



# 포팅매뉴얼 v.1

## 버전정보

### <BackEnd>

- Java jdk 11.0.15.1
- SpringBoot 2.7.4
- MySQL 8.0.30
- Hibernate 5.6.10.Final
- swagger 3.0.0
- redis 7.0
- Amazon S3 2.2.6
- QueryDSL 5.0.0

### <FrontEnd>

- node.js LTS 16.17.0
- React 18.2.0
- Recoil 0.7.6

### <CI/CD>

- AWS EC2 20.04 LTS
- Docker 20.10.21
- Nginx 1.18.0
- Jenkins 2.361.2

## 카카오 API

Redirect URI 열어두기



꽃바다



ID 814126

OWNER

Biz

Web

카카오 로그인 **ON**

[동의 화면 미리보기](#)

### 활성화 설정

상태

**ON**

카카오 로그인 API를 활용하면 사용자들이 번거로운 회원 가입 절차 대신, 카카오톡으로 서비스를 시작할 수 있습니다.  
상태가 OFF일 때도 카카오 로그인 설정 항목을 변경하고 서버에 저장할 수 있습니다.  
상태가 ON일 때만 실제 서비스에서 카카오 로그인 화면이 연결됩니다.

### OpenID Connect 활성화 설정

상태

**ON**

카카오 로그인의 확장 기능인 OpenID Connect를 활성화합니다.  
이 설정을 활성화하면 카카오 로그인 시 사용자 인증 정보가 담긴 ID 토큰을 액세스 토큰과 함께 발급받을 수 있습니다.

### Redirect URI


삭제


수정

Redirect URI	
	https://k7a405.p.ssafy.io/user/signin/redirect
	http://localhost:5173/user/signin/redirect

- 카카오 로그인에서 사용할 OAuth Redirect URI를 설정합니다. (최대 10개)
- REST API로 개발하는 경우 필수로 설정해야 합니다.

플랫폼에 도메인 열어두기



**꽃바다**


ID 814126
OWNER
Biz
Web

### Android

Android 플랫폼 등록

### iOS

iOS 플랫폼 등록

### Web

삭제 수정


사이트 도메인	https://k7a405.p.ssafy.io http://localhost:5173 http://localhost:8080
---------	---

- 카카오 로그인 사용 시 Redirect URI를 등록해야 합니다. [등록하러 가기](#)

## 카카오페이 데모 API 참고사항

### Kakao Developers

이 문서는 카카오페이 API를 소개합니다. 카카오페이 API는 PC웹, 모바일 웹, 모바일 앱 등 다양한 환경에서 카카오 페이로 결제하는 기능을 제공합니다. 데모 페이지에서 카카오페이를 통한 결제 프로세스를 체험해볼 수 있으며, [도구] 메뉴에서 카카오페이 결제 버튼 리소스 다운로드가 가능합니다. 카카오페이 API를 사용하려면 [내 애플리케이션]


 <https://developers.kakao.com/docs/latest/ko/kakaopay/common>

kakao developers

## 카카오톡 공유 API 참고사항

### Kakao Developers

이 문서는 Kakao SDK for JavaScript(이하 JavaScript SDK)를 사용한 카카오톡 공유 API 구현 방법을 안내합니다. 이 문서에 포함된 기능 일부는 [도구] > [JS 데모]를 통해 사용해 볼 수 있습니다. JavaScript SDK의 Kakao.Share 모듈을 사용해 카카오톡 공유 기능을 구현할 수 있습니다. 아래 내용을 참고하여 보낼 메시지의 종류와 사용할 함수를

 <https://developers.kakao.com/docs/latest/ko/message/js-link>

kakao developers

## 배포를 위한 매뉴얼

### 도커 사용

### certificate key, gpg key 받기

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

## 도커 설치

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

## docker-compose.yml에 아래 내용 넣기

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

```
Processing triggers for systemd (245.4-4ubuntu3.1)
ubuntu@ip-172-26-4-76:~$ vim docker-compose.yml
```

```
version: '3'

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

## docker-compose.yml 파일 실행 중 에러가 생긴다면?

```
$ sudo service docker status
```

위 명령어를 통해 현재 잘 돌고 있는지 확인하기

```
ubuntu@ip-172-26-4-76:~$ docker-compose up -d
ERROR: Couldn't connect to Docker daemon at http+docker://localhost - is it running?

If it's at a non-standard location, specify the URL with the DOCKER_HOST environment variable.
ubuntu@ip-172-26-4-76:~$ ls
docker-compose.yml
ubuntu@ip-172-26-4-76:~$ sudo service docker status
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2022-10-28 04:22:15 UTC; 2min 22s ago
   TriggeredBy: ● docker.socket
     Docs: https://docs.docker.com
    Main PID: 110537 (dockerd)
      Tasks: 9
     Memory: 22.3M
    CGroup: /system.slice/docker.service
            └─110537 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
```

그래도 안되면 권한 확인해보기

```
$ sudo ls -la /var/run/docker.sock
// srw-rw---- 1 root docker 0 Jun 11 17:25 /var/run/docker.sock
// 가 나와야함
```

그래도 안되면 아래 명령어를 통해 유저를 추가하고 서버 재부팅하기

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

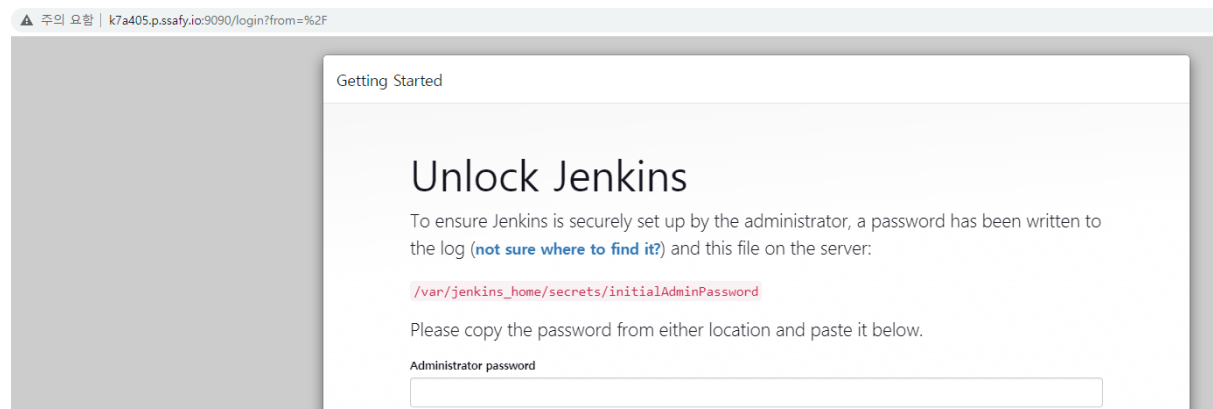
## 젠킨스 사용

### 젠킨스 띄우기

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

```
ubuntu@ip-10-0-2-10:~$ sudo docker ps
CONTAINER ID   IMAGE                  COMMAND                  CREATED        STATUS        PORTS
1a5b4c6d7e8f   jenkins/jenkins:lts   "/usr/bin/tini -- /u...  2 minutes ago   Up 2 minutes   50000/tcp, 0.0.0.0:9090->8080/tcp, :::9090->8080/tcp
jenkins
```

여기까지 하고 나서 도메인에 접속하면 아래와 같은 화면 확인이 가능!



여기 입력해야 할 비밀번호는

```
sudo docker logs jenkins
```

를 입력하면 볼 수 있다!

```
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access of
WARNING: All illegal access operations will be denied in a future release
2022-10-28 04:29:41.383+0000 [id=34] INFO jenkins.install.SetupWizard#init:

*****
*****
*****

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

[REDACTED]

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

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

그걸 다시 암호 입력창에 넣어준다

## 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

이제 왼쪽거 클릭

## Customize Jenkins

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

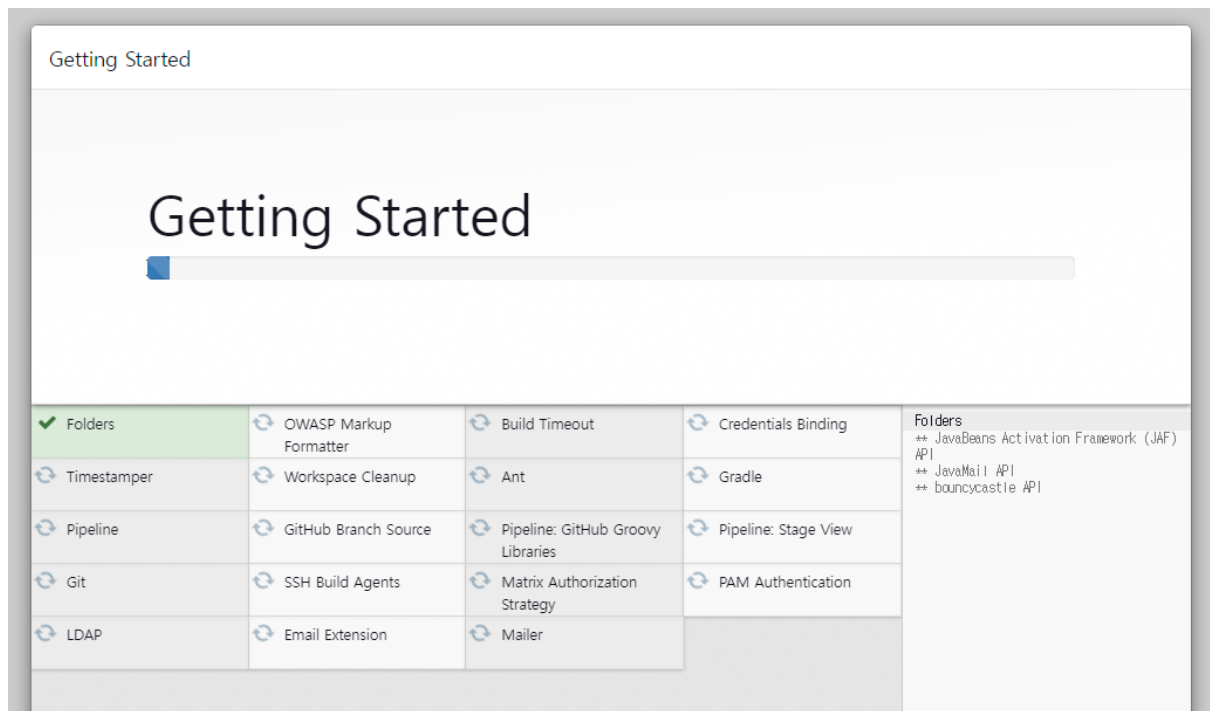
**Install suggested plugins**

Install plugins the Jenkins community finds most useful.

