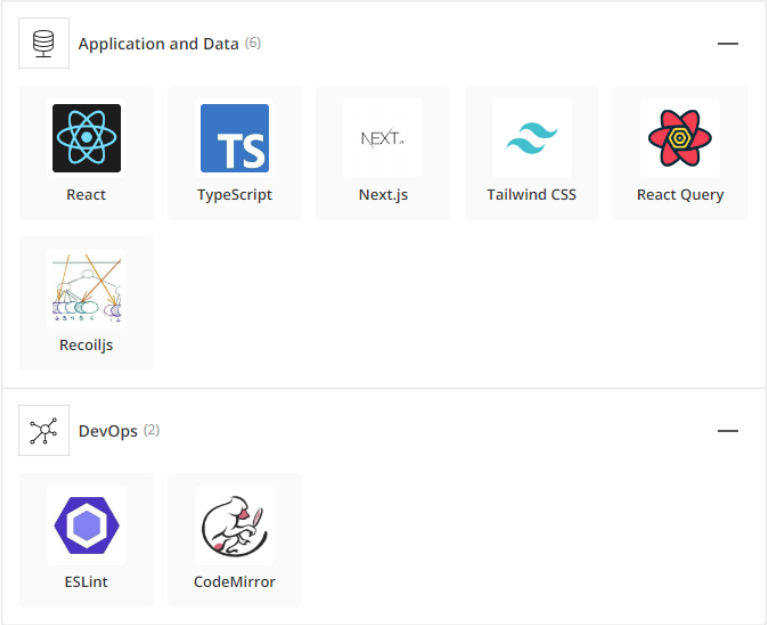
Code Bamboo 포팅매뉴얼

1. 프로젝트 아키텍처

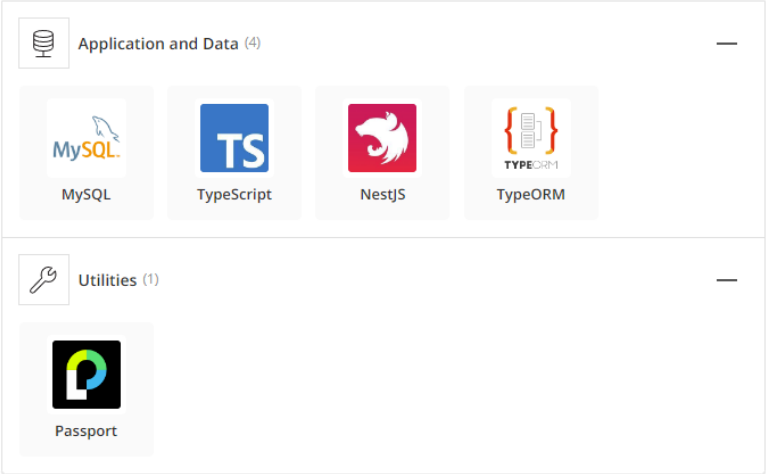
기술스택

Frontend



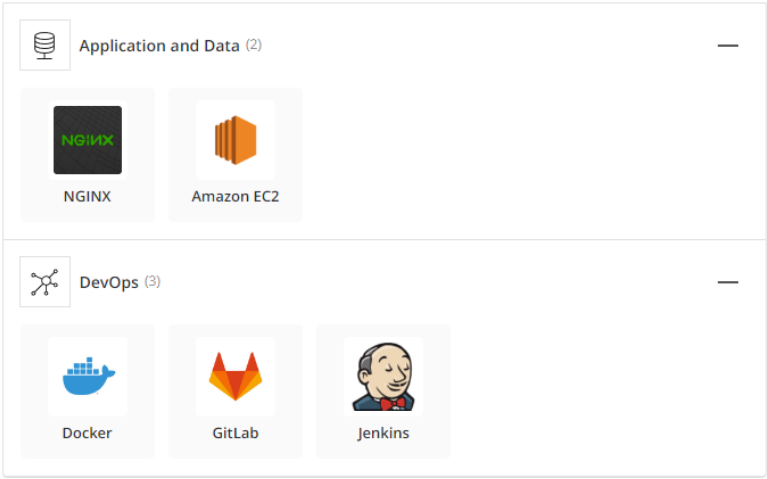
React	18.2.0
TypeScript	5.0.4
Next.js	13.3.0
TailwindCSS	3.3.1
ReactQuery	3.39.3
Recoiljs	0.7.7
ESLint	8.38.0
CodeMirror	5.56.13

