

# 집에서 운동중

PORTING MANUAL

# 목 차

## I. 빌드 및 배포

1. 개발 환경 .....	2
2. 설정 파일 목록 .....	2
3. 설정 파일 및 환경 변수 정보 .....	3
4. Docker 설치 .....	9
5. SSL 인증서 발급 .....	10
6. OpenVidu 배포 .....	10
7. DB 및 Infra 배포 .....	13
8. Backend CI/CD .....	17
9. Frontend CI/CD .....	20

## II. 외부 서비스

1. 소셜 로그인 .....	24
2. Teachable Machine .....	30

# I. 빌드 및 배포

## 1. 개발 환경

Server : AWS EC2 Ubuntu 20.04 LTS

Visual Studio Code : 1.70.1

IntelliJ IDEA : 2022.1.3(Ultimate Edition) 11.0.15 + 10-b2043.56 amd64

JVM : 11.0.16+8-post-Ubuntu-Oubuntu120.04

Docker : 20.10.17

Node.js : 18.7.0

MySQL : 8.0.30-1.el8

Redis : 7.0.4

Nginx : 1.23.1

Jenkins : 2.346.2

Openvidu : 2.22.0

Teachable Machine : 2.0

## 2. 설정 파일 목록 (경로와 파일은 4~9번 배포 과정 중 생성 될 수 있습니다.)

### React

- .env : /jenkins/workspace/frontend/frontend
- Dockerfile : /jenkins/workspace/frontend/frontend

### Spring

- application.properties : /jenkins/workspace/backend/backend/src/main/resources
- application-oauth.properties :  
/jenkins/workspace/backend/backend/src/main/resources
- application.properties : /jenkins/workspace/backend/backend/src/test/resources
- Dockerfile : /jenkins/workspace/backend/backend
- deploy.sh : /jenkins/workspace/backend/backend

## Docker

- `docker-compose.yml` : `/home/ubuntu`

## Nginx

- `app.conf` : `/home/ubuntu/nginx/conf.d`

## 3. 설정 파일 및 환경 변수 정보

### React

- `.env`

`WDS_SOCKET_PORT=0`

`REACT_APP_BASE_URL= REST API (BACKEND) 요청 URL`

`REACT_APP_GOOGLE_CLIENT_ID= 구글 클라이언트 ID`

`REACT_APP_GOOGLE_CLIENT_SECRET= 구글 클라이언트 SECRET`

`REACT_APP_GOOGLE_REDIRECT_URI= 구글 리다이렉트 URI`

`REACT_APP_NAVER_CLIENT_ID= 네이버 클라이언트 ID`

`REACT_APP_NAVER_CLIENT_SECRET= 네이버 클라이언트 SECRET`

`REACT_APP_NAVER_REDIRECT_URI= 네이버 리다이렉트 URI`

`REACT_APP_KAKAO_CLIENT_ID= 카카오 클라이언트 ID`

`REACT_APP_KAKAO_CLIENT_SECRET= 카카오 클라이언트 SECRET`

`REACT_APP_NAVER_REDIRECT_URI= 네이버 리다이렉트 URI`

- `Dockerfile`

`FROM node:alpine`

`WORKDIR /usr/src/app`

`COPY ./package* /usr/src/app/`

`RUN npm install`

`COPY ./ /usr/src/app/`

`CMD ["npm", "run", "start"]`

### Spring

- `application.properties(main)`

```
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://{DB컨테이너이름}:3306/{SCHEME이름}
?useSSL=false&useUnicode=true&characterEncoding=utf8&allowPublicKeyRetrieval=true&serverTimezone=Asia/Seoul&useLegacyDatetimeCode=false
spring.datasource.username=root
spring.datasource.password=ROOT계정 PASSWORD
```

```
spring.jpa.show-sql=true
```

```
spring.jpa.generate-ddl=true
spring.jpa.hibernate.ddl-auto=update
```

```
spring.jpa.properties.hibernate.format_sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect
```

```
spring.profiles.include=oauth
```

```
logging.level.root = info
```

```
spring.redis.host = REDIS컨테이너명
spring.redis.port = 6379
spring.redis.password = REDIS암호
```

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

#### - application-oauth.properties

#Google

```
spring.security.oauth2.client.registration.google.client-id= 구글
클라이언트 ID
spring.security.oauth2.client.registration.google.client-secret=
구글 클라이언트 SECRET
spring.security.oauth2.client.registration.google.redirect-uri= 구글
리다이렉트 URI
spring.security.oauth2.client.registration.google.scope= profile,
email
```

#Naver

```
spring.security.oauth2.client.registration.naver.client-id= 네이버
클라이언트 ID
spring.security.oauth2.client.registration.naver.client-secret=
```

```

네이버 클라이언트 SECRET
spring.security.oauth2.client.registration.naver.redirect-uri=
네이버 리다이렉트 URI
spring.security.oauth2.client.registration.naver.authorization-
grant-type=authorization_code
spring.security.oauth2.client.registration.naver.scope=name,email
spring.security.oauth2.client.registration.naver.client-name=Naver
#Provider-Naver
spring.security.oauth2.client.provider.naver.authorization-
uri=https://nid.naver.com/oauth2.0/authorize
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.user-name-
attribute=response

#Kakao
spring.security.oauth2.client.registration.kakao.client-id= 카카오
클라이언트 ID
spring.security.oauth2.client.registration.kakao.client-secret=
카카오 클라이언트 SECRET
spring.security.oauth2.client.registration.kakao.redirect-uri=카카오
리다이렉트 URI
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, account_email
spring.security.oauth2.client.registration.kakao.client-
name=Kakao
#Provider-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

```

- application.properties(test)

```
# Datasource
spring.datasource.driver-class-name=org.h2.Driver
spring.datasource.url=jdbc:h2:mem:db;DB_CLOSE_DELAY=-1
spring.datasource.username=h2test
spring.datasource.password=h2test

# JPA
spring.jpa.generate-ddl=true
spring.jpa.hibernate.ddl-auto=create-drop
spring.jpa.open-in-view=false
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.H2Dialect

#Google
spring.security.oauth2.client.registration.google.client-id=test
spring.security.oauth2.client.registration.google.client-secret=test
spring.security.oauth2.client.registration.google.scope=profile,
email

spring.redis.host = REDIS컨테이너명
spring.redis.port = 6379
spring.redis.password = REDIS PASSWORD
```

#### - Dockerfile

```
FROM openjdk:11-jdk
ARG JAR_FILE=build/libs/*.jar
COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java", "-jar", "-Duser.timezone=Asia/Seoul", "/app.jar"]
```

#### - deploy.sh

```
echo '실행 시작'
echo 'git pull'

echo 'jar 파일 삭제'
rm build/libs/*.jar

echo '빌드 시작'
./gradlew build
```

```
echo '도커파일 이미지 빌드'
docker build -t springbootapp .
```

```
echo '컨테이너 중지'
docker stop springbootapp
```

```
echo '컨테이너 삭제'
docker rm springbootapp
```

```
echo '컨테이너 실행'
docker run -p 8080:8080 --name springbootapp --network ubuntu_default
-d springbootapp
```

## Docker

### - docker-compose.yml

```
version: "3"
services:
  mydb:
    image: mysql
    container_name: 원하는 DB컨테이너명
    environment:
      MYSQL_DATABASE: 원하는 SCHEME명
      MYSQL_ROOT_PASSWORD: 원하는 ROOT 계정 PASSWORD
    volumes:
      - /mydb:/var/lib/mysql
    ports:
      - 3306:3306

  redis:
    container_name: redis
    image: redis
    ports:
      - 6379:6379
    command: redis-server --requirepass REDIS_PASSWORD
    environment:
      - REDIS_REPLICATION_MODE=master

  nginx:
    image: nginx
    container_name: 원하는 컨테이너명
```



```
ports:
  - 80:80
  - 443:443
volumes:
  - /etc/letsencrypt:/etc/letsencrypt
  - ./nginx/conf.d:/etc/nginx/conf.d

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

## Nginx

### - app.conf

