포팅 메뉴얼

B209팀

1. 배포

도커 및 젠킨스

- 사전 패키지 설치

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

- gpg 키 다운로드

```
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
```

- docker 설치

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

- docker-compose 설치
- 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
```

이후 `sudo docker-compose up -d` 명령어로 컨테이너 생성

`sudo docker ps` 명령어로 컨테이너가 올라가있는 것을 확인

젠킨스 계정 생성 및 플러그인 설치

- 포트 IP:9090으로 접속시 젠킨스 시작화면
- `sudo docker logs jenkins` 명령어를 통해 password확인 후 Administrator password에 입력
- `Install suggested plugins` 를 클릭
- create first admin user 에 젠킨스 계정 생성 form 입력 후 `save and continue` . 이후 `start using jenkins`
- jenkins 관리탭에서 플러그인 관리 페이지로 이동
  - gitlab
    - GiLab
    - Generic Webhook Trigger
    - Gitlab API
    - GitLab Authentication
  - docker
    - Docker
    - Docker Commons
    - Docker Pipeline
    - DockerAPI

- SSH
  - Publish OverSSH

  위 플러그인 모두 설치

젠킨스 프로젝트 생성

- Gitlab 최상단 branch에 SpringProject 용, React Project용 두 폴더 구분하여 생성
- 젠킨스 메인 페이지에서 `새로운 item` 클릭
- 프로젝트 이름 입력 후 `Freestyle project` 선택
- 소스코드 관리 탭에서 None로 되어있는 것을 git 라디오 버튼을 클릭
- `Repository URL`에는 싸피깃 레포지토리 URL을 입력 (이때는 오류가 뜸)
- `Credentials`에서, `add` -> `jenkins` 클릭
  - Username : 싸피깃 아이디
  - Password : 싸피킷 비밀번호
  - ID : Credential 구별할 아무 텍스트 입력

  위 내용 입력하고 `Add`버튼 클릭
- `Credentials`에서 이제 만들어진 `Credential`을 선택했을 때 오류메시지가 사라지면 성공
- 빌드 유발 탭에서는 먼저 Build when a change is pushed to… 의 체크박스를 체크
  - push events
  - open merge request events
  - approved merge request(EE-only)
  - comments

  하위 체크박스 모두 체크
- 그 후 생기는 고급 버튼을 클릭. 스크롤을 내려 `Secret token`을 찾아 Generate 버튼을 누르면 토큰이 생성. 이 토큰은 Gitlab과 WebHook을 연결할 때 사용되니 저장해둘 것
- Build 탭으로 이동. `Add build step`를 클릭하고, `Execute Shell`을 선택.
- execute shell 란에 연결 테스트를 위해 간단히 pwd명령어 입력 후 저장. 자동으로 이동한 프로젝읕 화면에서 `지금 빌드` 버튼을 눌러서 젠킨스 수동 빌드를 진행. 초록 체크가 뜨며 완료 표시가 뜰 경우 성공한 것.
  - 빌드 히스토리에서, `Console Output` 클릭 후 입력한 명령어가 잘 작동한 것 확인

깃랩 WebHook 연결

싸피깃 레포지토리와 젠킨스를 `WebHook`
으로 연결하여 자동 빌드를 진행

- 배포할 프로젝트가 있는 깃랩 `Repository`에서 밑줄친 위치로 `WebHooks`페이지로 이동
- URL에는 `http://배포서버공인IP:9090/project/생성한jenkins프로젝트이름/`을 입력
- Secret token에는 아까 위에서 젠킨스 프로젝트를 생성할 때 저장해둔 값을 입력
- 빌드 유발 Trigger으로, `Push events, Merge request events`를 설정. 대상 Branch는 main으로 설정
- 완료했다면 Add Webhook 버튼을 눌러 webhook을 생성
- WebHook을 생성하고 나면 빌드 테스트를 위해 생성된 WebHook에서 test를 누르고, Push events를 선택. 200 응답을 확인할 것. 젠킨스에서도 정상적으로 빌드가 수행되는 것을 확인

이로써 Jenkins 와 Gitlab이 연결되어 Gitlab의 main branch에 이벤트 발생 시 젠킨스에서는 빌드를 수행

젠킨스와 연결된 gitlab 프로젝트로 도커 이미지 빌드하기

젠킨스에서 도커 빌드를 하기 위해서는 젠킨스 컨테이너 안에 도커를 설치해야 합니다. 도커 설치 방법은 Ec2에 도커를 설치할 때와 동일하게 진행

- `sudo docker exec -it jenkins bash`
  를 입력 후 젠킨스 bash shell에 접근.
- 사전 패키지 설치

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

- gpg키를 다운