Backend

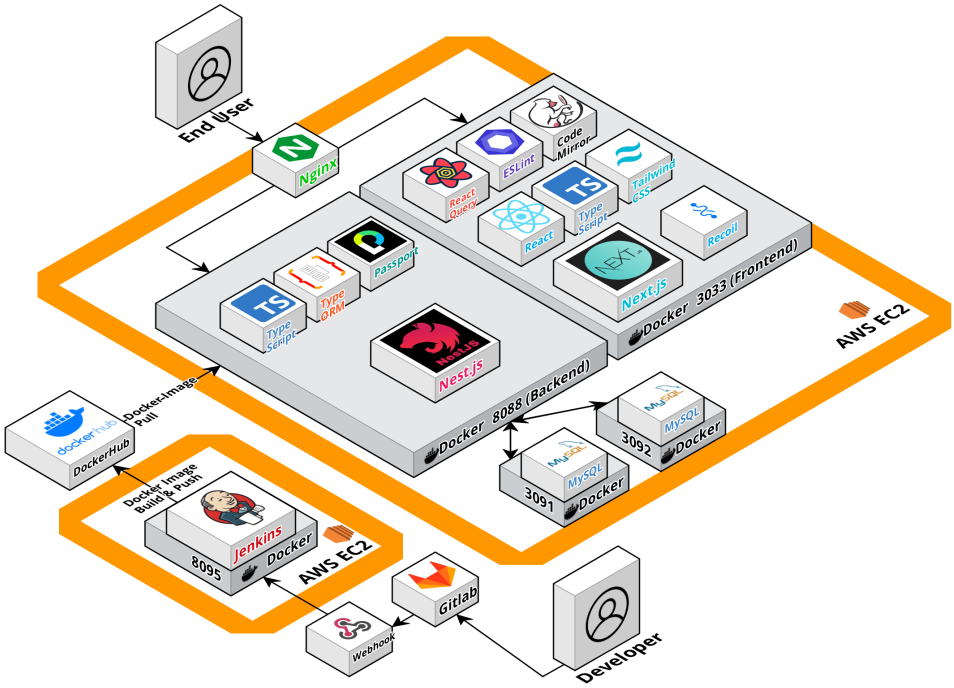


MySQL	8.3.0
TypeScript	4.7.4
NestJS	9.4.0
TypeORM	0.3.15
Passport	0.6.0

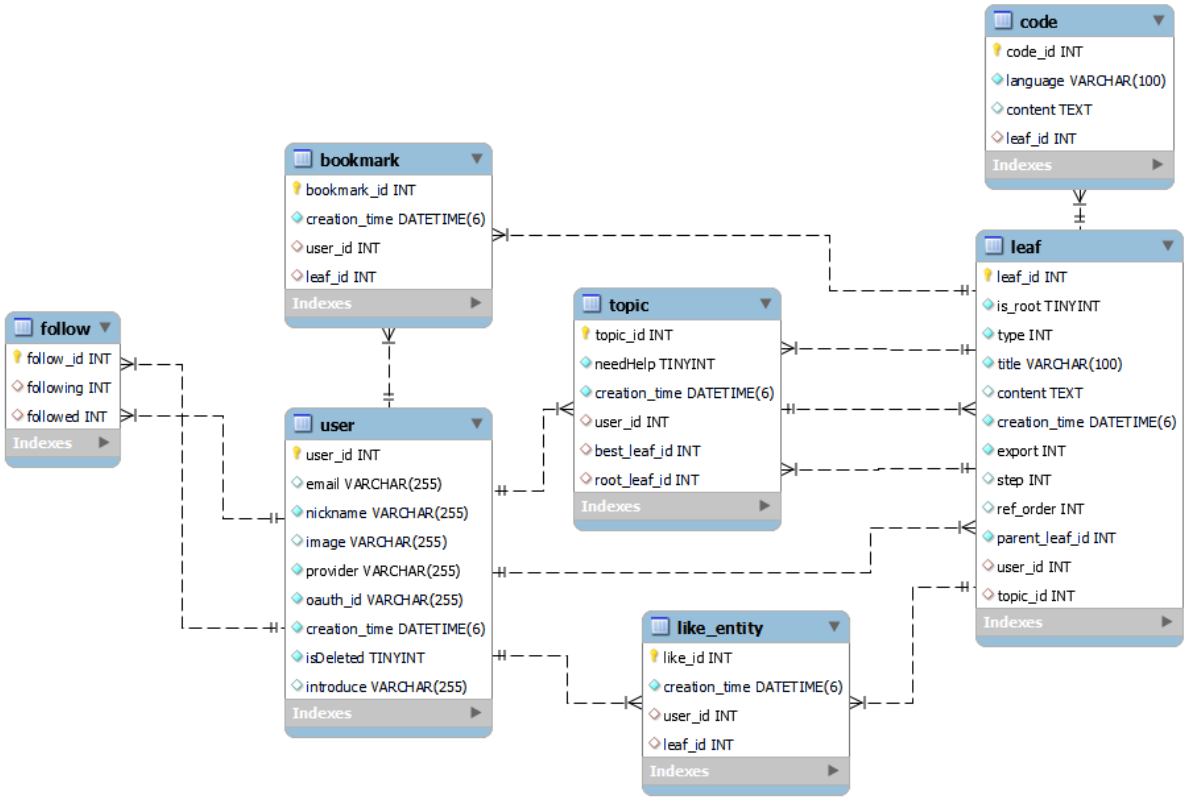
CICD



아키텍처 구조도



ERD



2. AWS EC2 환경 세팅

1. 빌드서버

Jenkins 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) 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

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

$ docker pull jenkins/jenkins:lts

$ sudo apt-get install openjdk-11-jdk -y

$ sudo apt-get install -y git
```

docker-compose.yml

```
version: "3"
services:
  jenkins:
    privileged: true
    restart: always
    container_name: jenkins
    image: jenkins/jenkins:lts
    user: root
    ports:
            - "8095:8080"
            - "50010:50000"
    expose:
            - "8080"
            - "50000"
    volumes:
      - './jenkins:/var/jenkins_home'
      - '/var/run/docker.sock:/var/run/docker.sock'
    environment:
        TZ: "Asia/Seoul"
```

Jenkins Container 내부에 도커 설치

```
# jenkins container 접속
docker exec -it jenkins /bin/bash

# linux 버전 확인
cat /etc/issue
# --------------- OS s-------------------------------
# root@DESKTOP-R4P59B3:/home/opendocs# cat /etc/issue
# Ubuntu 20.04.4 LTS \n \l
# --------------- jenkins Container OS -------------------------------
# root@DESKTOP-R4P59B3:/home/opendocs# docker exec -it jenkins /bin/bash
# root@8fc963af71bb:/# cat /etc/issue
# Debian GNU/Linux 11 \n \l

# Docker 설치
## - Old Version Remove
apt-get remove docker docker-engine docker.io containerd runc
## - Setup Repo
apt-get update
apt-get install \
    ca-certificates \
    curl \
    gnupg \
    lsb-release
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
## - Install Docker Engine
apt-get update
apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin
```

2. 배포서버

Container - frontend, backend, mysql(배포), mysql(개발)

