



AWS EC2 배포(Nginx,Vue,Spring) Jenkins CI/CD

EC2 접속

MobaXterm 설치

(참고) <https://makingrobot.tistory.com/94>

[MobaXterm]

Session → SSH → (Basic SSH settings) Remote host 주소 입력 → (Advanced SSH settings) Use private key 선택 후, 발급받은 .pem 파일 선택 → OK

- 왼쪽에 생성된 session 선택

login as : 에 ubuntu

⇒ ubuntu 계정 접속 가능 (계정을 하나 생성해서 사용하는 것을 추천)

EC2에 설치해야 하는 것

(sudo yum XXX으로 설치하는 방법도 있는데, 안먹혀서 sudo apt-get XXX로 설치 진행함)

▼ openJDK

[설치]

```
sudo apt-get update //업데이트
sudo apt-get install openjdk-8-jdk //JDK 설치(자바 버전 8)
java -version //설치된 JDK 버전 확인
```

▼ Database

! 사용할 디비에 맞는 설치 방법 찾아서 진행할 것

[설치]

```
sudo apt list | grep maria
sudo apt update && sudo apt-get -y upgrade
sudo apt-get install -y mariadb-server
```

[접근 설정 및 기본 명령어]

```
[접근 설정]
mysql -u root -p
(-> 안되면, sudo mysql로 접속)

Password 입력
```

```
[명령어들]
SHOW DATABASES;
use db이름;
SHOW TABLES;
exit; (종료)
```

(참고) <https://ojava.tistory.com/189>

[Springboot DB 설정]

```
[spring - AWS EC2 mariadb 연동]

build.gradle 파일
implementation ('org.mariadb.jdbc:mariadb-java-client')

application.properties 파일
spring.datasource.url=jdbc:mariadb://i5b207.p.ssafy.io:3306/jubging?useSSL=false&useUnicode=true&serverTimezone=Asia/Seoul&allowP
spring.datasource.driver-class-name=org.mariadb.jdbc.Driver
spring.datasource.username=root
spring.datasource.password=jubging2021ssafy

[형식]
spring.datasource.url=jdbc:mariadb://{host}:{port}/{db_name}
spring.datasource.driverClassName=org.mariadb.jdbc.Driver
spring.datasource.username={user_name}
spring.datasource.password={password}
```

(참고) [2번 springboot 설정] 부분 https://xlffm3.github.io/spring%20%20spring%20boot/Docker_Mariadb/

[DBeaver 연동]

```
[DBeaver에서 AWS EC2 mariadb 사용하기]
데이터베이스 탭 -> 새 데이터베이스 연결 -> mariadb 선택
-> (Server 부분) Server Host에 AWS 주소, Port에 설정한 port번호 작성
-> (Authentication부분) User name, password 부분 작성
-> 왼쪽 아래 Test Connection 해보고 잘 되면
-> 완료
```

▼ Nginx

[설치]

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

[Nginx 명령어]

```
//실행 명령어
sudo systemctl start nginx
sudo service nginx start

//중단 명령어
sudo systemctl stop nginx

//재실행 명령어
sudo systemctl restart nginx

//실행 상태 확인 명령어
sudo systemctl status nginx.service
sudo systemctl status nginx
```

```
ubuntu@ip-172-26-8-144:~$ sudo systemctl status nginx.service
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset:
   Active: active (running) since Thu 2021-08-19 20:00:33 KST; 5 days ago
     Docs: man:nginx(8)
   Main PID: 426778 (nginx)
    Tasks: 6 (limit: 19198)
   Memory: 11.2M
   CGroup: /system.slice/nginx.service
           └─426778 nginx: master process /usr/sbin/nginx -g daemon on; maste
              └─426779 nginx: worker process
                 └─426780 nginx: worker process
                    └─426781 nginx: worker process
                       └─426782 nginx: worker process
                          └─426783 nginx: cache manager process
```

→ Active 값 확인하여, nginx 실행 상태 확인

▼ Certbot, letsencrypt

- http 통신을 https로 바꾸기 위해서
- http는 암호화가 안되어 있음 → 해킹의 위험
- https는 http 통신에 SSL 기술을 더한, 보안이 강화된 인터넷 통신 프로토콜
 - HTTP : Hyper Text Transfer Protocol
 - HTTPS : Hyper Text Transfer Protocol over Secure Socket Layer
 - SSL : Secure Socket Layer

