



# 포팅매뉴얼

📅 날짜	@2023년 10월 6일
☰ 태그	프로젝트

## 개발 환경

MariaDB : 8.0.34

Nginx : 1.18.0

redis : 7.2.1

Docker : 24.0.6

Docker Compose : v2.21.0

Python : 3.7.5

node : v20.5.1

npm : 9.8.0

react : 18.2.0

## 배포



반드시 단계별로 실행! 에러난다고 넘어가지 말고 해결하고 넘어가기

## ssh 연결

### Ubuntu 서버로 들어오기

본인은 method 2 로 했다.

#### ▼ method 1

.pem 파일이 있는 디렉토리에서 gitbash 열어서 실행

```
ssh -i {pemkey} {ec2아이디}@{주소}
```

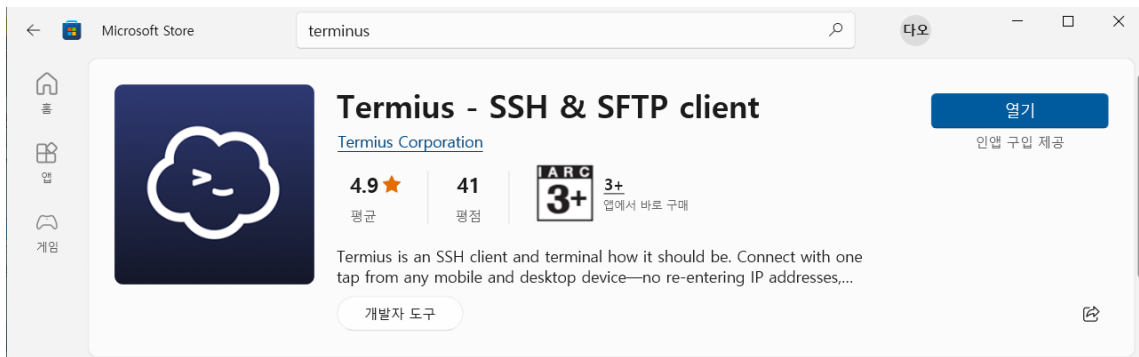
- pemkey : aws에서 다운받은 key의 이름
- ec2아이디 : ec2 사용자 아이디
- 주소: 사이트 주소