```
PORTS                                                   NAMES
0.0.0.0:8088->8000/tcp, :::8088->8000/tcp               backend
0.0.0.0:3033->3000/tcp, :::3033->3000/tcp               frontend
33060/tcp, 0.0.0.0:3092->3306/tcp, :::3092->3306/tcp    ecstatic_lederberg
33060/tcp, 0.0.0.0:3091->3306/tcp, :::3091->3306/tcp    mysql
```

docker-compose.yml

```
version: '3'

services:
  mysql:
    restart: always
    image: mysql
    container_name: mysql
    ports:
      - 3091:3306
    volumes:
      - /mysql:/var/lib/mysql
    environment:
      MYSQL_ROOT_PASSWORD: "${DB_ROOT_PASSWORD}"
      MYSQL_DATABASE: "${DB_DATABASE}"
      MYSQL_USER: "${DB_USER}"
      MYSQL_PASSWORD: "${DB_PASSWORD}"
      TZ: Asia/Seoul
    command:
      - --character-set-server=utf8mb4
      - --collation-server=utf8mb4_unicode_ci
```

Nginx 설치

```
sudo apt update -y
sudo apt install nginx -y
```

jenkins, docker, nginx를 설치 후 sudo service <service-name> status 명령으로 잘 실행되고 있는지 확인

SSL 인증서 발급

```
sudo apt-get install letsencrypt
sudo letsencrypt certonly --startalone -d k8a801.p.ssafy.io
```

NGINX conf

```
server {
        listen 80;
        server_name k8a801.p.ssafy.io;
        server_tokens off;

    if ($host = k8a801.p.ssafy.io) {
                return 301 https://$host$request_uri;
    } # managed by Certbot
```

```
    return 404; # managed by Certbot
}

server {
        listen 443 ssl;
        server_name k8a801.p.ssafy.io;
        server_tokens off;

        ssl_certificate /etc/letsencrypt/live/k8a801.p.ssafy.io/fullchain.pem; # managed by Certbot
        ssl_certificate_key /etc/letsencrypt/live/k8a801.p.ssafy.io/privkey.pem; # managed by Certbot

        location /{
                proxy_pass http://localhost:3033;
        }

        location /api/ {
                proxy_pass http://localhost:8088/;
        }
}
```

설정 적용

```
sudo ln -s /etc/nginx/sites-available/nginx.conf /etc/nginx/sites-enabled/nginx.conf
sudo nginx -t
sudo systemctl restart nginx
sudo systemctl status nginx
```

Frontend Dockerfile

```
# Use a node image as the base image
FROM node:16
ENV REACT_APP_SERVER_BASE_URL https://k8a801.p.ssafy.io/api
# Set the working directory
WORKDIR /usr/src/front
# Copy the package.json and package-lock.json files to the working directory
COPY ./package* /usr/src/front/
# Install the dependencies
RUN npm i --legacy-peer-deps
# Copy the rest of the source code to the working directory
COPY ./ /usr/src/front/
# Build the React app
RUN npm run build
EXPOSE 3000
CMD ["npm", "run", "start"]
```

Backend Dockerfile

