-ce-cli containerd.io
> docker -v\
```

```
ubuntu@ip-172-26-3-202:~$ docker -v
Docker version 20.10.3, build 48d30b5
```

6.1.2 Docker container에 Jenkins설치 및 진행

- Docker Container에 Jenkins 설치 후 실행

```
$ sudo docker run -d \
-u root \
-p 9090:8080 \
--name=jenkins \
-v /home/ubuntu/docker/jenkins-data:/var/jenkins_home \
-v /var/run/docker.sock:/var/run/docker.sock \
-v "$HOME"/home/jenkinsci/blueocean \
jenkinsci/blueocean
```

- Docker container 접속

```
> sudo docker exec -it jenkins /bin/bash
```

- `http://<your-aws-domain>:<jenkins port>` 접속 후 admin password 입력
ex) `http://t4coach44.p.ssafy.io:9090`

```
# admin password 확인
> cat /var/jenkins_home/secrets/initialAdminPassword
```

```
ubuntu@ip-172-26-3-202:~$ sudo docker exec -it jenkins /bin/bash
bash-5.0# cat /var/jenkins_home/secrets/initialAdminPassword
0020-70101010-1140641-0Fb26337
bash-5.0#
```

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

- Jenkins 설치 (install suggested plugin)

Getting Started

Getting Started

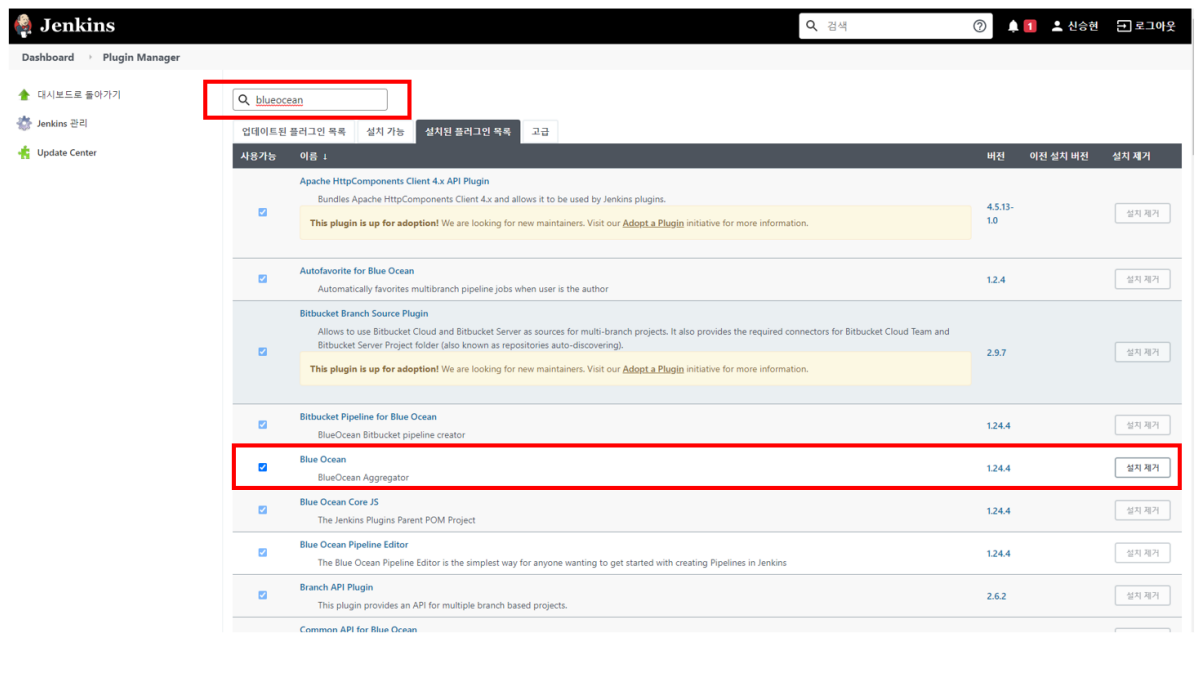