ex) `ssh -i J9B201T.pem ubuntu@j9b201.p.ssafy.io`

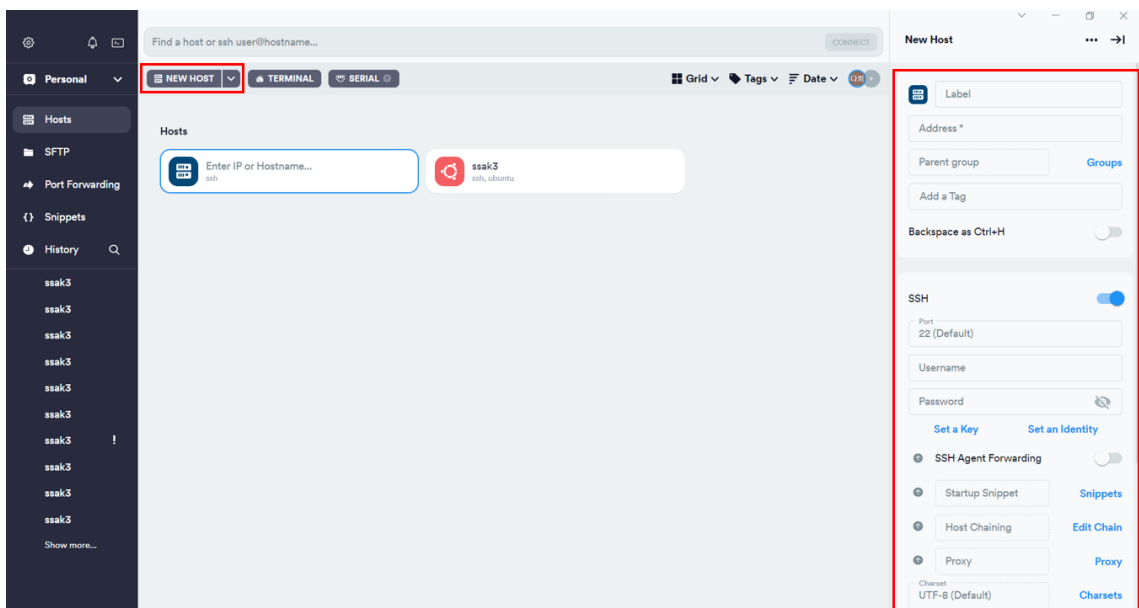
#### ▼ method 2

terminus 로 실행

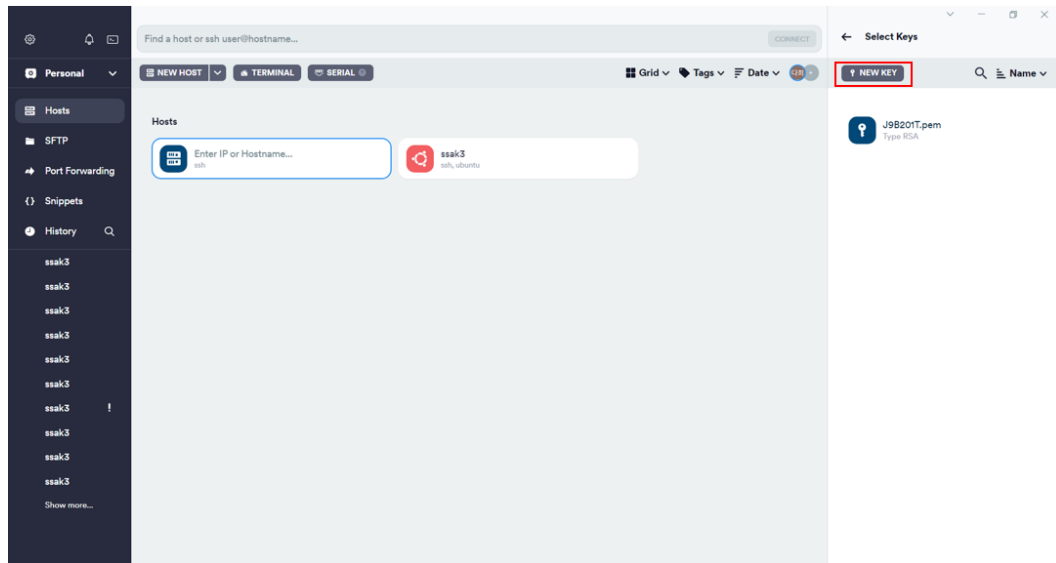
## 1. Microsoft Store 에서 terminus 앱 설치



## 2. Host 생성



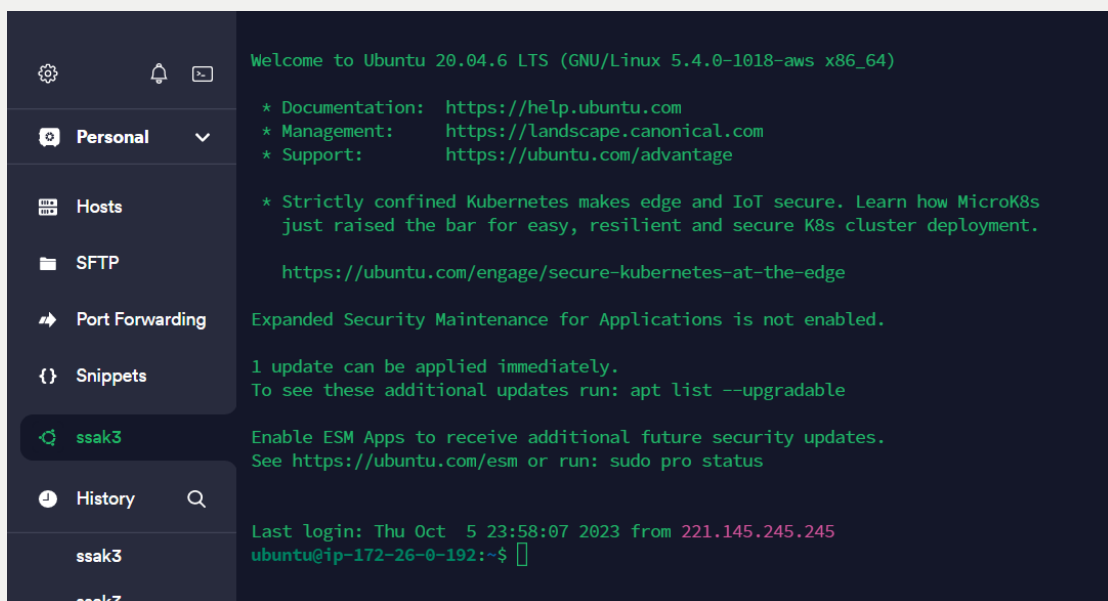
- Label : 서버 별칭
  - ssak3 : 우리 프로젝트 이름)
- Address : 서버 주소
  - j9b201.p.ssafy.io
- ▼ SSH
  - Port : 22
    - 싸피 프로젝트의 경우 초기 설정시 22번 포트만 활성화 됨
  - Username: ec2 사용자 아이디
    - ubuntu
- ▼ Set a key : pem.key 등록



- import from key file
  - 다운로드 받은 .pem 파일을 드래그 앤 드롭으로 올리면 자동 등록
  - save

## Docker, Docker Compose 설치

💡 모든 설치는 본인 컴퓨터에 직접 설치하는 것이 아닌 우분투 서버내에 설치한다.  
즉, terminus로 등록한 서버로 접속한 후에 shell에서 실행해야 한다.