**Select plugins to install**

Select and install plugins most suitable for your needs.

우리는 기본이면 충분하니까..



계정을 잘 만들어주면 끝

## Instance Configuration

Jenkins URL:

<http://k7a405.p.ssafy.io:9090/>

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.

### 젠킨스 설정

- 대시보드의 Jenkins 관리에서 플러그인 매니저에 들어가서 필요한 친구들을 깔아주자!

↑ 대시보드로 돌아가기

⚙ Jenkins 관리

📦 Update Center

## Plugin Manager

업데이트된 플러그인 목록

설치 가능

설치된 플러그인 목록

고급

Q gitlab

Install Name ↓

☐ **GitLab** 1.5.36  
Build Triggers

This plugin allows **GitLab** to trigger Jenkins builds and display their results in the GitLab UI.

This plugin is up for adoption! We are looking for new maintainers. Visit our [Adopt a Plugin](#) i

Q gitlab

Install Name ↓

☒ **GitLab** 1.5.36  
Build Triggers

This plugin allows **GitLab** to trigger Jenkins builds and display their results in the GitLab UI.

This plugin is up for adoption! We are looking for new maintainers. Visit our [Adopt a Plugin](#) i

☒ **Generic Webhook Trigger** 1.85.2  
notification github webhook Build Parameters gitlab

Can receive any HTTP request, extract any values from JSON and many more.

☒ **Gitlab API** 5.0.1-78.v47a\_45b\_9f78b\_7  
Library plugins (for use by other plugins)

This plugin provides **GitLab API** for other plugins.

☒ **GitLab Authentication** 1.16  
Authentication and User Management

This is the an authentication plugin using gitlab OAuth.

This plugin is up for adoption! We are looking for new maintainers. Visit our [Adopt a Plugin](#) i

Q docker

Install Name ↓

☒ **Docker** 1.2.10  
Cloud Providers Cluster Management docker

This plugin integrates Jenkins with **Docker**

This plugin is up for adoption! We are looking for new maintainers. Visit our [Adopt a Plugin](#) i

☒ **Docker Commons** 1.21  
Library plugins (for use by other plugins) docker

Provides the common shared functionality for various Docker-related plugins.

☒ **Docker Pipeline** 528.v7c193a\_0b\_e67c  
pipeline DevOps Deployment docker

Build and use Docker containers from pipelines.

☒ **Docker API** 3.2.13-37.vf3411c9828b9  
Library plugins (for use by other plugins) docker

This plugin provides **docker-java** API for other plugins.

This plugin is up for adoption! We are looking for new maintainers. Visit our [Adopt a Plugin](#) i

☒ **Publish Over SSH** 1.24  
Artifact Uploaders Build Tools  
Send build artifacts over SSH

설치를 기다리는 어린양..



GitLab	✓ 성공
Generic Webhook Trigger	✓ 성공
Gitlab API	✓ 성공
GitLab Authentication	✓ 성공
Loading plugin extensions	✓ Success
Authentication Tokens API	✓ 성공
Docker Commons	✓ 성공
Docker API	⋯ 대기중
Docker	⋯ 대기중
Docker Commons	⋯ 대기중
Docker Pipeline	⋯ 대기중
Docker API	⋯ 대기중
Infrastructure plugin for Publish Over X	⋯ 대기중
Publish Over SSH	⋯ 대기중
Loading plugin extensions	⋯ Pending

## 아이템 만들기

이제 Freestyle Project를 만들어보자!

왼쪽에 있는 새로운 item를 눌러 아이템을 만들수 있다!

Dashboard > All >

Enter an item name

» Required field

**Freestyle project**

이것은 Jenkins의 주요 기능입니다. Jenkins은 어느 빌드 시스템과 어떤 SCM(형상관리)으로 묶인 당신의 프로젝트를 빌드할 것이고, 소프

**Pipeline**

Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows)

**Multi-configuration project**

다양한 환경에서의 테스트, 플러그인 특성 빌드, 기타 동등 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 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

다음으로 넘어가서  
아래와 같이 입력한다  
근데 에러가 난다? (정상이다)

#### 소스 코드 관리

☐ None

☒ Git ?

Repositories ?

Repository URL ?

**Failed to connect to repository :** Command "git ls-remote -h -- https://lab.ssafy.com/s07-final/S07P31A405 HEAD" returned status code 128:  
 stdout:  
 stderr: remote: HTTP Basic: Access denied. The provided password or token is incorrect or your account has 2FA enabled and you must use a personal access token instead of a password.  
 See https://lab.ssafy.com/help/topics/git/troubleshooting\_git#error-on-git-fetch-http-basic-access-denied  
 fatal: Authentication failed for 'https://lab.ssafy.com/s07-final/S07P31A405.git/'

Credentials ?

- none -

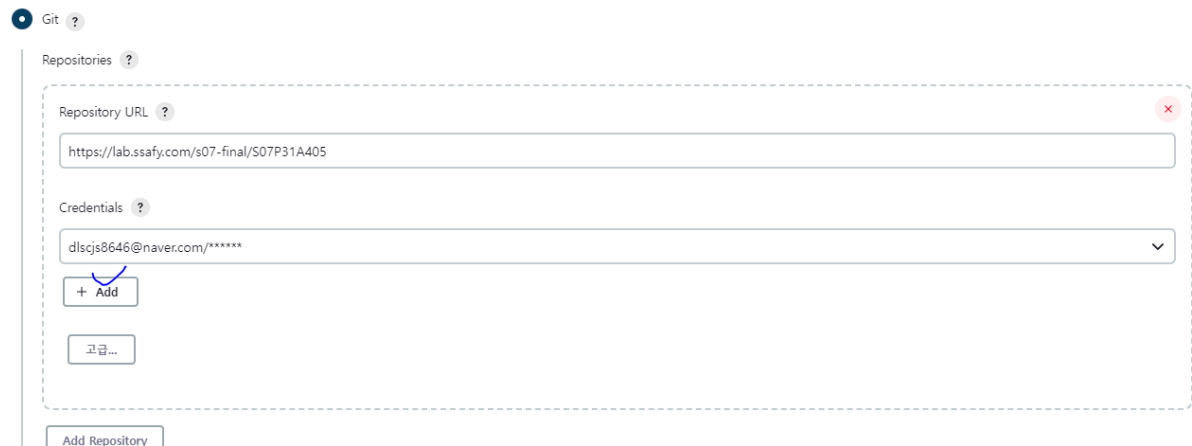
+ Add

고급...

Credentials를 추가해준다

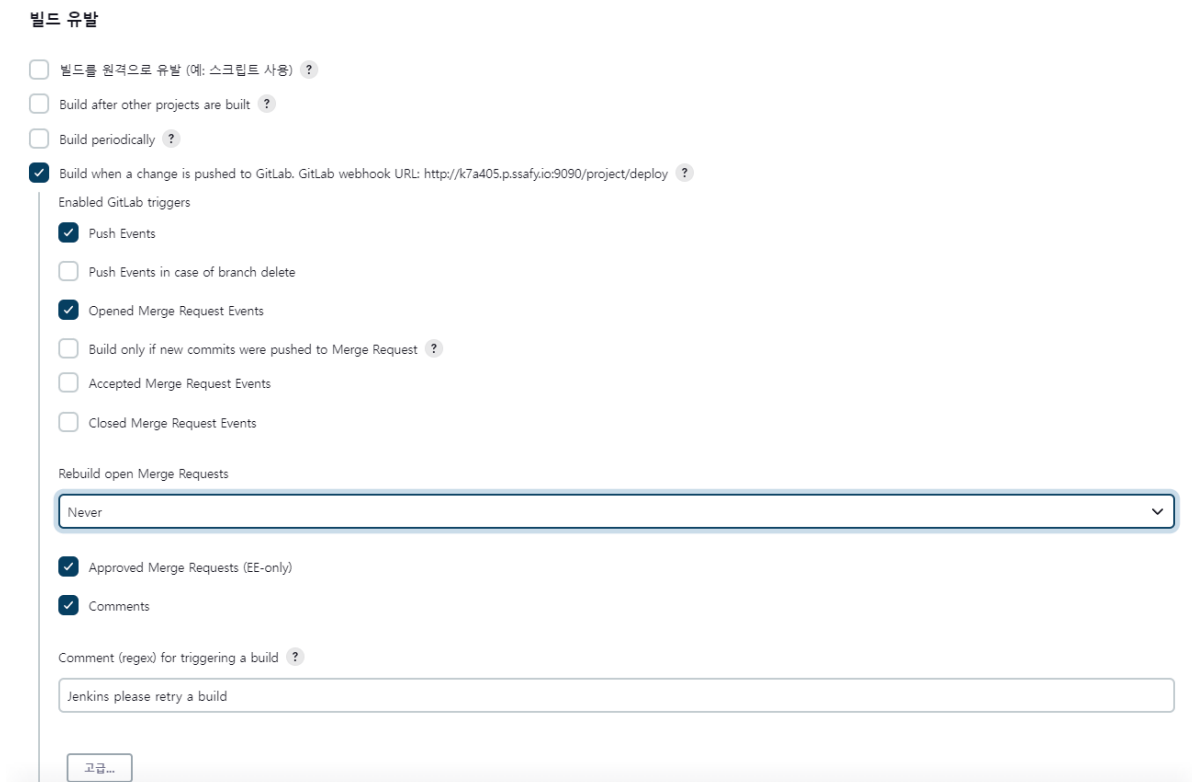
Username에는 깃랩아이디, password에는 깃랩비번, id에 Credential 구분자를 넣어준다

다 만들고 Credentials를 누르면 아까 생겼던 오류가 사라진다!