```
server {
    listen 80;
    server_name 서비스도메인 www.서비스도메인;

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

server {
    listen 443 ssl;
    server_name 서비스도메인;
    access_log off;

    ssl_certificate
    /etc/letsencrypt/live/서비스도메인/fullchain.pem;
    ssl_certificate_key
    /etc/letsencrypt/live/서비스도메인/privkey.pem;

    location / {
        proxy_pass http://서비스도메인:3000;
        proxy_set_header Host $host:$server_port;
        proxy_set_header X-Forwarded-Host $server_name;
    }
}
```

```

proxy_set_header X-Real-IP $remote_addr;
proxy_set_header X-Forwarded-For
$proxy_add_x_forwarded_for;
proxy_set_header X-Forwarded-Proto $scheme;
proxy_set_header Upgrade $http_upgrade;
proxy_set_header Connection "upgrade";
proxy_redirect off;
}

location /api/ {
proxy_pass http://서비스도메인:8080/;
proxy_set_header Host $host;
proxy_set_header X-Forwarded-Host $server_name;
proxy_set_header X-Real-IP $remote_addr;
proxy_set_header X-Forwarded-For
$proxy_add_x_forwarded_for;
proxy_set_header X-Forwarded-Proto $scheme;
proxy_set_header Upgrade $http_upgrade;
proxy_set_header Connection "upgrade";
proxy_redirect off;
# add_header 'Access-Control-Allow-Origin' '*' always;
# add_header 'Access-Control-Allow-Methods' 'GET, POST,
PUT, DELETE, OPTIONS' always;
# add_header 'Access-Control-Allow-Headers' 'content-type,
authorization, x-requested-with' always;
}
}

```

## 4. Docker 설치

### - Docker 인스톨

```

sudo apt-get update
sudo apt-get install W
ca-certificates W
curl W
gnupg W
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 W
"deb [arch=$(dpkg --print-architecture) signed-
by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/ubuntu W
$(lsb_release -cs) stable" | sudo tee
/etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update

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

#### - Docker Compose 인스톨

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

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

## 5. SSL 인증서 발급

```
sudo apt-get install letsencrypt

sudo letsencrypt certonly --standalone -d www제외한도메인이름

이메일 작성 후 Agree

뉴스레터 수신 여부 Yes/No

ssl_certificate /etc/letsencrypt/live/도메인이름/fullchain.pem;
ssl_certificate_key /etc/letsencrypt/live/도메인이름/privkey.pem;
```

## 6. OpenVidu 배포

#### - 기존 OpenVidu가 있다면 삭제

```
$ sudo docker ps -a
```

#openvidu, kurento media server 등의 컨테이너가 존재한다면 삭제한다.

```
$ sudo docker rm <ID or Name>
```

#컨테이너 모두 삭제를 원할 경우

```
$ sudo docker rm $(docker ps -a)
```

# 이미지도 삭제

```
$ sudo docker rmi <ID or IMAGE>
```

# 이미지 전체 삭제를 원할 경우

```
$ sudo docker rmi $(docker images)
```

#### - OpenVidu 설치

# 관리자 권한

```
$ sudo su
```

# openvidu가 설치되는 경로

```
$ cd /opt
```

# openvidu on promises 설치

```
$ curl https://s3-eu-west-1.amazonaws.com/aws.openvidu.io/install_openvidu_latest.sh | bash
```

```
$ exit
```

#### - OpenVidu 설정 및 SSL 인증서 적용

```
$ cd /opt/openvidu
```

```
$ vi .env
```

```
DOMAIN_OR_PUBLIC_IP=<도메인 또는 public IP>
OPENVIDU_SECRET=MY_SECRET
CERTIFICATE_TYPE=letsencrypt
LETSencrypt_EMAIL=인증서 발급 시 입력한 이메일
HTTP_PORT=8442
HTTPS_PORT=8443
...
```

```
# ESC 입력 후 :wq! 로 저장 후 나가기
```

#### - OpenVidu 포트 개방

```
sudo apt update
```

```
sudo apt install netfilter-persistent
```

```
sudo apt install iptables-persistent
```

```
sudo service iptables start
```

```
sudo iptables -A INPUT -p udp --match multiport --dports 40000:65535
-j ACCEPT
```

```
sudo iptables -A INPUT -p tcp --match multiport --dports 40000:65535
-j ACCEPT
```

```
sudo iptables -I INPUT 1 -p tcp --dport 22 -j ACCEPT
```

```
sudo iptables -I INPUT 1 -p tcp --dport 80 -j ACCEPT
```

```
sudo iptables -I INPUT 1 -p tcp --dport 443 -j ACCEPT
```

```
sudo iptables -I INPUT 1 -p tcp --dport 3478 -j ACCEPT
```

```
sudo iptables -I INPUT 1 -p udp --dport 3478 -j ACCEPT
```

```
sudo iptables -I INPUT 1 -p tcp --dport 8082 -j ACCEPT
sudo iptables -I INPUT 1 -p tcp --dport 8443 -j ACCEPT
```

```
sudo iptables -A INPUT -p tcp --dport 8080 -j ACCEPT
sudo service iptables save
sudo service iptables restart
sudo netfilter-persistent save
sudo netfilter-persistent start
```

#### - OpenVidu 서비스(관련 컨테이너) 실행

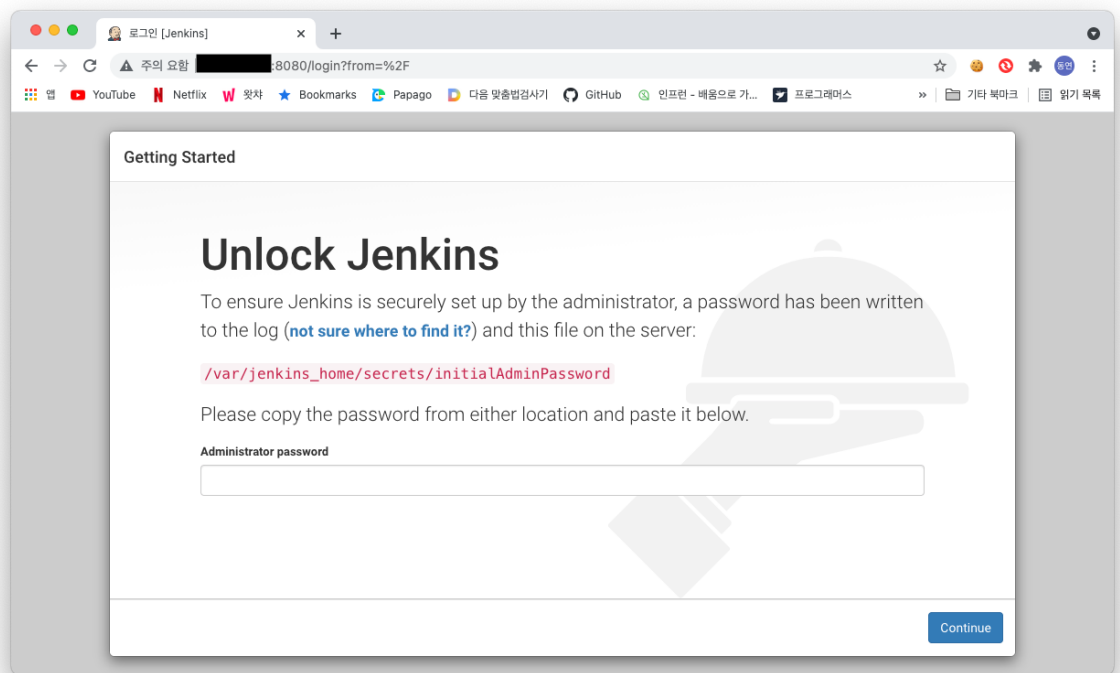
```
# /opt/openvidu 위치에서
$ sudo ./openvidu start
```

```
# 종료할 때는 같은 경로에서
$ ./openvidu stop
```