```
# Use a node image as the base image
FROM node:16
# Set the working directory
WORKDIR /usr/src/back
# Copy the package.json and package-lock.json files to the working directory
COPY ./package* /usr/src/back/
# Install the dependencies
RUN npm install
# Copy the rest of the source code to the working directory
COPY ./ /usr/src/back/
# Build the React app
RUN npm run build
EXPOSE 8000
CMD ["node", "dist/main.js"]
```

Jenkins Credentials 설정

Global credentials (unrestricted)

+ Add Credentials

Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind	Description
dockerhub-jenkins	irang/******	Username with password	
Deploy-Server-SSH-Credential	irang6v6	SSH Username with private key	
git-credential	111601joo/******	Username with password	

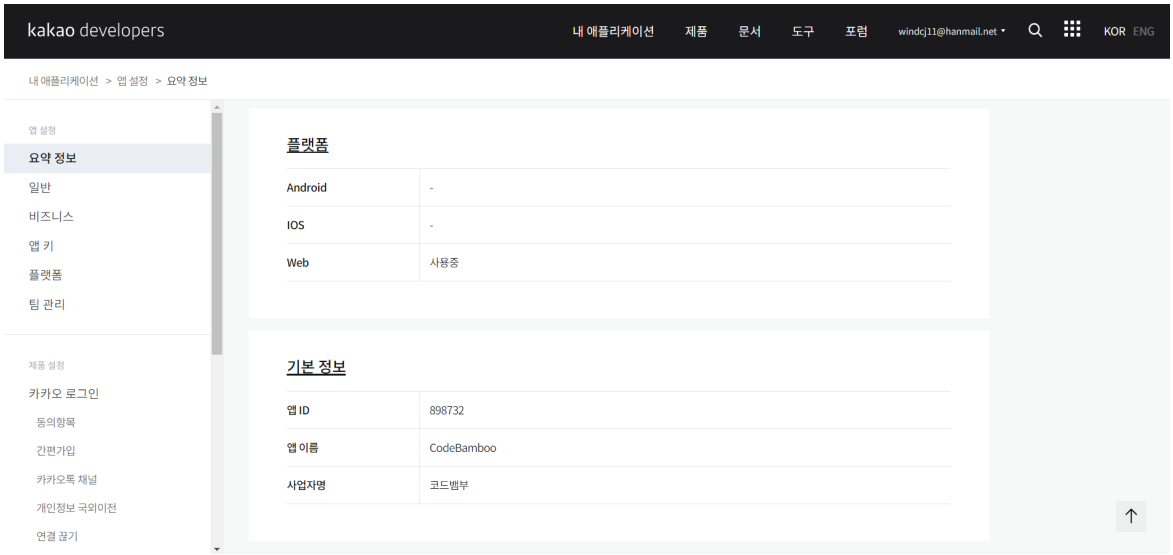
Jenkins CICD Pipeline

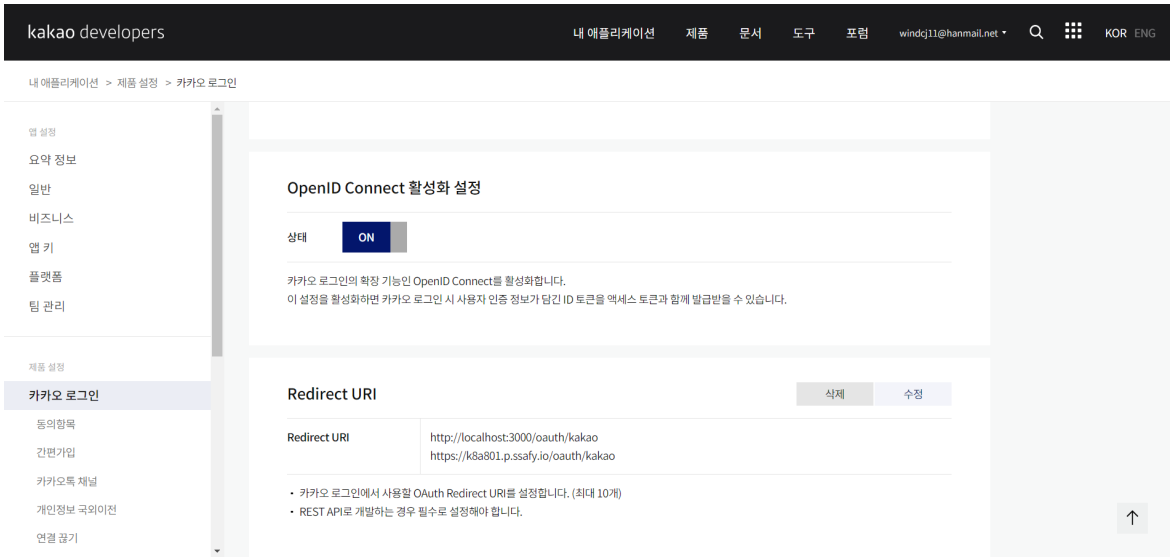
```
pipeline {
    agent any
    environment {
        repository = "codebamboo"
    }
    stages {
        stage("Set Variable") {
            steps {
                script {
                    PREV_BUILD_NUM = ("${env.BUILD_NUMBER}" as int) - 1
                    FRONT_CONTAINER_NAME = "frontend"
                    BACK_CONTAINER_NAME = "backend"
                    DOCKER_HUB_CREDENTIAL = "dockerhub-jenkins"
                    SSH_CONNECTION = "ubuntu@k8a801.p.ssafy.io"
                    SSH_CONNECTION_CREDENTIAL = "Deploy-Server-SSH-Credential"
                    GIT_CRED = "git-credential"
                    GIT_URL = "https://lab.ssafy.com/s08-final/S08P31A801.git"
                }
            }
        }

        stage('checkout') {
            steps {
                git branch: 'release',
                    credentialsId: "${GIT_CRED}",
                    url: "${GIT_URL}",
                    poll: true,
                    changelog: true
            }
        }

        stage("Build Container Image") {
            steps {
                script {
                    dir('frontend') {
                        front_image = docker.build repository + ":frontend_${env.BUILD_NUMBER}"
                    }
                    dir ('backend') {
                        back_image = docker.build repository + ":backend_${env.BUILD_NUMBER}"
                    }
                }
            }
        }

        stage("Push Container Image To Docker Hub") {
            steps {
                script {
                    docker.withRegistry("https://registry.hub.docker.com", DOCKER_HUB_CREDENTIAL) {
                        front_image.push("frontend_${env.BUILD_NUMBER}")
                        back_image.push("backend_${env.BUILD_NUMBER}")
                    }
                }
            }
        }

        stage("Cleaning up") {
        steps {
                sh """docker images | grep "frontend_" | awk '{print \$1 ":" \$2}' | xargs docker rmi""" // docker image 제거
                sh """docker images | grep "backend_" | awk '{print \$1 ":" \$2}' | xargs docker rmi""" // docker image 제거
            }
        }

