



# 포팅 매뉴얼

## 1. FrontEnd (보험사 웹)

1. 버전
2. 라이브러리 설치
3. Dockerfile

## 2. BackEnd

1. 버전
2. Dockerfile

## 3. Server

1. AWS 서버 연결 및 Docker 설치
  - ※ 배포받은 AWS 서비스 MobaXterm 를 활용해서 접속
  - ※ Docker 설치
2. MySQL 컨테이너 구축
  - ※ 설치한 Docker를 활용해 MySQL 를 컨테이너에 설치
3. Nginx 설치 및 무중단 배포 설정
4. Jenkins를 컨테이너에 설치 및 깃랩 자동배포 연동
  - ※ Jenkins 이미지 설치 및 컨테이너 실행
  - ※ 젠킨스 프로젝트 생성 WebHook 설정, 자동 빌드 테스트

## 1. FrontEnd (보험사 웹)

### 1. 버전

개발 환경과 동일

### 2. 라이브러리 설치

```
//패키지 설치
npm i

//로컬에서 실행
npm run start
```

### 3. Dockerfile

```
# 가져올 이미지를 정의
FROM node:16
# 경로 설정하기
WORKDIR /app
# package.json 워킹 디렉토리에 복사 (.은 설정한 워킹 디렉토리를 뜻함)
COPY package.json .
# 명령어 실행 (의존성 설치)
RUN npm install
# 현재 디렉토리의 모든 파일을 도커 컨테이너의 워킹 디렉토리에 복사한다.
COPY . .

# 3000번 포트 노출
EXPOSE 3000

# npm start 스크립트 실행
CMD ["npm", "start"]
```

## 2. BackEnd

### 1. 버전

개발 환경과 동일

### 2. Dockerfile

```
FROM adoptopenjdk/openjdk11 AS builder
COPY gradlew .
COPY gradle gradle
COPY build.gradle .
COPY settings.gradle .
COPY src src
RUN chmod +x ./gradlew
RUN ./gradlew bootJAR

FROM adoptopenjdk/openjdk11
COPY --from=builder build/libs/*.jar app.jar
EXPOSE 8080
ENTRYPOINT ["java", "-jar", "/app.jar"]
```

## 3. Server

### 1. AWS 서버 연결 및 Docker 설치

※ 배포받은 AWS 서비스 MobaXterm 를 활용해서 접속

※ Docker 설치

#### 1. 사전 패키지 설치

```
sudo apt update

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

#### 2. gpg 키 다운로드

```
sudo mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo 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/ubuntu \
    $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

#### 3. Docker 컨테이너 설치

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

#### 4. Docker compose 설치

```
sudo curl -L "https://github.com/docker/compose/releases/download/v1.26.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

## 2. MySQL 컨테이너 구축

### ※ 설치한 Docker를 활용해 MySQL 를 컨테이너에 설치

#### 1. mysql 이미지를 내려받는다.

```
docker pull mysql:8
```

#### 2. mysql compose yml 파일 생성

```
# 파일 규격 버전
version: "3"
# 이 항목 밑에 실행하려는 컨테이너 들을 정의
services:
  # 서비스 명
  db:
    # 사용할 이미지
    image: mysql
    # 컨테이너 이름 설정
    container_name: mysql
    # 접근 포트 설정 (컨테이너 외부:컨테이너 내부)
    ports:
      - "3306:3306"
    # -e 옵션
    environment:
      # MYSQL 패스워드 설정 옵션
      MYSQL_ROOT_PASSWORD: "P@ssw0rd12"
    # 명령어 실행 (한글 인코딩 관련)
    command:
      - --character-set-server=utf8mb4
      - --collation-server=utf8mb4_unicode_ci
    volumes:
      # -v 옵션 (다렉토리 마운트 설정)
      - /home/ubuntu/.dbdata-prod:/var/lib/mysql
```

#### 3. compose 파일 컨테이너 실행

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

## 3. Nginx 설치 및 무중단 배포 설정

#### 1. ubuntu 에 Nginx 설치

```
# ubuntu 이미지를 이용했을 경우
sudo apt-get install nginx

# 다운로드가 완료되면 해당 명령어를 통해 nginx 설치 폴더로 이동

# nginx 실행
sudo service nginx start

# nginx 실행 확인
ps -ef | grep nginx
```

## 2. nginx 가 프론트, 백엔드 서버를 바라보도록 reverse proxy 설정

/etc/nginx/sites-enabled/default

```
include /etc/nginx/conf.d/service-url.inc;

root /var/www/html;

# Add index.php to the list if you are using PHP
index index.html index.htm index.nginx-debian.html;

server_name _;

location /api {
    proxy_pass $back_url;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "Upgrade";
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-Proto $scheme;
    proxy_set_header X-Forwarded-Host $server_name;
}

location / {
    # First attempt to serve request as file, then
    # as directory, then fall back to displaying a 404.
    proxy_pass $front_url;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "Upgrade";
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-Proto $scheme;
    proxy_set_header X-Forwarded-Host $server_name;

