포팅 메뉴얼

1. 개발환경

1.1. Frontend

- React
  - Redux
  - Redux-persist
- Typescript

1.2 Backend

- Java
  - Java OpenJDK 11
  - Spring Book 2.7.15
    - Spring Data JPA
    - Spring Security
    - Lombok
    - JWT
    - Oauth 2.0
  - Gradle {version}

1.3 Server

- Ubuntu
- Nginx
- Docker
- Docker Compose
- Jenkins

1.4 Database

- MySQL
- Redis

- Amazon S3

1.5 UI/UX

- Figma, Canva

1.6 IDE

- Visual Studio Code
- IntelliJ IDEA

1.7 형상 / 이슈관리

- Gitlab
- Jira

1.8 기타 툴

- Postman
- Mattermost
- Notion

2. EC2 세팅

2.1 Docker 설치

```
### install-docker.sh

# update apt packages
sudo apt-get update

# install docker engine
sudo apt-get install -y docker docker-io

# test
docker ps
```

2.2 Nginx 설치

```
# 1. Nginx 설치
sudo apt-get install nginx

# 2. start
sudo systemctl status nginx
sudo systemctl start nginx

nginx -v
# 2. Let's Encrypt 설치 및 SSL 발급
sudo apt-get install letsencrypt
sudo systemctl stop nginx
sudo letsencrypt certonly --standalone -d 도메인명

# 3. Nginx 설정파일 생성
cd /etc/nginx/sites-available
vi configure
```

2.3 SSL 적용

```
sudo apt-get update

# letsencrypt 설치
sudo apt-get install letsencrypt

# ssl 인증서 생성
sudo certbot certonly --standalone

# 인증서 생성하면서, 이메일과, EC2 도메인 입력!

# 인증서 생성 후, /etc/letsencrypt/live/{도메인}에 인증서 파일 확인!
```

```
# /etc/nginx/site-available/default 파일 수정

##
# 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.
##

# Default server configuration
#

server {

        listen [::]:443 ssl ipv6only=on; # managed by Certbot
        listen 443 ssl; # managed by Certbot

        ssl_certificate /etc/letsencrypt/live/j9b302.p.ssafy.io/fullchain.pem;
        ssl_certificate_key /etc/letsencrypt/live/j9b302.p.ssafy.io/privkey.pem;

        server_name j9b302.p.ssafy.io;

        location / {
            proxy_pass http://localhost:3000;
        }
        location /api {
                proxy_pass http://localhost:8080;
        }
        location /login {
                proxy_pass http://localhost:8080;
        }
        location /oauth2 {
                proxy_pass http://localhost:8080;
        }
        location /jenkins {
                return 301 http://j9b302.p.ssafy.io:32773;
        }

}

server {
        if ($host = j9b302.p.ssafy.io) {
            return 301 https://$host$request_uri;
        }

        listen 80;
        listen [::]:80;

        return 404;
}
```

2.4 EC2 Port

EC2 1 (Front, API-Server)

EC2 2 (Live-Server)

Port 번호	내용
22	SSH
80	HTTP(HTTPS로 redirect)
443	HTTPS
3000	Nginx, React (Docker)
7897	Redis (JWT Token)
8080	Spring Boot (Docker)
32773	Jenkins

Port 번호	내용
22	SSH
80	HTTP(HTTPS로 redirect)
443	HTTPS
8080	Spring Boot (Websocket-Server)
8081	Spring Boot (Live Queue-Server)
9092	kafka-1
29092	kafka-2
12181	zookeeper-1
22181	zookeeper-2
32181	zookeeper-3
32773	jenkins

