



# 포팅 매뉴얼

## 목차

1. 프로젝트 기술 스택
2. MySQL 설정
3. MongoDB 설정
4. Redis 설치
5. 배포 사전 작업
6. 배포
7. 외부 서비스 설정
8. 기타 설정 파일

## 1. 프로젝트 기술 스택

### Front-end

- React 18.2.0
- JavaScript ES6+
- SCSS / HTML5
- Chrome / Edge
- Axios 0.27.2
- Node.js 16.16.0
- Npm 8.11.0
- Redux 4.2.0
- React Router Dom 6.3.0
- ESLint 8.20.0
- Webpack 5.74.0
- Ant Design 4.23.1

### Back-end

- JAVA 11
- Spring Boot 2.7.3
- Spring Security

- Hibernate
- JPA
- Gradle 7.5
- JWT 0.9.1
- MapStruct 1.5.2

#### Database

- Redis 7.0.4
- MongoDB 5.0.12
- MySQL

#### Distributed System

- Hadoop 3.2.2
- HDFS
- MapReduce
- Zookeeper 3.6.3
- Kafka 3.2.0

#### Infra

- NginX
- AWS EC2 Ubuntu 20.04 LTS
- Docker 20.10.18
- Docker Compose 1.29.2
- Jenkins

#### Crawling

- Selenium
- Scrapy

## 2. MySQL 설정

### 스키마 생성

```
create database consultant default charset utf8mb4;
```

### MYSQL 계정 생성 ( 8.0.28 기준 )

```
create user 'ssafy'@'localhost' identified by 'ssafy';
grant all privileges on consultant.* to 'ssafy'@'localhost' with grant option;
```

## 3. MongoDB 설정

### 계정 생성

1. cmd에서 MongoDB접속

```
$ mongosh
```

## 2. admin 데이터베이스 사용

```
$ use admin
```

## 3. 아이디 : ssafy 비밀번호 : ssafy인 root계정 생성

```
$ db.createUser( { user: "ssafy", pwd: "ssafy", roles: ["root"] })
```

## 4. Redis 설치

```
docker pull redis
docker run --name some-redis -d -p 6379:6379 redis
```

## 5. 배포 사전 작업

### 서버 접속

- SSH를 이용한, 명령어 위주의 방법으로 접속

### 1. 컴퓨터에 SSH를 설치하고 설치가 완료되면, CMD에서 SSH 명령어를 입력해서 설치 완료 확인

```
C:\Users\#yoong>ssh
usage: ssh [-46AaCfGgKkMNnqsTtVvXxYy] [-B bind_interface]
          [-b bind_address] [-c cipher_spec] [-D [bind_address:]port]
          [-E log_file] [-e escape_char] [-F configfile] [-I pkcs11]
          [-i identity_file] [-J [user@]host[:port]] [-L address]
          [-l login_name] [-m mac_spec] [-O ctl_cmd] [-o option] [-p port]
          [-Q query_option] [-R address] [-S ctl_path] [-W host:port]
          [-w local_tun[:remote_tun]] destination [command]
```

### 2. 키이름.pem 파일이 있는 경로로 이동해서 아래 명령어 입력

```
ssh -i 키이름.pem ubuntu@(도메인 혹은 IP)
```

( 위 명령어를 통해서 접속하기 위해서는 키 접근 권한을 줄여줘야 하므로 윈도우에서 진행하는 것 보다 우분투 등 WSL 기반 시스템을 설치해서 사용할길 권장 )

```
yjk@JKY:/mnt/e/SSAFY_PROJECT$ sudo ssh -i 키이름.pem ubuntu@1.1.1.1.p.ssafy.io
[sudo] password for yjk:
Welcome to Ubuntu 20.04 LTS (GNU/Linux 5.4.0-1018-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Thu Aug 18 01:52:28 UTC 2022

System load:          0.02
Usage of /:            3.5% of 310.15GB
Memory usage:         36%
Swap usage:           0%
Processes:            169
Users logged in:      0
```

성공적으로 접속한 화면

### EC2에 도커 설치

- 프로젝트를 도커를 통한 컨테이너화 할 예정이기 때문에 EC2에는 도커와 도커 컴포즈 이외에 설치할 것은 없음

#### 1. 먼저 설치에 필요한 사전 업데이트 및 설치 진행

```
sudo apt-get update
sudo apt install git
sudo apt-get install apt-transport-https ca-certificates curl gnupg-agent software-properties-common
```

#### 2. 아래 명령어를 입력 한 이후 콘솔에 출력되는 결과값이 'OK' 이면 성공

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

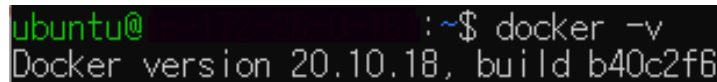
#### 3. 아래 명령어의 `arch=아키텍처` 에는 자신의 환경에 맞는 아키텍처를 작성

```
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
```

#### 4. 도커 설치

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

#### 5. 성공적으로 설치되었는지 확인



```
ubuntu@:~$ docker -v
Docker version 20.10.18, build b40c2f6
```

#### 6. 도커 실행

```
sudo systemctl enable docker && sudo service docker start
```

### 도커 On Off 명령어

- 도커 종료

```
sudo systemctl stop docker.socket
```

- 도커 실행

```
sudo systemctl start docker.socket
sudo systemctl enable docker && sudo service docker start
```

- 도커 상태 확인

```
service docker status
```

### Docker-Compose 설치

- 도커 컴포즈로 여러 개의 컨테이너 실행, 관리 가능

```
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
sudo chmod +x /usr/local/bin/docker-compose
```

```
ubuntu@ ~$ docker-compose -v
docker-compose version 1.29.2, build 5becea4c
```

## EC2에 카프카 설치

- standalone 모드로 진행하기 때문에 별도의 주키퍼와 카프카 properties를 별도로 설정하지 않아도 된다.
- Java 8 버전 이상 필요

### 1. 자바 다운로드

```
sudo apt install ssh openjdk-8-jdk ant -y
```

### 2. 자바 환경 변수 설정

bashrc 편집 → `export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64` 추가 → bashrc 재실행

```
vi ~/.bashrc
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
source ~/.bashrc
```

```
ubuntu@ ~$ java --version
openjdk 11.0.16 2022-07-19
OpenJDK Runtime Environment (build 11.0.16+8-post-Ubuntu-0ubuntu120.04)
OpenJDK 64-Bit Server VM (build 11.0.16+8-post-Ubuntu-0ubuntu120.04, mixed mode, sharing)
```

### 2. 카프카 tar 파일 다운로드 및 압축 해제

```
wget https://archive.apache.org/dist/kafka/3.2.0/kafka_2.13-3.2.0.tgz
tar -xzf kafka_2.13-3.2.0.tgz
```

```
ubuntu@ ~$ wget https://archive.apache.org/dist/kafka/3.2.0/kafka_2.13-3.2.0.tgz
--2022-10-06 14:41:54-- https://archive.apache.org/dist/kafka/3.2.0/kafka_2.13-3.2.0.tgz
Resolving archive.apache.org (archive.apache.org)... 138.201.131.134, 2a01:4f8:172:2ec5::2
Connecting to archive.apache.org (archive.apache.org)[138.201.131.134]:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 103955943 (99M) [application/x-gzip]
Saving to: 'kafka_2.13-3.2.0.tgz'

kafka_2.13-3.2.0.tgz      100%[=====] 99.14M  2.16MB/s   in 57s
2022-10-06 14:42:52 (1.73 MB/s) - 'kafka_2.13-3.2.0.tgz' saved [103955943/103955943]

ubuntu@ ~$ tar -xzf kafka_2.13-3.2.0.tgz
ubuntu@ ~$ ls
kafka_2.13-3.2.0  kafka_2.13-3.2.0.tgz
```