    # pass PHP scripts to FastCGI server
    #
    #location ~ \.php$ {
    #    include snippets/fastcgi-php.conf;
    -- INSERT --
```

## 3. 무중단 배포 시 nginx 포워딩을 위해 ec2에서 shell script 작성

deploy.sh

```
echo -e "set \$front_url http://127.0.0.1:${FRONT_PORT};\nset \$back_url http://127.0.0.1:${BACKEND_PORT};" | sudo tee /etc/nginx/conf
sudo service nginx reload
```

nginx 서버 주소를 수정하고, nginx reload

## 4. Jenkins를 컨테이너에 설치 및 깃랩 자동배포 연동

### ※ Jenkins 이미지 설치 및 컨테이너 실행

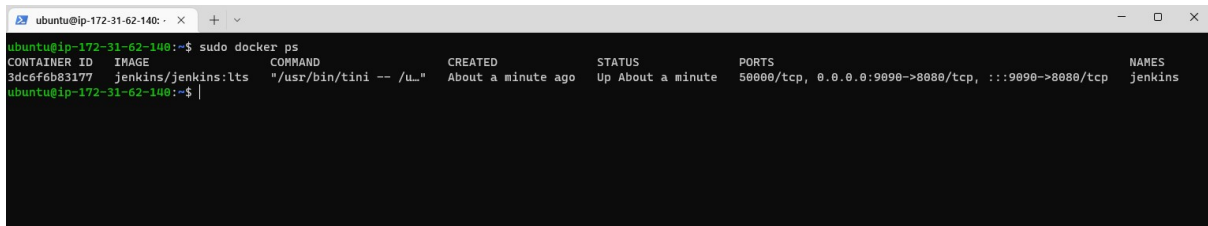
#### 1. docker 를 통해 jenkins 이미지 내려받기

```
docker pull jenkins/jenkins
```

2. jenkins 컨테이너 실행. 도커 인 도커가 가능하게 하기 위해, docker.sock 파일을 jenkins 컨테이너에서 이용할 수 있도록 설정해주어야 한다.

```
docker run --name jenkins -d -p 8080:8080 -p 50000:50000 -v ./jenkins:/var/jenkins_home -v /var/run/docker.sock:/var/run/docker.sock -
```

3. `sudo docker ps` 명령어로 확인해보면 다음과 같이 컨테이너가 올라가 있는 것을 확인 가능



```
ubuntu@ip-172-31-62-140:~$ sudo docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS
3dc6f6b83177   jenkins/jenkins:lts   "/usr/bin/tini -- /u..."   About a minute ago   Up About a minute   50000/tcp, 0.0.0.0:9090->8080/tcp, :::9090->8080/tcp
jenkins
```

4. jenkins 컨테이너에 접속

```
sudo docker exec -it {jenkins컨테이너id} /bin/bash
```

5. jenkins 컨테이너 내부에 도커 설치

```
# 1. jenkins 컨테이너에서 docker 설치하기
curl -fsSL https://get.docker.com/builds/Linux/x86_64/docker-17.04.0-ce.tgz

# 2. docker 파일 압축 풀기
tar xzvf docker-17.04.0-ce.tgz

# 3. bin 폴더로 docker 의 실행파일을 옮겨 docker 명령어를 사용할 수 있도록 설정
mv docker/docker /usr/local/bin

# 4. 압축 파일 삭제하기
rm -r docker docker-17.04.0-ce.tgz

# 5. docker 로그인
docker login
```

루트계정으로 접속되어있기 때문에, 젠킨스 컨테이너 내부에서는 명령어에 `sudo`를 지워야함

## 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
```

젠킨스에 gpg 키를 다운로드 받을 때의 변경사항

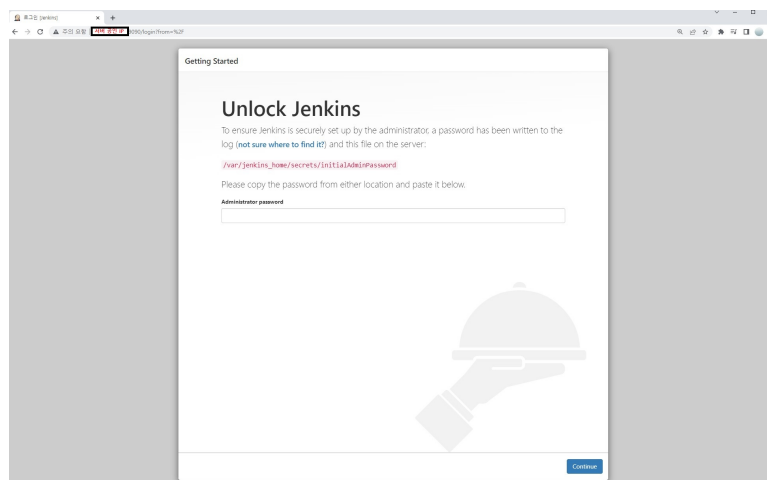
```
ubuntu@ip-172-31-62-140: ~$ sudo docker exec -it jenkins bash
root@3dc6f6b83177:/# cat /etc/issue
Debian GNU/Linux 11 \n \l

root@3dc6f6b83177:/# |
```