[Certbot 설치]

```
sudo apt-get update
sudo apt-get install software-properties-common
sudo add-apt-repository universe
sudo add-apt-repository ppa:certbot/certbot
sudo apt-get update

sudo apt-get install certbot python3-certbot-nginx
```

[인증서 설치 및 Nginx 자동설정]

```
sudo certbot --nginx

1. 도메인 선택
2. http 요청에 대한 설정 -> https로 redirect하기 선택
```

[Nginx 설정에서 확인하기]

```
cd /etc/nginx/sites-available
sudo vim default
```

(참고) certbot 설치 및 Nginx 설정 확인하기

<https://ahzick.tistory.com/entry/NginxSSL인증하기>

- ✓ frontend에서 api url 호출할 때, https://~~~로 바꿔줘야 함.
- ✓ 계속 http로 요청할 경우, **Mixed Content** 에러 발생

[letsencrypt 설치 및 인증서 발급]

```
sudo apt-get update -y & sudo apt-get install letsencrypt -y
sudo systemctl stop nginx
sudo letsencrypt certonly --standalone -d [도메인 네임]
sudo systemctl start nginx
```

(참고) letsencrypt 설치 및 인증서 발급, nginx에 설정하기

<https://shinjongpark.github.io/2020/02/17/AWS-nginx-vue-spring-ssl.html>

▼ Docker

[설치 및 확인]

```
sudo apt update

//필요한 util들 설치
sudo apt install apt-transport-https
sudo apt install ca-certificates
sudo apt install curl
sudo apt install software-properties-common

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable"
sudo apt update

sudo apt install docker-ce
```

(참고) 도커 설치 및 image 받아서 실행해보기

<https://velog.io/@wimes/AWS-EC2에-Docker-설치-및-Dockerfile로-웹서버-구동시키기>

[도커 기본 명령어]

```
docker ps //실행 중인 컨테이너 조회
docker ps -a //모든 컨테이너 조회

docker start [컨테이너 이름 or 컨테이너 아이디] //컨테이너 시작
docker stop [컨테이너 이름 or 컨테이너 아이디] //컨테이너 중지
docker restart [컨테이너 이름] //컨테이너 재시작

docker rm [컨테이너 이름 or 컨테이너 아이디] //컨테이너 삭제
docker rmi [도커 이미지 이름] //도커 이미지 삭제
```

▼ Jenkins

[설치]

```
wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -
sudo sh -c 'echo deb https://pkg.jenkins.io/debian-stable binary/ > \
/etc/apt/sources.list.d/jenkins.list'
sudo apt-get update
sudo apt-get install jenkins
```

[실행 및 상태확인]

```
[실행]
sudo service jenkins start //실행
sudo service jenkins restart //재실행

[상태 확인]
```

```
1. sudo service jenkins status
2. sudo systemctl status jenkins
```

!! 설치 후, http://서버 주소:8080 접속하면 jenkins 화면을 볼 수 있음 하지만 접속 포트(현재 8080)를 바꿔줘야 함

[포트 변경하기]

```
[jenkins 포트 변경하기]
1. /etc/default/ 폴더 안에 있는 jenkins 파일을 수정모드로 열기
sudo vi /etc/default/jenkins

2. 포트 번호 설정하기
HTTP_PORT=9999 (사용할 포트 번호 입력)

3. 저장
esc 누르고 -> :wq 입력

4. jenkins 재실행
sudo systemctl restart jenkins
```

[Jenkins 비밀번호 확인하기]

```
[Jenkins 계정 비밀번호]
1. 비밀번호가 적혀있는 파일 열기
sudo vim /var/lib/jenkins/secrets/initialAdminPassword

2. 나오는 PW 저장해두기
```

[Jenkins 접속하기]

- http://(hostip or hostName):포트번호 으로 접속

```
1. 계정 : admin / 아까 복사해둔 PW 입력
2. Install suggested plugins 선택
3. 설치
```