### 2. 주키퍼 실행

- daemon으로 끊기지 않게 백그라운드에서 실행
- 아래의 QuorumPeerMain이 주키퍼

```
cd kafka_2.13-3.2.0/
nohup bin/zookeeper-server-start.sh config/zookeeper.properties &
```

```
ubuntu@ip-172-26-0-161:~/kafka_2.13-3.2.0$ jps
7411 Jps
7027 QuorumPeerMain
```

## 5. 카프카 실행

- daemon으로 끊기지 않게 백그라운드에서 실행

```
cd kafka_2.13-3.2.0/
nohup bin/kafka-server-start.sh config/server.properties &
```

```
hadoop@ip-172-26-0-161:~/kafka_2.13-3.2.0$ jps
5688 Jps
3115 QuorumPeerMain
5263 Kafka
```

## EC2에 젠킨스 설치

### 1. 젠킨스 도커 이미지 다운로드

```
sudo docker pull jenkins/jenkins:its
```

```
ubuntu@ip-172-26-0-161:~$ sudo docker pull jenkins/jenkins:its
its: Pulling from jenkins/jenkins
1671565cc8df: Pull complete
1e010a8344e7: Pull complete
f7406b2e1315: Pull complete
a7516e8e83d2: Pull complete
a51dca64e82b: Pull complete
77ef07b6a141: Pull complete
2ac030a719df: Pull complete
263bf74244c0: Pull complete
620f54e03b44: Pull complete
59e43d37c904: Pull complete
c9dbe2415122: Pull complete
2c049b4765e9: Pull complete
c2b2538c867b: Pull complete
57c5d5e596fd: Pull complete
Digest: sha256:5508cb1317aa0ede06cb34767fb1ab3860d1307109ade577d5df871f62170214
Status: Downloaded newer image for jenkins/jenkins:its
docker.io/jenkins/jenkins:its
```

### 2. 젠킨스 컨테이너 실행 위한 Dockerfile 작성

```
FROM jenkins/jenkins:its

USER root

# install docker
RUN apt-get update && \
    apt-get -y install apt-transport-https \
        ca-certificates \
        curl \
        gnupg2 \
        zip \
        unzip \
        software-properties-common && \
    curl -fsSL https://download.docker.com/linux/$(. /etc/os-release; echo "$ID")/gpg > /tmp/dkey; apt-key add /tmp/dkey && \
    add-apt-repository \
    "deb [arch=amd64] https://download.docker.com/linux/$(. /etc/os-release; echo "$ID") \
```

```
$(lsb_release -cs) \
stable" && \
apt-get update && \
apt-get -y install docker-ce
```

### 3. 젠킨스 컨테이너 설정 위한 docker-compose.yml 작성

```
version: '3.7'
services:
  jenkins:
    build:
      context: .
    container_name: jenkins
    user: root
    privileged: true
    ports:
      - 3333:8080
      - 50000:50000
    volumes:
      - ./jenkins_home:/var/jenkins_home
      - /var/run/docker.sock:/var/run/docker.sock
```

### 4. 폴더 만들어서 3, 4번 파일 넣기

```
ubuntu@ip-172-26-0-161:~$ mkdir jenkins_build
ubuntu@ip-172-26-0-161:~$ cd jenkins_build/
ubuntu@ip-172-26-0-161:~/jenkins_build$ ls
Dockerfile  docker-compose.yml
ubuntu@ip-172-26-0-161:~/jenkins_build$
```

### 5. docker compose 하기

→ 비밀번호 확인하기 위해 백그라운드 실행은 안 함

```
sudo docker-compose up
```

```
ubuntu@ip-172-26-0-161:~/jenkins_build$ sudo docker-compose up
Creating network "jenkins_build_default" with the default driver
Building jenkins
Sending build context to Docker daemon  3.584kB
Step 1/3 : FROM jenkins/jenkins:its
----> 729c87ece8d0
Step 2/3 : USER root
----> Running in 54768b34eabc
Removing intermediate container 54768b34eabc
----> 8762079e7bbb
Step 3/3 : RUN apt-get update && apt-get -y install apt-transport-https ca-certificates curl
gnupg2 zip unzip software-properties-common && curl -fsSL https://download.docker.com/linux
/$(. /etc/os-release; echo "$ID")/gpg > /tmp/dkey; apt-key add /tmp/dkey && add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/$(. /etc/os-release; echo "$ID") $(lsb_release -cs) stable" && apt-get up
date && apt-get -y install docker-ce
----> Running in a304034e50ee
Get:1 http://deb.debian.org/debian bullseye InRelease [116 kB]
Get:2 http://deb.debian.org/debian-security bullseye-security InRelease [48.4 kB]
Get:3 http://deb.debian.org/debian bullseye-updates InRelease [44.1 kB]
Get:4 http://deb.debian.org/debian bullseye/main amd64 Packages [8184 kB]
```

```
jenkins *****
jenkins *****
jenkins *****
jenkins *****
jenkins Jenkins initial setup is required. An admin user has been created and a password generated.
jenkins Please use the following password to proceed to installation:
jenkins
jenkins 70f53b8800ea4d2ebca5e3ba786c6757
jenkins
jenkins This may also be found at: /var/jenkins_home/secrets/initialAdminPassword
jenkins
jenkins *****
jenkins *****
jenkins *****
jenkins *****
```