빌드 유발 조건을 아래와 같이 만든다

(추후에 accepted merge request event로 바꿔줬음)



맨 아래 고급 버튼을 누르고 좀 더 내리면 Secret Token이 나온다!

Generate를 해주자

☐ 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 [?](#)



[Generate](#)

[Clear](#)

Build Step으로 넘어가서

Execute shell을 선택해준다

지금은 테스트니 pwd만 입력해서 현재 경로가 잘 나오는지 확인하자

Build Steps

≡

Execute shell

?

✕

Command

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

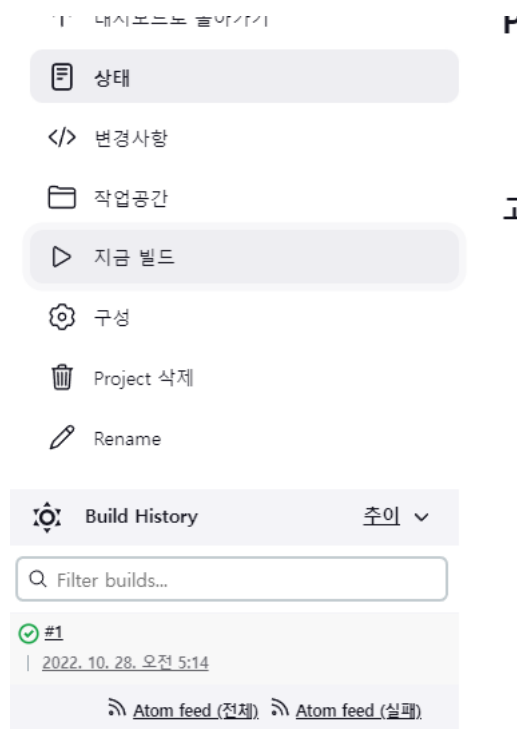
pwd

고급...

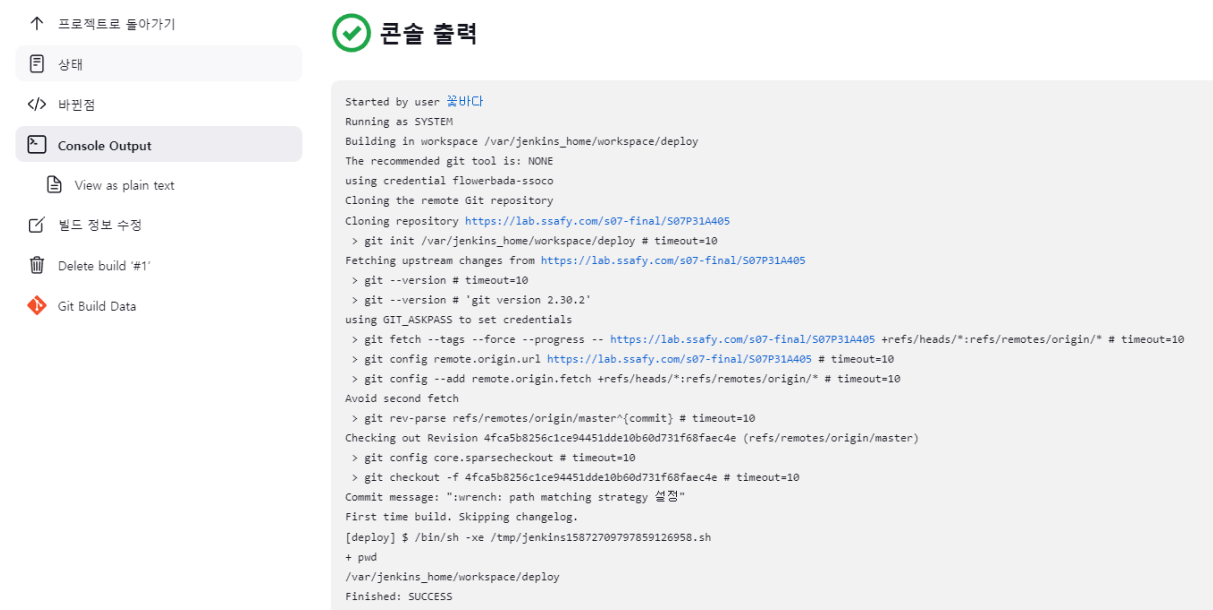
Add build step ▾

여기까지 하고 저장버튼을 누르자~!

지금 빌드 버튼을 눌러서 잘 돌아가는지 확인하자!



콘솔 확인해봐도 쟁쟁되는걸 확인할 수 있다



## 깃랩 설정

이제 깃랩 레포지의 설정의 웹훅으로 들어가자

여기서 URL에는 도메인/project/아까만든젠킨스이름

Secret token에는 Generate한 시크릿 토큰을 넣어준다

트리거에서 마스터에 푸시되면 웹훅이 동작하도록 해주고,

MR이 되도록 동작하도록 해주면 끝!

### Webhook

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.

이제 웹훅이 생성되었으면, Test를 누르고 Push Events를 눌러준다.

**Project Hooks (1)**

http://[redacted]/project/deploy/

Push Events

Merge Requests Events

SSL Verification: enabled

Test ✓

Edit

Delete

젠킨스를 확인해보면 빌드가 잘 된 모습을 볼 수 있다!

S	빌드	경과시간 ↑
✓	deploy #3	10 sec

이 작업을 프론트엔드와 백엔드로 나누어서 2개로 쪼갬다!

## 도커 연결

젠킨스에 도커를 깔자!

먼저 젠킨스 컨테이너에 들어가야 한다 (root 권한이 있다면 sudo를 빼도 된다!)

(중간에 sudo su는 계속 sudo 치기 귀찮아서....)

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

```
ubuntu@ip-173-0-0-78:~$ sudo docker ps
CONTAINER ID   IMAGE                                COMMAND                  NAMES          CREATED
b88756c0a340   jenkins/jenkins:lts               "/usr/bin/tini -- /u... About an
cp, 0.0.0.0:9090→8080/tcp, :::9090→8080/tcp   jenkins
ubuntu@ip-173-0-0-78:~$ sudo su
root@ip-173-0-0-78:~# sudo docker exec -it jenkins bash
root@b88756c0a340:/#
```

아까랑 같은 명령어로 certificate와 gpg키를 받아주자~

```
root@b88756c0a340:/# apt update
apt-get install -y ca-certificates \
    curl \
    software-properties-common \
    apt-transport-https \
    gnupg \
    lsb-release
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]
Get:5 http://deb.debian.org/debian-security bullseye-security/main amd64 Packages [193 kB]
Get:6 http://deb.debian.org/debian bullseye-updates/main amd64 Packages [14.6 kB]
Get:7 https://packagecloud.io/github/git-lfs/debian bullseye InRelease [24.4 kB]
Get:8 https://packagecloud.io/github/git-lfs/debian bullseye/main amd64 Packages [1728 B]
```

다만 ubuntu가 아니라 devian 환경이기 때문에 ubuntu자리에 devian만 넣어주면 된다!

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

도커도 깔아주자

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

클라이언트와 백엔드에 도커파일을 작성하자

- 백엔드

```
FROM openjdk:11-jdk
EXPOSE 8080
ARG JAR_FILE=build/libs/flower-0.0.1-SNAPSHOT.jar
COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java", "-jar", "/app.jar"]
```

- 프론트엔드

```
FROM node:16

WORKDIR /app

COPY package*.json ./
COPY yarn.lock ./

RUN yarn

COPY . ./

RUN yarn build

EXPOSE 5173

CMD ["yarn", "dev" ]
```

필요에 따라 .dockerignore도 작성해주자

```
/node_modules
```

이제 대시보드로 돌아가서 진짜 Execute shell 코드를 넣어주자

(mysql 부분이 있는데, 이건 아래에서 만든다)

```
cd backend
docker rm spring -f
chmod +x gradlew
./gradlew bootJar
docker build . -t flowerbada_server:latest
docker run -d -p 8080:8080 --link mysql-flowerbada --name spring flowerbada_server:latest
```

프론트쪽도 똑같이 넣어주자

(Vite의 경우 5173이 기본포트인데, ec2 서버가 5173포트를 허락하지 않아서.. 3000으로 접속해서 5173으로 포트포워딩했다)

```
cd frontend
docker rm frontend -f
docker image prune -a --force
docker build . -t flowerbada_client:latest
docker run -d -p 3000:5173 --name frontend flowerbada_client:latest
```

잘 작동하는 것을 확인하자



상세 내용 입력

All

+

S	W	Name ↓	최근 성공	최근 실패	최근 소요 시간	
✓	🔗	backend	18 min #22	19 min #21	10 sec	▶
✓	🔗	frontend	19 min #11	53 min #10	1 min 12 sec	▶

아이콘: S M L

Icon legend

Atom feed 모두

Atom feed 실패

Atom feed 최근 빌드

젠킨스 컨테이너 안에 내용물도 잘 있음

```
root@b88756c0a340:/var/jenkins_home/workspace/frontend# ls
backend backend@tmp deploy deploy@tmp frontend frontend@tmp
root@b88756c0a340:/var/jenkins_home/workspace#
```

## 도커 컨테이너 생성

젠킨스 관리의 시스템 설정으로 이동한다

Dashboard > Jenkins 관리 > Configure System >

☐ Test configuration by sending test e-mail

**Publish over SSH**

Jenkins SSH Key ?