```
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
```

- docker 설치

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

- 이후 깃랩 각 프로젝트 폴더에 DockerFile을 만들어 다음의 명령어 입력

FrontEnd Dockerfile

```
FROM node:16.16.0 as build-stage
WORKDIR /var/jenkins_home/workspace/GSDD/frontend/gsdd/
COPY package*.json ./
RUN npm install
COPY . .
RUN npm run build
FROM nginx:stable-alpine as production-stage

COPY --from=build-stage /var/jenkins_home/workspace/GSDD/frontend/gsdd/build /usr/share/nginx/html
COPY --from=build-stage /var/jenkins_home/workspace/GSDD/frontend/gsdd/deploy_conf/nginx.conf /etc/nginx/conf.d/default.conf

EXPOSE 80
CMD ["nginx", "-g","daemon off;"]
```

ngnix 설치를 위해 7번 코드는 주석처리하여 저장합니다

BackEnd Dockerfile

```
FROM openjdk:11

# set arg
ARG WORKSPACE=/var/jenkins_home/workspace/GSDD/backend/gsdd/
ARG BUILD_TARGET=${WORKSPACE}/build/libs
WORKDIR ${WORKSPACE}

# copy code & build
COPY . .
RUN ./gradlew clean bootJar

WORKDIR ${BUILD_TARGET}
RUN jar -xf *.jar
```

```
EXPOSE 8080
ENTRYPOINT ["java","org.springframework.boot.loader.JarLauncher"]
```

젠킨스에서 DockerFile이용 도커 이미지 생성

- 젠킨스 프로젝트 페이지에서, 구성 버튼을 클릭
- 적어두었던 pwd 명령어 대신 아래 명령어 입력 후 저장

```
docker image prune -a --force
mkdir -p /var/jenkins_home/images_tar
cd /var/jenkins_home/workspace/GSDD/frontend/gsdd/
docker build -t react .
docker save react > /var/jenkins_home/images_tar/react.tar

cd /var/jenkins_home/workspace/GSDD/backend/gsdd/
docker build -t springboot .
```

▼ 명령어에 대한 개별 설명

- docker image prune -a --force : 사용하지 않는 이미지 삭제
- mkdir -p /var/jenkins_home/images_tar : 도커 이미지 압축파일을 저장할 폴더 생성
- cd /var/jenkins_home/workspace/deploytest/testproject_react : 해당 경로로 이동(react 프로젝트 폴더)
- docker build -t react . : 도커 이미지 빌드(React 프로젝트)
- docker save react > /var/jenkins_home/images_tar/react.tar : 도커 이미지를 react.tar로 압축하여 위에서 생성한 폴더에 저장
- cd /var/jenkins_home/workspace/deploytest/testproject/ : 해당 경로로 이동(django 프로젝트 폴더)
- docker build -t django . : 도커 이미지 빌드(Django 프로젝트)
- docker save django > /var/jenkins_home/images_tar/django.tar : 도커 이미지를 django.tar로 압축하여 위에서 생성한 폴더에 저장
- ls /var/jenkins_home/images_tar : 해당 폴더에 있는 파일 목록 출력(잘 압축되어 저장되었는지 확인)

- 젠킨스의 빌드 성공 확인 후 젠킨스 컨테이너 의 `/var/jenkins_home/images_tar`
  폴더 안에 2개의 tar 파일이 생성되어있고, 폴더를 공유하는 EC2의 `/jenkins/images_tar`에도 똑같이 2개의 tar 파일이 생성되어 있는 것을 확인

젠킨스에서 SSH 명령어 전송을 통해 빌드한 도커 이미지를 베이스로 컨테이너 생성(기본 배포 완료)

- 젠킨스에서 AWS으로 SSH 명령어를 전송하려면 AWS 인증 키(EC2 생성할 때 사용한 pem 파일)를 등록해주어야 함.
- 젠킨스 홈페이지에서 `Jenkins 관리`
  를 클릭하고, 이어서 `시스템 설정`
  을 클릭
- 시스템 설정 칸에서 스크롤을 아래로 쭉 내리면 `Public over SSH` 항목이 있습니다. 여기서 `SSH Servers`추가 버튼 클릭
  - Name : 그냥 이름
  - Hostname : EC2 IP
  - Username : EC2 접속 계정 이름
- 고급 버튼을 클릭. 다른 건 건드리지 않고, `Use password authentication, or use different key`체크박스를 체크
  - EC2에서 생성했던 키 페어 pem 파일(싸피에서 받았을 경우 싸피에서 받은 pem 파일)을 VSCode로 오픈
  - Pem 파일은 다음과 같은 구성으로 되어있습니다. 이 텍스트 내용을 전체 복사하여 복붙
  - 이후 `Test Configuration`버튼을 눌렀을 때 Success가 나오면 성공
  - 빌드 후 조치, `Send build artifacts over SSH`를 선택

- `Source files`는 컨테이너에서 aws로 파일을 전송하는 부분인데, 의미가 없는데도 필수 입력 사항이기 때문에 적당히 아무거나 적어줌
- 중요한 부분은 `Exec command`부분에 아래 명령어 입력