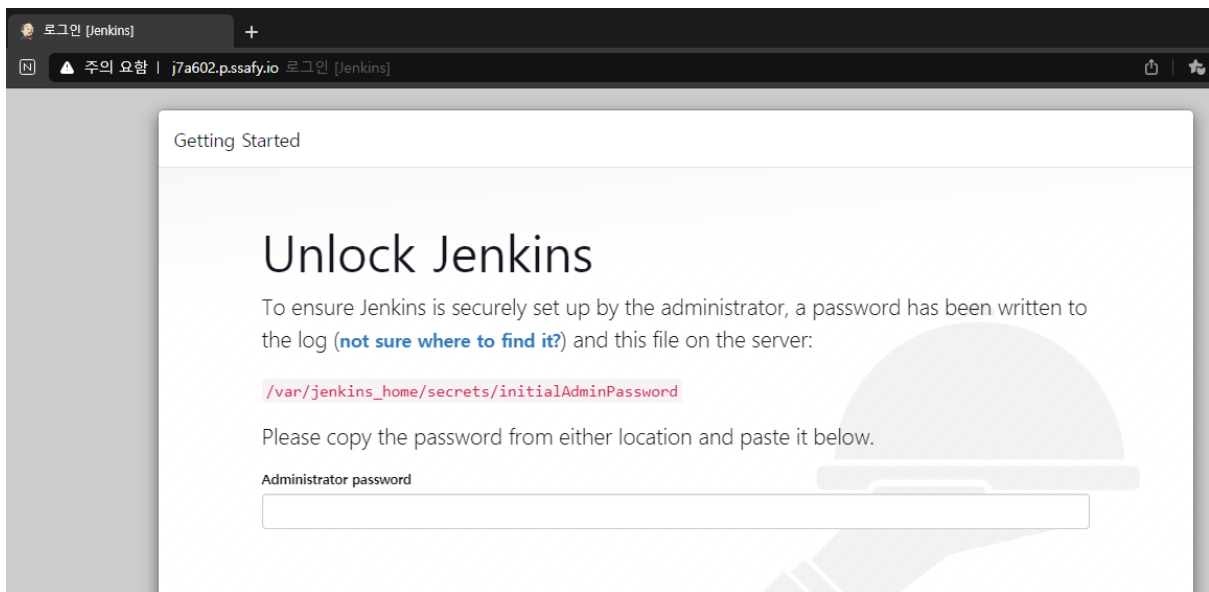
## 6. 비밀번호 저장

70f53b8800ea4d2ebca5e3ba786c6757

## 7. Jenkins 접속

→ jenkins는 인증서 설정이 안되어 있으므로 http로 접속

<http://j7a602.p.ssafy.io:3333>



8. 비밀번호 입력 후 진행 결과에서 왼쪽 클릭



# 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.

### 9. 클릭 후 화면

# Getting Started

✓ Folders	OWASP Markup Formatter	Build Timeout	Credentials Binding	<b>Folders</b> ** JavaBeans Activation Framework (JAF) API ** JavaMail API ** bouncycastle API
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		
				** - required dependency

Jenkins 2.361.1

### 10. 설치 완료 후 계정 생성

## Create First Admin User

계정명:

암호:

암호 확인:

이름:

이메일 주소:

Jenkins 2.361.1

[Skip and continue as admin](#)

[Save and Continue](#)

계정명: consultant  
암호: zxck35JF!jkvfc^%z  
암호확인: zxck35JF!jkvfc^%z  
이름 : consultant  
이메일주소: yoongh97@gmail.com

11. 다음 화면에서 [Save and Finish](#) 클릭

# Instance Configuration

Jenkins URL:

`http://j7a602.p.ssafy.io:3333/`

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the `BUILD_URL` environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Jenkins 2.361.1

Not now

Save and Finish

Getting Started

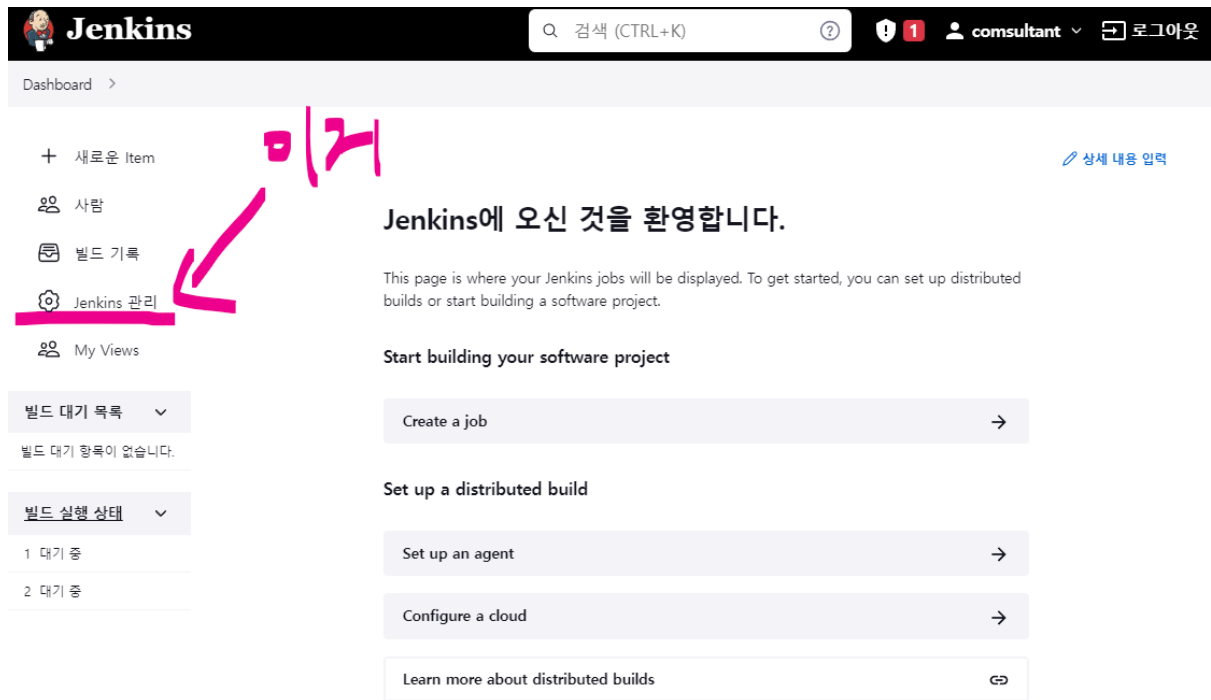
## Jenkins is ready!

Your Jenkins setup is complete.