Passphrase ?

Path to key ?

Key ?

☐ Disable exec ?

SSH Servers

추가

고급...

저장 Apply

**추가** 버튼을 누르고 내용을 입력해준다

name은 그냥 이름


hostname은 서버 아이피

username에는 ubuntu를 넣는다(이것도 지정해둔 아이디로!!)

#### SSH Servers

☰ SSH Server

Name ?  
flowerbada

Hostname ?  


Username ?  
ubuntu


조금 아래로 내려서 보안키를 등록해준다

지급받은 pem파일을 메모장으로 열어서 나온 내용을 전부 복사해준다

☒ Use password authentication, or use a different key ?

Passphrase / Password ?

Path to key ?

Key ?  


아래쪽에 있는 Test Configuration을 눌러서 success가 나오면 성공

Proxy password

Success  

Test Configuration

## MySQL

## MySQL 연결

```
// mysql 이미지 땡겨오기
docker pull mysql:8

// 계정 생성
docker run --name mysql-flowerbada -e MYSQL_ROOT_PASSWORD=ssafy -p 3306:3306 mysql:8

// mysql 컨테이너로 접속
docker exec -it mysql-flowerbada bash

// 루트 계정으로 접속 후 비밀번호 입력
mysql -u root -p
```

```
// DB 생성
CREATE DATABASE flower_db CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;

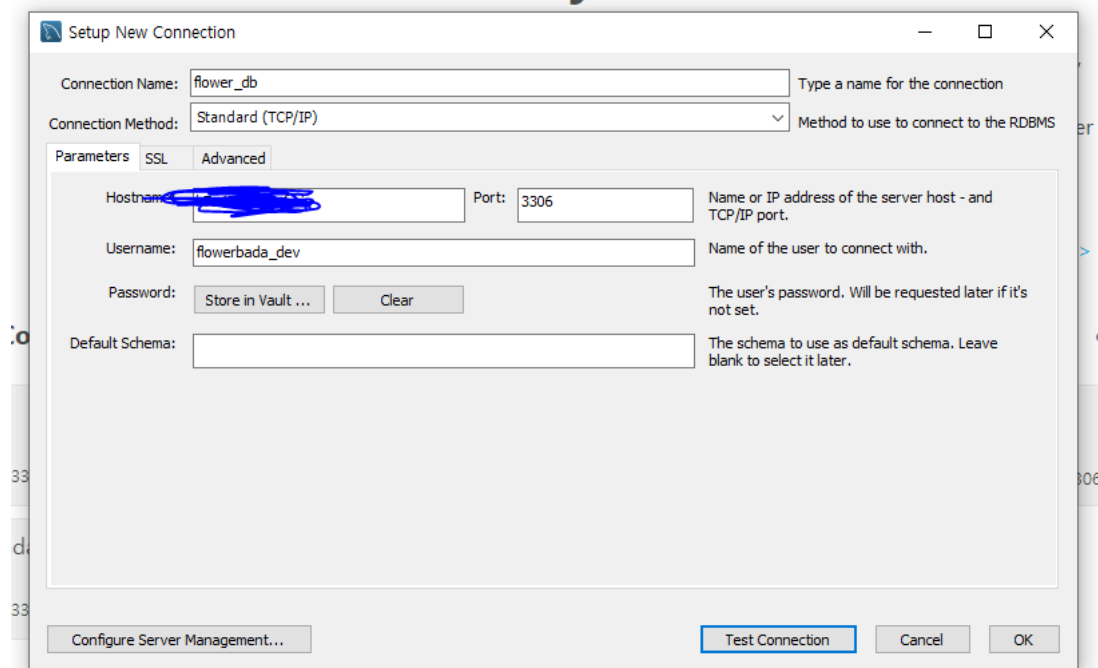
// 계정 생성
CREATE USER 'flowerbada_dev'@'%' IDENTIFIED BY '비밀번호';

// 유저 확인
USE mysql;
SELECT user FROM user;

// 권한 부여
GRANT ALL PRIVILEGES ON flower_db.* to flowerbada_dev@'%';
FLUSH PRIVILEGES;
```

여기까지 마치면 mysql 도커 컨테이너가 서버에 계속 돌게됨

이걸 mysql workbench에서 확인하려면 아래와 같이 접속할 수 있음!



## 포트관리

### 포트열기

```
ufw allow 22
ufw enable (방화벽 켜기)
// 반드시 22번 포트는 열어놓고 enable하기
```

To	Action	From
--	----	----
5173	ALLOW	Anywhere
80	ALLOW	Anywhere
443	ALLOW	Anywhere
8080	ALLOW	Anywhere
9090	ALLOW	Anywhere
5173 (v6)	ALLOW	Anywhere (v6)
80 (v6)	ALLOW	Anywhere (v6)
443 (v6)	ALLOW	Anywhere (v6)
8080 (v6)	ALLOW	Anywhere (v6)
9090 (v6)	ALLOW	Anywhere (v6)

... 22번 포트를 안열어서 서버가 날아가버렸다

다시 포트 조심히 열기

```
root@ip-10.10.10.10:/home/ubuntu# ufw allow 5173
Rules updated
Rules updated (v6)
root@ip-10.10.10.10:/home/ubuntu# ufw allow 8080
Rules updated
Rules updated (v6)
root@ip-10.10.10.10:/home/ubuntu# ufw allow 9090
Rules updated
Rules updated (v6)
root@ip-10.10.10.10:/home/ubuntu# ufw allow 3306
Rules updated
Rules updated (v6)
root@ip-10.10.10.10:/home/ubuntu# ufw status
Status: inactive
root@ip-10.10.10.10:/home/ubuntu# ufw allow 80
Rules updated
Rules updated (v6)
root@ip-10.10.10.10:/home/ubuntu# ufw allow 443
Rules updated
Rules updated (v6)
root@ip-10.10.10.10:/home/ubuntu#
```

To	Action	From
--	----	----
22	ALLOW	Anywhere
5173	ALLOW	Anywhere
8080	ALLOW	Anywhere
9090	ALLOW	Anywhere
3306	ALLOW	Anywhere
80	ALLOW	Anywhere
443	ALLOW	Anywhere
22 (v6)	ALLOW	Anywhere (v6)
5173 (v6)	ALLOW	Anywhere (v6)
8080 (v6)	ALLOW	Anywhere (v6)
9090 (v6)	ALLOW	Anywhere (v6)
3306 (v6)	ALLOW	Anywhere (v6)
80 (v6)	ALLOW	Anywhere (v6)
443 (v6)	ALLOW	Anywhere (v6)

## 인증서 발급 및 기타 작업

### 비밀자료 넣기

도커 이미지가 저장된 폴더에 들어간다

그리고 touch를 통해서 application.yml이나 기타 yaml파일을 생성한다.

(젠킨스 컨테이너 안에 있음)

```
jenkins.telemetry.correlator.xml workspace
root@58add3028274:/var/jenkins_home# cd workspace
root@58add3028274:/var/jenkins_home/workspace# ls
backend backend@tmp frontend frontend@tmp
root@58add3028274:/var/jenkins_home/workspace# cd backend
root@58add3028274:/var/jenkins_home/workspace/backend# ls
backend frontend
root@58add3028274:/var/jenkins_home/workspace/backend# cd backend
root@58add3028274:/var/jenkins_home/workspace/backend/backend# ls
build.gradle dockerfile gradle gradlew gradlew.bat settings.gradle src
root@58add3028274:/var/jenkins_home/workspace/backend/backend# cd src
root@58add3028274:/var/jenkins_home/workspace/backend/backend/src# ls
main test
root@58add3028274:/var/jenkins_home/workspace/backend/backend/src# cd main
root@58add3028274:/var/jenkins_home/workspace/backend/backend/src/main# ls
java
root@58add3028274:/var/jenkins_home/workspace/backend/backend/src/main# mkdir resources
root@58add3028274:/var/jenkins_home/workspace/backend/backend/src/main# ls
java resources
root@58add3028274:/var/jenkins_home/workspace/backend/backend/src/main# cd resources
root@58add3028274:/var/jenkins_home/workspace/backend/backend/src/main/resources# touch
```

nano를 이용해서 데이터를 넣어주자.

```
update-alternatives: using /usr/bin/nano to provide /usr/bin/ptex (ptex) in auto mode
root@58add3028274:/var/jenkins_home/workspace/backend/backend/src/main/resources# nano applicatio
n.yml
root@58add3028274:/var/jenkins_home/workspace/backend/backend/src/main/resources# nano env.yml
root@58add3028274:/var/jenkins_home/workspace/backend/backend/src/main/resources#
```

