포팅 메뉴얼

A205: MaFya

삼성SW 청년아카데미 서울캠퍼스 7기

특화프로젝트 인공지능 영상 7주, 2022/8/22~2022/10/07

포팅 매뉴얼

담당 컨설턴트- 김신일

박세현(팀장), 김무종, 김주한, 홍제민, 홍성영, 김병수(중도 취업)

<<목차>>

삼성 청년 SW 아카데미의 출결 시스템을 보완하고자 MaFya를 만들었습니다. 기존의 출결 시스템 하에서는 자신의 자리에 도착하여 홈페이지 클릭을 통해 출석했지만, 객체 탐지와 얼굴 인식, 마스크 인식을 이용하여 학생들은 더 빠르게 출석할 수 있습니다.

1. 프로젝트 기술 스택

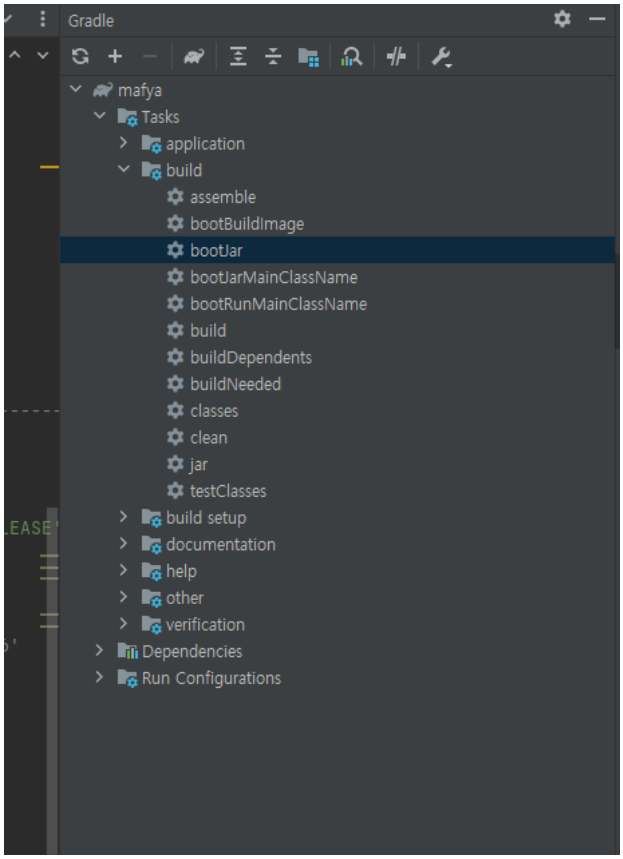
- 이슈관리 :Jira
- 형상관리: GitLab
- CI/CD: Jenkins
- 커뮤니케이션: Mattermost, Notion, Discord, Webex
- 개발환경
  - OS: Windows 10
  - IDE
    - Intellij IDEA Community Edition 2022.1.3
    - Visual Studio Code 1.69.1
  - Database: MySql WorkBench 8.0 CE
  - Server : AWS EC2
- 상세 사용
  - Backend
    - JAVA 8

        - Spring boot, Gradle
        - lombok, Swagger
        - Spring data JPA,
        - Spring Security, JWT
    - Frontend
        - React 17.0.0
        - React-router-dom 5.3.0
        - @tensorflow/tfjs 2.4.0
        - @tensorflow-models/coco-ssd 2.1.0
        - @react-webcam 5.2.0
    - AI
        - Flask
        - Tensorflow
        - Pytorch
    - AWS EC2
        - Nginx
        - Docker

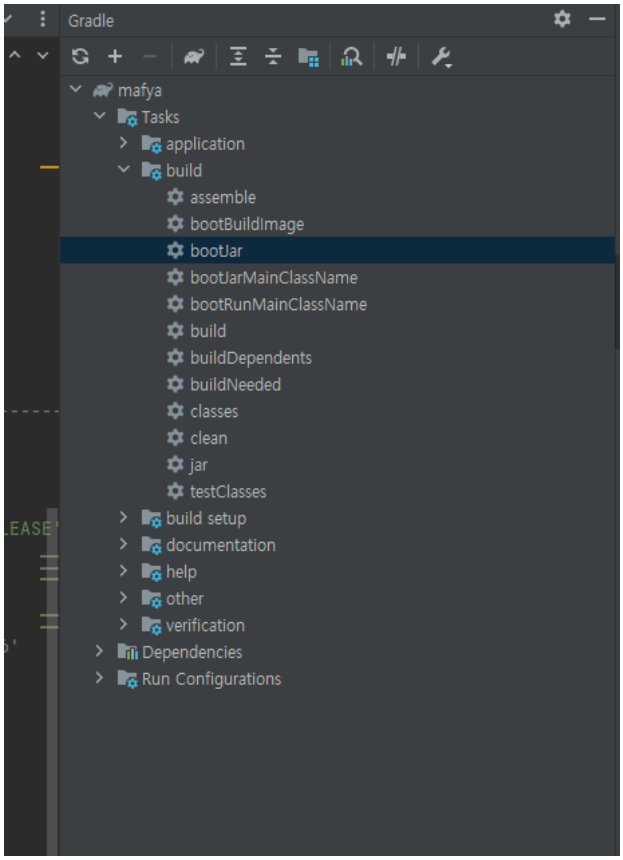
2. 빌드 상세 내용

Spring build

- 우측의 Gradle -> mafya -> Tasks -> build -> bootJar를 더블 클릭



- build 폴더가 생성되고, 그 아래에 libs 폴더 아래에 `mafya-0.0.1-SNAPSHOT.jar` 가 빌드되었는지 확인



- **실행 방법 (백그라운드)**

```
nohup java –jar mafya-0.0.1-SNAPSHOT.jar &
```

AI build

- **실행 방법(백그라운드)**
  - 클론 받은 후, ai 디렉터리 이동
  - 라이브러리 의존성 설치

```
# 라이브러리 다운로드
pip install -r requirements.txt

# torch 라이브러리 다운로드(컴퓨터 하드웨어에 따라 다름, 현재 서버는 CPU only 버전)
pip install torch==1.11.0+cpu torchvision==0.12.0+cpu torchaudio==0.11.0 --extra-index-url https://download.pytorch.org/whl/cpu

# torch 라이브러리 다운로드(컴퓨터 하드웨어에 따라 다름, 3080ti GPU 버전, 본인 GPU와 호환되는지 버전 찾아서 설치할것!)
pip install torch==1.11.0+cu113 torchvision==0.12.0+cu113 torchaudio==0.11.0 --extra-index-url https://download.pytorch.org/whl/cu

# 필요버전 체크후 torch 설치시 버전 참고사이트
https://pytorch.kr/get-started/previous-versions/
```