## 6. Docker-compose 설치

```
sudo curl -L "https://github.com/docker/compose/releases/download/v1.26.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/do
```

## 젠킨스 계정 생성 및 플러그인 설치



서버 공인 IP:9090 포트로 접속하면 다음과 같은 젠킨스 시작 화면이 나타남

→ AWS 인스턴스 할당된 IPv4 주소:9090 으로 접속, 인바운드 규칙에 9090 추가 필수!

```
*****
*****
*****
*****

Jenkins initial setup is required. An admin user has been created and a password generated.
Please use the following password to proceed to installation:

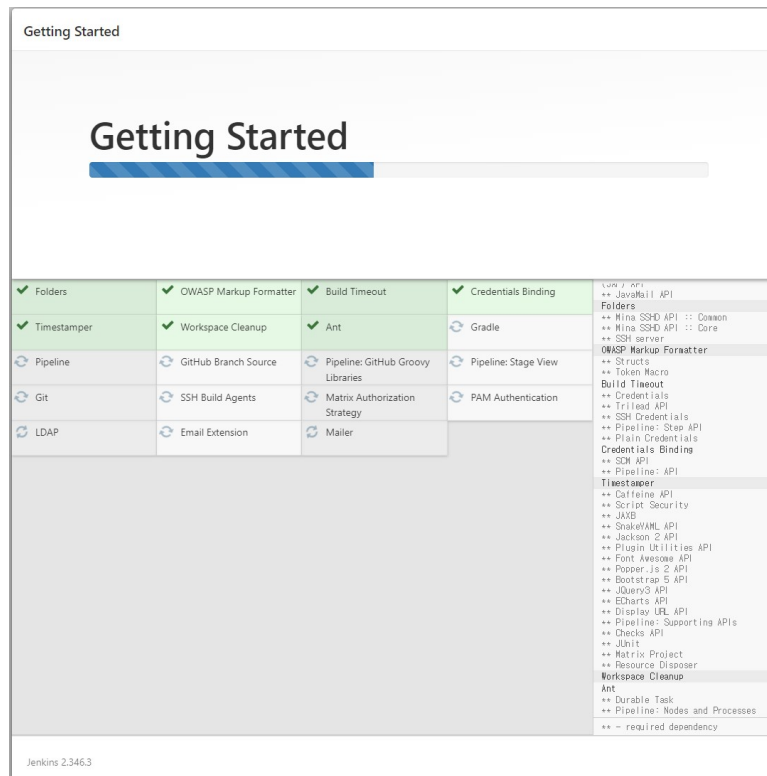
613e613f92844ca98db1adc4e2aa38f

This may also be found at: /var/jenkins_home/secrets/initialAdminPassword

*****
*****
*****
```

여기서 말하는 Administrator password는 `sudo docker logs jenkins` 명령어를 통해 위 사진의 빨간 네모 상자 안의 값을 입력

다음으로 두 개의 버튼 중 `Install suggested plugins` 를 클릭



이것 저것 플러그인들이 설치되는 모습

Getting Started

## Create First Admin User

계정명:

암호:

암호 확인:

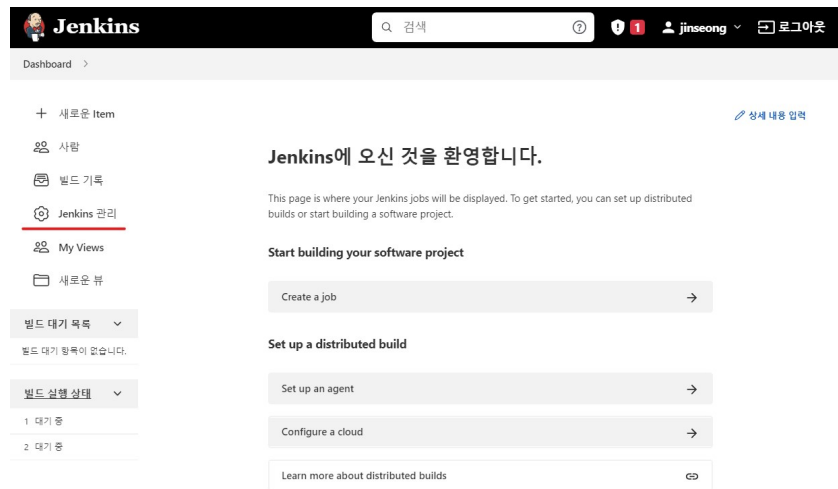
이름:

이메일 주소:

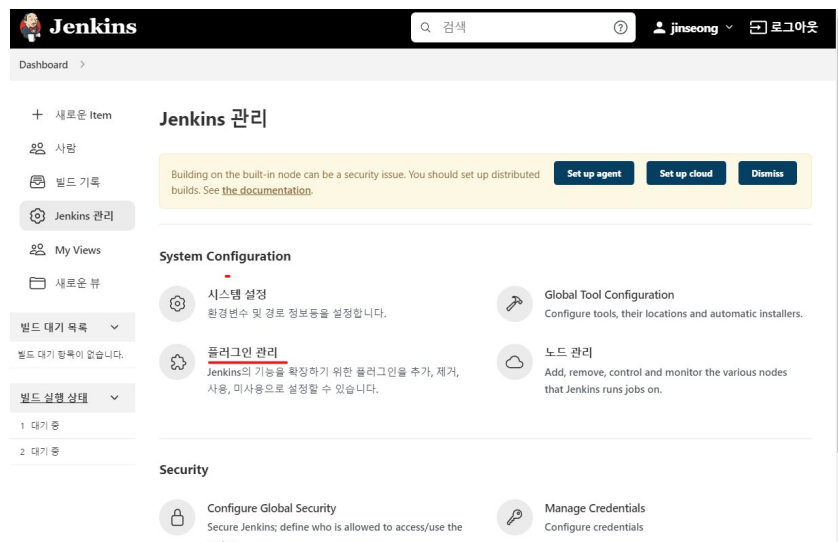
Jenkins 2.346.3

Skip and continue as admin

Save and Continue



젠킨스 시작 메인화면, 플러그인 설치를 위해 jenkins 관리 탭을 클릭



플러그인 관리 페이지로 이동



## Plugin Manager

업데이트된 플러그인 목록

설치 가능

설치된 플러그인 목록

고급

Q gitlab

Install	Name ↓	Released
<input checked="" type="checkbox"/>	<a href="#">GitLab</a> 1.5.35 <a href="#">Build Triggers</a> This plugin allows <a href="#">GitLab</a> to trigger Jenkins builds and display their results in the GitLab UI. <div>This plugin is up for adoption! We are looking for new maintainers. Visit our <a href="#">Adopt a Plugin</a> initiative for more information.</div>	2 mo 9 days ago
<input checked="" type="checkbox"/>	<a href="#">Generic Webhook Trigger</a> 1.8.4 <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.	4 mo 20 days ago
<input checked="" type="checkbox"/>	<a href="#">Gitlab API</a> 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.	1 mo 13 days ago
<input checked="" type="checkbox"/>	<a href="#">GitLab Authentication</a> 1.16 <a href="#">Authentication and User Management</a> This is the an authentication plugin using gitlab OAuth. <div>This plugin is up for adoption! We are looking for new maintainers. Visit our <a href="#">Adopt a Plugin</a> initiative for more information.</div>	4 mo 20 days ago

[Install without restart](#) [Download now and install after restart](#) Update information obtained: 21 min ago [지금 확인](#)

먼저 설치 가능 탭으로 탭을 변경해주고, 검색어에 `gitlab`을 검색

그 후, 밑줄 친 플러그인들을 체크하고, `install without restart` 버튼을 클릭

## Plugin Manager

업데이트된 플러그인 목록 **설치 가능** 설치된 플러그인 목록 고급

Install	Name	Released
<input checked="" type="checkbox"/>	<a href="#">Docker</a> 1.2.9 Cloud Providers Cluster Management docker	4 mo 12 days ago
This plugin integrates Jenkins with Docker This plugin is up for adoption! We are looking for new maintainers. Visit our <a href="#">Adopt a Plugin</a> initiative for more information.		
<input checked="" type="checkbox"/>	<a href="#">Docker Commons</a> 1.21 Library plugins (for use by other plugins) docker	11 days ago
Provides the common shared functionality for various Docker-related plugins.		
<input checked="" type="checkbox"/>	<a href="#">Docker Pipeline</a> 521.v1a_a_dd2073b_2e pipeline DevOps Deployment docker	28 days ago
Build and use Docker containers from pipelines.		
<input checked="" type="checkbox"/>	<a href="#">Docker API</a> 3.2.13-37.vf3411c9828b9 Library plugins (for use by other plugins) docker	4 mo 20 days ago
This plugin provides <a href="#">docker-java</a> 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.		
	<a href="#">docker-build-step</a> 2.8	

Install without restart Download now and install after restart Update information obtained: 25 min ago [지금 확인](#)

같은 방식으로 Docker 관련 플러그인도 설치

## Plugin Manager

업데이트된 플러그인 목록 **설치 가능** 설치된 플러그인 목록 고급

Install	Name	Released
<input type="checkbox"/>	<a href="#">SSH</a> 2.6.1 Build Wrappers	4 yr 4 mo ago
This plugin executes shell commands remotely using SSH protocol. Warning: This plugin version may not be safe to use. Please review the following security notices: <ul style="list-style-type: none"> <li>CSRF vulnerability and missing permission checks allow capturing credentials</li> <li>Missing permission check allows enumerating credentials IDs</li> </ul>		
<input checked="" type="checkbox"/>	<a href="#">Publish Over SSH</a> 1.24 Artifact Uploaders Build Tools	6 mo 16 days ago
Send build artifacts over SSH		
<input type="checkbox"/>	<a href="#">SSH Agent</a> 295.v9ca_a_1c7cc3a_a_ This plugin allows you to provide SSH credentials to builds via a ssh-agent in Jenkins.	3 mo 28 days ago
<input type="checkbox"/>	<a href="#">SSH Pipeline Steps</a> 2.0.39.v831c5e6468b_c pipeline	4 mo 2 days ago
Jenkins pipeline steps which provides SSH facilities such as command execution or file transfer for continuous delivery.		
<input type="checkbox"/>	<a href="#">SSH2 Easy</a> 1.4 This plugin allows you to ssh2 remote server to execute linux commands , shell , sftp upload, download etc	6 yr 3 mo ago

