

# 포팅매뉴얼

## 기술 스택 및 개발 환경

- **BackEnd**
  - **IDE** : IntelliJ
  - **Java** : 11.0.16.1
  - **SpringBoot** : 2.5.2
  - **RestAPI** : RestDocs
  - **Testing** : JUnit5, Mockito
  - **Build Tool**: Gradle
  - **Authorization** : Spring Security, JWT
  - **DB access**: JPA, QueryDSL
  - **CI/CD** : Jenkins
  - **Open Source** : Docker
- **FrontEnd**
  - **IDE** : Visual Studio Code
  - **Node.js** : 16.16.0
  - **React** : 18.2.0
  - **typescript** : 4.8.4
  - **webpack** : 5.74.0
  - **Chrome** : 107.0.5304.107
- **Web Server**
  - **nginx**
    - port: 443 (HTTPS)
    - 정적 리소스(웹서버) url: / → response index.html
    - api url: /api/\* → localhost:8080(스프링부트)로 리버스 프록시
  - **AWS EC2**
    - ssh-pem key
- **DB**

- **MySQL** : version: 8.0.31
- **VCS**
  - **Gitlab**
- **Co-working Tool**
  - **Jira**
- **디자인** : Figma

## 개발 설정

---

### Front-end

```
# 프론트엔드 빌드
$ cd frontend/
$ npm install
$ touch .env
$ npm run build

# dist 폴더 업로드
# 새 탭을 열었을 때 대시보드가 보이면 성공
```

### 확장프로그램 폴더 업로드하기

(<https://developer.chrome.com/docs/extensions/mv3/getstarted/development-basics/#load-unpacked>)

- open Chrome and go to `chrome://extensions/`,
- enable "Developer mode",
- click "Load unpacked extension",
- select the `dist` folder in this repo.

```
# watch 모드로 개발
$ npm run watch
```

## Optional

### 1. .env

```
# OpenWeather API (https://openweathermap.org/api)
REACT_APP_WEATHER_API_KEY=
# Bing News API (https://learn.microsoft.com/en-us/bing/search-apis/bing-news-search/overview)
REACT_APP_MSEdgeKey=
# Backend URL
REACT_APP_BACKURL=
```

### 2. 환경설정

마켓플레이스에서 [`eslint`](https://marketplace.visualstudio.com/items?itemName=dbaeumer.vscode-eslint), [`prettier`](https://marketplace.visualstudio.com/items?itemName=esbenp.prettier-vscode) 설치

## Back-end

### 1. application.aws.yml

```
cloud:
  aws:
    credentials:
      access-key: // your key
      secret-key: // your key
    s3:
      bucket: // your bucket
      public-url: // your public-url
    region:
      static: ap-northeast-2
    stack:
      auto: 'false'
  spring:
    servlet:
      multipart:
        max-request-size: 20MB
        max-file-size: 20MB
```

### 2. application-secret.yml

```
# Google
spring:
  security:
    oauth2:
      client:
        registration:
          google:
            client-id: // your id
```

```
client-secret: //  
redirect-uri: "{baseUrl}/oauth2/code/{registrationId}"  
scope:  
  - email  
  - profile  
  
app:  
  auth:  
    token:  
      secret-key: // your key  
      refresh-cookie-key: "refreshToken"
```



# EC2 서버 배포 방법

## EC2 IP 주소 확인

```
curl ident.me
```

## EC2 셋팅

1. pemKey 이용해서 ec2 서버 들어가기

b. cmd

```
ssh -i K7A204T.pem ubuntu@k7a204.p.ssafy.io
```

2. 방화벽 오픈

- 기본적으로 방화벽이 설정돼있음
- 필요한 포트번호 열기
  - https : 443, http: 80, ssh : 22/tcp
  - Jenkins: 8080, SpringBoot: 8081, mysql:3306

```
sudo ufw allow <Port>
```

3. 방화벽 활성화 ⇒ 최대한 마지막에 활성화 하자!

```
sudo ufw enable
```

4. 방화벽 상태 확인

```
sudo ufw status
```

```
ubuntu@ip-172-26-14-171:~$ sudo ufw status
Status: active

To Action From
--
443 ALLOW Anywhere
80 ALLOW Anywhere
22 ALLOW Anywhere
8080 ALLOW Anywhere
8081 ALLOW Anywhere
22/tcp ALLOW Anywhere
443 (v6) ALLOW Anywhere (v6)
80 (v6) ALLOW Anywhere (v6)
22 (v6) ALLOW Anywhere (v6)
8080 (v6) ALLOW Anywhere (v6)
8081 (v6) ALLOW Anywhere (v6)
22/tcp (v6) ALLOW Anywhere (v6)
```

## 도커 설정

### 1. 오래된 버전의 도커 삭제

```
sudo apt-get remove docker docker-engine docker.io containerd runc
```

### 2. apt package update

```
sudo apt-get update
```

### 3. 도커 설치를 위해 필요한 패키지 설치

```
sudo apt-get install -y ca-certificates \
    curl \
    software-properties-common \
    apt-transport-https \
    gnupg \
    lsb-release
```