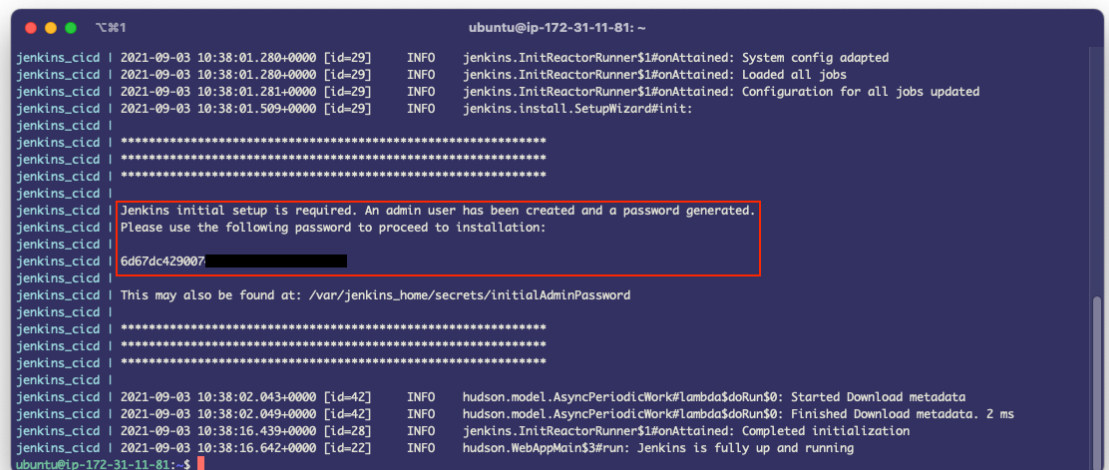
## 7. DB 및 Infra 배포

- **docker-compose 작성** : 본 문서의 1-2, 1-3의 docker-compose.yml 참조
- **docker-compose 실행**

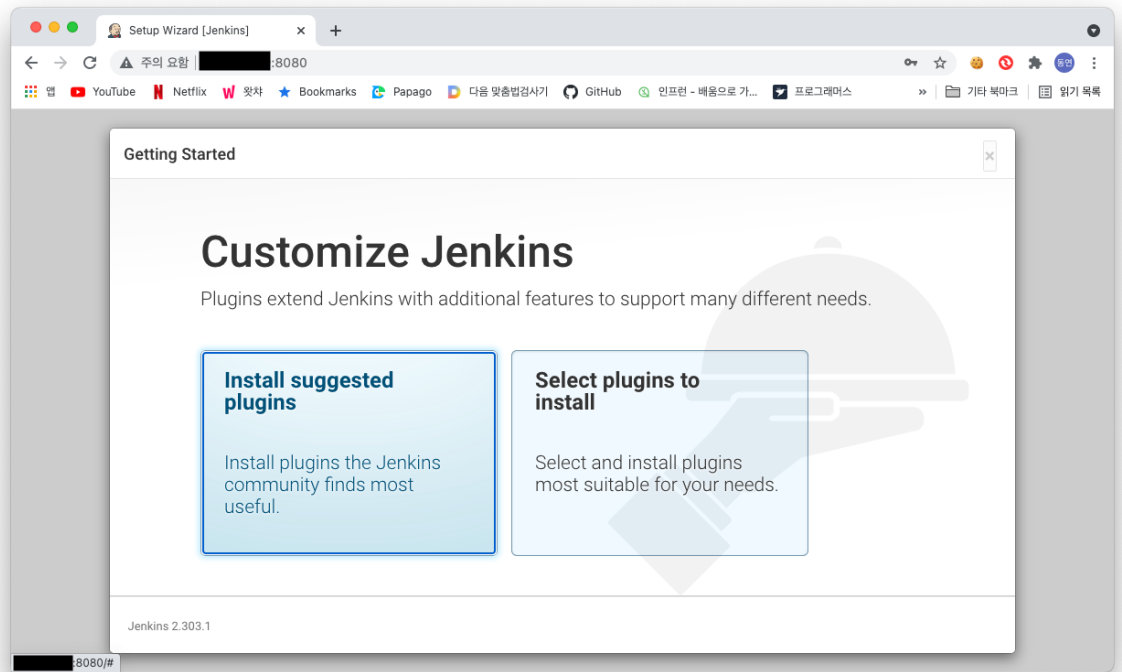
```
# /home/ubuntu (docker-compose.yml 경로에서)
$ sudo docker-compose up -build -d
```
- **Nginx 설정** : 본 문서의 1-2, 1-3의 app.conf 참조
- **Jenkins 설정**



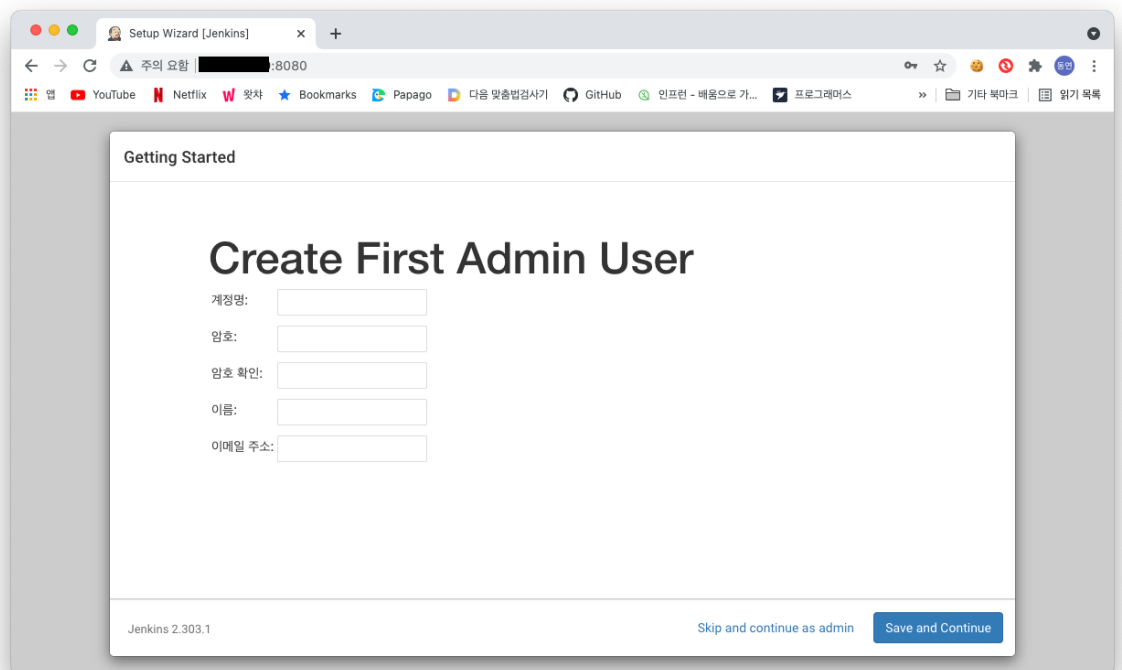
1. http://서비스도메인:9090 으로 접속하여 jenkins 페이지 진입



2. 서버 콘솔에서 sudo docker logs [Jenkins 컨테이너 이름]으로 Administrator password를 확인하고 입력합니다.



