



PLANEAT 포팅메뉴얼

I. 빌드 및 배포

1. 개발 환경
2. 설정 파일 목록
 - [React](#)
 - [Spring](#)
 - [Docker](#)
 - [Nginx](#)
3. 설정 파일 및 환경 변수 정보
 - [React](#)
 - [Spring](#)
 - [Docker](#)
 - [Nginx](#)
4. Docker 설치
5. SSL 인증서 발급
6. DB 및 Infra 배포
7. CI/CD
 - [BackEnd - 설정파일](#)
 - [BackEnd - Jenkins 설정](#)
 - [FrontEnd - 설정파일](#)
 - [FrontEnd - Jenkins 설정](#)

II. 외부 서비스

1. 소셜 로그인
 - [Google\(구글\)](#)
 - [Naver\(네이버\)](#)
 - [Kakao\(카카오\)](#)
2. Hadoop

I. 빌드 및 배포

1. 개발 환경

Server : AWS EC2 Ubuntu 20.04 LTS
Visual Studio Code : 1.70.1
IntelliJ IDEA : 2022.2(Ultimate Edition) 17.0.3+7-b469.32 amd64
JDK : openjdk-8
Docker : 20.10.18
Node.js : 18.9.1
MySQL : 8.0.30-1.el8
Nginx : 1.23.1
Jenkins : 2.361.1

2. 설정 파일 목록

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
- 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 요청 URL : ex) 서비스 도메인/api}

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_JAVASCRIPT_KEY={카카오 자바스크립트 KEY}
REACT_APP_KAKAO_REST_API_KEY={카카오 REST API KEY}
REACT_APP_KAKAO_CLIENT_ID={카카오 클라이언트 ID}
REACT_APP_KAKAO_CLIENT_SECRET={카카오 클라이언트 SECRET}
REACT_APP_KAKAO_REDIRECT_URI={카카오 리다이렉트 URI}
```

- Dockerfile

```
FROM node:alpine
WORKDIR /usr/src/app
COPY ./package* /usr/src/app/
RUN npm install --save --legacy-peer-deps
COPY ./ /usr/src/app/
CMD ["npm", "run", "start"]
```

Spring

- application.properties

```
server.port=9000

spring.datasource.url=jdbc:mysql://{DB컨테이너이름}:3306/{SCHEME이름}?serverTimezone=Asia/Seoul&useUnicode=yes&characterEncoding=UTF-8&zeroDateTimeBehavior=convertToNull
spring.datasource.username={사용자계정ID}
spring.datasource.password={사용자계정PASSWORD}

spring.jpa.show-sql=true

spring.jpa.generate-ddl=true
spring.jpa.hibernate.ddl-auto=none

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.jackson.serialization.fail-on-empty-beans=false

# context path
server.servlet.context-path=/api

# AWS S3 IAM user
cloud.aws.credentials.accessKey={AWS ACCESS KEY}
cloud.aws.credentials.secretKey={AWS SECRET KEY}
# AWS S3 bucket
```

```
cloud.aws.s3.bucket={버킷 이름}
cloud.aws.region.static=ap-northeast-2
cloud.aws.stack.auto=false

# swagger path
spring.mvc.pathmatch.matching-strategy=ant_path_matcher
```

- application-oauth.properties

```
# GOOGLE LOGIN
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

# KAKAO LOGIN
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

# KAKAO PROVIDER
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

# NAVER LOGIN
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

# NAVER PROVIDER
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
```

- Dockerfile

```
FROM openjdk:8-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 9000:9000 --name springbootapp --network ubuntu_default -d springbootapp
```

Docker

- docker-compose.yml

```

version: "3"
services:
  mysql:
    image: mysql
    container_name: mysql
    environment:
      MYSQL_DATABASE: {SCHEME 이름}
      MYSQL_ROOT_PASSWORD: {ROOT 계정 PASSWORD}
    volumes:
      - /mysql:/var/lib/mysql
    ports:
      - 3306:3306
  nginx:
    image: nginx
    container_name: nginx
    ports:
      - 80:80
      - 443:443
    volumes:
      - /etc/letsencrypt:/etc/letsencrypt
      - ./nginx/conf.d:/etc/nginx/conf.d
  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

```

Nginx

- app.conf

