



포팅 매뉴얼

1. 프로젝트 개발 환경

1) FRONT END

3) IDE

2) BACK END

4) Server

2. 설정 파일 목록

1) Front End

2) Back End

3. EC2 설정 시나리오

1) 사용 포트

2) 방화벽 설정

3) Docker 설치

4) Jenkins 설치 및 설정

5) Nginx 설치

6) SSL 인증서 발급

7) Nginx 설정

8) MySQL 설치

9) Redis 컨테이너 생성

3. 배포

1) Dockerfile 및 Nginx 설정

2) Jenkins & GitLab 연동

(1) jenkins에서 프로젝트 생성

(2) 소스 코드 관리

(3) 빌드 유발

(4) Webhook 설정

(5) Execute Shell

4. 외부 서비스

1) Kakao

2) Youtube API

3) Google Cloud Storage

1. 프로젝트 개발 환경

1) FRONT END

react	18.2.0
node.js	18.15.0 LTS
vite	4.1.0
typescript	4.9.3
three.js	0.150.1
react-three-fiber	8.12.0
blender	3.4.1
react query	3.39.3
react-router-dom	6.8.2
tailwindcss	3.2.7
recoil	0.7.7

3) IDE

Visual Studio Code	1.77.1
IntelliJ	IDEA 2022.3.2

2) BACK END

java	11.0.18
springboot	2.7.9
gradle	Openjdk 11.0.18+10
swagger	org.springdoc:springdoc-openapi-starter-webmvc-ui:2.0.0
MySQL	8.0.30
Redis	7.0.10
Python	3.10.9
Django	3.2.13

4) Server

Nginx	1.18.0
Jenkins	1.18.0
Docker	23.0.1

2. 설정 파일 목록

1) Front End

- .env

◦ 카카오톡 REDIRECT URI 설정

◦ Youtube API KEY 설정

```
VITE_APP_IMAGE_ROOT = 구글 스토리지 링크
VITE_REST_API_KEY = REST API 요청 KEY
VITE_REDIRECT_URI = 카카오 리다이렉트 URI
VITE_LOGOUT_URI = 카카오 로그아웃 URI
VITE_ADMIN_KEY = 카카오 로그아웃 ADMIN KEY
VITE_YOUTUBE_API_KEY = YOUTUBE API KEY
```

- package.json

◦ build port 번호 설정

```
"scripts": {
  "dev": "vite --port 3000",
  "build": "vite build",
  "preview": "vite preview"
},
```

- 배포 설정 파일
 - Docker File
 - nginx.conf

2) Back End

- Spring Boot 설정 파일
 - WebConfig

```
@Override
public void addCorsMappings(CorsRegistry registry) {
    registry.addMapping("/**")
        .allowedHeaders("*")
        .allowedOrigins("http://localhost:5173", "http://localhost:8080", "http://localhost:3000"
            , "https://j8a705.p.ssafy.io", "https://j8a705.p.ssafy.io:80", "https://j8a705.p.ssafy.io:8080", "https://j8a705.p.ssafy.io:5173", "https://j8a705.p.ssafy.io:3000", "http://j8a705.p.ssafy.io")
        .allowedMethods("OPTIONS", "GET", "POST", "PUT", "DELETE", "PATCH");
}
```

- application.properties

```
#port
server.port=8080

#base url
server.servlet.contextPath=/api
server.servlet.encoding.charset=UTF-8
server.servlet.encoding.enabled=true
server.servlet.encoding.force=true

# MySQL Driver
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://서버주소:3306/omz?serverTimezone=UTC&useUnicode=yes&characterEncoding=UTF-8&allowMultiQueries=true

# DB username
spring.datasource.username=omz

# DB password
spring.datasource.password=omz7581
```

- 배포 설정 파일
 - Docker File
 - build.gradle

3. EC2 설정 시나리오

1) 사용 포트

구분	포트 번호
Front-end	3000
Back-end	8080
Sub-Back-end	9090
MySQL	3306
Redis	6379
Django	8000