2.4 방화벽 설정

```
# EC2 1 (Front, API-Server) 방화벽 확인
sudo ufw status

# 1. 해당 포트 개방
# 22 TCP
# 80 TCP

sudo ufw allow 22
sudo ufw allow 80

# Firewall 활성화 상태 확인
sudo ufw enable
sudo ufw status verbose

# 3. Nginx reverse proxy 설정 후 Frontend, Backend, Jenkins 서버 포트 닫기
sudo ufw deny 3000/tcp # React Nginx
sudo ufw deny 8080/tcp # Spring Boot
sudo ufw deny 32773/tcp # Jenkins

# EC2 2 (Live-Server) 방화벽 확인
sudo ufw status

# 1. 해당 포트 개방
# 22 TCP
# 80 TCP
```

```
sudo ufw allow 22
sudo ufw allow 80

# Firewall 활성화 상태 확인
sudo ufw enable
sudo ufw status verbose

# 3. Nginx reverse proxy 설정 후 websocket-server, liveQueue-server, Jenkins 서버 포트 닫기
sudo ufw deny 8080/tcp # websocket-server
sudo ufw deny 8081/tcp # LiveQueue-Server
sudo ufw deny 32773/tcp # Jenkins
```

4. CI/CD 구축

4.1. Jenkins 도커 이미지 + 컨테이너 생성

```
# jenkins 실행
sudo docker run --name jenkins -d -p 32773:8080 -p 50000:50000 -v /var/jenkins_home:/v
a
r/jenkins_home -v /var/run/docker.sock:/var/run/docker.sock -u root jenkins/jenkins

# jenkins 접속
sudo docker exec -it jenkins bash

# jenkins에 docker 설치
apt-get update
apt-get install docker docker.io

# docker 확인
docker ps
```

4.2. Jenkins 설정

[CI/CD] Gitlab과 Jenkins로 CI/CD 구축하기

Gitlab과 Jenkins로 CI/CD 구축하기

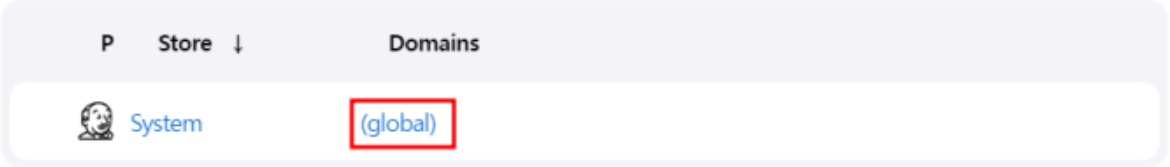
velog

https://velog.io/@hanif/Gitlab과-Jenkins로-CICD-구축하기

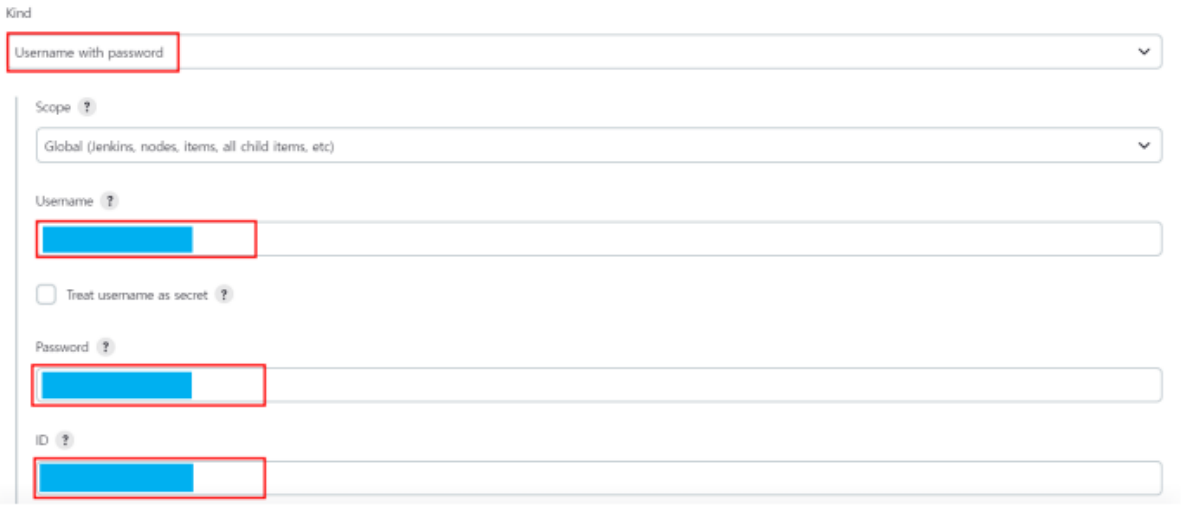
4.2.1 GitLab Credentials 설정

1. jenkins 관리 → “Credentials” 클릭
2. “Store : System” → “(global)” → “+ Add Credentials” 클릭