```

server {
    listen 80;
    server_name {서비스 도메인} www.{서비스 도메인};
    return 301 https://$server_name$request_uri;
}
server {
    listen 443 ssl;
    server_name {서비스 도메인};
    access_log off;

    ssl_certificate
    /etc/letsencrypt/live/{서비스 도메인}/fullchain.pem;
    ssl_certificate_key
    /etc/letsencrypt/live/{서비스 도메인}/privkey.pem;

    location / {
        proxy_pass http://{서비스 도메인}: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://{서비스 도메인}:9000/api/;
        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 설치

```
sudo apt-get update
sudo apt-get install \
ca-certificates \
curl \
gnupg \
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 \
"deb [arch=$(dpkg --print-architecture) signedby=/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
sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io dockercompose-plugin
```

- Docker Compose 설치

```
sudo curl -L
"https://github.com/docker/compose/releases/download/1.29.2/dockercompose-$(uname -s)-$(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 {www제외한 서비스 도메인}
```

이메일 작성 후 Agree

뉴스레터 수신 여부 No

```
ssl_certificate /etc/letsencrypt/live/{서비스 도메인}/fullchain.pem;
ssl_certificate_key /etc/letsencrypt/live/{서비스 도메인}/privkey.pem;
```

6. DB 및 Infra 배포

1. docker-compose 작성 ([docker-compose.yml](#))

위치 : /home/ubuntu

2. docker-compose 실행

/home/ubuntu (docker-compose.yml 있는 경로에서)

```
sudo docker-compose up -build -d
```

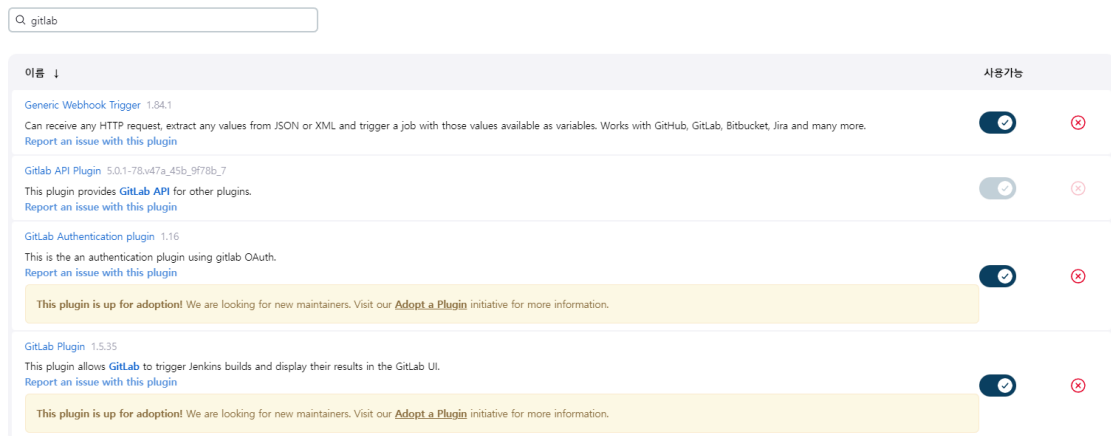
3. Nginx 설정 ([app.conf](#))