### 4. gpg Key 다운

- 리눅스 패키지 툴이 프로그램 패키지가 유효한지 확인하기 위해 설치 전 gpg키를 통해 검증함

```
sudo mkdir -p /etc/apt/keyrings
## Docker의 Official GPG Key 등록
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

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

### 5. Docker Engine 설치

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

## 6. Docker 설치 확인

```
docker --version
```

## 도커 설정

### 1. 사용자 권한 설정

- 설정 후 재시작을 해야 적용됨

```
sudo usermod -aG docker $USER
```

### 2. docker-compose 설치

- docker-compose
  - 젠킨스 설치할 때 편하게 하기 위해 설치
  - 여러 개의 컨테이너로부터 이루어진 서비스를 구축, 실행하는 순서를 자동으로 하여, 관리를 간단히하는 기능

```
#컴포즈 설치
sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
##컴포즈 권한 설정
sudo chmod +x /usr/local/bin/docker-compose
##컴포즈 설치 확인
docker-compose --version
```

## 젠킨스 설치(도커 컨테이너) 및 계정 생성

### 1. 젠킨스 컨테이너 생성

- docker-compose 이용
- 도커에 젠킨스를 이미지를 이용하여 설치

```
vim docker-compose.yml
```

### 2. 네트워크 생성

- default 네트워크를 harunetwork로 설정
- 이 네트워크로 컨테이너 간 통신을 한다.

```
sudo docker network create harunetwork
```

### 2. docker-compose.yml 생성

```
version: "3.8"
services:
  jenkins:
    privileged: true
    image: jenkins/jenkins:lts
    container_name: jenkins
    user: root
    ports:
      - "8080:8080"
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - /jenkins:/var/jenkins_home
    environment:
      TZ: "Asia/Seoul"
  nginx:
    container_name: nginx
```

```

image: nginx:alpine
ports:
  - "80:80"
  - "443:443"
volumes:
  - ./data/nginx:/etc/nginx/conf.d
  - ./data/certbot/conf:/etc/letsencrypt
  - ./data/certbot/www:/var/www/certbot
  - ./nginx/var/www/dist:/var/www/dist
certbot:
  container_name: certbot
  image: certbot/certbot
  volumes:
    - ./data/certbot/conf:/etc/letsencrypt
    - ./data/certbot/www:/var/www/certbot
networks:
  default:
    name: harunetwork

```

- jenkins의 home, docker.sock을 바인딩 시켜 젠킨스 컨테이너를 삭제해도 데이터는 남도록 한다.
  - jenkins는 8080포트를 받는다.
  - nginx의 포트 80, 443을 열어 http, https를 받을 수 있도록 한다.
  - nginx 컨테이너의 conf.d폴더, letsencrypt폴더, certbot폴더, dist폴더를 바인딩한다.
  - certbot 컨테이너의 letsencrypt폴더, certbot폴더를 바인딩한다. 이 때 nginx의 컨테이너의 letsencrypt, certbot폴더와 같은 곳을 바인딩 시켜야 한다.
    - certbot : Let's Encrypt 인증서를 사용하여 자동으로 HTTPS를 활성화하는 무료 오픈 소스 소프트웨어 도구
  - services : 컨테이너 서비스
  - jenkins : 서비스 이름
  - image : 컨테이너 생성시 사용할 image, 여기서는 jenkins/jenkins:lts 이미지를 사용(jenkins의 lts버전을 가져온다는 뜻)
  - container\_name : 컨테이너 이름
  - volumes : 공유 폴더 느낌, aws의 /var/run/docker.sock와 컨테이너 내부의 /var/run/docker.sock를 연결, /jenkins 폴더와 /var/jenkins\_home 폴더를 연결.
  - ports : 포트 매핑, aws의 8080포트와 컨테이너의 8080 포트를 연결한다.
  - privileged : 컨테이너 시스템의 주요 자원에 연결할 수 있게 하는 것 기본적으로 False로 한다고 한다.
  - user : 젠킨스에 접속할 유저 계정 (root로 할 경우 관리자)
3. docker-compose.yml 실행

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

#### 4. 도커 컨테이너 확인