✓ Folders Plugin	✓ OWASP Markup Formatter Plugin	✓ Build Timeout	✓ Credentials Binding Plugin	Folders
<input type="radio"/> Timestamper	<input type="radio"/> Workspace Cleanup	<input type="radio"/> Ant	<input type="radio"/> Gradle	OWASP Markup Formatter
Pipeline	GitHub Branch Source Plugin	Pipeline: GitHub Groovy Libraries	<input type="radio"/> Pipeline: Stage View	Build Timeout
<input type="radio"/> Git plugin	<input type="radio"/> SSH Build Agents	Matrix Authorization Strategy Plugin	<input type="radio"/> PAM Authentication	Credentials Binding
<input type="radio"/> LDAP	<input type="radio"/> Email Extension	<input type="radio"/> Mailer Plugin		

** - required dependency

Jenkins 2.263.3

6.1.3 Jenkins blue ocean 설치및 확인

- Jenkins 관리 → 플러그인 관리 → 설치된 플러그인 목록 → filter에 [blueocean] 검색후 확인

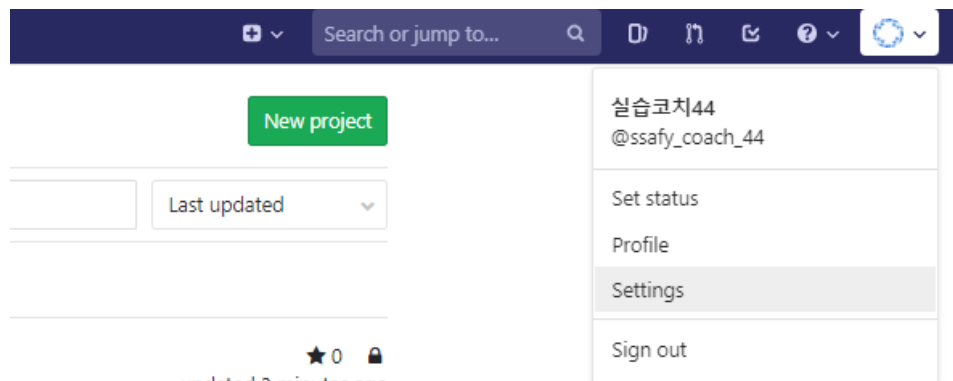


6.2 Gitlab 플러그인 설치 및 연동

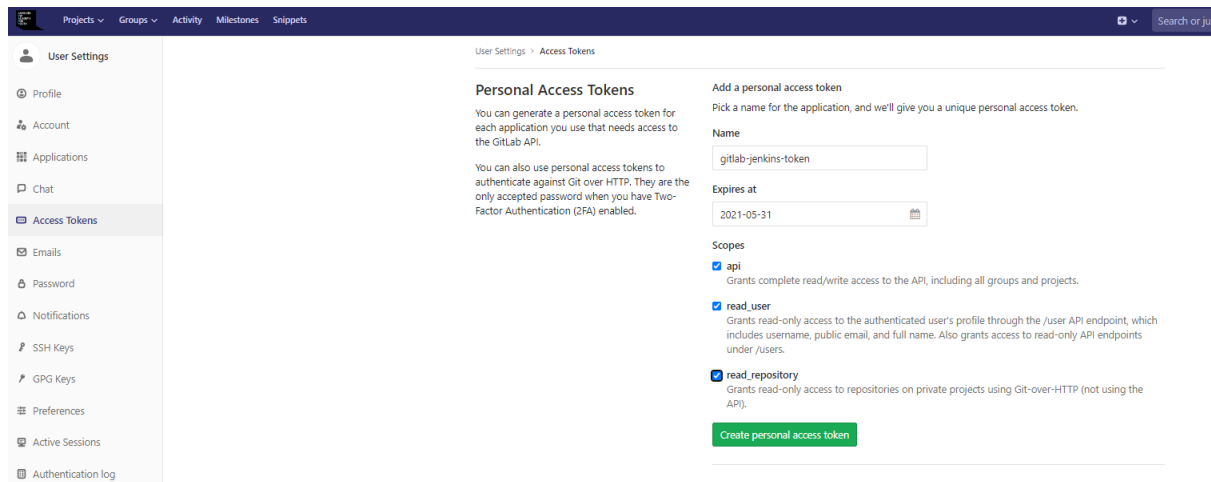
Jenkins를 설치함으로써 우리는 비어있는 리모컨을 만들었습니다. 리모컨의 버튼을 만들기 전에 git clone을 수행할 수 있도록 Gitlab 플러그인을 설치해 해봅시다.

6.2.1 Gitlab API Token 발급

- lab.ssafy.com의 User → settings



- access token → 내용 작성 후 토큰 발급