위치 : /home/ubuntu/nginx/conf.d

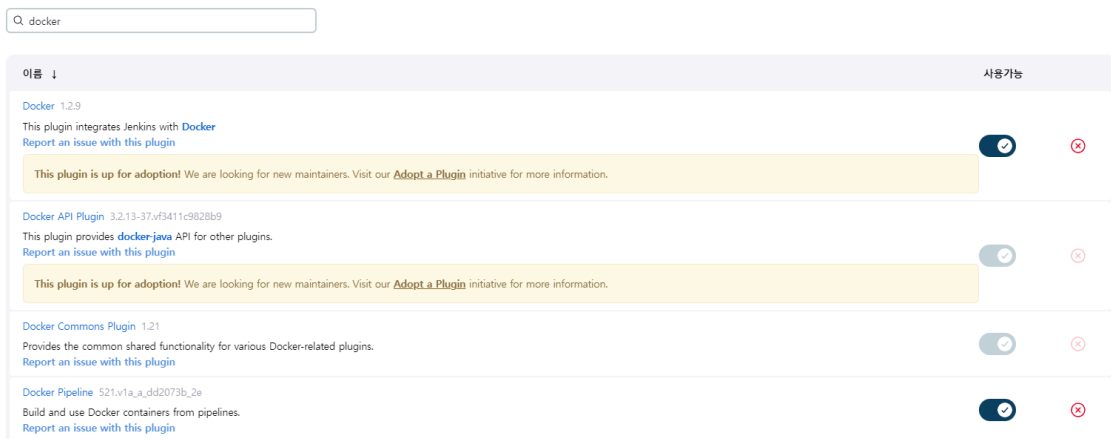
4. Jenkins 설정

- i. [http://{서비스 도메인}:9090](#) 으로 접속하여 jenkins 페이지 진입
- ii. 서버 콘솔에서 `sudo docker logs {Jenkins 컨테이너 이름}`으로 Administrator password를 확인하고 입력
- iii. Install suggested plugins를 선택하여 권장 플러그인 설치
- iv. 생성할 관리자 계정 정보를 입력하고 Save and Continue
- v. Jenkins 접속 URL 확인 후 Save and Finish

- vi. 메인 화면에서 DashBoard > Manager Jenkins > Plugin Manager
gitlab 검색 후 스크린샷과 같이 설치



- docker 검색 후 스크린샷과 같이 설치



- vii. 서버 콘솔에서 Jenkins 내부에 docker를 설치

```
$ sudo docker exec -it {jenkins 컨테이너명} /bin/bash
$ apt-get update -y
$ apt-get install -y
$ apt-get install docker.io -y
$ docker -v
```

7. CI/CD

BackEnd - 설정파일

- application.properties 작성 (application.properties)
- application-oauth.properties 작성 (application-oauth.properties)
- Dockerfile 작성 (Dockerfile)

BackEnd - Jenkins 설정

1. Jenkins 메인 화면 > Dashboard > 새로운 Item
2. FreeStyle project 선택하고 item name은 backend로 설정 후 OK
3. Repositories URL에는 프로젝트 레포지토리의 HTTPS 주소 입력
 - Credentials > Add 클릭

- Domain > Global credentials
 - Kind > Username with password
 - Username > 레포지토리 접근 권한이 있는 GitLab 계정 아이디
 - Password > 레포지토리 접근 권한이 있는 GitLab 계정 비밀번호
4. 생성된 Credential을 선택
 5. 스크린샷과 같이 설정

Branches to build ?

Branch Specifier (blank for 'any') ?

refs/heads/{브랜치 이름}

{브랜치 이름} 브랜치의 내용만을 받아와서 빌드

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://j7a701p.ssafy.io: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

Push, Merge Request가 발생할 때마다 빌드 유발

Build Steps

Execute shell ?

Command

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

```
cd backend
chmod +x gradlew
chmod +x deploy.sh
./deploy.sh
```

고급...

backend 폴더로 이동하여 gradlew 와 deploy.sh 파일에 실행할 수 있는 권한 부여

```
cd backend
chmod +x gradlew
chmod +x deploy.sh
./deploy.sh
```

6. Jenkins에 sudo 권한 부여 (중요한 파일이기 때문에 오타 없는지 한번 더 확인)

```
$ 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

7. GitLab Webhook 작성 (Settings > Webhook)

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.

FrontEnd - 설정파일

- .env 작성 (.env)
- Dockerfile 작성 (Dockerfile)

FrontEnd - Jenkins 설정

1. Jenkins 메인 화면 > Dashboard > 새로운 Item
2. FreeStyle project 선택하고 item name은 frontend로 설정 후 OK
3. Repositories URL에는 프로젝트 레포지토리의 HTTPS 주소 입력
4. 이전 과정에서 생성했던 Credential을 선택
5. 스크린샷과 같이 설정

Branches to build ?

Branch Specifier (blank for 'any') ?

{브랜치 이름} 브랜치의 내용만을 받아와서 빌드

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://j7a701.p.ssafy.io: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

Push, Merge Request가 발생할 때마다 빌드 유발

Build Steps

Execute shell ?

Command

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

```
cd frontend
docker build -t reactapp .
docker stop reactapp
docker rm reactapp
```

고급...

frontend 폴더로 이동하여 도커 컨테이너 생성, 기존 컨테이너 중지 후 삭제, 새로운 컨테이너 run