```
sudo docker load < /jenkins/images_tar/react.tar
sudo docker load < /jenkins/images_tar/springboot.tar

if (sudo docker ps | grep "react"); then sudo docker stop react; fi
if (sudo docker ps | grep "springboot"); then sudo docker stop springboot; fi

sudo docker run -it -d --rm -v /etc/letsencrypt:/etc/letsencrypt -p 80:80 -p 443:443 --name react react

echo "Run frontend"

sudo docker run -it -d --rm -p 8080:8080  --name springboot springboot

echo "Run backend"
```

빌드 후 80은 프론트, 8080은 백으로 서비스되는것 확인

Nginx를 통해 React와 SpringBoot 경로 설정

- EC2 Ubuntu 콘솔에서 `cd /jenkins/workspace/deploytest/testproject_react`명령으로 디렉토리를 이동
- 이후 `sudo mkdir deploy_conf`명령어로 디렉토리를 생성하고 `cd deploy_conf`
  를 이용해 이동. 그 후 `sudo vim nginx.conf`명령어로 `nginx.conf`파일을 생성하고 편집기로 이동하여 아래 코드 입력

nginx 설정 파일

```
default.conf

upstream backend{
  ip_hash;
  server j7b209.p.ssafy.io:8080;
}

server {
  listen 80;
  server_name j7b209.p.ssafy.io;

  location / {
    return 301 https://$host$request_uri;
  }
}

server {
    listen  443 ssl;
    server_name  j7b209.p.ssafy.io;

    ssl_certificate /etc/letsencrypt/live/j7b209.p.ssafy.io/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/j7b209.p.ssafy.io/privkey.pem;


    #access_log  /var/log/nginx/host.access.log  main;

    location / {
        root   /usr/share/nginx/html;
        index  index.html index.htm;
        try_files $uri $uri/ /index.html $uri.html = 404;
        proxy_redirect off;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header   Host $host;
        proxy_set_header   X-Real_IP  $remote_addr;
        proxy_set_header   X-Forwarded-Host  $server_name;
        proxy_set_header   X-Forwarded-For   $proxy_add_x_forwarded_for;
    }

    location /api {
        proxy_pass http://backend;
        proxy_redirect     off;
        proxy_set_header   X-Forwarded-Proto $scheme;
```

```
        proxy_set_header   Host $host;
        proxy_set_header   X-Real-IP $remote_addr;
        proxy_set_header   X-Forwarded-Host $server_name;
        proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
    }


    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
        root   /usr/share/nginx/html;
    }

}
```

이후 react 프로젝트 도커파일에 있는 주석 모두 제거

도커파일이 수정사항을 반영시키려면, gitlab에 Push 해주어야함! 최종적으로 모든 기능이 잘 작동하는지 테스트

Https 설정

letsencrypt 설치

