

포팅 매뉴얼

1. CI/CD를 위한 공통 사항

- jenkins 관리 > 시스템 설정 > Publish over SSH 설정 에 들어가서 AWS 접근에 필요한 key파일을 지정

1. Key 파일 지정

- 제공받은 pem파일의 내용을 넣어주면 된다.

Publish over SSH

Jenkins SSH Key

Passphrase

Concealed

Change Password

Path to key

Key

-----BEGIN RSA PRIVATE KEY-----

-----END RSA PRIVATE KEY-----

저장

Apply

2. ssh server 설정

- 제공받은 hostname과 username을 적어주면 된다.
- 여기서는 Hostname : j5b101.p.ssafy.io, Username : ubuntu으로 설정

SSH Servers

SSH Server

Name ?

ssafy_b101

Hostname ?

Username ?

ubuntu

Remote Directory ?

Gitlab 플러그인 설치 및 설정

- **jenkins 관리** > **플러그인 관리** > **설치가능** 에서 GitLab 설치
- **jenkins 관리** > **시스템 설정** > **Gitlab 설정** 에서 접근하려고 하는 Gitlab을 지정

Gitlab

☒ Enable authentication for '/project' end-point

GitLab connections

Connection name

b101

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

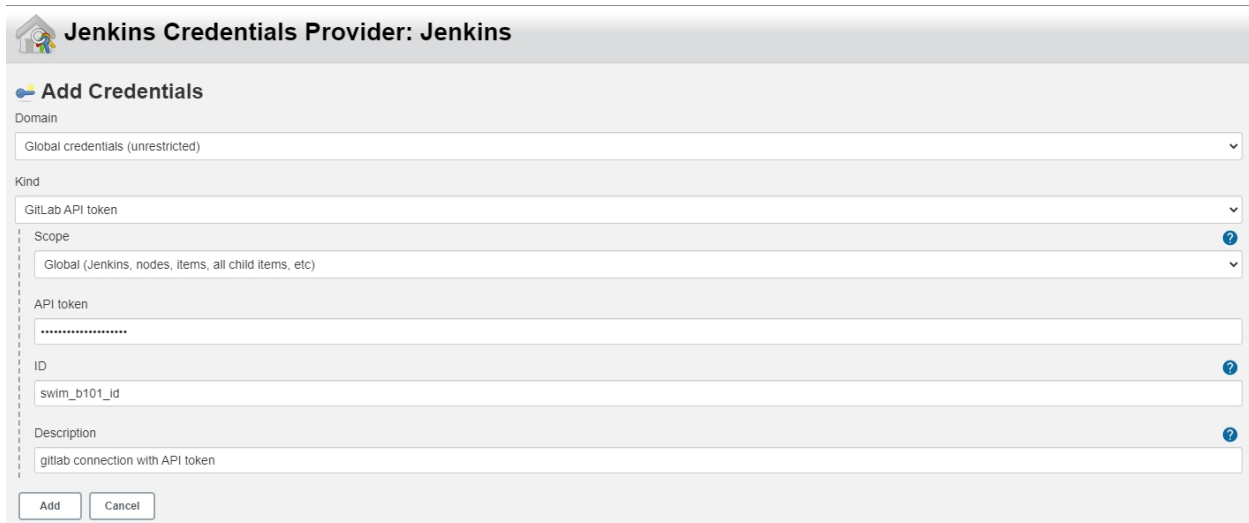
API Token for Gitlab access required

API Token for accessing Gitlab

- **Credentials** 가 없다면 Add를 눌러 새롭게 추가하면 된다. 여기서는 GitLab API token을 이용해서 등록했다.

Gitlab API는 gitlab 프로젝트에서 **Settings** > **Access Tokens** 에 들어가면 받을 수 있다.

나의 경우 : R2KiicAnBb7Zz3QJmXmG



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
swim_b101_id

Description
gitlab connection with API token

Add Cancel

- 그리고 해당 API token을 등록해주면 된다

Gitlab

☒ Enable authentication for '/project' end-point

GitLab connections

Connection name

b101

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

GitLab API token (gitlab connection with API token)

Add

API Token for accessing Gitlab

- Test Connection 버튼을 눌러서 gitlab과 연동이 되었는지 확인할 수 있다
- Success가 뜨면 성공

소스 코드 관리 설정하기

- Pull할 repository를 등록한다. git branch 전략에 따라 배포하고 싶은 branch로 설정한다.
 - Branch Specifier로 설정 가능

소스 코드 관리

☐ None
☒ Git

Repositories

Repository URL
 https://lab.ssafty.com/s05-blockchain/S05P218101.git

Credentials
 aysul0@naver.com/***** Add

고급...

Add Repository

Branches to build

Branch Specifier (blank for 'any')
 */master

Add Branch

Credential은 Username with password로 Username은 gitlab의 아이디, password는 비밀번호를 입력해주면 된다.

Jenkins Credentials Provider: Jenkins

Add Credentials

Domain
 Global credentials (unrestricted)

Kind
 Username with password

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

Username
 aysul0@naver.com

☐ Treat username as secret

Password

ID

Description

Add Cancel

빌드 유발 설정하기

- webhook 시그널을 받고 빌드할 수 있도록 트리거 설정을 해준다.

☒ Build when a change is pushed to GitLab webhook URL: 을 체크

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://3.34.191.232:8080/project/b101 ?

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

Gitlab 시크릿 토큰값 설정하기

- Gitlab webhook 설정에 필요한 시크릿 토큰값을 생성한다.