```
cd frontend
docker build -t reactapp .
docker stop reactapp
docker rm reactapp
docker run -p 3000:3000 --name reactapp --network ubuntu_default -d reactapp
```

7. GitLab Webhook 작성 (Settings > Webhook)

Search page

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

<http://{서비스 도메인}:9090/project/frontend>

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

{프론트엔드 브랜치 이름}

Push to the repository.

II. 외부 서비스

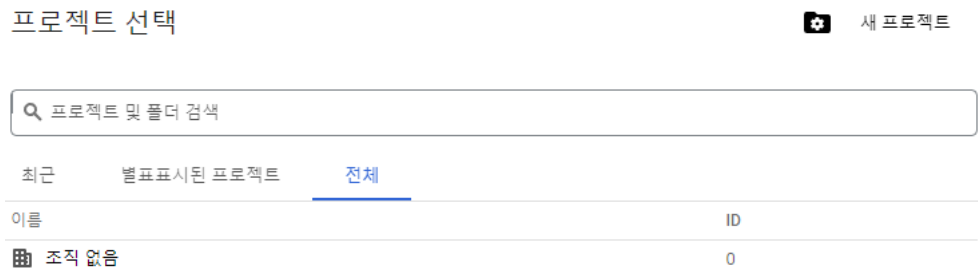
1. 소셜 로그인

Google(구글)

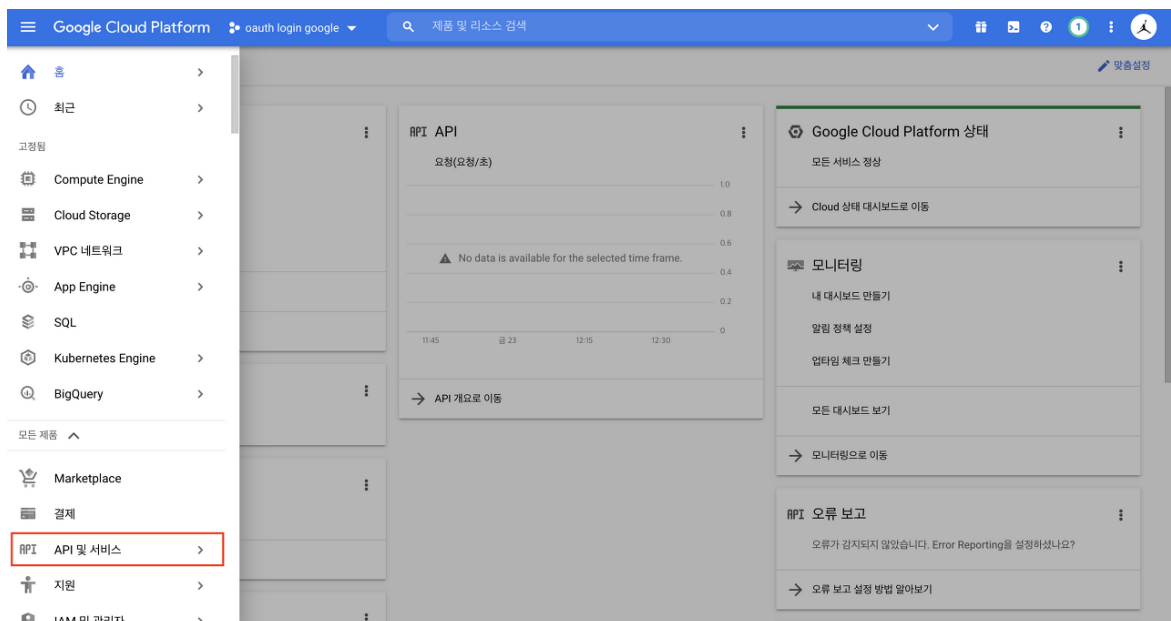
1. <https://console.cloud.google.com/home/dashboard>로 접속한다.



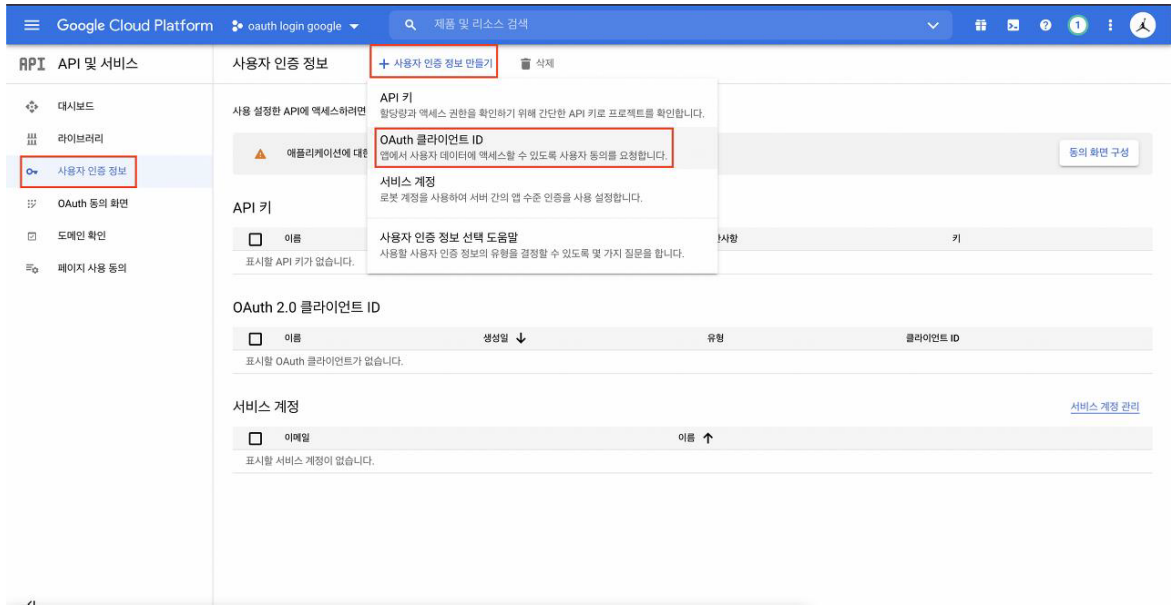
2. 프로젝트 선택을 클릭하고 모달 우측 상단의 새 프로젝트 클릭



3. API 및 서비스



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



- 동의 화면 구성을 클릭하고 해당 화면에서 외부로 선택하여 외부 사용자들이 사용할 수 있도록 설정한다.

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. [외부 앱을 제출할 필요는 없습니다.](#)

☒ 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](#)

[서비스 계정 관리](#)

[만들기](#)

- 앱 정보를 입력하고 , 개발자 연락처 정보에 이메일을 입력한다.

앱 정보

동의 화면에 표시되어 최종 사용자가 개발자를 확인하고 문의할 수 있습니다.

앱 이름 *

PLANEAT

동의를 요청하는 앱의 이름

사용자 지원 이메일 *

0atx.yh@gmail.com

사용자가 동의 관련 질문을 위해 문의할 때 이용합니다.

앱 로고

PLANEAT_logo_120.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.



개발자 연락처 정보

이메일 주소 *

0atx.yh@gmail.com

이 이메일 주소는 Google에서 프로젝트 변경사항에 대해 알림을 보내기 위한 용도입니다.

7. 범위 추가 또는 삭제 > email, profile 선택 후 저장
민감하지 않은 범위에 추가된 것 확인

Google Cloud PLANEAT

API 및 서비스

앱 등록 수정

사용 설정된 API 및 서비스

라이브러리

사용자 인증 정보

OAuth 동의 화면

도메인 확인

페이지 사용 동의

범위 추가 또는 삭제

민감하지 않은 범위

API ↑ 범위 사용자에게 표시되는 설명