2) 방화벽 설정

현재 방화벽 status 확인 및 설정

```
$ sudo ufw status
$ sudo ufw allow ssh
```

3) Docker 설치

Ubuntu에 도커 설치

```
$ sudo apt-get update

# 필수 패키지 설치
$ sudo apt-get install \
    apt-transport-https \
    ca-certificates \
    curl \
    gnupg \
    lsb-release

# GPG Key 인증
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

# docker repository 등록
$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable"

# 도커 설치
$ sudo apt-get update
$ sudo apt-get install docker-ce docker-ce-cli containerd.io
```

```
# 도커 확인
$ sudo service docker status
```

4) Jenkins 설치 및 설정

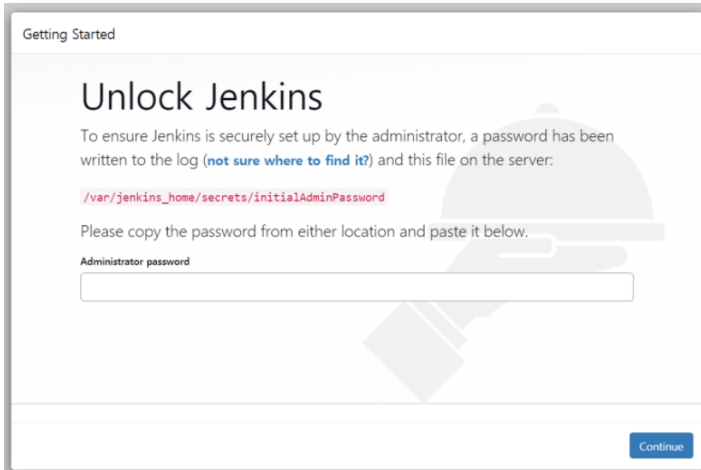
1. jenkins 이미지 설치 및 실행

```
# 설치 및 실행
$ sudo docker run -u 0 -d -p 9090:8080 -p 50000:50000 -v /var/jenkins:/var/jenkins_home -v /var/run/docker.sock:/var/run/docker.sock --name jenkins jenkins/jenkins:lts-jdk11

# 재시작
$ sudo docker restart jenkins

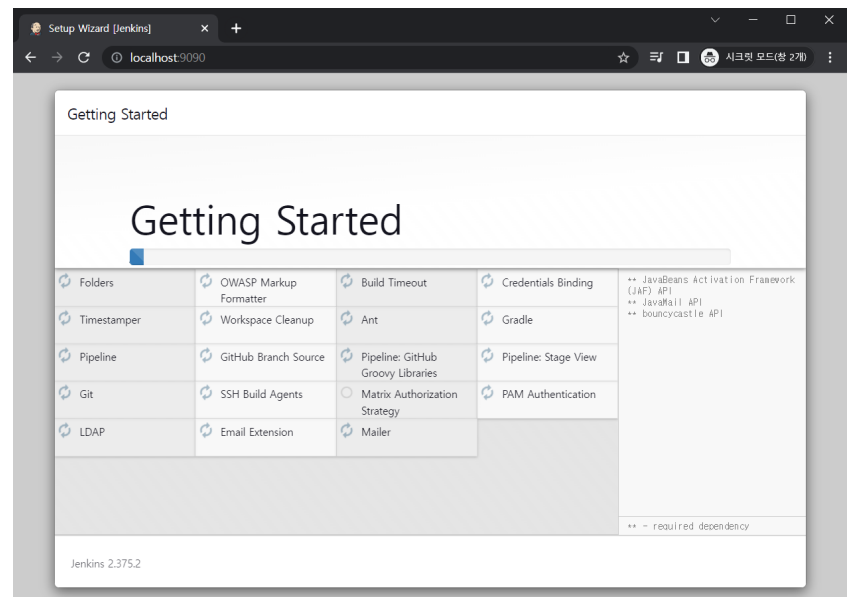
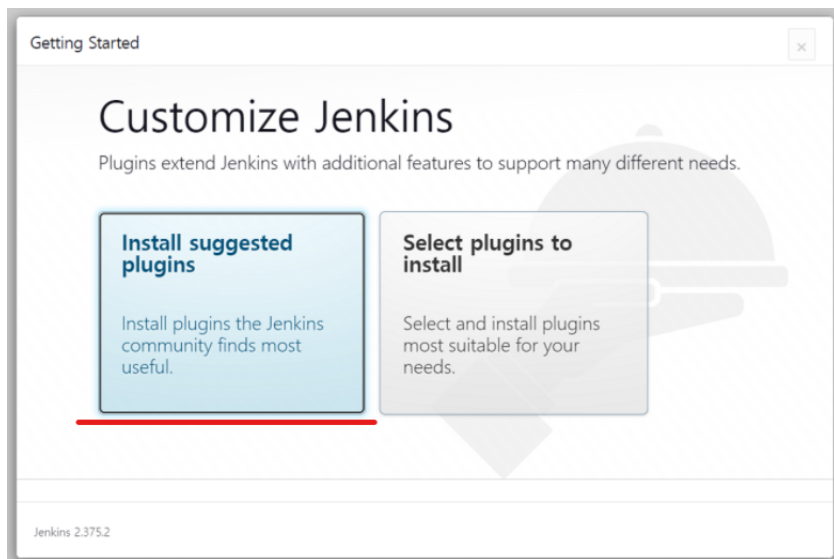
# 이미지 설치 확인
$ sudo docker images
```