Stores scoped to Jenkins



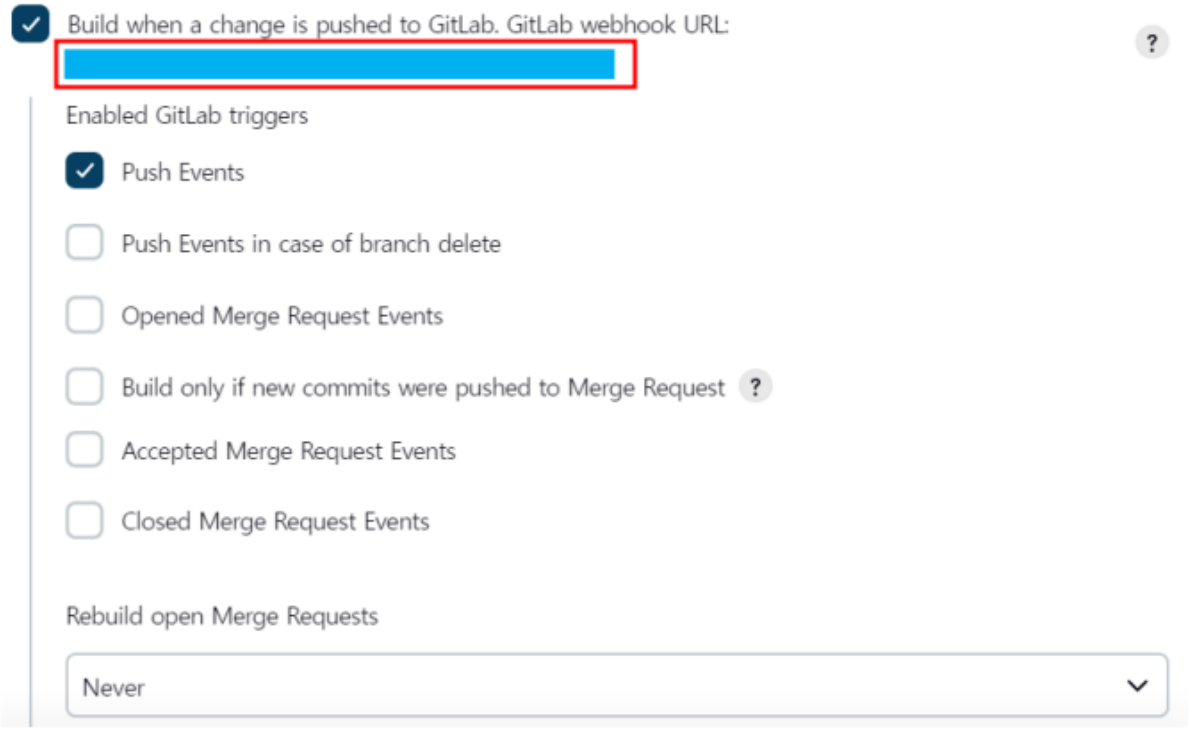
3. “Username with password” 입력 → “Username” 에 GitLab ID 입력 → “password에 Gitlab Personal Access Tokens 입력” → “ID”에 임의 아이디 입력 → 생성