Start using Jenkins

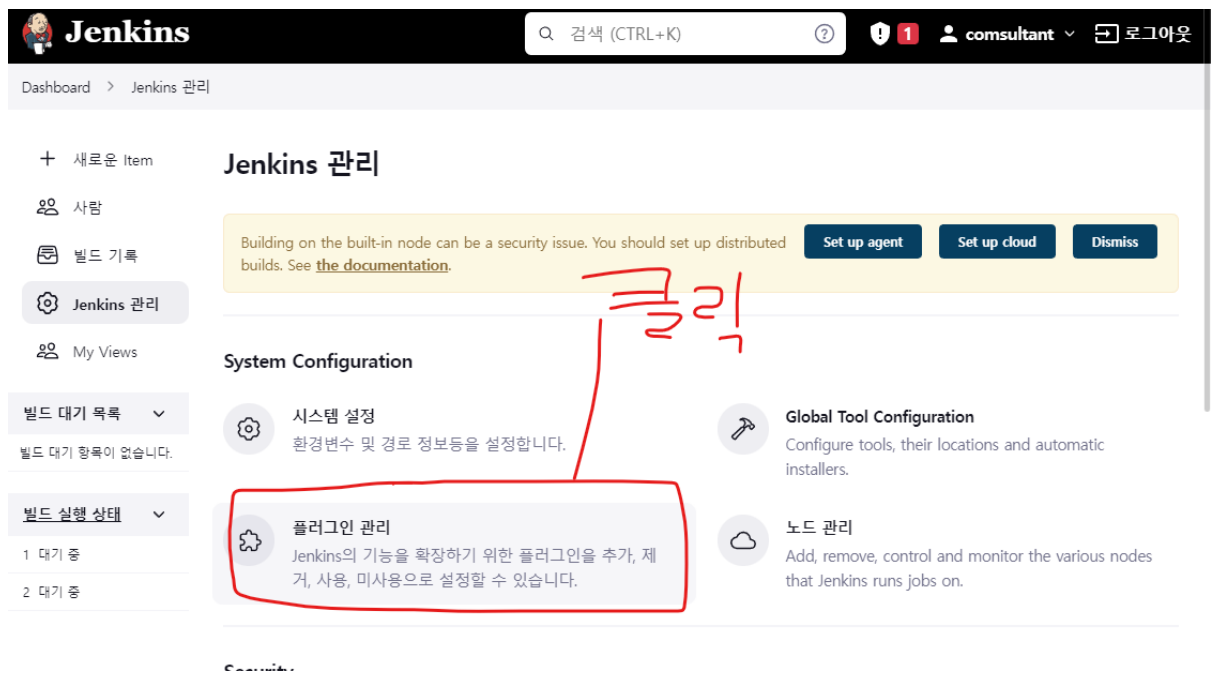
### 젠킨스 설정

1. `Jenkins 관리` 클릭



The screenshot shows the Jenkins Dashboard. The top navigation bar includes the Jenkins logo, a search bar, and user information. The left sidebar contains links to '새로운 Item', '사람', '빌드 기록', 'Jenkins 관리' (highlighted with a red arrow and the handwritten word '미저'), and 'My Views'. The main content area displays a welcome message, instructions on how to get started, and buttons for 'Create a job', 'Set up a distributed build' (with sub-options 'Set up an agent', 'Configure a cloud', and 'Learn more about distributed builds'), and 'Start building your software project'.

## 2. 플러그인 관리 클릭



The screenshot shows the 'Jenkins 관리' (Jenkins Management) page. The left sidebar is the same as the dashboard. The main content area is titled 'Jenkins 관리' and includes a warning about security issues with built-in nodes. Below this, there are three main configuration sections: 'System Configuration' (containing '시스템 설정' and '플러그인 관리' - highlighted with a red box and the handwritten word '클릭'), 'Global Tool Configuration', and '노드 관리'.

## 3. Gitlab 연동 위해 Gitlab 플러그인 설치. React 빌드를 위한 Node JS도 설치

## Plugin Manager

업데이트된 플러그인 목록

설치 가능

설치된 플러그인 목록

고급

Q gitlab

Install	Name ↓	Released
<input checked="" type="checkbox"/>	<b>GitLab</b> 1.5.35 Build Triggers This plugin allows <b>GitLab</b> 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 29 days ago

Q nodejs

Install	Name ↓	Released
<input checked="" type="checkbox"/>	<b>NodeJS</b> 1.5.1 npm NodeJS Plugin executes <b>NodeJS</b> script as a build step.	8 mo 16 days ago

#### 4. Gitlab에 연동하기 위해 Credentials 생성

Jenkins

Dashboard > Jenkins 관리

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.

Security

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

Manage Credentials  
Configure credentials

Configure Credential Providers

Manage Users

클릭!