표시할 행이 없습니다.

민감한 범위

민감한 범위는 비공개 사용자 데이터에 대한 액세스를 요청하는 범위입니다.

API ↑ 범위 사용자에게 표시되는 설명

표시할 행이 없습니다.

선택한 범위 업데이트

아래에는 사용 설정된 API의 범위만 나와 있습니다. 이 화면에 누락된 범위를 추가하려면 [Google API 라이브러리](#)에서 API를 찾아 사용 설정하거나 아래의 '보여줄은 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	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)

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

이름 *

PLANEAT

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



아래에 추가한 URI의 도메인이 [승인된 도메인](#)으로 [OAuth 동의 화면](#)에 자동으로 추가됩니다.

승인된 자바스크립트 원본 ?

브라우저 요청에 사용

URI 1 *

https://j7a701.p.ssafy.io

+ URI 추가

승인된 리디렉션 URI ?

웹 서버의 요청에 사용

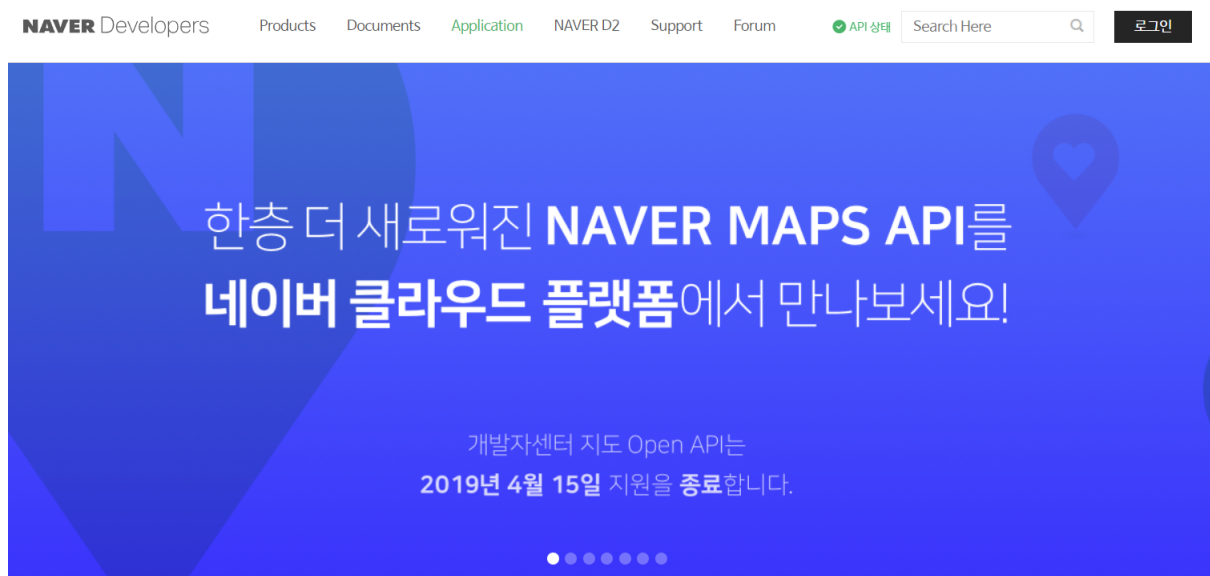
URI 1 *

https://j7a701.p.ssafy.io/api/login/oauth2/code/google

+ URI 추가

Naver(네이버)

1. <https://developers.naver.com/main/>로 접속한다.



2. 상단의 Application > 애플리케이션 등록 클릭

3. 필요 정보를 기입한다.

애플리케이션 이름

PLANEAT

카테고리

건강/운동

선택하세요.

네이버 로그인

제공 정보 선택(이용자 식별자는 기본 정보로 제공) ②

필수 항목은 개인정보보호법 제3조 제1항, 제16조 제1항 등에 따라 서비스 제공을 위해 필요한 최소한의 개인정보만을 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 type="checkbox"/>	<input type="checkbox"/>
성별	<input type="checkbox"/>	<input type="checkbox"/>
생일	<input type="checkbox"/>	<input type="checkbox"/>
연령대	<input type="checkbox"/>	<input type="checkbox"/>
출생연도	<input type="checkbox"/>	<input type="checkbox"/>
휴대전화번호	<input type="checkbox"/>	<input type="checkbox"/>

[알림] 추가권한에 대한 네이버 로그인 공지사항을 확인하세요.

로그인 오픈 API
서비스 환경 ②

PC 웹
X
^

서비스 URL

https://j7a701.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 사용
이 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 노출됩니