3. Install suggested plugins을 선택하여 플러그인들을 설치합니다.



4. 생성할 관리자 계정 정보를 입력하고 Save and Continue

5. Jenkins 접속 URL 확인 후 Save and Finish



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

Q gitlab

이름 ↓	사용가능
<b>Generic Webhook Trigger Plugin</b> 1.84 Can receive any HTTP request, extract any values from JSON or XML and trigger a job with those values available as variables. Works with GitHub, GitLab, Bitbucket, Jira and many more. <a href="#">Report an issue with this plugin</a>	<input checked="" type="checkbox"/> <input type="checkbox"/>
<b>Gitlab API Plugin</b> 5.0.1-78.v47a_45b_9f78b_7 This plugin provides <a href="#">GitLab API</a> for other plugins. <a href="#">Report an issue with this plugin</a>	<input type="checkbox"/> <input type="checkbox"/>
<b>GitLab Authentication plugin</b> 1.16 This is the an authentication plugin using gitlab OAuth. <a href="#">Report an issue with this plugin</a>	<input checked="" type="checkbox"/> <input type="checkbox"/>
<b>GitLab Plugin</b> 1.5.35 This plugin allows <a href="#">GitLab</a> to trigger Jenkins builds and display their results in the GitLab UI. <a href="#">Report an issue with this plugin</a>	<input checked="" type="checkbox"/> <input type="checkbox"/>

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

Q docker

이름 ↓	사용가능
<b>Docker API Plugin</b> 3.2.13-37.vf3411c9828b9 This plugin provides <a href="#">docker-java</a> API for other plugins. <a href="#">Report an issue with this plugin</a>	<input type="checkbox"/> <input type="checkbox"/>
<b>Docker Commons Plugin</b> 1.19 Provides the common shared functionality for various Docker-related plugins. <a href="#">Report an issue with this plugin</a>	<input type="checkbox"/> <input type="checkbox"/>
<b>Docker Pipeline</b> 1.29 Build and use Docker containers from pipelines. <a href="#">Report an issue with this plugin</a>	<input checked="" type="checkbox"/> <input type="checkbox"/>
<b>Docker plugin</b> 1.2.9 This plugin integrates Jenkins with <a href="#">Docker</a> <a href="#">Report an issue with this plugin</a>	<input checked="" type="checkbox"/> <input type="checkbox"/>

6. 메인 화면에서 Dashboard -> Manager Jenkins -> Plugin Manager에서 gitlab, docker에 대해서 플러그인을 설치합니다.

7. 서버 콘솔로 돌아가 Jenkins 내부에 docker를 설치합니다.

```
$ sudo docker exec -it jenkins컨테이너명 /bin/bash
```

```
$ apt-get update -y
```

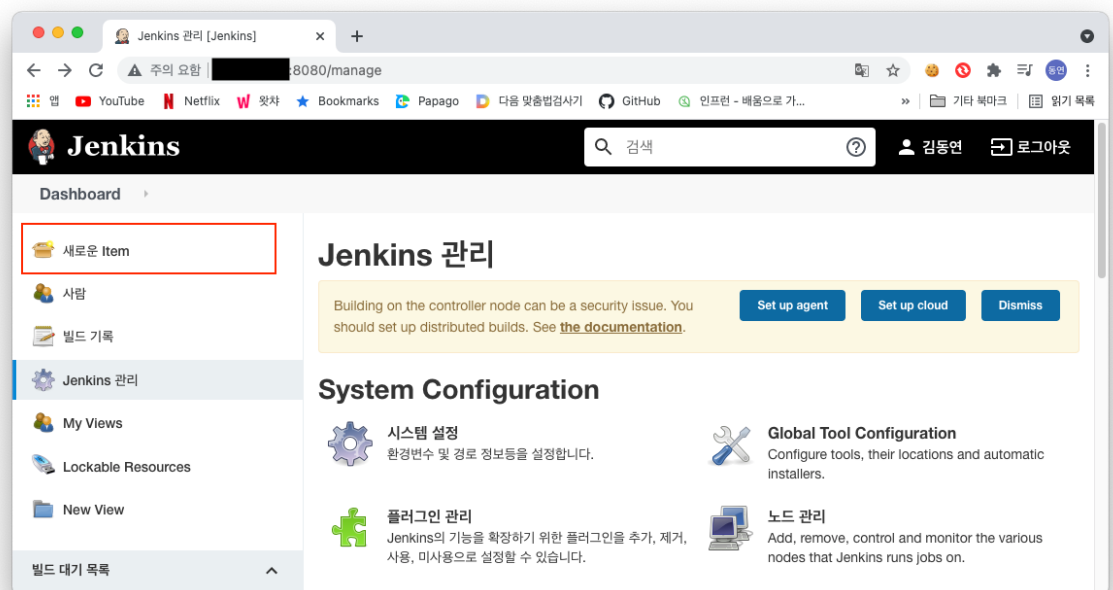
```
$ apt-get install -y
```

```
$ apt-get install docker.io -y
```

```
$ docker -v
```

## 8. Backend CI/CD

- application.properties 작성: 본 문서의 I-2, I-3 applictaion.properties 참조
- application-oauth.properties 작성: 본 문서 I-2, I-3 application-oauth.properties 참조  
(본 문서의 II-1 소셜 로그인 부분을 먼저 진행하시는 것이 좋습니다.)
- application.properties(test) 작성: 본 문서의 I-2, I-3 application.properties(test) 참조
- Dockerfile 작성: 본 문서의 I-2, I-3 Backend(Spring) Dockerfile 참조
- Jenkins Job 설정



1. Jenkins 메인 화면 -> Dashboard -> 새로운 Item 클릭

## 2. FreeStyle project를 선택하고 item name은 backend로 설정 후 OK

소스 코드 관리

☐ None

☒ Git ?

Repositories ?

Repository URL ? ✕

https://lab.ssafy.com/s07-webmobile1-sub2/S07P12A805.git

Credentials ?

las1260@naver.com/\*\*\*\*\* ▼

+ Add

고급...

- Repositories URL에는 프로젝트 레포지토리의 HTTPS Clone 주소 입력  
Credentials의 Add를 클릭하고,  
Domain -> Global credentials  
Kind -> Username with password  
Username -> 레포지토리 접근 권한이 있는 Gitlab 계정 아이디  
Password -> Username에 작성한 Gitlab 계정 비밀번호  
를 순서대로 입력하고 드롭박스에서 생성된 Credential을 선택해준다.

Branches to build ?

Branch Specifier (blank for 'any') ? ✕

refs/heads/backend

Add Branch

- backend 브랜치의 내용만을 받아와서 빌드하기 위한 설정을 해줍니다.

**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://3.39.251.36:9090/project/backend> ?

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

저장 Apply

5. 빌드 유발을 다음과 같이 설정합니다.

하단의 고급 기능을 눌러 Secret Token을 Generate하여 기록해둡니다.

**Build**

Execute shell ?

Command

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

```
echo 'jenkinsbuild started...'
pwd
cd backend
chmod +x gradlew
```

고급...

Add build step

6. Build 탭에 Excute shell을 선택하고

cd backend

chmod +x gradlew

./deploy.sh

을 작성해줍니다.

- deploy.sh 작성: 본 문서의 1-2, 1-3의 deploy.sh 참조
- jenkins에 sudo 권한 부여

```
$ sudo vim /etc/sudoers
```

```
# Allow members of group sudo to execute any command
%sudo    ALL=(ALL:ALL) ALL
jenkins  ALL=(ALL) NOPASSWD: ALL
```

해당 부분에 jenkins ALL=(ALL) NOPASSWD: ALL 작성

- Gitlab Webhook 작성

#### Webhooks

Webhooks enable you to send notifications to web applications in response to events in a group or project. We recommend using an [integration](#) in preference to a webhook.

#### URL

URL must be percent-encoded if it contains one or more special characters.

#### Secret token

Used to validate received payloads. Sent with the request in the `X-Gitlab-Token` HTTP header.

#### Trigger

☒ Push events

Push to the repository.