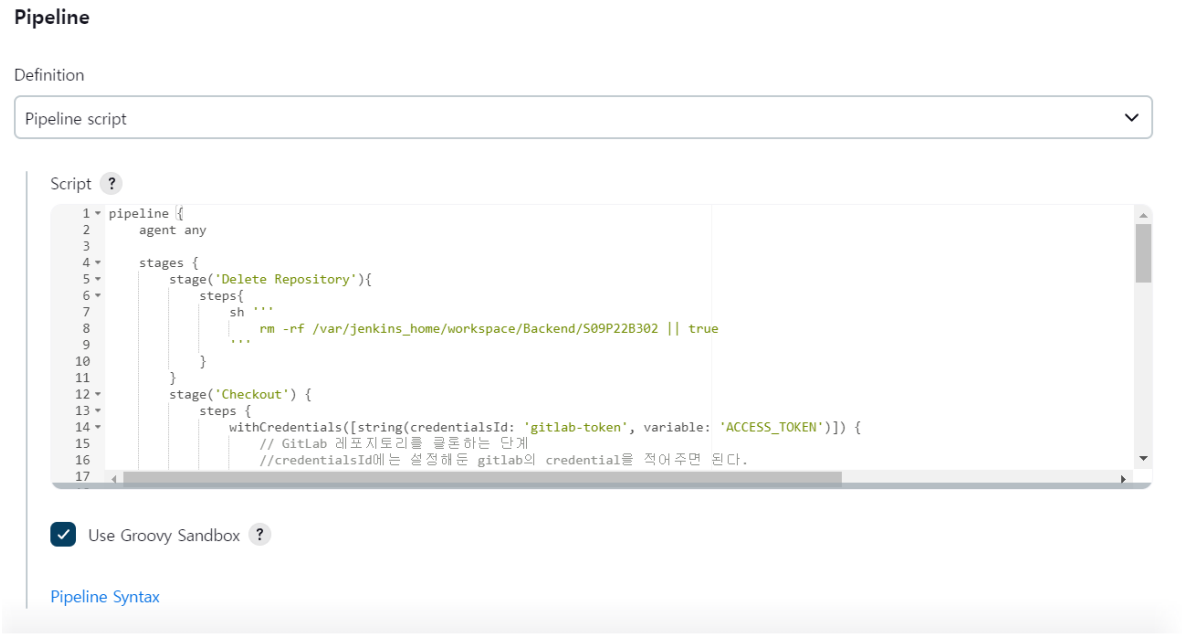
4.2.1 Jenkins item 생성

1. item 생성
2. “Enter an item name” 에 임의 item 이름 입력 → Plpeline 클릭

3. “Build Triggers” → “Build when a change is pushed to GitLab” 클릭 (WebHook 설정: GitLab 특정 브랜치 push 시 자동 빌드 → 배포 설정)



