빌드 및 정리 문서

1. 사용 기술 스택

기술 스택

Aa 분류	이름	버전
SERVER	AWS	Ubuntu 20.04 LTS (GNU/Linux 5.4.0-1018-aws x86_64)
제목 없음	Docker	20.10.23
제목 없음	openVidu	2.25.0
CI/CD	Jenkins	2.375.2
DB	Redis	7.0.8
제목 없음	MySql	8.0.32
FE	VisualStudioCode	1.74.2
제목 없음	nodeJS	16.13.2
제목 없음	npm	8.1.2
제목 없음	React	6
BE	intelliJ	2022.3.1
제목 없음	Spring-Boot	2.7.7
제목 없음	JAVA	11
제목 없음	Gradle	7.6

FE 설정 파일

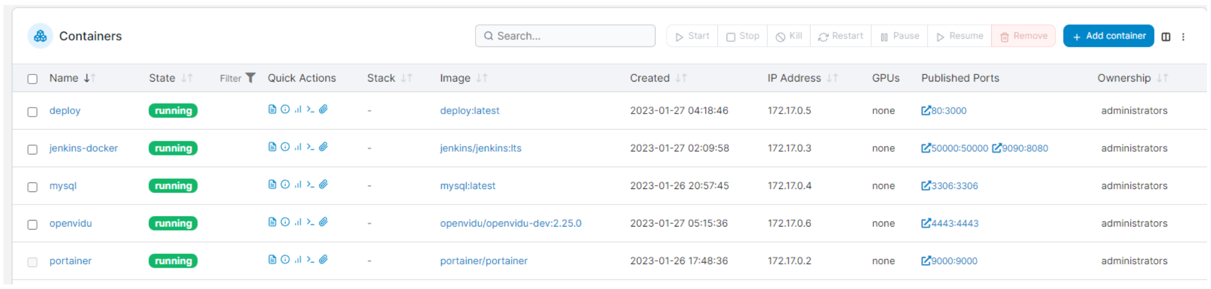
```
// .env
REACT_APP_API_URL=https://i8e104.p.ssafy.io/api
REACT_APP_APPLICATION_SERVER_URL=https://i8e104.p.ssafy.io/

REACT_APP_GOOGLE_REDIRECT_URI=https://i8e104.p.ssafy.io/login/google
REACT_APP_GOOGLE_REST_KEY=148458737954-dlel68c7r0p1b6k5f3fa0v0jugqhte9v.apps.googleusercontent.com

REACT_APP_KAKAO_JS_KEY=fc5f834b5ad79978e1b16032d5303873
REACT_APP_KAKAO_REST_KEY=7523ac59284fff835cf86b20b76876b7
REACT_APP_KAKAO_REDIRECT_URI=https://i8e104.p.ssafy.io/login/kakao

REACT_APP_NAVER_REDIRECT_URI=https://i8e104.p.ssafy.io/login/naver
REACT_APP_NAVER_REST_KEY=Nz3LX8iCIQg8eJPAOPuo

REACT_APP_PROD_CLIENT_URL=https://i8e104.p.ssafy.io
```

BE 설정 파일

```
# application.properties
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.datasource.url=jdbc:mysql://i8e104.p.ssafy.io:3306/patpatDB?serverTimezone=Asia/Seoul&characterEncoding=UTF-8
spring.datasource.username=root
spring.datasource.password=patpat104

spring.main.allow-circular-references=true

spring.jpa.open-in-view=false
```

```
# file-upload
spring.servlet.multipart.max-request-size=200MB
spring.servlet.multipart.max-file-size=200MB


#app.fileupload.uploadPath=C:\\Users\\SSAFY\\Desktop\\leeflection\\S08P12E104\\backend\\src\\main\\resources\\static
#app.fileupload.uploadPath=C:\\Users\\SSAFY\\Desktop\\ssafy\\test
# app.fileupload.uploadPath=C:\\Users\\User\\Desktop\\ssafy\\patpat\\S08P12E104\\backend\\src\\main\\resources\\static
app.fileupload.uploadPath=static
app.fileupload.uploadDir=upload

#logging
logging.level.com.ssafy.patpat=DEBUG

jwt.header=Authorization
jwt.secret=bXNoLWt5bS1jY2cta2p5LWxqaC1qa2gtcGF0cGF0LWdvb2QtbG9zdGFyay13aGF0aXNnb29kLXNzYWZ5LXNsZWVwLW5ld2plYW5zLWZpZ2h0aW5n
#
jwt.access-token-validity-in-seconds=36000000
#
jwt.refresh-token-validity-in-seconds=360000000
# 김씨꺼 95fbb288eef0efa602e9ea13e27cb4fb
# 내꺼 7523ac59284fff835cf86b20b76876b7
spring.security.oauth2.client.registration.kakao.client-id=7523ac59284fff835cf86b20b76876b7
spring.security.oauth2.client.registration.kakao.client-authentication-method=POST
spring.security.oauth2.client.registration.kakao.authorization-grant-type=authorization_code
spring.security.oauth2.client.registration.kakao.scope=profile_nickname, profile_image, account_email, age_range
spring.security.oauth2.client.registration.kakao.client-name=kakao
spring.security.oauth2.client.provider.kakao.authorization-uri = https://kauth.kakao.com/oauth/authorize
spring.security.oauth2.client.provider.kakao.token-uri=https://kauth.kakao.com/oauth/token
spring.security.oauth2.client.provider.kakao.user-info-uri=https://kapi.kakao.com/v2/user/me
spring.security.oauth2.client.provider.kakao.user-name-attribute=id

spring.security.oauth2.client.registration.naver.client-id=Nz3LX8iCIQg8eJPAOPuo
spring.security.oauth2.client.registration.naver.client-secret=Ig6tqHs28c
spring.security.oauth2.client.registration.naver.authorization-grant-type=authorization_code
spring.security.oauth2.client.provider.naver.token-uri=https://nid.naver.com/oauth2.0/token
spring.security.oauth2.client.provider.naver.user-info-uri=https://openapi.naver.com/v1/nid/me
spring.security.oauth2.client.provider.naver.authorization-uri = https://nid.naver.com/oauth/authorize

spring.security.oauth2.client.registration.google.client-id = 148458737954-dlel68c7r0p1b6k5f3fa0v0jugqhte9v.apps.googleusercontent.com
spring.security.oauth2.client.registration.google.client-secret = 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
spring.security.oauth2.client.registration.google.scope = profile,email,openid
spring.security.oauth2.client.registration.google.authorization-grant-type=authorization_code
spring.security.oauth2.client.provider.google.authorization-uri=https://oauth2.googleapis.com
spring.security.oauth2.client.provider.google.token-uri=https://oauth2.googleapis.com/token
spring.security.oauth2.client.provider.google.user-info-uri=https://www.googleapis.com/oauth2/v1/userinfo?alt=json

spring.redis.port=6379
spring.redis.host=i8e104.p.ssafy.io
spring.redis.password=patpat104
# spring.redis.username=moski

openvidu.url=https://i8e104.p.ssafy.io:8443/
openvidu.secret=PATPAT

# 배포시 사용
spring.jpa.show-sql=false
#spring.jpa.show-sql=true
# 배포시 사용
#spring.jpa.hibernate.ddl-auto=none
spring.jpa.hibernate.ddl-auto=update
# 배포시 사용
spring.jpa.properties.hibernate.format_sql=false
#spring.jpa.properties.hibernate.format_sql=true
# 배포시 사용
#app.filecall.url=https://i8e104.p.ssafy.io/api/img
app.filecall.url=http://i8e104.p.ssafy.io:8081/api/img
#배포시 사용
#spring.security.oauth2.client.registration.google.redirect-uri=https://i8e104.p.ssafy.io/login/google
spring.security.oauth2.client.registration.google.redirect-uri=http://localhost:3000/login/google
#배포시 사용
#spring.security.oauth2.client.registration.kakao.redirect-uri=https://i8e104.p.ssafy.io/login/kakao
spring.security.oauth2.client.registration.kakao.redirect-uri=http://localhost:3000/login/kakao
#배포시 사용
#spring.security.oauth2.client.registration.naver.redirect-uri=https://i8e104.p.ssafy.io/login/naver
spring.security.oauth2.client.registration.naver.redirect-uri=http://localhost:3000/login/naver
```

배포



Name	State	Quick Actions	Stack	Image	Created	IP Address	GPUs	Published Ports	Ownership
deploy	running		-	deploy:latest	2023-01-27 04:18:46	172.17.0.5	none	80:3000	administrators
jenkins-docker	running		-	jenkins/jenkins:lts	2023-01-27 02:09:58	172.17.0.3	none	50000:50000 9090:8080	administrators
mysql	running		-	mysql:latest	2023-01-26 20:57:45	172.17.0.4	none	3306:3306	administrators
openvidu	running		-	openvidu/openvidu-dev:2.25.0	2023-01-27 05:15:36	172.17.0.6	none	4443:4443	administrators
portainer	running		-	portainer/portainer	2023-01-26 17:48:36	172.17.0.2	none	9000:9000	administrators

포트번호 : 용도

80 : 배포용

9090 : jenkins

3306 : mysql

4443 : openvidu

9000 : portainer

로 사용중입니다.

docker 설치

1. 패키지 툴 업데이트

```
sudo apt-get update
```

2. 도커 레포지토리 설치

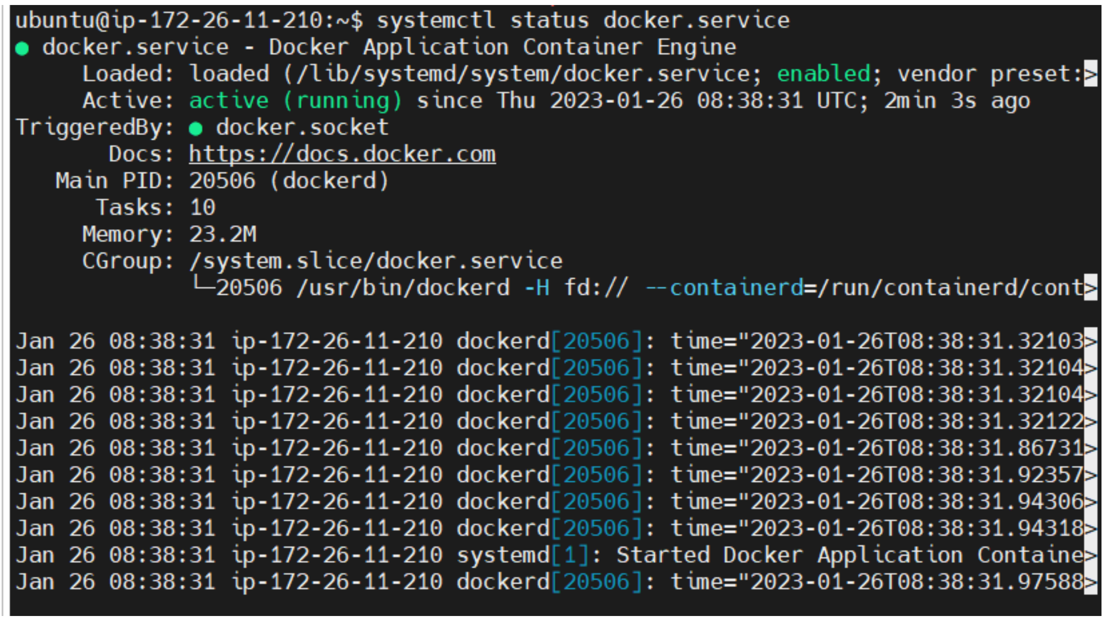
```
sudo apt-get install ca-certificates curl gnupg lsb-release

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

sudo apt-get update
```

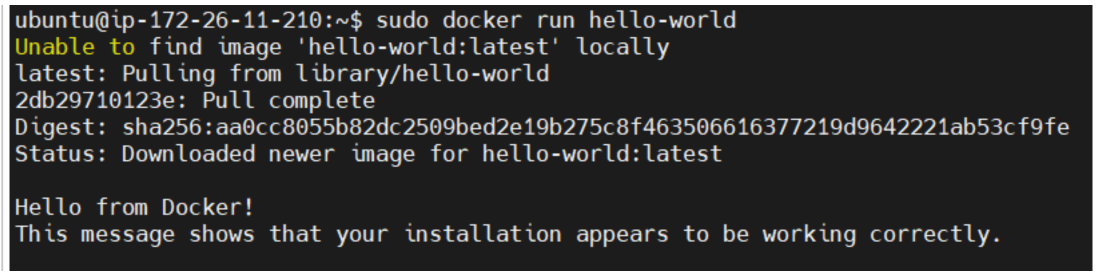
3. 도커 엔진 설치

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

4. 실행 확인하기

```
systemctl status docker.service

sudo docker run hello-world
```


```
ubuntu@ip-172-26-11-210:~$ systemctl status docker.service
● docker.service - Docker Application Container Engine
     Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset:>
     Active: active (running) since Thu 2023-01-26 08:38:31 UTC; 2min 3s ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 20506 (dockerd)
      Tasks: 10
     Memory: 23.2M
     CGroup: /system.slice/docker.service
             └─20506 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/cont>

Jan 26 08:38:31 ip-172-26-11-210 dockerd[20506]: time="2023-01-26T08:38:31.32103>
Jan 26 08:38:31 ip-172-26-11-210 dockerd[20506]: time="2023-01-26T08:38:31.32104>
Jan 26 08:38:31 ip-172-26-11-210 dockerd[20506]: time="2023-01-26T08:38:31.32104>
Jan 26 08:38:31 ip-172-26-11-210 dockerd[20506]: time="2023-01-26T08:38:31.32122>
Jan 26 08:38:31 ip-172-26-11-210 dockerd[20506]: time="2023-01-26T08:38:31.86731>
Jan 26 08:38:31 ip-172-26-11-210 dockerd[20506]: time="2023-01-26T08:38:31.92357>
Jan 26 08:38:31 ip-172-26-11-210 dockerd[20506]: time="2023-01-26T08:38:31.94306>
Jan 26 08:38:31 ip-172-26-11-210 dockerd[20506]: time="2023-01-26T08:38:31.94318>
Jan 26 08:38:31 ip-172-26-11-210 systemd[1]: Started Docker Application Containe>
Jan 26 08:38:31 ip-172-26-11-210 dockerd[20506]: time="2023-01-26T08:38:31.97588>
```


```
ubuntu@ip-172-26-11-210:~$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:aa0cc8055b82dc2509bed2e19b275c8f463506616377219d9642221ab53cf9fe
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.
```

설치 확인 완료!!