        stage("Server Run") {
            steps {
```

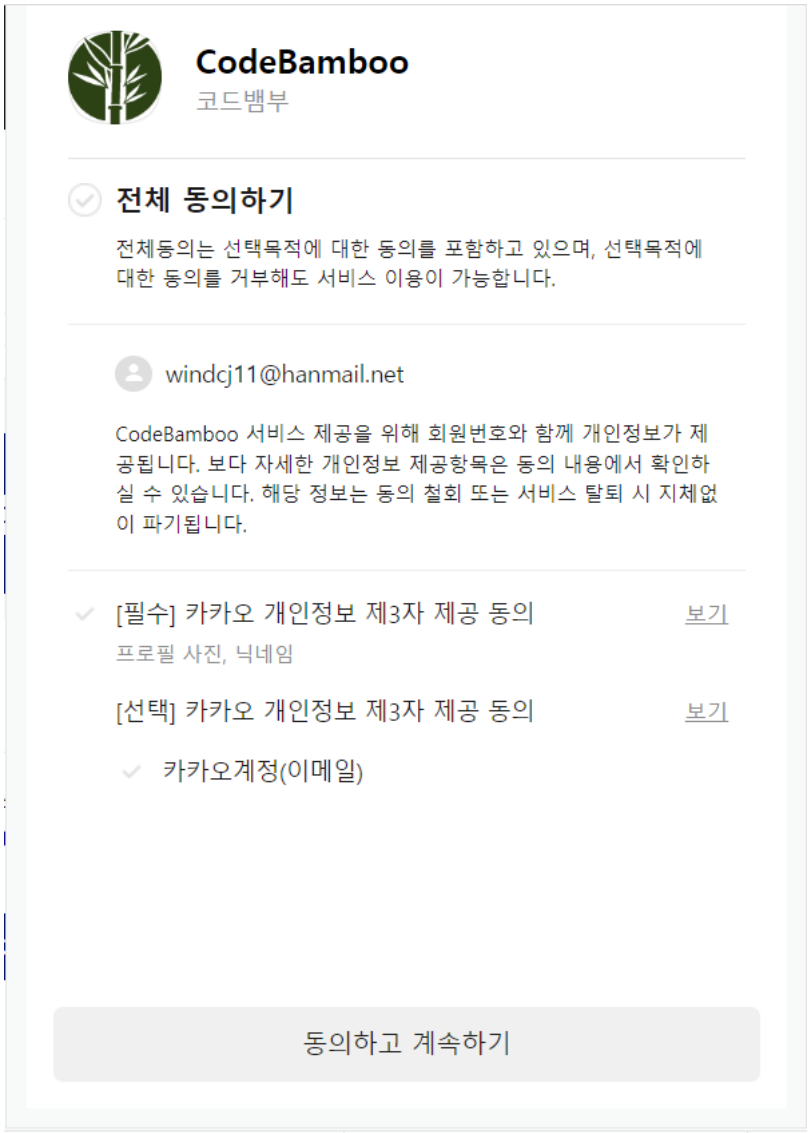
```
                sshagent([SSH_CONNECTION_CREDENTIAL]) {
                    // 이전 컨테이너 삭제
                    sh "ssh -o StrictHostKeyChecking=no ${SSH_CONNECTION} 'docker rm -f ${FRONT_CONTAINER_NAME}'"
                    sh "ssh -o StrictHostKeyChecking=no ${SSH_CONNECTION} 'docker rm -f ${BACK_CONTAINER_NAME}'"
                    // 이전 이미지 삭제
                    sh "ssh -o StrictHostKeyChecking=no ${SSH_CONNECTION} 'docker rmi -f $repository:frontend_${PREV_BUILD_NUM}'"
                    sh "ssh -o StrictHostKeyChecking=no ${SSH_CONNECTION} 'docker rmi -f $repository:backend_${PREV_BUILD_NUM}'"
                    // 최신 이미지 PULL
                    sh "ssh -o StrictHostKeyChecking=no ${SSH_CONNECTION} 'docker pull $repository:frontend_${env.BUILD_NUMBER}'"
                    sh "ssh -o StrictHostKeyChecking=no ${SSH_CONNECTION} 'docker pull $repository:backend_${env.BUILD_NUMBER}'"
                    // 이미지 확인
                    sh "ssh -o StrictHostKeyChecking=no ${SSH_CONNECTION} 'docker images'"
                    // 환경변수 파일 실행권한 주기
                    sh "ssh -o StrictHostKeyChecking=no ${SSH_CONNECTION} 'chmod +x /home/ubuntu/main.env'"
                    sh "ssh -o StrictHostKeyChecking=no ${SSH_CONNECTION} 'chmod +x /home/ubuntu/.env.production'"
                    // 최신 이미지 RUN
                    sh "ssh -o StrictHostKeyChecking=no ${SSH_CONNECTION} 'docker run -d --name ${FRONT_CONTAINER_NAME} --env-file /ho
                    sh "ssh -o StrictHostKeyChecking=no ${SSH_CONNECTION} 'docker run -d --name ${BACK_CONTAINER_NAME} --env-file /hom
                    // 컨테이너 확인
                    sh "ssh -o StrictHostKeyChecking=no ${SSH_CONNECTION} 'docker ps'"
                }
            }
        }