Gitlab 프로젝트 Repository의 Setting > Webhooks 로 이동

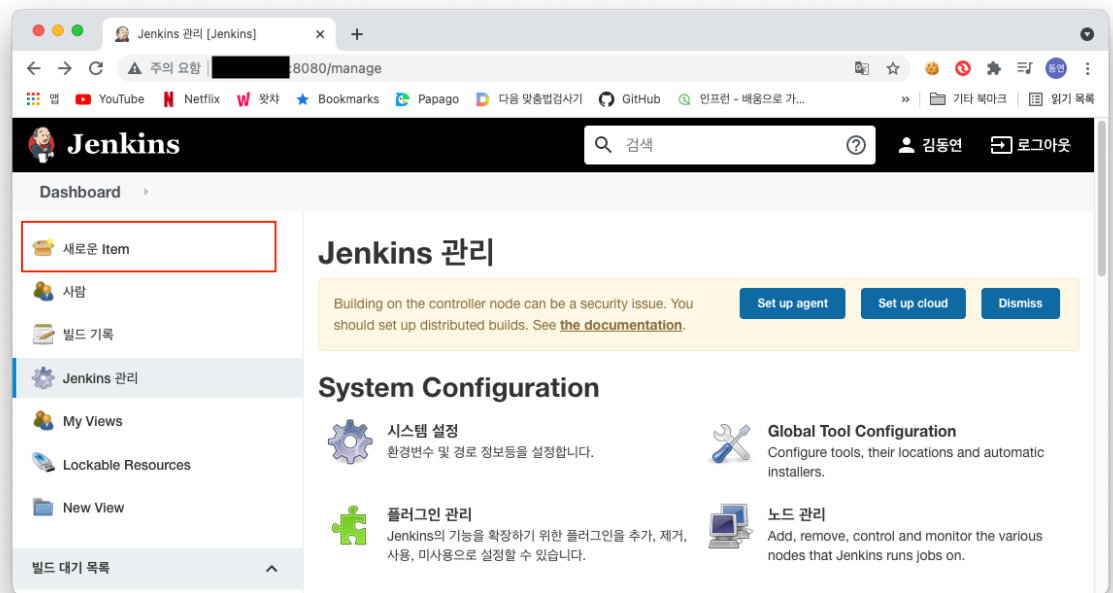
URL에는 Jenkins의 빌드 유발의 Webhook URL을 입력

Secret Token에는 빌드 유발에서 생성했던 Secret Token을 입력

Trigger의 Push events를 체크하고 backend 입력 후 Add webhook

## 9. Frontend CI/CD

- .env 작성: 본 문서의 1-2, 1-3 .env 참조
- Dockerfile 작성: 본 문서의 1-2, 1-3 Frontend(React) Dockerfile 참조
- jenkins Job 설정



1. Jenkins 메인 화면 -> Dashboard -> 새로운 Item 클릭
2. FreeStyle project를 선택하고 item name은 frontend로 설정 후 OK

소스 코드 관리

☐ None

☒ Git ?

Repositories ?

Repository URL ?

https://lab.ssafy.com/s07-webmobile1-sub2/S07P12A805.git

Credentials ?

las1260@naver.com/\*\*\*\*\*

+ Add

고급...

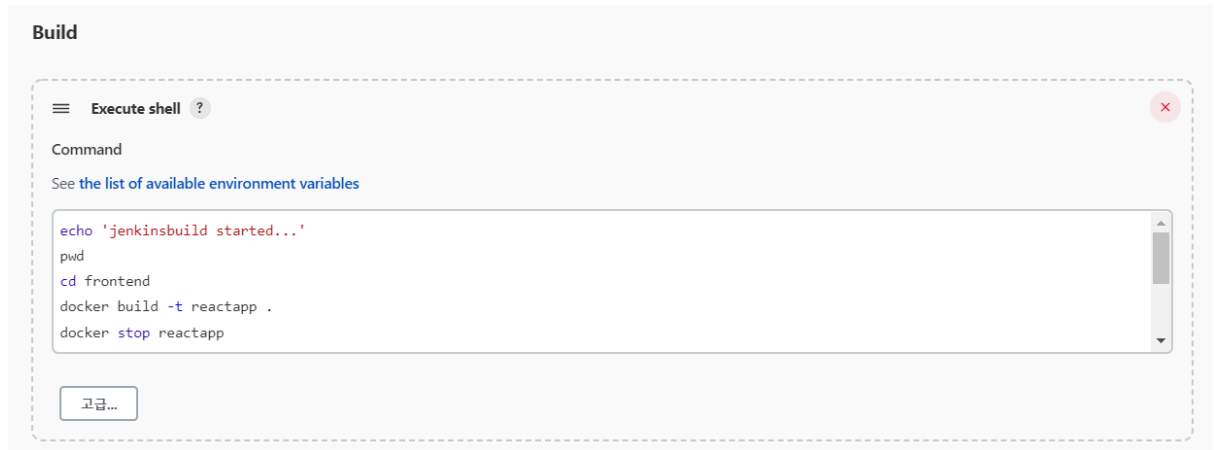
3. Repositories URL에는 프로젝트 레포지토리의 HTTPS Clone 주소 입력  
Credentials의 Add를 클릭하고,  
Domain -> Global credentials  
Kind -> Username with password

Username -> 레포지토리 접근 권한이 있는 Gitlab 계정 아이디  
 Password -> Username에 작성한 Gitlab 계정 비밀번호  
 를 순서대로 입력하고 드롭박스에서 생성된 Credential을 선택해준다.

#### 4. frontend 브랜치의 내용만을 받아와서 빌드하기 위한 설정

#### 5. 빌드 유발을 다음과 같이 설정합니다.

하단의 고급 기능을 눌러 Secret Token을 Generate하여 기록해둡니다.



## 6. Build 탭에 Excute shell을 선택하고

cd frontend

docker build -t reactapp .

docker stop reactapp

docker rm reactapp

docker run -p 3000:3000 --name reactapp --network ubuntu\_default  
-d reactapp

을 작성해줍니다.

## - Gitlab Webhook 작성

### Webhooks

Webhooks enable you to send notifications to web applications in response to events in a group or project. We recommend using an [integration](#) in preference to a webhook.

### URL

URL must be percent-encoded if it contains one or more special characters.

### Secret token

Used to validate received payloads. Sent with the request in the `X-Gitlab-Token` HTTP header.

### Trigger

☒ Push events

Push to the repository.