Portainer 설치 (9000 port)

- portainer : Docker 를 웹상에서 관리할 수 있게 도와주는 툴

1. portainer에서 사용할 volume 생성

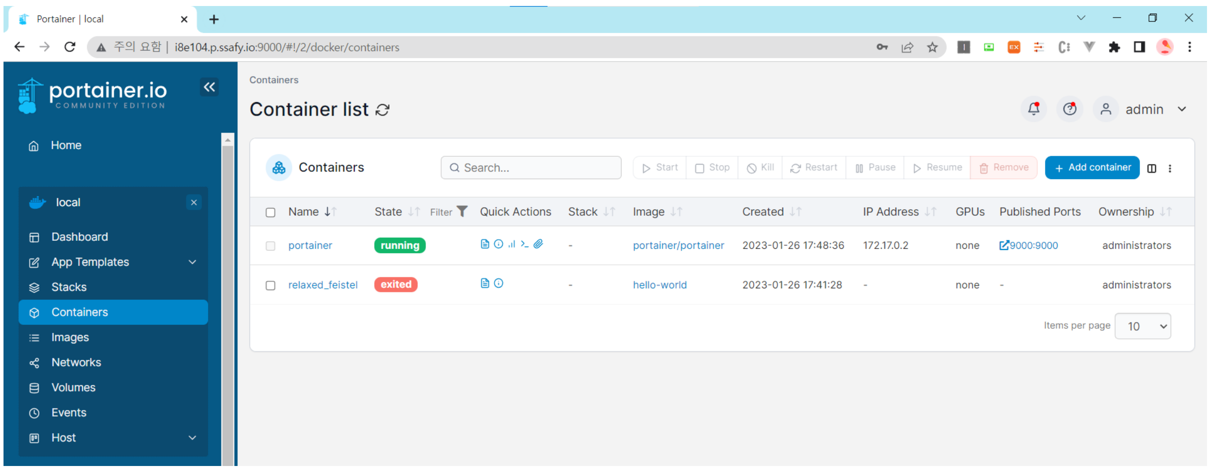
```
docker volume create portainer_data
```

2. 이미지 다운로드 및 컨테이너 생성 후 실행

```
docker run -d -p 9000:9000 -v /var/run/docker.sock:/var/run/docker.sock \
 -v portainer_data:/data --name portainer --restart=always portainer/portainer
```

- -d : 백그라운드 실행
- -p : 외부 포트와 내부 포트 연결
- -v : 데이터 바운딩
- —name : 컨테이너 이름 지정
- —restart : 재시작시 실행 여부

http://i8e104.p.ssafy.io:9000/
접속시 portainer 사용가능



Name	State	Quick Actions	Stack	Image	Created	IP Address	GPUs	Published Ports	Ownership
portainer	running		-	portainer/portainer	2023-01-26 17:48:36	172.17.0.2	none	9000:9000	administrators
relaxed_feistel	exited		-	hello-world	2023-01-26 17:41:28	-	none	-	administrators

Items per page 10

user : admin

pwd : patpat104104

MySQL 설치(3306 port)

- 서버에서 사용할 db 연결

1. mysql 이미지 다운

```
sudo docker pull mysql
```

2. 컨테이너 생성 및 실행

```
sudo docker run --name mysql -e MYSQL_ROOT_PASSWORD=patpat104 -d -p 3306:3306 mysql:latest
```

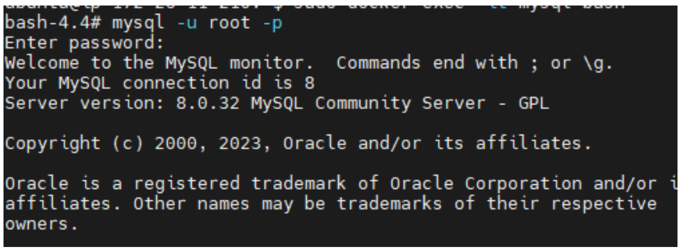
-e MYSQL_ROOT_PASSWORD=<password> : mysql root계정의 비밀번호 설정. root 계정 연결 시 사용

3. mysql 컨테이너 접속

```
sudo docker exec -it <컨테이너 이름> bash
// 컨테이너 이름. 여기같은 경우 mysql
```

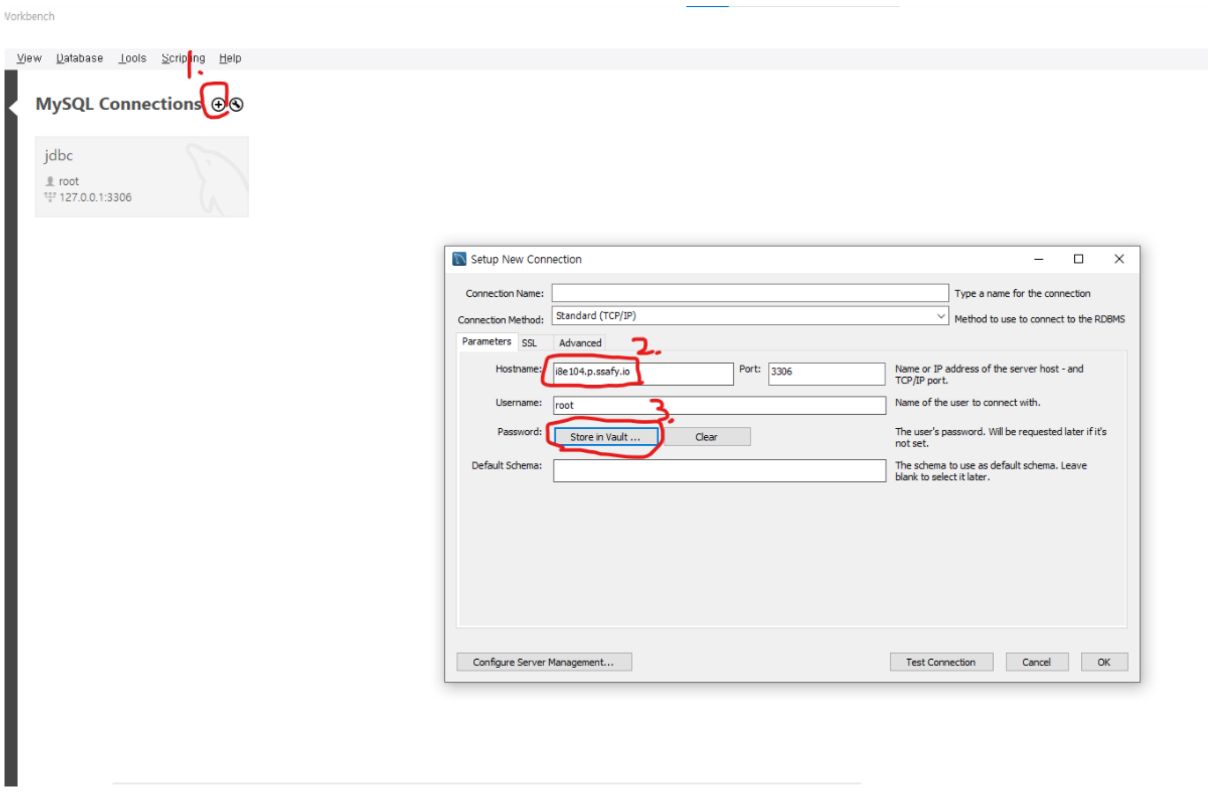
4. mysql root 계정 접속 가능

```
mysql -u root -p

// 이후 Enter password 등장시 위에 2번에서 입력했던 비밀번호 입력
```



```
bash-4.4# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.32 MySQL Community Server - GPL

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or i
affiliates. Other names may be trademarks of their respective
owners.
```

- 여기서 db만들고 입력도 가능하지만 간편하게 workbench 사용



jdbc
root
127.0.0.1:3306

Setup New Connection

Connection Name: — Type a name for the connection

Connection Method: Standard (TCP/IP) — Method to use to connect to the RDBMS

Hostname: i8e104.p.ssafy.io Port: 3306 — Name or IP address of the server host - and TCP/IP port.

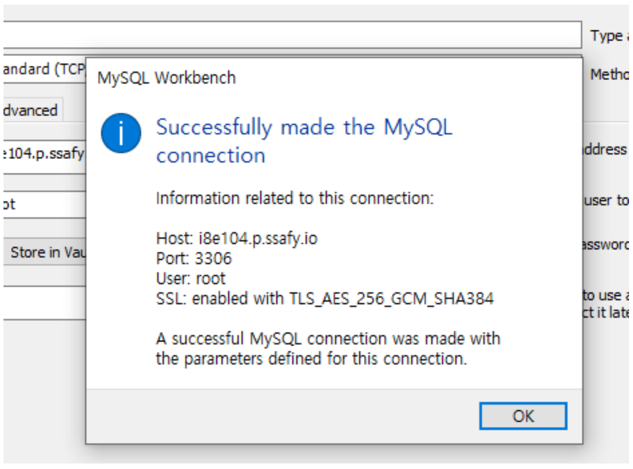
Username: root — Name of the user to connect with.

Password: — The user's password. Will be requested later if it's not set.

Default Schema: — The schema to use as default schema. Leave blank to select it later.

1. + 선택
2. 도메인 입력 (포트는 고정)
3. 사전에 입력했던 비밀번호 입력

- 이후 test connection 클릭하고 연결 확인



MySQL Workbench

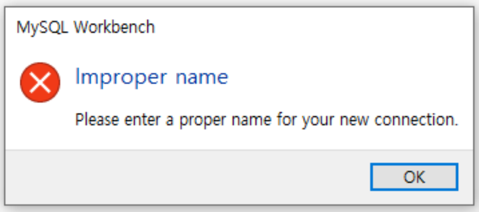
Successfully made the MySQL connection

Information related to this connection:

Host: i8e104.p.ssafy.io
Port: 3306
User: root
SSL: enabled with TLS_AES_256_GCM_SHA384

A successful MySQL connection was made with the parameters defined for this connection.

- 이름을 안적어서 이런 에러가 떴었다.


MySQL Workbench

Improper name

Please enter a proper name for your new connection.

Jenkins 설치(9090 port)

- Jenkins : CI/CD를 편하게 해주는 툴. 소프트웨어 개발시 지속적인 통합 서비스를 제공한다.(자동 배포등등)
- front - React
- back - spring boot(gradle)

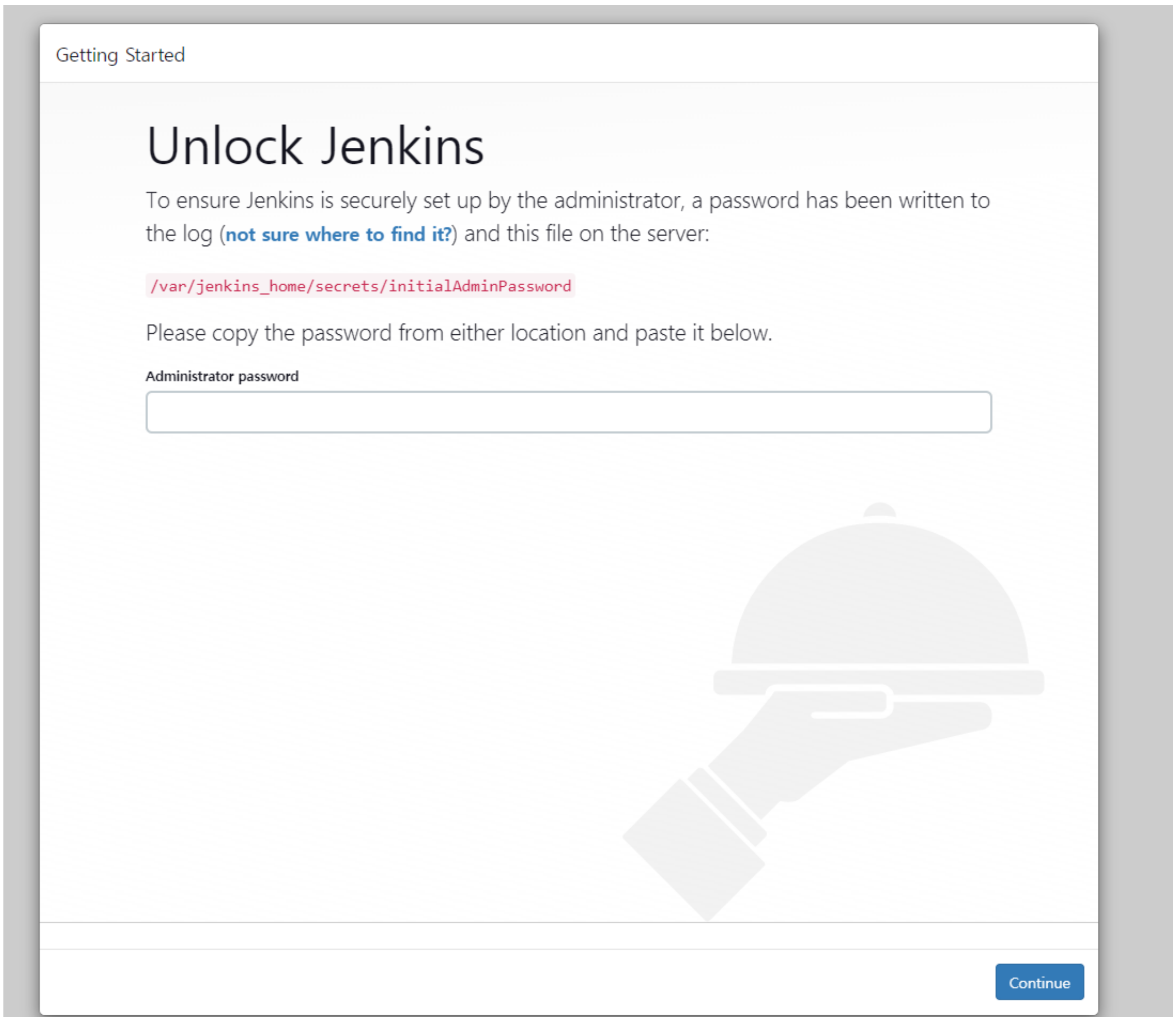
1. jenkins 이미지 다운

```
sudo docker pull jenkins/jenkins:lts
```

2. 컨테이너 생성 및 실행

```
sudo docker run --name jenkins-docker -d -p 9090:8080 -p 50000:50000 -v /home/jenkins:/var/jenkins_home -u root jenkins/jenkins:lts
```

http://i8e104.p.ssafy.io:8080/ 접속시 초기 화면


Getting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log (**not sure where to find it?**) and this file on the server:

`/var/jenkins_home/secrets/initialAdminPassword`

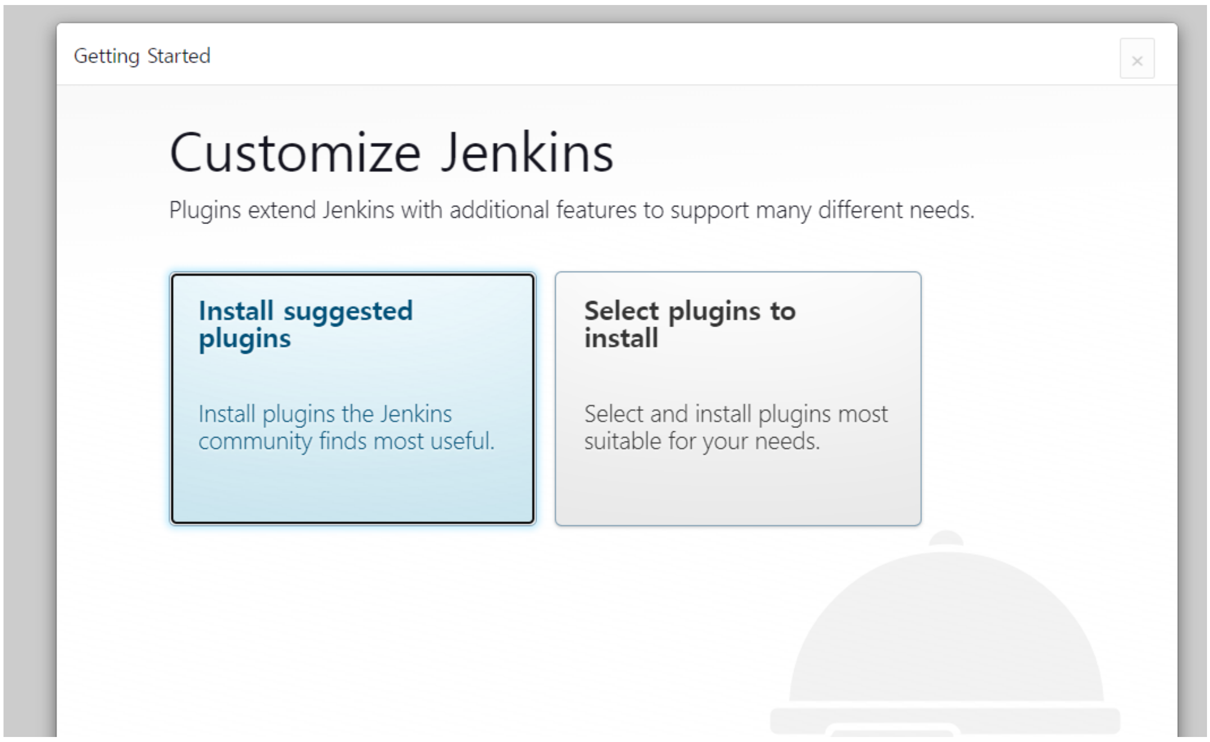
Please copy the password from either location and paste it below.