4. pipeline 작성



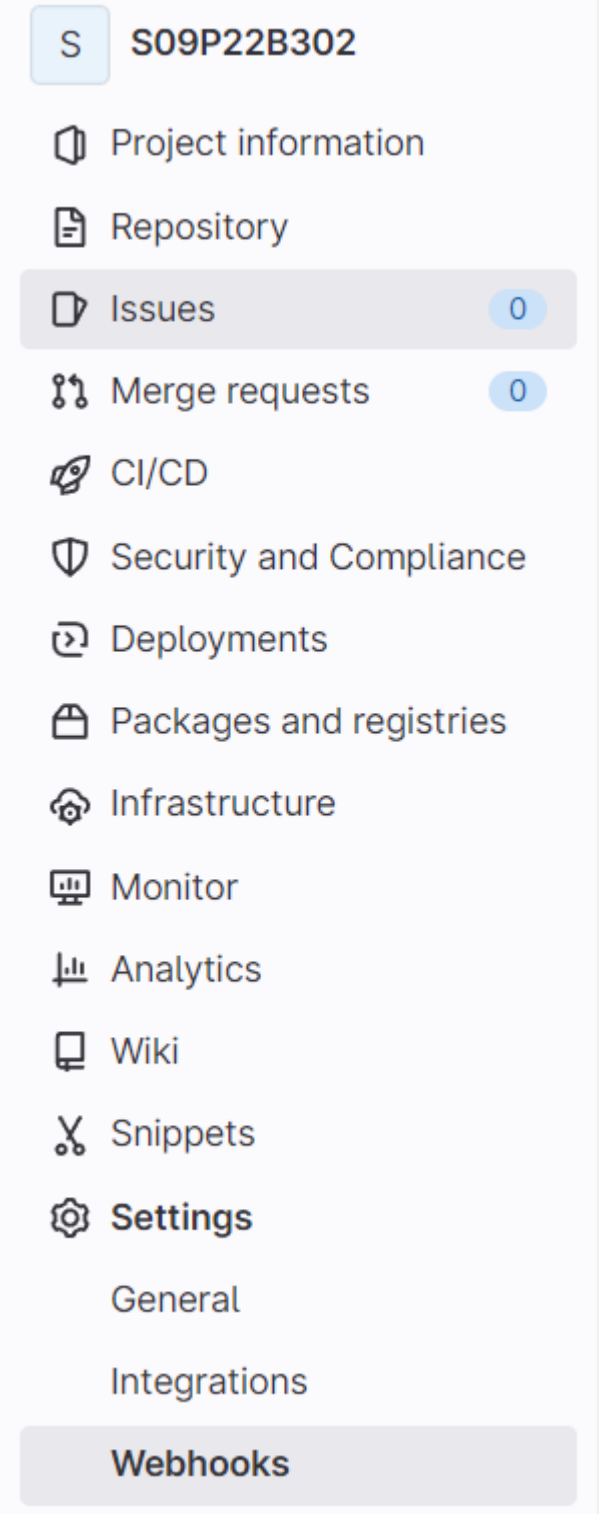

```
pipeline {
    agent any

    stages {
        stage('Delete Repository'){
            steps{
                sh '''
                    rm -rf /var/jenkins_home/workspace/Backend/S09P22B302 || true
                '''
            }
        }
        stage('Checkout') {
            steps {
                withCredentials([string(credentialsId: 'gitlab-token', variable: 'ACCE
SS_TOKEN')]) {
                    // GitLab 레포지토리를 클론하는 단계
                    //credentialsId에는 설정해둔 gitlab의 credential을 적어주면 된다.
                    sh '''
                        git clone -b backend https://gitlab-ci-token:${ACCESS_TOKEN}@l
ab.ssafy.com/s09-bigdata-dist-sub2/S09P22B302.git
                        cp /var/jenkins_home/workspace/secret/application.yml /var/jen
kins_home/workspace/Backend/S09P22B302/zizon/src/main/resources/
                        cp /var/jenkins_home/workspace/secret/application-oauth2.yml /
var/jenkins_home/workspace/Backend/S09P22B302/zizon/src/main/resources/
                        cp /var/jenkins_home/workspace/secret/application-connection.y
ml /var/jenkins_home/workspace/Backend/S09P22B302/zizon/src/main/resour ces/
                        cp /var/jenkins_home/workspace/secret/application-app.yml /va
r/jenkins_home/workspace/Backend/S09P22B302/zizon/src/main/resources
                    '''
                    sh 'pwd'
                }
            }
```

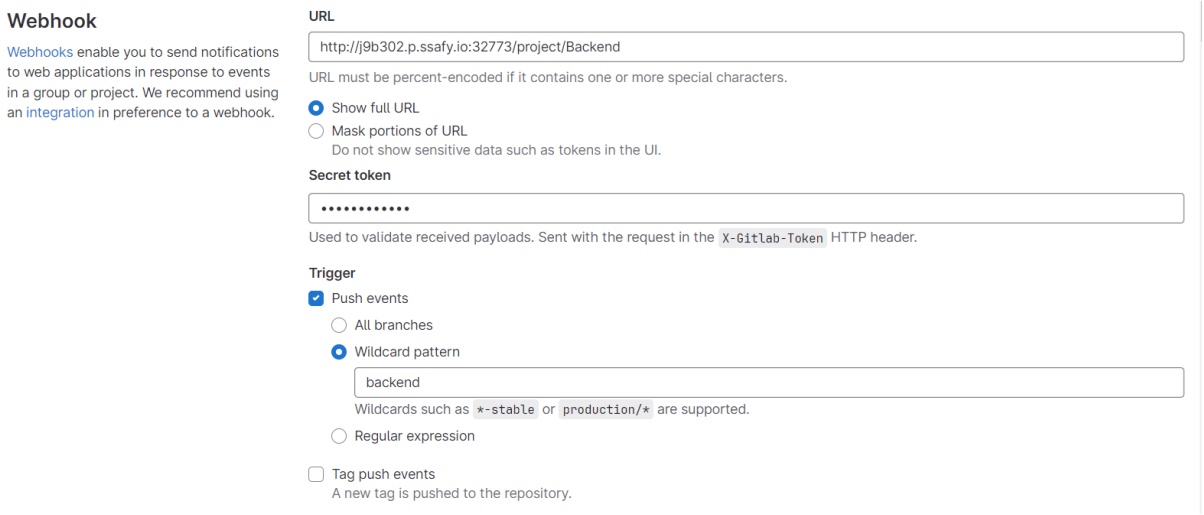
```
        }
        stage('Build Jar'){
            tools {
                // Gradle을 사용할 수 있도록 설정합니다.
                //위에서 설정한 Gradle 이름
                gradle 'Gradle'
            }
            steps {
                // Gradle을 실행하여 Jar 빌드를 수행합니다.
                //프로젝트에서 gradlew가 있는 위치로 이동
                sh '''
                    cd /var/jenkins_home/workspace/Backend/S09P22B302/zizon
                    chmod +x gradlew
                    ./gradlew build
                 '''
            }
        }
    }
}
```