```
sudo apt-get install letsencrypt

sudo systemctl stop nginx

sudo letsencrypt certonly --standalone -d 도메인 이름
```

springboot https 적용

- 인증서 → PCKS12형식으로 변환
- password 설정

```
.pem 파일이 위치한 경로에서 진행
openssl pkcs12 -export -in fullchain.pem -inkey privkey.pem
-out keystore.p12 -name tomcat -CAfile chain.pem -caname root
```

- keystore.p12 파일을 /src/main/resources에 이동

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

2. crontab 설정

cron 설치

sudo apt update -y
sudo apt install -y cron

cron 시작

sudo service cron start

cron systemctl 활성화

sudo systemctl enable cron.service

cron systemctl 등록 확인

sudo systemctl list-unit-files | grep cron
sudo service cron status

Crontab 편집

crontab -e

Crontab List 조회

crontab -l

Crontab List 전체 삭제

crontab -r

Crontab실행 코드

crontab -e
#m h dom mon dow command
[0~59분] [0~23시] [1~31일] [1~12월] [0(월)~6(토)요일]
0 15 * * * /home/ubuntu/news/gsdd.sh >> /home/ubuntu/news/gsdd_cron.log 2>&1

3. SSH Key

SSH key

- 서버에 접속할 때 비밀번호 대신 key 제출
- 비밀번호 보다 높은 수준의 보안
- 로그인 없이 자동으로 서버 접속

SSH key 만들기

키 생성

ssh-keygen

키 확인

ls -al ~/.ssh/

- id_rsa : private key , 타인에게 노출되면 안된다.
- id_rsa.pub : 접속하려는 리모트 머신의 authorized_keys에 입력
- authorized_keys : 리모트 머신의 .ssh 디렉토리 아래에 위치하면서 id_rsa.pub 키의 값을 저장한다.

실행 코드

scp $HOME/.ssh/id_rsa 리모트서버 아이디@호스트주소:저장할파일

cat $HOME/id_rsa.pub >> $HOME/.ssh/authorized_keys

SSH 이용해서 접속
ssh 리모트서버 아이디@호스트주소

id_rsa 파일의 위치가 다를때 -i 옵션 사용
(접속 오류시 -v 옵션 사용하면 어디에 문제가 발생했는지 추적)
ssh -i (path) 리모트서버 아이디@호스트주소

4. Hadoop

1. ant 명령어 사용하여 빌드

```
hadoop@hadoop-virtual-machine:~/gsdd$ ant
Buildfile: /home/hadoop/gsdd/build.xml

init:

compile-works:
    [javac] Compiling 6 source files to /home/hadoop/gsdd/build/works
    [javac] Note: Some input files use or override a deprecated API.
    [javac] Note: Recompile with -Xlint:deprecation for details.