다.

네이버 로그인
Callback URL (최대 5개)

https://j7a701.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 입
력되어 있으면 정확한 값으로 수정하실 때 까지 네이버 로그인 사용이 일시적으로 제한됩니
다.
입력한 주소와 다른 Callback URL로 리다이렉트 될 경우, 이용이 제한될 수 있습니다.

로그 이미지



4. 멤버 관리에서 테스터 ID를 등록한다.

PLANEAT

개요	API 설정	네이버 로그인 검수상태	멤버관리	로그인 통계	API 통계	Playground(Beta)
----	--------	-----------------	------	--------	--------	------------------

관리자 ID 등록(최대 3개)	<div> <div>wjtkdl96</div> <div>—</div> </div> <div> <div></div> <div>+</div> </div>
<p>애플리케이션 관리자를 3명까지 설정함으로써 개발자 부재시 다른 사람이 애플리케이션 설정을 할 수 있습니다. 아울러 애플리케이션을 테스트할 수 있는 테스트 아이디를 최대 20개까지 등록할 수 있도록 함으로써 배포전에 테스트할 수 있습니다.</p>	
테스터 ID 등록(최대 20개)	<div> <div>peace0704</div> <div>—</div> </div> <div> <div>gpdks_668</div> <div>—</div> </div> <div> <div>shinzan</div> <div>—</div> </div> <div> <div>yywwxxzz</div> <div>—</div> </div> <div> <div></div> <div>+</div> </div>
<p>테스터 ID는 애플리케이션이 '개발 중'상태일 때 해당 애플리케이션에 로그인하여 테스트할 수 있는 ID입니다. 최대 20개까지 등록이 가능하며, 관리자 아이디는 등록하지 않아도 로그인이 가능합니다.</p>	

Kakao(카카오)

1. <https://developers.kakao.com/>로 접속한다.



2. 내 애플리케이션 > 애플리케이션 추가하기 > 앱 이름 입력 > 사업자명 입력 > 저장

애플리케이션 추가하기

앱 아이콘



파일 선택

JPG, GIF, PNG

권장 사이즈 128px, 최대 250KB

앱 이름

PLANEAT

사업자명

A701

- 입력된 정보는 사용자가 카카오 로그인할 때 표시됩니다.
- 정보가 정확하지 않은 경우 서비스 이용이 제한될 수 있습니다.

☒ [서비스 이용이 제한되는 카테고리, 금지된 내용, 금지된 행동](#) 관련 운영정책을 위반하지 않는 앱입니다.

취소

저장

3. 좌측 Nav 바에서

앱 설정 > 요약 정보 > 앱 키 > REST API 키와

제품 설정 > 카카오 로그인 > 보안 > Client Secret 의 코드를 기록해둔다.



앱 키

플랫폼	앱 키		재발급
네이티브 앱 키		복사	재발급
REST API 키		복사	재발급
JavaScript 키		복사	재발급
Admin 키		복사	재발급