- 발급된 토큰은 다시 볼 수 없으므로 복사에서 저장하기

Your new personal access token has been created.

Personal Access Tokens

You can generate a personal access token for each application you use that needs access to the GitLab API.

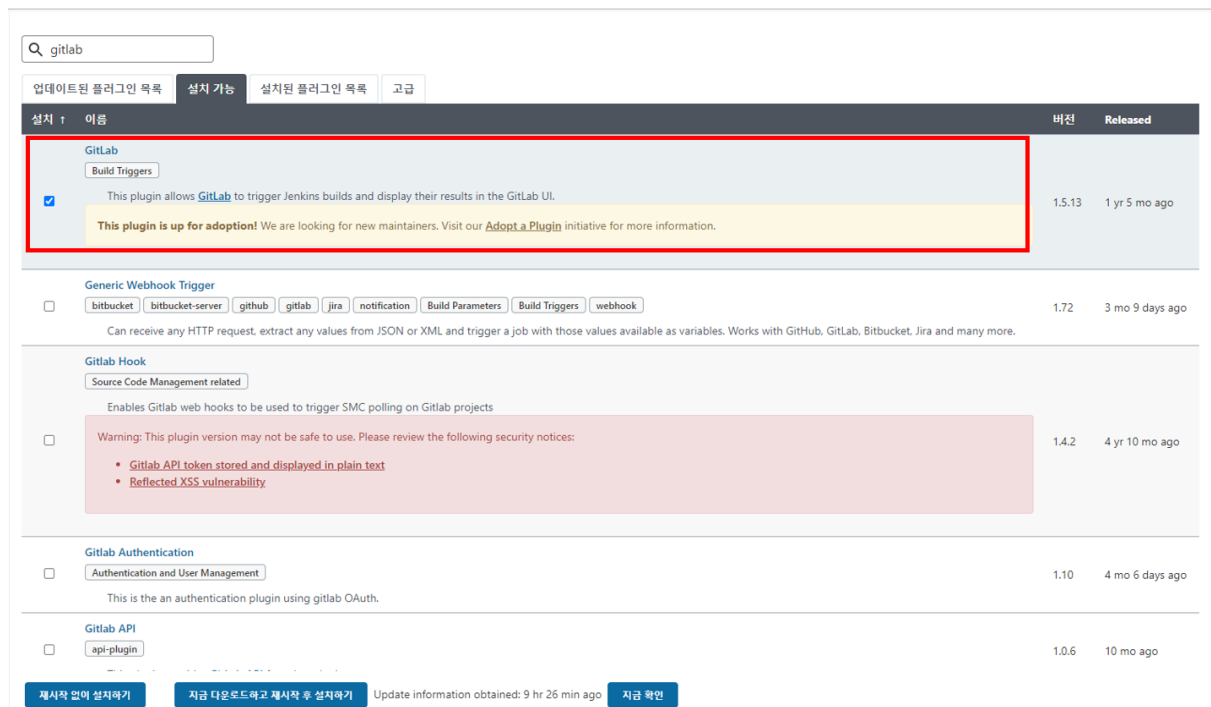
Your New Personal Access Token

[REDACTED]

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

6.2.2 Gitlab 플러그인 설치 및 연결 설정

- Jenkins 관리 → 플러그인 관리 → 설치 가능 → filter에 [gitlab] 검색 → gitlab 체크 후 설치



- Jenkins 관리 → 시스템 설정 → gitlab 부분 작성

Gitlab

Enable authentication for '/project' end-point ☒

GitLab connections

Connection name gitlab_connection

A name for the connection

Gitlab host URL https://lab.ssafy.com

The complete URL to the Gitlab server (e.g. http://gitlab.mydomain.com)

Credentials

- none -

Add

API Token

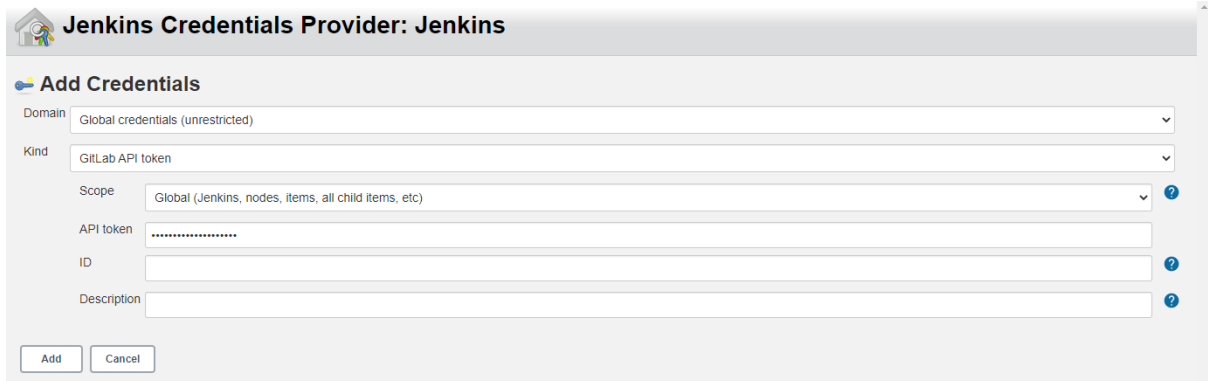
Jenkins

required

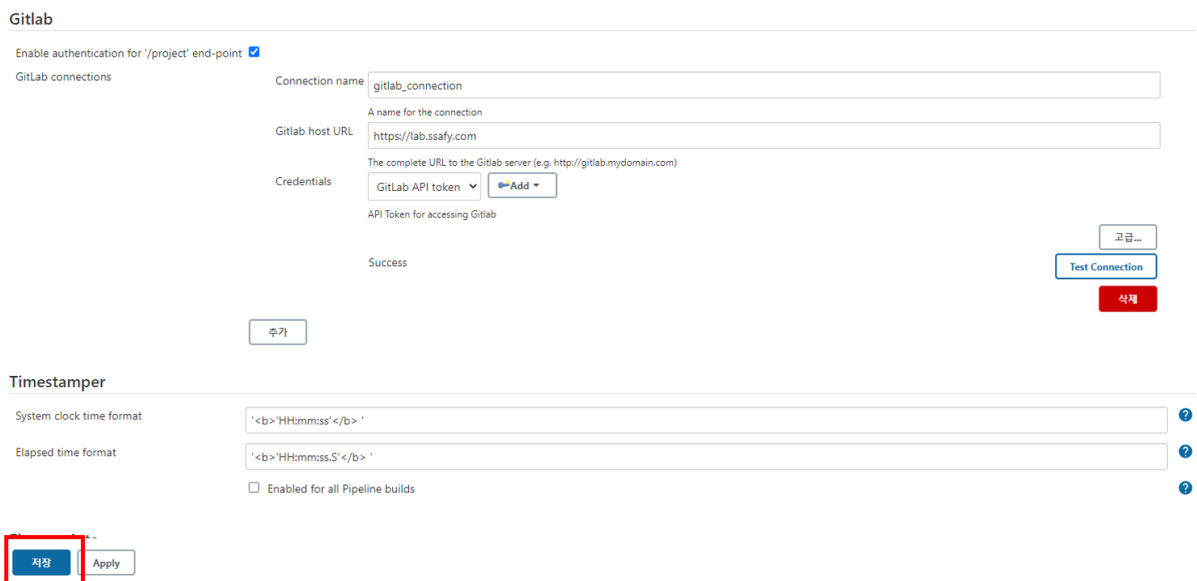
API Token for accessing Gitlab

Jenkins Credentials Provider

- Credentials → Add → Jenkins를 클릭하여 Gitlab API Token 생성
- 위에서 발급한 Gitlab API Token을 API Token란에 기입



- Test Connection으로 확인 및 저장