Jenkins

Dashboard > Jenkins 관리 > Credentials

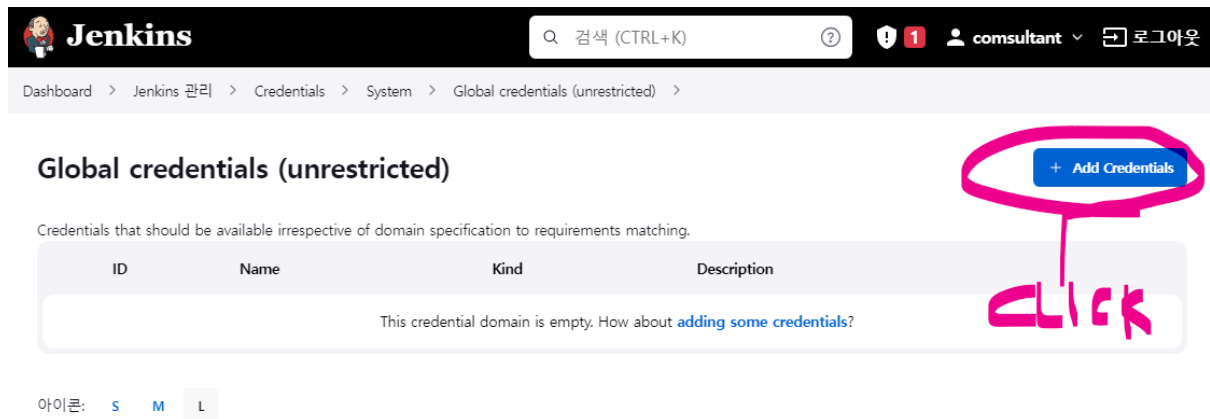
Credentials

T	P	Store ↓	Domain	ID	Name
---	---	---------	--------	----	------

Stores scoped to Jenkins

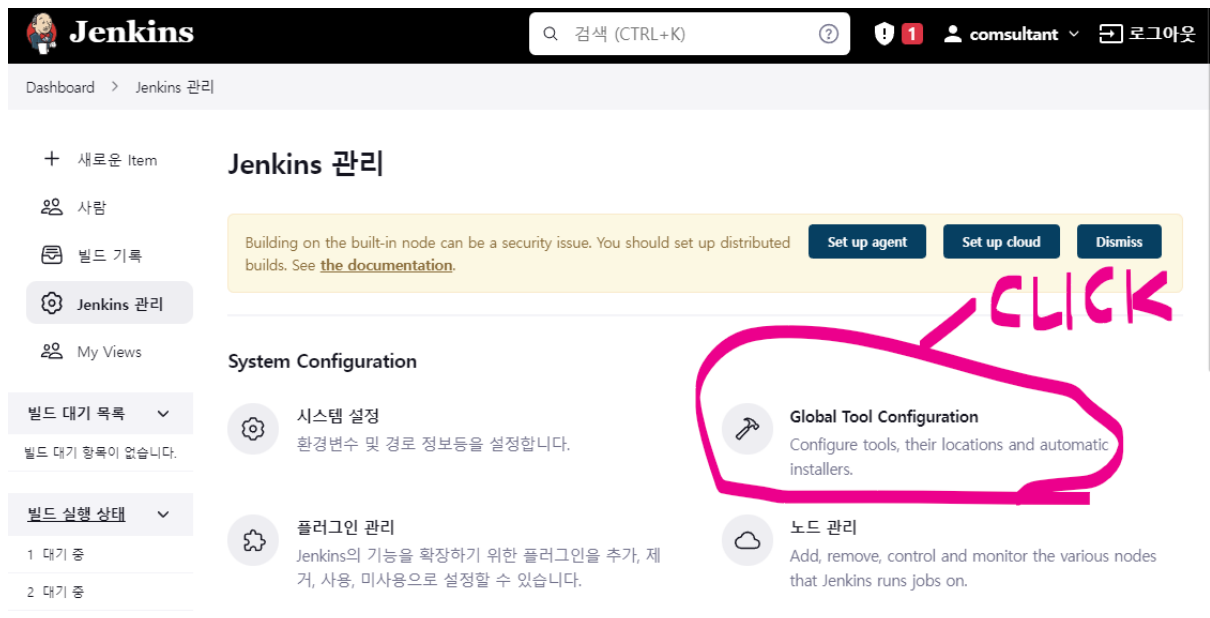
P	Store ↓	Domains
System	(global)	CLICK

아이콘: S M L

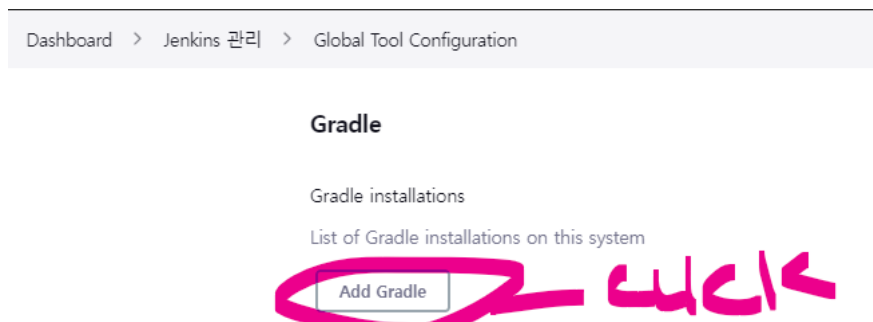


5. 종류에서 **Username with password** 클릭 후 깃랩 이메일과, 비밀번호를 입력하여 생성

6. Gradle 설정 위해 **Global Tool Configuration** 클릭



7. 프로젝트 Gradle이 7.5이므로 버전에 맞게 설치



하단의 Save 클릭



## 8. Gradle 설정 한 것처럼, 같은 페이지에서 JDK랑 NodeJS 버전 설정

NodeJS

Name

node16

☒ Install automatically

Install from nodejs.org

Version

NodeJS 16.16.0

For the underlying architecture, if available, force the installation of the 32bit package. Otherwise the build will fail

☐ Force 32bit architecture

Global npm packages to install

Specify list of packages to install globally -- see npm install -g. Note that you can fix the packages version by using the syntax `packageName@version`

Global npm packages refresh hours

Duration, in hours, before 2 npm cache update. Note that 0 will always update npm cache

72

Add Installer

- JDK 9버전 이후는 없으므로 컨테이너 만들 때 미리 설정해두거나 컨테이너 안에서 따로 설치 필요 (우리는 11버전 사용)

## 9. JDK11 설치

```
ubuntu@ip-172-26-0-161:~/jenkins_build$ docker ps
CONTAINER ID   IMAGE                     COMMAND
d41ef4dd799b   jenkins_build_jenkins    "/usr/bin/tini --
13a4f1176d62   deploy_backend           "java -jar -Dsprin
bd0cd82c7919   mysql:8.0.28             "docker-entrypoint
34b4d5147bc5   redis:alpine             "docker-entrypoint
1545585b6cf2   deploy_nginx             "/docker-entrypoint
```

컨테이너 ID 확인

```
docker exec -itu 0 d41ef4dd799b /bin/sh
```

포팅 매뉴얼

16



## 10. 컨테이너에 루트 권한으로 접속

```
ubuntu@ip-172-26-0-161:~/jenkins_build$ docker exec -itu 0 d41ef4dd799b /bin/sh
# sudo apt-get update
/bin/sh: 1: sudo: not found
# apt-get update
Hit:1 https://download.docker.com/linux/debian bullseye InRelease
Hit:2 http://deb.debian.org/debian bullseye InRelease
Hit:3 http://deb.debian.org/debian-security bullseye-security InRelease
Get:4 http://deb.debian.org/debian bullseye-updates InRelease [44.1 kB]
Hit:5 https://packagecloud.io/github/git-lfs/debian bullseye InRelease
Fetched 44.1 kB in 1s (33.5 kB/s)
Reading package lists... Done
# apt-get install openjdk-11-jdk
```

JDK11 설치

```
Adding debian:Staat_der_Nederlanden_Root_CA_-_G3.pem
Adding debian:QuoVadis_Root_CA_3.pem
Adding debian:emSign_Root_CA_-_G1.pem
Adding debian:AffirmTrust_Commercial.pem
Adding debian:SecureTrust_CA.pem
Adding debian:IdenTrust_Public_Sector_Root_CA_1.pem
Adding debian:Hellenic_Academic_and_Research_Institutions_RootCA_2015.pem
Adding debian:GlobalSign_ECC_Root_CA_-_R5.pem
Adding debian:Microsec_e-Szigno_Root_CA_2009.pem
done.
Processing triggers for libc-bin (2.31-13+deb11u3) ...
Processing triggers for ca-certificates (20210119) ...
Updating certificates in /etc/ssl/certs...
0 added, 0 removed; done.
Running hooks in /etc/ca-certificates/update.d...
done.
done.
#
```

설치 완료

```
# java --version
openjdk 11.0.16.1 2022-08-12
OpenJDK Runtime Environment Temurin-11.0.16.1+1 (build 11.0.16.1+1)
OpenJDK 64-Bit Server VM Temurin-11.0.16.1+1 (build 11.0.16.1+1, mixed mode)
#
```

버전 확인

## 11. 컨테이너에서 나와서 아까 JDK 설정 페이지로 돌아가기

## JDK

### JDK installations

List of JDK installations on this system

Add JDK

≡ JDK

×

Name

jdk11

JAVA\_HOME

/usr/lib/jvm/java-11-openjdk-amd64

☐ Install automatically ?

Add JDK

→ 사진처럼 설정

 /usr/lib/jvm/java-11-openjdk-amd64

12. 젠킨스 안에서 도커 컴포즈 설정을 위해 추가 작업 필요

```
ubuntu@ip-172-26-0-161:~/jenkins_build$ docker exec -it d41ef4dd799b /bin/sh
```

13. 젠킨스 컨테이너에 접속 후 다운

```
curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-c
```

```
# curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           %             %          Dload  Upload  Total   Spent    Left   Speed
  0     0    0     0    0     0      0      0  --:--:-- --:--:-- --:--:--     0
100 12.1M 100 12.1M    0     0 3807k      0  0:00:03  0:00:03 --:--:-- 5401k
```

14. 권한 부여

```
chmod +x /usr/local/bin/docker-compose
```

15. 확인

```
# docker-compose -v
docker-compose version 1.29.2, build 5becea4c
#
```

완료

## EC2에 하둡 설치

- 하둡은 ec2에서 hadoop 계정을 만든 후 hadoop 계정에 설치한다.
- standalone 모드로 실행

### 1. 계정 생성

```
sudo adduser hadoop
```

### 2. hadoop 계정에 sudo 권한 부여

```
sudo usermod -a -G sudo hadoop
```

### 3. hadoop 계정 변경

```
su - hadoop
```

### 4. 설치 및 압축 해제

```
wget http://kdd.snu.ac.kr/~kddlab/Project.tar.gz
tar -zxf Project.tar.gz
```

### 5. 파일 사용자 그룹 변경

```
sudo chown -R hadoop:hadoop Project
```

### 6. hadoop 파일 이동

```
cd Project
sudo mv hadoop-3.2.2 /usr/local/hadoop
```

### 7. 환경 변수 설정 후 재실행

```
./set_hadoop_env.sh
source ~/.bashrc
```

### 8. ssh key 생성

```
ssh-keygen -t rsa -P ""
cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
```

- 아래와 같이 나오면 성공

```

hadoop@:~$ ssh localhost
Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.10.16.3-microsoft-standard-WSL2 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Thu Oct  6 15:32:10 KST 2022

System load:  0.11               Processes:    30
Usage of /:   1.4% of 250.98GB   Users logged in: 0
Memory usage: 16%               IPv4 address for eth0: 172.28.28.175
Swap usage:   0%

34 updates can be applied immediately.
18 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Tue Sep 27 14:16:51 2022 from 127.0.0.1