- 네이티브 앱 키: Android, iOS SDK에서 API를 호출할 때 사용합니다.
- JavaScript 키: JavaScript SDK에서 API를 호출할 때 사용합니다.
- REST API 키: REST API를 호출할 때 사용합니다.
- Admin 키: 모든 권한을 갖고 있는 키입니다. 노출이 되지 않도록 주의가 필요합니다.

카카오 로그인

ON

Client Secret

삭제

토큰 발급 시, 보안을 강화하기 위해 Client Secret을 사용할 수 있습니다. (REST API인 경우에 해당)

코드		재발급
활성화 상태	사용함	설정

4. 좌측 Nav 바의

앱 설정 > 플랫폼 > Web > Web 플랫폼 등록에서 필요 정보를 기입한다.

Web 플랫폼 등록

사이트 도메인

JavaScript SDK, 카카오톡 공유, 카카오톡, 메시지 API 사용시 등록이 필요합니다.

여러개의 도메인은 줄바꿈으로 추가해주세요. 최대 10까지 등록 가능합니다. 추가 등록은 포럼(데브톡)으로 문의주세요.

예시: (O) <https://example.com> (X) <https://www.example.com>

<https://j7a701.p.ssafy.io>

기본 도메인

기본 도메인은 첫 번째 사이트 도메인으로, 카카오톡 공유와 카카오톡 메시지 API를 통해 발송되는 메시지의 Web 링크 기본값으로 사용됩니다.

<https://j7a701.p.ssafy.io>

취소

저장

5. 제품 설정 > 카카오 로그인 클릭 후 Redirect URI 를 입력한다.

Redirect URI

Redirect URI

카카오 로그인에서 사용할 OAuth Redirect URI를 설정합니다.

여러개의 URI를 줄바꿈으로 추가해주세요. (최대 10개)

REST API로 개발하는 경우 필수로 설정해야 합니다.

예시: (O) <https://example.com/oauth> (X) <https://www.example.com/oauth>

<https://j7a701.p.ssafy.io/api/login/oauth2/code/kakao>

취소

저장

6. 제품 설정 > 카카오 로그인 > 동의항목에서 아래와 같이 설정한다.

동의항목

카카오 로그인으로 서비스를 시작할 때 동의 받는 항목을 설정합니다. 미리 보기를 통해 사용자에게 보여질 화면을 확인할 수 있습니다.

비즈니스 채널을 연결하면 권한이 없는 동의 항목에 대한 검수 신청을 할 수 있습니다.

[카카오비즈니스 관리자센터에서 비즈니스 채널 연결 →](#)

개인정보

항목 이름	ID	상태
닉네임	profile_nickname	● 필수 동의 설정
프로필 사진	profile_image	● 사용 안함 설정
카카오계정(이메일)	account_email	● 필수 동의 [수집] 설정
성별	gender	● 사용 안함 설정
연령대	age_range	● 사용 안함 설정

- application-oauth.properties 작성 ([application-oauth.properties](#))

2. Hadoop

1. 데이터 분산 처리

- 데이터를 하둡 분산 파일 시스템에 저장