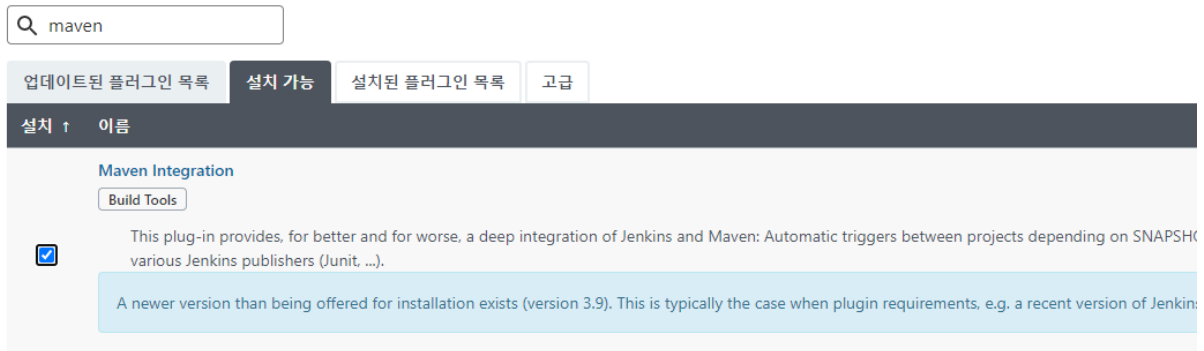
6.3 Maven, NodeJS 플러그인 설치

Git clone을 수행할 수 있게 되었다면, build도 자동으로 수행할 수 있게 만들어야겠죠?
build를 수행하기 위해서 Spring의 패키지 매니저인 Maven과 Vue.js의 패키지 매니저인 NodeJS 플러그인을 설치해 봅시다.

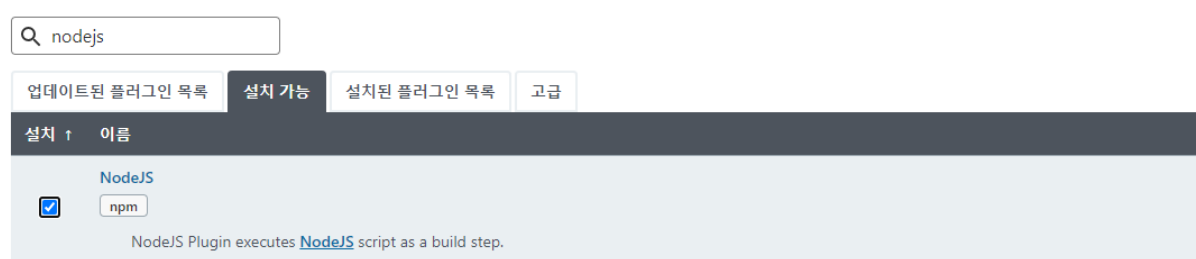
Spring을 사용한다면 Maven, Vue.js를 사용한다면 Node.js 플러그인을 설치합니다.

6.3.1 플러그인 설치

- Jenkins 관리 → 플러그인 설정 → 설치가능 탭 이동 → maven 검색 후 설치

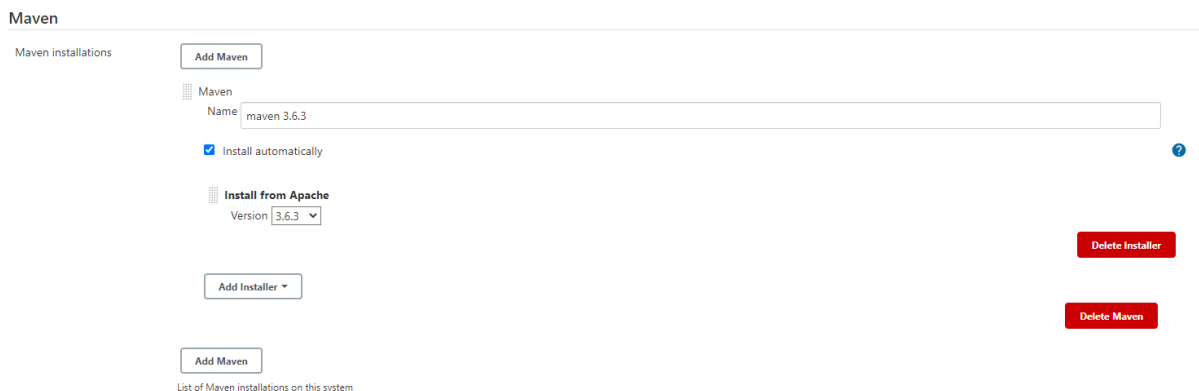


- Jenkins 관리 → 플러그인 설정 → 설치가능 탭 이동 → nodejs 검색 후 설치



6.3.2 플러그인 설정

- Jenkins 관리 → Global Tool Configuration에서 Maven, NodeJS 설정하기
- 개발에 사용된 버전과 동일한 버전 사용



NodeJS

NodeJS installations

[Add NodeJS](#)

NodeJS

Name

☒ Install automatically

[Install from nodejs.org](#)

Version

Force 32bit architecture ☐

Global npm packages to install

Global npm packages refresh hours

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

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'

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

[Delete Installer](#)

[Add Installer](#)

[Delete NodeJS](#)

[Add NodeJS](#)

List of NodeJS installations on this system