## HTTPS 적용을 위한 letsencrypt 인증서 발급

standalone방식으로 발급받기

```
sudo apt update
sudo apt install certbot -y
certbot certonly --standalone -d 사이트명
```

이렇게 하면 /etc/letsencrypt/live/도메인명 에 인증서가 추가된다.

```
0 upgraded, 0 newly installed, 0 to remove and 249 not upgraded.
root@ip-172-26-15-179:~/secrets# certbot certonly --standalone -d k7a405.p.ssafy.io
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Plugins selected: Authenticator standalone, Installer None
Obtaining a new certificate
Performing the following challenges:
http-01 challenge for k7a405.p.ssafy.io
Waiting for verification...
Cleaning up challenges

IMPORTANT NOTES:
- Congratulations! Your certificate and chain have been saved at:
  /etc/letsencrypt/live/k7a405.p.ssafy.io/fullchain.pem
  Your key file has been saved at:
  /etc/letsencrypt/live/k7a405.p.ssafy.io/privkey.pem
  Your cert will expire on 2023-01-31. To obtain a new or tweaked
  version of this certificate in the future, simply run certbot
  again. To non-interactively renew *all* of your certificates, run
  "certbot renew"
- If you like Certbot, please consider supporting our work by:

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

root@ip-172-26-15-179:~/secrets#
```

추가적인 연장을 원한다면 certbot을 사용하자

## NginX 설치 및 리버스프록시와 HTTPS 설정

```
# Nginx 설치
$ sudo apt install nginx

# Nginx 실행
$ sudo service nginx start

# Nginx 확인
$ sudo service nginx status
```

```
Processing triggers for libc-bin (2.31-0ubuntu9) ...
root@ip-10-10-10-10:~/.secrets# service nginx start
root@ip-10-10-10-10:~/.secrets# service nginx status
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2022-11-02 18:39:24 UTC; 47s ago
     Docs: man:nginx(8)
    Main PID: 68991 (nginx)
      Tasks: 5 (limit: 19204)
     Memory: 5.9M
    CGroup: /system.slice/nginx.service
            └─68991 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;
               └─68992 nginx: worker process
                  └─68993 nginx: worker process
                     └─68994 nginx: worker process
                        └─68995 nginx: worker process

Nov 02 18:39:24 ip-10-10-10-10 systemd[1]: Starting A high performance web server and a reverse proxy server ...
Nov 02 18:39:24 ip-10-10-10-10 systemd[1]: Started A high performance web server and a reverse proxy server.
root@ip-10-10-10-10:~/.secrets#
```

certbot의 도움을 받아 nginx를 작성하기 위해 python3-certbot-nginx를 받자

```
# 우분투 20 이후 버전
$ sudo apt-get install python3-certbot-nginx

# 우분투 20 이전 버전 (16, 18...)
$ sudo apt-get install python-certbot-nginx
```

자동 적용 도우미를 실행하면 된다.

```
$ sudo certbot --nginx
```

위 명령어를 치면 순서대로 도메인, 인증서 설정, 다이렉트 여부가 나온다.

첫 번째는 그냥 도메인 입력하면 된다.

두 번째는 인증서가 있으면 1, 새로운 간이 인증서를 만드려면 2를 입력한다.

세 번째는 http를 https로 리다이렉트하지 않으려면 1, 모든 http요청에 대해 리다이렉트하려면 2를 입력한다.

```

root@ip-10-10-10-10:~/.secrets# certbot --nginx
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Plugins selected: Authenticator nginx, Installer nginx
No names were found in your configuration files. Please enter in your domain
name(s) (comma and/or space separated) (Enter 'c' to cancel): k7a405.p.ssafy.io
Cert not yet due for renewal

You have an existing certificate that has exactly the same domains or certificate name you requested and isn't close to expiry.
(ref: /etc/letsencrypt/renewal/k7a405.p.ssafy.io.conf)

What would you like to do?
-----
1: Attempt to reinstall this existing certificate
2: Renew & replace the cert (limit ~5 per 7 days)
-----
Select the appropriate number [1-2] then [enter] (press 'c' to cancel): 1
Keeping the existing certificate
Deploying Certificate to VirtualHost /etc/nginx/sites-enabled/default

Please choose whether or not to redirect HTTP traffic to HTTPS, removing HTTP access.
-----
1: No redirect - Make no further changes to the webserver configuration.
2: Redirect - Make all requests redirect to secure HTTPS access. Choose this for
new sites, or if you're confident your site works on HTTPS. You can undo this
change by editing your web server's configuration.
-----
Select the appropriate number [1-2] then [enter] (press 'c' to cancel): 2
Redirecting all traffic on port 80 to ssl in /etc/nginx/sites-enabled/default
-----
Congratulations! You have successfully enabled https://k7a405.p.ssafy.io

You should test your configuration at:
https://www.ssllabs.com/ssltest/analyze.html?d=k7a405.p.ssafy.io
-----

IMPORTANT NOTES:
- Congratulations! Your certificate and chain have been saved at:
  /etc/letsencrypt/live/k7a405.p.ssafy.io/fullchain.pem
  Your key file has been saved at:
  /etc/letsencrypt/live/k7a405.p.ssafy.io/privkey.pem
  Your cert will expire on 2023-01-31. To obtain a new or tweaked
  version of this certificate in the future, simply run certbot again
  with the "certonly" option. To non-interactively renew *all* of
  your certificates, run "certbot renew"
- If you like Certbot, please consider supporting our work by:

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

```

이제 페이지를 가보면 잘 적용되었는 모습을 볼 수 있다.

 <https://k7a405.p.ssafy.io>

## Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to [nginx.org](https://nginx.org).  
Commercial support is available at [nginx.com](https://nginx.com).

Thank you for using nginx.

자동으로 만들어진 nginx 설정파일은

/etc/nginx/sites-available/default

에서 확인 가능하다.

아래는 개인적으로 수정한 default.conf 파일의 내용이다.

```

##
# You should look at the following URL's in order to grasp a solid understanding
# of Nginx configuration files in order to fully unleash the power of Nginx.
# https://www.nginx.com/resources/wiki/start/
# https://www.nginx.com/resources/wiki/start/topics/tutorials/config_pitfalls/
# https://wiki.debian.org/Nginx/DirectoryStructure
#
# In most cases, administrators will remove this file from sites-enabled/ and
# leave it as reference inside of sites-available where it will continue to be
# updated by the nginx packaging team.
#

```

```

# This file will automatically load configuration files provided by other
# applications, such as Drupal or Wordpress. These applications will be made
# available underneath a path with that package name, such as /drupal8.
#
# Please see /usr/share/doc/nginx-doc/examples/ for more detailed examples.
##

upstream frontend{
    server k7a405.p.ssafy.io:3000;
}

upstream backend{
    server k7a405.p.ssafy.io:8080;
}

# Default server configuration
#
server {
    listen 80 default_server;
    listen [::]:80 default_server;

    # SSL configuration
    #
    # listen 443 ssl default_server;
    # listen [::]:443 ssl default_server;
    #
    # Note: You should disable gzip for SSL traffic.
    # See: https://bugs.debian.org/773332
    #
    # Read up on ssl_ciphers to ensure a secure configuration.
    # See: https://bugs.debian.org/765782
    #
    # Self signed certs generated by the ssl-cert package
    # Don't use them in a production server!
    #
    # include snippets/snakeoil.conf;

    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 / {
        # First attempt to serve request as file, then
        # as directory, then fall back to displaying a 404.
        try_files $uri $uri/ /index.html;
    }

    # pass PHP scripts to FastCGI server
    #
    #location ~ \.php$ {
    #    include snippets/fastcgi-php.conf;
    #
    #    # With php-fpm (or other unix sockets):
    #    fastcgi_pass unix:/var/run/php/php7.4-fpm.sock;
    #    # With php-cgi (or other tcp sockets):
    #    fastcgi_pass 127.0.0.1:9000;
    #}

    # deny access to .htaccess files, if Apache's document root
    # concurs with nginx's one
    #
    #location ~ /\.ht {
    #    deny all;
    #}
}

# Virtual Host configuration for example.com
#
# You can move that to a different file under sites-available/ and symlink that
# to sites-enabled/ to enable it.
#
#server {
#    listen 80;
#    listen [::]:80;
#
#    server_name example.com;
#
#    root /var/www/example.com;
#    index index.html;
#
#    location / {
#        try_files $uri $uri/ =404;