ssafy-works:
      [jar] Building jar: /home/hadoop/gsdd/build/ssafy.jar

package:
     [copy] Copying 1 file to /home/hadoop/gsdd

BUILD SUCCESSFUL
Total time: 5 seconds
```

2. crontab , shell script 활용하여 하둡 실행

```
#!/bin/bash

cd /home/ubuntu/news
java -jar news.jar
scp news.txt j7b209@cluster.ssafy.io:/home/j7b209
ssh j7b209@cluster.ssafy.io hdfs dfs -rm news.txt
ssh j7b209@cluster.ssafy.io hdfs dfs -put news.txt
ssh j7b209@cluster.ssafy.io hadoop jar gsdd.jar article news.txt news_output
ssh j7b209@cluster.ssafy.io hdfs dfs -cat news_output/job2/* > DBarticle.txt
java -jar /home/ubuntu/news/db.jar
~
```

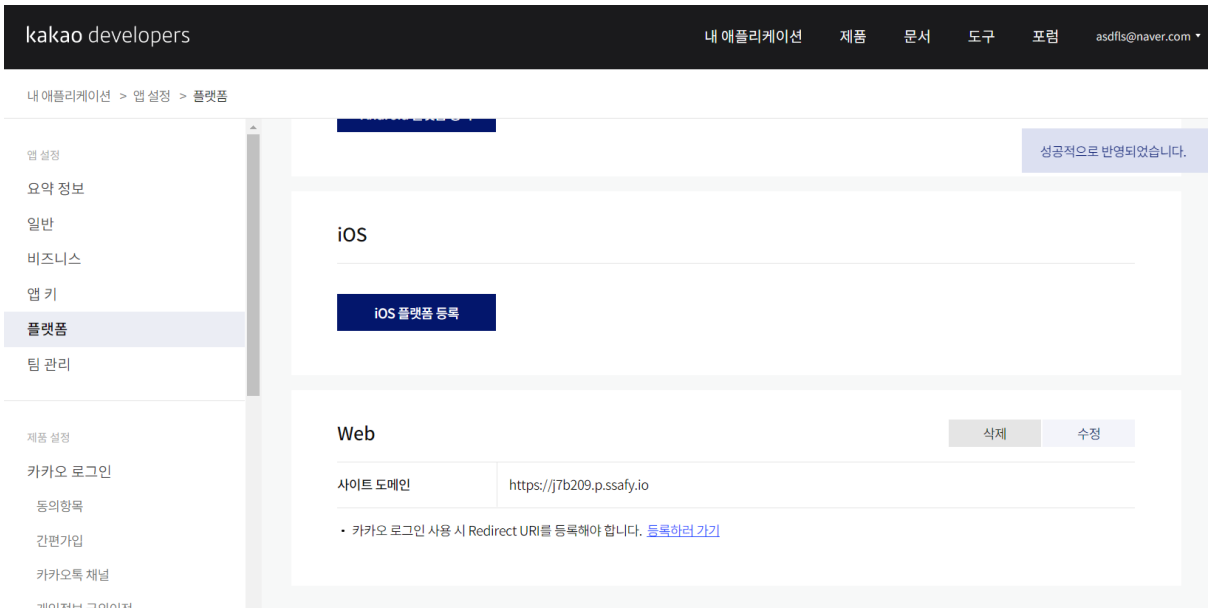
```
  GNU nano 4.8                                                    /tmp/cr
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