```
sudo docker ps
```

## 젠킨스 생성

### 1. 젠킨스 확인

```
http://{IP}:8080
```

### 2. password 확인



```
sudo docker logs jenkins
```

### 3. 플러그인 설치

- install suggested plugins 클릭
- 기본 플러그인 자동 설치
- select plugins to install : 사용자 지정 플러그인 설치

#### Getting Started



## 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.

- 설치중

# Getting Started

✓ Folders	✓ OWASP Markup Formatter	🔄 Build Timeout	🔄 Credentials Binding	<b>Folders</b> ** JavaBeans Activation Framework (JAF) API ** JavaMail API ** bouncycastle API ** Instance Identity ** Mina SSHD API :: Common ** Mina SSHD API :: Core ** SSH server <b>OWASP Markup Formatter</b>
🔄 Timestampers	🔄 Workspace Cleanup	🔄 Ant	🔄 Gradle	
🔄 Pipeline	🔄 GitHub Branch Source	🔄 Pipeline: GitHub Groovy Libraries	🔄 Pipeline: Stage View	
🔄 Git	🔄 SSH Build Agents	🔄 Matrix Authorization Strategy	🔄 PAM Authentication	
🔄 LDAP	🔄 Email Extension	🔄 Mailer		

## 4. Admin User 생성

# Jenkins is ready!

Your Jenkins setup is complete.

Start using Jenkins

## 젠킨스 설정

### 1. Jenkins 관리 선택



Dashboard >

+ 새로운 Item

사람 사람

빌드 기록


Jenkins 관리


My Views


## 2. 플러그인 관리

### System Configuration

 시스템 설정  
환경변수 및 경로 정보등을 설정합니다.

 Global Tool Configuration  
Configure tools, their locations and automatic installers.

 플러그인 관리  
Jenkins의 기능을 확장하기 위한 플러그인을 추가, 제거, 사용, 미사용으로 설정할 수 있습니다.

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

## 3. 플러그인 설치

- gitlab

## Plugin Manager

업데이트된 플러그인 목록

설치 가능

설치된 플러그인 목록

고급

Q gitlab

Install	Name ↓	Released
<input checked="" type="checkbox"/>	<b>GitLab</b> 1.5.36 <a href="#">Build Triggers</a> This plugin allows <a href="#">GitLab</a> to trigger Jenkins builds and display their results in the GitLab UI. This plugin is up for adoption! We are looking for new maintainers. Visit our <a href="#">Adopt a Plugin</a> initiative for more information.	25 days ago
<input checked="" type="checkbox"/>	<b>Generic Webhook Trigger</b> 1.85.2 <a href="#">notification</a> <a href="#">github</a> <a href="#">webhook</a> <a href="#">Build Parameters</a> <a href="#">gitlab</a> <a href="#">Build Triggers</a> <a href="#">bitbucket</a> <a href="#">bitbucket-server</a> <a href="#">jira</a> 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.	13 days ago
<input checked="" type="checkbox"/>	<b>GitLab API</b> 5.0.1-78.v47a_45b_9f78b_7 <a href="#">Library plugins (for use by other plugins)</a> This plugin provides <a href="#">GitLab API</a> for other plugins.	3 mo 8 days ago
<input checked="" type="checkbox"/>	<b>GitLab Authentication</b> 1.16 <a href="#">Authentication and User Management</a> This is the an authentication plugin using gitlab OAuth. This plugin is up for adoption! We are looking for new maintainers. Visit our <a href="#">Adopt a Plugin</a> initiative for more information.	6 mo 15 days ago
<input type="checkbox"/>	<b>GitLab Branch Source</b> 642.v9ed86b_b_54384 Provides branch source and folder organization functionality for GitLab Repositories in Jenkins	10 days ago
<input type="checkbox"/>	<b>Gitlab Merge Request Builder</b> 2.0.0 <a href="#">Source Code Management</a> <a href="#">Miscellaneous</a> Integrates Jenkins with Gitlab to build Merge Requests	6 yr 10 mo ago
<input type="checkbox"/>	<b>GitLab Logo</b> 1.1.0	3 days 0 hr ago

Install without restart

Download now and install after restart

Update information obtained: 19 min ago

지금 확인

- docker

## Plugin Manager

업데이트된 플러그인 목록

설치 가능

설치된 플러그인 목록

고급

Q docker

Install	Name ↓	Released
<input checked="" type="checkbox"/>	<b>Docker</b> 1.2.10 Cloud Providers Cluster Management docker This plugin integrates Jenkins with <b>Docker</b> This plugin is up for adoption! We are looking for new maintainers. Visit our <a href="#">Adopt a Plugin</a> initiative for more information.	1 mo 3 days ago
<input checked="" type="checkbox"/>	<b>Docker Commons</b> 1.21 Library plugins (for use by other plugins) docker Provides the common shared functionality for various Docker-related plugins.	2 mo 6 days ago
<input checked="" type="checkbox"/>	<b>Docker Pipeline</b> 528.v7c193a_0b_e67c pipeline DevOps Deployment docker Build and use Docker containers from pipelines.	7 days 1 hr ago
<input checked="" type="checkbox"/>	<b>Docker API</b> 3.2.13-37.vf3411c9828b9 Library plugins (for use by other plugins) docker This plugin provides <b>docker-java</b> API for other plugins. This plugin is up for adoption! We are looking for new maintainers. Visit our <a href="#">Adopt a Plugin</a> initiative for more information.	6 mo 15 days ago
<input type="checkbox"/>	<b>docker-build-step</b> 2.8 Build Tools docker This plugin allows to add various docker commands to your job as build steps.	1 yr 4 mo ago
<input type="checkbox"/>	<b>CloudBees Docker Build and Publish</b> 1.4.0 Build Tools docker This plugin enables building Dockerfile based projects, as well as publishing of the built images/repos to the docker registry.	2 mo 9 days ago
<input type="checkbox"/>	<b>Amazon ECR</b> 1.107.ve50d37906739 aws This plugin integrates Jenkins with Amazon ECR.	13 hr ago

Install without restart

Download now and install after restart

Credentials to access  
Update information obtained: 20 min ago

지금 확인

- ssh

## Plugin Manager

업데이트된 플러그인 목록

설치 가능

설치된 플러그인 목록

고급

ssh

Install	Name ↓	Released