        stage("Send Mattermost Alarm"){
            steps{
                script{
                    mattermostSend (
                        color: "good",
                        message: "Deploy SUCCESS! 여기까지- #${env.BUILD_NUMBER} (<https://k8a801.p.ssafy.io/|Link to Site>)"
                    )
                }
            }
        }
    }
}
```

소셜로그인 설정

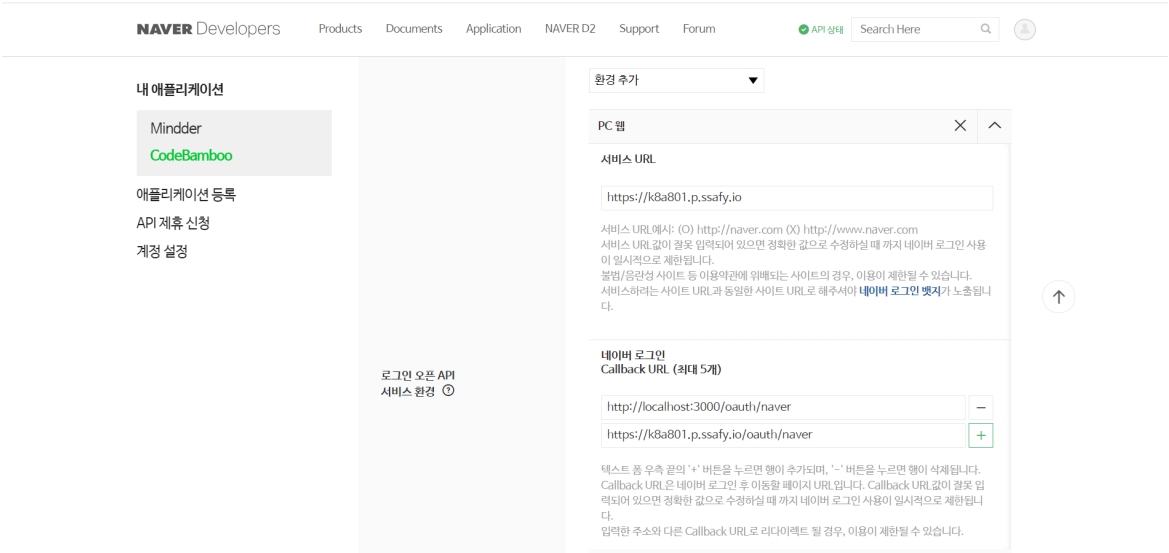
- 카카오 로그인

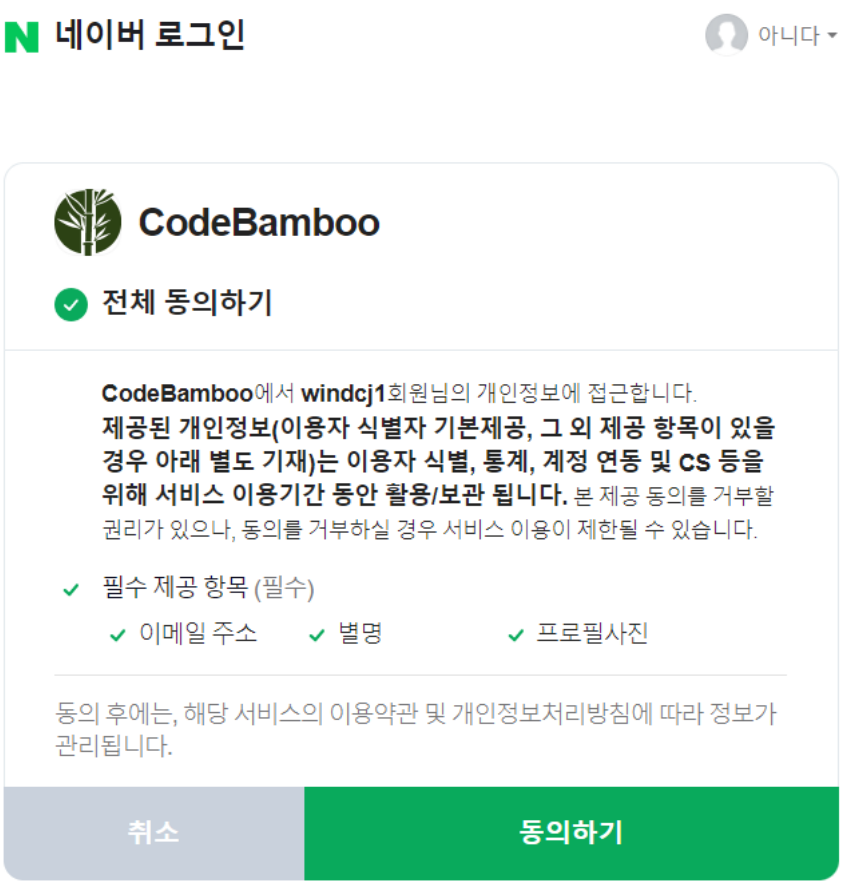



kakao developers
내 애플리케이션
제품
문서
도구
포럼
windcj11@hanmail.net
KOR ENG
내 애플리케이션 > 제품 설정 > 카카오 로그인
앱 설정
요약 정보
일반
비즈니스
앱 키
플랫폼
팀 관리
제품 설정
카카오 로그인
동의항목
간편가입
카카오톡 채널
개인정보 국외이전
연결 끊기
OpenID Connect 활성화 설정
상태
ON
카카오 로그인의 확장 기능인 OpenID Connect를 활성화합니다.
이 설정을 활성화하면 카카오 로그인 시 사용자 인증 정보가 담긴 ID 토큰을 액세스 토큰과 함께 발급받을 수 있습니다.
Redirect URI
삭제
수정
Redirect URI
http://localhost:3000/oauth/kakao
https://k8a801.p.ssafy.io/oauth/kakao
• 카카오 로그인에서 사용할 OAuth Redirect URI를 설정합니다. (최대 10개)
• REST API로 개발하는 경우 필수로 설정해야 합니다.

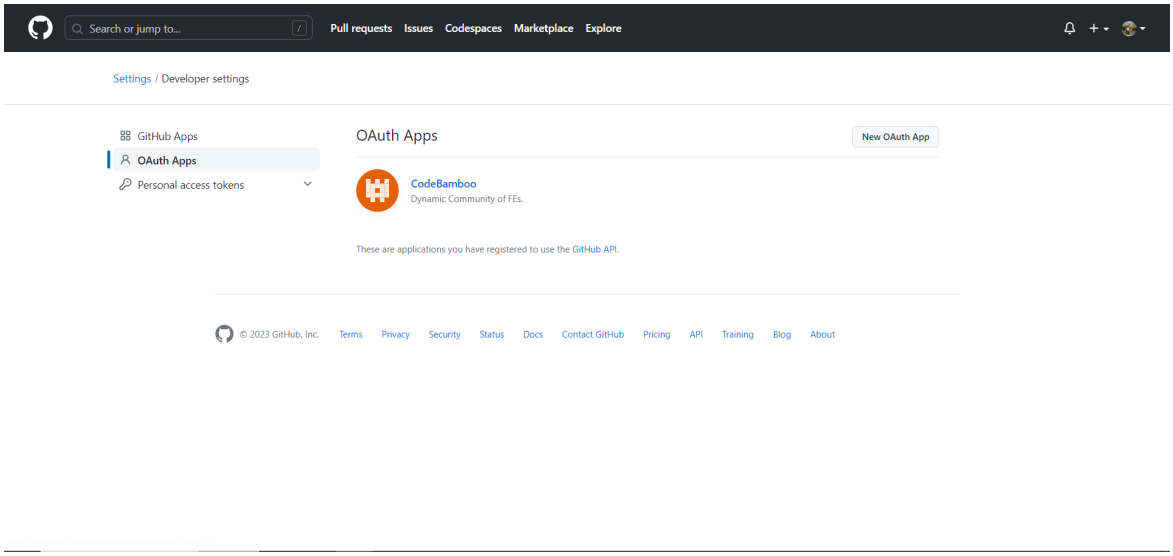

CodeBamboo
코드뱀부
전체 동의하기
전체동의는 선택목적에 대한 동의를 포함하고 있으며, 선택목적에 대한 동의를 거부해도 서비스 이용이 가능합니다.
windcj11@hanmail.net
CodeBamboo 서비스 제공을 위해 회원번호와 함께 개인정보가 제공됩니다. 보다 자세한 개인정보 제공항목은 동의 내용에서 확인하실 수 있습니다. 해당 정보는 동의 철회 또는 서비스 탈퇴 시 지체없이 파기됩니다.
[필수] 카카오 개인정보 제3자 제공 동의
보기
프로필 사진, 닉네임
[선택] 카카오 개인정보 제3자 제공 동의
보기
카카오계정(이메일)
동의하고 계속하기

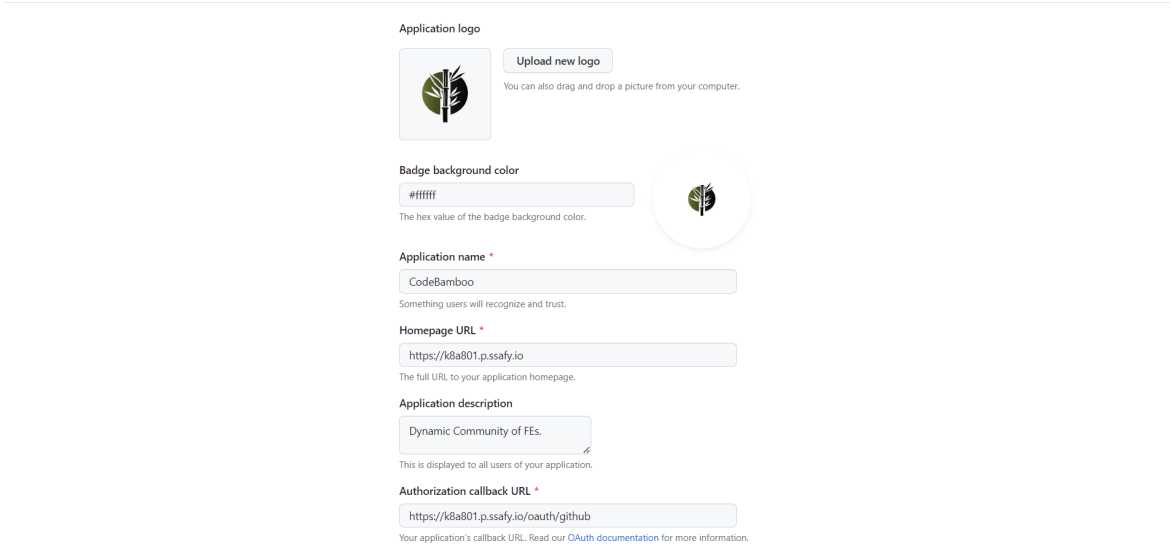