  - facebank폴더에 분류할 때 사용되는 값인 라벨로 폴더의 이름으로 설정하고, 해당 폴더 이름으로 학습할 사진을 넣어준다.
  - 다음 커멘드 입력

```
nohup python3 model_master.py & # 새로 추가 학습 없이 실행
nohup python3 model_master.py  --update UPDATE 1> ~/ai_server.out 2> ~/ai_server.err & # 데이터 추가시 추가학습 후 실행
```

React build

- 빌드 방법 - clone 받은 /frontend/mafya로 이동한 후, 다음 커멘드 입력

```
npm install
npm run build
```

- build 파일 생성되었는지 확인

3. 배포 특이사항

docker engine 설치

https://docs.docker.com/engine/install/ubuntu/

- Repository 설정

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

- GPG 키

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

Jenkins 설치

https://www.jenkins.io/download/

- 젠킨스 컨테이너 초기 설정

```
Dockerfile
# Dockerfile

FROM jenkins/jenkins:lts  # 젠킨스 lts 버젼을 다운로드 받음

USER root  # 이후의 명령어의 사용자를 root로 설정함

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

 # docker-compose install
 RUN curl -L "https://github.com/docker/compose/releases/download/1.28.5/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bir
    chmod +x /usr/local/bin/docker-compose && \
    ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose
```

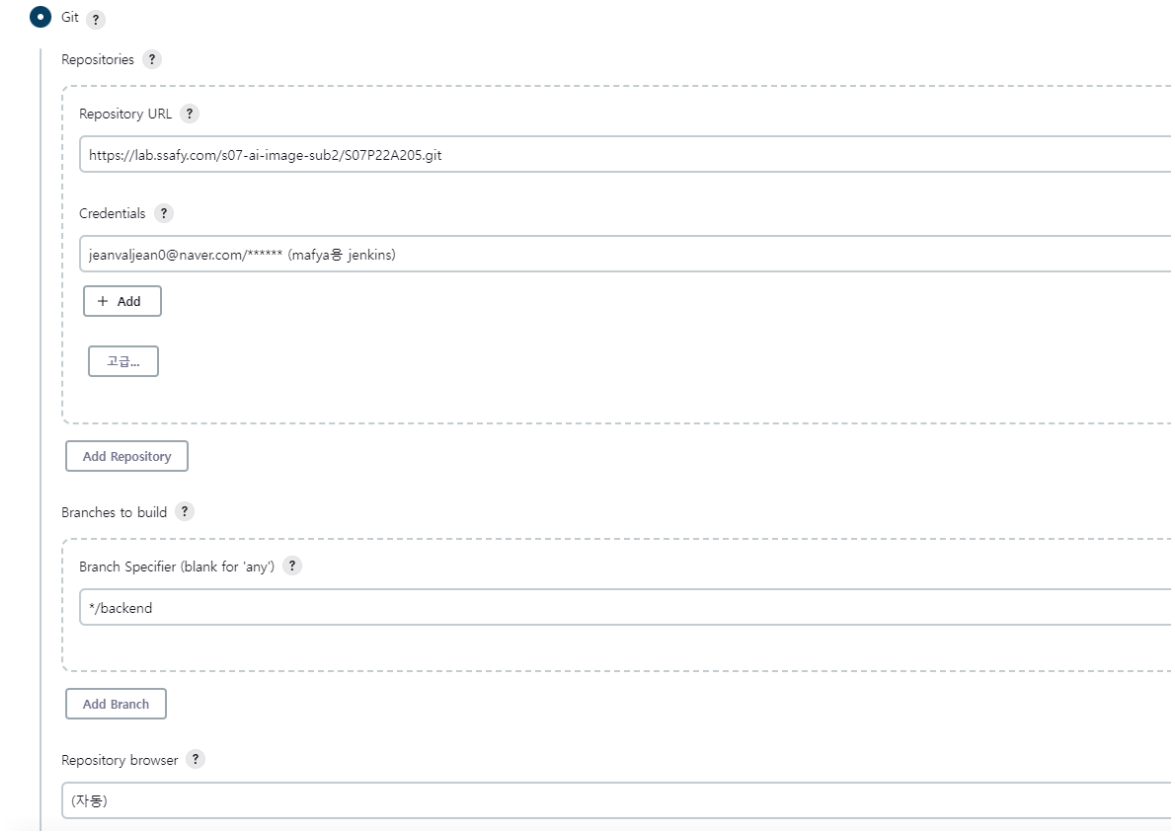
- 다음의 명령어로 외부 도커 환경과 젠킨스 컨테이너의 볼륨 연결해서 올려질 젠킨스 컨테이너 내부에서 호스트OS의 도커 데몬으로 접근할 수 있도록 설정

```
docker run -d -p 9090:8080 -v /var/run/docker.sock:/var/run/docker.sock -v /home/ubuntu/jenkins_home:/var/jenkins_home jenkins/jer
```