Administrator password

3. 비밀번호 확인

```
sudo cat /home/jenkins/secrets/initialAdminPassword
```

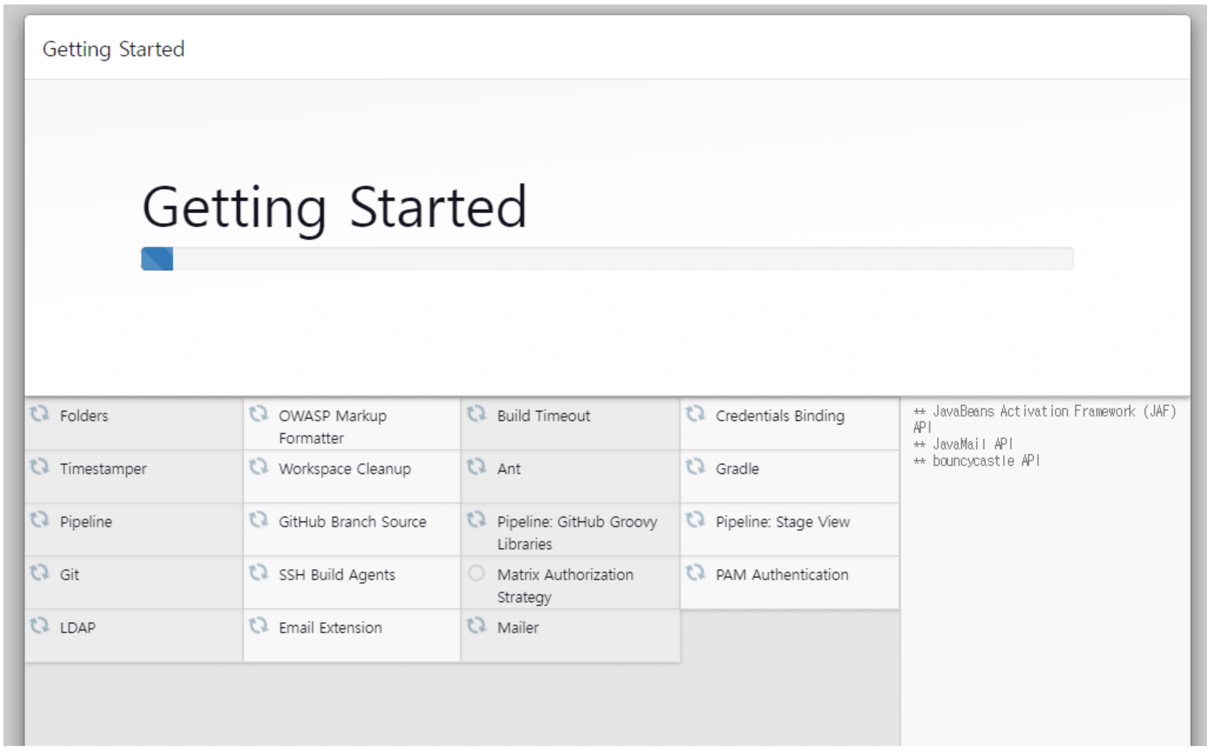
4. install suggested plugins 선택


Getting Started

Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

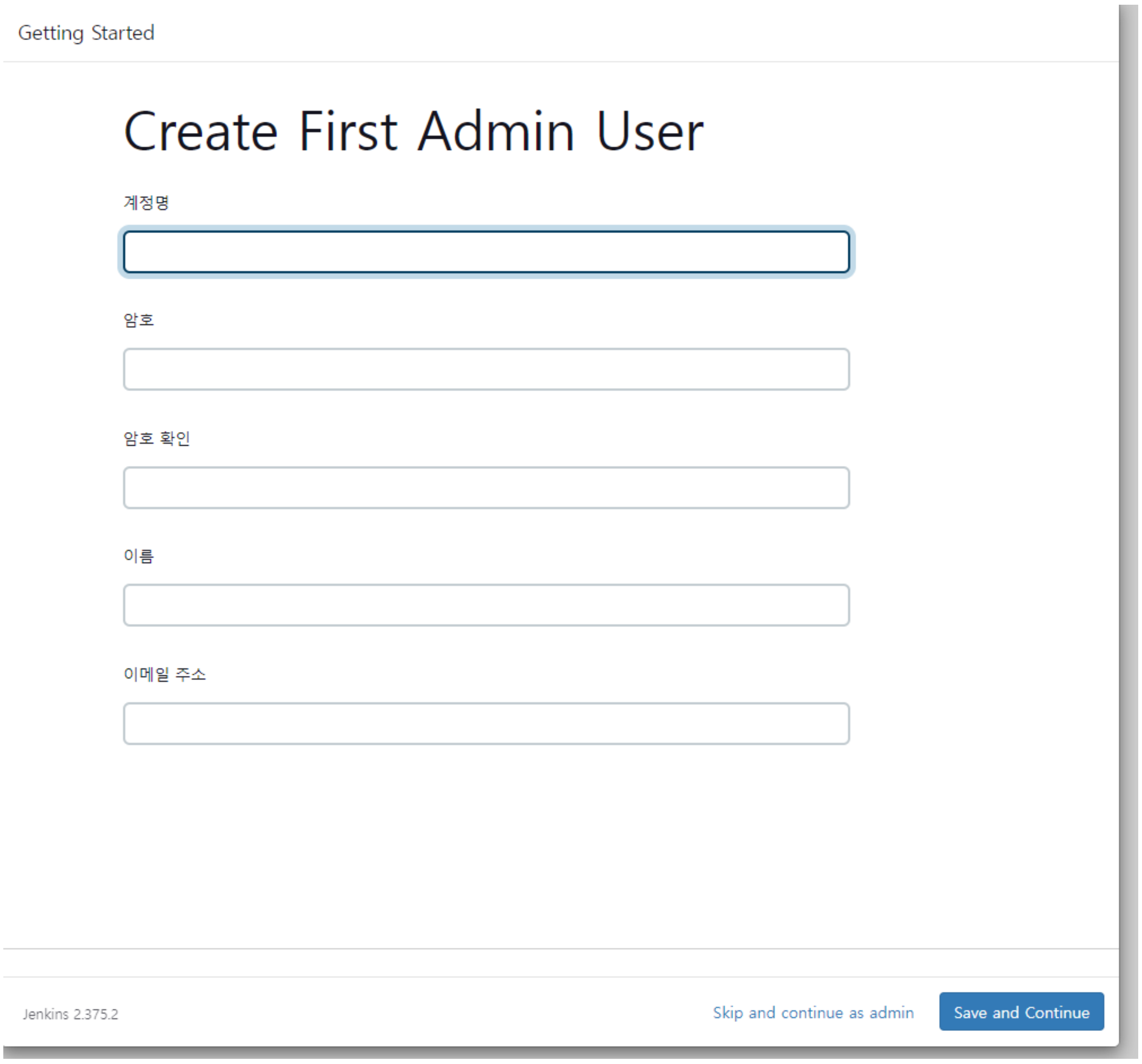
Install suggested plugins
Install plugins the Jenkins community finds most useful.
Select plugins to install
Select and install plugins most suitable for your needs.


Getting Started

Getting Started

Folders	OWASP Markup Formatter	Build Timeout	Credentials Binding	** JavaBeans Activation Framework (JAF) API ** JavaMail API ** bouncycastle API
Timestamper	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		

5. 계정 생성



Getting Started

Create First Admin User

계정명

암호

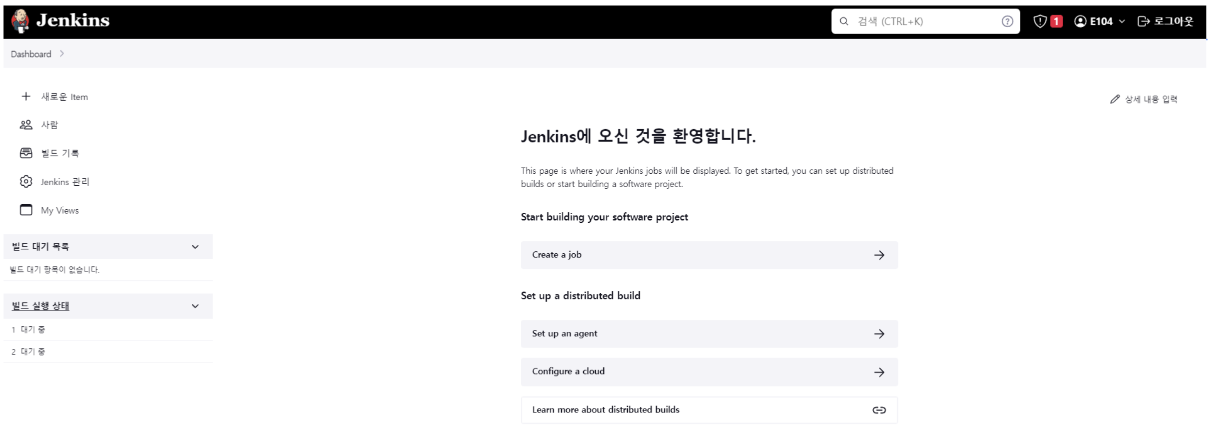
암호 확인

이름

이메일 주소

Jenkins 2.375.2

6. 접속 URL 생성 (바로 다음을 누른다.) 후 메인화면



빌드 대기 항목이 없습니다.

1 대기 중

2 대기 중

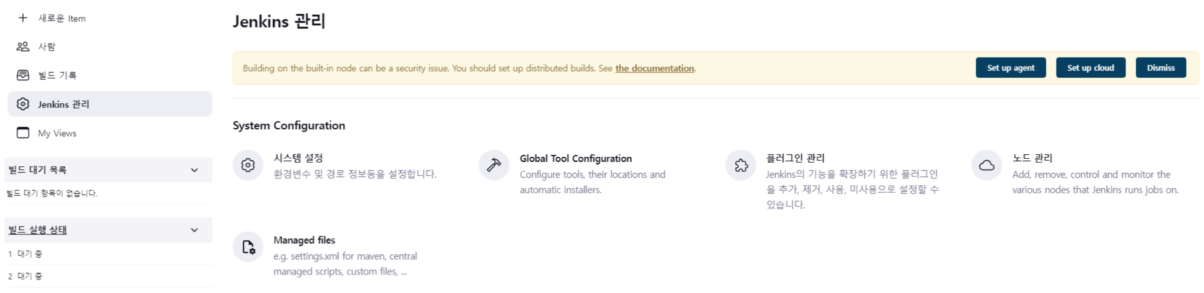
Jenkins에 오신 것을 환영합니다.

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

Set up a distributed build

7. 플러그인 설치



빌드 대기 항목이 없습니다.

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.

System Configuration

시스템 설정
환경변수 및 경로 정보등을 설정합니다.

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

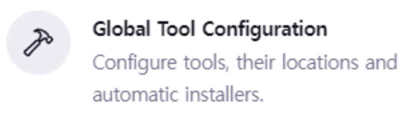
플러그인 관리
Jenkins의 기능을 확장하기 위한 플러그인을 추가, 제거, 사용, 미사용으로 설정할 수 있습니다.

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

Managed files
e.g. settings.xml for maven, central managed scripts, custom files, ...

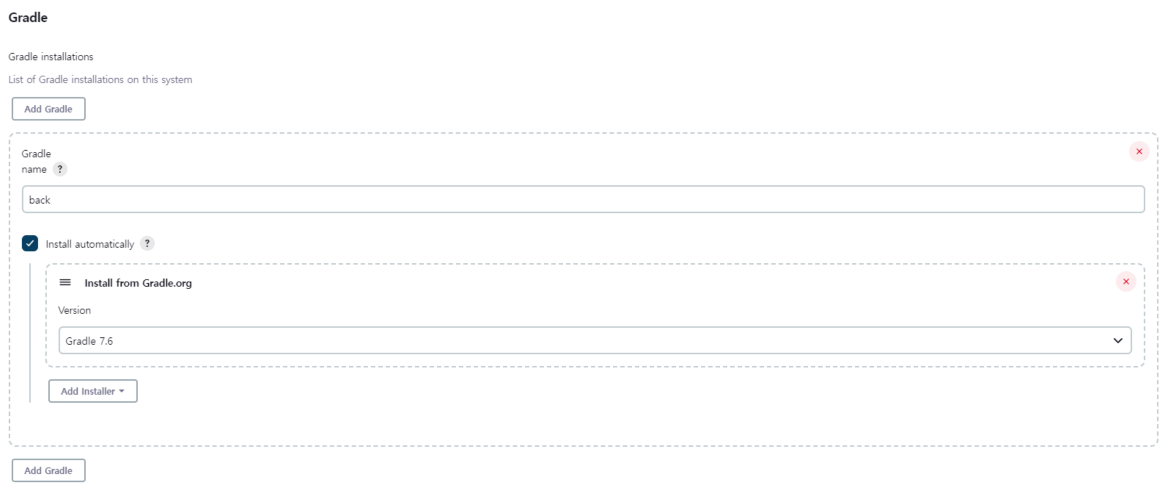
- 플러그인 관리 클릭
- Available plugins에서 아래 플러그인 설치
  - publish Over ssh
  - NodeJS Plugin
  - Generic Webhook Trigger Plugin
  - Gitlab API Plugin
  - GitLab Authentication plugin
  - GitLab Branch Source Plugin
  - GitLab Plugin

8. 기타 환경 설정



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

- Jenkins 관리 - Global Tool Configuration
- gradle에서 자신이 사용하는 gradle 선택 (back 전용) - name과 version만 선택



Gradle

Gradle installations

List of Gradle installations on this system

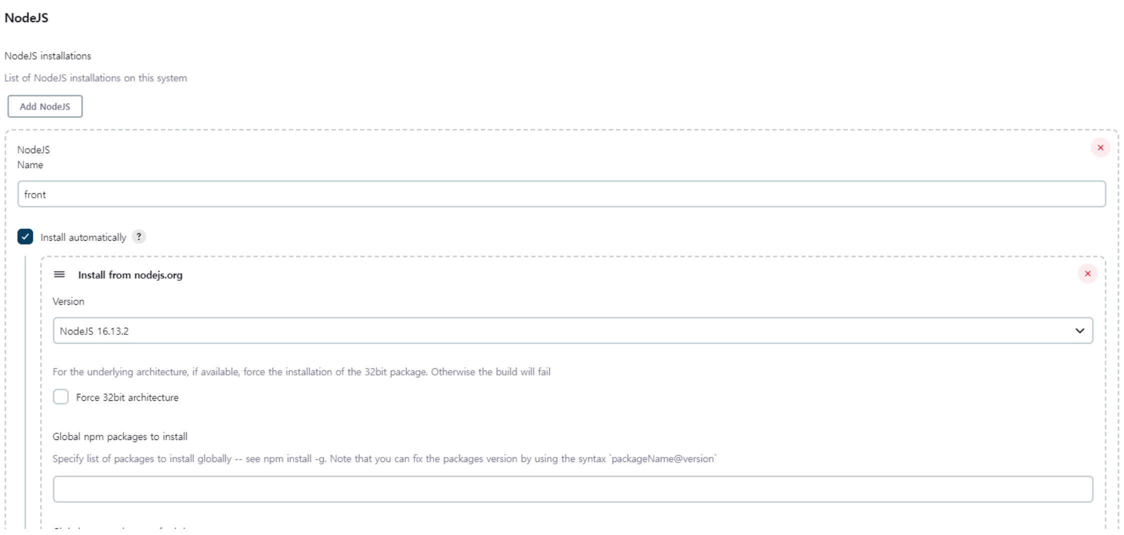
Gradle name: back

✓ Install automatically

Install from Gradle.org

Version: Gradle 7.6

- NodeJS에서 자신이 사용하는 Nodejs 버전 선택 (front 전용) - name과 version만 선택



NodeJS

NodeJS installations

List of NodeJS installations on this system

Add NodeJS

NodeJS
Name

front

☑ Install automatically ?