- 네이버 로그인


NAVER Developers
Products
Documents
Application
NAVER D2
Support
Forum
API 상태
Search Here
내 애플리케이션
Mindder
CodeBamboo
애플리케이션 등록
API 제휴 신청
계정 설정
환경 추가
PC 웹
서비스 URL
https://k8a801.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 뱃지가 노출됩니다.
로그인 오픈 API
서비스 환경
네이버 로그인
Callback URL (최대 5개)
http://localhost:3000/oauth/naver
https://k8a801.p.ssafy.io/oauth/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 리디이렉트 될 경우, 이용이 제한될 수 있습니다.

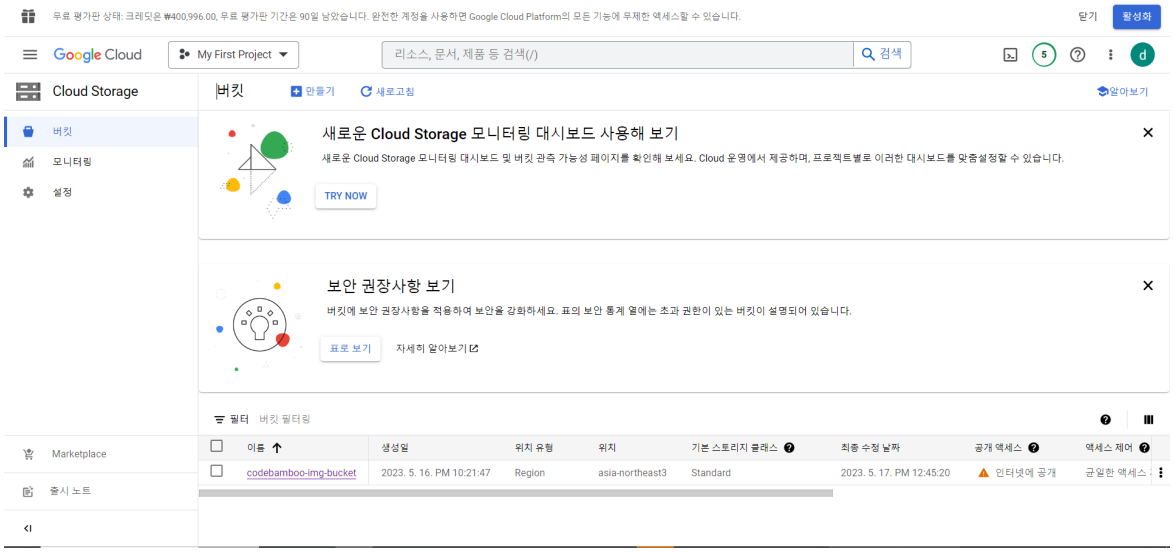

네이버 로그인
아니다
CodeBamboo
전체 동의하기
CodeBamboo에서 windcj1회원님의 개인정보에 접근합니다.
제공된 개인정보(이용자 식별자 기본제공, 그 외 제공 항목이 있을 경우 아래 별도 기재)는 이용자 식별, 통계, 계정 연동 및 cs 등을 위해 서비스 이용기간 동안 활용/보관 됩니다. 본 제공 동의를 거부할 권리가 있으나, 동의를 거부하실 경우 서비스 이용이 제한될 수 있습니다.
필수 제공 항목 (필수)
이메일 주소
별명
프로필사진
동의 후에는, 해당 서비스의 이용약관 및 개인정보처리방침에 따라 정보가 관리됩니다.
취소
동의하기

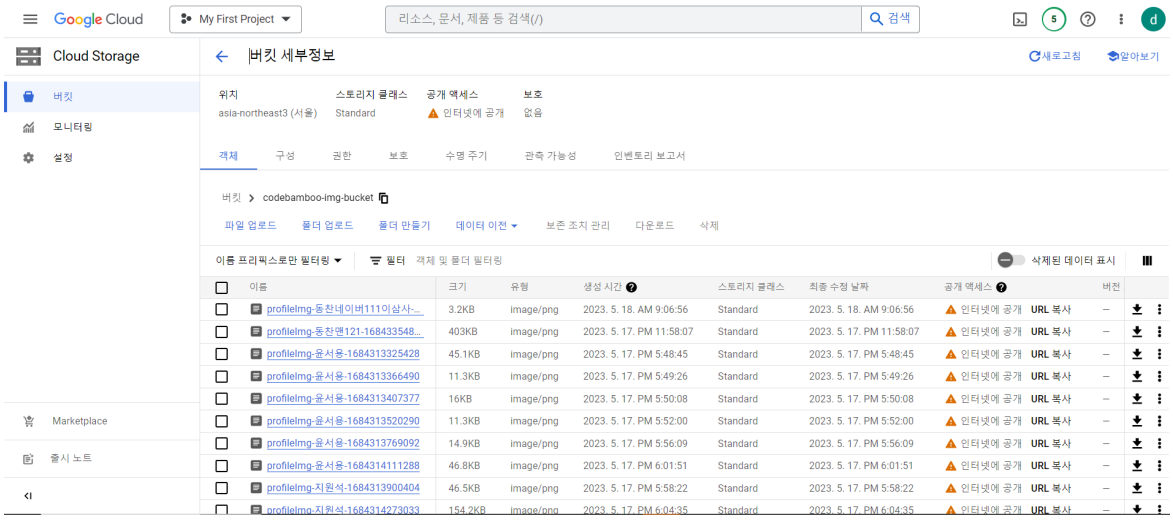
- 깃허브 로그인


Search or jump to...
Pull requests
Issues
Codespaces
Marketplace
Explore
Settings / Developer settings
GitHub Apps
OAuth Apps
Personal access tokens
OAuth Apps
New OAuth App
CodeBamboo
Dynamic Community of FEs.
These are applications you have registered to use the GitHub API.
© 2023 GitHub, Inc.
Terms
Privacy
Security
Status
Docs
Contact GitHub
Pricing
API
Training
Blog
About


Application logo
Upload new logo
You can also drag and drop a picture from your computer.
Badge background color