0 15 * * * /home/ubuntu/news/gsdd.sh >> /home/ubuntu/news/gsdd_cron.log 2>&1
```

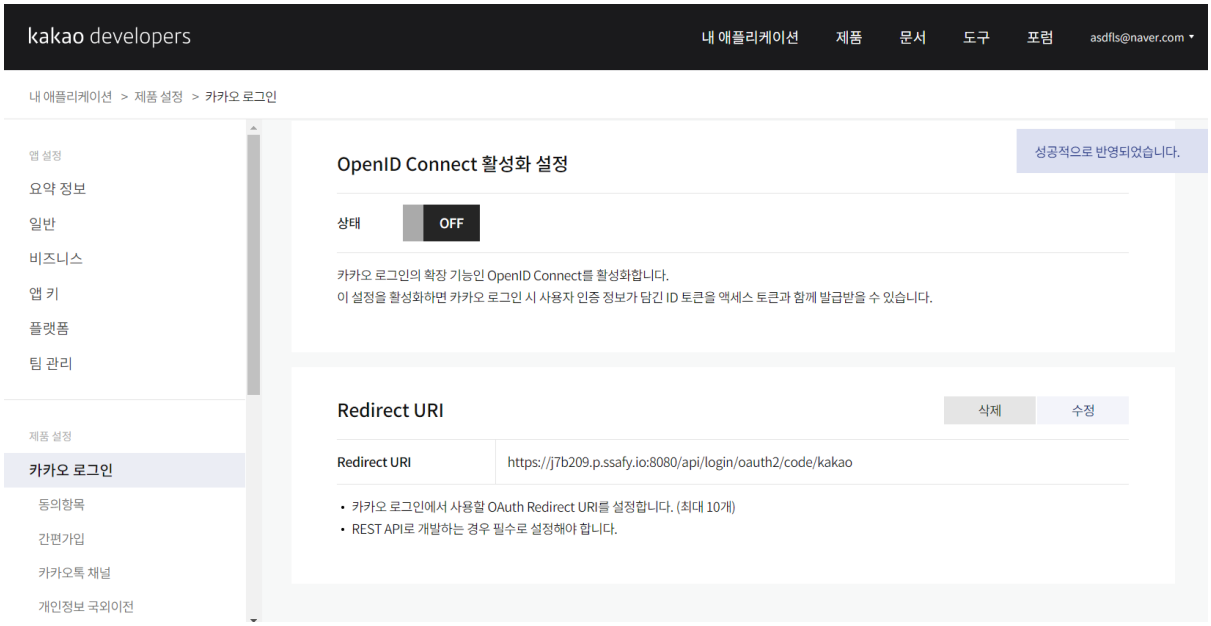
3. 실행 결과

newsId	title	url	keyword	area
1	"검찰청 불이 꺼진다"...평검사대표...	https://n.news.naver.com/mnew...	범죄,,	대전,,
2	"내 인생 망치고 싶냐" 신당역 스토...	https://n.news.naver.com/mnew...	살인,폭행,살해,,	중구,,
3	"너무 무서워" 마약 후 엄마에 전화...	https://n.news.naver.com/mnew...	범죄,,	서구,,
4	"다시 만나자"...전 여친 흉기 위협·...	https://n.news.naver.com/mnew...	폭행,강간,,	중구,,
5	"다시 사귀자"...헤어진 연인 찾아...	https://n.news.naver.com/mnew...	폭행,강간,,	중구,,
6	"다시 친구하자" 제안했다 거부당...	https://n.news.naver.com/mnew...	폭행,범죄,,	서구,,
7	"마스크 없이 못타" 승객 다치게 한...	https://n.news.naver.com/mnew...	범죄,,	동구,,
8	"미친 짓 했다, 죄송하다"...'신당역 ...	https://n.news.naver.com/mnew...	살해,범죄,살인,,	중구,,
9	"불질러 버릴거야" 장기투숙 모텔 ...	https://n.news.naver.com/mnew...	범죄,폭행,,	동구,,