Install from nodejs.org

Version

NodeJS 16.13.2

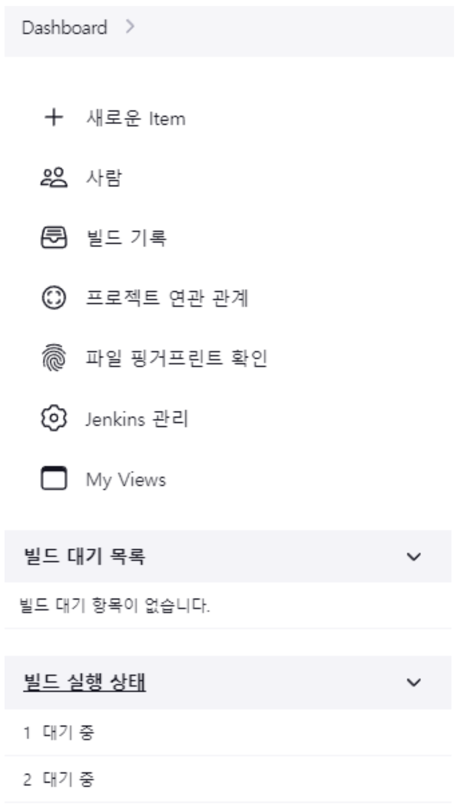
For the underlying architecture, if available, force the installation of the 32bit package. Otherwise the build will fail

☐ 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`

9. 새로운 Item 생성



Dashboard >

새로운 Item

사람

빌드 기록

프로젝트 연관 관계

파일 핑거프린트 확인

Jenkins 관리

My Views

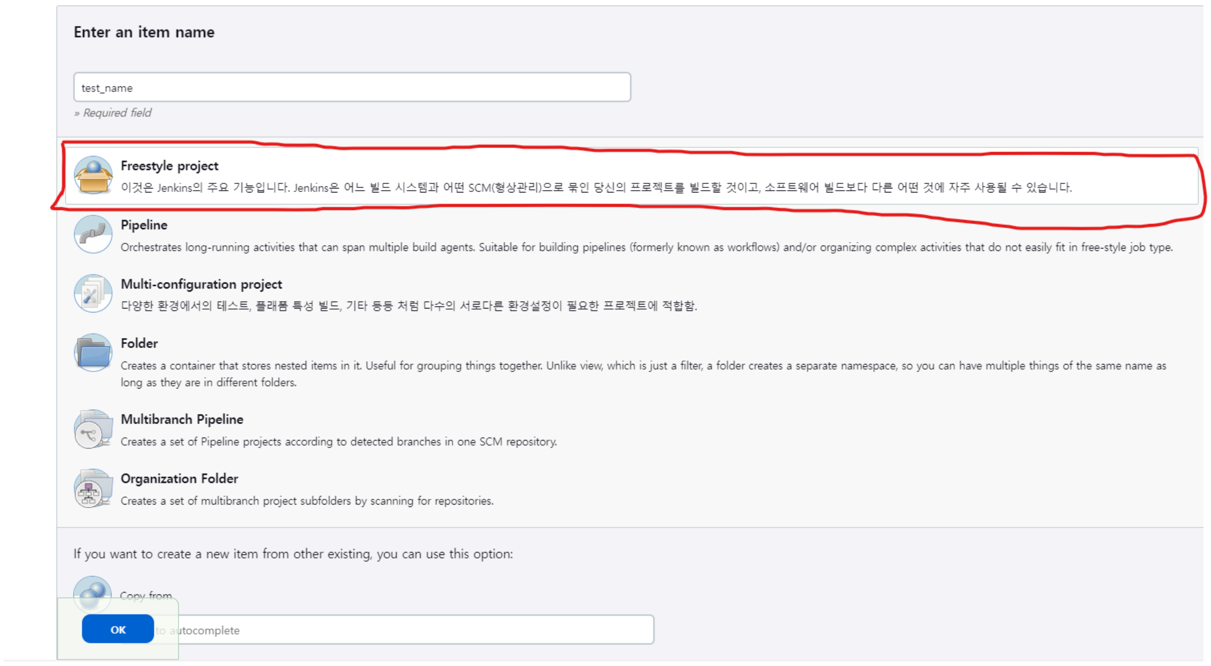
빌드 대기 목록

빌드 대기 항목이 없습니다.

빌드 실행 상태

1 대기 중

2 대기 중


Enter an item name

test_name

» *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.

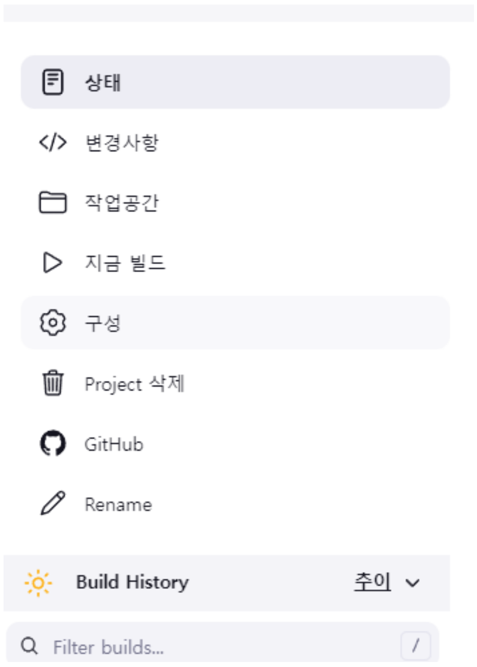
Organization Folder
Creates a set of multibranch project subfolders by scanning for repositories.

If you want to create a new item from other existing, you can use this option:

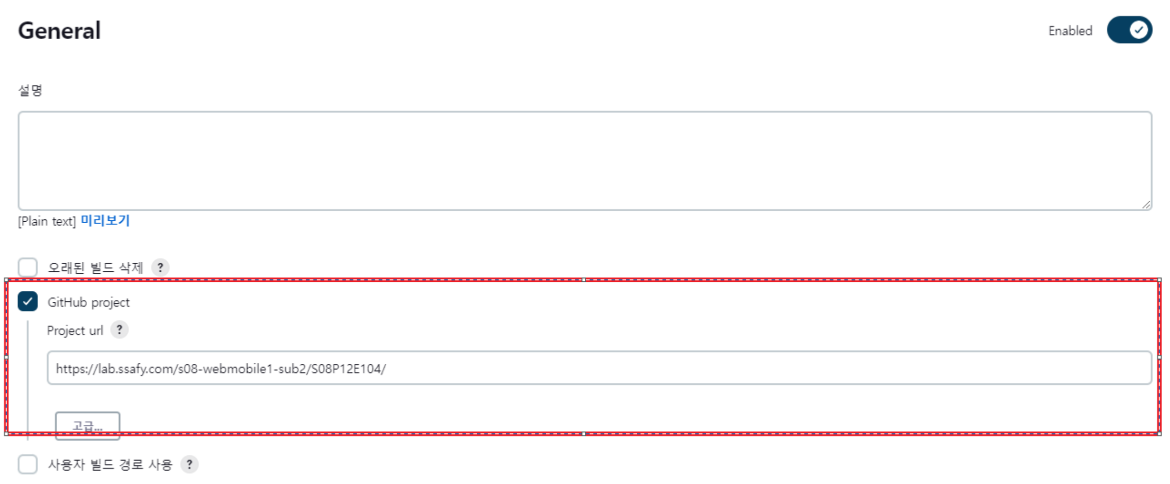
Copy from

OK

10. 프로젝트 구성 - 프로젝트 입장 후 구성 클릭



- GitHub project 선택 후 project url 입력



General

Enabled

설명

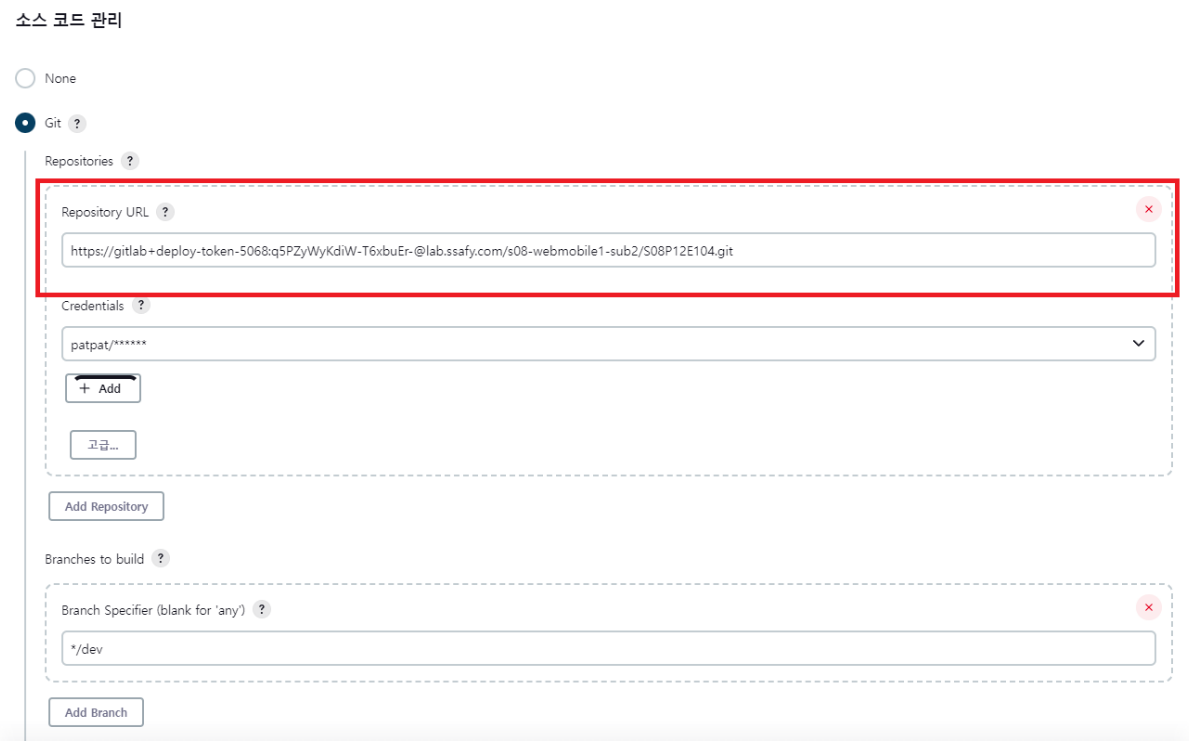
[Plain text] 미리보기

- [ ] 오래된 빌드 삭제
- [x] GitHub project

Project url: https://lab.ssafy.com/s08-webmobile1-sub2/S08P12E104/

- [ ] 사용자 빌드 경로 사용

- 소스 코드 관리 - Repository URL 입력 (deploy token 보유시 해당 token 도 함께 입력)



소스 코드 관리

None [✓ Git]

Repositories

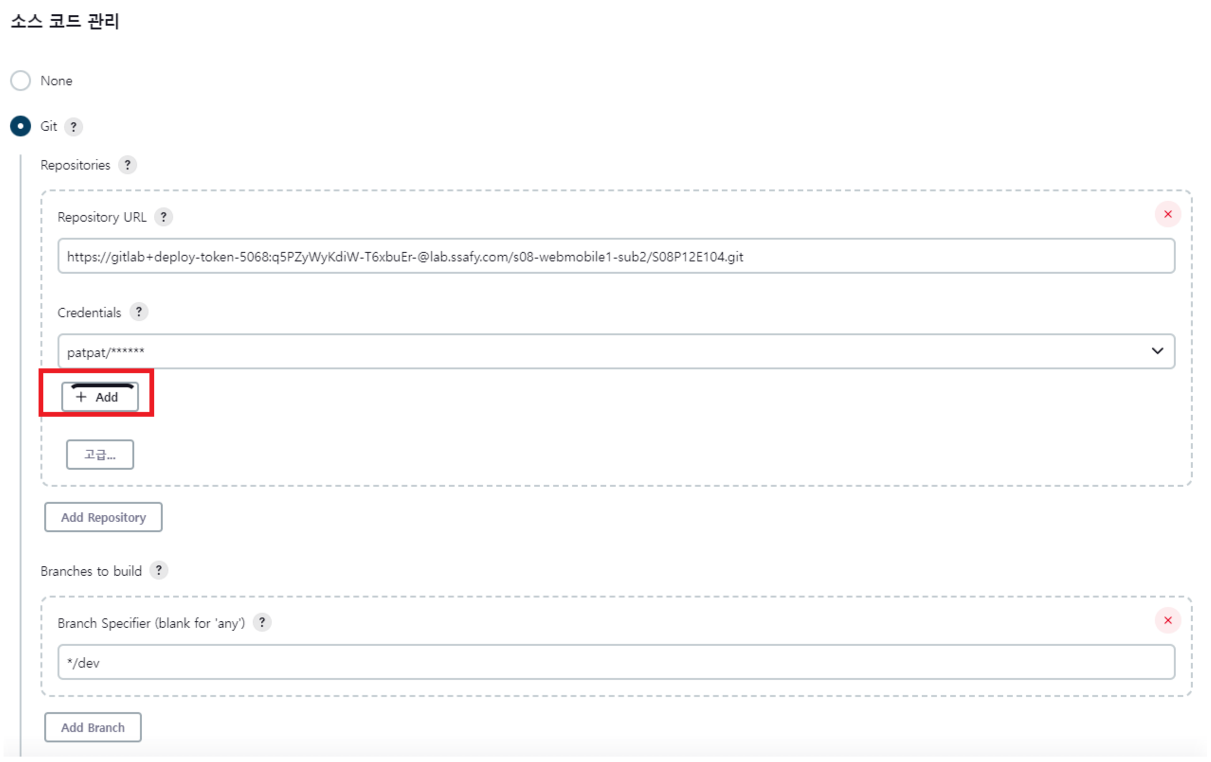
Repository URL: https://gitlab+deploy-token-5068:q5PZyWyKdiW-T6xbuEr-@lab.ssafy.com/s08-webmobile1-sub2/S08P12E104.git

Credentials: patpat/******

Branches to build

Branch Specifier (blank for 'any'): */dev

- 소스 코드 관리 - Credentials (webhooks 설정시 사용)


소스 코드 관리

None

[✓ Git]

Repositories

Repository URL

https://gitlab+deploy-token-5068:q5PZyWyKdiW-T6xbuEr-@lab.ssafy.com/s08-webmobile1-sub2/S08P12E104.git

Credentials

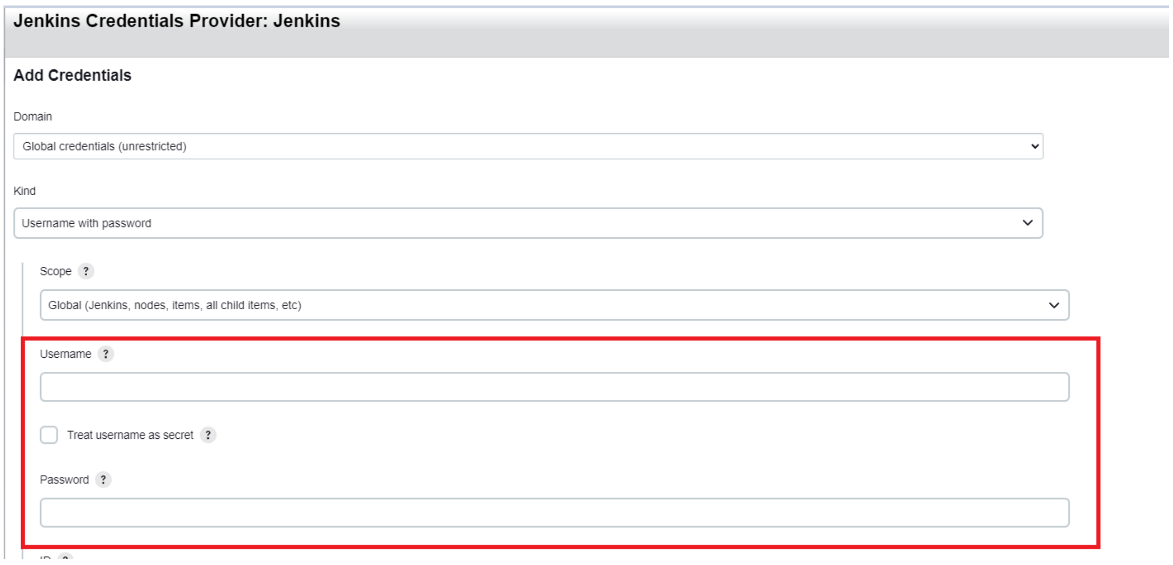
patpat/******

Branches to build

Branch Specifier (blank for 'any')