Gitlab 프로젝트 Repository의 Setting > Webhooks 로 이동

URL에는 Jenkins의 빌드 유발의 Webhook URL을 입력

Secret Token에는 빌드 유발에서 생성했던 Secret Token을 입력

Trigger의 Push events를 체크하고 backend 입력 후 Add Webhook

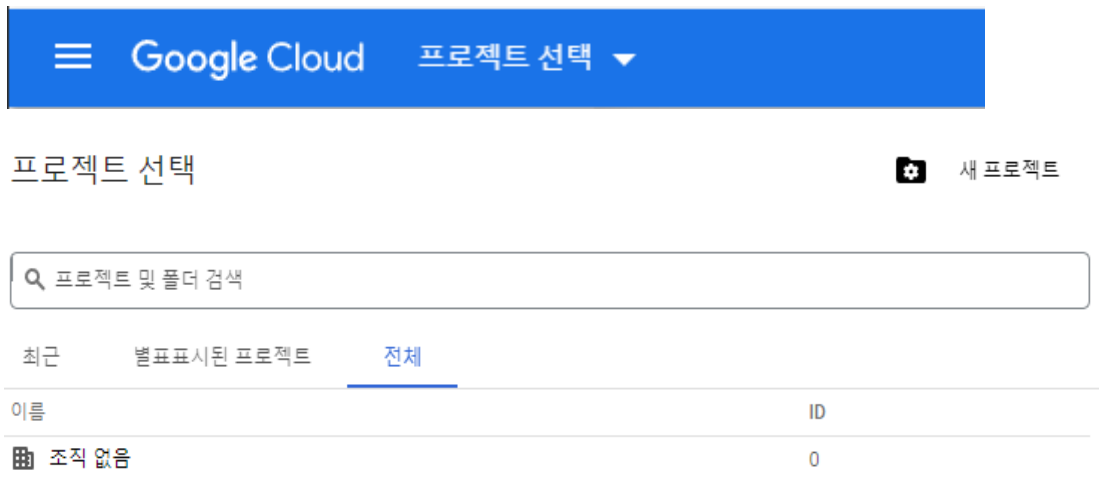


## II. 외부 서비스

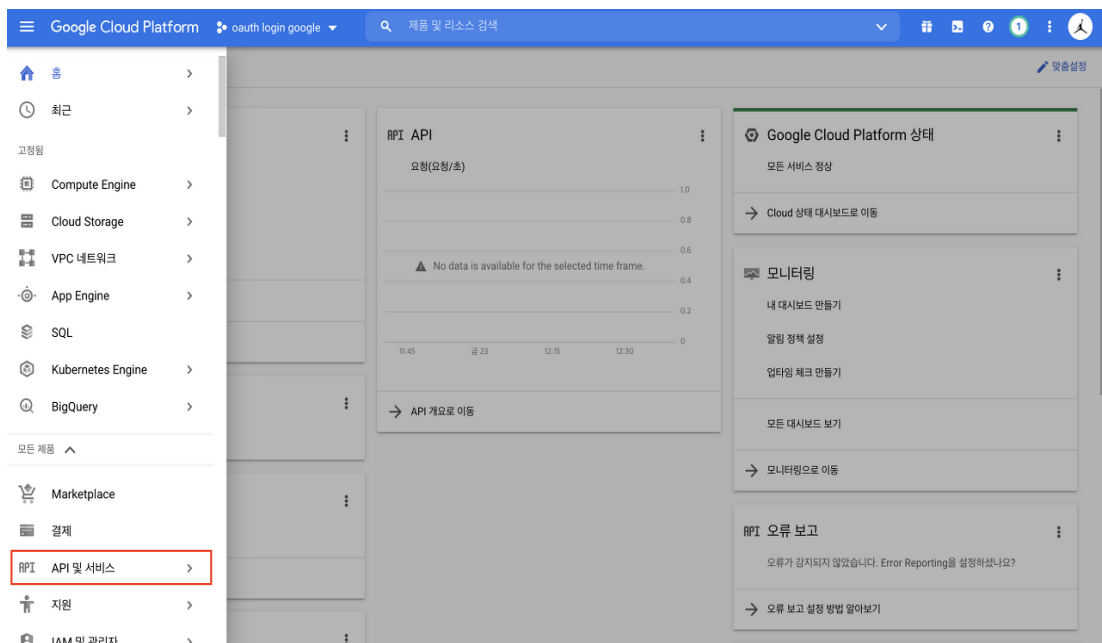
### 1. 소셜 로그인

- 구글(Google)

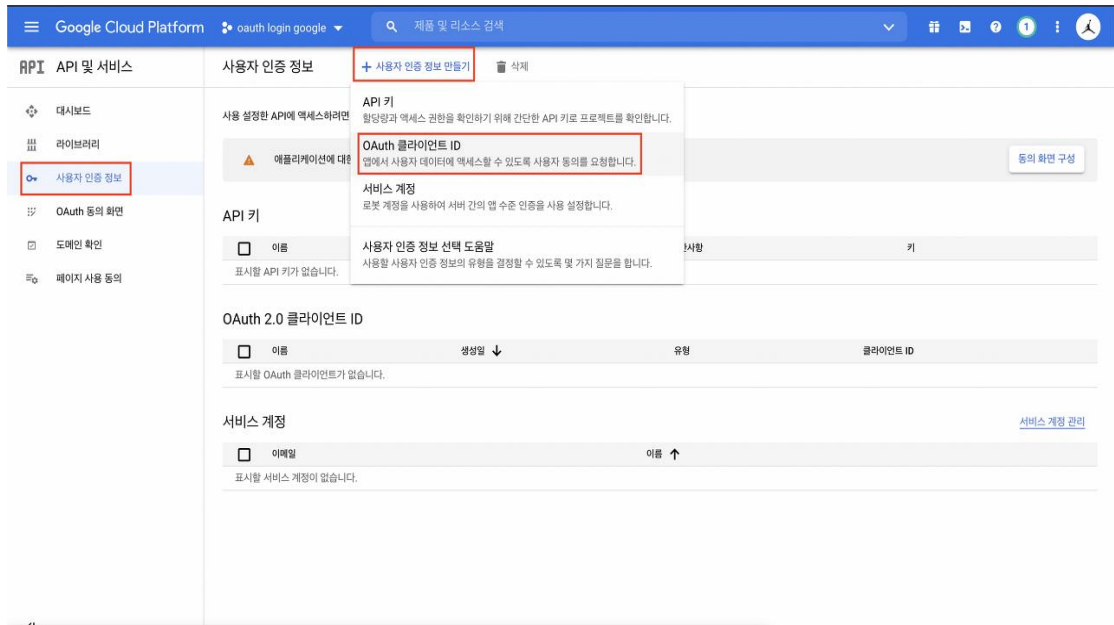
1. <https://console.cloud.google.com/home/dashboard> 로 접속한다.



2. 프로젝트 선택을 클릭하고 모달 우측 상단의 새 프로젝트 클릭



3. API 및 서비스 선택



#### 4. 사용자 인증정보 > 사용자 인증 정보 만들기 > OAuth 클라이언트 ID

### OAuth 동의 화면

대상 사용자를 비롯해 앱을 구성하고 등록하려는 방식을 선택하세요. 프로젝트에는 하나의 앱만 연결할 수 있습니다.

#### User Type

☐ 내부 ?

Google Workspace 사용자가 아니기 때문에 앱을 외부(일반 잠재고객) 사용자에게 제공하는 것만 가능합니다. [이제 앱 제출할 필요는 없](#)

☒ 외부 ?

Google 계정이 있는 모든 테스트 사용자가 사용할 수 있습니다. 앱이 테스트 모드로 시작되고 테스트 사용자 목록에 추가된 사용자에게만 제공됩니다. 앱을 프로젝트에 푸시할 준비가 되면 앱을 인증해야 할 수도 있습니다.

[사용자 유형 자세히 알아보기](#)

만들기

5. 동의 화면 구성을 클릭하고 해당 화면에서 외부로 선택하여 외부 사용자들이 사용할 수 있도록 설정합니다.

## 앱 정보

동의 화면에 표시되어 최종 사용자가 개발자를 확인하고 문의할 수 있습니다.

앱 이름 \*

집에서 운동중

동의를 요청하는 앱의 이름

사용자 지원 이메일 \*

platinadark@gmail.com

사용자가 동의 관련 질문을 위해 문의할 때 이용합니다.

앱 로고

로고.png

× 찾아보기

사용자가 앱을 알아보는 데 도움이 되도록 동의 화면에 대한 이미지(1MB 이하 크기)를 업로드합니다. 허용되는 이미지 형식은 JPG, PNG, BMP입니다. 최적의 결과를 위해서는 로고가 120x120픽셀 크기의 정사각형이어야 합니다.



6. 앱 정보를 입력하고, 개발자 연락처 정보에 이메일을 입력합니다.

Google Cloud zipzoong

API API 및 서비스

- 사용 설정된 API 및 서비스
- 라이브러리
- 사용자 인증 정보
- OAuth 동의 화면**
- 도메인 확인
- 페이지 사용 동의

앱 등록 수정

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

범위는 사용자에게 앱 승인을 요청하는 권한을 나타내며 프로젝트에서 사용자의 Google 계정에 있는 특정 유형의 비공개 사용자 데이터에 액세스하도록 허용합니다. [자세히 알아보기](#)

범위 추가 또는 삭제

민감하지 않은 범위

API	범위	사용자에게 표시되는 설명
...	../auth/userinfo.email	기본 Google 계정의 이메일 주소 확인
...	../auth/userinfo.profile	개인정보(공개로 설정한 개인정보 포함) 보기

⚠ Your sensitive scopes

민감한 범위는 비공개 사용자 데이터에 대한 액세스를 요청하는 범위입니다.