- ☒ 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 ?

b4f399ba173c5eb5a43cbc1134b1e219

Generate

Clear

Webhook 지정하기

- Gitlab에 지정한 트리거가 발생하면 jenkins로 시그널을 보내줘야 한다. 자동으로 이벤트를 감지하고 시그널을 보낼 수 있도록 다음과 같이 gitlab webhook을 지정해준다.
- Gitlab 프로젝트에서 **Settings** > **Webhooks**

- URL과 secret token을 입력
 - Jenkins에 webhook 시그널이 갈 수 있도록 URL을 입력해준다. 빌드 유발에서 체크할 때 명시되는 URL을 적어주면 된다.
 - 그리고 인증을 위해 빌드 유발 고급 옵션에서 생성한 시크릿 토큰을 입력해 준다.

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.

URL

URL must be percent-encoded if necessary.

Secret token

Use this token to validate received payloads. It is sent with the request in the X-Gitlab-Token HTTP header.

Trigger

☒ Push events

URL is triggered by a push to the repository

빌드 플러그인 설정

빌드를 위한 플러그인 설정

- NodeJS 빌드를 위한 NodeJS 플러그인을 설치

빌드 플러그인 설정하기

- `jenkins 관리` > `Global Tool Configuration` 에서 nodejs와 gradle 버전을 설치

NodeJS
NodeJS installations
Add NodeJS
NodeJS
Name
node 16.9.1
☒ Install automatically
Install from nodejs.org
Version
NodeJS 16.9.1
☐ 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
Add Installer
Delete Installer
Delete NodeJS

Gradle
Gradle installations
Add Gradle
Gradle
name
gradle 6.9.1
☒ Install automatically
Install from Gradle.org
Version
Gradle 6.9.1
Add Installer
Delete Installer
Delete Gradle

빌드 환경 설정하기

- jenkins > [Item_name] > 구성 에서 빌드환경을 추가
- 아까 설치한 NodeJS 플러그인을 등록

빌드 환경

- ☐ Delete workspace before build starts
- ☐ 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

node 16.9.1

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

Pull 테스트

- `jenkins` > `[Item_name]` 에서 Build Now 를 클릭하여 빌드가 되는지 확인
- 정상적으로 실행되고 나면 git repository에 있는 master branch를 땡겨온 것을 확인할 수 있다.
- Docker volume 설정한 `/app/swim/workspace`에 git repository 땡겨오기

```
ubuntu@ip-172-26-1-29:/app/swim/workspace/b101$ ls
README.md  SUB1  backend  frontend
```

2. 프론트엔드(Vue.js 3.0)

- Jenkins에서 merge, push event가 발생했을 때, `Execute shell` 을 통해 아래의 command를 수행한다.

```
cd frontend
npm install -g yarn
yarn install
yarn build
```

- AWS EC2에 Nginx (nginx/1.18.0 (Ubuntu)) 를 설치하고 `/etc/nginx/sites-enabled` 폴더의 default 파일에서 환경설정을 한다.


```

server {
    listen 80 default_server;
    listen [::]:80 default_server;

    root /home/ubuntu/deploy/dist; # Front build file location

    index index.html index.htm; # index filename

    server_name _; # server domain

    location / {
        try_files $uri $uri/ /index.html;
    }

    location /api {
        proxy_pass http://3.34.191.232:8000/api/;
        proxy_redirect off;
        charset utf-8;

        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-Nginx-Proxy true;
    }
}

```

- 빌드 후 조치는 `Send build artifacts over SSH` 를 통해 `/home/ubuntu/deploy/dist` 경로로 vue build 파일인 dist directory를 통째로 옮기도록 지정합니다.

3. 백엔드

자바 버전, IDE 버전

```

IntelliJ IDEA 2021.1.3 x64
sts-3.9.14.RELEASE

openjdk version "12" 2019-03-19
OpenJDK Runtime Environment (build 12+32)
OpenJDK 64-Bit Server VM (build 12+32, mixed mode, sharing)

```

배포 스크립트(deploy.sh)

- Jenkins에서 merge, push event가 발생했을 때, `Send build artifacts over SSH` 를 통해 `/home/ubuntu/deploy/server` 경로로 jar파일을 이동시킨다.

- 그 후 deploy.sh 파일을 실행한다.
- Jenkins에서 time out이 발생할 경우 수동으로 deploy.sh파일을 실행한다.

```
#!/bin/bash

cd /home/ubuntu/deploy/server

chmod +x ./recruit-0.0.1-SNAPSHOT.jar

echo "> 현재 구동 중인 애플리케이션 pid 확인"

CURRENT_PID=$(pgrep -f recruit-0.0.1-SNAPSHOT.jar)

echo "현재 구동 중인 애플리케이션 pid : $CURRENT_PID"

if [ -z "$CURRENT_PID" ]; then
    echo "> 현재 구동 중인 애플리케이션이 없으므로 종료하지 않습니다."
else
    echo "> kill -15 $CURRENT_PID"
    kill -15 $CURRENT_PID
    sleep 5
fi

echo "> 새 애플리케이션 배포"

nohup java -jar /home/ubuntu/deploy/server/recruit-0.0.1-SNAPSHOT.jar &
```

4. 데이터베이스 정보 - AWS + Docker + MariaDB

AWS EC2 root 계정

PW: ssafy

MariaDB 계정

ID: root

PW: ssafy

public ip address: 3.34.191.232

- application.properties

```
spring.datasource.driver-class-name=org.mariadb.jdbc.Driver  
spring.datasource.url=jdbc:mariadb://{public-ip}:3306/B101  
spring.datasource.username=root  
spring.datasource.password=비밀번호
```