Install without restart Download now and install after restart Update information obtained: 26 min ago [지금 확인](#)

마지막으로 SSH 연결 관련 플러그인까지 설치

Matrix Authorization Strategy	✓ 성공
PAM Authentication	✓ 성공
LDAP	✓ 성공
Email Extension	✓ 성공
Mailer	✓ 성공
Loading plugin extensions	✓ Success
Jersey 2 API	✓ 성공
GitLab	✓ 성공
Generic Webhook Trigger	✓ 성공
Gitlab API	✓ 성공
GitLab Authentication	✓ 성공
Gitlab API	✓ 성공
Loading plugin extensions	✓ Success
Authentication Tokens API	✓ 성공
Docker Commons	✓ 성공
Docker API	✓ 성공
Docker	✓ 성공
Docker Pipeline	✓ 성공
Docker API	✓ 성공
Loading plugin extensions	✓ Success
Infrastructure plugin for Publish Over X	⋯ 대기중
Publish Over SSH	⋯ 대기중
Loading plugin extensions	⋯ Pending

→ [메인 페이지로 돌아가기](#)

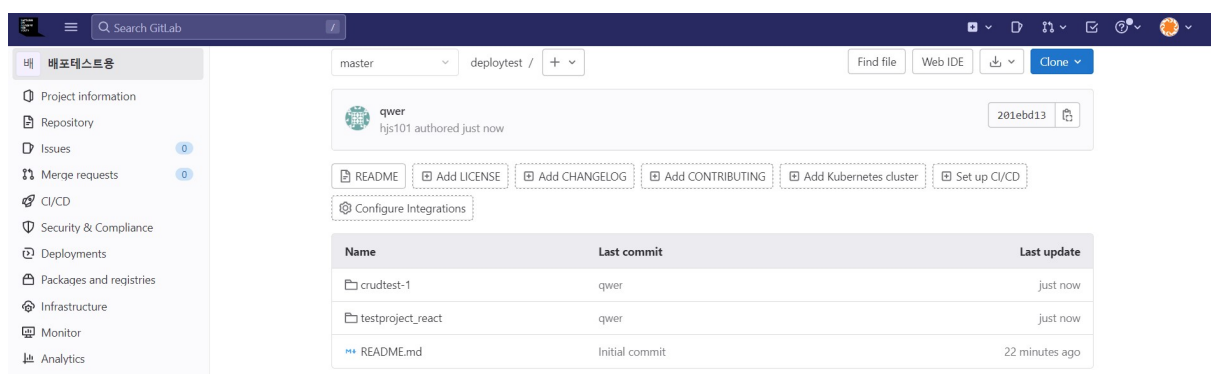
(설치된 플러그인을 바로 이용하실 수 있습니다.)

→ ☐ 설치가 끝나고 실행중인 작업이 없으면 Jenkins 재시작.

모두 설치가 완료

## ※ 젠킨스 프로젝트 생성 WebHook 설정, 자동 빌드 테스트

### 1. 깃랩 Repo 생성



**crudtest-1** 는 Spring Boot 프로젝트, **testproject\_react** 는 React 프로젝트입니다.

### 2. 젠킨스에서 gitlab credentials 추가 (Jenkins 관리 → Credentials → System → Global Credentials)

Dashboard > Jenkins 관리 > Credentials > System > Global credentials (unrestricted) >

## New credentials

Kind  
Username with password

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

Username ?  
yangjinee@naver.com

☐ Treat username as secret ?

Password ?  
\*\*\*\*\*

ID ?  
gitlab\_id

❗ This ID is already in use

Create

gitlab 의 id & pw 입력, id 는 `gitlab_id` 로 설정

### 3. 젠킨스 프로젝트 생성

검색

jinseong

로그아웃

Dashboard >

+

새로운 Item

사람

빌드 기록

Jenkins 관리

My Views

새로운 뷰

빌드 대기 목록

빌드 대기 항목이 없습니다.

빌드 실행 상태

1 대기 중

2 대기 중

## 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

환경 변수 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.


젠킨스 메인페이지에서 `새로운 item` 을 클릭


**Enter an item name**


deploytest


» 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 저렴 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.

**OK**

**Pipeline** 를 클릭하고 **OK** 버튼을 클릭

## 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://3.34.144.69:8080/project/web\_server ?

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

push event 가 일어났을 때 build 되도록 설정

아래의 고급 버튼을 클릭, secret token 생성

☒ 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 ?