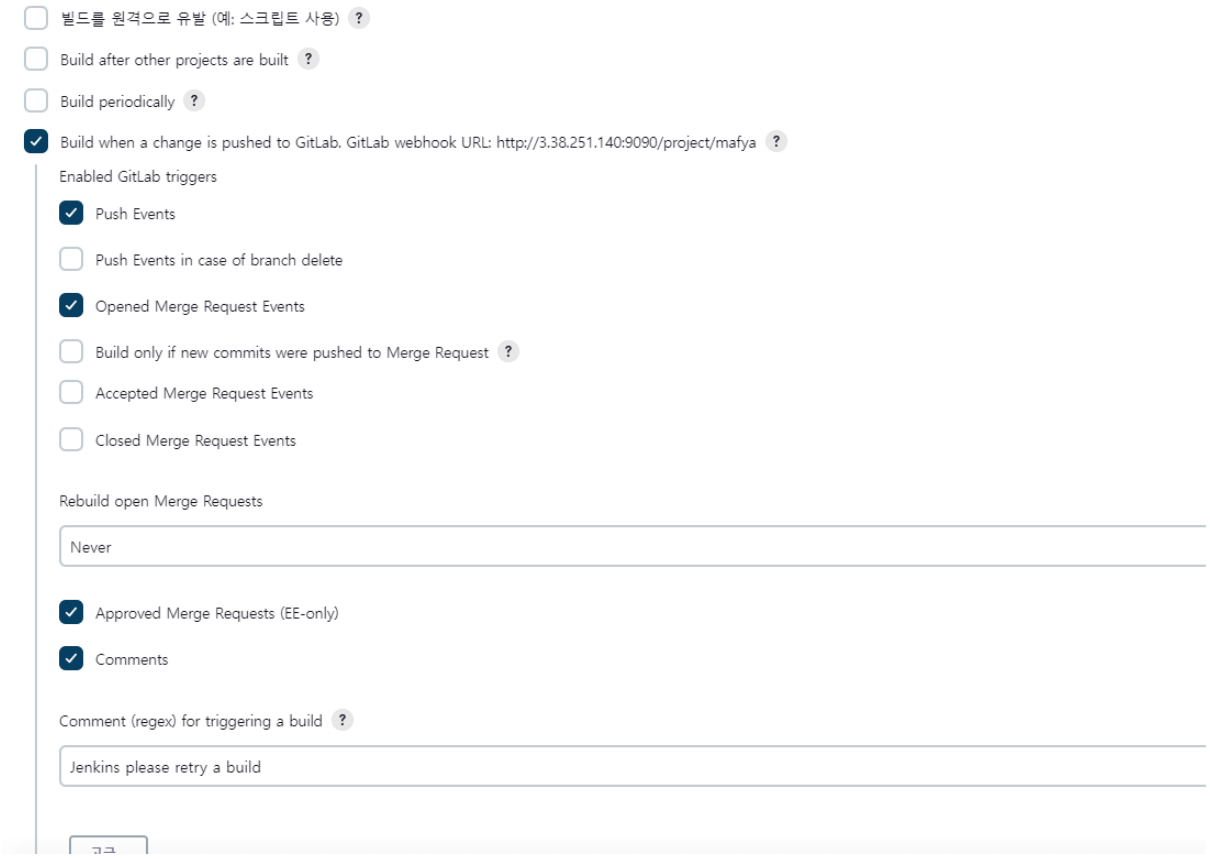
- 서버 url의 9090 포트로 접속하면, Jenkins 초기 화면 등장. `sudo cat /var/lib/jenkins/secrets/initialAdminPassword` 를 통해 초기 비밀번호 확인 후 입력
- install suggested plugins 선택하여 초기 플러그인 설치한 후, admin 계정을 생성한다.
- Docker, git, Node, Java를 설치

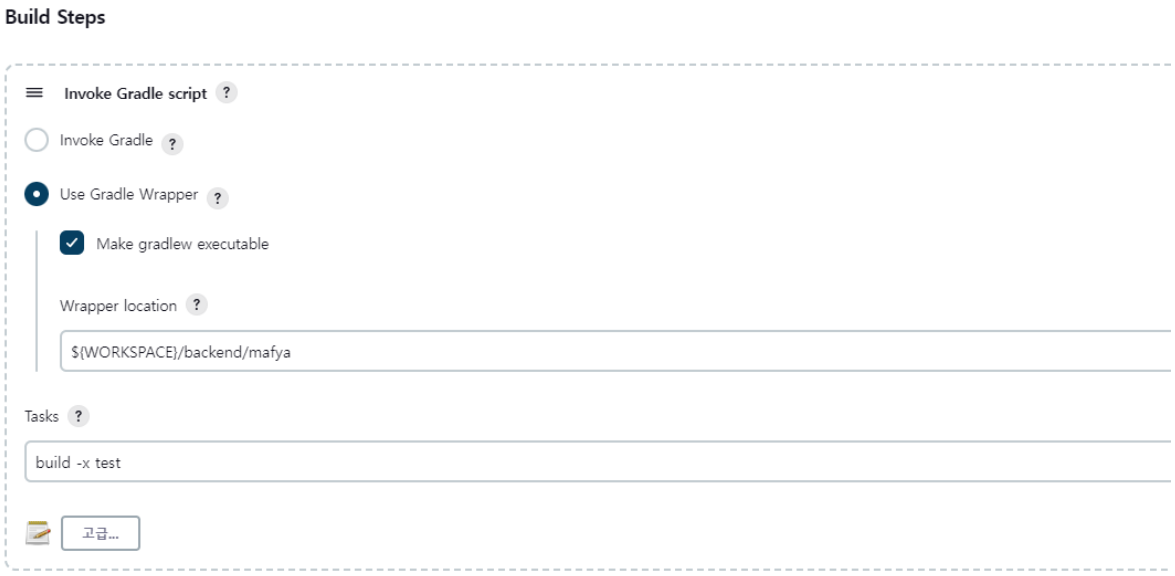
젠킨스와 Gitlab 연동(자동 배포)

- backend 프로젝트 생성
  - 필요한 플러그인을 설치 후, project 생성. branches to build는 merge request 또는 push event가 발생했을 경우 자동 빌드되는 브랜치를 의미. 이 프로젝트에서는 backend 브렌치로 설정했음
  - Execute shell을 다음과 같이 설정


Git
Repositories
Repository URL
https://lab.ssafy.com/s07-ai-image-sub2/S07P22A205.git
Credentials
jeanvaljean0@naver.com/****** (mafya용 jenkins)
+ Add
고급...
Add Repository
Branches to build
Branch Specifier (blank for 'any')
*/backend
Add Branch
Repository browser
(자동)