```



```

#     }
#}

server {

    # SSL configuration
    #
    # listen 443 ssl default_server;
    # listen [::]:443 ssl default_server;
    #
    # Note: You should disable gzip for SSL traffic.
    # See: https://bugs.debian.org/773332
    #
    # Read up on ssl_ciphers to ensure a secure configuration.
    # See: https://bugs.debian.org/765782
    #
    # Self signed certs generated by the ssl-cert package
    # Don't use them in a production server!
    #
    # include snippets/snakeoil.conf;

    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 k7a405.p.ssafy.io; # managed by Certbot

    #location / {
        # First attempt to serve request as file, then
        # as directory, then fall back to displaying a 404.
        #try_files $uri $uri/ /index.html;

        #}

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

            #
            # With php-fpm (or other unix sockets):
            fastcgi_pass unix:/var/run/php/php7.4-fpm.sock;
            # With php-cgi (or other tcp sockets):
            fastcgi_pass 127.0.0.1:9000;
            #}

            # deny access to .htaccess files, if Apache's document root
            # concurs with nginx's one
            #
            #location ~ /\.ht {
                # deny all;
            #}

            listen [::]:443 ssl ipv6only=on; # managed by Certbot
            listen 443 ssl; # managed by Certbot
            ssl_certificate /etc/letsencrypt/live/k7a405.p.ssafy.io/fullchain.pem; # managed by Certbot
            ssl_certificate_key /etc/letsencrypt/live/k7a405.p.ssafy.io/privkey.pem; # managed by Certbot
            include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
            ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot

            location / {
                proxy_pass http://frontend;
            }

            location /api {
                proxy_pass http://backend;
            }

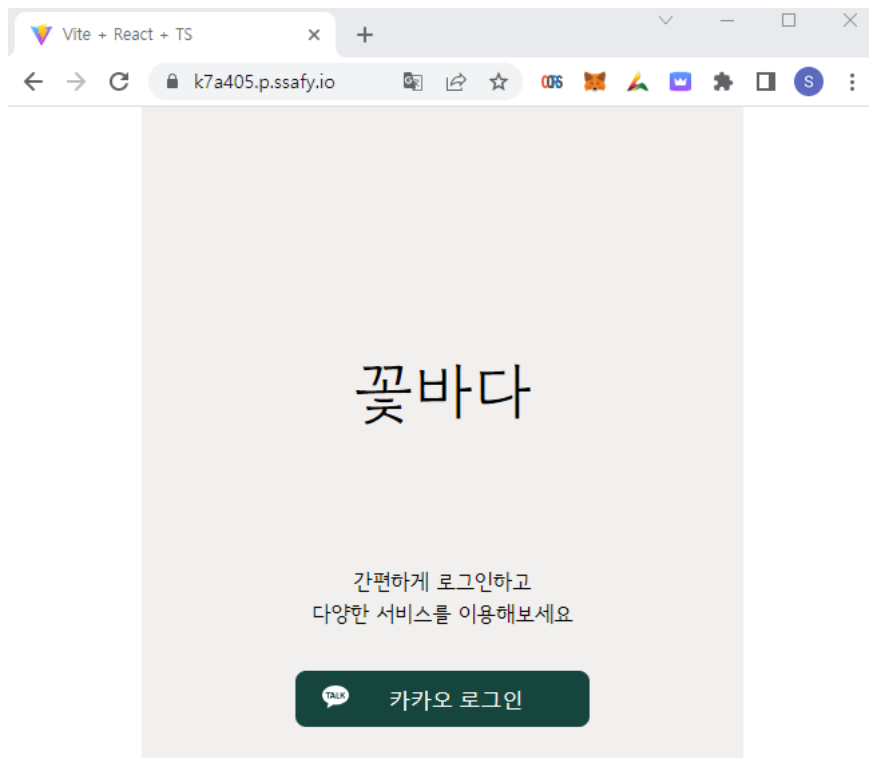
        }
    }
    server {
        if ($host = k7a405.p.ssafy.io) {
            return 301 https://$host$request_uri;
        } # managed by Certbot

        listen 80 ;
        listen [::]:80 ;
        server_name k7a405.p.ssafy.io;
        return 404; # managed by Certbot

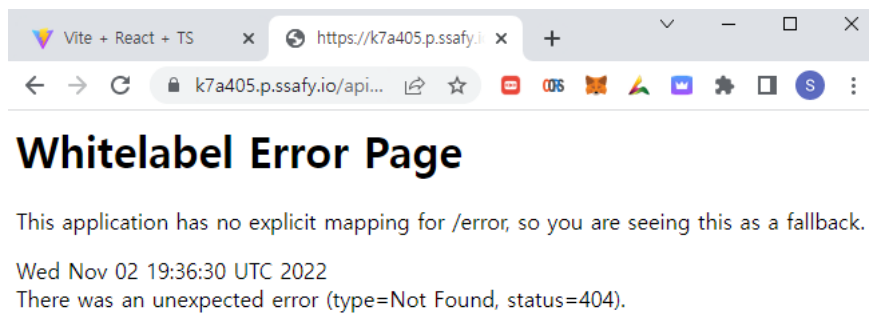
    }
}

```

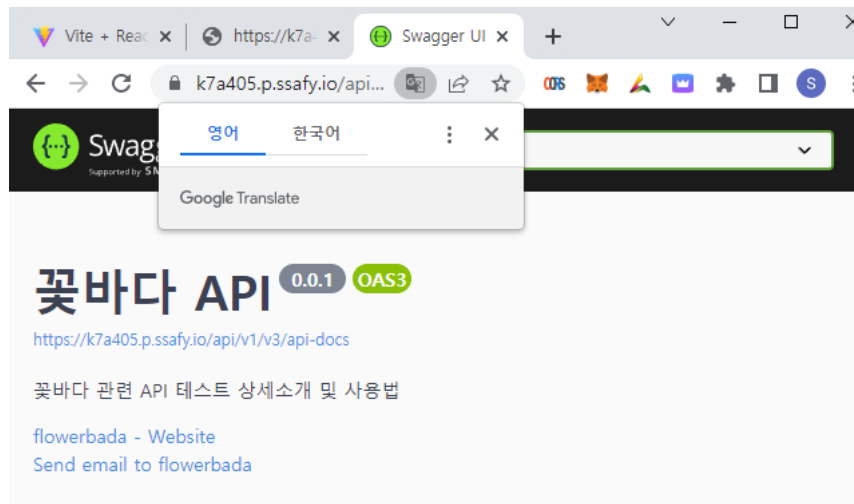
여기까지만 하면 정말 잘 작동한다!!



백엔드도 동일하게 적용되었다!



스웨거도 잘된다!



이제 각 설정에 맞게 서버로 주소를 바꾸면 된다.

(일괄로 관리하는 것이 편할 듯 싶다)

## 백엔드 https 적용을 위한 p12키 만들기

pem키가 저장된 폴더로 이동한 후 아래 명령어를 작성한다.

비밀번호는 굳이 입력하지 않아도 될 것 같아서 엔터만 두번 눌렀다

```
openssl pkcs12 -export -in fullchain.pem -inkey privkey.pem -out keystore.p12 -name airpageserver -CAfile chain.pem -caname root
```

```
root@ip-10.0.2.15:~# cd /etc/letsencrypt/live
root@ip-10.0.2.15:/etc/letsencrypt/live# ls
README  k7a405.p.ssafy.io
root@ip-10.0.2.15:/etc/letsencrypt/live# cd k7a405.p.ssafy.io
root@ip-10.0.2.15:/etc/letsencrypt/live/k7a405.p.ssafy.io# ls
README  k7a405.p.ssafy.io
root@ip-10.0.2.15:/etc/letsencrypt/live/k7a405.p.ssafy.io# k7a405.p.ssafy.io
k7a405.p.ssafy.io: command not found
root@ip-10.0.2.15:/etc/letsencrypt/live/k7a405.p.ssafy.io# cd k7a405.p.ssafy.io
root@ip-10.0.2.15:/etc/letsencrypt/live/k7a405.p.ssafy.io# openssl pkcs12 -export -in fullchain.pem -inkey
privkey.pem -out keystore.p12 -name airpageserver -CAfile chain.pem -caname root
Enter Export Password:
Verifying - Enter Export Password:
```

잘 생성됐음

```
root@ip-10.0.2.15:/etc/letsencrypt/live/k7a405.p.ssafy.io# ls
README  cert.pem  chain.pem  fullchain.pem  keystore.p12  privkey.pem
```

```
root@ip-10.0.2.15:/etc/letsencrypt/live/k7a405.p.ssafy.io# docker cp /etc/letsencrypt/live/k7a405.p.ssafy.io/keystore.p12 jenkins:/var/jenkins_home/workspace/backend/backend/src/main/resources
```

application.yml 파일 수정하기

```
# 포트
server:
  port: '8080'
  servlet:
    context-path: /api/v1

ssl:
  key-store: classpath:keystore.p12
  key-store-type: PKCS12
  key-store-password:
```

여기까지 하면 백엔드 https 처리 완료

추가로 카카오로그인의 경우 redirect URL을 바꿔주고, kakao Developers에서도 해당 redirect URL 추가해줘야함

## Docker 컨테이너로 Redis 띄우고 연결하기

```
2022-11-07 04:41:10.550 ERROR 1 --- [nio-8080-exec-8] o.a.c.c.C.[.][dispatcherServlet] : Servlet.service() for servlet [dispatcherServlet] in context with path [/api/v1] threw exception [Request processing failed; nested exception is org.springframework.data.redis.RedisConnectionFailureException: Unable to connect to Redis; nested exception is io.lettuce.core.RedisConnectionException: Unable to connect to localhost:6379] with root cause
java.net.ConnectException: Connection refused
    at java.base/sun.nio.ch.SocketChannelImpl.checkConnect(Native Method) ~[na:na]
    at java.base/sun.nio.ch.SocketChannelImpl.finishConnect(SocketChannelImpl.java:777) ~[na:na]
    at io.netty.channel.socket.nio.NioSocketChannel.doFinishConnect(NioSocketChannel.java:337) ~[netty-transport-4.1.82.Final.jar!/:4.1.82.Final]
```

위와 같은 문제로 레디스가 연결이 안되면 관련 기능들이 전부 마비되서 안된다

먼저 server와 redis컨테이너를 띄기 위해 미리 도커네트워크를 만들어놓자

같은 네트워크에 있으면 컨테이너끼리 통신이 가능해진다!