(참고) 젠킨스 설치 및 포트 설정

[https://imbf.github.io/devops/2020/11/26/Install-Jenkins-in-Ubuntu\(18.04\).html](https://imbf.github.io/devops/2020/11/26/Install-Jenkins-in-Ubuntu(18.04).html)

도커파일 작성하기

- springboot dockerfile

```
FROM openjdk:8-jdk-alpine
EXPOSE 7777
ARG JAR_FILE=/build/libs/*.jar
COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java", "-jar", "/app.jar"]
```

FROM

- 베이스 이미지, 빌드 할 이미지가 어떤 이미지를 기반으로 하고 있는지

EXPOSE

- 도커 컨테이너 내부에서 몇번 포트로 받을 건지를 의미

- 컨테이너 내부 포트를 7777로 설정

COPY

- 파일을 이미지에 추가하는 것을 의미
- JAR_FILE에 빌드된 .jar의 경로를 설정하고, (build/libs/*.jar 파일)
- COPY \${JAR_FILE} app.jar을 통해서 {JAR_FILE}을 app.jar에 복사하겠다는 의미

ENTRYPOINT

- 컨테이너가 생성, 시작될 때 실행되는 명령어
- java -jar app.jar 명령어가 실행되어 복사해두었던 jar파일을 실행하여 SpringBoot 프로젝트를 실행시킨다는 의미가 됨

(참고) <https://velog.io/@haeny01/AWS-EC2-X-Docker-Spring-Boot-프로젝트-올리기>

• vue dockerfile

```
# base image
FROM node:12.2.0-alpine

COPY package*.json ./

RUN npm install
RUN npm i webstomp-client sockjs-client
COPY ./ .
RUN npm run build

RUN npm run-script build

#RUN mkdir /app

EXPOSE 3000

CMD ["npm", "run", "serve"]
```

(참고) <https://kdeon.tistory.com/6>

• 도커 빌드하여 이미지 만들기

dockerfile이 있는 위치에서

```
docker build -t 도커이미지 이름 .
(뒤에 반드시 . 찍어야 함)

ex)
docker build -t jubging-be .
```

실행하면 설정한 도커이미지 이름으로 도커이미지가 생성됨

• 도커 실행하기

```
docker run -d -p {EC2 포트}:{도커포트} --name [컨테이너로 사용할 이름] [이미지이름]

ex)
docker run -d -p 3000:3000 --name jubging-fe jubging-fe
docker run -d -p 7777:7777 --name jubging-be jubging-be
```

-p 7777:7777

- 👉 EC2의 7777포트를 도커의 7777포트와 연결시켜주겠다는 의미
- 👉 EC2의 7777포트로 접근하게 되면, 도커의 7777포트로 연결되어 springboot 프로젝트에 도달

👉 dockerfile에서 EXPOSE를 7777로 설정했기 때문에 -p 7777:7777로 함

빌드

• Springboot 빌드하기

[Docker 없이 배포하는 방법]

```
1. 백엔드 폴더 이동 .jar 파일 생성
cd backend
chmod 777 ./gradlew
./gradlew build

2. .jar 파일이 만들어진 곳으로 이동
cd build/libs
ls (하면 현재 디렉토리에 있는 파일 목록들 확인 가능)

3. 실행하기
java -jar 생성된jar파일이름.jar
```

[Docker 파일로 배포하는 방법]

```
1. 백엔드 폴더 이동 .jar 파일 생성
cd backend
chmod 777 ./gradlew
./gradlew build

2. dockerfile이 있는 위치로 이동
cd backend

3. 도커 이미지 빌드 및 실행
docker build -t jubging-be .
docker run -d -p 7777:7777 --name jubging-be jubging-be
```

• Vue 빌드하기

[Docker 없이 빌드하기]

```
1. 프론트엔드 폴더 이동
cd frontend

2. 빌드
npm install
sudo npm run build

-> 빌드를 하고 나면 frontend 디렉토리 안에 dist라는 빌드 폴더 생성됨
-> 이 빌드된 폴더를 nginx에서 설정한 위치로 옮겨야함

3. 빌드 폴더 이동하기

기존에 옮겨놓은 dist폴더가 있을 경우
-> cd /var/www/html (그 위치로 이동)
-> sudo rm -rf dist (원래 있던 dist 폴더를 삭제)

새로 빌드하여 만들어진 dist 폴더가 있는 곳으로 이동
cd frontend
sudo mv dist /var/www/html (dist 폴더를 /var/www/html로 이동)

-----
nginx에서 vue 빌드 폴더 설정을 /var/www/html/dist로 두었기 때문에
/var/www/html 아래에 dist 폴더를 이동시킨 것
```