*/dev

- 자신의 gitlab의 username와 password를 입력


Jenkins Credentials Provider: Jenkins

Add Credentials

Domain

Global credentials (unrestricted)

Kind

Username with password

Scope

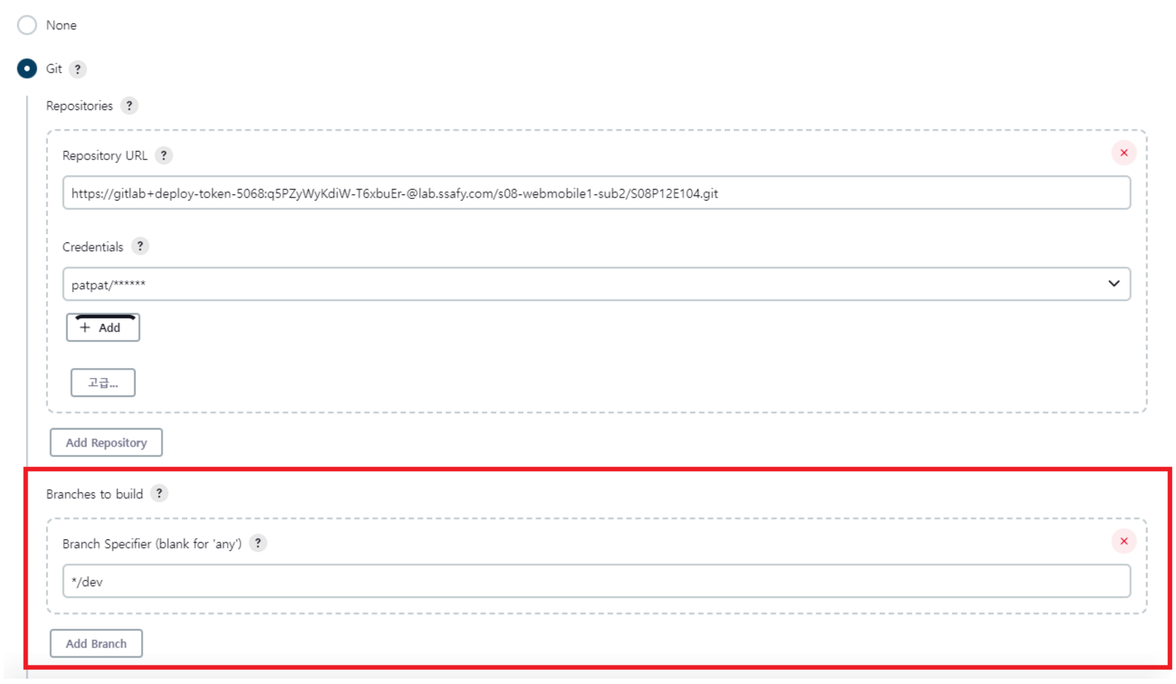
Global (Jenkins, nodes, items, all child items, etc)

Username

Treat username as secret

Password

- build 대상이 되는 branch 입력



( ) None
(●) Git

Repositories

Repository URL

https://gitlab+deploy-token-5068:q5PZyWyKdiW-T6xbuEr-@lab.ssafy.com/s08-webmobile1-sub2/S08P12E104.git

Credentials

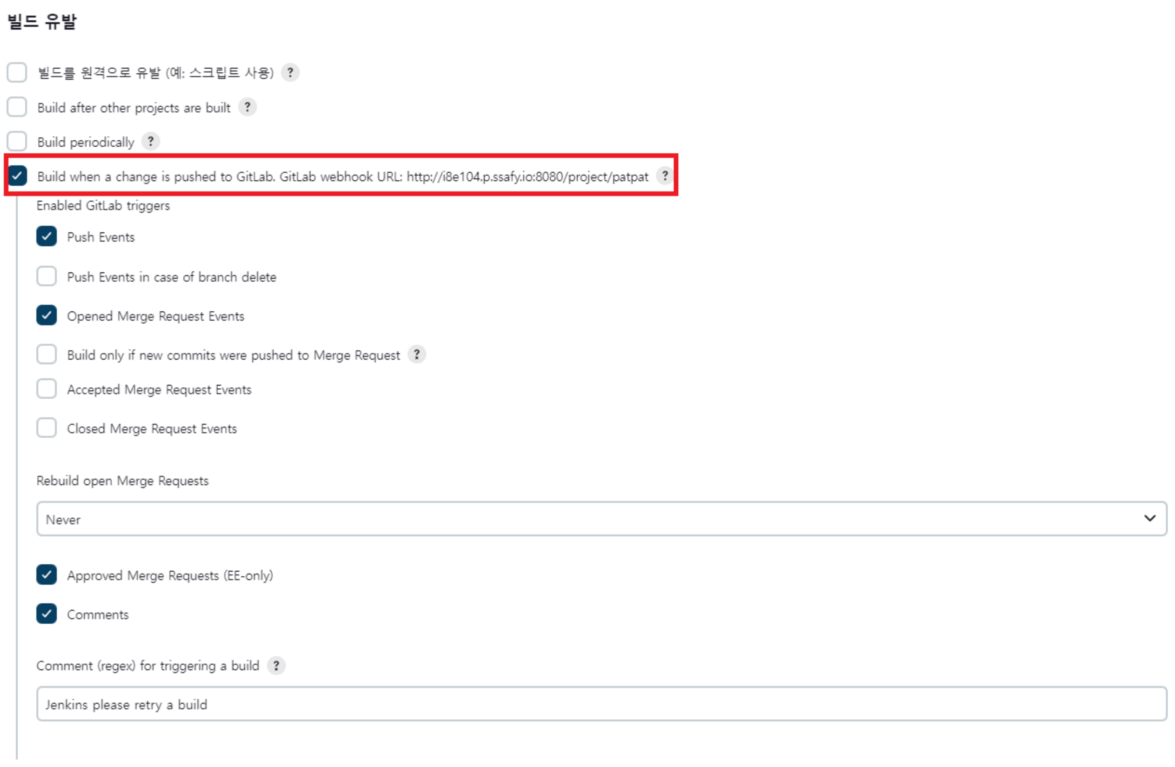
patpat/******

Branches to build

Branch Specifier (blank for 'any')

*/dev

- 빌드 유발 - Build when a change is pushed to GitLab. GitLab webhook URL…. 선택



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://i8e104.p.ssafy.io:8080/project/patpat

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

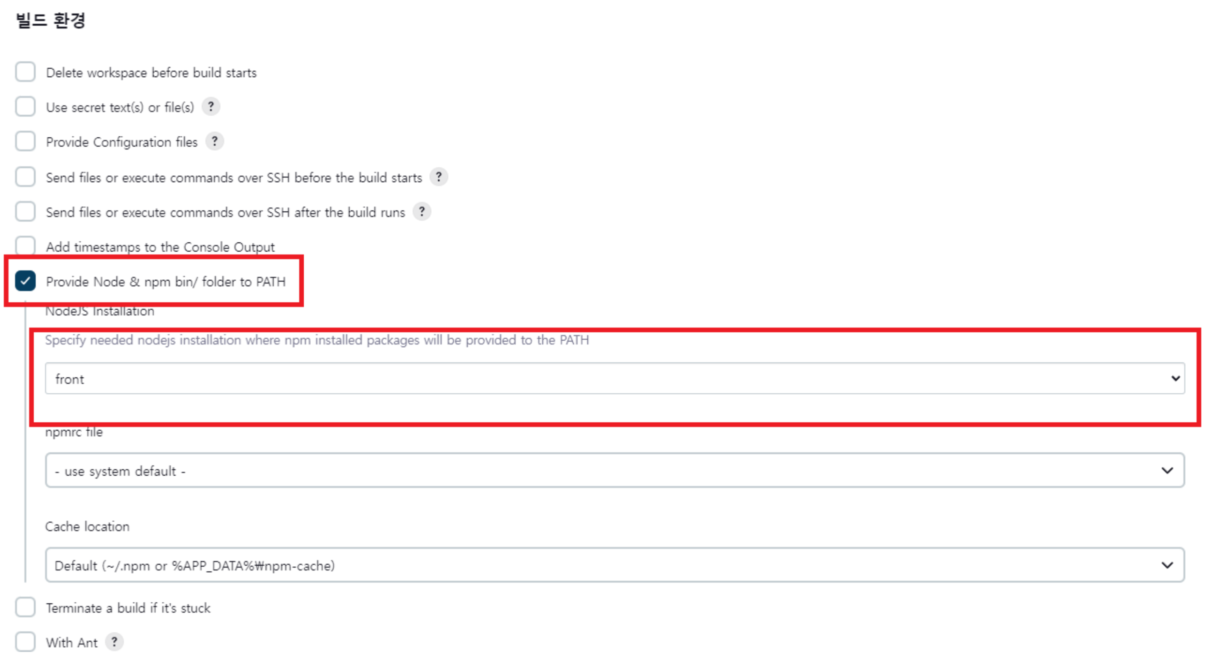
Never

- [✓] Approved Merge Requests (EE-only)
- [✓] Comments

Comment (regex) for triggering a build

Jenkins please retry a build

- 빌드 환경 - Provide Node & npm bin/ folder to PATH 선택 후 NodeJS Installation 선택 (global 선택에서 설정한 NodeJS 선택)


빌드 환경

- [ ] Delete workspace before build starts
- [ ] Use secret text(s) or file(s)
- [ ] Provide Configuration files
- [ ] Send files or execute commands over SSH before the build starts
- [ ] Send files or execute commands over SSH after the build runs
- [ ] Add timestamps to the Console Output
- [✓] Provide Node & npm bin/ folder to PATH

NodeJS Installation

Specify needed nodejs installation where npm installed packages will be provided to the PATH

front

npmrc file

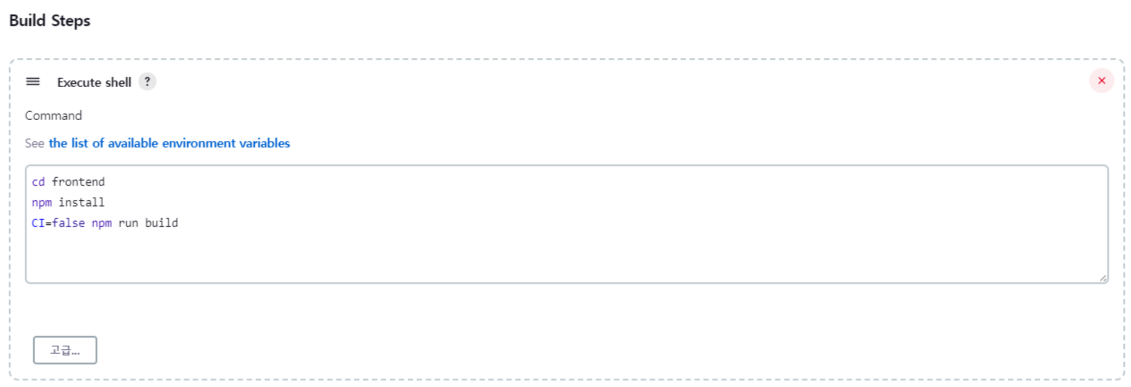
- use system default -

Cache location

Default (~/.npm or %APP_DATA%₩npm-cache)

- [ ] Terminate a build if it's stuck
- [ ] With Ant

- Build Steps - Add build step에서 Excute shell 선택
- CI=false → eslint 무시


Build Steps

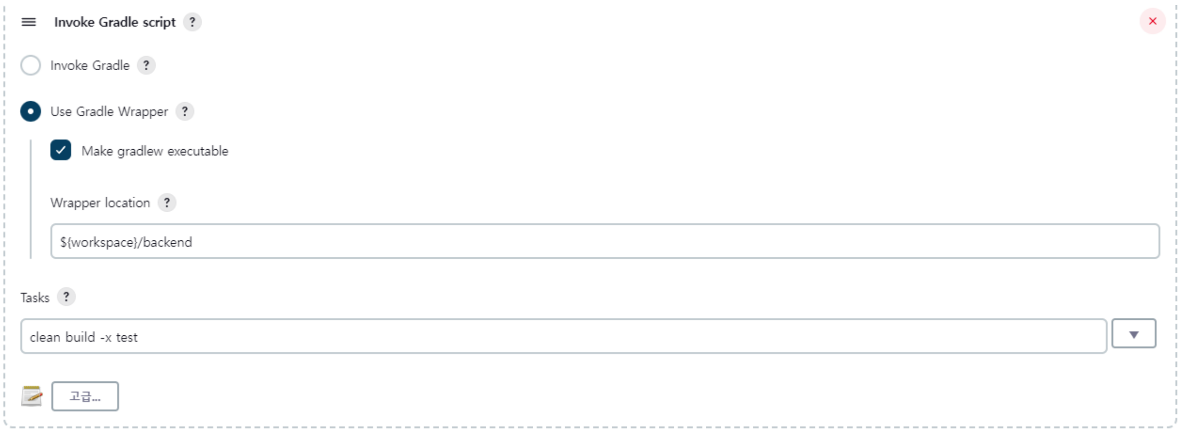
Execute shell

Command

See the list of available environment variables

```
cd frontend
npm install
CI=false npm run build
```

- Add build step에서 Invoke Gradle script 선택
- -x test → test 파일 무시


Invoke Gradle script

( ) Invoke Gradle

(●) Use Gradle Wrapper

✓ Make gradlew executable

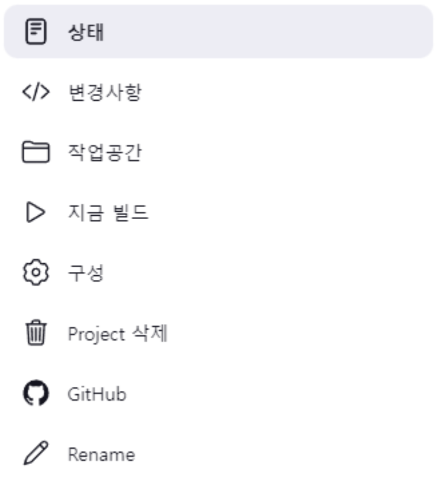
Wrapper location: `${workspace}/backend`

Tasks: `clean build -x test`

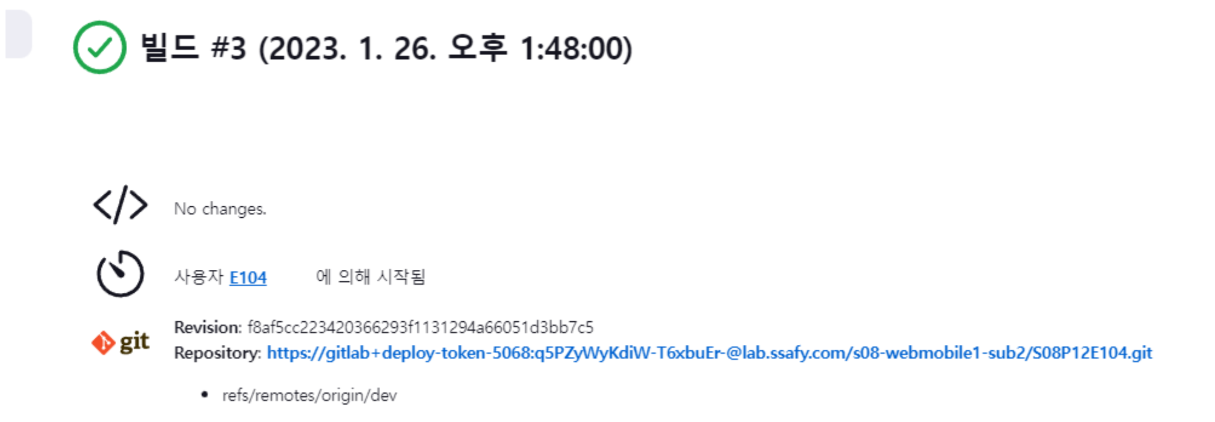
- 저장하기

11. 지금 빌드

- 왼쪽 리스트중 지금 빌드 클릭



- 빌드 성공 확인



빌드 #3 (2023. 1. 26. 오후 1:48:00)

No changes.