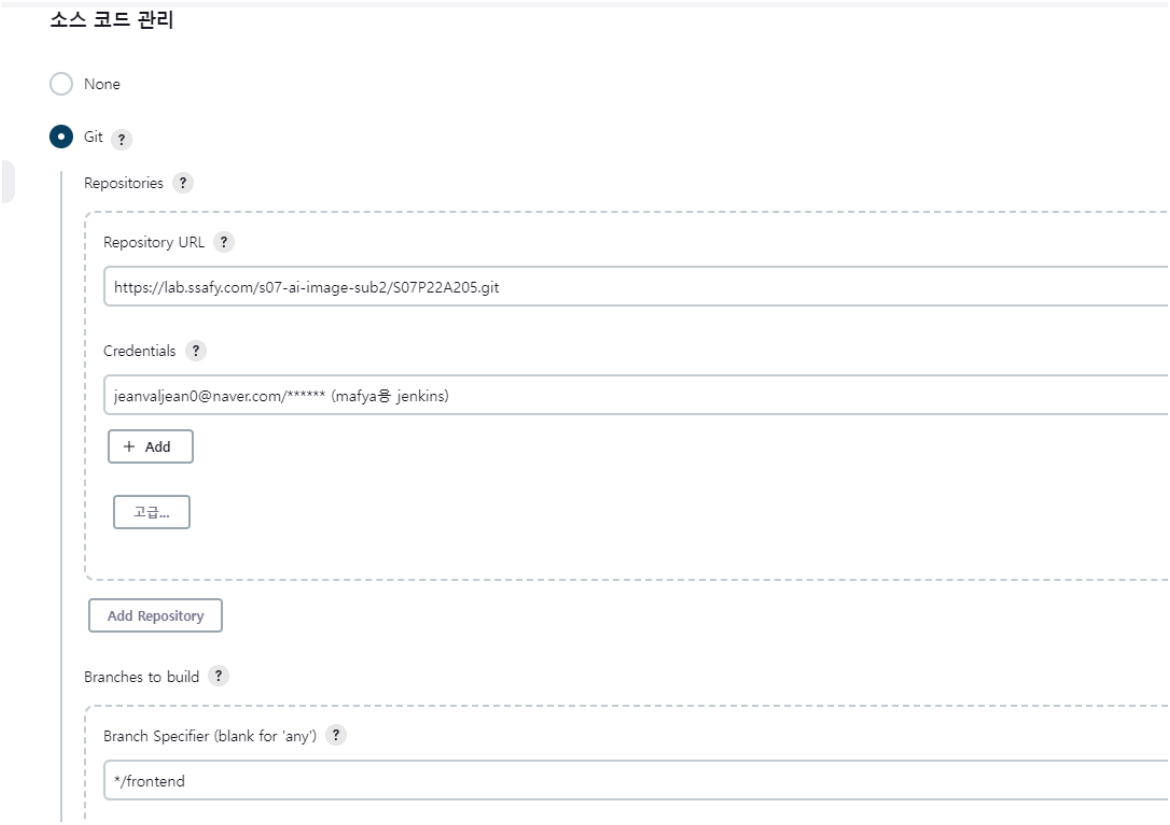
빌드 유발

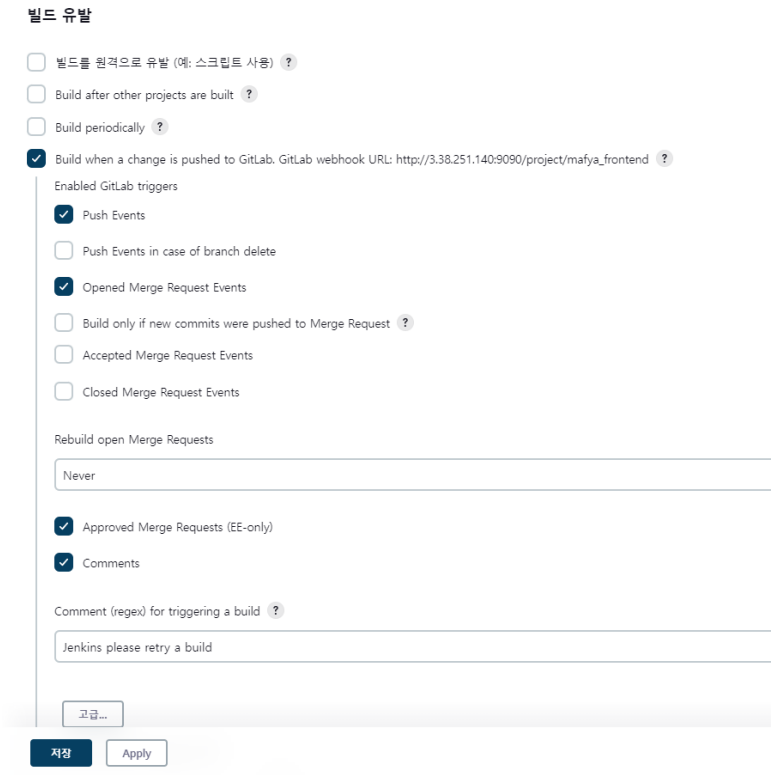

빌드를 원격으로 유발 (예: 스크립트 사용)
Build after other projects are built
Build periodically
Build when a change is pushed to GitLab. GitLab webhook URL: http://3.38.251.140:9090/project/mafya
Enabled GitLab triggers
Push Events
Push Events in case of branch delete
Opened Merge Request Events
Build only if new commits 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




```
cd ${WORKSPACE}/backend/mafya

docker ps -q --filter name=mafya_spring_container | grep -q . && docker stop mafya_spring_container && docker rm mafya_spring_containe

docker ps -a -q --filter name=mafya_spring_container | grep -q . && docker rm mafya_spring_container

docker build -t mafya_spring_image .
docker run -d -p 8080:8080 -v /home/ubuntu/ai/server/ai/facebank:/sehyeon -v /home/ubuntu/ai/server/ai/identify:/identify --name mafya
```

- frontend 프로젝트 생성
  - 이 프로젝트에서는 branches to build를 frontend로 설정
  - Execute Shell을 다음과 같이 설정






```
cd ${WORKSPACE}/frontend/mafya

docker ps -q --filter name=mafya_react_container | grep -q . && docker stop mafya_react_container && docker rm mafya_react_container

docker ps -a -q --filter name=mafya_react_container | grep -q . && docker rm mafya_react_container

docker build -t mafya_react_image .
docker run -d -p 3000:3000 --name mafya_react_container mafya_react_image
```

3 - 1 frontend 수동 배포

가) 클론받은 프로젝트 폴더에서 프론트엔드 폴더에 있는 MaFya 폴더로 이동합니다.