```

## 9. Name node 포맷

```
hadoop namenode -format
```

## 10. Dfs daemon 실행

```
start-dfs.sh
```

```

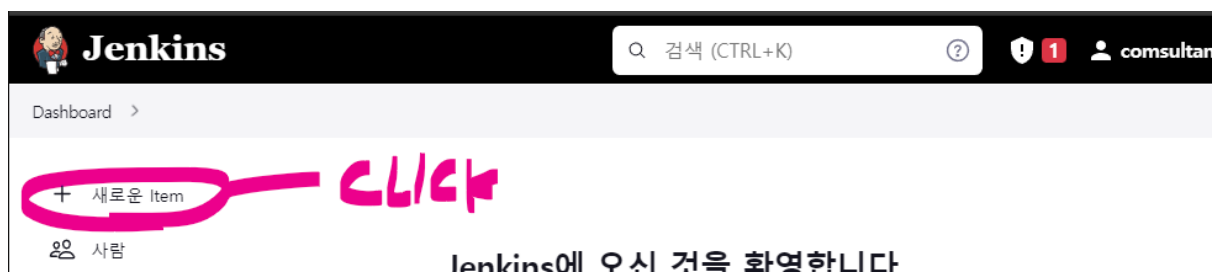
hadoop@:~$ jps
13729 SecondaryNameNode
13857 Jps
13506 DataNode
13324 NameNode

```

## 6. 배포

### 젠킨스 백엔드 CI/CD

1. 새로운 Item 클릭




2. 이름 입력 후 Pipeline 클릭 후 Ok 클릭


**Enter an item name**


Consultant-Back


» 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, 플래폼 특성 빌드, 기타 등등 처럼 다수의 서로다른 환경설정이 필요한 프로젝트에 적합함.

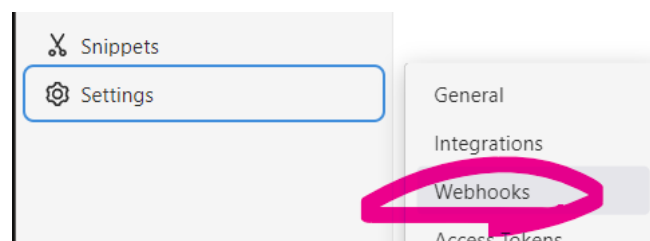
 **Folder**  
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

 **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**

3. Webhook 설정을 하기 위해 GitLab 저장소로 이동 후 Settings - Webhooks 클릭



4. URL에 Jenkins의 내용 작성

## 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.

## Build Triggers

☐ Build after other projects are built ?☐ Build periodically ?☒ Build when a change is pushed to GitLab. GitLab webhook URL: <http://j7a602.p.ssafy.io:3333/project/Consultant-Back> ?

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

## 5. 젠킨스에서 시크릿 토큰 발급 받기

Rebuild open Merge Requests

Never

☒ Approved Merge Requests (EE-only)

☒ Comments

Comment (regex) for triggering a build ?

Jenkins please retry a build

**CLICK**

Secret token ?

Generate  
Clear

복사해서 아래 위치에 붙여넣기

## 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.

## 6. 브랜치 이름 작성

### Trigger

☒ Push events

Push to the repository.

## 7. webhook 추가 후 테스트 진행

Add webhook

CLICK

Project Hooks (1)

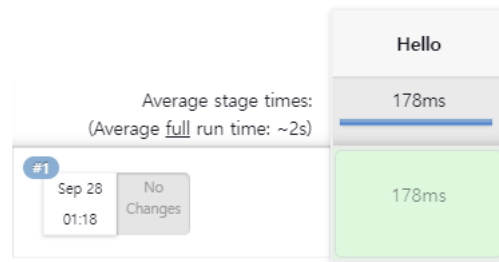
http://j7a602.p.ssafy.io:3333/project/Consultant-Back

Test ▼
Edit
Delete

Push Events
SSL Verification: enabled

## 8. 깃랩에서 확인

## Stage View



### 9. 젠킨스 파이프라인에서 Pipeline script 선택

#### Pipeline

Definition

Pipeline script

Pipeline script

Pipeline script from SCM

### 10. 파이프라인 작성

```
pipeline {
    agent any
    tools {gradle "gradle7.5"}
    stages {
        stage('Prepare') {
            steps {
                echo 'Clonning Repository'
                git url: 'https://lab.ssfy.com/s07-bigdata-dist-sub2/S07P22A602.git',
                    branch: 'be',
                    credentialsId: 'comsultant_gitlab'
            }
            post {
                success {
                    echo 'Successfully Cloned Repository'
                }
                failure {
                    error 'This pipeline stops here...'
                }
            }
        }
        stage('Copy Properties') {
            steps {
                echo 'Copy Properties'
                sh 'cp ./properties/application-mail.yml backend/src/main/resources'
                sh 'cp ./properties/application-prod.yml backend/src/main/resources'
                sh 'cp ./properties/application-social.yml backend/src/main/resources'
            }
            post {
                success {
                    echo 'Successfully Copied'
                }
                failure {
                    error 'This pipeline stops here...'
                }
            }
        }
        stage('Build Gradle') {
            steps {
                echo 'Build Gradle'
                dir ('./backend') {
                    sh """
                        gradle clean build --exclude-task test
                    """
                }
            }
        }
    }
}
```



```

    }
  }
  post {
    success {
      echo 'Successfully Builded'
    }
    failure {
      error 'This pipeline stops here...'
    }
  }
}
stage('Copy Jar') {
  steps {
    echo 'Copy Jar'
    sh 'rm back/backend-0.0.1-SNAPSHOT.jar || true'
    sh 'mv ./backend/build/libs/backend-0.0.1-SNAPSHOT.jar ./back'
  }
  post {
    success {
      echo 'Successfully Copied'
    }
    failure {
      error 'This pipeline stops here...'
    }
  }
}
stage('Compose Down') {
  steps {
    echo 'Down Docker'
    sh 'docker-compose -f docker-compose-back.yml down -v'
    // sh 'docker stop -f $(docker ps -a -q -f name=nginx)'
    // sh 'docker rm -f $(docker ps -a -q -f name=nginx)'
    sh 'docker rmi -f com_backend'
  }
  post {
    success {
      echo 'Successfully Build Down'
    }
    failure {
      error 'This pipeline stops here...'
    }
  }
}
stage('Compose Up') {
  steps {
    echo 'Push Docker'
    sh 'docker-compose -f docker-compose-back.yml up -d'
  }
  post {
    success {
      echo 'Successfully Up'
    }
    failure {
      error 'This pipeline stops here...'
    }
  }
}
}
}
}

```


## 젠킨스 프론트엔드 CI/CD

### 1. 파이프라인 아이템 생성


Enter an item name

Consultant-Front


» 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, 플레폼 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 space, so you can have multiple things of the same name as long as they are in different folders.