[Docker에 Vue.js 올리기]

<https://kdeon.tistory.com/6>

<https://withhamit.tistory.com/159>

Nginx 설정

/etc/nginx/sites-available 폴더 안의, default 파일

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

    server_name i5b207.p.ssafy.io;

    return 301 https://$server_name$request_uri;

    index index.html index.htm ;
}
```

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

    ssl on;
    ssl_certificate /home/ubuntu/zeross/certificate.crt;
    ssl_certificate_key /home/ubuntu/zeross/private_jubging.key;

    root /var/www/html/dist;

    index index.html index.htm ;

    server_name i5b207.p.ssafy.io;

    location / {
        proxy_cache disk-cache;
        proxy_cache_valid any 1m;
        proxy_ignore_headers Cache-Control Expires;
        #alias /var/www/html/frontend/;
        try_files $uri $uri/ /index.html;
    }

    location /api {

        rewrite /api/(.*) /api/$1 break;

        proxy_pass http://localhost:7777;
        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;
    }
}
```

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

    server_name i5b207.p.ssafy.io;
```



```

    if($host = i5b207.p.ssafy.io){
        return 301 https://$server_name$request_uri;
    } //certbot 설정

    index index.html index.htm ;
}

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

    //ssl부분도 certbot 설정하면 자동으로 인증서 경로 지정하는 파라미터가 생김
    ssl on;
    ssl_certificate /인증서 위치 경로/certificate.crt;
    ssl_certificate_key /인증서 위치 경로/private_jubgung.key;

    root /var/www/html/dist;

    index index.html index.htm ;

    server_name i5b207.p.ssafy.io;

    location / {
        proxy_cache disk-cache;
        proxy_cache_valid any 1m;
        proxy_ignore_headers Cache-Control Expires;
        try_files $uri $uri/ /index.html;
    }

    location /api {

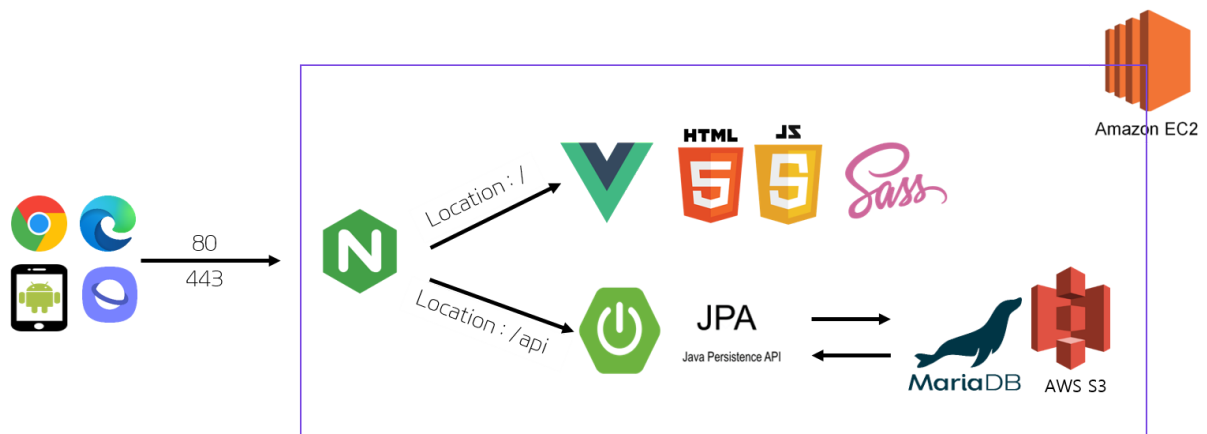
        rewrite /api/(.*) /api/$1 break;

        proxy_pass http://localhost:7777;
        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;
    }
}

```

[Nginx 구조 설명]



사용자가 80/443으로 요청을 보내면 Nginx가 주소에 따라 매핑해줌

location / {}