<input type="checkbox"/>	<b>SSH</b> 2.6.1 <a href="#">Build Wrappers</a> This plugin executes shell commands remotely using SSH protocol. <div>Warning: This plugin version may not be safe to use. Please review the following security notices:<ul style="list-style-type: none"><li><a href="#">CSRF vulnerability and missing permission checks allow capturing credentials</a></li><li><a href="#">Missing permission check allows enumerating credentials IDs</a></li></ul></div>	4 yr 6 mo ago
<input checked="" type="checkbox"/>	<b>Publish Over SSH</b> 1.24 <a href="#">Artifact Uploaders</a> <a href="#">Build Tools</a> Send build artifacts over SSH	8 mo 11 days ago
<input type="checkbox"/>	<b>SSH Agent</b> 295.v9ca_a_1c7cc3a_a_ This plugin allows you to provide SSH credentials to builds via a ssh-agent in Jenkins.	5 mo 23 days ago
<input type="checkbox"/>	<b>SSH Pipeline Steps</b> 2.0.39.v831c5e6468b_c <a href="#">pipeline</a> Jenkins pipeline steps which provides SSH facilities such as command execution or file transfer for continuous delivery.	5 mo 27 days ago
<input type="checkbox"/>	<b>SSH2 Easy</b> 1.4 This plugin allows you to ssh2 remote server to execute linux commands , shell , sftp upload, download etc	6 yr 5 mo ago
<input type="checkbox"/>	<b>SCP publisher</b> 1.8 <a href="#">Artifact Uploaders</a> This plugin uploads build artifacts to repository sites using SCP (SSH) protocol. <div>Warning: This plugin version may not be safe to use. Please review the following security notices:<ul style="list-style-type: none"><li><a href="#">CSRF vulnerability and missing permission check</a></li><li><a href="#">Insecure credential storage and transmission</a></li></ul></div>	11 yr ago

Install without restart

Download now and install after restart

Update information obtained: 21 min ago

지금 확인

## 젠킨스 프로젝트 생성 WebHook 설정, 자동 빌드 테스트

1. 깃랩 Repo
2. 젠킨스 프로젝트 생성
  - 새로운 Item 클릭



Dashboard >

+ 새로운 Item

사람 사람

빌드 기록

⚙ Jenkins 관리

사람 My Views


### 3. 프로젝트 설정

- freestyle project 클릭


Enter an item name

haru


» Required field


**Freestyle project**


이것은 Jenkins의 주요 기능입니다. Jenkins은 어느 빌드 시스템과 어떤 SCM(형상관리)으로 묶인 당신의 프로젝트를 빌드할 것이고, 소프트웨어 빌드보다 다른 어떤 것에 자주 사용될 수 있습니다.


**Pipeline**


Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.


**Multi-configuration project**


다양한 환경에서의 테스트, 플러그인 특성 빌드, 기타 등등 처럼 다수의 서로다른 환경설정이 필요한 프로젝트에 적합함.


**Folder**

Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.


**Multibranch Pipeline**

Creates a set of Pipeline projects according to detected branches in one SCM repository.


**Organization Folder**

Creates a set of multibranch project subfolders by scanning for repositories.

OK

- 깃랩 url 입력



## 소스 코드 관리

☐ None

☒ Git ?

Repositories ?

Repository URL ?

https://

**!** Failed to connect to repository : Command "git ls-remote -h -- https://  
returned status code 128: HEAD"  
stdout:  
stderr: remote: HTTP Basic: Access denied. The provided password or token is incorrect or your account has 2FA  
enabled and you must use a personal access token instead of a password. See  
[https://lab.ssfy.com/help/topics/git/troubleshooting\\_git#error-on-git-fetch-http-basic-access-denied](https://lab.ssfy.com/help/topics/git/troubleshooting_git#error-on-git-fetch-http-basic-access-denied)  
fatal: Authentication failed for 'https://'

Credentials ?

- none -

+ Add



Jenkins

고급...

Add Repository

→ 에러 메시지가 나타나는게 정상!!

- jenkins 추가

- Username : 싸피깃 아이디
- Password : 싸피깃 비밀번호
- ID : Credential 구별할 아무 텍스트 입력하면 됩니다.

- Credential 추가

→ 오류 메시지 사라지면 성공!

- MR 시 빌드 될 브랜치 설정

Branches to build ?

Branch Specifier (blank for 'any') ?

\*/backend

Branch Specifier (blank for 'any') ?

\*/frontend

Add Branch

#### 4. 빌드 설정

- 프로젝트 merge시 자동 빌드

##### 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://k7a204.p.ssafy.io:8080/project/haru> ?

##### 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...

- 고급 클릭 - secret token 생성
- build step - execute shell 추가

### Build Steps

Execute shell

?

✕

Command

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

pwd

고급...

Add build step ▾

→ 연결 테스트를 위한 pwd 명령어 입력(추후 수정)

## 깃랩 WebHook 연결

### 1. 깃랩 Settings - Webhooks

Menu

Search GitLab

3

29

Webhook Settings

Search page

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.

Settings

General

Integrations

Webhooks

Access Tokens

Repository

CI/CD

Packages & Registries

Monitor

Usage Quotas

URL

http://example.com/trigger-ci.json

URL must be percent-encoded if it contains one or more special characters.

Secret token

Used to validate received payloads. Sent with the request in the X-Gitlab-Token HTTP header.

Trigger

☒ Push events

Branch name or wildcard pattern to trigger on (leave blank for all)

Push to the repository.

☐ Tag push events

A new tag is pushed to the repository.

☐ Comments

A comment is added to an issue or merge request.

☐ Confidential comments

A comment is added to a confidential issue.

☐ Issues events

An issue is created, updated, closed, or reopened.

☐ Confidential issues events

A confidential issue is created, updated, closed, or reopened.

☐ Merge request events

A merge request is created, updated, or merged.

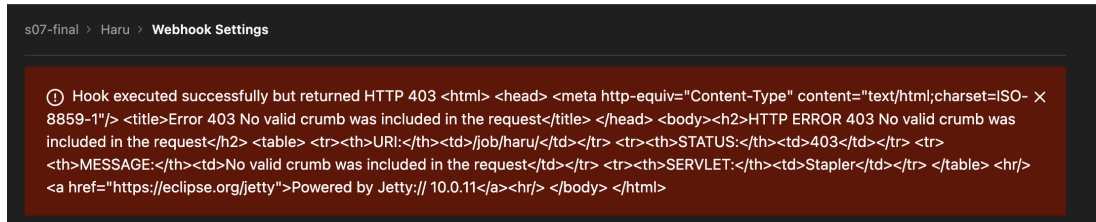
☐ Job events

A job's status changes.

EC2 서버 배포 방법

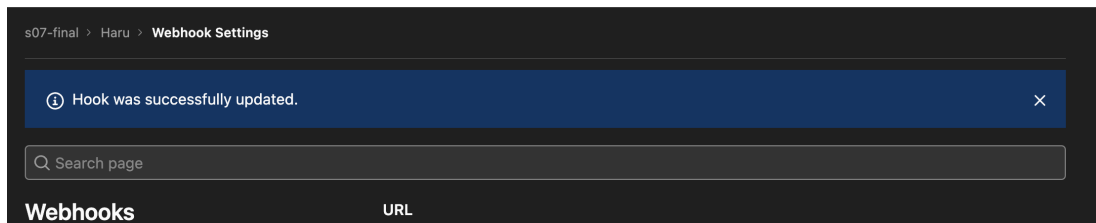
15

- Error 발생!



→ jenkins url 확인 잘하자...

- 성공 화면



## 젠킨스와 연결된 gitlab 프로젝트로 도커 이미지 빌드

### 1. 젠킨스 커네이너 안에 도커 설치

- 젠킨스에서 도커 빌드를 하기 위함

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