4.2.3. GitLab Webhook 설정

1. 프로젝트 GitLab → setting → webhook 클릭



2. jenkins URL 입력 → Secret token 입력 → push events 클릭

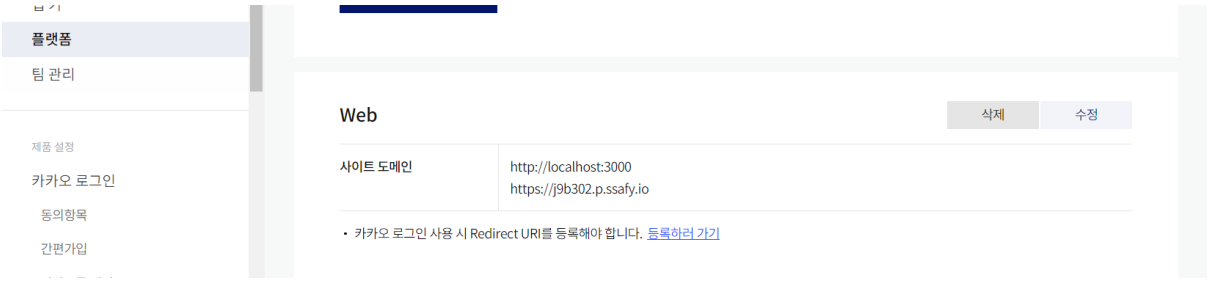


5. 외부서비스

5.1. 소셜로그인 - Kakao

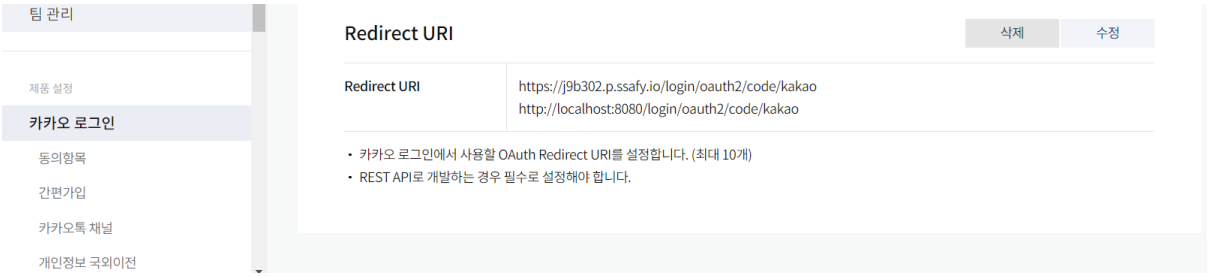
카카오 디벨로퍼

1. 사이트도메인 등록



2. 카카오 로그인 활성화

3. 리다이렉트 URI 지정

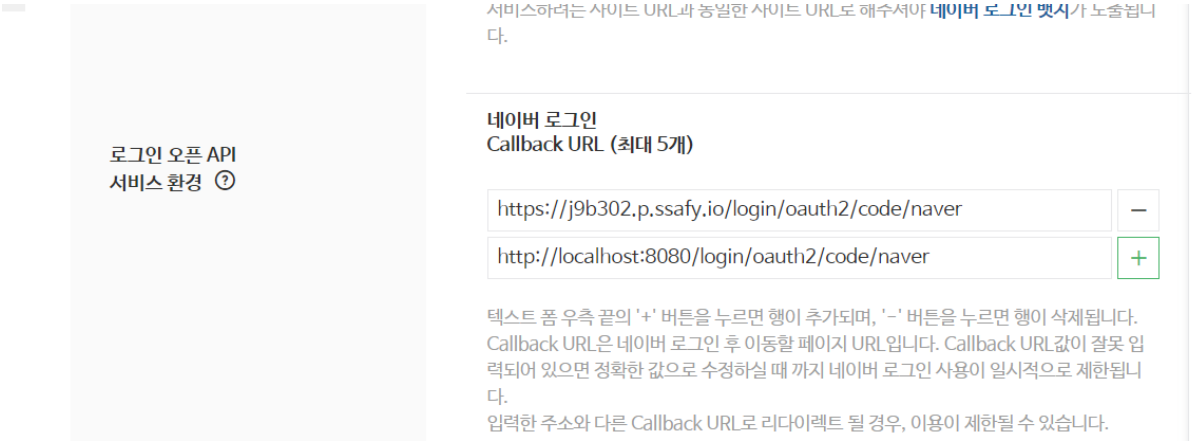


/login/oauth2/code/{socialName}
해당 경로는 spring-boot-starter-oauth2-client가 자동으로 리다이렉트를 잡아줌.

5.2. 소셜로그인 - Naver

네이버 개발자

1. 리다이렉트 URI 지정



네이버 소셜 로그인은 검수 요청을 받아야 외부 인원이 로그인 서비스를 이용할 수 있다.

검수를 받지 않으면 테스트 계정 ID를 등록해놓은 유저만 로그인 기능이 이용 가능함.(즉 검수 필수)

yml

```
spring:
  security:
    oauth2:
```

```
    client:
      registration:
        naver:
          clientId: {네이버 id}
          clientSecret: {네이버 시크릿키}
          clientAuthenticationMethod: post
          authorizationGrantType: authorization_code
          redirectUri: http://{서버 주소}/login/oauth2/code/naver
          scope: nickname, email, profile_image
          clientName: Naver
        kakao:
          client-id: {카카오 id}
          client-secret: {카카오 시크릿키}
          scope:
            - account_email
            - profile_nickname
            - profile_image
          authorization-grant-type: authorization_code