API	범위	사용자에게 표시되는 설명
...	../auth/userinfo.email	기본 Google 계정의 이메일 주소 확인
...	../auth/userinfo.profile	개인정보(공개로 설정한 개인정보 포함) 보기

선택한 범위 업데이트

아래에는 사용 설정된 API의 범위만 나와 있습니다. 이 화면에 누락된 범위를 추가하려면 [Google API 라이브러리](#)에서 API를 찾아 사용 설정하거나 아래의 '붙여넣은 범위' 텍스트 상자를 사용하세요. 라이브러리에서 사용 설정한 새 API를 확인하려면 페이지를 새로고침하세요.

필터 속성 이름 또는 값 입력

API	범위	사용자에게 표시되는 설명
<input checked="" type="checkbox"/>	../auth/userinfo.email	기본 Google 계정의 이메일 주소 확인
<input checked="" type="checkbox"/>	../auth/userinfo.profile	개인정보(공개로 설정한 개인정보 포함) 보기
<input type="checkbox"/>	openid	Google에서 내 개인 정보를 나와 연결
<input type="checkbox"/>	BigQuery API ../auth/bigquery	View and manage your data in Google BigQuery and see the email address for your Google Account
<input type="checkbox"/>	BigQuery API ../auth/cloud-platform	Google Cloud 데이터 확인, 수정, 구성, 삭제 및 Google 계정의 이메일 주소 확인
<input type="checkbox"/>	BigQuery API ../auth/bigquery.readonly	Google BigQuery에서 데이터를 봅니다.
<input type="checkbox"/>	BigQuery API ../auth/cloud-platform.read-only	Google Cloud 서비스 전체의 데이터 조회 및 Google 계정의 이메일 주소 확인
<input type="checkbox"/>	BigQuery API ../auth/devstorage.full_control	Manage your data and permissions in Cloud Storage and see the email address for your Google Account
<input type="checkbox"/>	BigQuery API ../auth/devstorage.read-only	Google 클라우드 저장소에서 데이터 조회
<input type="checkbox"/>	BigQuery API ../auth/devstorage.read_write	Cloud Storage의 데이터 관리 및 Google 계정의 이메일 주소 확인

페이지당 행 수: 10 1 - 10 (전체 25행)

7. 범위 추가 또는 삭제 > email, profile 선택 후 저장

민감하지 않은 범위에 추가된 것 확인

이름 \*  
웹 클라이언트 1

OAuth 2.0 클라이언트의 이름입니다. 이 이름은 콘솔에서 클라이언트를 식별하는 용도로만 사용되며 최종 사용자에게 표시되지 않습니다.



아래에 추가한 URI의 도메인이 승인된 도메인으로 OAuth 동의 화면에 자동으로 추가됩니다.

## 승인된 자바스크립트 원본 ?

브라우저 요청에 사용

+ URI 추가

## 승인된 리디렉션 URI ?

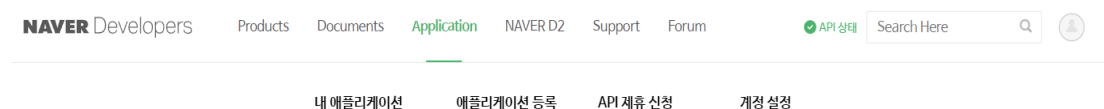
웹 서버의 요청에 사용

URI 1 \*  
`https://i7a805.p.ssafy.io/api/login/oauth2/code/google`

8. 사용자 인증정보 > 사용자 인증정보 만들기 -> OAuth 클라이언트 ID  
승인된 리디렉션 URI 작성

### - 네이버(Naver)

1. `https://developers.naver.com/main/` 로 접속한다



2. 상단의 Application > 애플리케이션 등록 클릭

애플리케이션 이름

집에서 운동중

- 네이버 로그인할 때 사용자에게 표시되는 이름이므로 서비스 브랜드를 대표할 수 있는 이름으로 가급적 10자 이내로 간결하게 설정해주세요.
- 40자 이내의 영문, 한글, 숫자, 공백문자, 쉼표(,), "/" , "-" , "\_" , 만 입력 가능합니다.

선택하세요.

▼

✓

네이버 로그인

제공 정보 선택(이용자 식별자는 기본 정보로 제공) ?

필수 항목은 개인정보보호법 제3조 제1항, 제16조 제1항 등에 따라 서비스 제공을 위해 필요한 최소한의 개인정보를 선택해야 합니다.

권한	필수	추가
회원이름	<input checked="" type="checkbox"/>	<input type="checkbox"/>
이메일 주소	<input checked="" type="checkbox"/>	<input type="checkbox"/>
별명	<input type="checkbox"/>	<input type="checkbox"/>
프로필 사진	<input checked="" type="checkbox"/>	<input type="checkbox"/>

## 서비스 URL

https://i7a805.p.ssafy.io

서비스 URL예시: (O) http://naver.com (X) http://www.naver.com  
서비스 URL값이 잘못 입력되어 있으면 정확한 값으로 수정하실 때 까지 네이버 로그인 사용이 일시적으로 제한됩니다.  
불법/음란성 사이트 등 이용약관에 위배되는 사이트의 경우, 이용이 제한될 수 있습니다.  
서비스하려는 사이트 URL과 동일한 사이트 URL로 해주셔야 **네이버 로그인 뱃지**가 노출됩니다.

## 네이버 로그인 Callback URL (최대 5개)

https://i7a805.p.ssafy.io/api/login/oauth2/code/naver



텍스트 폼 우측 끝의 '+' 버튼을 누르면 행이 추가되며, '-' 버튼을 누르면 행이 삭제됩니다.  
Callback URL은 네이버 로그인 후 이동할 페이지 URL입니다. Callback URL값이 잘못 입력되어 있으면 정확한 값으로 수정하실 때 까지 네이버 로그인 사용이 일시적으로 제한됩니다.  
입력한 주소와 다른 Callback URL로 리다이렉트 될 경우, 이용이 제한될 수 있습니다.

## 로고 이미지



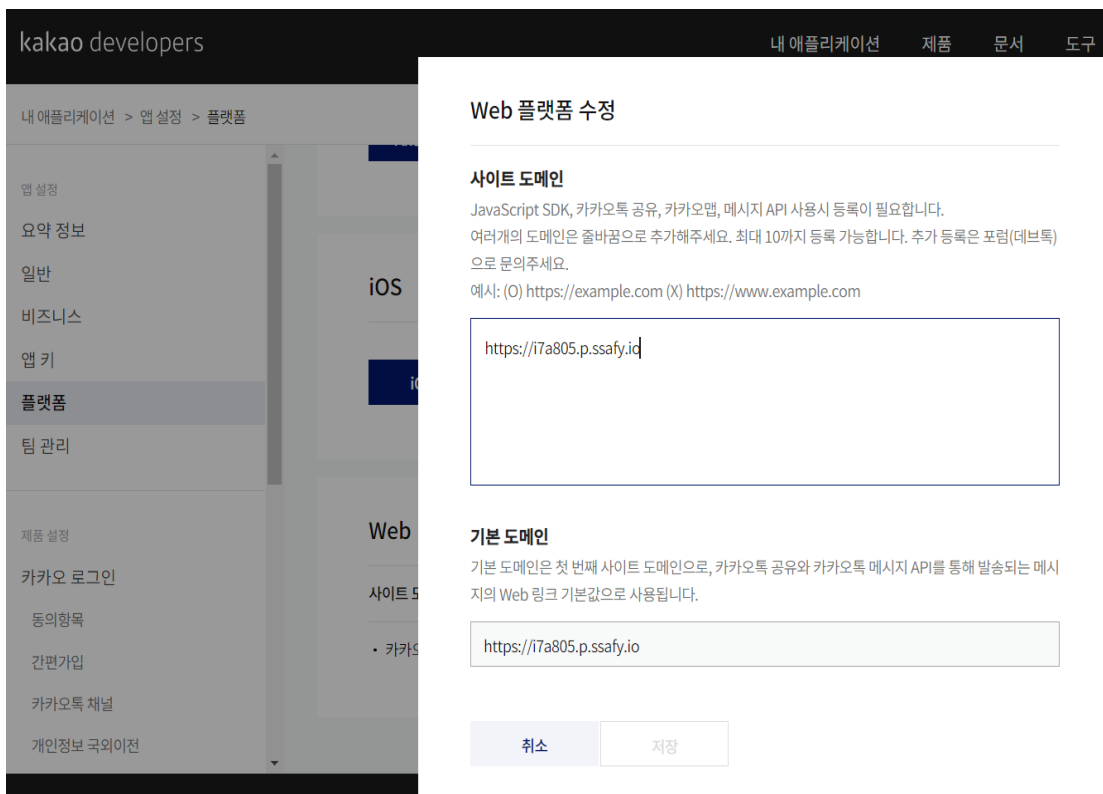
파일선택

네이버 로그인 연동 과정에서 사용자에게 보여지는 이미지이므로 서비스를 대표할 수 있는 이미지로 설정해주세요.  
권장 크기는 140X140 사이즈이며 500KB 이하의 jpg, png, gif만

3. 필요 정보를 기입합니다.