네트워크 만드는건 아래 명령어로 가능!

```
docker network create redis-network
```

만든 네트워크를 확인하자!

```
docker inspect redis-network
```

```

root@ip-10.10.10.10: /jenkins/workspace/backend/backend/src/main/resources# docker inspect redis-network
[
  {
    "Name": "redis-network",
    "Id": "7eec6b94b3fb5844fa5f1ed1ea5f24cbe3666a5d05ca65b20bc80154efe94258",
    "Created": "2022-11-07T06:16:23.949697129Z",
    "Scope": "local",
    "Driver": "bridge",
    "EnableIPv6": false,
    "IPAM": {
      "Driver": "default",
      "Options": {},
      "Config": [
        {
          "Subnet": "10.10.10.0/16",
          "Gateway": "10.10.10.1"
        }
      ]
    },
    "Internal": false,
    "Attachable": false,
    "Ingress": false,
    "ConfigFrom": {
      "Network": ""
    },
    "ConfigOnly": false,
    "Containers": {},
    "Options": {},
    "Labels": {}
  }
]

```

해당 네트워크에 redis와 server 컨테이너를 엮자!

그 전에 redis.conf를 미리 받아두자

/etc/redis/redis.conf에 저장된다.

```
wget http://download.redis.io/redis-stable/redis.conf -O /etc/redis/redis.conf
```

redis설정에 사용한 redis.conf파일에서 bind를 127.0.0.1이 아닌 0.0.0.0으로 풀어서 외부 접속이 가능하게 만들자 (필수는 아닌듯)

```

#
# You will also need to set a password unless you explicitly disable protected
# mode.
# ~~~~~
bind 0.0.0.0:::1

# By default, outgoing connections (from replica to master, from Sentinel to
# instances, cluster bus, etc.) are not bound to a specific local address. In
# most cases, this means the operating system will handle that based on routing
# and the interface through which the connection goes out.
#
# Using bind-source-addr it is possible to configure a specific address to bind

```

이제 redis 컨테이너를 도커로 띄워보자

아까 받은 redis.conf를 이용하고, 만들어둔 network에 바로 연결시켰다.

```
docker run --name redis-flowerbada -p 6379:6379 --network redis-network -v redis_temp:/data -d redis:latest redis-server --appendonly
```

그리고 백엔드 서버 역시 해당 네트워크에 연결해주자

```
docker network connect redis-network spring
```

이제 백엔드 서버의 application.yml파일에서 redis의 host를 아까 inspect했을 때 Config에 나온 subnet 주소로 바꿔준다

```
# redis
redis:
  host: localhost
  port: 6379
```

이 작업까지 마치면 전부 연결이 완료되었다

하지만.. 이상태로는 젠킨스에 의해 새로운 빌드가 이루어지면 redis-network에서 백엔드서버가 빠져나가는 현상이 발생한다

이걸 해결하기 위해서는 젠킨스 백엔드 서버의 shell script를 변경해주면 된다

```
See the list of available environment variables

chmod +x gradlew
./gradlew bootJar
docker build . -t flowerbada_server:latest
docker run -d -p 8080:8080 --network redis-network --link mysql-flowerbada --name spring flowerbada_s
```

## S3 연결

### Amazon S3(Simple Storage Service)

- AWS에서 제공하는 객체 스토리지 서비스
- 프로필 이미지, NFT 원본 파일을 업로드 및 관리하기 위해 사용

### S3 버킷 접근 권한을 가진 AWS IAM 사용자 생성



AWS의 IAM 페이지에 접근하여 액세스 관리 > 사용자 > 사용자 추가 버튼을 누른다.

그 후, 액세스 키 방식을 선택하고 기존 정책 직접 연결 > AmazonS3FullAccess 를 체크하여 사용자를 추가한다.

## 사용자 추가

1 2 3 4 5

### 사용자 세부 정보 설정

동일한 액세스 유형 및 권한을 사용하여 한 번에 여러 사용자를 추가할 수 있습니다. [자세히 알아보기](#)

사용자 이름\* aws-s3-access

[+ 다른 사용자 추가](#)

### AWS 액세스 유형 선택

이러한 사용자가 주로 AWS에 액세스하는 방법을 선택합니다. 프로그래밍 방식의 액세스만 선택하면 사용자가 위임된 역할을 사용하여 콘솔에 액세스하는 것을 방지할 수 없습니다. 액세스 키와 자동 생성된 암호가 마지막 단계에서 제공됩니다. [자세히 알아보기](#)

#### AWS 자격 증명 유형 선택\*

- ☒ 액세스 키 - 프로그래밍 방식 액세스  
AWS API, CLI, SDK 및 기타 개발 도구에 대해 액세스 키 ID 및 비밀 액세스 키 을(를) 활성화합니다.
- ☐ 암호 - AWS 관리 콘솔 액세스  
사용자가 AWS Management Console에 로그인할 수 있도록 허용하는 비밀번호 을(를) 활성화합니다.

## 사용자 추가

1 2 3 4 5

### ▼ 권한 설정



그룹에 사용자 추가



기존 사용자에서 권한 복사



기존 정책 직접 연결

정책 생성



정책 필터		s3		7 결과 표시	
	정책 이름	유형	사용 용도		
<input type="checkbox"/>	AmazonDMSRedshiftS3Role	AWS 관리형	없음		
<input checked="" type="checkbox"/>	AmazonS3FullAccess	AWS 관리형	Permissions policy (2)		
<input type="checkbox"/>	AmazonS3ObjectLambdaExecutionRolePolicy	AWS 관리형	없음		
<input type="checkbox"/>	AmazonS3OutpostsFullAccess	AWS 관리형	없음		
<input type="checkbox"/>	AmazonS3OutpostsReadOnlyAccess	AWS 관리형	없음		
<input type="checkbox"/>	AmazonS3ReadOnlyAccess	AWS 관리형	없음		
<input type="checkbox"/>	QuickSightAccessForS3StorageManagementAnalyticsReadOnly	AWS 관리형	없음		

## AWS S3 bucket 생성

Amazon S3에 접근하여 버킷 만들기를 클릭하고 퍼블릭 액세스가 가능하도록 버킷을 생성

한다.

그 중에서, 새 ACL(액세스 제어 목록)을 통해 부여된 버킷 및 객체에 대한 퍼블릭 액세스 차단에 대한 허용이 필수!

## 퍼블릭 액세스 차단 편집(버킷 설정) Info

### 퍼블릭 액세스 차단(버킷 설정)

퍼블릭 액세스는 ACL(엑세스 제어 목록), 버킷 정책, 액세스 지점 정책 또는 모두를 통해 버킷 및 객체에 부여됩니다. 모든 S3 버킷 및 객체에 대한 퍼블릭 액세스가 차단되었는지 확인하려면 [모든 퍼블릭 액세스 차단]을 활성화합니다. 이 설정은 이 버킷 및 해당 액세스 지점에만 적용됩니다. AWS에서는 [모든 퍼블릭 액세스 차단]을 활성화하도록 권장하지만, 이 설정을 적용하기 전에 퍼블릭 액세스가 없어도 애플리케이션이 올바르게 작동하는지 확인합니다. 버킷 또는 내부 객체에 어느 정도 수준의 퍼블릭 액세스가 필요한 경우 특정 스토리지 사용 사례에 맞게 아래 개별 설정을 사용자 지정할 수 있습니다. [자세히 알아보기](#)

#### ☐ 모든 퍼블릭 액세스 차단

이 설정을 활성화하면 아래 4개의 설정을 모두 활성화한 것과 같습니다. 다음 설정 각각은 서로 독립적입니다.

#### ☐ 새 ACL(엑세스 제어 목록)을 통해 부여된 버킷 및 객체에 대한 퍼블릭 액세스 차단

S3은 새로 추가된 버킷 또는 객체에 적용되는 퍼블릭 액세스 권한을 차단하며, 기존 버킷 및 객체에 대한 새 퍼블릭 액세스 ACL 생성을 금지합니다. 이 설정은 ACL을 사용하여 S3 리소스에 대한 퍼블릭 액세스를 허용하는 기존 권한을 변경하지 않습니다.

#### ☒ 임의의 ACL(엑세스 제어 목록)을 통해 부여된 버킷 및 객체에 대한 퍼블릭 액세스 차단

S3은 버킷 및 객체에 대한 퍼블릭 액세스를 부여하는 모든 ACL을 무시합니다.

#### ☒ 새 퍼블릭 버킷 또는 액세스 지점 정책을 통해 부여된 버킷 및 객체에 대한 퍼블릭 액세스 차단

S3은 버킷 및 객체에 대한 퍼블릭 액세스를 부여하는 새 버킷 및 액세스 지점 정책을 차단합니다. 이 설정은 S3 리소스에 대한 퍼블릭 액세스를 허용하는 기존 정책을 변경하지 않습니다.

#### ☒ 임의의 퍼블릭 버킷 또는 액세스 지점 정책을 통해 부여된 버킷 및 객체에 대한 퍼블릭 및 교차 계정 액세스 차단

S3은 버킷 및 객체에 대한 퍼블릭 액세스를 부여하는 정책을 사용하는 버킷 또는 액세스 지점에 대한 퍼블릭 및 교차 계정 액세스를 무시합니다.

취소

변경 사항 저장

버킷 만들기 버튼 클릭

Amazon S3

버킷

엑세스 지점

객체 Lambda 액세스 지점

다중 리전 액세스 지점

배치 작업

S3용 액세스 분석기

이 계정의 퍼블릭 액세스 차단 설정

Storage Lens

대시보드

AWS Organizations 설정

Amazon S3

계정 스냅샷

Storage Lens 대시보드 보기

총 스토리지

객체 수

평균 객체 크기

어드밴스드 메트릭은 "default-account-dashboard" 구성에서 활성화할 수 있습니다.