10	"비상벨 왜 눌러"... 공주교도소서 ...	https://n.news.naver.com/mnew...	폭행,,	대전,,
11	"수괴가 학살 부인" 2심도 전두환 ...	https://n.news.naver.com/mnew...	살인,범죄,,	동구,,
12	"수사기관 사칭"...보이스피싱 현금...	https://n.news.naver.com/mnew...	범죄,,	대전,,
13	"스토킹 살인사건" 정계·시민사회 ...	https://n.news.naver.com/mnew...	살인,범죄,,	중구,,

5. 소셜로그인 설정

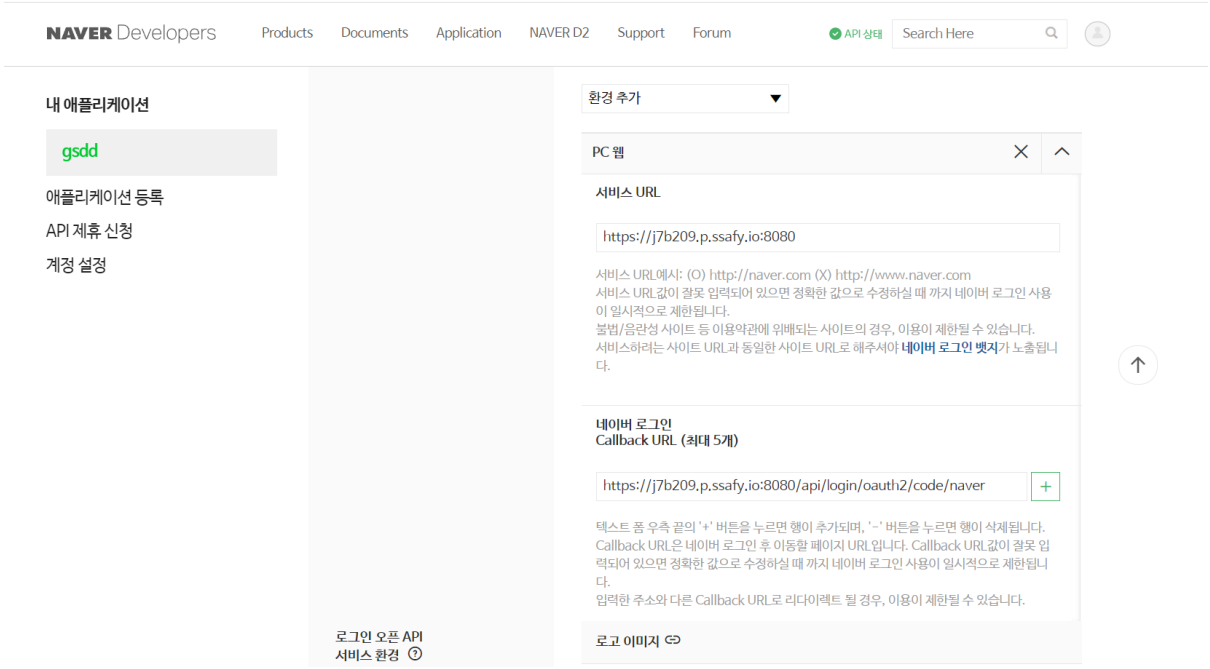
카카오 로그인





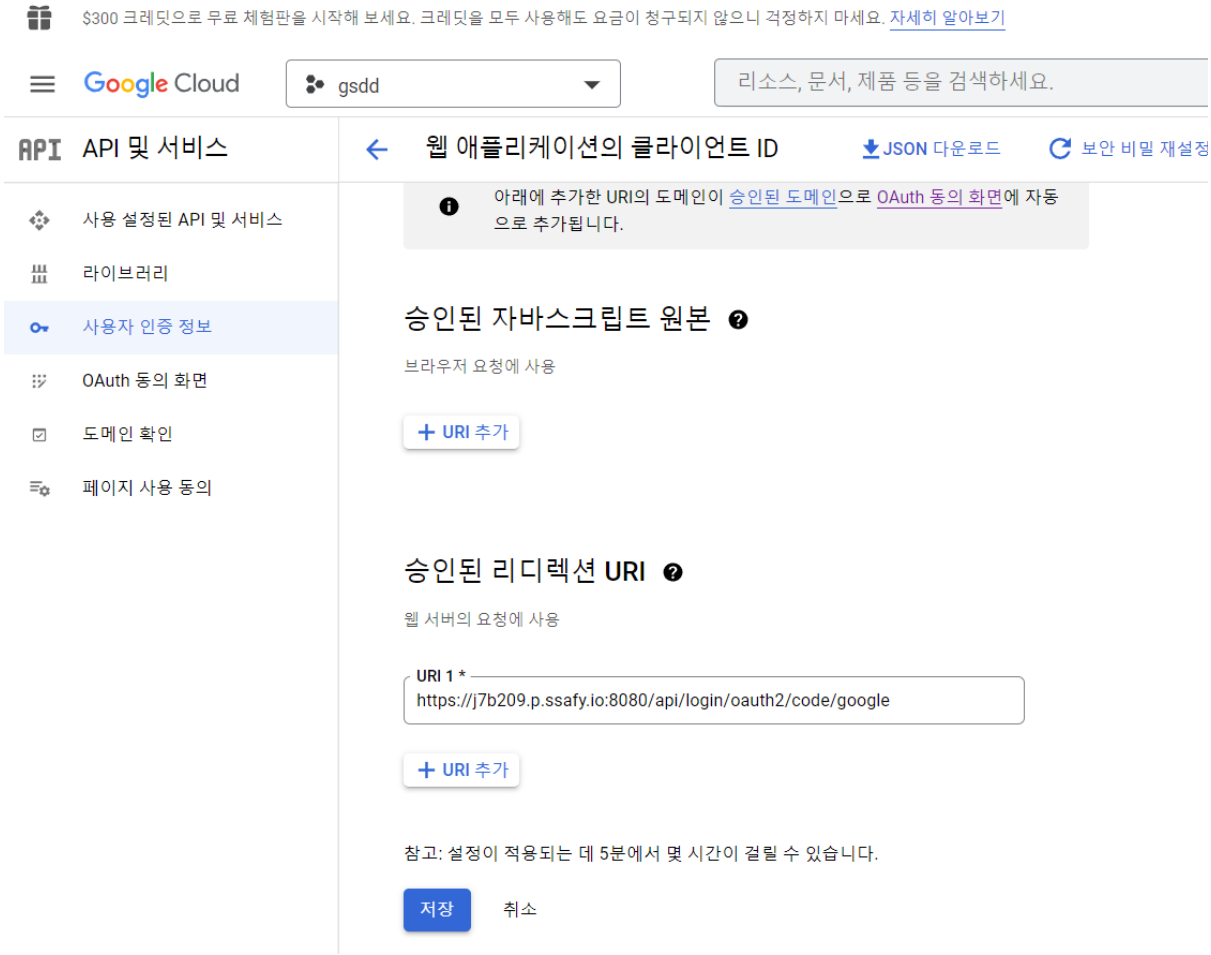
도메인과 REdirect URI 설정

네이버



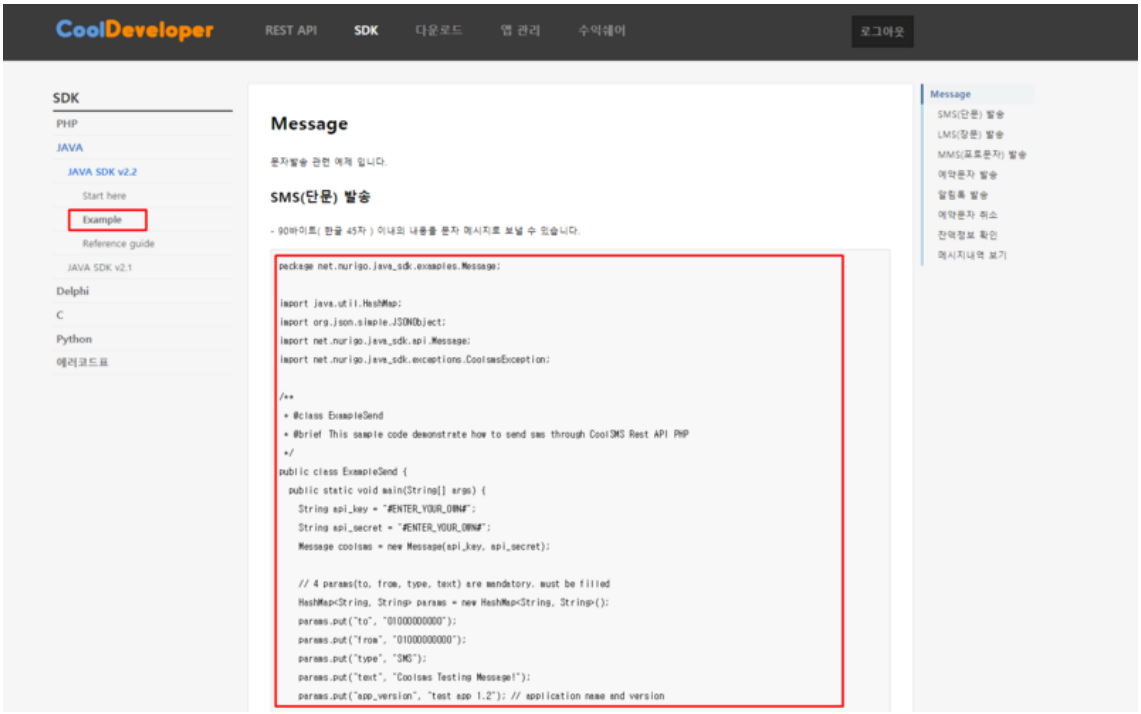
URL과 CALLback URL 설정

구글



URL 등록

CoolSMS




```
import java.util.HashMap;
import org.json.simple.JSONObject;
import net.nurigo.java_sdk.api.Message;
import net.nurigo.java_sdk.exceptions.CoolsmsException;

/**
 * @class ExampleSend
 * @brief This sample code demonstrate how to send sms through CoolSMS Rest API PHP
 */
public class ExampleSend {
  public static void main(String[] args) {
    String api_key = "키입력";
    String api_secret = "시크릿키입력";
    Message coolsms = new Message(api_key, api_secret);

    // 4 params(to, from, type, text) are mandatory. must be filled
    HashMap<String, String> params = new HashMap<String, String>();
    params.put("to", "발신번호");
    params.put("from", "수신번호"); //무조건 자기번호 (인증)
    params.put("type", "SMS");
    params.put("text", "보낼 메시지를 입력하시오");
    params.put("app_version", "test app 1.2"); // application name and version

    try {
      //send() 는 메시지를 보내는 함수
      JSONObject obj = (JSONObject) coolsms.send(params);
      System.out.println(obj.toString());
    } catch (CoolsmsException e) {
      System.out.println(e.getMessage());
      System.out.println(e.getCode());
    }
  }
}
```