```
hdfs dfs -put review(파일명) review(파일명)
```

2. 하둡 mapreduce 실행

- wordcount 로직

```
package ssafy;

import java.io.IOException;
import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.input.FileSplit;
import org.apache.hadoop.util.GenericOptionsParser;

public class Wordcount {
    /*
    Object, Text : input key-value pair type (always same (to get a line of input file))
    Text, IntWritable : output key-value pair type
    */
    public static class TokenizerMapper
        extends Mapper<Object,Text,Text,IntWritable> {

        // variable declairations
        private final static IntWritable one = new IntWritable(1);
        private Text word = new Text();
        private String filename;
        protected void setup(Context context) throws IOException, InterruptedException{
            filename = ((FileSplit)context.getInputSplit()).getPath().getName();
        }

        // map function (Context -> fixed parameter)
        public void map(Object key, Text value, Context context)
            throws IOException, InterruptedException {

            // value.toString() : get a line
            StringTokenizer itr = new StringTokenizer(value.toString());
            while ( itr.hasMoreTokens() ) {
                String token = filename + " " + itr.nextToken();
                word.set(token);

                // emit a key-value pair
                context.write(word,one);
            }
        }
    }

    /*
    Text, IntWritable : input key type and the value type of input value list
    Text, IntWritable : output key-value pair type
    */
    public static class IntSumReducer
        extends Reducer<Text,IntWritable,Text,IntWritable> {

        // variables
        private IntWritable result = new IntWritable();

        // key : a disticnt word
        // values : Iterable type (data list)
        public void reduce(Text key, Iterable<IntWritable> values, Context context)
            throws IOException, InterruptedException {

            int sum = 0;
            for ( IntWritable val : values ) {
                sum += val.get();
            }
            result.set(sum);
        }
    }
}
```

```

        context.write(key,result);
    }
}

/* Main function */
public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    String[] otherArgs = new GenericOptionsParser(conf,args).getRemainingArgs();
    if ( otherArgs.length != 2 ) {
        System.err.println("Usage: <in> <out>");
        System.exit(2);
    }
    Job job = new Job(conf,"word count");
    job.setJarByClass(Wordcount.class);

    // let hadoop know my map and reduce classes
    job.setMapperClass(TokenizerMapper.class);
    job.setReducerClass(IntSumReducer.class);

    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);

    // set number of reduces
    job.setNumReduceTasks(2);

    // set input and output directories
    FileInputFormat.addInputPath(job,new Path(otherArgs[0]));
    System.exit(job.waitForCompletion(true) ? 0 : 1 );
}
}

```

- Driver.java

```

package ssafy;

import org.apache.hadoop.util.ProgramDriver;

public class Driver {
    public static void main(String[] args) {
        int exitCode = -1;
        ProgramDriver pgd = new ProgramDriver();
        try {

            pgd.addClass("wordcount", Wordcount.class, "A map/reduce program that performs word counting.");
            pgd.addClass("wordcount1char", Wordcount1char.class, "A map/reduce program that counts the 1st character of wo");
            pgd.addClass("wordcountsort", Wordcountsort.class, "A map/reduce program that output frequency of the words in");
            pgd.addClass("inverted", InvertedIndex.class, "A map/reduce program that generates the inverted index using wo");
            pgd.addClass("matadd", MatrixAdd.class, "A map/reduce program that computes the addition of two matrices.");
            pgd.driver(args);
            exitCode = 0;
        }
        catch(Throwable e) {
            e.printStackTrace();
        }

        System.exit(exitCode);
    }
}

```

- 수집한 리뷰 데이터 wordcount 실행

```
hadoop jar ssafy.jar(jar파일명) wordcount review(파일명) wordcount_review(결과 파일명)
```

3. 결과 파일 가져오기

```
hdfs dfs -get wordcount_review wordcount_review
```