나) 도커 이미지 생성에 필요한 Dockerfile은 다음과 같습니다.

```
# DockerFile
FROM node:16

RUN mkdir -p /usr/src/app

WORKDIR /usr/src/app

ENV PATH /usr/src/app/node_modules/.bin:$PATH

COPY package*.json ./

RUN npm install

COPY ./ ./

EXPOSE 3000

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

다) 도커 이미지를 생성한 후, 도커 컨테이너를 통해 프론트엔드를 배포합니다.

```
# 저장소 클론
git clone https://lab.ssafy.com/s07-ai-image-sub2/S07P22A205.git

# 프론트엔드 폴더로 이동
cd frontend/mafya

# 도커 컨테이너에 있는 기존 도커 이미지 stop
docker ps -q --filter name=mafya_react_container | grep -q . && docker stop mafya_react_container && docker rm mafya_react_containe

# 도커 컨데이너에 있는 기존 도커 이미지 삭제
docker ps -a -q --filter name=mafya_react_container | grep -q . && docker rm mafya_react_container

# 도커 이미지 생성
docker build -t mafya_react_image .

# 도커 이미지 실행
docker run -d -p 3000:3000 --name mafya_react_container mafya_react_image
```

3 - 2 Backend 수동 배포

Dockerfile 내용 확인

가) 클론받은 프로젝트 폴더에서 백엔드 폴더에 있는 MaFya 폴더로 이동합니다.

나) 도커 이미지를 생성하기 위한 도커파일은 다음과 같습니다.

```
FROM openjdk:8-jdk

WORKDIR .

COPY build/libs/mafya-0.0.1-SNAPSHOT.jar application.jar

EXPOSE 8080

CMD ["java", "-jar", "application.jar"]
```

다)도커 이미지를 생성한 후, 도커 컨테이너를 통해 백엔드를 배포합니다,

```
git clone https://lab.ssafy.com/s07-ai-image-sub2/S07P22A205.git

cd backend/mafya
```

```
docker ps -q --filter name=mafya_spring_container | grep -q . && docker stop mafya_spring_container && docker rm mafya_spring_conta

docker ps -a -q --filter name=mafya_spring_container | grep -q . && docker rm mafya_spring_container

docker build -t mafya_spring_image .
docker run -d -p 8080:8080 -v /home/ubuntu/ai/server/ai/facebank:/sehyeon -v /home/ubuntu/ai/server/ai/identify:/identify --name ma
```

3 - 3 AI 서버 수동 배포

가) 클론받은 프로젝트 폴더에서 AI 폴더로 이동합니다.

나) 실행 하기 위한 스크립트는 다음과 같습니다.

```
rm -f ~/ai_server.out
rm -f ~/ai_server.err

target=$(ps -ef | grep model_master.py | awk '{print $2}' | head -1)
sudo kill -9 $target
target=$(ps -ef | grep model_master.py | awk '{print $2}' | head -1)
sudo kill -9 $target

nohup python3 model_master.py --update UPDATE 1> ~/ai_server.out 2> ~/ai_server.err &

cd ~

echo "Done"
```

---

3 - 4 SSL 인증서 적용

- SSL 인증서를 적용하기 위해 다음과 같은 명령어를 입력합니다.

```
sudo apt-get install certbot python3-certbot-nginx

sudo service stop nginx

sudo certbot certonly --standalone -d mafya.ml
```

- 이 경우 /etc/letsencrypt/live/mafya.ml에 ssl 인증서가 설치됩니다.
- /etc/nginx/sites-available로 이동, 아래와 같은 파일을 생성합니다.

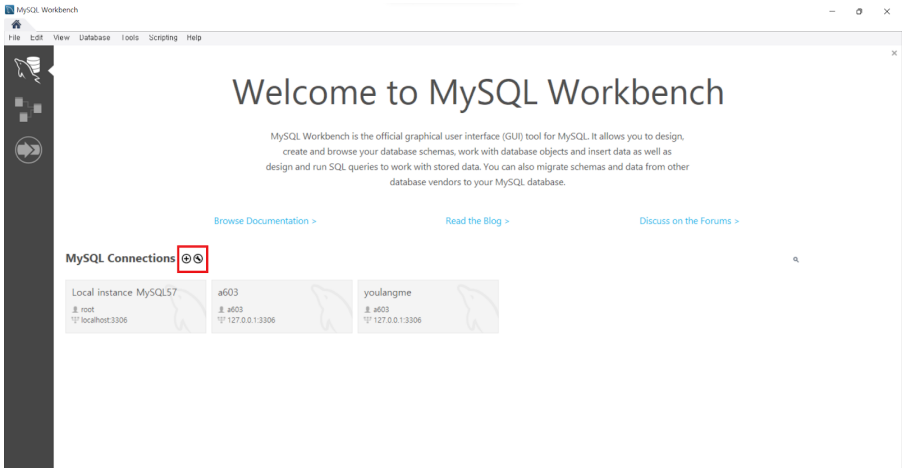
```
server {
         location /{
            proxy_pass http://localhost:3000/;
          }

          location /api{
            proxy_pass http://localhost:8080/;
         }

         location /ai{
            proxy_pass http://localhost:8081/;
         }

         location /api/images {
               alias /home/ubuntu/ai/server/ai/facebank;
         }

         listen 443 ssl;
         server_name mafya.ml;
               ssl_certificate /etc/letsencrypt/live/mafya.ml/fullchain.pem; # managed by Certbot
               ssl_certificate_key /etc/letsencrypt/live/mafya.ml/privkey.pem; # managed by Certbot
}
server {
    if ($host = mafya.ml) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

        listen 80;
        server_name mafya.ml;
    return 404; # managed by Certbot
 }
```

- 80 port로 접근할 경우 ssl인증서가 적용된 443 port 로 리다이렉트됩니다.
- 또한 443 port의 /api로 접근 시에는 localhost:8080로 분기 처리됩니다.
- 한편 443 port의 /ai로 접근 시에는 localhost:8081로 분기 처리됩니다.
- 이후 다음 명령어를 입력하면 ssl 인증서가 적용됩니다