```
ubuntu@ip-172-26-14-171:~$ sudo docker exec -it jenkins bash
root@686cf1757cc1:/#
```

### 2. docker 사전 패키지 설치

```
apt update
apt-get install -y ca-certificates \
  curl \
  software-properties-common \
  apt-transport-https \
  gnupg \
  lsb-release
```

- 루트계정으로 접속되어있기 때문에, 젠킨스 컨테이너 내부에서는 명령어에 sudo를 지움

### 3. gpg 키 다운로드

```
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
```

#### 4. 젠킨스 컨테이너 내부에 설치된 os 확인

```
cat /etc/issue
```

```
root@686cf1757cc1:/# cat /etc/issue
Debian GNU/Linux 11 \n \l
```

- debian으로 나타남
- 기존 제공 방식은 ubuntu에 대한 gpg키를 다운로드 하는것이기 때문에 debian으로 바꿔야함
- 바꾸지 않으면 패키지를 못찾음
- 기존 명령어에서 ubuntu로 되어있는 부분을 debian으로 바꾸어주면 됨

#### 5. Docker 설치

```
apt update
apt install docker-ce docker-ce-cli containerd.io docker-compose
```

- Jenkins Container에 Docker 설치 완료!

#### 6. 프로젝트에 DockerFile 작성

##### a. SpringBoot DockerFile

```
FROM openjdk:11
COPY /build/libs/api-0.0.1-SNAPSHOT.jar app.jar
COPY /src/main/resources/application.yml application.yml
EXPOSE 8081
ENTRYPOINT ["java", "-jar", "-Duser.timezone=Asia/Seoul", "-Dspring.profiles.active=prod", "-Dspring.config.location=/applica"]
```

- classpath 인식 안되어 application.yml을 스프링이 읽지 못했음 → application.yml 을 컨테이너에 직접 복사 후 사용함

##### b. React Project DockerFile

\*\*\* 경로 수정하기

```
FROM node:16.17.0 as build-stage
WORKDIR /var/jenkins_home/workspace/deploytest/testproject_react
COPY package*.json ./
RUN npm install
COPY . .
RUN npm run build
FROM nginx:stable-alpine as production-stage

COPY --from=build-stage /var/jenkins_home/workspace/deploytest/testproject_react/build /usr/share/nginx/html
#COPY --from=build-stage /var/jenkins_home/workspace/deploytest/testproject_react/deploy_conf/nginx.conf /etc/nginx/conf.d/de
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

## 젠킨스에서 DockerFile 이용 도커 이미지 생성(Backend)

### 1. Jenkins 관리 클릭



+ 새로운 Item

사람 사람

빌드 기록

프로젝트 연관 관계

파일 핑거프린트 확인

Jenkins 관리

My Views

빌드 대기 목록



빌드 대기 항목이 없습니다.