6.4 Jenkins project 구성하고 배포하기


플러그인들을 다 설치했다면 clone, build, deploy의 각 3가지 파트를 생각하면서 프로젝트(버튼)를 구성해 봅시다.

6.4.1 Freestyle project 생성

- 새로운 item → freestyle project 생성

Enter an item name

» Required field

 **Freestyle project**

이것은 Jenkins의 주요 기능입니다. Jenkins은 어느 빌드 시스템과 어떤 SCM(형상관리)으로 묶인 당신의 프로젝트를 빌드할 수 있습니다.

6.4.2 Project에 gitlab 연동

- 소스 코드 관리 탭 작성

`https://[token-id]:[token-value]@lab.ssafy.com/[user-name]/[project-name]`

소스 코드 관리

☐ None
☒ Git

Repositories

Repository URL:

Credentials:

Branches to build

Branch Specifier (blank for 'any'):

Repository browser:

Additional Behaviours:

- Build now 클릭 후 SUCCESS 확인

대시보드로 돌아가기

상태

변경사항

작업공간

Build Now

구성

Project 삭제

Favorite

블루 옵션 열기

Rename

Project project_01

작업 공간

최근 변경사항

고정링크

- [Last build, \(#1\), 3 hr 12 min 전](#)
- [Last stable build, \(#1\), 3 hr 12 min 전](#)
- [Last successful build, \(#1\), 3 hr 12 min 전](#)
- [Last completed build, \(#1\), 3 hr 12 min 전](#)

Build History 추이 ^

#1	2021. 2. 15 오전 12:28
Success > 콘솔 출력 (전체) Atom feed (실패)	

- 콘솔을 보고 어떤 명령들이 수행됐는지를 살펴보자

콘솔 출력