2. jenkins 접속 (서버 주소)



```
# 비밀번호 확인 방법
$ sudo docker logs jenkins
```

3. 기본 설치



4. jenkins 플러그인 설치

- **Dashboard → Jenkins 관리 → 플러그인 관리 → Available plugins**
- **Gitlab 관련 항목 설치**
 - Gitlab, Generic Webhook Trigger, Gitlab API, Gitlab Authentication 설치
- **Docker 관련 항목 설치**
 - Docker, Docker Commons, Docker Pipeline, Docker API 설치
- **백엔드에서 Gradle을 사용하였다면 Gradle Plugin도 설치**

5. Gradle 사용하는 경우

- **Jenkins 관리 → Global Tool Configuration → Gradle 버전 설정 후 추가**

Gradle

Gradle installations

List of Gradle installations on this system

Add Gradle

Gradle

name ?

vieweongeeGradle

Install automatically ?

Install from Gradle.org

Version

Gradle 7.6

Add Installer

Add Gradle

Save

Apply

6. 젠킨스 컨테이너 안에 도커 설치

```
# 젠킨스 컨테이너 실행
$ docker exec -it jenkins bash

# 도커 설치
$ apt-get update

$ apt-get install \
  apt-transport-https \
  ca-certificates \
  curl \
  gnupg \
  lsb-release

$ apt-get install docker.io -y

# 도커 확인
$ sudo service docker status
```

5) Nginx 설치

```
# 설치
$ sudo apt-get install nginx

# 설치 확인 및 버전 확인
$ nginx -v

# Nginx 설정은 SSL 인증서 발급 후에
```

6) SSL 인증서 발급

```
$ sudo apt-get install letsencrypt

# 만약 nginx를 사용 중이라면 중지
$ sudo systemctl stop nginx

# 인증서 발급
sudo letsencrypt certonly --standalone -d j8a705.p.ssafy.io

# 이메일 작성 후 Agree
```

```
j8a705@ip-172-26-13-90:~$ sudo letsencrypt certonly --standalone -d j8a705.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 j8a705.p.ssafy.io
Waiting for verification...
Cleaning up challenges

IMPORTANT NOTES:
- Congratulations! Your certificate and chain have been saved at:
  /etc/letsencrypt/live/j8a705.p.ssafy.io/fullchain.pem
  Your key file has been saved at:
  /etc/letsencrypt/live/j8a705.p.ssafy.io/privkey.pem
  Your cert will expire on 2023-06-14. 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
```