1.2GB

67

18.8MB

버킷 (2) Info

버킷은 S3에 저장되는 데이터의 컨테이너입니다. [자세히 알아보기](#)

이름으로 버킷 찾기

이름

AWS 리전

엑세스

생성 날짜

버킷 만들기



Amazon S3

버킷

엑세스 지점

객체 Lambda 액세스 지점

다중 리전 액세스 지점

배치 작업

S3용 액세스 분석기

이 계정의 퍼블릭 액세스 차단 설정

Storage Lens

대시보드

AWS Organizations 설정

기능 스포트라이트

3

S3용 AWS Marketplace

버킷 만들기

Info

버킷은 S3에 저장되는 데이터의 컨테이너입니다. 자세히 알아보기

일반 구성

버킷 이름

fileUploadBucket

버킷 이름은 고유해야 하며 공백 또는 대문자를 포함할 수 없습니다. 버킷 이름 지정 규칙 참조

AWS 리전

아시아 태평양(서울) ap-northeast-2

기존 버킷에서 설정 복사 - 선택 사항

다음 구성의 버킷 설정만 복사됩니다.

버킷 선택

이 버킷의 퍼블릭 액세스 차단 설정

퍼블릭 액세스는 ACL(엑세스 제어 목록), 버킷 정책, 액세스 지점 정책 또는 모두를 통해 버킷 및 객체에 부여됩니다. 이 버킷 및 해당 객체에 대한 퍼블릭 액세스는 차단되었는지 확인하려면 모든 퍼블릭 액세스 차단을 활성화합니다. 이 설정은 이 버킷 및 해당 액세스 지점에만 적용됩니다. AWS에서는 모든 퍼블릭 액세스 차단을 활성화하도록 권장하지만, 이 설정을 적용하기 전에 퍼블릭 액세스가 없어도 애플리케이션이 올바르게 작동하는지 확인합니다. 이 버킷 또는 내부 객체에 대한 어느 정도 수준의 퍼블릭 액세스가 필요한 경우 특정 스토리지 사용 사례에 맞게 여러 개별 설정을 사용자 지정할 수 있습니다. 자세히 알아보기

☐ 모든 퍼블릭 액세스 차단
 이 설정을 활성화하면 아래 4개의 설정을 모두 활성화한 것과 같습니다. 다음 설정 각각은 서로 독립적입니다.

☒ 새 ACL(엑세스 제어 목록)을 통해 부여된 버킷 및 객체에 대한 퍼블릭 액세스 차단
 S3은 새로 추가된 버킷 또는 객체에 적용되는 퍼블릭 액세스 권한을 차단하며, 기존 버킷 및 객체에 대한 새 퍼블릭 액세스 ACL 생성을 금지합니다. 이 설정은 ACL을 사용하여 S3 리소스에 대한 퍼블릭 액세스를 허용하는 기존 권한을 변경하지 않습니다.

☐ 임의의 ACL(엑세스 제어 목록)을 통해 부여된 버킷 및 객체에 대한 퍼블릭 액세스 차단
 S3은 버킷 및 객체에 대한 퍼블릭 액세스를 부여하는 모든 ACL을 무시합니다.

☐ 새 퍼블릭 버킷 또는 액세스 지점 정책을 통해 부여된 버킷 및 객체에 대한 퍼블릭 액세스 차단
 S3은 버킷 및 객체에 대한 퍼블릭 액세스를 부여하는 새 버킷 및 액세스 지점 정책을 차단합니다. 이 설정은 S3 리소스에 대한 퍼블릭 액세스를 허용하는 기존 정책을 변경하지 않습니다.

☐ 임의의 퍼블릭 버킷 또는 액세스 지점 정책을 통해 부여된 버킷 및 객체에 대한 퍼블릭 및 교차 계정 액세스 차단
 S3은 버킷 및 객체에 대한 퍼블릭 액세스를 부여하는 정책을 사용하는 버킷 또는 액세스 지점에 대한 퍼블릭 및 교차 계정 액세스를 무시합니다.

모든 퍼블릭 액세스 차단을 비활성화하면 이 버킷과 그 안에 포함된 객체가 퍼블릭 상태가 될 수 있습니다.

정적 웹 사이트 호스팅과 같은 구체적으로 확인된 사용 사례에서 퍼블릭 액세스가 필요한 경우가 아니면 모든 퍼블릭 액세스 차단을 활성화하는 것이 좋습니다.

☒ 현재 설정으로 인해 이 버킷과 그 안에 포함된 객체가 퍼블릭 상태가 될 수 있음을 알고 있습니다.

생성된 버킷에 접근하여 권한을 클릭 → 버킷 정책의 편집 버튼 클릭

정책 생성기 클릭하여 정책 생성

포팅매뉴얼 v.1

33



## AWS Policy Generator

The AWS Policy Generator is a tool that enables you to create policies that control access to Amazon Web Services (AWS) products and resources. For more information about creating policies, see [key concepts in Using AWS Identity and Access Management](#). Here are [sample policies](#).

### Step 1: Select Policy Type

A Policy is a container for permissions. The different types of policies you can create are an IAM Policy, an S3 Bucket Policy, an SNS Topic Policy, a VPC Endpoint Policy, and an SQS Queue Policy.

Select Type of Policy S3 Bucket Policy

### Step 2: Add Statement(s)

A statement is the formal description of a single permission. See [a description of elements](#) that you can use in statements.

Effect ☒ Allow ☐ Deny

Principal   
Use a comma to separate multiple values.

AWS Service Amazon S3 ☐ All Services ('\*')

Use multiple statements to add permissions for more than one service.

Actions 3 Action(s) Selected ☐ All Actions ('\*')

Amazon Resource Name (ARN) 앞서 복사한 ARN  
ARN should follow the following format: arn:aws:s3:::{BucketName}/{Key Name}.  
Use a comma to separate multiple values.

Add Conditions (Optional)

Add Statement

You added the following statements. Click the button below to Generate a policy.

Principal(s)	Effect	Action	Resource	Conditions
* *	Allow	<ul style="list-style-type: none"><li>s3:DeleteObject</li><li>s3:GetObject</li><li>s3:PutObject</li></ul>		None

### Step 3: Generate Policy

A policy is a document (written in the [Access Policy Language](#)) that acts as a container for one or more statements.

Generate Policy [Start Over](#)

아래와 같은 형태의 json파일 생성, 변경 사항 저장 버튼 클릭

```
{
  "Version": "2012-10-17",
  "Id": "Policy1663652945650",
  "Statement": [
    {
      "Sid": "Stmnt1663652943856",
      "Effect": "Allow",
      "Principal": "*",
      "Action": [
        "s3:DeleteObject",
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3:::masterpiecebucket/*"
    }
  ]
}
```

Amazon S3

버킷

액세스 지점

객체 Lambda 액세스 지점

다중 리전 액세스 지점

배치 작업

S3용 액세스 분석기

이 계정의 퍼블릭 액세스 차단 설정

Storage Lens

대시보드

AWS Organizations 설정

기능 스포트라이트 3

S3용 AWS Marketplace

Amazon S3 > earth-s3-upload-img > 객체 소유권 편집

객체 소유권 편집 Info

객체 소유권

다른 AWS 계정에서 이 버킷에 작성하고 ACL(액세스 제어 목록)을 사용하여 부여된 객체의 소유권을 제어합니다. 객체 소유권은 객체에 대한 액세스를 지정할 수 있는 사용자를 결정합니다.

☐ ACL 비활성화됨(권장)  
이 버킷의 모든 객체는 이 계정이 소유합니다. 이 버킷과 그 객체에 대한 액세스는 정책을 통해서만 지정됩니다.

☒ ACL 활성화됨  
이 버킷의 객체는 다른 AWS 계정에서 소유할 수 있습니다. 이 버킷 및 객체에 대한 액세스는 ACL을 사용하여 지정할 수 있습니다.

**ACL을 활성화하면 버킷 소유자가 객체 소유권에 대해 적용한 설정이 비활성화됩니다.**  
 버킷 소유자 적용 설정이 해제되면 ACL(액세스 제어 목록) 및 연결된 권한이 복원됩니다. 소유하지 않은 객체에 대한 액세스는 버킷 정책이 아닌 ACL을 기반으로 합니다.

☒ ACL이 복원된다는 것을 확인합니다.

객체 소유권

☐ 버킷 소유자 선호  
이 버킷에 작성된 새 객체가 bucket-owner-full-control 삽입 ACL을 지정하는 경우 새 객체는 버킷 소유자가 소유합니다. 그렇지 않은 경우 객체 라이터가 소유합니다.

☒ 객체 라이터  
객체 라이터는 객체 소유자로 유지됩니다.

취소

변경 사항 저장

## build.gradle에 의존성 추가

```
implementation 'org.springframework.cloud:spring-cloud-starter-aws:2.2.6.RELEASE'
```

## S3 config 파일 추가

```
@Configuration
@EnableConfigurationProperties
public class AmazonS3Config {
    @Value("${cloud.aws.credentials.access-key}")
    private String accessKey;

    @Value("${cloud.aws.credentials.secret-key}")
    private String secretKey;

    @Value("${cloud.aws.region.static}")
    private String region;

    @Bean
    public AmazonS3Client amazonS3Client(){
        BasicAWSCredentials awsCreds = new BasicAWSCredentials(accessKey,secretKey);
        return (AmazonS3Client) AmazonS3ClientBuilder.standard().withRegion(region)
            .withCredentials(new AWSStaticCredentialsProvider(awsCreds))
            .build();
    }
}
```