```
Started by user 신승현
Running as SYSTEM
Building in workspace /var/jenkins_home/workspace/project_01
The recommended git tool is: NONE
No credentials specified
> git rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://gitlab-jenkins-token:pAafLEyxosbJQIXDbkCI@lab.ssafy.com/ssafy_coach_44/testproject # timeout=10
Fetching upstream changes from https://gitlab-jenkins-token@lab.ssafy.com/ssafy_coach_44/testproject
> git --version # timeout=10
> git --version # 'git version 2.26.2'
> git fetch --tags --force --progress -- https://gitlab-jenkins-token:pAafLEyxosbJQIXDbkCI@lab.ssafy.com/ssafy_coach_44/testproject +refs/heads/:
> git rev-parse refs/remotes/origin/develop^{commit} # timeout=10
Checking out Revision ddad44b9d7bec84d17938c8d2adec84d674d3e12 (refs/remotes/origin/develop)
> git config core.sparsecheckout # timeout=10
> git checkout -f ddad44b9d7bec84d17938c8d2adec84d674d3e12 # timeout=10
Commit message: "update jf"
First time build. Skipping changelog.
Finished: SUCCESS
```

- workspace에 clone이 받아졌는지 확인

```
> cd ~/docker/jenkins-data/workspace/[jenkins-project-name]/
```

```
ubuntu@ip-172-26-3-202:~$ cd docker/jenkins-data/workspace/project_01/
ubuntu@ip-172-26-3-202:~/docker/jenkins-data/workspace/project_01$ ls
Jenkinsfile  README.md  backend  frontend
```

6.4.3 Build stage 구성

- Backend : 구성 → Build → Add Build step → Invoke top-level Maven targets 추가 후 Maven 빌드 설정

Build

Invoke top-level Maven targets

Maven Version: maven 3.6.3

Goals: clean package

POM: backend/pom.xml

Properties:

JVM Options:

☐ Inject build variables

☐ Use private Maven repository

Settings file: Use default maven settings

Global Settings file: Use default maven global settings

Add build step

- Frontend : 구성 → 빌드 환경 → Provide Node & npm bin/ folder to PATH 체크

빌드 환경

- ☐ Delete workspace before build starts
- ☐ Use secret text(s) or file(s)
- ☐ Provide Configuration files
- ☐ 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 10.19.0

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)

☐ With Ant

- **Frontend** : 구성 → Build → Add Build step → Execute shell 추가 후 스크립트 설정

Execute shell

Command

```
### frontend build ###
cd frontend
npm install
npm run build
```

See [the list of available environment variables](#)

고급...

- Build now 클릭 후 SUCCESS 확인

콘솔 출력

```
Started by user 신승현
Running as SYSTEM
Building in workspace /var/jenkins_home/workspace/project_01
The recommended git tool is: NONE
No credentials specified
> git rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://gitlab-jenkins-token:pAafLEyxosbJQIXDbkCI@lab.ssaify.com:ssaify_coach_44/testproject # timeout=10
Fetching upstream changes from https://gitlab-jenkins-token@lab.ssaify.com:ssaify_coach_44/testproject
> git --version # timeout=10
> git --version # 'git version 2.26.2'
> git fetch --tags --force --progress -- https://gitlab-jenkins-token:pAafLEyxosbJQIXDbkCI@lab.ssaify.com:ssaify_coach_44/testproject +re
> git rev-parse refs/remotes/origin/develop^{commit} # timeout=10
Checking out Revision ddad44b9d7bec84d17938c8d2adec84d674d3e12 (refs/remotes/origin/develop)
> git config core.sparsecheckout # timeout=10
> git checkout -f ddad44b9d7bec84d17938c8d2adec84d674d3e12 # timeout=10
Commit message: "update jf"
> git rev-list --no-walk ddad44b9d7bec84d17938c8d2adec84d674d3e12 # timeout=10
[project_01] $ /var/jenkins_home/tools/hudson.tasks.Maven_MavenInstallation/maven-3.6.3/bin/mvn -f backend/pom.xml clean package
[INFO] Scanning for projects...
[INFO]
[INFO] -----< com.web.curation:webcuration >-----
[INFO] Building webcuration 0.0.1-SNAPSHOT
[INFO] -----[ jar ]-----
[INFO]
[INFO] --- maven-clean-plugin:3.1.0:clean (default-clean) @ webcuration ---
[INFO] Deleting /var/jenkins_home/workspace/project_01/backend/target
[INFO]
[INFO] --- maven-resources-plugin:3.1.0:resources (default-resources) @ webcuration ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 1 resource
[INFO] Copying 0 resource
[INFO]
[INFO] --- maven-compiler-plugin:3.8.1:compile (default-compile) @ webcuration ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 8 source files to /var/jenkins_home/workspace/project_01/backend/target/classes
[INFO]
[INFO] --- maven-resources-plugin:3.1.0:testResources (default-testResources) @ webcuration ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /var/jenkins_home/workspace/project_01/backend/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.8.1:testCompile (default-testCompile) @ webcuration ---
[INFO]
[INFO]
[project_01] $ /bin/sh -xe /tmp/jenkins5990124410938557034.sh
+ cd frontend
+ npm install
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@2.3.2 (node_modules/fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.3.2: wanted {"os":"darwin","arch":"any"} (current: {"os":"linux","arch":"x64"})
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.13 (node_modules/jest-haste-map/node_modules/fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.13: wanted {"os":"darwin","arch":"any"} (current: {"os":"linux","arch":"x64"})
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.13 (node_modules/watchpack-chokidar2/node_modules/fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.13: wanted {"os":"darwin","arch":"any"} (current: {"os":"linux","arch":"x64"})
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.13 (node_modules/webpack-dev-server/node_modules/fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.13: wanted {"os":"darwin","arch":"any"} (current: {"os":"linux","arch":"x64"})

audited 1719 packages in 12.566s

65 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

+ npm run build
> saffy_vue@0.1.0 build /var/jenkins_home/workspace/project_01/frontend
> vue-cli-service build

- Building for production...
DONE Compiled successfully in 3226ms4:52:37 AM

  File                                Size      Gzipped
  dist/js/chunk-vendors.b9479660.js   135.94 KiB  47.02 KiB
  dist/js/app.d055de09.js             33.52 KiB   11.07 KiB
  dist/css/app.890f04ef.css           45.15 KiB   10.93 KiB

Images and other types of assets omitted.