```
sudo ln -s /etc/nginx/sites-available/[파일명] /etc/nginx/sites-enabled/[파일명]
# 필자의 경우 mafya.conf

sudo nginx -t

sudo service restart nginx
```

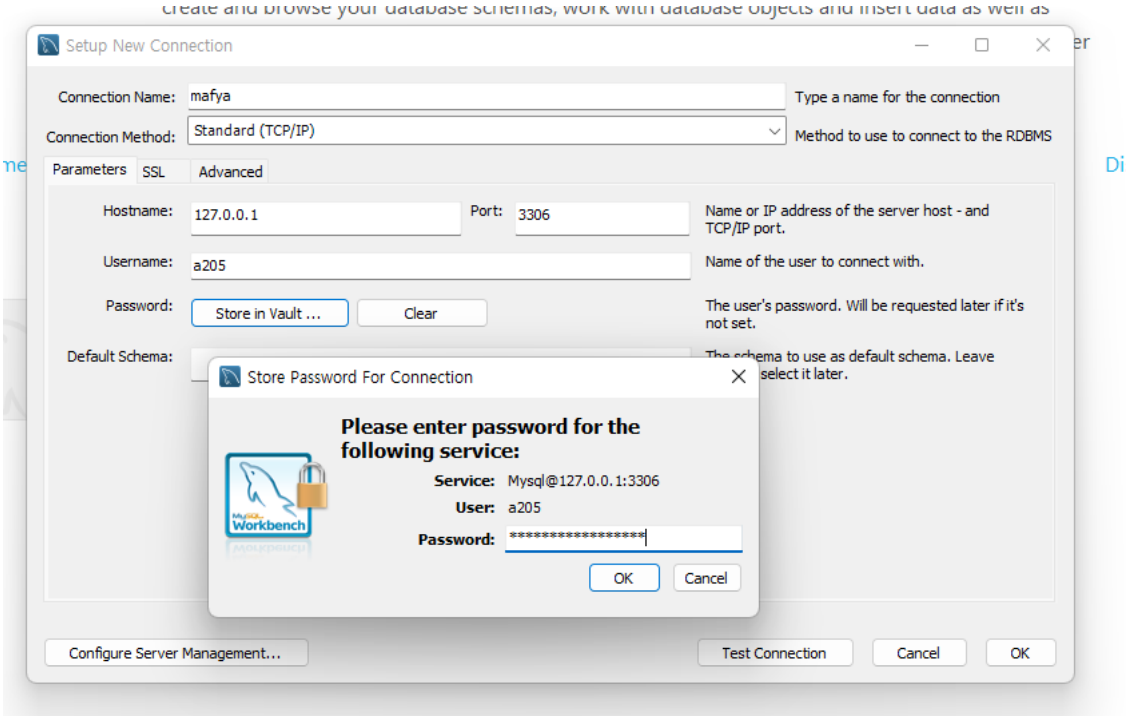
4. DB 계정

가. MySql workbench 추가하기



MySql Workbench를 열어서 새로운 내용을 추가하기 위해 “+” 버튼을 눌러줍니다.

나. EC2 계정 정보 설정



username은 a205, password는 akfldkchlrh!를 사용하였습니다.

기존 root 계정이 아닌 별도의 계정을 생성하여 프로젝트를 진행하였습니다.

5. 프로퍼티 정의

가) nginx 세팅

- Docker 사용 시에는 /etc/nginx/sites-available로 이동한 후 아래와 같은 파일을 생성합니다.

```
server {
         location /{
            proxy_pass http://localhost:3000/;
          }

          location /api{
            proxy_pass http://localhost:8080/;
         }

         location /ai{
            proxy_pass http://localhost:8081/;
         }

         location /api/images {
                alias /home/ubuntu/ai/server/ai/facebank;
         }

         listen 443 ssl;
         server_name mafya.ml;
                ssl_certificate /etc/letsencrypt/live/mafya.ml/fullchain.pem; # managed by Certbot
                ssl_certificate_key /etc/letsencrypt/live/mafya.ml/privkey.pem; # managed by Certbot
}
server {
    if ($host = mafya.ml) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

        listen 80;
        server_name mafya.ml;
    return 404; # managed by Certbot
 }
```

- 다음 명령어를 입력합니다.

```
sudo ln -s /etc/nginx/sites-available/[파일명] /etc/nginx/sites-enabled/[파일명]
# 필자의 경우 mafya.conf