7971adce175924cc2bf2933b84202074

Generate

secret token 은 gitlab 에서 webhook 연결할 때 사용할 것이니 복사해두기

Pipeline 생성. 무중단 배포 pipeline 작성 (blue-green 방식)

```

pipeline {
  agent any
  stages {
    stage('Github') {
      steps {
        // credentials 에서 설정한 gitlab_id 입력
        git branch: 'develop', credentialsId: 'gitlab_id', url : 'https://lab.ssafy.com/s08-final/S08P31E101.git'
      }
    }
    stage('Health Check') {
      steps {
        script {
          //blue 체크
          BLUE_EXIST = sh(returnStdout: true, script: "curl -s 'http://${host_ip}:3000' > /dev/null || curl -s 'http://${host_ip}:8080' > /dev/null")
          BLUE_EXIST = BLUE_EXIST.trim()

          BEFORE_COLOR = ''
          AFTER_COLOR = ''
          if(BLUE_EXIST.equals('true')) {
            BEFORE_COLOR = 'blue'
            AFTER_COLOR = 'green'
          } else {
            BEFORE_COLOR = 'green'
            AFTER_COLOR = 'blue'
          }

          println("BEFORE_COLOR: ${BEFORE_COLOR} // AFTER_COLOR: ${AFTER_COLOR}")
        }
      }
    }
    stage('Build') {
      steps {
        script {
          sh(returnStdout: false, script: "docker-compose -f docker-compose.${AFTER_COLOR}.yaml --env-file ../.env.dbprod build")
        }
      }
    }
    stage('Deploy') {
      steps {
        script {

```

```

        sh(returnStdout: false, script: "docker-compose -f docker-compose.${AFTER_COLOR}.yaml --env-file ../.env.dbprod up -d")
    }
}
}

stage('Health Check And delete') {
    steps{
        script {
            def count = 1
            while(count <= 20) {
                // SLEEP 10sec
                sh(returnStdout: false, script: 'sleep 10')

                def IS_DONE = "";
                if(BLUE_EXIST.equals('true')) {
                    // GREEN에 배포됐는지 확인
                    IS_DONE = sh(returnStdout: true, script: "curl -s 'http://${host_ip}:3001' > /dev/null || curl -s 'http://${host_ip}:8000' > /dev/null")
                } else {
                    // BLUE에 배포됐는지 확인
                    IS_DONE = sh(returnStdout: true, script: "curl -s 'http://${host_ip}:3000' > /dev/null || curl -s 'http://${host_ip}:8001' > /dev/null")
                }
                IS_DONE = IS_DONE.trim()
                println("IS_DONE: ${IS_DONE}")

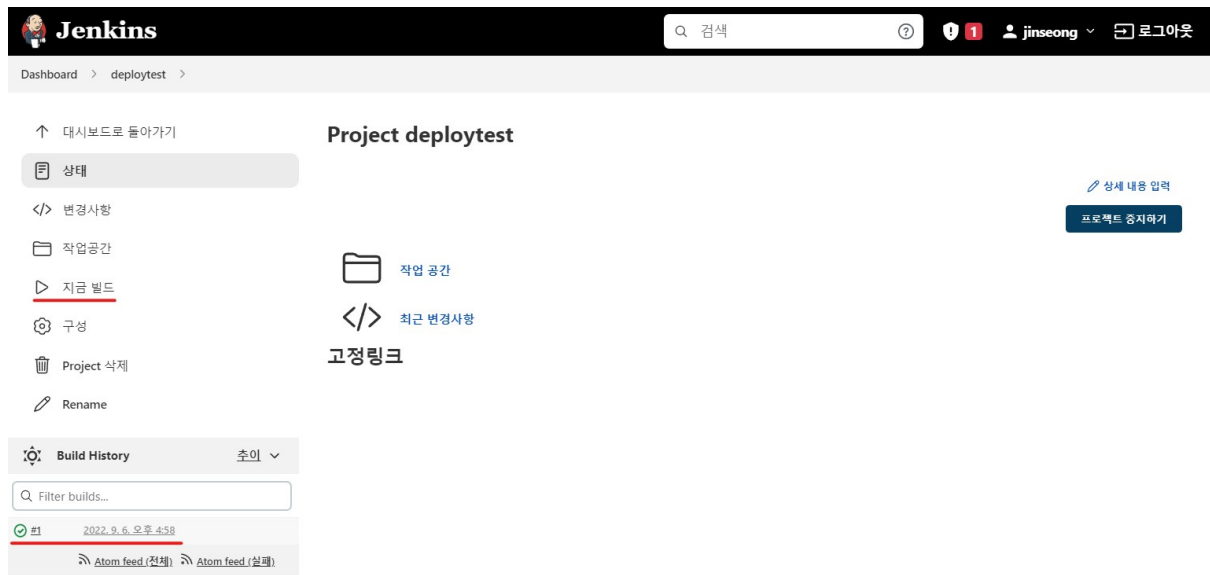
                if(IS_DONE.equals('true')) {
                    // 이전 버전 내리기
                    sh(returnStdout: false, script: "docker-compose -f docker-compose.${BEFORE_COLOR}.yaml down --rmi all")
                    // nginx 설정 변경
                    if(BLUE_EXIST.equals('true')) {
                        //green nginx 설정 변경
                        sshagent (credentials: ['host_server']) {
                            sh """
                                ssh -o StrictHostKeyChecking=no ubuntu@${host_ip} 'FRONT_PORT=3001 BACKEND_PORT=8001 ./deploy.sh'
                                """
                            }
                        }
                    } else {
                        //blue nginx 설정 변경
                        sshagent (credentials: ['host_server']) {
                            sh """
                                ssh -o StrictHostKeyChecking=no ubuntu@${host_ip} 'FRONT_PORT=3000 BACKEND_PORT=8000 ./deploy.sh'
                                """
                            }
                        }
                    }

                    println("Shutting down... ${BEFORE_COLOR}")
                    break
                } else {
                    println("Can't shut down... ${BEFORE_COLOR}")
                    count++
                }
            }
        }
    }
}
}
}
}

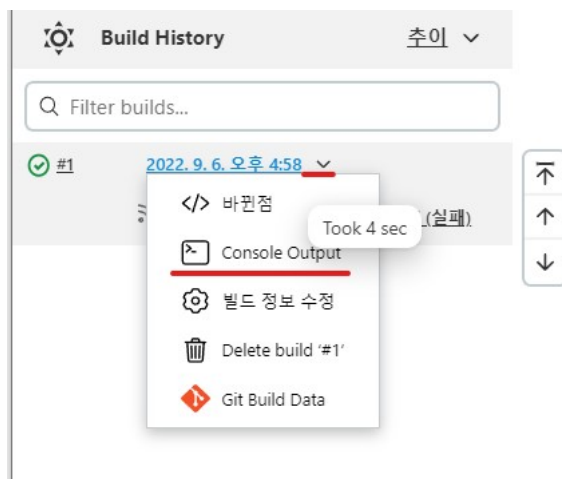
```