7) Nginx 설정

```
$ sudo vi /etc/nginx/conf.d/default.conf
$ sudo vi /etc/nginx/sites-available/default
```

```
# 443 포트로 접근시 ssl을 적용한 뒤 3000, 8080 포트로 요청을 전달
server {
    # 특정 경로에 대하여 적절한 WAS로 요청 전달
    location / {
        proxy_hide_header Access-Control-Allow-Origin;
        add_header 'Access-Control-Allow-Origin' '*';
        add_header 'Access-Control-Allow-Methods' 'GET, POST, OPTIONS';
        add_header 'Access-Control-Allow-Headers' 'DNT,User-Agent,X-Requested-With, If-Modified-Since,Cache-Control,Content-Type,Range';
        add_header 'Access-Control-Expose-Headers' 'Content-Length,Content-Range';
        proxy_connect_timeout      90;
        proxy_send_timeout          90;
        proxy_read_timeout          90;
```

포팅 매뉴얼

4

```

    proxy_pass http://localhost:3000;
}

location /api {
    proxy_hide_header Access-Control-Allow-Origin;
    add_header 'Access-Control-Allow-Origin' '*';
    add_header 'Access-Control-Allow-Methods' 'GET, POST, OPTIONS';
    add_header 'Access-Control-Allow-Headers' 'DNT,User-Agent,X-Requested-With,If-Modified-Since,Cache-Control,Content-Type,Range';
    add_header 'Access-Control-Expose-Headers' 'Content-Length,Content-Range';

    proxy_connect_timeout      90;
    proxy_send_timeout          90;
    proxy_read_timeout          90;
    proxy_pass http://localhost:8080/api;
}

location /django {
    proxy_hide_header Access-Control-Allow-Origin;
    add_header 'Access-Control-Allow-Origin' '*';
    add_header 'Access-Control-Allow-Methods' 'GET, POST, OPTIONS';
    add_header 'Access-Control-Allow-Headers' 'DNT,User-Agent,X-Requested-With,If-Modified-Since,Cache-Control,Content-Type,Range';
    add_header 'Access-Control-Expose-Headers' 'Content-Length,Content-Range';

    proxy_connect_timeout      90;
    proxy_send_timeout          90;
    proxy_read_timeout          90;
    proxy_pass http://localhost:8080/django;
}

# https 적용
listen 443 ssl;

# 앞서 준비한 도메인과 연결된 SSL 인증서의 chain key, private key를
# nginx가 설치된 서버의 특정 폴더에 저장하고,
# 그 키가 위치한 폴더의 경로를 작성
# 443으로 들어오는 요청을 SSL 인증서로 암호화 하겠다는 의미
ssl_certificate /etc/letsencrypt/live/j8a705.p.ssafy.io/fullchain.pem;
ssl_certificate_key /etc/letsencrypt/live/j8a705.p.ssafy.io/privkey.pem;
}

# 80 포트로 접근시 443 포트로 리다이렉트 시켜주는 설정
server {
    server_name j8a705.p.ssafy.io;

    # http 기본 포트
    listen 80;

    return 301 https://$server_name$request_uri;
}

```

```

# nginx 실행
$ sudo systemctl start nginx

# 실행 확인
$ sudo systemctl status nginx

```

8) MySQL 설치

(1) Ubuntu에 MySQL 설치

```

# MySQL을 설치
sudo apt-get update
sudo apt-get install mysql-server

# MySQL 구동
sudo systemctl start mysql.service

# MySQL 접속
$ sudo mysql

```

```

# 계정 생성
mysql> CREATE USER '계정이름'@'%' IDENTIFIED BY '비밀번호';

# GRANT로 권한 부여 - 어떠한 ip에서든 해당 계정에 모든 권한 부여
mysql> GRANT ALL PRIVILEGES ON . TO '계정이름'@'%' WITH GRANT OPTION;
mysql> FLUSH PRIVILEGES;

# 현재 mysql에서 기본으로 세팅 되어있는 유저들과 추가된 유저를 확인
mysql > SELECT user,authentication_string,plugin,host FROM mysql.user;

# database 생성- utf8 확장 버전
mysql > CREATE DATABASE '데이터베이스명' CHARACTER SET utf8mb4 collate utf8mb4_general_ci;

# 해당 계정이 database의 모든 테이블에 모든 권한 행사
mysql > GRANT ALL PRIVILEGES ON '데이터베이스명'.* TO '계정이름'@'%' ;