사용자 E104 에 의해 시작됨

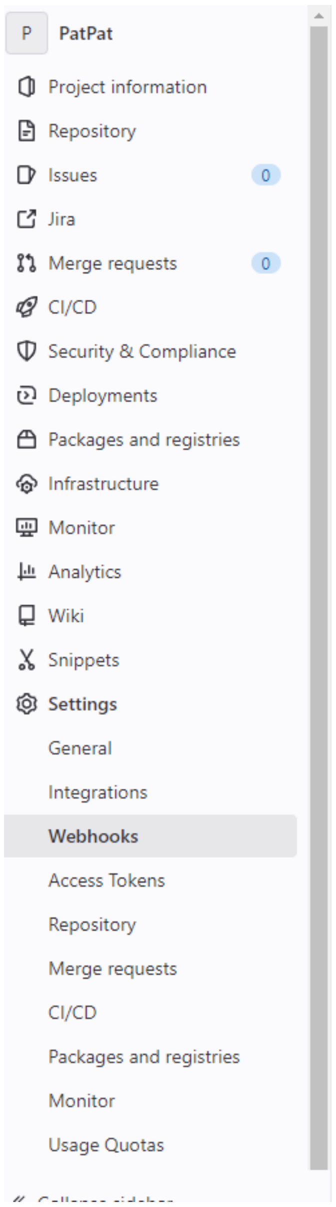
Revision: f8af5cc223420366293f1131294a66051d3bb7c5
Repository: https://gitlab+deploy-token-5068:q5PZyWyKdiW-T6xbuEr-@lab.ssafy.com/s08-webmobile1-sub2/S08P12E104.git

- refs/remotes/origin/dev

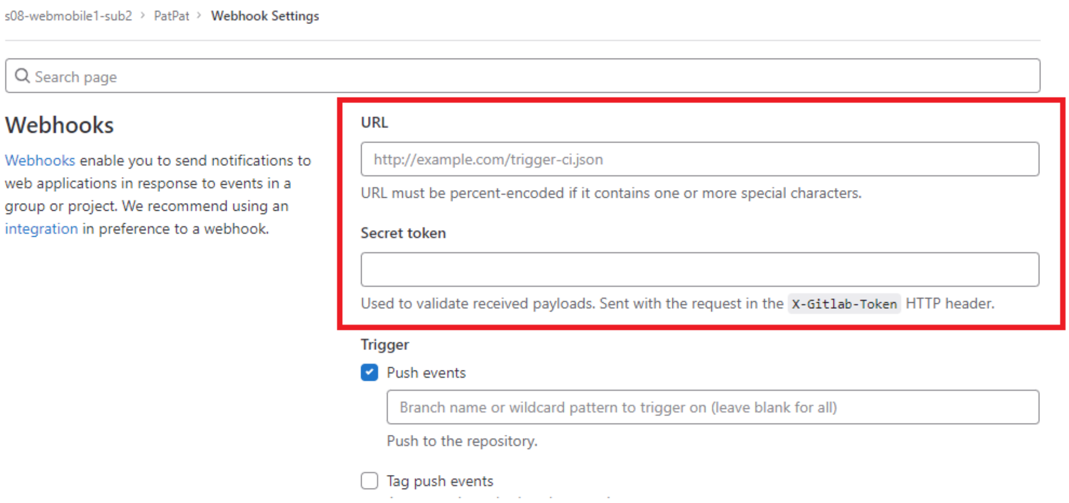
- Console Output을 클릭하면 콘솔창으로 확인 가능

12. WebHook 추가

- gitlab의 setting에 Webhooks 선택



- URL과 Secret token 입력



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

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

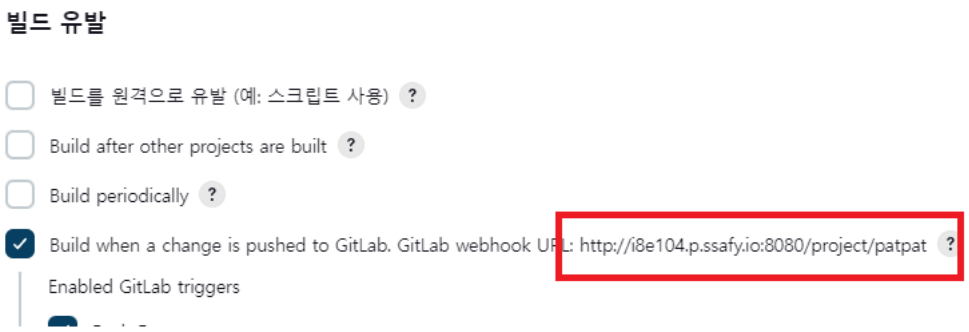
[✓] Push events

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

Push to the repository.

[ ] Tag push events

- URL은 젠킨스→ 프로젝트 구성 → 빌드 유발 → Build when a change is pushed to GitLab. GitLab webhook URL : ~~~~에 붙어 있는 URL

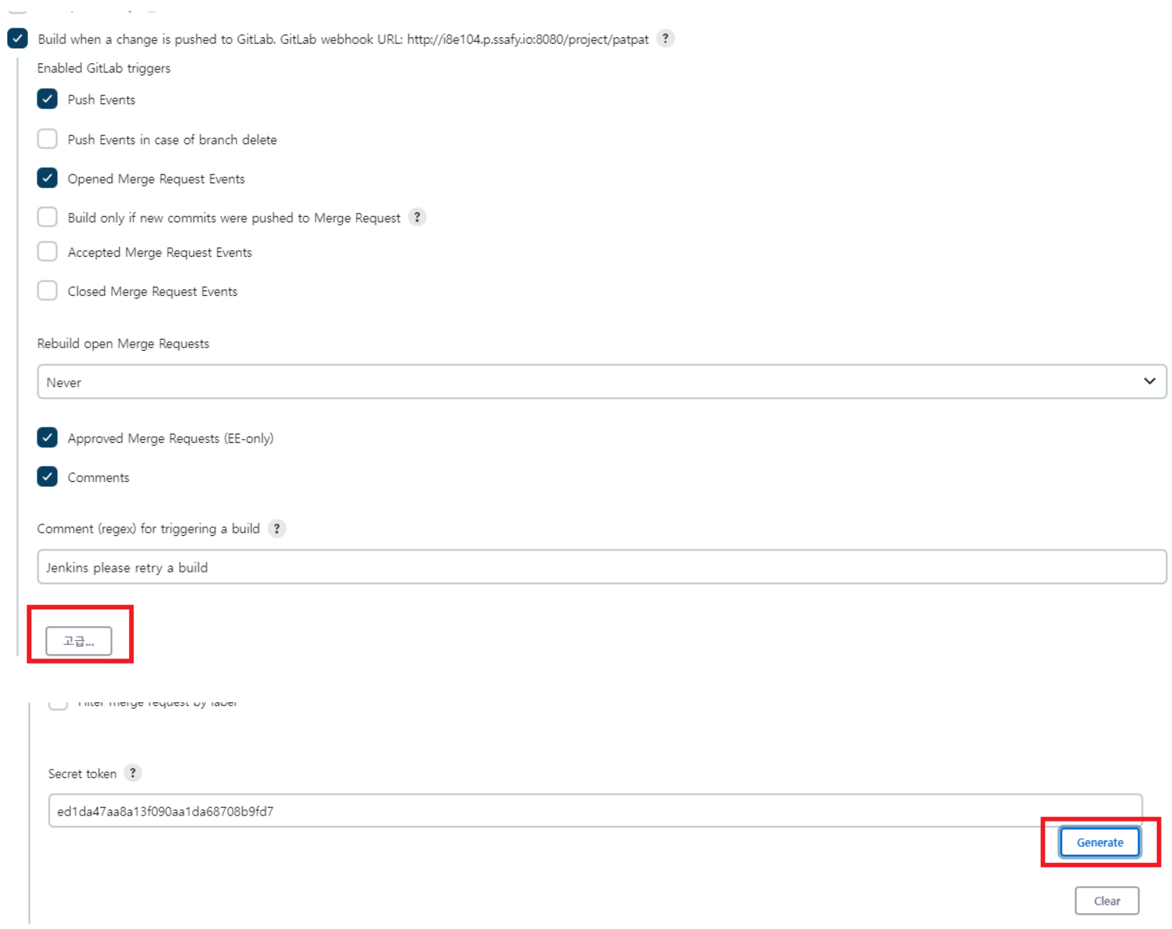


빌드 유발

- [ ] 빌드를 원격으로 유발 (예: 스크립트 사용)
- [ ] Build after other projects are built
- [ ] Build periodically
- [x] Build when a change is pushed to GitLab. GitLab webhook URL: http://i8e104.p.ssafy.io:8080/project/patpat

Enabled GitLab triggers

- Secret token은 빌드 유발 → Build when ~~~ → 고급 → 맨 밑에 secret token generate 클릭



- [x] Build when a change is pushed to GitLab. GitLab webhook URL: http://i8e104.p.ssafy.io:8080/project/patpat

Enabled GitLab triggers
- [x] Push Events
- [ ] Push Events in case of branch delete
- [x] Opened Merge Request Events
- [ ] Build only if new commits were pushed to Merge Request
- [ ] Accepted Merge Request Events
- [ ] Closed Merge Request Events

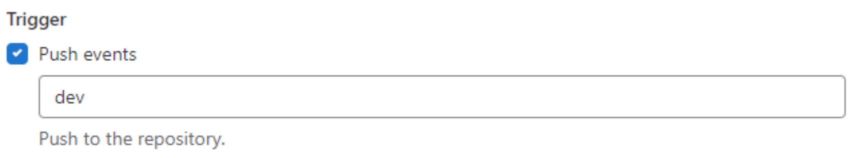
Rebuild open Merge Requests: Never

- [x] Approved Merge Requests (EE-only)
- [x] Comments

Comment (regex) for triggering a build: Jenkins please retry a build

Secret token: ed1da47aa8a13f090aa1da68708b9fd7

- URL과 Secret token 입력을 마친 뒤 Trigger 설정



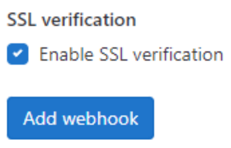
Trigger
- [x] Push events

dev

Push to the repository.

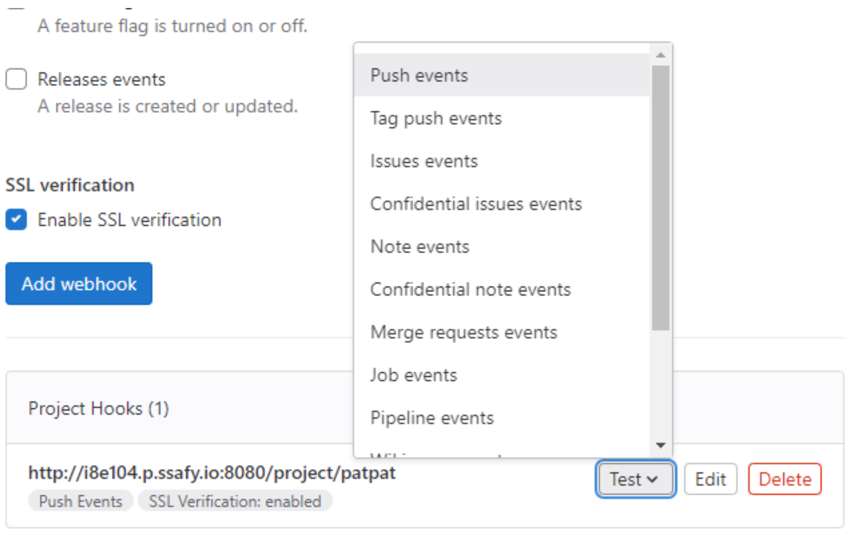
- 자동으로 빌드 시키고 싶은 브랜치 이름 적기

- SSL verification 선택 후 Add webhook


SSL verification

☑ Enable SSL verification

13. 생성된 웹훅 테스트 해보기


A feature flag is turned on or off.

☐ Releases events
A release is created or updated.

SSL verification

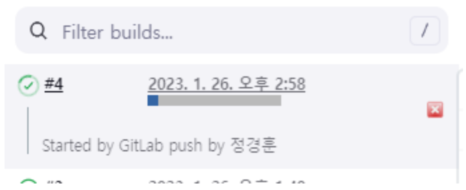
☑ Enable SSL verification

Project Hooks (1)

http://i8e104.p.ssafy.io:8080/project/patpat

Push Events | SSL Verification: enabled

- 선택 후 젠킨스 확인


#4 2023. 1. 26. 오후 2:58

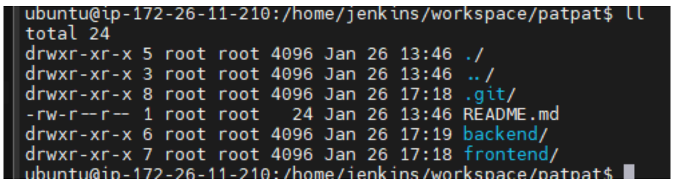
Started by GitLab push by 정경훈

- 빌드 성공 !