## - 카카오(Kakao)

1. <https://developers.kakao.com/> 로 접속한다
2. 내 애플리케이션 > 애플리케이션 추가하기 > 앱 이름 입력 > 사업자명 입력 > 저장
3. 좌측 Nav 바에서  
 앱 설정 > 요약 정보 > 앱 키 > REST API 키  
 제품 설정 > 카카오 로그인 > 보안 > Client Secret 의 코드 발급  
 기록해둡니다.



4. 좌측 Nav 바에서  
 앱 설정 > 플랫폼 > Web > Web 플랫폼 등록  
 필요 정보를 기입합니다.

Redirect URI

삭제 수정

Redirect URI	https://i7a805.p.ssafy.io/api/login/oauth2/code/kakao
--------------	---

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

## 5. 제품 설정 > 카카오 로그인 클릭

Redirect URI를 입력해줍니다.

### 개인정보

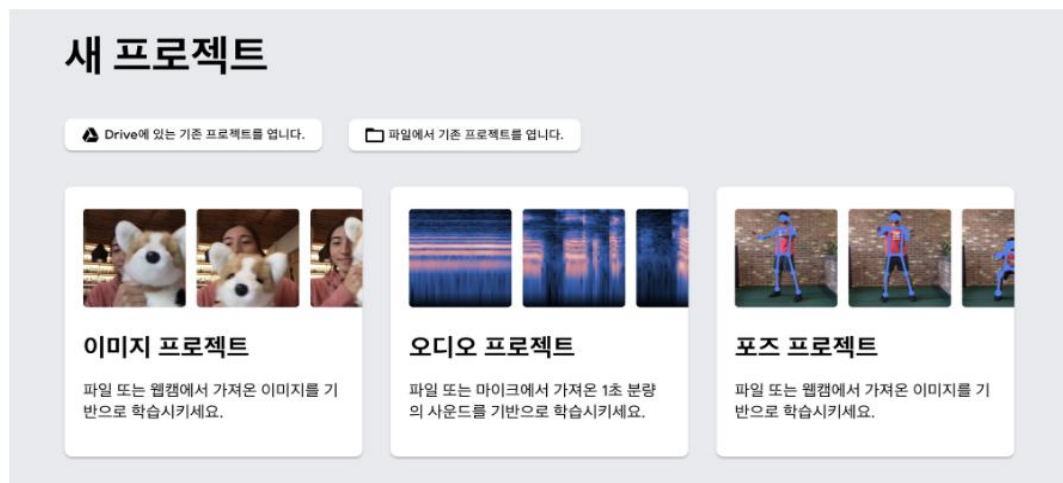
항목 이름	ID	상태	
닉네임	profile_nickname	필수 동의	설정
프로필 사진	profile_image	필수 동의	설정
카카오계정(이메일)	account_email	선택 동의 [수집]	설정

## 6. 제품 설정 > 카카오 로그인 > 동의항목에서 위처럼 설정합니다.

- **application-oauth.properties 작성:** 본 문서의 I-2, I-3의 Backend(Spring) application-oauth.properties를 참조

## 2. Teachable Machine

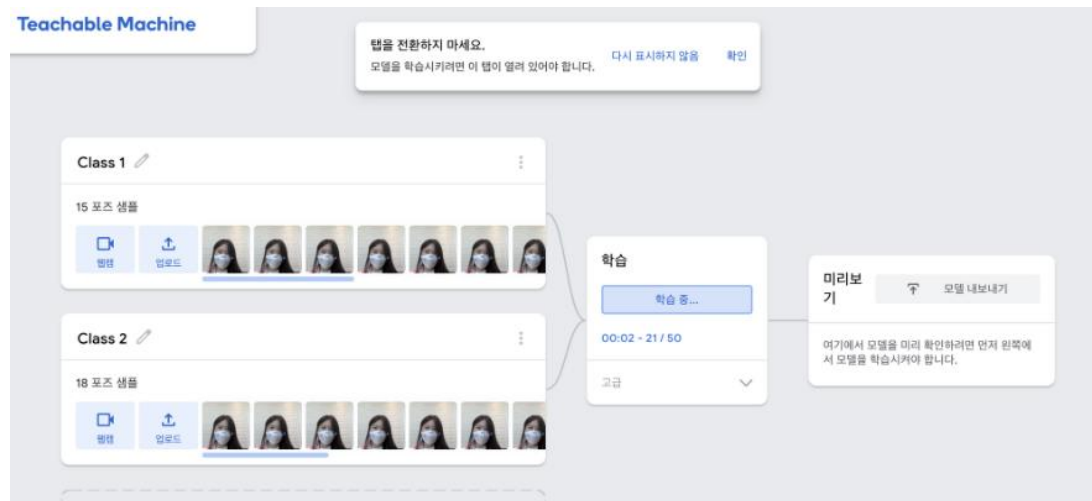
- 모델 학습



## 1. Teachable Machine 홈페이지에서 포즈 프로젝트 선택

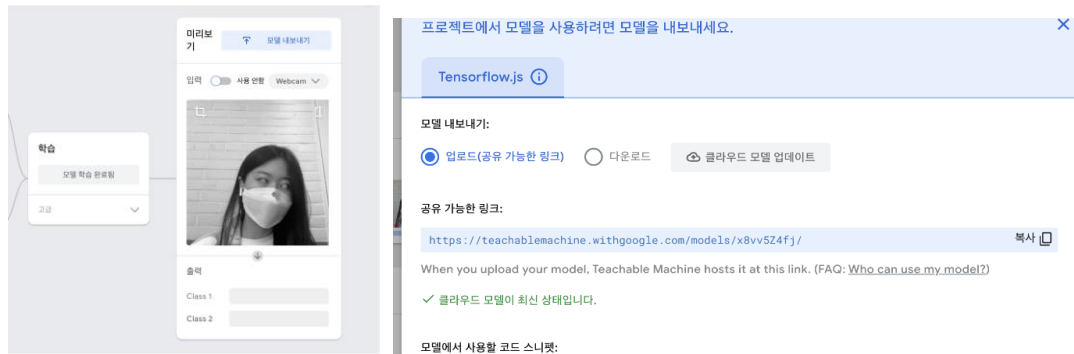


## 2. 모델의 클래스 촬영



## 3. 모델 학습시키기 버튼





#### 4. 학습 완료 후 모델 내보내기

##### - 모델 학습

1. Tensor flow, Teachable Machine Pose 라이브러리 CDN으로 삽입

```
<script
src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@1.3.1/dist/tf.min.js"></script>
<script
src="https://cdn.jsdelivr.net/npm/@teachablemachine/pose@0.8/dist/teachablemachine-pose.min.js"></script>
```

2. 학습한 모델 url로 모델 load

```
model = await tmPose.load(modelURL, metadataURL);
```