## 2. Global Tool Configuration 클릭

### System Configuration



#### 시스템 설정

환경변수 및 경로 정보등을 설정합니다.



#### Global Tool Configuration

Configure tools, their locations and automatic installers.



#### 플러그인 관리

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



#### 노드 관리

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

## 3. Gradle 추가

## Gradle

### Gradle installations

List of Gradle installations on this system

Add Gradle

Gradle name ?

gradle

☒ Install automatically ?

≡ Install from Gradle.org

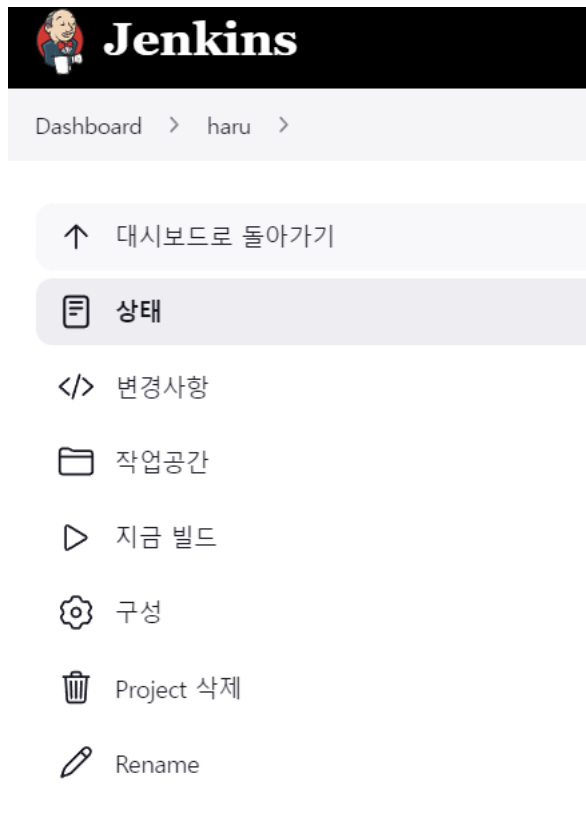
Version

Gradle 7.5.1

Add Installer ▾

Add Gradle

4. 젠킨스 프로젝트 페이지에서 구성 버튼 클릭



## 2. Build Steps 수정

**\*\* 순서 중요!!!**



## Build Steps

≡ Invoke Gradle script ?

☒ Invoke Gradle ?

Gradle Version

gradle

☐ Use Gradle Wrapper ?

Tasks ?

clean build

고급...

≡ Execute shell ?

Command

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

```
docker image prune -a --force
mkdir -p /var/jenkins_home/images_tar
cd /var/jenkins_home/workspace/haru/backend/
docker build -t backend:latest .
docker save backend:latest > /var/jenkins_home/images_tar/backend:latest.tar
```

고급...

Add build step ▾

```
# 도커 시작 전, 기존에 실행중인 도커를 멈추고 제거하는 작업.
docker ps -f name=frontend -q | xargs --no-run-if-empty docker container stop

# 컨테이너 제거
docker container ls -a -f name=frontend -q | xargs -r docker container rm

# frontend 도커 이미지 생성
docker build -t frontend:latest ./frontend

# frontend 도커 컨테이너 실행
docker run -d --name frontend -p 80:80 -p 443:443 -v /etc/letsencrypt:/etc/letsencrypt/ -v /etc/localtime:/etc/localtime:ro --net
```