## Install Docker Desktop on Ubuntu

Learn how to install, launch and upgrade Docker Desktop on Ubuntu. This quick guide will cover prerequisites, installation methods, and more.

 <https://docs.docker.com/desktop/install/ubuntu/>

### ▼ 도커 설치 매뉴얼

#### 1. 우분투 시스템 패키지 업데이트

```
sudo apt-get update
```

#### 2. 필요한 패키지 설치

```
sudo apt-get install ca-certificates curl gnupg lsb-release
```

#### 3. 폴더 생성

```
mkdir -p /etc/apt/keyrings
```

#### 4. Docker의 공식 GPG키를 추가

```
curl -fsSL https://download.docker.com/linux/debian/gpg | gpg --dearmor -o /etc/apt/keyrings/docker.gpg
echo \
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/debian \
$(lsb_release -cs) stable" | tee /etc/apt/sources.list.d/docker.list > /dev/null
```

#### 5. 시스템 패키지 업데이트

```
sudo apt-get update
```

#### 6. Docker 설치

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin docker-compose
```

#### 7. Docker 설치 확인

- 도커 실행상태 확인

```
sudo systemctl status docker
```

- 도커 실행

```
sudo docker run hello-world
```

#### 8. Docker 버전 확인

```
sudo docker -v
```

- 24.0.4

#### 9. Docker Compose 버전 확인

```
sudo docker compose version
```

- 2.19.1



`sudo docker ps` : process status → 현재 실행중인 컨테이너를 보여준다.

`sudo docker ps -a` : 실행 중지된 컨테이너까지 모두 다 보여준다.

## Docker 내에 MariaDB설치

### EC2를 통한 서버배포 - Docker에 DB 설치

앞선 글은 EC2와 RDS를 연결하여 EC2는 리액트 프론트 서버를, 데이터 베이스는 RDS에서 사용할 수 있다. RDS 사용이 어려울 경우, EC2에서 Docker를 설치하여 Database를 사용할 수 있다. AWS EC2에 도커(Docker)를 설치 - Git Bash에서 ssh로 접속하여 진행하였다. 1. 최신 버전으로 패키지 업데이트 `sudo apt-get update` 2. 도커 다

 <https://5sangs.tistory.com/255>



## ▼ Docker-Compose로 DB 올리기

### 1. 새로운 파일 생성

```
mkdir build
```

- mkdir : make directory 폴더 생성

```
cd build
```

- cd : change directory 경로 이동

```
vi docker-compose.yml
```

- vi 명령어를 통해 docker-compose.yml 생성

```
version: '3'

services:
  mariadb:
    image: mariadb:latest
    container_name: ssak3_db
    environment:
      MYSQL_DATABASE: ssak3
      MYSQL_ROOT_PASSWORD: 2201
      TZ: Asia/Seoul
    command:
      - --lower_case_table_names=1
      - --character-set-server=utf8mb4
      - --collation-server=utf8mb4_unicode_ci
    volumes:
      - ./db:/var/lib/mariadb
    ports:
      - 2714:3306
    restart: always
```

### 2. docker-compose.yml 실행

```
sudo docker-compose up -d
```

## ▼ MySQL 접속

```
sudo docker exec -it {container_name} bash
```

- container\_name: ssak3\_db

```
mysql -u root -p
```

- mysql : command not found가 뜬다면
  - MySQL Ubuntu 설치

```
apt-get update
apt-get install mysql-client
```
- 비밀번호는 내가 설정한 root password 치면된다.
  - 화면에 비번 안나오는건 그냥 설정상이니깐 강 치면됨

## TEST

```
CREATE USER 'abc'@'%' IDENTIFIED BY '1234';
```

```
GRANT ALL PRIVILEGES ON *.* TO 'abc'@'%';
```

```
flush privileges;
```

```
quit
```

## MariaDB Workbench에서 접속

Connection 등록에 서버URL, 포트번호, 위에서 설정한 Hostname을 입력한다.

**Test Connection** 을 누르고, 위에서 설정한 비밀번호를 입력하면 연결이 성공하였음을 확인할 수 있다.

## SSL 발급 받기

### ▼ Port 열기

ufw : Ubuntu - Fire - Wall : 우분투 방화벽 설정

ubuntu 서버 내에 접속 후에 확인한다.

#### 1. ufw 상태 확인