14. build 결과물 서버에 저장되어 있는지 확인

- docker이미지의 데이터가 바인딩된 폴더 체크

```
cd /home/jenkins/workspace/patpat
```

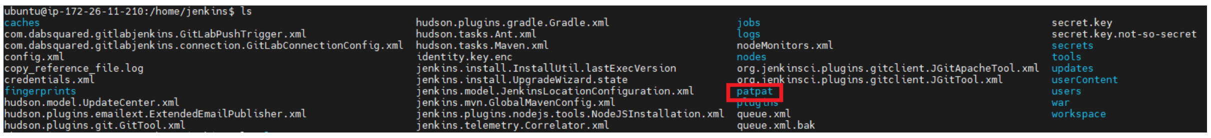


```
ubuntu@ip-172-26-11-210:/home/jenkins/workspace/patpat$ ll
total 24
drwxr-xr-x 5 root root 4096 Jan 26 13:46 ./
drwxr-xr-x 3 root root 4096 Jan 26 13:46 ../
drwxr-xr-x 8 root root 4096 Jan 26 17:18 .git/
-rw-r--r-- 1 root root   24 Jan 26 13:46 README.md
drwxr-xr-x 6 root root 4096 Jan 26 17:19 backend/
drwxr-xr-x 7 root root 4096 Jan 26 17:18 frontend/
ubuntu@ip-172-26-11-210:/home/jenkins/workspace/patpat$
```

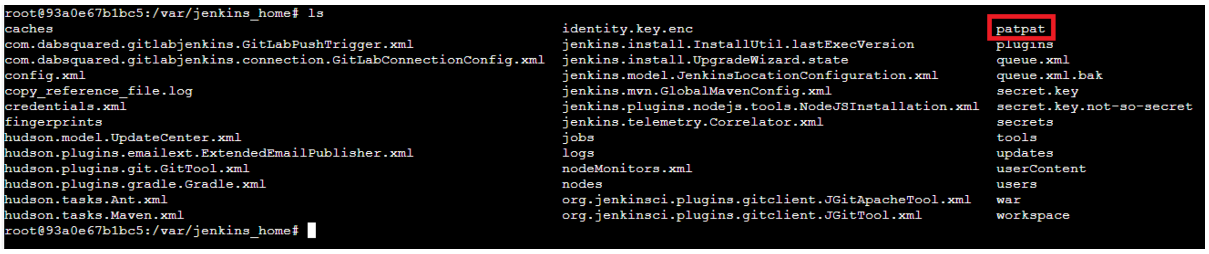
15. build된 데이터들을 jenkins 내부의 도커에 올려서 배포

- 위 단계를 진행하기 위해 build file들을 jenkins 내부의 작업 폴더로 옮길 필요가 있다.
- /home/jenkins → /var/jenkins_home으로 바인딩되어 있으므로 /home/jenkins에 폴더를 만들면 /var/jenkins_home에도 폴더가 생성된다.

```
cd /home/jenkins

sudo mkdir patpat
```



```
ubuntu@ip-172-26-11-210:/home/jenkins$ ls
caches                                                  hudson.plugins.gradle.Gradle.xml                jobs                                                secret.key
com.dabsquared.gitlabjenkins.GitLabPushTrigger.xml      hudson.tasks.Ant.xml                            logs                                                secret.key.not-so-secret
com.dabsquared.gitlabjenkins.connection.GitLabConnectionConfig.xml  hudson.tasks.Maven.xml              nodeMonitors.xml                                    secrets
config.xml                                              identity.key.enc                                nodes                                               tools
copy_reference_file.log                                 jenkins.install.InstallUtil.lastExecVersion     org.jenkinsci.plugins.gitclient.JGitApacheTool.xml  updates
credentials.xml                                         jenkins.install.UpgradeWizard.state             org.jenkinsci.plugins.gitclient.JGitTool.xml        userContent
fingerprints                                            jenkins.model.JenkinsLocationConfiguration.xml  patpat                                              users
hudson.model.UpdateCenter.xml                           jenkins.mvn.GlobalMavenConfig.xml               plugins                                             war
hudson.plugins.emailext.ExtendedEmailPublisher.xml      jenkins.plugins.nodejs.tools.NodeJSInstallation.xml  queue.xml                                      workspace
hudson.plugins.git.GitTool.xml                          jenkins.telemetry.Correlator.xml                queue.xml.bak
```



```
root@93a0e67b1bc5:/var/jenkins_home# ls
caches                                                              identity.key.enc                                     patpat
com.dabsquared.gitlabjenkins.GitLabPushTrigger.xml                  jenkins.install.InstallUtil.lastExecVersion          plugins
com.dabsquared.gitlabjenkins.connection.GitLabConnectionConfig.xml  jenkins.install.UpgradeWizard.state                  queue.xml
config.xml                                                          jenkins.model.JenkinsLocationConfiguration.xml       queue.xml.bak
copy_reference_file.log                                             jenkins.mvn.GlobalMavenConfig.xml                    secret.key
credentials.xml                                                     jenkins.plugins.nodejs.tools.NodeJSInstallation.xml  secret.key.not-so-secret
fingerprints                                                        jenkins.telemetry.Correlator.xml                     secrets
hudson.model.UpdateCenter.xml                                       jobs                                                 tools
hudson.plugins.emailext.ExtendedEmailPublisher.xml                  logs                                                 updates
hudson.plugins.git.GitTool.xml                                      nodeMonitors.xml                                     userContent
hudson.plugins.gradle.Gradle.xml                                    nodes                                                users
hudson.tasks.Ant.xml                                                org.jenkinsci.plugins.gitclient.JGitApacheTool.xml   war
hudson.tasks.Maven.xml                                              org.jenkinsci.plugins.gitclient.JGitTool.xml         workspace
root@93a0e67b1bc5:/var/jenkins_home#
```

- 여기서 우리는 자동 배포를 위해 해당 build 파일을 도커 이미지로 만들고 이를 컨테이너로 실행시킴으로써 배포할 예정이다.
- 해당 작업을 위해 여러 파일을 만들어 준다.

```
cd patpat

# 프론트 파일을 담을 폴더
sudo mkdir frontend

# 백 파일을 담을 폴더
sudo mkdir backend

# nginx와 다른 실행 스크립트를 담을 폴더
sudo mkdir common

# 도커 이미지 파일
sudo touch DockerFile

# 이미지를 생성하고 컨테이너를 생성하는 스크립트
sudo touch docker_exec.sh

cd common

# nginx 초기 세팅
sudo touch myapp.conf

# 프론트와 백을 실행시킬 스크립트
sudo touch deploy_start.sh
```

- 해당 작업을 위한 스크립트 파일 docker_exec.sh 파일은 아래와 같다.

```
#docker_exec.sh
# deploy라는 이름을 가진 컨테이너를 탐색한다.
NODE_CONTAINER_ID=`docker ps -aq --filter 'name=deploy'`

# 해당 폴더의 DockerFile을 이용하여 deploy 도커 이미지 생성 해당 경로 . 찍는거 주의
docker build -t deploy .

# 만약 deploy 컨테이너가 있다면
if [ -n "$NODE_CONTAINER_ID" ];
  then
    # deploy 컨테이너를 중지
    docker stop $NODE_CONTAINER_ID
    # deploy 컨테이너를 삭제
    docker rm $NODE_CONTAINER_ID
    # deploy 컨테이너 새로 생성
    docker run -itd --restart=unless-stopped --name deploy -p 80:3000 -v /usr/share/zoneinfo/Asia/Seoul:/etc/timezone:ro -v /home/depl
else
    # 만약 deploy 컨테이너가 없었다면
    # deploy 컨테이너 생성
    docker run -itd --restart=unless-stopped --name deploy -p 80:3000 -v /usr/share/zoneinfo/Asia/Seoul:/etc/timezone:ro -v /home/depl
fi

# 사용하지 않는 이미지 삭제
docker rmi $(docker images -f "dangling=true" -q)
```

- 해당 스크립트에서 사용되는 DockerFile은 아래와 같다.

```
# 자바 8 기준으로 실행
FROM openjdk:8-jre-slim

# 인자 설정
ARG BACKEND_FILE=backend/*.jar
ARG FRONTEND_FILE=frontend/
ARG NGINX_FILE=common/myapp.conf
ARG EXEC_FILE=common/deploy_start.sh

# nginx 설치 및 설정
RUN apt-get update
RUN apt-get -y install nginx
RUN rm /etc/nginx/sites-available/default
RUN rm /etc/nginx/sites-enabled/default
COPY ${NGINX_FILE} /etc/nginx/sites-available
RUN ln -s /etc/nginx/sites-available/myapp.conf /etc/nginx/sites-enabled/myapp.conf

# FE, BE build 파일 복사
RUN mkdir /frontend
COPY ${FRONTEND_FILE} /frontend
COPY ${BACKEND_FILE} /backend/app.jar

# Exec 파일 복사
COPY ${EXEC_FILE} ./

# 복사된 deploy파일 실행
ENTRYPOINT ["/bin/sh", "deploy_start.sh"]
```

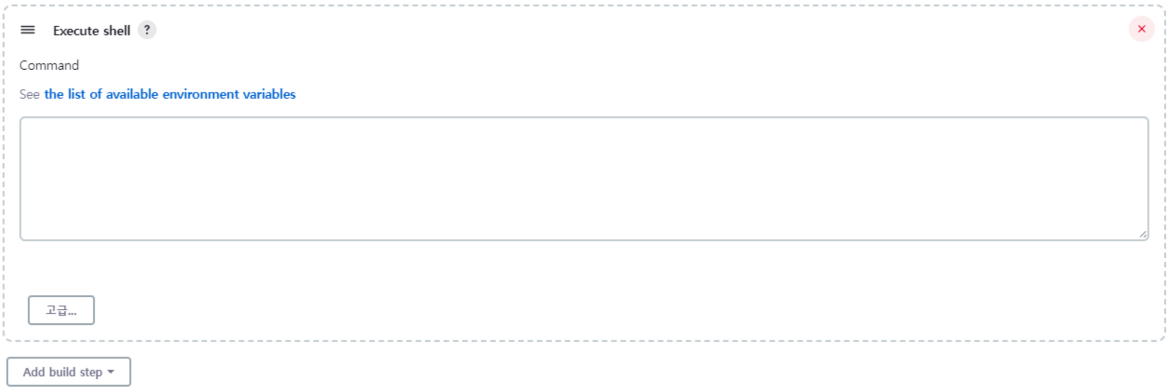
- nginx 기본 설정 파일인 myapp.conf는 다음과 같다.

```
server {
  listen 3000;
  location / {
    root   /frontend;
    index  index.html index.htm;
    try_files $uri $uri/ /index.html;
  }
}
```

- 프론트와 백 서버를 실행 시킬 스크립트 파일인 deploy_start.sh 아래와 같다.

```
#/bin/bash
/etc/init.d/nginx start
java -jar /backend/app.jar
```

- 이러한 실행 과정을 자동화하기 위해 jenkins 프로젝트 내부의 build steps를 활용한다.
- Add build step에서 Execute shell을 선택하여 새로운 창을 띄운다.



- 위 shell창은 jenkins 내부에서 돌아가는 코드이다. 고로 파일을 옮겨줄때도 jenkins 내부에 바운딩되어 있는 폴더를 기준으로 작성한다.

```
# backend build 파일 옮기기
cp -rp /var/jenkins_home/workspace/patpat/backend/build/libs/*SNAPSHOT.jar /var/jenkins_home/patpat/backend/

# frontend build 파일 옮기기
cp -rp /var/jenkins_home/workspace/patpat/frontend/build/* /var/jenkins_home/patpat/frontend/
```

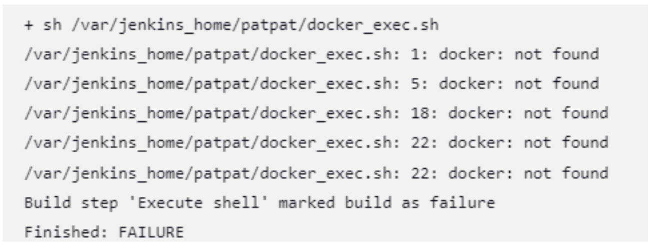
- 이후 해당 파일로 이동한다.

```
cd /var/jenkins_home/patpat
```

- 스크립트 파일을 실행한다.

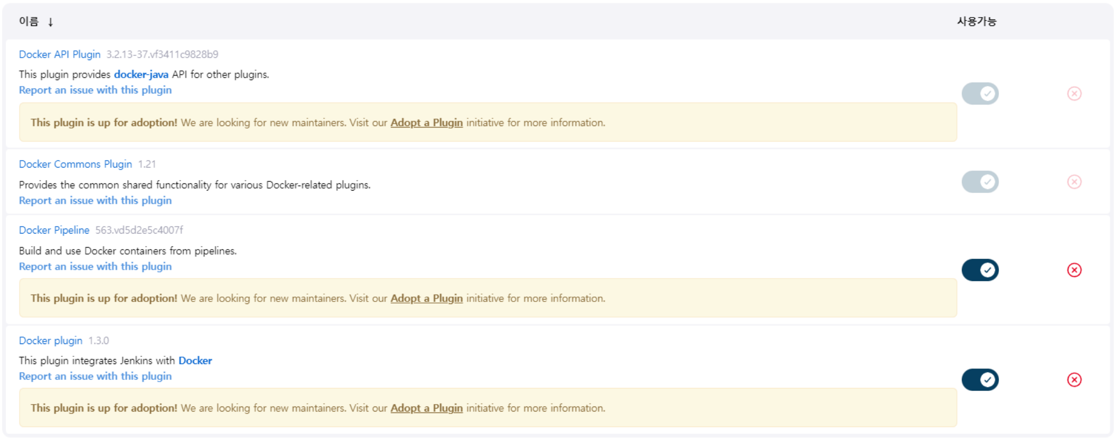
```
sh /var/jenkins_home/patpat/docker_exec.sh
```

❌ 이렇게 실행하다 보면 에러가 하나 발생한다.



```
+ sh /var/jenkins_home/patpat/docker_exec.sh
/var/jenkins_home/patpat/docker_exec.sh: 1: docker: not found
/var/jenkins_home/patpat/docker_exec.sh: 5: docker: not found
/var/jenkins_home/patpat/docker_exec.sh: 18: docker: not found
/var/jenkins_home/patpat/docker_exec.sh: 22: docker: not found
/var/jenkins_home/patpat/docker_exec.sh: 22: docker: not found
Build step 'Execute shell' marked build as failure
Finished: FAILURE
```

- jenkins 내부에 도커가 없어서 도커를 실행할 수 없다고 뜬다.
- 해결방법
  1. jenkins 에서 docker관련 플러그인 설치



이름 ↓ | 사용가능

Docker API Plugin 3.2.13-37.vf3411c9828b9
This plugin provides docker-java API for other plugins.
Report an issue with this plugin
This plugin is up for adoption! We are looking for new maintainers. Visit our Adopt a Plugin initiative for more information.

Docker Commons Plugin 1.21
Provides the common shared functionality for various Docker-related plugins.
Report an issue with this plugin

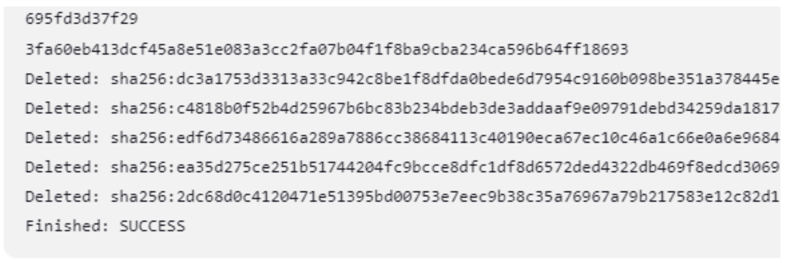
Docker Pipeline 563.vd5d2e5c4007f
Build and use Docker containers from pipelines.
Report an issue with this plugin
This plugin is up for adoption! We are looking for new maintainers. Visit our Adopt a Plugin initiative for more information.

Docker plugin 1.3.0
This plugin integrates Jenkins with Docker
Report an issue with this plugin
This plugin is up for adoption! We are looking for new maintainers. Visit our Adopt a Plugin initiative for more information.

2. jenkins 컨테이너 내부에 docker 설치

```
apt-get remove docker docker-engine docker.io containerd runc
apt-get update
apt-get install \
    ca-certificates \
    curl \
    gnupg \
    lsb-release
mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/debian/gpg | gpg --dearmor -o /etc/apt/keyrings/docker.gpg
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/debian \
  $(lsb_release -cs) stable" | tee /etc/apt/sources.list.d/docker.list > /dev/null
## - Install Docker Engine
apt-get update
apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin
```

- 위 두 방법을 통해서 해결했다.