- green 포트가 실행되고 있는지 확인. 실행 중이면 blue 에 배포 시작
- green 이 실행중이지 않으면 green 에 배포 시작
- docker compose-up 를 통해 이미지 빌드, 컨테이너 실행함
- 배포한 서버가 동작하는지 10초 주기로 확인
- 서버가 정상 동작하면, 이전 버전의 서버 내리고, nginx 에서 포트포워딩을 이전서버에서 새로 올린 서버로 바꿔준다. (deploy.sh)

파이프라인 저장, 빌드 준비 완료



빌드 시험테스트 완료



빌드 히스토리에서, **Console Output** 에 진입



## 콘솔 출력

```


Started by user jinseong
Running as SYSTEM
Building in workspace /var/jenkins_home/workspace/deploytest
The recommended git tool is: NONE
using credential jinseong
Cloning the remote Git repository
Cloning repository https://lab.ssafy.com/h5282001/deploytest
> git init /var/jenkins_home/workspace/deploytest # timeout=10
Fetching upstream changes from https://lab.ssafy.com/h5282001/deploytest
> git --version # timeout=10
> git --version # 'git version 2.30.2'
using GIT_ASKPASS to set credentials
> git fetch --tags --force --progress -- https://lab.ssafy.com/h5282001/deploytest +refs/heads/*:refs/remotes/origin/* # timeout=10
> git config remote.origin.url https://lab.ssafy.com/h5282001/deploytest # timeout=10
> git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision 70a029ec77f1385b6e26ccd7f7923f5cf610b3ef (refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10
> git checkout -f 70a029ec77f1385b6e26ccd7f7923f5cf610b3ef # timeout=10
Commit message: "test"
First time build. Skipping changelog.
[deploytest] $ /bin/sh -xe /tmp/jenkins18179672897912898776.sh
+ pwd
/var/jenkins_home/workspace/deploytest
Finished: SUCCESS
  
```

## 깃랩 WebHook 연결

배 배포테스트용


- Project information
- Repository
- Issues 0
- Merge requests 0
- CI/CD
- Security & Compliance
- Deployments
- Packages & Registries
- Infrastructure
- Monitor
- Analytics
- Wiki
- Snippets
- Settings**

한진성 > 배포테스트용



**배포테스트용**
  
Project ID: 178966

33 Commits
 1 Branch
 0 Tags
 696 KB Project Storage



**Auto DevOps**
  
It will automatically build, test, and deploy your application based on a predefined CI/CD configuration.
   
Learn more in the Auto DevOps documentation
   
[Enable in settings](#)

master
 deploytest /
 Find file
 Web IDE
 Clone

test
 70a029ec

한진성 authored 6 days ago

Add LICENSE
 Add CHANGELOG
 Add CONTRIBUTING
 Add Kubernetes cluster
 Set up CI/CD

Add integrations

	Last commit	Last update
project	tre	6 days ago
project_react	test	6 days ago
.gitignore	test	1 week ago
README.md	Update README.md	1 week ago

배포할 프로젝트가 있는 깃랩 **Repository** 에서 밑줄친 위치로 **WebHooks** 페이지로 이동

### 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 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**  

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.

☐ Pipeline events  
A pipeline's status changes.

☐ Wiki page events  
A wiki page is created or updated.

☐ Deployment events  
A deployment starts, finishes, fails, or is canceled.