#ffffff
The hex value of the badge background color.
Application name *
CodeBamboo
Something users will recognize and trust.
Homepage URL *
https://k8a801.p.ssafy.io
The full URL to your application homepage.
Application description
Dynamic Community of FEs.
This is displayed to all users of your application.
Authorization callback URL *
https://k8a801.p.ssafy.io/oauth/github
Your application's callback URL. Read our OAuth documentation for more information.

- gcp 이미지 업로드


무료 평가판 상태: 크레딧은 ₩400,996.00, 무료 평가판 기간은 90일 남았습니다. 완전한 계정을 사용하면 Google Cloud Platform의 모든 기능에 무제한 액세스할 수 있습니다.
닫기
활성화
Google Cloud
My First Project
리소스, 문서, 제품 등 검색(/)
검색
Cloud Storage
버킷
만들기
새로고침
알아보기
버킷
모니터링
설정
새로운 Cloud Storage 모니터링 대시보드 사용해 보기
새로운 Cloud Storage 모니터링 대시보드 및 버킷 관측 가능성 페이지를 확인해 보세요. Cloud 운영에서 제공하며, 프로젝트별로 이러한 대시보드를 맞춤설정할 수 있습니다.
TRY NOW
보안 권장사항 보기
버킷에 보안 권장사항을 적용하여 보안을 강화하세요. 표의 보안 통계 열에는 초과 권한이 있는 버킷이 설명되어 있습니다.
표로 보기
자세히 알아보기
필터
버킷 필터링
이름
생성일
위치 유형
위치
기본 스토리지 클래스
최종 수정 날짜
공개 액세스
액세스 제어
codebamboo-img-bucket
2023. 5. 16. PM 10:21:47
Region
asia-northeast3
Standard
2023. 5. 17. PM 12:45:20
인터넷에 공개
균일한 액세스
Marketplace
출시 노트



시연 시나리오

1. 메인화면 좌측 로그인 버튼으로 소셜 로그인 진행
2. 메인화면에서 다양한 프런트엔드 코드 확인 가능
3. 다른 사람의 질문(토픽)에 답변 달기 - 계층형 누적답변
4. 내가 질문하기-답이 해결되지 않은 경우 손들기 기능을 활용
5. 답변 즐겨찾기, 추천
6. 검색기능으로 내가 원하는 글 찾기
7. 마이페이지에서 내 정보 조회 및 수정, 팔로워 및 팔로잉 확인