```
sudo ufw status
```

- inactive 아니면 active 상태일 꺼임

#### 2. 사용할 포트 허용하기

```
sudo ufw allow 22
```

- 22번 포트 허용 - ufw inactive 상태

#### 3. ufw 활성화 하기

```
sudo ufw enable
```

```
Command may disrupt existing ssh connections. Proceed with operation (y|n)? y
```

#### 4. ufw 상태 및 등록된 rule 확인하기

```
sudo ufw status numbered
```

- 이런식으로 뜨면 성공

```
Status: active

      To      Action     From
      --      -
[ 1] 22      ALLOW IN   Anywhere
[ 2] 22 (v6)  ALLOW IN   Anywhere (v6)
```

### ▼ SSL 적용을 위해 80, 443 포트 열기

등록

```
sudo ufw allow 80
```

```
sudo ufw enable
```

```
sudo ufw status numbered
```

- 똑같이 443도 해주세요

삭제 (ex 80번 포트 삭제)

```
sudo ufw status numbered
```

```
Status: active

      To      Action     From
      --      -
[ 1] 22      ALLOW IN   Anywhere
[ 2] 80      ALLOW IN   Anywhere
[ 3] 22 (v6)  ALLOW IN   Anywhere (v6)
[ 4] 80 (v6)  ALLOW IN   Anywhere (v6)
```

```
sudo ufw delete 4
```

```
sudo ufw delete 2
```

```
sudo ufw status numbered
```

```
sudo ufw enable
```

### ▼ Let's Encrypt

```
sudo apt get install letsencrypt
```

```
sudo letsencrypt certonly standalone d {도메인 주소}
```

- 도메인 주소 : j9b201.p.ssafy.io

```
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Plugins selected: Authenticator standalone, Installer None
Enter email address (used for urgent renewal and security notices)
```

```
(Enter 'c' to cancel): dhekgml1234@gmail.com

- - - - -
Please read the Terms of Service at
https://letsencrypt.org/documents/LE-SA-v1.2-November-15-2017.pdf. You must
agree in order to register with the ACME server. Do you agree?
- - - - -
(Y)es/(N)o: Y

- - - - -
Would you be willing, once your first certificate is successfully issued, to

share your email address with the Electronic Frontier Foundation, a founding
partner of the Let's Encrypt project and the non-profit organization that
develops Certbot? We'd like to send you email about our work encrypting the web,
EFF news, campaigns, and ways to support digital freedom.
- - - - -
(Y)es/(N)o: N

IMPORTANT NOTES:
- Congratulations! Your certificate and chain have been saved at:
  /etc/letsencrypt/live/j9b201.p.ssafy.io/fullchain.pem <- {2} 발급된 인증서 경로
  Your key file has been saved at:
  /etc/letsencrypt/live/j9b201.p.ssafy.io/privkey.pem <- {2} 발급된 인증서 경로
  Your certificate will expire on 2021-05-16. To obtain a new or
  tweaked version of this certificate in the future, simply run
  certbot again. To non-interactively renew *all* of your
  certificates, run "certbot renew"
- If you like Certbot, please consider supporting our work by:

    Donating to ISRG / Let's Encrypt:  https://letsencrypt.org/donate
    Donating to EFF:                  https://eff.org/donate-le
```

## 인증서 경로

### 1. ssl\_certificate

```
/etc/letsencrypt/live/ 도메인이름 /fullchain.pem
```

### 2. ssl\_certificate\_key

```
/etc/letsencrypt/live/ 도메인이름 /privkey.pem
```

## Jenkins Docker-compose로 컨테이너 생성

### ▼ Jenkins 설치

#### 1. 아까 생성한 build폴더의 docker-compose 파일 수정

```
cd build
```

```
sudo vi docker-compose.yml
```

#### 2. docker-compose.yml 수정

코드 추가

```
version: '3'

services:
  mariadb:
    image: mariadb:latest
    container_name: ssak3_db
    environment:
      MYSQL_DATABASE: ssak3
      MYSQL_ROOT_PASSWORD: 2201
      TZ: Asia/Seoul
    command:
      - --lower_case_table_names=1
      - --character-set-server=utf8mb4
      - --collation-server=utf8mb4_unicode_ci
    volumes:
      - ./db:/var/lib/mariadb
    ports:
      - 2714:3306
    restart: always

jenkins:
```

```
image: jenkins/jenkins:lts
container_name: jenkins
volumes:
  - /var/run/docker.sock:/var/run/docker.sock
  - /jenkins:/var/jenkins_home
ports:
  - "9090:8080"
user: root
```

### 3. 컨테이너 올리기