DONE Build complete. The dist directory is ready to be deployed.
INFO Check out deployment instructions at https://cli.vuejs.org/guide/deployment.html
Finished: SUCCESS
```

- workspace에서 build가 잘 되었는지 확인

```
ubuntu@ip-172-26-3-202:~/docker/jenkins-data/workspace/project_01$ cd backend/
ubuntu@ip-172-26-3-202:~/docker/jenkins-data/workspace/project_01/backend$ ls
Dockerfile mvnw mvnw.cmd pom.xml src ssafy-sk.sql target
ubuntu@ip-172-26-3-202:~/docker/jenkins-data/workspace/project_01/backend$ cd ../frontend/
ubuntu@ip-172-26-3-202:~/docker/jenkins-data/workspace/project_01/frontend$ ls
Dockerfile babel.config.js eslintrc.js node_modules package.json src
README.md dist jest.config.js package-lock.json public vue.config.js
```

6.4.3 Deploy stage 구성

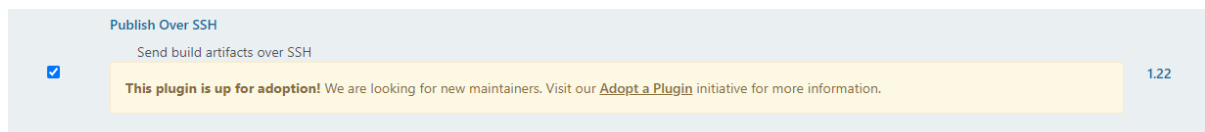
- Nginx setting 변경

```
sudo vim /etc/nginx/sites-enabled/default
```

```
root /home/ubuntu/deploy/dist;
...
```

```
sudo nginx -t
sudo service nginx restart
```

- SSH 플러그인 설치



- Jenkins 관리 → 시스템 설정 → Publish over SSH 탭 설정 후 test configuration으로 확인

Publish over SSH

Jenkins SSH Key

Passphrase: [Change Password](#)

Path to key:

Key:

```
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAmRyJJPrSc+4MAW1igQ6YpdUixM64u83awjhvriRhInDUaldw
2EV/h6aSOUYRqD3FmddbCk8MXxUPeyefbwQ1jEOQ1VtX5NU7ZeLL7eDrDpWnzD
IEVVAOX48Yn0kzb5ZtyFaitn03CyYuN2w3GGFr3orMinz6apBotfYbbjgVeQ8Yxv
vnt8gHlZwBfItiOD8sryq39kLk4t3U1nLPu6GiZop49/LnnjnOdcHUo4RQWEMk3
-----
```

Disable exec: ☐

SSH Servers

SSH Server

Name:

Hostname:

Username:

Remote Directory:

[추가](#) [고급...](#) [삭제](#) [Test Configuration](#)

key : ec2 서버 접속에 사용되는 pem키

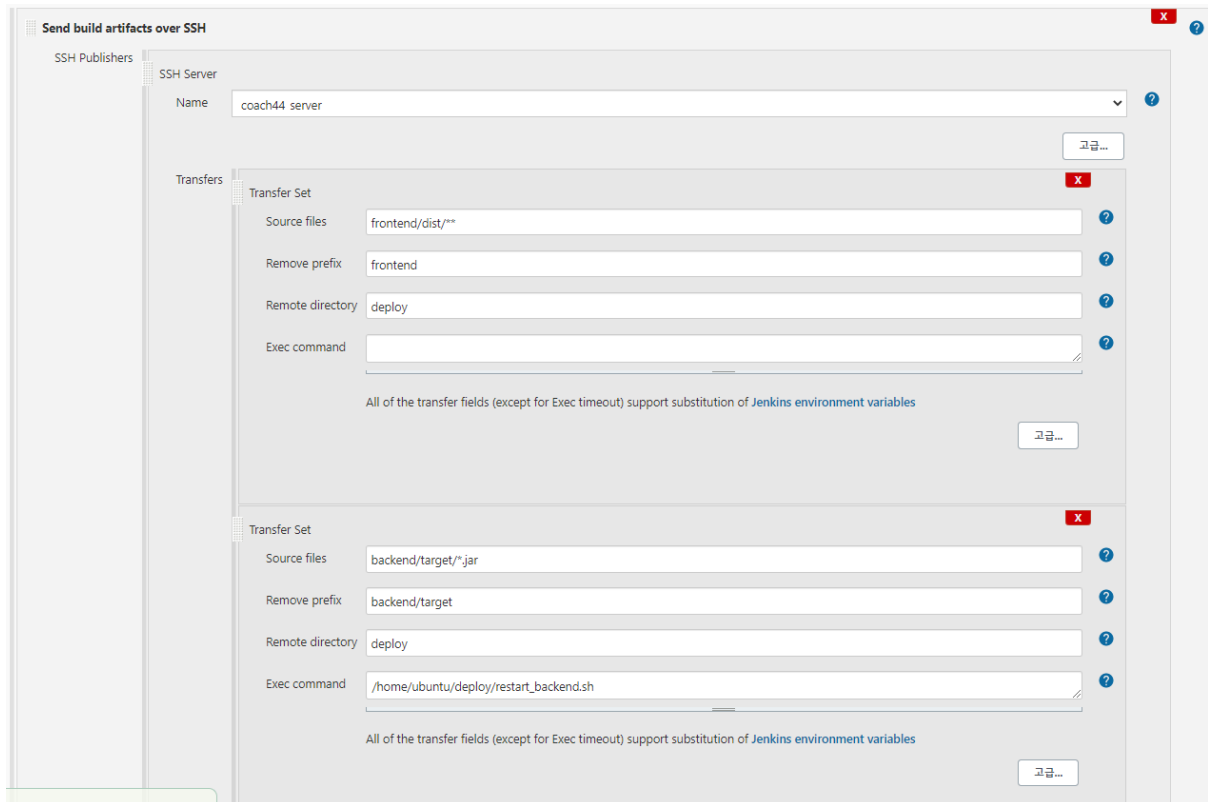
Name : 원하는 SSH 서버 이름

Hostname : ec2 서버의 주소

Username : 접속할 username (우리는 ubuntu)

Remote Directory : 배포할 서버의 기본 workspace, 파일을 업로드 할 기본 디렉토리, default: ~/

- 프로젝트 구성 → 빌드 후 조치 → 빌드 후 조치 추가 클릭 후 Send build artifacts over SSH 추가 및 작성



Name : Jenkins 시스템에서 등록한 SSH 서버

Source files : 어떤 파일을 배포할 것인지 설정

Remove prefix : 제거할 접두사

Remote directory : 파일이 저장될 디렉토리, 없다고 자동으로 생성해 주지 않기 때문에 미리 만들어 놓아야 한다.

Exec command : 배포 후 실행할 명령어, 서버에 있는 스크립트를 지정할수도 있다. 지정시에는 절대경로를 적어줘야 함

- restart_backend.sh 작성