          redirect-uri: http://{서버 주소}/login/oauth2/code/kakao
          client-name: Kakao
          client-authentication-method: POST
      provider:
        naver:
          authorizationUri: https://nid.naver.com/oauth2.0/authorize
          tokenUri: https://nid.naver.com/oauth2.0/token
          userInfoUri: https://openapi.naver.com/v1/nid/me
          userNameAttribute: response
        kakao:
          authorizationUri: https://kauth.kakao.com/oauth/authorize
          tokenUri: https://kauth.kakao.com/oauth/token
          userInfoUri: https://kapi.kakao.com/v2/user/me
          userNameAttribute: id
```

요청 흐름

1. 프론트엔드 로그인 요청 주소
   `{서버 주소}/oauth2/authorization/kakao` 해당 주소로 로그인 요청 보냄.
2. 백엔드 Spring Security Oauth2.0이 /oauth2/authorization/{registrationId} 엔드포인트를 자동으로 잡아서 소셜 로그인의 화면으로 리다이렉트 해준다.
3. 로그인 성공하면 설정해놓은 redirect-uri에 따라 리다이렉트를 시켜준다.(인가 코드 포함)
4. SecurityConfig에서 `.oauth2Login()` 해당 부분이 자동으로
   /login/oauth2/code/{registrationId}
   를 잡아서 인가코드를 유저의 정보에 접근할 수 있는 액세스토큰으로 자동 교환해준다.
5. `principalOauth2UserService` 여기서 액세스토큰으로 유저의 정보를 꺼내서 없으면 회원가입 시키고 유저의 액세스 토큰을 `{서버 주소}/success?access=` 로 리다이렉트 시킨다.

6. 이후에는 access토큰을 우리 서비스 로그인 api로 보내서 access토큰과 일치하는 유저 정보를 찾아서 jwt 토큰 발급과 함께 로그인 로직을 실행시키면 된다.