```

https://i5b207.p.ssafy.io/ 일 경우 → frontend
https://i5b207.p.ssafy.io/api/ 일 경우 → backend

backend 요청일 경우,
proxy_pass 주소인 http://localhost:7777 (springboot 백엔드 주소)로 프록시됨
-> 그래서 프론트에서 api baseUrl을 [https://i5b207.p.ssafy.io/api/~] 로 설정
-> 백엔드에는 [server.servlet.context-path=/api]를 추가

```

```
api 호출할 때, api를 추가를 안하면
frontend, backend 모두
location / {}
이렇게 되는데, 이러면 같은 주소로 2개를 생성하는 것이 되어서 nginx가 실행 안됨
```

+ springboot - application.properties 설정 추가

```
server.servlet.context-path=/api
server.port = 7777
```

자동배포 - Gitlab 연동

[jenkins에 필요한 플러그인 설치하기]

The screenshot shows the Jenkins dashboard with the 'Jenkins 관리' (Jenkins Management) section. The '플러그인 관리' (Plugin Management) link is circled in blue. Below the screenshot, a list of plugins to install is shown: Git plugin, GitLab Plugin, and Publish over SSH.

```
Git plugin
GitLab Plugin
Publish over SSH
```

(그 외 필요한 플러그인은 찾아서 설치)

[AWS Jenkins 연동하기]

Jenkins

검색

admin

로그아웃

Dashboard

새로운 Item

사람

빌드 기록

프로젝트 연관 관계

파일 링거프린트 확인

Jenkins 관리

My Views

Lockable Resources

New View

빌드 대기 목록

빌드 실행 상태

1 대기 중

2 대기 중

Jenkins 관리

시스템 설정

환경변수 및 경로 정보등을 설정합니다.

노드 관리

Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

Global Tool Configuration

Configure tools, their locations and automatic installers.

플러그인 관리

Jenkins의 기능을 확장하기 위한 플러그인을 추가, 제거, 사용, 미사용으로 설정할 수 있습니다.
업데이트 가능함

Configure Global Security

Secure Jenkins; define who is allowed to access/use the system.

Manage Credentials

Configure credentials

Configure Credential Providers

Configure the credential providers and types

Manage Users

Create/delete/modify users that can log in to this Jenkins

In-process Script Approval


Allows a Jenkins administrator to review proposed scripts (written e.g. in Groovy) which run inside the Jenkins process and so could bypass security restrictions.

시스템 설정에서 [Publish over SSH] 설정하기

Publish over SSH

Jenkins SSH Key

Passphrase

 Concealed

Path to key

Key