OK

2. 백엔드처럼 빌드 트리거 설정

3. 파이프라인 작성

```

pipeline {
    agent any
    tools {nodejs "node16"}

    stages {
        stage('Prepare') {
            steps {
                echo 'Clonning Repository'
                git url: 'https://lab.ssafty.com/s07-bigdata-dist-sub2/S07P22A602.git',
                    branch: 'fe',
                    credentialsId: 'consultant_gitlab'
            }
            post {
                success {
                    echo 'Successfully Cloned Repository'
                }
                failure {
                    error 'This pipeline stops here...'
                }
            }
        }
        stage('Build NodeJS') {
            steps {
                echo 'Build NodeJS'
                dir ('./frontend') {
                    sh """
                        npm install && npm run build
                    """
                }
            }
            post {
                success {
                    echo 'Successfully Buildded'
                }
                failure {
                    error 'This pipeline stops here...'
                }
            }
        }
        stage('Copy Files') {
            steps {
                echo 'Copy FrontEnd Build File'
                sh 'cp frontend/dist/main.js frontend/public'
                sh 'rm -r public || true && mv frontend/public ./'
                sh 'rm -r front || true && mv public front'
            }
        }
    }
}
    
```

```

    }
    post {
        success {
            echo 'Successfully Copied'
        }
        failure {
            error 'This pipeline stops here...'
        }
    }
}
stage('Compose Down') {
    steps {
        echo 'Down Docker'
        sh 'docker-compose -f docker-compose-front.yml down -v'
        // sh 'docker stop -f $(docker ps -a -q -f name=nginx)'
        // sh 'docker rm -f $(docker ps -a -q -f name=nginx)'
        sh 'docker rmi -f com_nginx'
    }
    post {
        success {
            echo 'Successfully Build Down'
        }
        failure {
            error 'This pipeline stops here...'
        }
    }
}
stage('Compose Up') {
    steps {
        echo 'Push Docker'
        sh 'docker-compose -f docker-compose-front.yml up -d'
    }
    post {
        success {
            echo 'Successfully Up'
        }
        failure {
            error 'This pipeline stops here...'
        }
    }
}
}
}
}

```

- credentialsId는 젠킨스 설정에서 만든 credentials이용 → 단, 액세스 토큰으로 하면 접근 거부되므로 Id/pw 로 해야 함!
- 중간에 agent any 작성하면, 다른 폴더에서 실행됨

## CertBot 이용 SSL 인증서 발급

### 1. 파일 전송

```
sudo scp -i J7A602T.pem -r makeCertBot ubuntu@j7a602.p.ssafy.io:/home/ubuntu
```

### 2. 도커 컴포즈

```

ubuntu@ip-172-26-0-161:~/makeCertBot$ docker-compose up
Creating network "certbottest_default" with the default driver
Pulling nginx (nginx:latest)...
latest: Pulling from library/nginx
31b3f1ad4ce1: Pull complete
fd42b079d0f8: Pull complete
30585fbbbec6: Pull complete
18f4ffdd25f4: Pull complete
9dc932c8fba2: Pull complete
600c24b8ba39: Pull complete
Digest: sha256:0b970013351304af46f322da1263516b188318682b2ab1091862497591189ff1
Status: Downloaded newer image for nginx:latest
Pulling certbot (certbot/certbot)...
latest: Pulling from certbot/certbot
339de151aab4: Pull complete
a860e27ad689: Pull complete
910a9a405b4b: Pull complete
bde2ad12a253: Pull complete
c6c8e9f0153d: Pull complete
ee185b36c37e: Pull complete
baedaa7e0794: Pull complete
8a9b412afa74: Pull complete

```

```
e69b0d58d2b4: Pull complete
6e266a1f6d9d: Pull complete
14bb8fb58f70: Pull complete
baaa7e601a29: Pull complete
b55a5f4fe626: Pull complete
Digest: sha256:3103d00d773379cb540aa03b714b999fafc21fd27cf88571e386f656696c4ec
Status: Downloaded newer image for certbot/certbot:latest
Creating certbottest_nginx_1 ... done
Creating certbot ... done
Attaching to certbottest_nginx_1, certbot
nginx_1 | /docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
nginx_1 | /docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
nginx_1 | /docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
nginx_1 | 10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
nginx_1 | 10-listen-on-ipv6-by-default.sh: info: /etc/nginx/conf.d/default.conf differs from the packaged version
nginx_1 | /docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
nginx_1 | /docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
nginx_1 | /docker-entrypoint.sh: Configuration complete; ready for start up
certbot | Account registered.
certbot | Requesting a certificate for j7a602.p.ssafy.io
certbot |
certbot | Successfully received certificate.
certbot | Certificate is saved at: /etc/letsencrypt/live/j7a602.p.ssafy.io/fullchain.pem
certbot | Key is saved at: /etc/letsencrypt/live/j7a602.p.ssafy.io/privkey.pem
certbot | This certificate expires on 2022-12-21.
certbot | These files will be updated when the certificate renews.
certbot | NEXT STEPS:
certbot | - The certificate will need to be renewed before it expires. Certbot can automatically renew the certificate in the background.
certbot | Saving debug log to /var/log/letsencrypt/letsencrypt.log
certbot |
certbot | - - - - -
certbot | If you like Certbot, please consider supporting our work by:
certbot | * Donating to ISRG / Let's Encrypt: https://letsencrypt.org/donate
certbot | * Donating to EFF: https://eff.org/donate-le
certbot | - - - - -
certbot exited with code 0
```

## 결과

```
Attaching to certbottest_nginx_1, certbot
nginx_1 | /docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
nginx_1 | /docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
nginx_1 | /docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
nginx_1 | 10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
nginx_1 | 10-listen-on-ipv6-by-default.sh: info: /etc/nginx/conf.d/default.conf differs from the packaged version
nginx_1 | /docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
nginx_1 | /docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
nginx_1 | /docker-entrypoint.sh: Configuration complete; ready for start up
certbot | Account registered.
certbot | Requesting a certificate for j7a602.p.ssafy.io
certbot |
certbot | Successfully received certificate.
certbot | Certificate is saved at: /etc/letsencrypt/live/j7a602.p.ssafy.io/fullchain.pem
certbot | Key is saved at: /etc/letsencrypt/live/j7a602.p.ssafy.io/privkey.pem
certbot | This certificate expires on 2022-12-21.
certbot | These files will be updated when the certificate renews.
certbot | NEXT STEPS:
certbot | - The certificate will need to be renewed before it expires. Certbot can automatically renew the certificate in the background, but you may need to take steps to enable that functionality. See https://certbot.org/renewal-setup for instructions.
certbot | Saving debug log to /var/log/letsencrypt/letsencrypt.log
certbot |
certbot | - - - - -
certbot | If you like Certbot, please consider supporting our work by:
certbot | * Donating to ISRG / Let's Encrypt: https://letsencrypt.org/donate
certbot | * Donating to EFF: https://eff.org/donate-le
certbot | - - - - -
certbot exited with code 0
^CGracefully stopping... (press Ctrl+C again to force)
Stopping certbottest_nginx_1
```

## 7. 외부 서비스 설정

### 이메일 설정

#### 1. 구글에 가입



#### 2. 2단계 인증 활성화

☒ 보안

2단계 인증
 ☒ 사용 안함

☒ 사용자 미 공유



보안

2단계 인증



사용

### 3. 앱 비밀번호를 생성한다.



보안

앱 비밀번호

없음

## ← 앱 비밀번호

앱 비밀번호를 사용하면 2단계 인증을 지원하지 않는 기기의 앱에서 Google 계정에 로그인할 수 있습니다. 비밀번호를 한 번만 입력하면 기억할 필요가 없습니다. [자세히 알아보기](#)

앱 비밀번호가 없습니다.