### 3. 빌드 성공

```

66223a710990: Download complete
001c52e26ad5: Pull complete
d9d4b9b6e964: Pull complete
2068746827ec: Pull complete
db38d58ec8ab: Verifying Checksum
db38d58ec8ab: Download complete
9daef329d350: Pull complete
d85151f15b66: Pull complete
66223a710990: Pull complete
db38d58ec8ab: Pull complete
Digest: sha256:99bac5bf83633e3c7399aed725c8415e7b569b54e03e4599e580fc9cdb7c21ab
Status: Downloaded newer image for openjdk:11
----> 47a932d998b7
Step 2/7 : ARG JAR_FILE=/build/libs/api-0.0.1-SNAPSHOT.jar
----> Running in ab0148a907c8
Removing intermediate container ab0148a907c8
----> 34acd6b52c04
Step 3/7 : COPY ${JAR_FILE} app.jar
----> c12f680b336e
Step 4/7 : ARG YML_FILE=/src/main/resources/application.yml
----> Running in 64498915e089
Removing intermediate container 64498915e089
----> ccalc40350a1
Step 5/7 : COPY ${YML_FILE} application.yml
----> acde06cf9ala
Step 6/7 : EXPOSE 8081
----> Running in 4200e7546432
Removing intermediate container 4200e7546432
----> 83d6d6b19bc1
Step 7/7 : ENTRYPOINT ["java", "-jar", "-Duser.timezone=Asia/Seoul", "-Dspring.profiles.active=prod",
"-Dspring.config.location=/application.yml,/home/ubuntu/properties/application-db.yml", "app.jar"]
----> Running in 2c208fc323a9
Removing intermediate container 2c208fc323a9
----> ba37d9aeb53f
Successfully built ba37d9aeb53f
Successfully tagged backend:latest
+ docker save backend:latest
+ ls /var/jenkins_home/images_tar
backend:latest.tar
Finished: SUCCESS

```

#### 4. tar파일 생성 확인

```

root@686cf1757cc1:/var/jenkins_home# cd images_tar/
root@686cf1757cc1:/var/jenkins_home/images_tar# ls
backend:latest.tar

```

- 젠킨스에서 도커 이미지 빌드 후 tar 압축파일로 생성 완료!

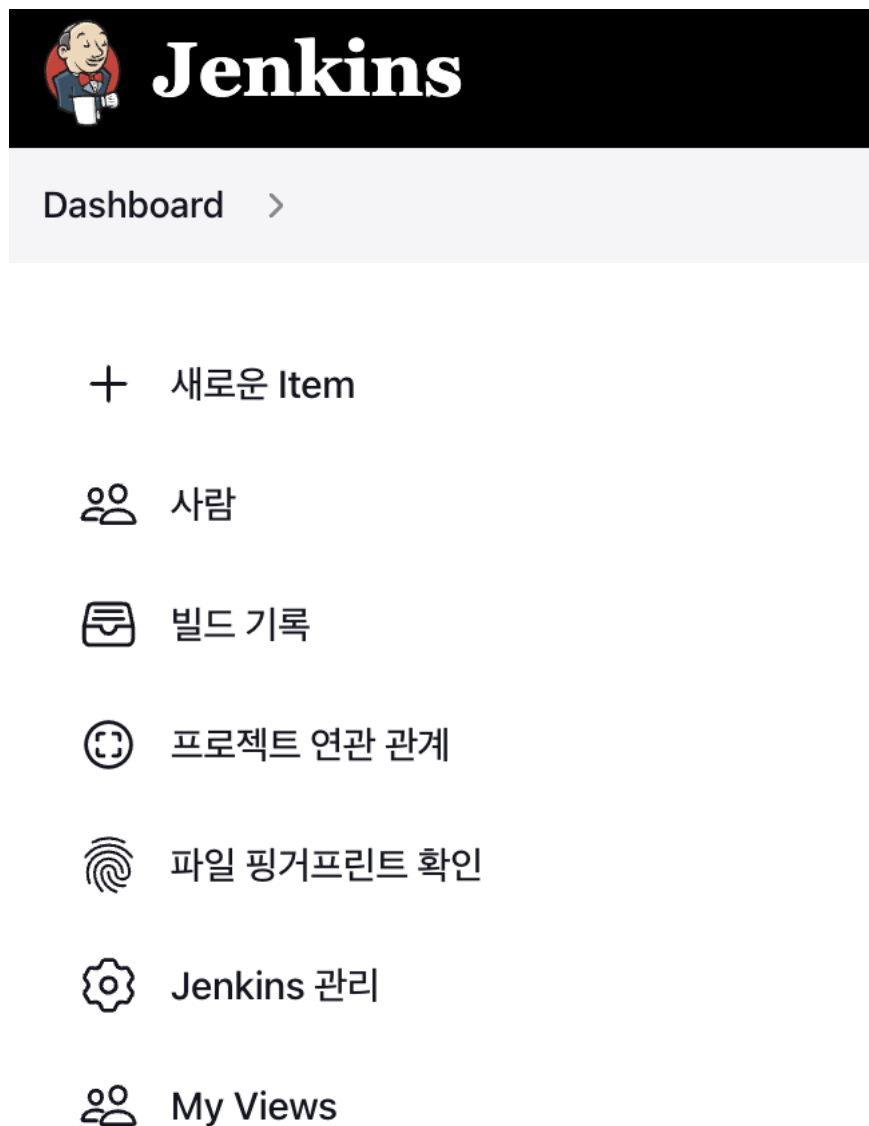
### ERROR : 프로젝트 빌드 안됨!!!! jar파일 검색 불가

```
Step 2/7 : ARG JAR_FILE=/build/libs/api-0.0.1-SNAPSHOT.jar
--> Running in 79119555a415
Removing intermediate container 79119555a415
--> f3d7abaaff43
Step 3/7 : COPY ${JAR_FILE} app.jar
COPY failed: file not found in build context or excluded by .dockerignore: stat build/libs/api-0.0.1-SNAPSHOT.jar: file does not exist
Build step 'Execute shell' marked build as failure
Finished: FAILURE
```

- Dockerfile 이미지 빌드 전 스프링 프로젝트 빌드 필요!!