```
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAIZVlq+mVke4z/RyugPKIYnO2NpgxaZG0calVfXy/KK/RrNa
EhiWp0xGpNgtMtlb/qDDrMKfTbTeHoUZkgDTJDKOZ5ZtTOhmOMllaZuThyldH6ol
z1ficcJn4RExTSR5FvNlskogw63WXg5z+lh2fzB6UF+qQ3BQOY3qxdTHbnZfB
G+6C+9We5lRkqJCNiQLMP93BvFisuWFJTcRm8b/NglZRfY6uQqSW42p/Q7a0Lr/L
Ej7AFjrextXa5kH7AtdfwC8q/mziPnzr4upaEhUbfQlQfQXOdJTL3dqVPeqztrQ
mQUa2DJeHO7O+OpVy5kZ27RVweJagRrIZ5Q8M9QIDAQABAOlBAEngr4Zop2awMKcE
pGUa2kNg/Lg9ou37YZ7Z7W63OntSlexVSHcHjeCy2z6b9UJ7AWmgP4IVZ6upWMra
Efb6Uu3ioCqpMPJ7FYGS03TVUde8EY2CESqFlrwBuCyAwUjlasXbEK8tZWDYxrh
Jix5p8U4id/0sS9JS6M+U5Imr2EhW7NjnhzQiN6jCcULpVLWv9+Du8xj5OME/PW
n8ZvdZMgDjXjkwabRHAL4eKvmFrUgy2fV7SxpvXV9lu3/XKNUoaDuyhY6xSTGW7
B6RXujpsTmjc2MeJ/CkqjhnLsSK9bWovzBHT1WTDY/NGFnlO7CHLF+YfzxeGo7u
7ai+/OECgYEAA4GuZfqaBZgAe6toTAyhRvmhJZdPharOvKwVlpFbWPOZ+OXQGXkz
EP3G1FaRX1/xN663vCyQH3oWZH5oSW9bO1ZKB5ReUGHMHwhvYmsbBZvacwj/iideU
wopsu5jtot4M3IFfehY/vgbT2a58N7L6ekOp/wtSNlaOglqGpfaPQsCgYEAqHr
Ea7jYzjd/fXchFw2DltklDY0ASf8STZT1CUOobFFxLqCB1/VjBZ2B6vReDyDyk
Y0Kxz7Lfm73vYtvuUK9/iUbQoXtp7U/m6ZaD5nluX0AZWNl6QeS45tYD43falp
It8/65b/SyuzgWBcqZeUc2VG0clEQO3T5KKGHf8CgYEAuMC8R+DjzHAzA8LuPf42
dZujOTpabB5rTSbaSu91lWB0lPKVBYoUWVujEXIHec6JZf7ncfn52lAl6UilKfEl
eU8LePvzPreWRWSBSMJxu9MwsFYpQLMa26Fb7ID0uc02J142fOwfD6Cv0wln8Hdq
xjHAIBj6KuWm9JSFjkkPOlscGyEAhSqm9X8C8W3xOoFF33rdqULESf8z5myRFRMa
YXmEpvd9uM8rMlRc8SLyicZjl+4VKBHDzb/AW7Ni9f4utH2mA3/jjVTSSIGTbz5x
7PzP7rsR4UhgJcCESXBiwAWIAauszPzgk9md3jRS+4yYpFDbIHQvJdwXICSugunZ
Y+c/OmsCgYBqeuPPZfd+tk29WhqkcGcdhuWCw9N+XfgztCmh1R9Qu4eqPuHxVlJN
jHxZmxl+oXe4edY+Rhi5lWfxc7i3WuHfsOUfKQd0GSBw9WnJOBFZfZug2JUW/Kp3
Elr1N2ljpbPaxEKuKyzH7vFn8ZShDNTUbY7/Enxy7Eoyk/k+VIRqW==
-----END RSA PRIVATE KEY-----
```

☐ Disable exec

SSH Servers

 SSH Server

Name

aws-ec2-deploy

Hostname

i5b207.p.ssafy.io

Username

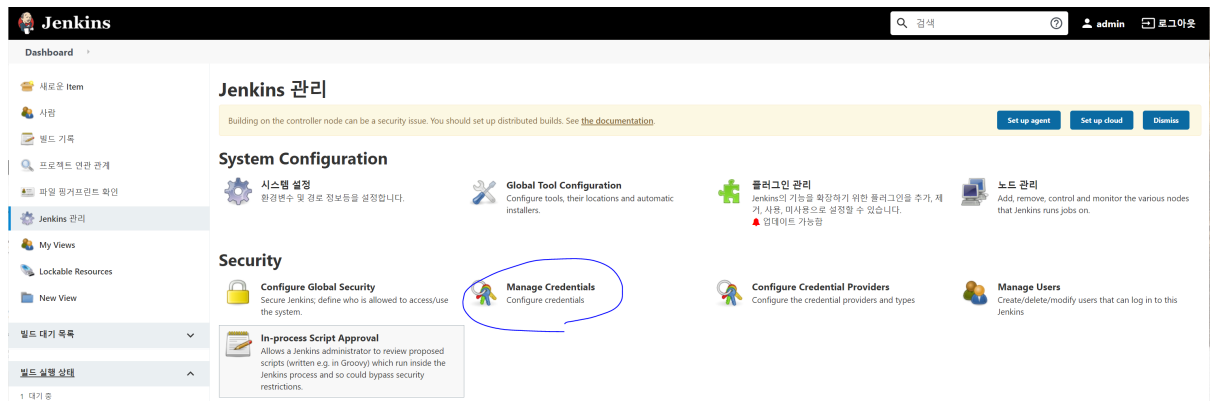
ubuntu

Remote Directory

/home/ubuntu/deploy

Name : SSH server name을 지정
Hostname : 실제로 접속할 원격 서버 ip, 접속 경로 (i5b207.p.ssafy.io)
Username : 접속할 원격 서버의 user 이름
Remote Directory : 원격 서버에서 접속하여 작업을 하게 되는 디렉토리
-> 젠킨스로 빌드하고 나면 해당 경로에 폴더, 파일이 올라간 것을 확인할 수 있음

[Jenkins에 gitlab 계정 Credentials 생성하기]



Stores scoped to Jenkins



Kind

Username with password

Scope

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

Username

☐ Treat username as secret

Password

ID

Description

OK

username : 깃랩아이디
 password : 깃랩 패스워드
 ID : 임의로 사용할 ID (중요하지 않은 듯)
 description : 해당 설정에 대한 설명 (생성된 credentials의 이름처럼 씀)

lee.sujeong10111/***** (jubging6)

description에 jubging6을 쓰면 credentials 선택시 이렇게 보임

[배포 설정하기]

새로운 item → Freestyle project 선택

✓ 소스코드 관리 설정

소스 코드 관리

☐ None

☒ Git

Repositories

Repository URL

https://lab.ssafy.com/s05-webmobile2-sub3/S05P13B207.git

Credentials

lee.sujeong10111/***** (jubqinq6)

Add

고급...

Add Repository

Branches to build

Branch Specifier (blank for 'any')

*/develop

Add Branch

Repository browser

(자동)

Additional Behaviours

Add

Git 선택 → 깃랩 주소 입력 → 생성한 credentials 선택 → branch 선택

*/develop ⇒ develop 브랜치를 빌드

✓ 빌드 유발 설정

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://i5b207.p.ssafy.io:9999/project/Jubging> ?

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...

☐ GitHub hook trigger for GITScm polling ?

☐ Poll SCM ?

✓ Build

Build

Execute shell

Command