```
kill $(cat deploy/app.pid)

nohup java -jar deploy/*.jar --server.servlet.context-path=/api --server.address=127.0.0.1 --server.port=8080 \
--spring.pid.file=deploy/app.pid >> deploy/app.log 2>&1 &
```

- chmod 변경

```
chmod +x ~/deploy/restart_backend1.sh
```

- chmod가 변경되면 초록색으로 바뀐다.

```
ubuntu@ip-172-26-3-202:~/deploy$ ls
restart_backend.sh
```

- pid 파일이 생성되도록 ApplicationPidFileWriter 추가 (서버 실행시 자동 생성되는 application.pid 파일이 커밋되지 않도록 .gitignore에 등록)

```
@SpringBootApplication
public class WebCurationApplication {
    public static void main(String[] args) {
        SpringApplication app = new SpringApplication(WebCurationApplication.class);
        app.addListeners(new ApplicationPidFileWriter()); // pid 파일을 생성하는 writer 등록
        app.run(args);
    }
}
```

- Build now 클릭 후 배포 확인

```
ubuntu@ip-172-26-3-202:~/deploy$ ls
app.log app.pid dist restart_backend.sh webcuration-0.0.1-SNAPSHOT.jar
ubuntu@ip-172-26-3-202:~/deploy$ ps -ef | grep java
root      353240  353189  0 Feb10 ?        00:21:17 java -Duser.home=/var/jenkins
_home -Djenkins.model.Jenkins.slaveAgentPort=50000 -jar /usr/share/jenkins/jenkins.war
ubuntu    396318      1 47 17:16 ?        00:00:12 java -jar deploy/webcuration-
0.0.1-SNAPSHOT.jar --server.servlet.context-path=/api --server.address=127.0.0.1
--server.port=8080 --spring.pid.file=deploy/app.pid
ubuntu    396367  382093  0 17:16 pts/0    00:00:00 grep --color=auto java
ubuntu@ip-172-26-3-202:~/deploy$
```

- 한번 더 Build now 클릭 후 pid 바뀌는지 확인

```
ubuntu@ip-172-26-3-202:~/deploy$ ps -ef | grep java
root      353240  353189  0 Feb10 ?        00:21:17 java -Duser.home=/var/jenkins
_home -Djenkins.model.Jenkins.slaveAgentPort=50000 -jar /usr/share/jenkins/jenkins.war
ubuntu    396318      1 47 17:16 ?        00:00:12 java -jar deploy/webcuration-
0.0.1-SNAPSHOT.jar --server.servlet.context-path=/api --server.address=127.0.0.1
--server.port=8080 --spring.pid.file=deploy/app.pid
ubuntu    396367  382093  0 17:16 pts/0    00:00:00 grep --color=auto java
ubuntu@ip-172-26-3-202:~/deploy$ ps -ef | grep java
root      353240  353189  0 Feb10 ?        00:21:19 java -Duser.home=/var/jenkins
_home -Djenkins.model.Jenkins.slaveAgentPort=50000 -jar /usr/share/jenkins/jenkins.war
ubuntu    396620      1 99 17:17 ?        00:00:13 java -jar deploy/webcuration-
0.0.1-SNAPSHOT.jar --server.servlet.context-path=/api --server.address=127.0.0.1
--server.port=8080 --spring.pid.file=deploy/app.pid
ubuntu    396660  382093  0 17:17 pts/0    00:00:00 grep --color=auto java
```