```

(2) MySQL Workbench 사용법

1. MySQL Workbench 설치

2. Connection 설정

- MySQL Connection → '+' 버튼
- Server에 있는 MySQL과 연결
 - Connection Name: 원하는 이름
 - Hostname: 접속할 서버 주소
 - Username: 생성한 MySQL 계정의 username

9) Redis 컨테이너 생성

```

# 도커 이미지 다운로드
$ docker pull redis

```

```
# redis 컨테이너 실행
$ docker run -d -p 6379:6379 redis
```

3. 배포

1) Dockerfile 및 Nginx 설정

(1) Frontend

- Dockerfile

```
# node.js로 빌드 ( base로 사용할 Image name )
FROM node:18.15.0 as builder

# 작업 폴더를 만들고 npm 설치
RUN mkdir /usr/src/app
WORKDIR /usr/src/app

ENV PATH /usr/src/app/node_modules/.bin:$PATH
COPY package.json /usr/src/app/package.json
RUN npm install --force

COPY . /usr/src/app
RUN npm run build
FROM nginx:latest

RUN rm /etc/nginx/conf.d/default.conf
COPY nginx/nginx.conf /etc/nginx/conf.d/default.conf

COPY --from=builder /usr/src/app/dist /usr/share/nginx/html

# 80포트 오픈하고 nginx 실행
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

- nginx.conf

```
server {
    # 포트 번호
    listen 3000;

    # 경로 설정
    location / {
        # index 파일이 있는 경로
        root    /usr/share/nginx/html;

        # index 파일로 지정할 파일 설정
        index   index.html index.htm;

        # 요청한 주소의 uri를 무시하고 index.html 파일을 제공
        try_files $uri $uri/ /index.html;
    }
}
```

(3) Backend

- Dockerfile

```
# openjdk11로 실행
FROM openjdk:11-jdk

# 해당 경로의 모든 jar파일을 변수로 담기
ARG JAR_FILE=build/libs/*.jar

# 빌드된 jar파일을 api.jar파일 이라는 이름으로 생성
COPY ${JAR_FILE} app.jar

# 컨테이너가 리스닝할 포트
EXPOSE 8080

# 환경 변수 설정
ENV TZ=Asia/Seoul

# 컨테이너를 실행할 때 실행할 커맨드
ENTRYPOINT ["java", "-jar", "app.jar"]
```

- build.gradle

```
bootJar{
    bootJar.enabled=true
}

jar {
    enabled = false
}
```

(4) Django

```
# 도커에 설치될 python 버전
FROM python:3.9

RUN pip install --upgrade pip
# 필요한 모듈 설치
RUN pip3 install django
RUN pip install django-cors-headers
RUN pip uninstall mysql-connector
RUN pip install mysql-connector-python

# 컨테이너 내 프로젝트 root directory 설정
# /usr/src/app 디렉토리 생성
```

```
WORKDIR /usr/src/app

# 현재 폴더 내용을 복사
COPY . .

#manage.py를 실행할 수 있는 디렉토리로 이동
WORKDIR .

RUN pip install -r requirements.txt
# django를 가동시켜주는 코드
CMD ["python3", "manage.py", "runserver", "0.0.0.0:8000"]

EXPOSE 8000
```

2) Jenkins & GitLab 연동

(1) jenkins에서 프로젝트 생성

Dashboard → 새로운 Item → 프로젝트 이름 입력 → Freestyle project

(2) 소스 코드 관리

소스 코드 관리 > Git 선택 > git clone 주소 입력

- Credentials 아래의 Add > 깃 아이디와 비밀번호 저장
- 저장한 credential을 클릭했을 때 에러메세지가 뜨지 않으면 정상 접근 연동 성공

소스 코드 관리

☐ None

☒ Git ?

Repositories ?

Repository URL ?

Credentials ?

+ Add

고급...

Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ?

*/main

(3) 빌드 유발

- Build when a change is pushed to GitLab 체크

(4) Webhook 설정

1. Jenkins project 선택 > 구성 > 빌드 유발 > 저장

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:

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

☒ Enable [ci-skip]

☒ Ignore WIP Merge Requests

Labels that forces builds if they are added (comma-separated)

☒ Set build description to build cause (eg. Merge request or Git Push)

☐ Build on successful pipeline events

Pending build name for pipeline ?

☐ Cancel pending merge request builds on update

Allowed branches

☒ Allow all branches to trigger this job ?

☐ Filter branches by name ?

☐ Filter branches by regex ?

☐ Filter merge request by label

Secret token ?

Generate

Clear

2. GitLab Settings > Webhooks

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

main

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.

SSL verification

☒ Enable SSL verification

Save changes

Test

Push events

Tag push events

Issues events

Confidential issues events

Note events

Confidential note events

Merge requests events

Job events

Pipeline events

	Elapsed time	Request time	
k	0.04 sec	1 hour ago	View details
k	0.02 sec	1 hour ago	View details
k	0.02 sec	3 hours ago	View details
k	0.01 sec	3 hours ago	View details

Delete

3. GitLab Webhooks의 URL과 Secret token에 Jenkins의 URL과 Secret token 작성
4. main 브랜치에 push할 때마다 자동 빌드 되도록 설정

(5) Execute Shell