```
ls -al
ls
cd backend
echo "springboot jar build 시작"
chmod 777 ./gradlew
./gradlew build
echo "springboot 빌드 끝"

cd /var/lib/jenkins/workspace/Jubging

cd backend
docker ps -q --filter name=jubging-be | grep -q . && docker rm -f $(docker ps -aq --filter name=jubging-be)

docker build --no-cache=false -t jubging-be .
docker run -d -p 7777:7777 --name jubging-be jubging-be

cd
cd /var/lib/jenkins/workspace/Jubging
cd frontend
docker ps -q --filter name=jubging-fe | grep -q . && docker rm -f $(docker ps -aq --filter name=jubging-fe )
docker build --no-cache=false -t jubging-fe .

docker run -d -p 3000:3000 --name jubging-fe jubging-fe
ls

npm install
npm i webstomp-client sockjs-client
sudo npm run build

cd
cd /var/www/html
ls
sudo rm -rf dist

cd
cd /var/lib/jenkins/workspace/Jubging/frontend
ls
sudo mv dist /var/www/html

sudo systemctl stop nginx
sudo systemctl start nginx
```

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

빌드 요청이 들어오게 되면 실행할 명령어들

(참고) <https://velog.io/@sa1341/Jenkins에서-EC2로-배포하기>

[Gitlab Webhook 설정]

Gitlab → Settings → Webhooks

Webhook

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

☐ **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

☐ **Wiki page events**

URL is triggered when a wiki page is created or updated

☐ **Deployment events**

URL is triggered when a deployment starts, finishes, fails, or is canceled

☐ **Feature flag events**

URL is triggered when a feature flag is turned on or off

☐ **Releases events**

URL is triggered when a release is created or updated

SSL verification

☒ **Enable SSL verification**

Save changes

Test ▾

Delete

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://i5b207.p.ssafy.io:9999/project/Jubqing> ?

Secret token

Generate

URL : Jenkins에서 빌드 유발에 있는 URL 입력

secret token : 빌드유발 -> 고급 -> (secret token) generate 눌러서 생긴 token 값 입력

Trigger : 내가 원하는 설정 선택

🕒 Hook executed successfully: HTTP 200



trigger까지 설정하고 test버튼에서 이벤트 발생시켰을 때, 위처럼 successfully 문구 뜨면 성공

> 혹시 fail 뜰 경우

! jenkins에서 secret token 생성하고 apply -> save 했는지 확인해 볼 것