```
sudo docker-compose up -d
```

```
sudo docker ps
```

jenkins 가 뜬다면 설치 성공

## ▼ Jenkins 내에 Docker, Docker Compose 설치

위에서 Docker, Docker Compose 설치 한것과 동일함 하지만 jenkins 내부로 들어와야 함

### 1. jenkins 내부 접속

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

### 2. 내부 코드 실행

```
apt-get update

apt-get install ca-certificates curl gnupg lsb-release

mkdir -p /etc/apt/keyrings

curl -fsSL https://download.docker.com/linux/debian/gpg | gpg --dearmor -o /etc/apt/keyrings/docker.gpg
echo \
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/debian \
$(lsb_release -cs) stable" | tee /etc/apt/sources.list.d/docker.list > /dev/null

apt-get update

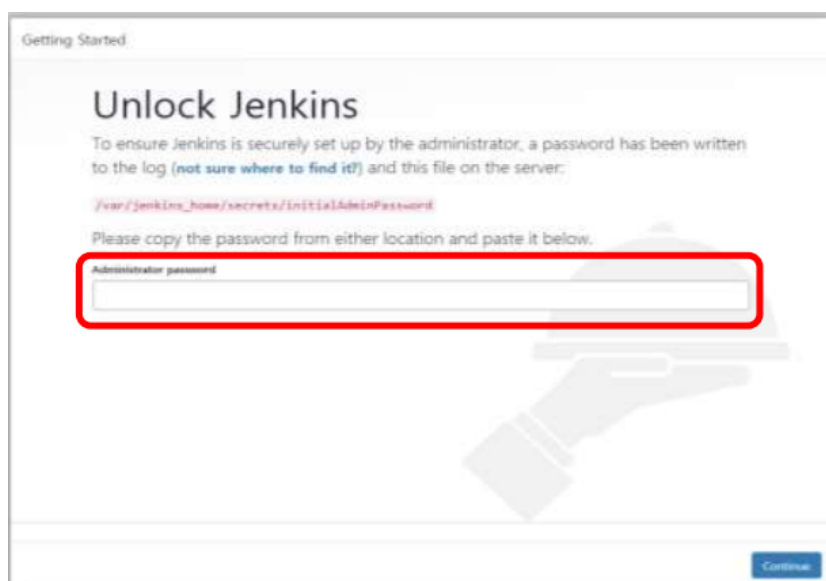
apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin docker-compose
```

## ▼ Jenkins 접속

### 1. 포트 9090열기

- SSL설정할때 했던 port여는 방식으로 똑같이 9090 열어주기

### 2. 인터넷으로 `http://{도메인주소}:9090` 으로 jenkins 접속



administrator password 확인

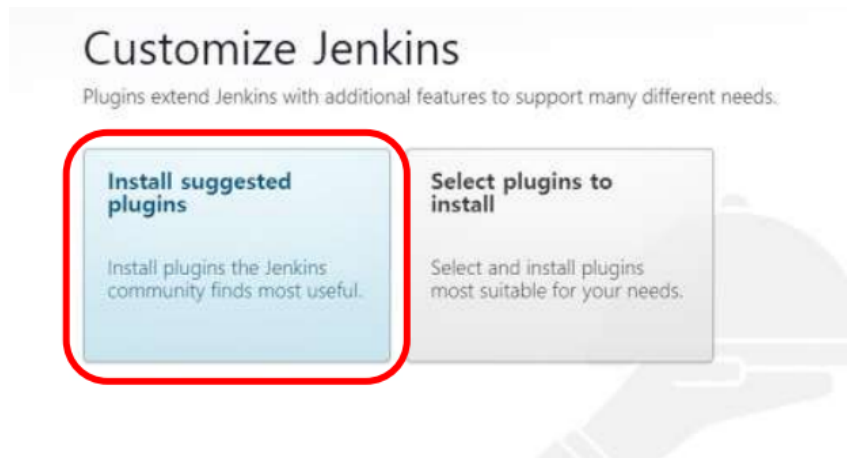


ubuntu 접속후 password 확인하기위해 아래 입력