앱 비밀번호를 생성할 앱 및 기기를 선택하세요.

CONSULTANT\_SMTP



생성

## 생성된 앱 비밀번호

기기용 앱 비밀번호

bevr ytkz lbku kfnq

사용 방법

설정하려는 애플리케이션 또는 기기의 Google 계정 설정으로 이동합니다. 비밀번호를 위에 표시된 16자리 비밀번호로 교체합니다. 일반적인 비밀번호와 마찬가지로 이 앱 비밀번호는 Google 계정에 대한 완전한 액세스 권한을 부여합니다. 비밀번호를 기억하지 않아도 되므로 적어 놓거나 다른 사용자와 공유하지 마세요.

확인

### 네이버 소셜 로그인 설정

1. 네이버 개발자 사이트에서 애플리케이션을 등록 - 사용 API는 네이버 로그인 선택

## Application

API 이용을 위해 애플리케이션을 등록하고 API 설정을 할 수 있습니다.

내 애플리케이션

애플리케이션 등록

API 제휴 신청

계정 설정

## 애플리케이션 등록 (API 이용신청)

애플리케이션의 기본 정보를 등록하면, 좌측 **내 애플리케이션** 메뉴의 서버 메뉴에 등록하신 애플리케이션 이름으로 서버 메뉴가 만들어집니다.

애플리케이션 이름 ⓘ	<input type="text" value="애플리케이션 이름"/> ⓘ <ul style="list-style-type: none"> <li>네이버 로그인할 때 사용자에게 표시되는 이름이므로 서비스 브랜드를 대표할 수 있는 이름으로 가급적 10자 이내로 간결하게 설정해주세요.</li> <li>40자 이내의 영문, 한글, 숫자, 공백문자, 점표(.), "/", "-", "_", 만 입력 가능합니다.</li> </ul>
사용 API ⓘ	<div> <input type="button" value="선택하세요."/> ⓘ         </div> <p>사용할 API를 추가해 주세요.</p>

- [애플리케이션 이름] 설정을 확인해 주세요.
- [사용 API] 설정을 확인해 주세요.

2. 등록된 애플리케이션을 클릭해서 클라이언트 아이디와 시크릿 확인

## 애플리케이션 정보

Client ID	nCYDIM4yiReKniB6s0eE
Client Secret	<div>.....</div> <div>보기</div>

3. 콜백 URL을 설정하고 저장

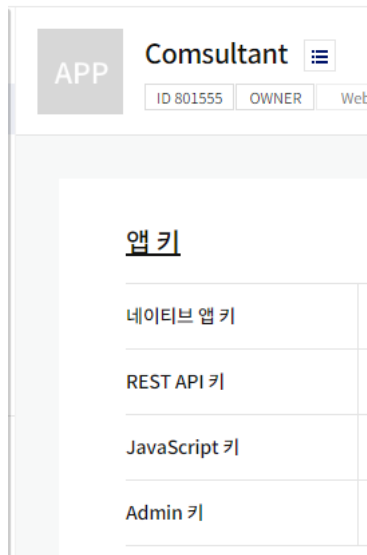
네이버 로그인

Callback URL (최대 5개)

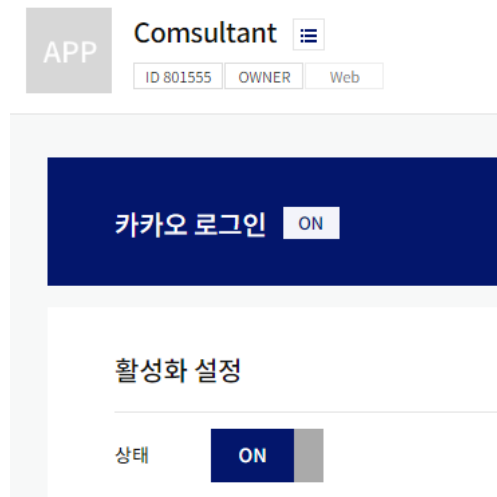
https://j7a602.p.ssafy.io/social/naver	-
http://localhost:3000/social/naver	+

## 카카오 소셜 로그인 설정

1. 네이버와 마찬가지로 애플리케이션을 등록하고 발급받은 키를 확인



2. 카카오 로그인을 활성화시키고 Redirect URI를 설정



## Redirect URI

삭제

수정

Redirect URI	<a href="http://localhost:3000/social/kakao">http://localhost:3000/social/kakao</a> <a href="https://j7a602.p.ssafy.io/social/kakao">https://j7a602.p.ssafy.io/social/kakao</a>
--------------	--

## 구글 소셜 로그인 설정

1. [Google API 콘솔](#)에 접속
2. 새 프로젝트 만들기
3. Google+ API를 사용
4. OAuth 동의화면을 구성

API	API 및 서비스	OAuth 동의 화면
⚙	사용 설정된 API 및 서비스	Consultant <a href="#">앱 수정</a>
📖	라이브러리	
🔑	사용자 인증 정보	게시 상태 <a href="#">?</a>
🌐	OAuth 동의 화면	테스트
🏠	도메인 확인	<a href="#">앱 게시</a>
📄	페이지 사용 동의	사용자 유형

5. 순서대로 설정

1 OAuth 동의 화면 — 2 범위 — 3 테스트 사용자 — 4 요약

6. 사용자 인증 정보에서 클라이언트 ID를 생성

🔑 사용자 인증 정보

🌐 OAuth 동의 화면

🏠 도메인 확인

📄 페이지 사용 동의

☐ 이름

표시할 API 키가 없습니다.

OAuth 2.0 클라이언트 ID

☐ 이름

☐ [Consultant](#)

7. 클라이언트 ID 설정에서 리디렉션 URI도 설정

### 승인된 리디렉션 URI [?](#)

웹 서버의 요청에 사용

URI 1 \*  
https://j7a602.p.ssafy.io/social/google

URI 2 \*  
http://localhost:3000/social/google

[+ URI 추가](#)

## 8. Jenkins Workspace 설정

### 8-1 Backend

```
- /Consultant-Back
├─ /backend ( 깃랩 클론 파일 )
├─ /properties
├─ application-social.yml
└─ application-prod.yml
```



```
└─ application-mail.yml
└─ /back
└─ backend-0.0.1-SNAPSHOT.jar
└─ Dockerfile
└─ docker-compose-back.yml
```

## 8-2 Frontend

```
- /Consultant-Front
└─ /frontend ( 깃랩 클론 파일 )
└─ /nginx
└─ Dockerfile
└─ default.conf
└─ /front
└─ /assets
└─ index.html
└─ main.js
└─ robots.txt
└─ docker-compose-front.yml
```

## 9. Docker Network 생성

```
docker network create --driver=bridge com_net
```