```
# [Django]
docker build -t omz_django:latest ./Development/BE/django
if (docker ps | grep omz_django) then docker stop omz_django; fi
docker run -d --rm --name omz_django -p 8000:8000 omz_django

# [frontend]
echo '-----[front-end]-----'
# 컨테이너 생성
docker build -t omz_frontend:latest ./Development/FE/om
```



```
# 이미 실행 중인 컨테이너가 있다면 중단
if (docker ps | grep omz_frontend) then docker stop omz_frontend; fi

# 컨테이너 실행
docker run -d --rm --name omz_frontend -p 3000:3000 omz_frontend

# [backend]
echo '-----[back-end]-----'
# 권한 부여 및 빌드 시작
cd Development/BE/omz
chmod +x gradlew
./gradlew build

# 컨테이너 생성
docker build -t omz_backend:latest .

# 이미 실행 중인 backend 컨테이너가 있다면 중단하기
if (docker ps | grep omz_backend) then docker stop omz_backend; fi

# 컨테이너 실행
docker run -d --rm --name omz_backend -p 8080:8080 omz_backend
```

4. 외부 서비스

1) Kakao

- 기본 정보

기본 정보

앱 ID	877015
앱 이름	OMZ
사업자명	OMZ

- redircet uri

```
http://j8a705.p.ssafy.io/redirect
http://j8a705.p.ssafy.io/logout
http://j8a705.p.ssafy.io:3000/redirect
http://j8a705.p.ssafy.io:3000/logout
```

- 수집 정보


- 닉네임, 프로필사진, 카카오톡계정 (이메일)

개인정보

항목 이름	ID	상태
닉네임	profile_nickname	● 필수 동의 설정
프로필 사진	profile_image	● 필수 동의 설정
카카오톡계정(이메일)	account_email	● 필수 동의 [수집] 설정

2) Youtube API

- 기본 정보



YouTube Data API v3

The YouTube Data API v3 is an API that provides access to YouTube data, such as videos, playlists, and channels.


제공: Google


서비스 이름	유형	상태
youtube.googleapis.com	공개 API	사용 설정됨

- 키 발급

이름 *
API 키 1개

키 제한사항

 이 키는 제한되지 않습니다. 무단 사용을 방지하려면 API를 사용할 수 있는 위치와 대상을 제한하는 것이 좋습니다. [자세히 알아보기](#)

API Key


key=API_KEY 매개변수로 키를 전달하여 애플리케이션에서 이 키를 사용하세요.

생성일	2023년 4월 6일 AM 10시 48분 56 초 GMT+9
-----	---

3) Google Cloud Storage

- 키 발급
 - google cloud platform console 접속 후 project 생성
 - Storage > browser에서 버킷 생성
- SpringBoot 설정
 - build.gradle

```
implementation group: 'com.google.cloud', name: 'spring-cloud-gcp-starter', version: '3.4.3'
implementation group: 'com.google.cloud', name: 'spring-cloud-gcp-storage', version: '3.4.3'
```

- application.properties

```
# Google cloud key
spring.cloud.gcp.storage.credentials.location=classpath:omz-project-13bcf2dbae0.json
```