```
docker exec {CONTAINER_NAME} cat /var/jenkins_home/secrets/initialAdminPassword
```

- CONTAINER\_NAME : jenkins
- password : fe64247b4187483a81844809e0523b31

### 3. Plugin 설치



### 4. Create First Admin User 설정

- Username : 계정명
  - ssak3
- password : 비밀번호
  - b201\_9090 (아무거나 어렵게 설정하세요)
- fullname: 이름 설정
  - admin
- email : 이메일 설정
  - dhekgml1234@gmail.com

### 5. Instance Configuration 입력

- http://{도메인 주소}:9090 입력  
`http://j9b201.p.ssafy.io:9090/b201`

### 6. Plugin 추가

- GitLab
- GenericWebhook Trigger
- GitLab API
- GitLab Authentication
- Docker
- Docker Pipeline
- Docker Common
- Docker API

**Install without restart 누르기**

**메인 페이지로 돌아가기**

### 7. Credentials 설정

Dashboard > ManageJenkins > Credentials > global

## Add Credentials

- 방식은 GitLab API token 으로 설정
1. lab.ssafy.com 접속
  2. 우측 상단 프로필 > Edit Profile
  3. 좌측메뉴에서 Access Tokens
  4. Token name 설정 아무거나
  5. Expiration date 설정 만료날자  
프로젝트 끝날때 까지 넉넉하게 잡기
  6. 체크박스 다 선택 후 Create personal access token 클릭하기
  7. Your new personal access token 저장  
Token : RZE9qWZsutTxTMkPXk8\_
8. Jenkins Gitlab 연동
- Dashboard > ManageJenkins > System

## GitLab

☒ Enable authentication for '/project' end-point ?

GitLab connections

Connection name ?

A name for the connection

ssak3

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

Advanced

Test Connection

- 이름 설정
- URL 설정  
`https://lab.ssafy.com/`

- Credentials  
아까 만든 Credential 등록

Test Connection 눌러서 200OK 뜨면 성공

## 9. Jenkins Project 생성

새로운 아이템 추가를 통해 Pipeline 선택  
아까 만들어놓은 connection 연결

GitLab Connection

ssak3

☐ Use alternative credential

☐ Pipeline speed/durability override ?

☐ Preserve stashes from completed builds ?

☐ This project is parameterized ?

☐ Throttle builds ?

## 10. Webhook 설정

a. 스크롤 해서 내려서 Build Trigger 설정

**Build Triggers**

☐ Build after other projects are built ?

☐ Build periodically ?

☒ Build when a change is pushed to GitLab. GitLab webhook URL: <http://j9b201.p.ssafy.io:9090/project/ssak3> ?

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

Advanced

b. Build when a change is pushed to GitLab. GitLab webhook URL: <http://j9b201.p.ssafy.io:9090/project/ssak3>

에서 URL 다음 주소 복사

<http://j9b201.p.ssafy.io:9090/project/ssak3>

c. 체크 한 후, Advanced 클릭

d. Secret token > Generate 클릭

- token : b18db9d0b9ef8c4d4b803932ae9f7d85

e. [lab.ssafy.com](http://lab.ssafy.com) 접속

코드 올리는 git에 접속

좌측 메뉴에서 Settings > Webhooks

- URL : 아까 복사한 주소 등록  
`http://j9b201.p.ssafy.io:9090/project/ssak3`
- Secret token 입력  
`b18db9d0b9ef8c4d4b803932ae9f7d85`
- Trigger 체크
  - Push event, Tag push events, Merge request events
- Add webhook
- Test > Push events
  - 200 ok 뜨면 webhook 성공

#### 11. Pipeline 설정

- Definition  
Pipeline script from SCM
- SCM  
Git
- Repository URL  
git 주소 입력 (git clone 할때 쓰는 거랑 같은것)  
`https://lab.ssafy.com/s09-mobility-smarthome-sub2/S09P22B01.git`
- Credentials 생성
  1. Add 누르기
  2. Kind - Username with password  
주로 싸피에서 처음 준 git 계정 아이디 비번입력
    1. Username : dhekgml1234@gmail.com
    2. Password : dhekgml1234@gmail.com
  3. Credentials 추가 (Add)
  4. 방금 생성한 Credentials 선택
- Branch 설정  
build 할 브랜치 설정
  - 우리 프로젝트에서는 develop에서 build 해서 develop 으로 함
- Script Path 설정
  - Jenkinsfile 입력
    - 이걸 나중에 pipeline작성할 파일 develop 브랜치에 올리면됨 지금 안해도 ok

---

## Ubuntu내에 git 설치

### ▼ git 설치

#### 1. git install