sudo nginx -t

sudo service restart nginx
```

6. 외부 서비스

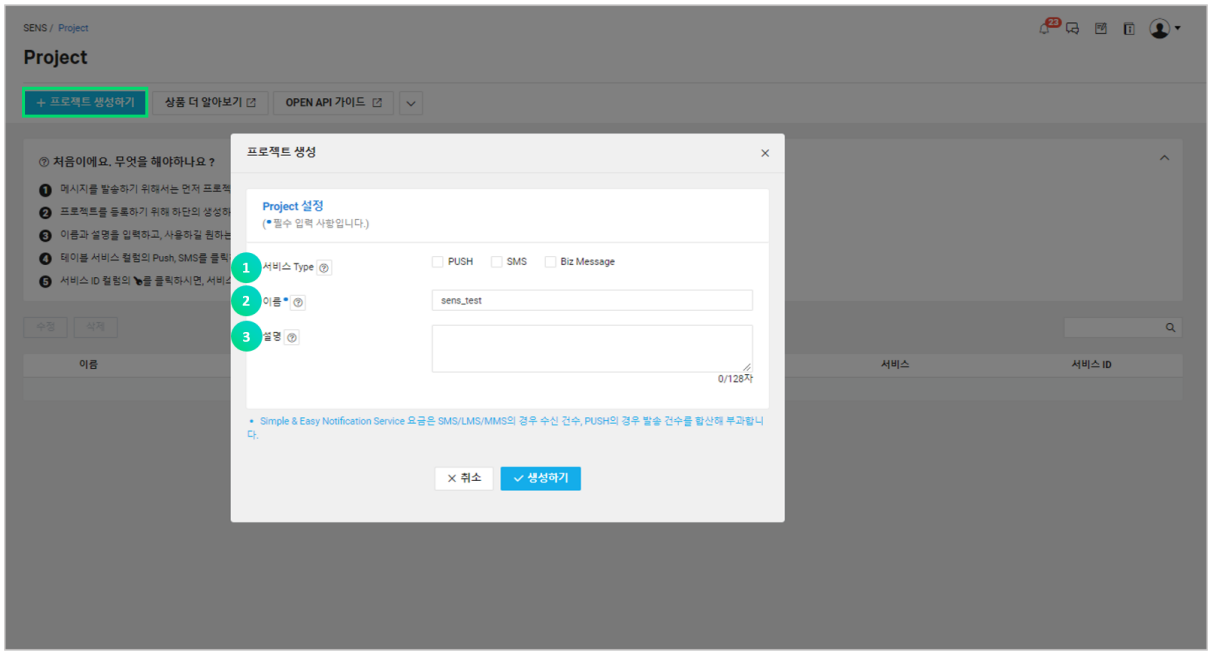
가. 네이버 Simple & Easy Notification Service

MMS 기능을 추가하기 위해 네이버 Cloud Platform의 **Simple & Easy Notification Service**를 사용했습니다.
(https://www.ncloud.com/product/applicationService/sens)

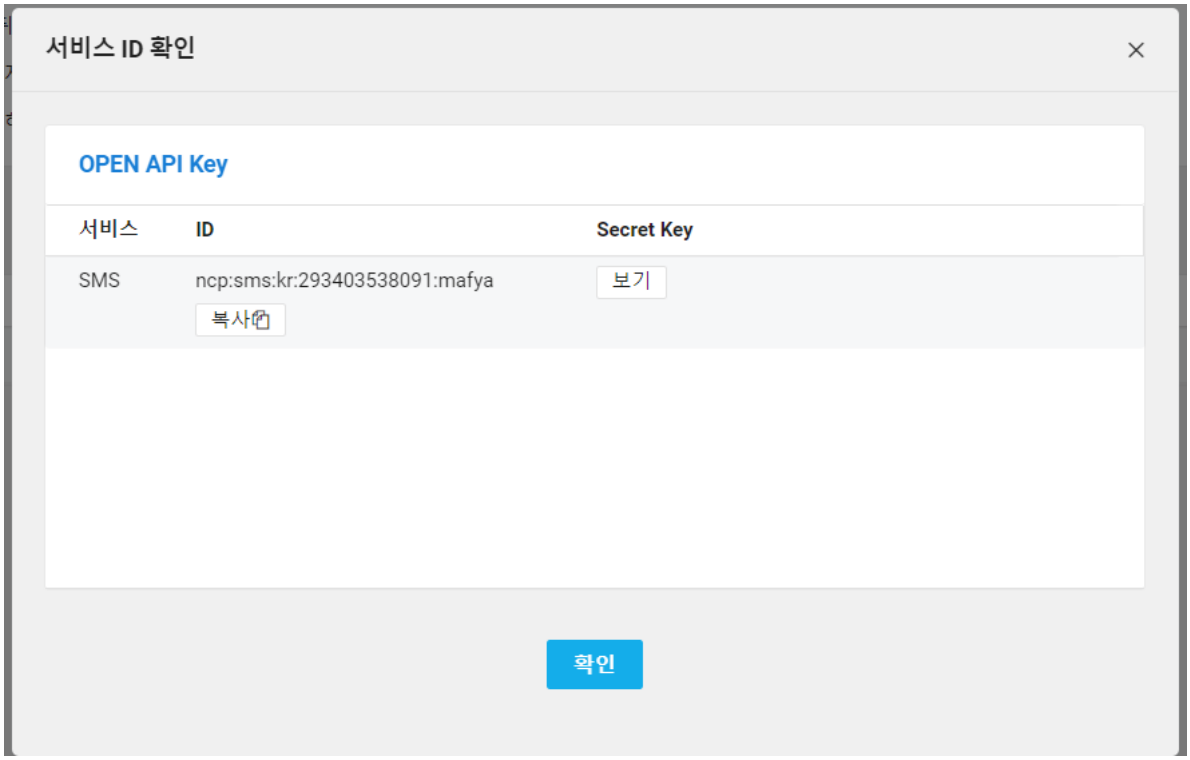
이를 통해 팀원에게 문자 메세지를 전송하는 서비스를 제공 할 수 있게 됩니다.

공식 가이드 문서(https://guide.ncloud-docs.com/docs/sens-sens-1-2)

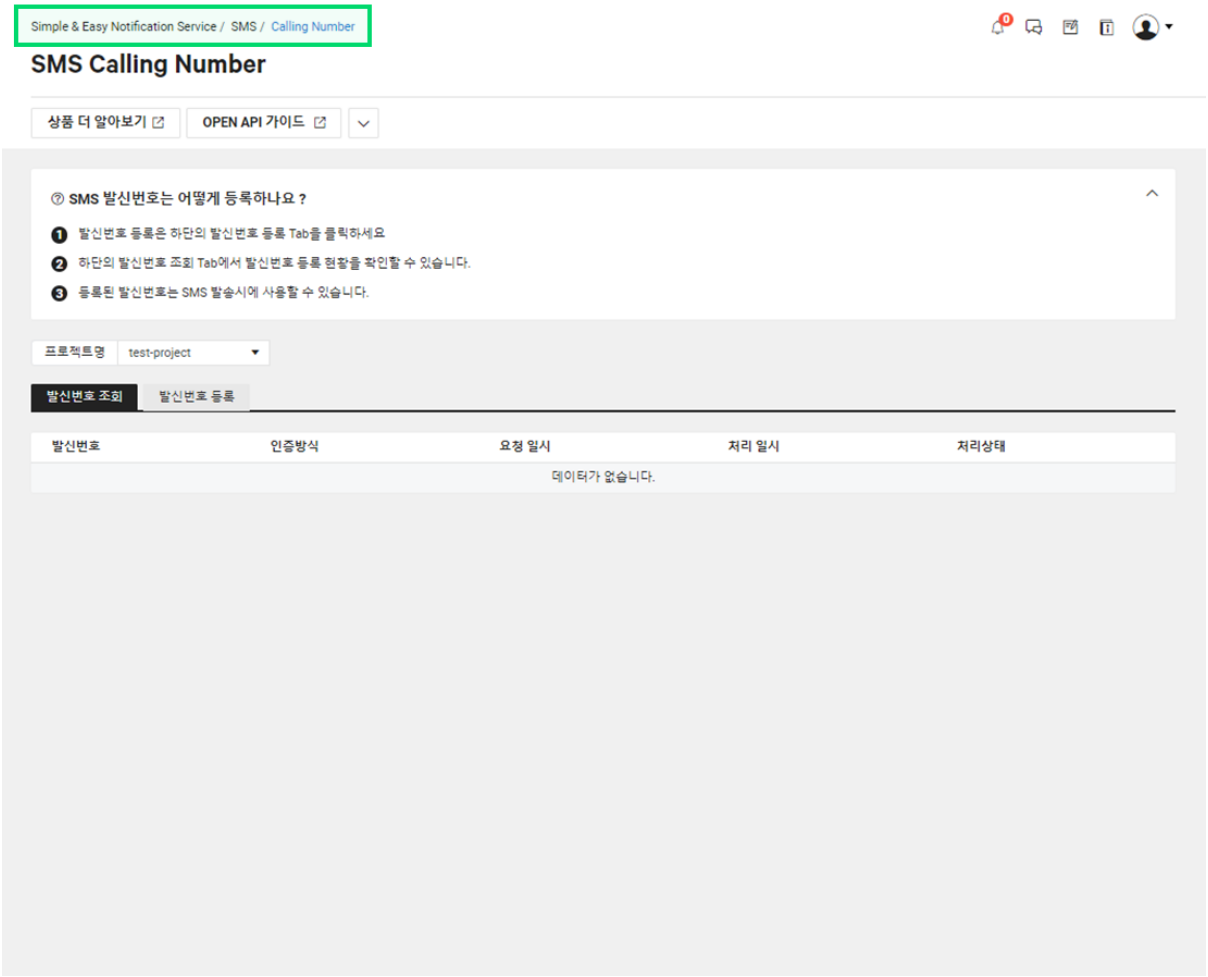