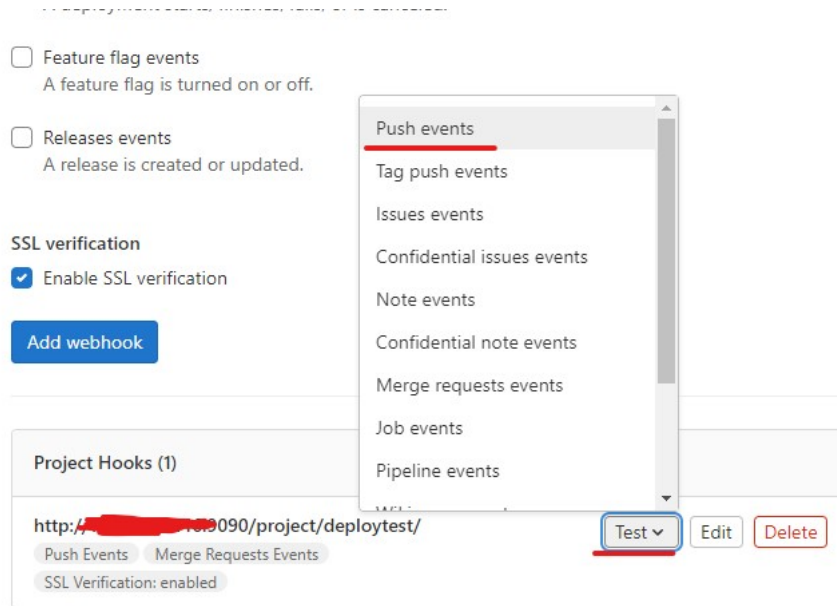
URL에는 `http://배포서버공인IP:9090/project/생성한jenkins프로젝트이름/` 을 입력

Secret token에는 아까 위에서 젠킨스 프로젝트를 생성할 때 저장해둔 값을 입력

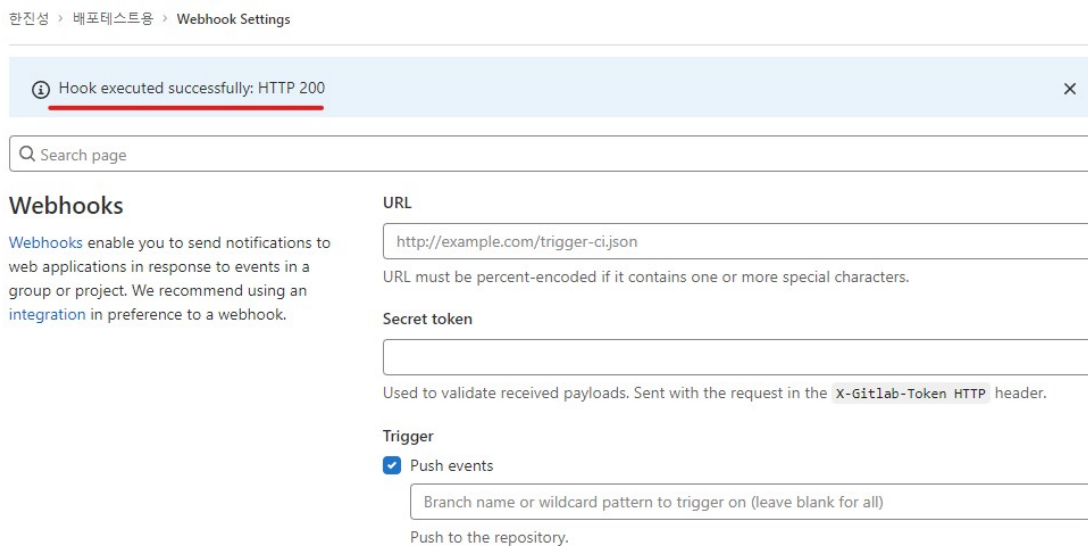
빌드 유발 Trigger으로, **Push events**, **Merge request events** 를 설정

대상 Branch는 master으로 설정


완료했다면 Add Webhook 버튼을 눌러 webhook을 생성



WebHook을 생성하고 나면 빌드 테스트를 위해 생성된 WebHook에서 test를 누르고, Push events를 선택



응답이 잘 넘어온 것을 확인할 수 있음



# Jenkins

Dashboard > deploytest >

↑

대시보드로 돌아가기

☰

상태

</>

변경사항

📁

작업공간

▶

지금 빌드

⚙️

구성

🗑️

Project 삭제

✏️

Rename

⚙️

Build History

추이 ▼

🔍

Filter builds...

✅ #2

2022. 9. 6. 오후 5:12

Started by GitLab push by 한진성

✅ #1

2022. 9. 6. 오후 4:58

📡

Atom feed (전체)

📡

Atom feed (실패)

📁

작업 공간

</>

최근 변경사항

## 고정링크

- Last build, (#1),4 min 23 sec 전
- Last stable build, (#1),4 min 23 sec 전
- Last successful build, (#1),4 min 23 sec 전
- Last completed build, (#1),4 min 23 sec 전

젠킨스에서도 정상적으로 빌드가 수행되는 것을 확인할 수 있음