```
sudo apt install git
```

#### 2. git clone

clone 할 git 똑같이

```
git clone {주소}
```

---

## Nginx 설정

### ▼ ubuntu 내에 nginx 설치

#### 1. Nginx 설치

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

#### 2. Nginx 실행

```
sudo systemctl start nginx
```

#### 3. Nginx 파일 변경

우분투 경로 변경

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

```
sudo vi default
```

nginx 설정 추가

밑에 코드 보고 sever\_name, ssl\_certificate, ssl\_certificate\_key 도메인 수정

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

    server_name j9b201.p.ssafy.io;
    server_tokens off;

    access_log /var/log/nginx/reverse-access.log;
    error_log /var/log/nginx/reverse-error.log;

    location / {
        return 301 https://$host$request_uri;
    }
}
server {
    listen 443 ssl;
    server_name j9b201.p.ssafy.io
    server_tokens off;

    ssl_certificate /etc/letsencrypt/live/j9b201.p.ssafy.io/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/j9b201.p.ssafy.io/privkey.pem;

    location / {

        proxy_pass http://localhost:3000;
        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 $scheme;
    }
    location /api/ {
        rewrite ^/api(.*)$ $1 break;
        proxy_pass http://localhost:8081;
        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;
    }
}
```

## Redis 설치

### ▼ ubuntu 내에 redis 설치

```
cd build
```

```
sudo vi docker-compose.yml
```

코드 추가 + 포트 6379(default) 열어주기

```
redis:
  image: redis:alpine
  command: redis-server --port 6379
```

```

container_name: redis_boot
hostname: redis_boot
labels:
  - "name=redis"
  - "mode=standalone"
ports:
  - 6379:6379

```

### redis 접속

```
sudo docker exec -it redis_boot redis-cli
```

### 권한 추가

```
127.0.0.1:6379> slaveof no one
```

- slave no one 만 치면 됨
- 127.0.0.1:6379> 가 떠야 redis내부로 접속한거임
- OK 뜨면 성공

## ▼ Docker file, DockerCompose, Jenkins 작성



이제는 우분투에서 하는게 아닌 git에 코드를 직접 추가하는 것임

## Backend

backend main파일 있는 곳에다 Dockerfile 추가

Name	Last commit	Last update
..		
📁 api	Merge branch 'back-hotfix', remote-tracking bran...	4 hours ago
📁 db	style: db연결url에 포트 추가	1 week ago
📁 models	fix: 동작 코드 수정	1 day ago
📄 Dockerfile	test:volumes 경로 설정	1 week ago
📄 main.py	fix:소켓 경로 수정	1 day ago
📄 middleware.py	불필요한 코드 삭제	1 week ago
📄 requirement...	pywin	13 hours ago

## Dockerfile

```

FROM python:3.7.5

WORKDIR /workspace

COPY requirements.txt ./

# RUN pip install fastapi uvicorn[standard] --no-cache-dir
RUN pip install --upgrade pip && \
    pip install --no-cache-dir -r requirements.txt && \

```









```
rm -rf requirements.txt
# RUN pip install -r requirements.txt

COPY . ./

CMD ["uvicorn", "main:app", "--host", "0.0.0.0"]
```

## Frontend

public, src 있는 폴더에서 Dockerfile 작성

Name	Last commit	Last update
..		
 public	favicon, ssak3 변경	5 hours ago
 src	refactor: test다	6 hours ago
 .gitignore	init : frontend env	2 weeks ago
 Dockerfile	test: 빌드된 파일에 배포 적용	1 week ago
 README.md	init : frontend env	2 weeks ago
 nginx.conf	test: 빌드된 파일에 배포 적용	1 week ago
 package-lock.js...	feat: token 로컬스토리지 저장 기능 추가	1 week ago
 package.json	feat: token 로컬스토리지 저장 기능 추가	1 week ago

## Dockerfile

```
# Use an official Node.js runtime as the base image
FROM node:18 as build
# Set the working directory in the container
WORKDIR /workspace
# Copy the package.json and package-lock.json files
COPY package*.json ./
# Install the dependencies
RUN npm install
# Copy the React project files
COPY . ./
# Build the Vue project
RUN npm run build

# Use Nginx as the web server
FROM nginx
# nginx 의 default.conf 를 삭제
RUN rm /etc/nginx/conf.d/default.conf
# host pc 의 nginx.conf 를 아래 경로에 복사
COPY ./nginx.conf /etc/nginx/conf.d
# Copy the build files to the Nginx web root directory
COPY --from=build /workspace/build /usr/share/nginx/html
# Start Nginx
CMD ["nginx", "-g", "daemon off;"]
```

## Nginx

추가로 front에서는 nginx를 설정해야함

- 안하면 docker container 올리고 난 후에 npm 이 바로 꺼져서 container가 내려갑니다.