1. **Simple & Easy Notification Service 서비스를 사용하려면 먼저 프로젝트를 생성해야 합니다. 고객의 용도에 따라 여러 개의 프로젝트를 생성하고 사용할 수 있습니다**



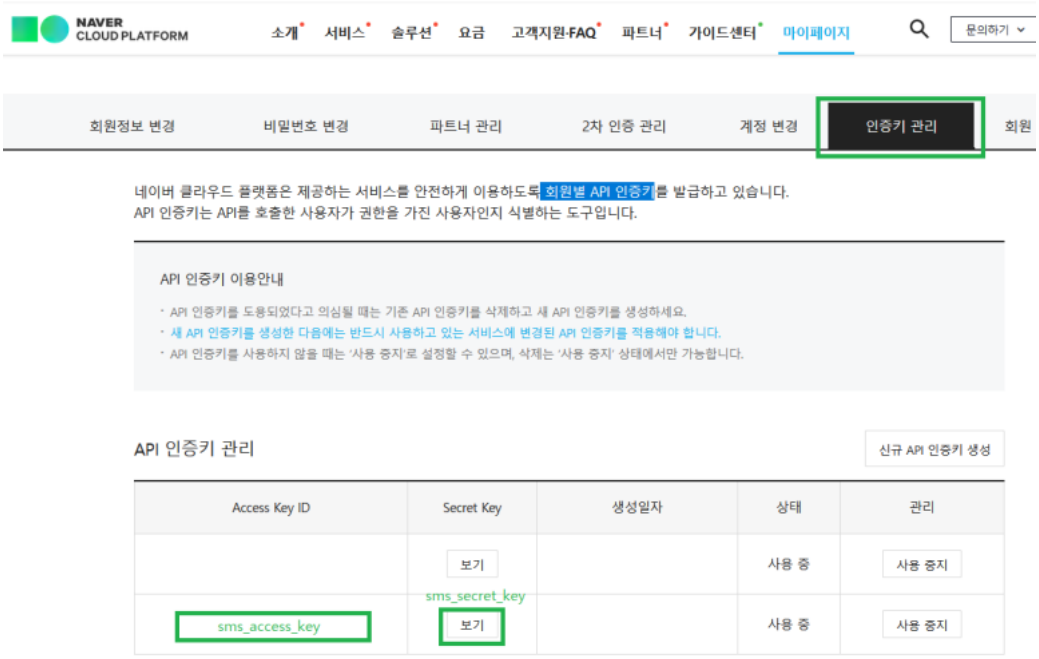
2. **생성한 프로젝트의 ServiceID & SecretKey를 확인합니다.**



3. **발신번호(senderPhoneNum) 등록하기**



4. **네이버 클라우드 플랫폼의 마이페이지>인증키 관리 에서 AccessKey를 확인합니다.**



5. **Spring project의 application.properties에 MMS 설정에 위의 확인한 키 값들을 넣어줍니다.**

```
# MMS
mms.accessKey = k1mxFBZqYBfMvaXzqtND
mms.secretKey =
mms.serviceId = ncp:sms:kr:293403538091:mafya
mms.senderPhoneNum =
```