## 빌드한 도커 이미지를 베이스로 컨테이너 생성(기본 배포 완료)

### 1. Jenkins 관리 선택



### 2. 시스템 설정 클릭

## Jenkins 관리

Building on the built-in node can be a security issue. You should set up distributed builds. See [the documentation](#).

Set up agent

Set up cloud

Dismiss

### System Configuration



#### 시스템 설정

환경변수 및 경로 정보등을 설정합니다.



#### Global Tool Configuration

Configure tools, their locations and automatic installers.



#### 플러그인 관리

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



#### 노드 관리

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

### 3. Publish over ssh - ssh servers 추가

- Name : 그냥 이름
- Hostname : EC2 IP
- Username : EC2 접속 계정 이름
- Key : EC2 에서 생성했던 pem파일(VSCode로 오픈 후 복사)

### 4. Test Configuration 클릭

Success

Test Configuration

### 5. 컨테이너 확인

- backend 컨테이너가 추가됨

## 컨테이너간 서버 통신을 위한 네트워크 설치

### 1. 네트워크 설치

```
sudo docker network create harunetwork
```

## EC2서버에 mysql 설치

### 1. docker-compose.mysql.yml 생성

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

```

version: '3'
services:
  mysql:
    image: mysql:8.0
    container_name: mysql
    ports:
      - 3306:3306 # HOST:CONTAINER
    environment:
      MYSQL_DATABASE: haru
      MYSQL_ROOT_PASSWORD: password
      MYSQL_USER: develop
      MYSQL_PASSWORD: password
      TZ: Asia/Seoul
    command:
      - --character-set-server=utf8mb4
      - --collation-server=utf8mb4_unicode_ci
    volumes:
      - ./mysql/data:/var/lib/mysql
    networks:
      - harunetwork
  -
networks:
  idontknownetwork:
    external: true

```

## 2. 실행

```
docker-compose -f docker-compose.mysql.yml up -d
```

```

ubuntu@ip-172-26-14-171:~$ docker-compose -f docker-compose.mysql.yml up -d
WARNING: Found orphan containers (jenkins, nginx, certbot) for this project. If you r
ompose file, you can run this command with the --remove-orphans flag to clean it up.
Starting mysql ... done

```

## 3. 확인

```
docker ps -a
```

mysql container 실행중!!

# Docker-compose + Nginx SSL 적용하기 (certbot)

## nginx 설정

## SSL 인증서 발급

### 1. certbot 설치

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

```

ubuntu@ip-172-26-14-171:~$ sudo snap install certbot --classic
Download snap "certbot" (2414) from channel "stable"
68% 211kB/s 1m12s

```

```

ubuntu@ip-172-26-14-171:~$ sudo snap install certbot --classic
certbot 1.31.0 from Certbot Project (certbot-eff✓) installed

```

```
certbot certonly --standalone -d k7a204.p.ssafy.io
```

```

ubuntu@ip-172-26-14-171:/var/log$ sudo certbot certonly --standalone -d k7a204.p.ssafy.io
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Enter email address (used for urgent renewal and security notices)
(Enter 'c' to cancel): gkdms6575@naver.com

-----
Please read the Terms of Service at
https://letsencrypt.org/documents/LE-SA-v1.3-September-21-2022.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: Y
Account registered.
Requesting a certificate for k7a204.p.ssafy.io

-----
Could not bind TCP port 80 because it is already in use by another process on
this system (such as a web server). Please stop the program in question and then
try again.
-----
(R)etry/(C)ancel: C
Could not bind TCP port 80 because it is already in use by another process on this system (such as a web server). Please stop the program in question and then
try again.
Ask for help or search for solutions at https://community.letsencrypt.org. See the logfile /var/log/letsencrypt/letsencrypt.log or re-run Certbot with -v for
more details.

```

- nginx 컨테이너에서 80포트를 사용중이어서 컨테이너 중지!

```
docker stop nginx
```

```
ubuntu@ip-172-26-14-171:~$ docker stop nginx
nginx
```

## 2. 재설치

## 3. 인증서 경로 확인

```

Certificate is saved at: /etc/letsencrypt/live/k7a204.p.ssafy.io/fullchain.pem
Key is saved at:      /etc/letsencrypt/live/k7a204.p.ssafy.io/privkey.pem

```

## 4. nginx container 볼륨 설정

- ec2서버 경로에 저장된 인증서를 nginx container 안에 넣어줘야 함

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

```

nginx:
  container_name: nginx
  image: nginx:alpine
  ports:
    - "80:80"
    - "443:443"
  volumes:
    - ./data/nginx:/etc/nginx/conf.d
    - /etc/letsencrypt:/etc/letsencrypt
    - ./nginx/var/www/dist:/var/www/dist

```

## 5. nginx 이미지 다시 올리기

```
docker-compose up -d nginx
```

---

## 6. 컨테이너 확인

```
docker ps -a
```

## 7. 확인

---