## nginx.conf

```
server {
    listen 3000;
    location / {
        root    /usr/share/nginx/html;
        index   index.html;
        try_files $uri $uri/ /index.html;
    }
    error_page 500 502 503 504 /50x.html;
    location = /50x.html {
        root    /usr/share/nginx/html;
    }
}
```

## Docker-compose, Jenkins

배포할 브랜치에다가 DockerCompose, Jenkinsfile 만들기

- 나는 pipeline 설정할때 develop으로 설정했기 때문에 develop에다 만들
- 만들때 코드 보고 경로 잘 설정하세요!!

develop ▾

S09P22B201 / + ▾

Find file

Web IDE

↓ ▾

Clone ▾

📄 README

+ Add LICENSE

+ Add CHANGELOG

+ Add CONTRIBUTING

+ Add Kubernetes cluster

+ Set up CI/CD

+ Add Wiki

⚙️ Configure Integrations

Name	Last commit	Last update
📁 .gitlab/merge_request_t...	env: MR template 추가	1 month ago
📁 assets	feat: asset 추가 [S09P22B201-37]	1 month ago
📁 backend/ssak3	Merge branch 'back-hotfix', remo...	4 hours ago
📁 frontend/ssak3	favicon, ssak3변경	5 hours ago
📁 ros/catkin_ws	feat: 세탁물 수거 완료시 API요청...	4 hours ago
🔥 .gitignore	이그노어추가	1 week ago
📄 Jenkinsfile	jenkins 최종	1 week ago
📖 README.md	Update README.md	1 month ago
🐳 docker-compose.yml	Merge branch 'back-develop' int...	1 week ago

## docker-comopse.yml

```
version: "3"

services:
  backend:
    image: backend-ssak3:latest
    container_name: back
    build:
      context: backend/ssak3/.
```



```

    dockerfile: Dockerfile
  ports:
    - 8081:8000
  environment:
    - TZ=Asia/Seoul
  volumes:
    - ../backend/ssak3:/workspace
  tty: true

frontend:
  image: frontend-ssak3:latest
  container_name: front
  build:
    context: frontend/ssak3/.
    dockerfile: Dockerfile
  ports:
    - 3000:3000
  depends_on:
    - backend
  volumes:
    - ../frontend/ssak3:/workspace
  tty: true

```

## Jenkinsfile

```

pipeline {
  agent any
  stages {
    // WebHook에서 감지된 변경 코드를 clone 해서 가져오기
    stage('Prepare') {
      steps {
        sh 'echo "Clone Repository"'
        git branch: 'develop',
            url: 'https://lab.ssafy.com/s09-mobility-smarthome-sub2/S09P22B201.git',
            credentialsId: '69daef35-2872-44f4-8e64-396a1a04dc02'
      }
      post {
        success {
          sh 'echo "Successfully Cloned Repository"'
        }
        failure {
          sh 'echo "Failed in Cloning Repository"'
        }
      }
    }

    stage('Docker container delete') {
      steps {
        script {
          def containerNames = ['back', 'front']

          for (String containerName in containerNames) {
            sh "docker stop ${containerName}"
            sh "docker rm ${containerName}"
          }
        }
      }
    }

    stage('Docker image delete') {
      steps {
        script {
          def imageNames = ['backend-ssak3', 'frontend-ssak3']

          for (String imageName in imageNames) {
            sh "docker rmi ${imageName}"
          }
        }
      }
    }

    stage('Run Docker Compose') {
      steps {
        script {
          echo pwd()
          // sh 'chmod +x /var/jenkins_home/workspace/ssak3/docker-comopse.yml'
          sh 'docker-compose up -d'
        }
      }
    }
  }
}

```

```
}  
}
```

git에다가 push 하고 jenkins 사이트 들어가서 build 시도 해보면 성공하면 됨

## ++ 만약 안될 때, 수동배포

### 1. 컨테이너 중지

```
sudo docker stop {back 컨테이너 이름} {front 컨테이너 이름}
```

### 2. 컨테이너 제거

```
sudo docker rm {back 컨테이너 이름} {front 컨테이너 이름}
```

### 3. 이미지 삭제

```
sudo docker images
```

```
sudo docker rmi {back 이미지 이름} {front 이미지 이름}
```

### 4. ubuntu git 접속

### 5. git pull

### 6. git docker-compose.yml 있는 위치에서 build

```
sudo docker-compose up -d
```