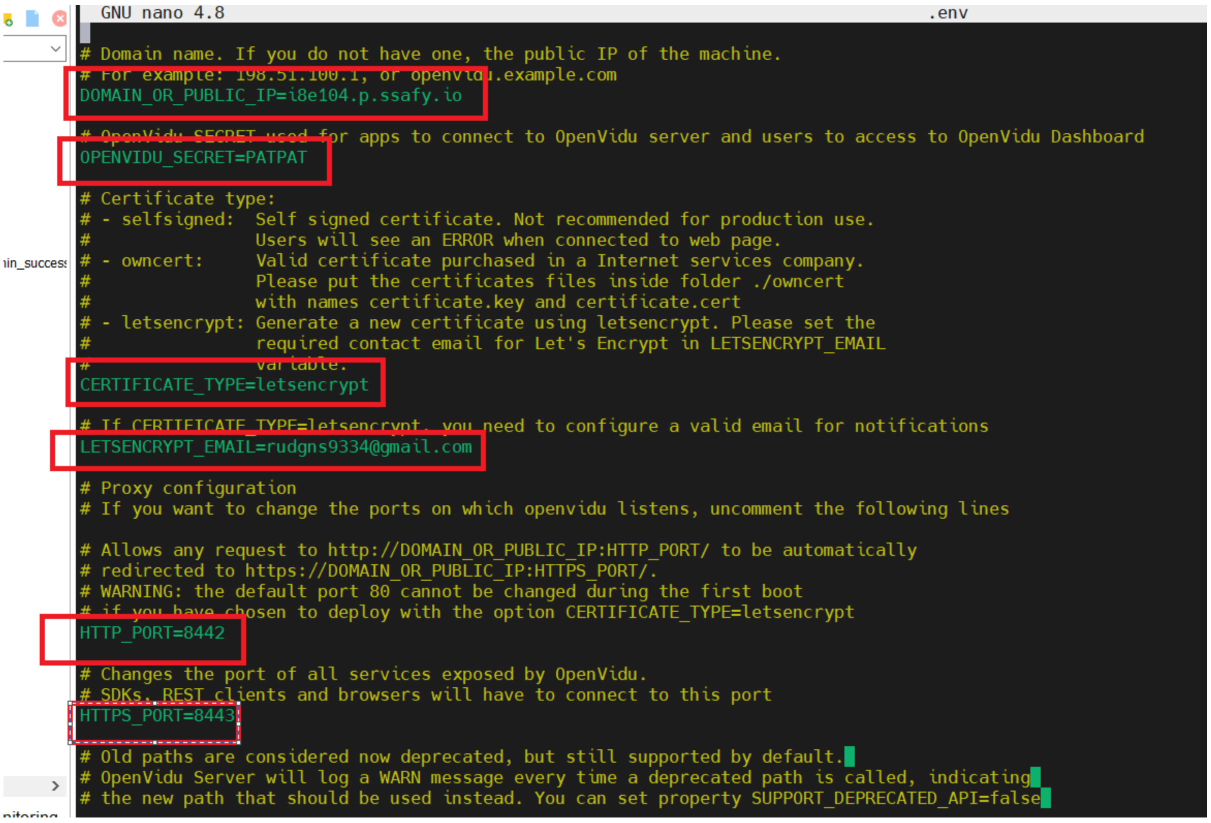
```
695fd3d37f29
3fa60eb413dcf45a8e51e083a3cc2fa07b04f1f8ba9cba234ca596b64ff18693
Deleted: sha256:dc3a1753d3313a33c942c8be1f8dfda0bede6d7954c9160b098be351a378445e
Deleted: sha256:c4818b0f52b4d25967b6bc83b234bdeb3de3addaaf9e09791debd34259da1817
Deleted: sha256:edf6d73486616a289a7886cc38684113c40190eca67ec10c46a1c66e0a6e9684
Deleted: sha256:ea35d275ce251b51744204fc9bcce8dfc1df8d6572ded4322db469f8edcd3069
Deleted: sha256:2dc68d0c4120471e51395bd00753e7eec9b38c35a76967a79b217583e12c82d1
Finished: SUCCESS
```

😅 성공..

Openvidu 설치(8443 port)

-

```
# 배포를 위한 권한 얻기
sudo su
# 위치이동
cd /opt
# openvidu 다운
curl <https://s3-eu-west-1.amazonaws.com/aws.openvidu.io/install_openvidu_latest.sh> | bash
# 위치이동
cd openvidu
# 환경 설정
nano .env
```

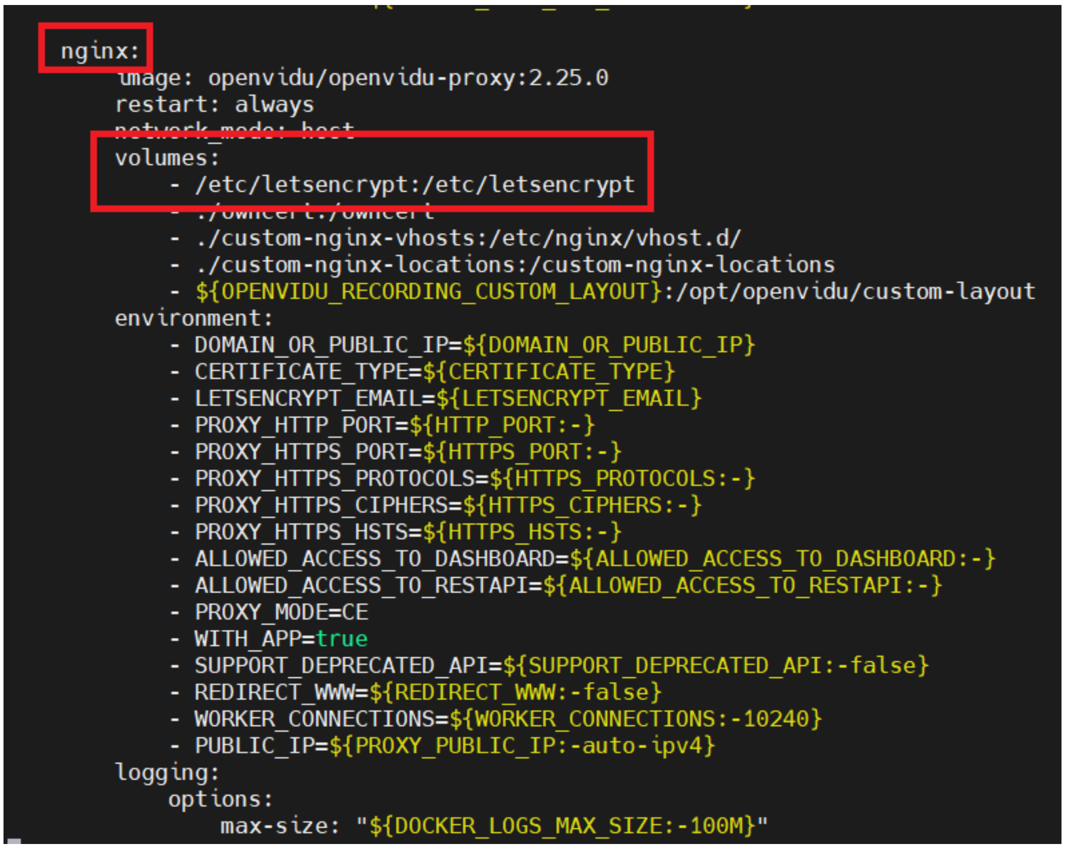


```
  GNU nano 4.8                                                        .env
# Domain name. If you do not have one, the public IP of the machine.
# For example: 198.51.100.1, or openvidu.example.com
DOMAIN_OR_PUBLIC_IP=i8e104.p.ssafy.io

# OpenVidu SECRET used for apps to connect to OpenVidu server and users to access to OpenVidu Dashboard
OPENVIDU_SECRET=PATPAT

# Certificate type:
# - selfsigned:  Self signed certificate. Not recommended for production use.
#                Users will see an ERROR when connected to web page.
# - owncert:     Valid certificate purchased in a Internet services company.
#                Please put the certificates files inside folder ./owncert
#                with names certificate.key and certificate.cert
# - letsencrypt: Generate a new certificate using letsencrypt. Please set the
#                required contact email for Let's Encrypt in LETSENCRYPT_EMAIL
#                variable.
CERTIFICATE_TYPE=letsencrypt

# If CERTIFICATE_TYPE=letsencrypt, you need to configure a valid email for notifications
LETSENCRYPT_EMAIL=rudgns9334@gmail.com

# Proxy configuration
# If you want to change the ports on which openvidu listens, uncomment the following lines

# Allows any request to http://DOMAIN_OR_PUBLIC_IP:HTTP_PORT/ to be automatically
# redirected to https://DOMAIN_OR_PUBLIC_IP:HTTPS_PORT/.
# WARNING: the default port 80 cannot be changed during the first boot
# if you have chosen to deploy with the option CERTIFICATE_TYPE=letsencrypt
HTTP_PORT=8442

# Changes the port of all services exposed by OpenVidu.
# SDKs, REST clients and browsers will have to connect to this port
HTTPS_PORT=8443

# Old paths are considered now deprecated, but still supported by default.
# OpenVidu Server will log a WARN message every time a deprecated path is called, indicating
# the new path that should be used instead. You can set property SUPPORT_DEPRECATED_API=false
```

- DOMAIN_OR_PUBLIC_IP=자신 도메인의 주소
- OPENVIDU_SECRET=원하는 비밀번호
- CERTIFICATE_TYPE=nginx로 했을땐 letsencrypt 고정
- LETSENCRYPT_EMAIL=정보확인용 개인 이메일
- HTTP_PORT=HTTP용 포트
- HTTPS_PORT=HTTPS용 포트
- 위 두 항목은 원하는 값을 주면 되는데 사전에 certbot을 통해 letsencrypt 인증을 받았을 경우 설정을 하나 더 해준다.

```
nano docker-compose.yml
```

- 들어가서 쭉 내리다보면 nginx부분이 있다.



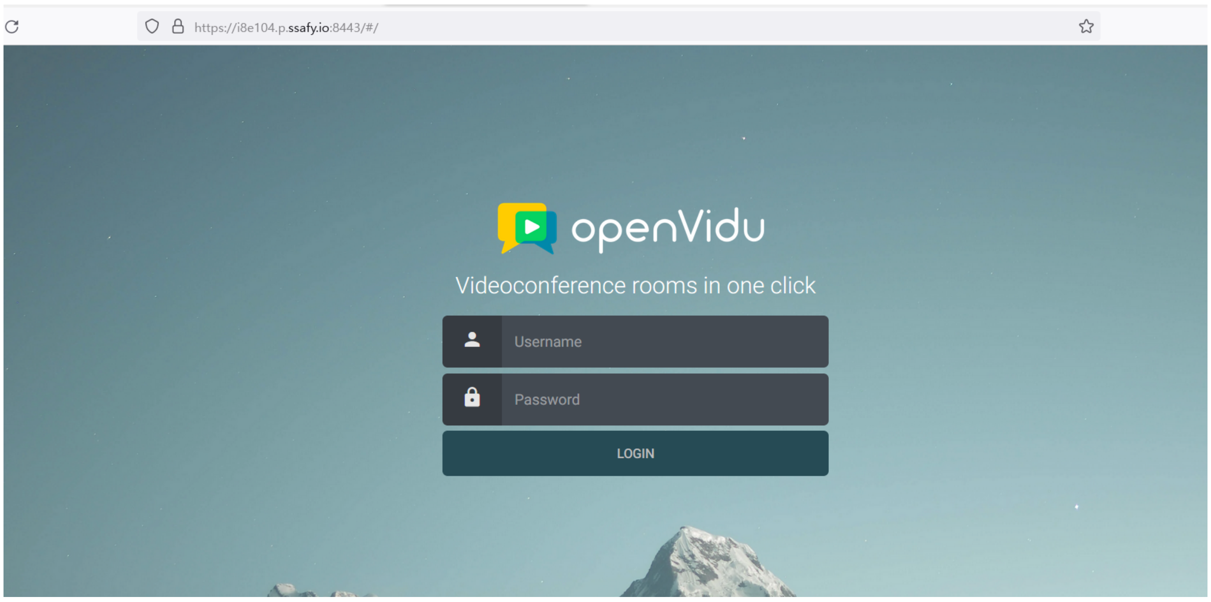
```
nginx:
    image: openvidu/openvidu-proxy:2.25.0
    restart: always
    network_mode: host
    volumes:
        - /etc/letsencrypt:/etc/letsencrypt
        - ./owncert:/owncert
        - ./custom-nginx-vhosts:/etc/nginx/vhost.d/
        - ./custom-nginx-locations:/custom-nginx-locations
        - ${OPENVIDU_RECORDING_CUSTOM_LAYOUT}:/opt/openvidu/custom-layout
    environment:
        - DOMAIN_OR_PUBLIC_IP=${DOMAIN_OR_PUBLIC_IP}
        - CERTIFICATE_TYPE=${CERTIFICATE_TYPE}
        - LETSENCRYPT_EMAIL=${LETSENCRYPT_EMAIL}
        - PROXY_HTTP_PORT=${HTTP_PORT:-}
        - PROXY_HTTPS_PORT=${HTTPS_PORT:-}
        - PROXY_HTTPS_PROTOCOLS=${HTTPS_PROTOCOLS:-}
        - PROXY_HTTPS_CIPHERS=${HTTPS_CIPHERS:-}
        - PROXY_HTTPS_HSTS=${HTTPS_HSTS:-}
        - ALLOWED_ACCESS_TO_DASHBOARD=${ALLOWED_ACCESS_TO_DASHBOARD:-}
        - ALLOWED_ACCESS_TO_RESTAPI=${ALLOWED_ACCESS_TO_RESTAPI:-}
        - PROXY_MODE=CE
        - WITH_APP=true
        - SUPPORT_DEPRECATED_API=${SUPPORT_DEPRECATED_API:-false}
        - REDIRECT_WWW=${REDIRECT_WWW:-false}
        - WORKER_CONNECTIONS=${WORKER_CONNECTIONS:-10240}
        - PUBLIC_IP=${PROXY_PUBLIC_IP:-auto-ipv4}
    logging:
        options:
            max-size: "${DOCKER_LOGS_MAX_SIZE:-100M}"
```

- 기존에는 ./certificates:/etc/letsencrypt 라고 되어 있을 것이다.
- 이를 나의 native 인증키가 저장된 곳과 바운딩 시켜주자.

- 이후 openvidu를 실행한다.

```
./openvidu start
```

- 그러면 해당 url 접속시 아래와 같은 화면이 나온다.


https://i8e104.p.ssafy.io:8443/#/
openVidu
Videoconference rooms in one click
Username
Password
LOGIN

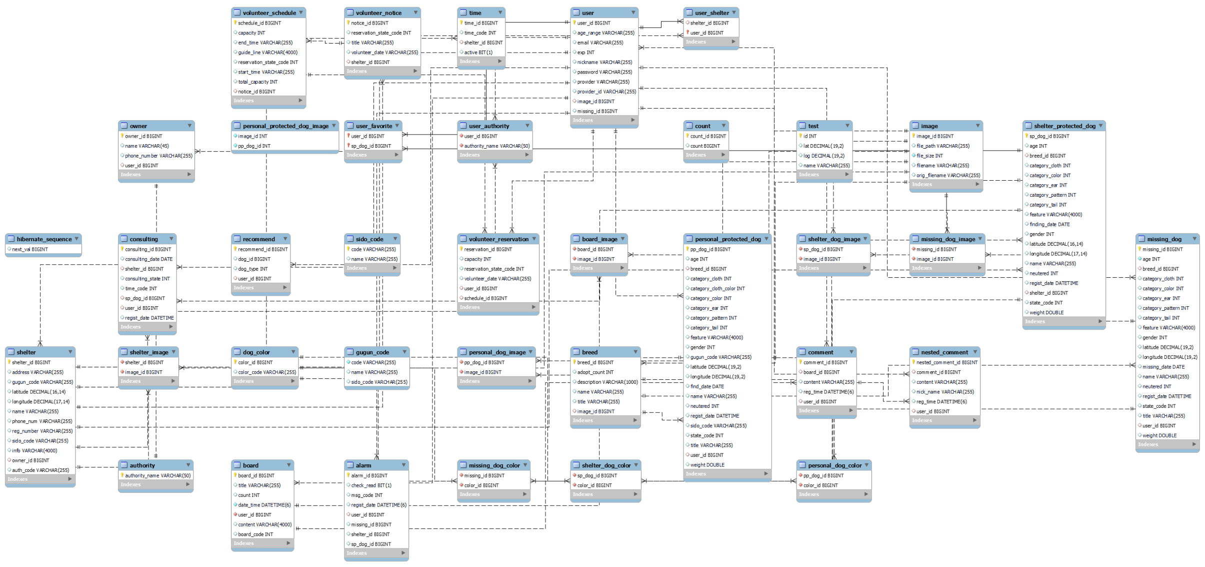
Redis(6379 port)

```
docker pull redis:latest

docker run --name patpat-redis -d redis redis-server --save 60 1 --loglevel warning --requirepass patpat104
```

포테이너로 해당 컨테이너 가서 콘솔로 드간다.

```
# 접속하기
redis-cli -p 6379

# 슈퍼 계정 만들기
ACL SETUSER [유저네임] on >[비밀번호] allkeys allcommands

auth [유저네임] [비밀번호]
또는 redis-cli 밖에서 실행할때
redis-cli --user [유저네임] --pass [비밀번호] -p 6379
```

최종 ERD



db port : 3306

user : root

password : patpat104